HZ BOOKS
华章IT

数据分析与决策技术丛书

Data Analysis and Data Operation with Python

Python数据分析与数据化运营

宋天龙◎著

国内一线数据分析专家撰写，10余位互联网大咖和多个数据科学相关组织联合推荐

50个数据工作流知识点，14个数据分析和挖掘主题，8个综合性运营案例，涵盖会员、商品、流量、内容4大数据化运营主题，360度把脉运营问题并贴合数据场景落地

机械工业出版社
China Machine Press

数据分析与决策技术丛书

Python数据分析与数据化运营

宋天龙　著

ISBN：978-7-111-58460-5

本书纸版由机械工业出版社于2017年出版，电子版由华章分社（北京华章图文信息有限公司，北京奥维博世图书发行有限公司）全球范围内制作与发行。

客服热线：+ 86-10-68995265

客服信箱：service@bbbvip.com

官方网址：www.hzmedia.com.cn

新浪微博 @华章数媒

微信公众号 华章电子书（微信号：hzebook）

目录

赞誉

本书围绕数据化运营，从数据获取、处理、分析、技巧和案例进行有节奏、步步深入的讲解，辅以Python工具手把手的教你如何进行操作、实现，是实操性非常强的一本图书。

——郑来轶　中国统计网创始人

近几年来，Python以其简洁、易读、可扩展的功能特性，逐渐成为最受欢迎的程序开发语言之一。同时，Python和自身丰富的扩展库可以帮助程序开发者完成各种高级任务。特别是在大数据时代，在对外部数据获取、分析和挖掘、数据化运营的要求越来越普遍的情况下，Python可谓是一把利器，可以很好地协助开发者实现以上需求。本书独辟蹊径，弱化以往的纯工具和代码讲解，提供大量的实际业务场景。这本书不仅告诉读者怎么用Python，更强调什么时候用、在哪里用。理论和实践相结合，学以致用，很值得大家学习。

——梁勇　天善智能联合创始人

2016年后，大数据进入数据应用时代，本书列举大量数据分析应用案例且可落地，很难得，值得借鉴与分享。

——赵良　中国统计网联合创始人

本书从运营数据来源、经验总结、走过的哪些坑，延伸到会员运营、商品运营、流量运营、数据化运营的终极诀窍，全面系统地讲解了数据化运营的方法论，是不可多得的运营参考资料！

——王兴宝　数盟社区创始人

Python作为数据科学家的首选程序语言，在数据分析市场有很大的市场份额。该书从数据运营出发，横向打通数据价值周期，包括数据整理、清洗、建模、分析、反馈等，纵向涵盖商品、会员、流量、内容等方面，在模型和实际应用场景的映射层面打开一扇独特的思维窗口，在众多Python读物中脱颖而出，具有非常强的实战指导意义。

——高峡　重庆大数据应用联盟创始人

天龙兄是一线实战派，在电商数据运营方面有很深的造诣，在大数据的应用和实践方面，有这本书作为指导，非常值得期待。其中的会员运营、商品运营、内容运营都是非常一线的作战方案，推荐给大家，感谢天龙兄有此大作。

——王子枫（庖丁的刀）　深圳市上中下网络供应链服务有限公司创始人

随着互联网技术发展的日新月异，信息海量增加，企业要在竞争激烈的商业社会中脱颖而出，需要存储、抓取、分析各种运营数据，Python作为数据分析和挖掘最知名的语言，有极大的优势。宋天龙作为国内拥有多年经验的商业数据分析专家，对Python进行了深入的讲解和剖析，不仅限于工具层面，更在于工具逻辑。希望这本书能够为大家在企业数据分析和运营中拨开数据的迷雾，事半功倍！

——胡力　Netconcepts华南分公司总经理

数据驱动的精细化运营作为企业决策的基石、其精髓在于对数据化运营的深刻理解以及科学应用。本书以数据化运营的基础语言和常用工具入手，对数据的获取、预处理、分析和挖掘给出了完整的处理方案，并且系统地从会员、商品、流量和内容四个维度对数据化运营的具体操作给予了实战方法和实例讲解。本书结尾更是点睛详述了数据报告生成、数据运营扩展和价值提升的奥秘和技巧，最终帮助数据运营者打通从数据获得到决策形成的完整通路。无论是系统学习还是作为工具书随时查阅，本书轻松严谨的笔触能够在潜移默化间帮助数据管理者提升对于数据的敏感性和逻辑处理能力，从而真正实现对数据化运营的从容劝驾。

——凌晨　飞鹤集团电商事业部VP

数据分析是一门既需要懂数据分析原理，又需要结合实战操作的一门科学。如何更好结合实战学习数据分析，也一直是困扰网站分析入门者和从业者的问题。这本书较系统地对数据分析的方法论、常用工具和业务洞察做了阐述，是数据分析从业者值得参考的一本实战经验的书籍。

——李俊　艺龙网技术部数据平台总监

随着人口红利的逐渐消解，中国的互联网市场已经从粗犷做用户的1.0时代走向精细做运营的2.0时代。当各行各业都已经互联网化的时候，运营的精细化程度成为了产品间的护城河。很高兴可以看到，近些年网络上关于用户激励成长体系、关联销售、RFM用户分层的文章越来越多，但大多停留在理论层面，关于最核心的数值设计部分却鲜有提及。而宋天龙的这本书在大量数据化运营案例的基础上，结合Python语言对具体的数据分析过程进行了详细的讲解，让读者不光可以“知其然”，还可以“知其所以然”，读完有种酣畅淋漓，跃跃欲试的感觉。掩卷遐思，相见恨晚。

——柳晨龙　百度阅读数据运营经理/数据挖掘_PHP博主/资深分析师

前言

为什么要写这本书

随着商业竞争形式的日益严峻，企业需要不断寻找提高利润率、降低成本、提高产出价值的有效方法，而数据化运营恰好是满足企业这一需求的关键武器。数据化运营包含了运营和数据两种要素，前者需要较多的业务经验，而后者对数据分析提出了更高的要求。只有把二者结合起来，在技能、经验和技术的支持下，数据化运营才能在企业内部真正落地、生根、发芽。

对数据化运营而言，各企业普遍关注的结构化数据分析、挖掘的场景非常丰富，例如销售预测、会员生命周期维护、商品结构分析等，这些普遍的共同认知为本书提供了接地气的基础；但除了这些“传统内容”外，还有很多非结构化的数据主题，它们在数据化运营过程中的作用越来越重要，例如主题挖掘、图片分析、文本挖掘、图像识别、语音识别等，这些内容拓展了数据化运营发挥价值的场景基础。

Python作为数据工作领域的关键武器之一，具有开源、多场景应用、快速上手、完善的生态和服务体系等特征，使其在数据分析与数据化运营中的任何场景都能游刃有余；即使是在为数不多的短板上，Python仍然可以基于其“胶水”的特征，引入对应的第三方工具、库、程序等来实现全场景、全应用的覆盖。在海量数据背景下，Python对超大数据规模的支持性能、数据分析处理能力、建模的专业程度及开发便捷性的综合能力方面要远远高于其他工具。因此，Python几乎是数据化运营工作的不二之选。

纵观整个国内市场，有关Python的书籍不少，但普遍的思路都是基于工具层面的介绍，而且侧重于工具本身的方法、参数、调用、实例，与真正实践结合的较少；有关数据化运营的书籍，目前市场上还为数不多，现有的数据化运营方面的书籍大多是基于Excel等工具的入门级别的分析类书籍。本书结合了Python和数据化运营两个方面，在结合了数据分析工作流程和数据化运营主题的基础上，通过指标、模型、方法、案例配合工具的形式，详细介绍了如何使用Python来支持数据化运营，尤其是传统工具无法满足的应用场景。

我希望能尽自己的微薄之力，将过往所学、所感、所知提炼出来供更多人了解。如果读者能从本书中感悟一二，我将倍感欣慰；如果读者能将其用于工作实践，这将是本书以及数据工作之福！

读者对象

本书定位于提供数据与运营结合的相关知识，虽然基础工具是Python，但本书并没有就Python基础规则和语法做详细介绍，因此要求读者具有一定的Python基础。相信我，只要你认真看Python教学视频（网络上很多），只需大概2个小时就能具备这种基础。

本书对读者的知识背景没有特定要求，书中的内容都尽量言简意赅、深入浅出。本书适合以下几类读者阅读：

·企业运营人员。本书的核心命题就是运营，其中涉及会员运营、商品运营、流量运营和内容运营四大主题，无论运营人员希望获得运营知识，还是希望获得数据分析和挖掘方法，都可以从书中获益。

·数据分析师。毫无疑问，数据分析师是本书的核心受众群体之一，本书中介绍的数据抽取、预处理和分析挖掘经验一定能为数据分析师带来很多“不一样”的收获，每个运营主题下的小技巧、模型和案例更能激发数据分析师的灵感——原来数据工作还能这样做。

·Python工程师。坦白讲，本书不是一本专门介绍Python语法、规则的书籍。但Python作为一种“万能”工具，在数据分析和挖掘领域具有举足轻重的地位，任何一个Python工程师如果工作领域中涉及数据（或大数据），那么本书的价值会成倍增长。本书中对Python数据处理、计算和挖掘库的应用介绍，以及对有关工具库的用法、注意点和小知识的介绍一定会使Python工程师的工作和认知更上一层楼。

·数据挖掘工程师。数据分析与挖掘在实际运营中是不分家的，本书没有冠以“挖掘”之名但并不意味着没有挖掘（或机器学习）算法。本书第4章基本都是围绕常用算法展开的，其中各个算法类的“大坑”都是笔者多年经验的总结；在运营主题中提到的基于超参数优化的Gradient Boosting的预测，基于LogisticRegression、RandomForest、Bagging概率投票组合模型的异常检测，基于自动K值的KMeans聚类分析，基于潜在狄利克雷分配（LDA）的内容主题挖掘，基于多项式贝叶斯的增量学习

的文本分类等都是与“挖掘算法”相关的应用。算法是数据工作的核心部分，其介绍必不可少。

如何阅读本书

本书内容从逻辑上共分为两大部分，第一部分是有关数据分析类的主题，第二部分是有关数据化运营的主题。

第一部分的内容包括第1～4章和附录，主要介绍了Python和数据化运营的基本知识、数据来源获取、数据预处理，以及数据分析和挖掘的关键经验。其中：

·第2章对传统的结构化和非结构化数据来源及获取方法进行了介绍，包括数据文件、数据库、API、流式数据、外部公开数据等，也提到了如何读取网页、文本、图片、视频、语音等类型的数据。

·第3章总结了常用的11条结构化数据的预处理经验，并介绍了有关网页数据解析、日志解析、图像预处理和自然语言预处理的内容。

·第4章总结了数据分析、挖掘和网站分析方法的8个主题类，各个类别中都以关键经验为基础展开详细介绍。

第二部分的内容包括第5～9章，分别介绍了会员运营、商品运营、流量运营和内容运营四大主题，以及提升数据化运营价值度的方法。在每个数据化运营主题中都包含了基本知识、评估指标、应用场景、数据分析模型、数据分析小技巧、数据分析大实话及2个应用案例。

·基本知识：有关运营主题的基本内涵、价值、用途等方面的介绍。

·评估指标：运营主题的评估指标，按类别拆分和归纳。

·应用场景：总结数据对于运营的价值落地在哪些场景中。

·数据分析模型：“大型”的数据分析方法，包括统计分析、数据挖掘、网站分析、数学模型。

·数据分析小技巧：“小型”的数据分析方法，看起来相对简单但非

常有效。

·数据分析大实话：有关运营或数据分析的潜在规律的解释及介绍。

·应用案例：每个运营主题都包含2个应用案例，基本上每个案例的应用算法和技巧都不相同，目的是呈现不同算法在不同场景下的差异化应用。

除了以上内容外，以下信息是在本书中涉及特定内容的解释和说明：

渐进式的内容：本书的Python代码和实现部分，在不同章节可能会具有不同代码风格的写法，包括定义规则、注释、功能实现等，这是因为笔者试图遵循循序渐进的原则，先介绍功能实现，然后再介绍其他的备选方案，以及规范、原则等来辅助Python的实现。这种做法一方面是希望尽量多地展示解决同一类问题的不同方法，让读者能根据自身实际情况选择最“合适”的用法示例；另一方面，可能有很多读者不具备较强的Python基础知识，因此笔者不希望一上来就让这些读者感觉到要用Python工作会受到各种“条条框框”的限制，从而打击他们使用Python的信心，毕竟，能实现功能需求是第一要素。

内容延伸：本书第1～4章都有内容延伸章节，其内容是有关非结构化主题的读取、分析、处理，由于每个主题展开来写都能成一本书，因此仅在内容延伸中抛砖引玉，有兴趣的读者可以了解和学习。

相关知识点：本书很多章节中都有“相关知识点”部分，其内容是关于特定工具、知识、算法、库等方面的较为详细的介绍，充当了本书的知识堡垒。

本章小结：每章的结尾都有“本章小结”，在小结中包含4部分内容：

·内容小结：有关本章内容的总结

·重点知识：本章需要读者重点掌握的知识和内容

·外部参考：本章提到但是无法详细介绍的内容，都在外部参考中列出，有兴趣的读者可以基于外部参考构建自己的知识图谱。

·应用实践：基于本章内容给出的读者在实践中落地的建议。

提示：对于知识点的重要提示和应用技巧，相对“相关知识点”而言，每条提示信息内容量较少，一般都是经验类的总结。

注意：特定知识需要引起注意的方面，这些注意点是应用过程中需要避免的“大坑”。

特定名词的混用：本书中提到了库和包、模型和算法等词，虽然含义有差异，但本书并没有划清它们的界限，因此在很多时候它们都是等价的。

关于附件的使用方法：除了第9章外，本书的每一章都有对应源数据和完整代码，该内容可在本书附件中找到，附件可以在华章网站http://www.hzbook.com或者笔者网站——数据常青藤http://www.dataivy.cn/book/python_book.zip下载。需要注意的是，为了更好地让读者了解每行代码的含义，笔者在注释信息中都使用了中文标注，每个程序文件的编码格式都是UTF-8。

勘误和支持

由于笔者水平有限，加之撰稿时间也有限，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。读者可通过以下途径联系并反馈建议或意见：

·即时通信：添加个人QQ（517699029）或微信（TonySong2013）反馈问题。

·直接扫描二维码添加个人微信。

·网站讨论区：在笔者网站——数据常青藤的书籍讨论区http://www.dataivy.cn/python-data-analysis-and-data-operations/留言。

·电子邮件：发送email到517699029@qq.com。

致谢

在本书的撰写过程中，得到了多方的指导、帮助和支持。

首先，感谢彭亮先生和史研先生。彭亮先生使我感受到什么是高度和专注，并促使我的数据工作真正意义上步入正途。史研先生对于大数据的广博认知和敏锐洞察力，让我有更多机会深入到不同的数据分支去探索未曾了解的领域。

其次，感谢的是机械工业出版社华章公司的总编辑杨福川老师，杨老师在我出版了两本书之后鼓励我继续撰写本书，并为此书的撰写提供了方向和思路指导。另外，感谢全程参与审核、校验等工作的孙海亮老师以及其他背后默默支持的出版工作者，他们的辛勤付出保证了本书的顺利面世。

再次，感谢在各个数据项目和工作中提供支持的领导、朋友、伙

伴，尤其是田学锋，他是我的良师益友，他有着非比寻常的视野、胸怀和独到的见解，在我的人生道路上给与了我非常多的指导和启迪；其他还有很多一起工作的小伙伴（排名不分先后）：庞程程、徐子东、赵光娟、王成、吕兆星、郑传峰、杨晓鹏、陈骏、江涛、曹佳佳、麻建昕、史晓春、杨勇等。

最后，感谢我的父母、家人和朋友，尤其是我的夫人姜丽女士，是她在我写书的这段期间里把家里的一切料理得井井有条，使得我有精力完成本书的全部撰写工作。

谨以此书献给热爱数据工作并为之奋斗的朋友们，愿大家身体健康、生活美满、事业有成！

宋天龙（Tony Song）

第1章　Python和数据化运营

企业的数据化运营是提高利润、降低成本、优化运营效率、最大化企业财务回报的必要课题。Python作为数据科学界的关键工具之一，几乎可以应用于所有数据化运营分析和实践的场景。本章将首先介绍Python与数据化运营的基本内容，然后围绕数据化运营分析所需的构建Python相关工具进行介绍，最后通过一个入门级的案例介绍如何将Python用于数据化运营。

1.1 用Python做数据化运营

Python是什么？数据化运营又是什么？为什么要将Python用于数据化运营？本节先来回答这几个问题。

1.1.1 Python是什么

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Guido van Rossum于1989年发明，第一个公开发行版发行于1991年。Python开发的初衷其实是一个开发程序语言，而非相对数据工作和科学计算的数据处理或建模程序。

为什么我们要选择Python而非其他语言（例如R）进行数据处理、分析和挖掘？这是因为Python先天和后天具有的一些特殊条件和能力使其成为目前企业（尤其是大数据领域）做数据化运营最为合适的工具。

·开源/免费：使用Python（及其第三方库）无须购买产品、授权或license费用，无论对于个人还是对于企业都是如此。

·可移植性：Python程序可以跨Windows、Linux、Mac等多平台运行，这点决定了它的移植性非常强，一次开发，多平台应用。

·丰富的结构化和非结构化数据工作库和工具：Python除了自带数学计算库外，还包括丰富的第三方库和工具，例如用于连接Oracle、MySQL、SQLite等数据库的连接库，数据科学计算库Numpy、Scipy、Pandas，文本处理库NLTK，机器学习库Scikit-Learn、Theano，图形视频分析处理和挖掘库PIL和Opencv，以及开源计算框架TensorFlow等。

·强大的数据获取和集成能力：Python除了可以支持多种类型的文件（图像、文本、日志、语音、视频等）和数据库集成外，还能通过API、网络抓取等方式获取外部数据，内、外部数据源整合、多源数据集成、异构数据并存、多类型数据交错正是当前企业数据运营的基本形态。

·海量数据的计算能力和效率：当面对超过GB甚至TB规模的海量数据时，传统数据工具通常无法支撑，更不要提计算效率了。Python对于这个规模数据的计算能力和工作效率要远超过其他数据工作语言。

·与其他语言的集成：Python具备“胶水”能力，能与Java、C、C++、MATLAB、R等语言集成使用，这意味着你既可以把其他语言写成的脚本嵌入Python，也可以把Python脚本嵌入其他语言。

·强大的学习交流和培训资源：Python已经成为世界上最主流的编程语言和数据处理工作的核心工具之一，有非常多的社区、博客、论坛、培训机构、教育机构提供交流和学习的机会。

·开发效率高：Python语言简洁、规范，这使得在使用Python开发程序时用时更少。这对于以效率优先的程序工作或验证性项目来说非常关键，效率决定商机。

·简单易学：Python语法简单，即使是没有任何代码基础的人也能在几个小时内掌握基本的Python编程技巧，这对于初学者而言至关重要，因为这说明对他们来说编程式数据分析不再遥不可及，他们能像使用Excel一样使用Python。

总而言之，在具备一定Python经验和技巧的情况下，几乎没有Python无法胜任的工作场景！如果有，那么用Python调用其他语言或用其他语言调用Python后，就会胜任。

1.1.2 数据化运营是什么

数据化运营是指通过数据化的工具、技术和方法，对运营过程中的各个环节进行科学分析、引导和应用，从而达到优化运营效果和效率、降低成本、提高效益的目的。

运营是一个范围“弹性”非常大的概念，最大可以延伸到所有公司的事务管理，最小可能只包括网站运营管理工作。本书中若无特殊说明，运营的范围包括会员运营、商品运营、流量运营和内容运营四方面内容。

1.数据化运营的重要意义

数据化运营的核心是运营，所有数据工作都是围绕运营工作链条展开的，逐步强化数据对于运营工作的驱动作用。数据化运营的价值体现在对运营的辅助、提升和优化上，甚至某些运营工作已经逐步数字化、自动化、智能化。

具体来说，数据化运营的意义如下：

1）提高运营决策效率。在信息瞬息万变的时代，抓住转瞬即逝的机会对企业而言至关重要。决策效率越高意味着可以在更短的时间内做出决策，从而跟上甚至领先竞争对手。数据化运营可使辅助决策更便捷，使数据智能引发主动决策思考，从而提前预判决策时机并提高决策效率。

2）提高运营决策正确性。智能化的数据工作方式，可以基于数据科学方法进行数据演练并得出可量化的预期结果，再配合决策层的丰富经验，会提高运营决策的正确性。

3）优化运营执行过程。数据化运营可以通过标准口径的数据、信息和结论，为运营部门提供标准统一、目标明确的KPI管理，结合数据化的工作方法和思路，优化运营过程中的执行环节，从而降低沟通成本、提高工作效率、提升执行效果。

4）提升投资回报。数据化运营过程中，通过对持续的正确工作目

标的树立、最大化工作效率的提升、最优化工作方法的执行能有效降低企业冗余支出，提升单位成本的投资回报。

2.数据化运营的2种方式

从数据发挥作用的角度来看，数据化运营分为辅助决策式数据化运营和数据驱动式数据化运营。

（1）辅助决策式数据化运营

辅助决策式数据化运营是运营的决策支持，它是以决策主题为中心的，借助计算机相关技术辅助决策者通过数据、模型、知识等进行业务决策，起到帮助、协助和辅助决策者的目的。例如，通过为决策者提供商品促销销量信息，来为企业促销活动提供有关订货、销售等方面的支持。

（2）数据驱动式数据化运营

数据驱动式数据化运营是指整个运营运作流程以最大化结果为目标，以关键数据为触发和优化方式，将运营业务的工作流程、逻辑、技巧封装为特定应用，借助计算机技术并结合企业内部流程和机制形成一体化的数据化工作流程。例如，个性化推荐就是一种数据驱动数据化运营方式。

辅助决策式数据化运营和数据驱动式数据化运营是两个层次的数据应用，数据驱动相对于辅助决策的实现难度更高、数据价值体现更大。

·辅助决策式数据化运营为业务决策方服务，整个过程都由运营人员掌控，数据是辅助角色。

·数据驱动式数据化运营的过程由数据掌控，数据是主体，实现该过程需要IT、自动化系统、算法等支持，数据驱动具有自主导向性、自我驱动性和效果导向性。

由于数据和流程本身会存在缺陷，同时运营业务通常都有强制性规则的需求，因此即使在数据驱动数据化运营过程中也会加入人

工干预因素。但即使如此，数据作为数据驱动的核心是不变的，也就是说，数据是决策主体本身。

3.数据化运营的工作流程

在上一节我们介绍了数据化运营的两种方式——辅助决策式数据化运营和数据驱动式数据化运营，其中数据驱动式数据化运营取决于应用场景，不同的场景其工作流程不同。有关该部分内容在本书后面章节会具体讲到。本节重点介绍数据驱动式数据化运营的工作流程。

数据驱动式数据化运营工作包含数据和运营两个主体，在实际工作过程中需要二者协同；在某些大型工作项目上，还有可能涉及跟IT、信息中心等部门的联动。工作流程分为三个阶段，如图1-1所示。

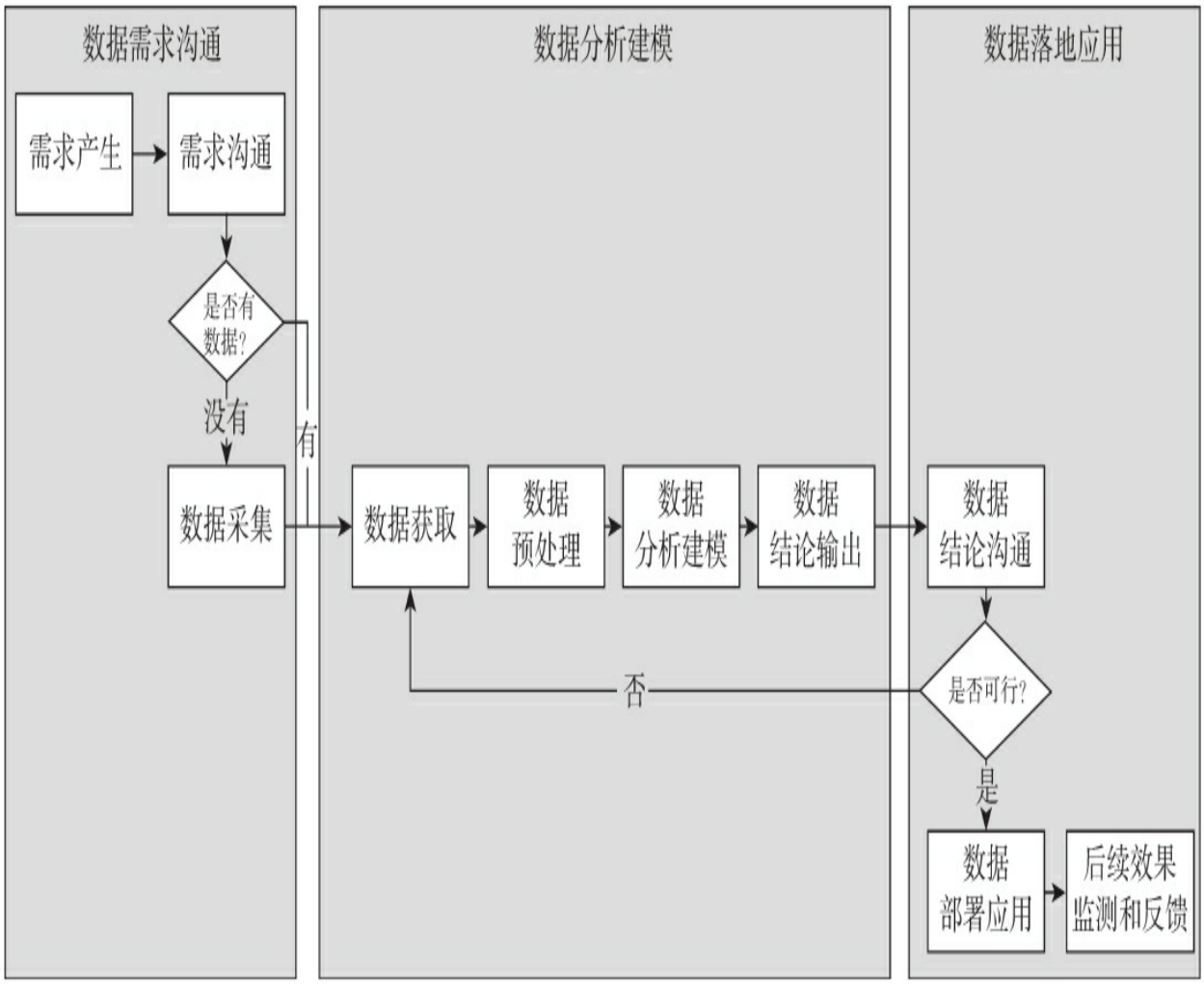


图1-1　数据驱动式数据化运营工作流程

（1）第一阶段：数据需求沟通。

该阶段主要包括需求产生和需求沟通两个步骤。

1）需求产生：由运营部门产生的某些数据化运营需求，例如预测商品销量、找到异常订单、确定营销目标人群名单等。

2）需求沟通：针对运营部门提出的需求进行面对面沟通和交流，沟通主要包含三方面：一是业务需求沟通，包括需求产生的背景、要解决的问题、预期达到的效果等；二是数据现状沟通，包括数据存储环境、主要字段、数据字典、数据量、更新频率、数据周期等，如果没有数据则需要制定数据采集规则并开始采集数据，该过程中可能需要IT部门的协助；三是数据与分析的关联性沟通，根据与运营人员的沟通，了解业务背景下哪些是常见的、带有业务背景的数据、不同场景会导致数据如何变化、分析中会涉及哪些关键字段或场景数据等，业务人员丰富的经验会帮助数据工作者少走很多弯路。

（2）第二阶段：数据分析建模。

从这一阶段开始进入正式的数据工作流程，包括数据获取、数据预处理、数据分析建模和数据结论输出四个步骤。

1）获取数据：数据化运营分析所需的数据需要经过特定授权从数据库或文件中得到。

2）数据预处理：在该过程中对数据进行质量检验、样本均衡、分类汇总、合并数据集、删除重复项、分区、排序、离散化、标准化、过滤变量、转置、查找转换、脱敏、转换、抽样、异常值和缺失值处理等。

3）数据分析建模：运用多种数据分析和挖掘方法，对数据进行分析建模。方法包括统计分析、OLAP分析、回归、聚类、分类、关联、异常检测、时间序列、协同过滤、主题模型、路径分析、漏斗分析等。

4）数据结论输出：数据结论的输出有多种方式，常见的方式是数

据分析或挖掘建模报告，另外还包括Excel统计结果、数据API输出、数据结果返回数据库、数据结果直接集成到应用程序中进行自动化运营（例如短信营销）。

（3）第三阶段：数据落地应用。

该阶段是数据化运营的落地的关键阶段，前期所有的准备和处理工作都通过该阶段：生价值。该阶段包括数据结论沟通、数据部署应用和后续效果监测和反馈三个步骤。

1）数据结论沟通：对于输出的形式为报告、Excel统计结果等方式的内容，通常都需要与运营对象进行深入沟通，主要沟通的内容是将通过数据得到的结论和结果与业务进行沟通，通过沟通来初步验证结论的正确性、可靠性和可行性，并对结果进行修正。如果没有可行性，那么需要返回第二阶段重新开始数据分析建模流程。

2）数据部署应用：经过沟通具有可行性的数据结论，可直接应用到运营执行环节。例如，将预测结果作为下一月份的KPI目标，将选择出来的用户作为重点客户进行二次营销。

3）后续效果监测和反馈：大多数的数据化运营分析都不是“一次性”的，尤其当已经进行部署应用之后，需要对之前的数据结论在实践中的效果做二次验证，若有必要则需要进行结论的再次修正和意见反馈。

很多人认为数据化运营工作应该从数据产生之后开始，这是错误的观念，原因在于数据化运营工作的起始是需求产生，而需求的产生跟数据的产生往往没有必然关系。

1.1.3 Python用于数据化运营

Python用于数据化运营，将充分利用Python的强大功能和效率来满足数据化运营的复杂需求。

·Python可以将数据化运营过程中的来源于企业内外部的海量、多类型、异构、多数据源的数据有效整合到一起，提供丰富的集成、开发、分析、建模和部署应用。

·Python高效的开发效率能帮助数据化运营在最短的时间内进行概念验证，并提供科学的预测结果，为数据化运营的快速和准确提供基础。

·Python可以将数据工作流程和IT工作流程无缝对接，有利于实现数据工作跟运营工作的融合，这也是数据驱动式数据化运营的工作方法，有利于真正实现数字化、智能化的运营工作。

1.2 数据化运营所需的Python相关工具和组件

本书将以Python为主要数据工作工具，本节将重点介绍Python相关工具，包括Python程序、IDE、Python第三方库、数据库和客户端、SSH远程客户端、OCR工具和机器学习框架等。

1.2.1 Python程序

目前，Python仍然是两个系列的版本并存，一个是Python 2（最新版本是2.7.13），另一个是Python 3（最新版本是3.7.1）。这两个版本的语法不完全兼容，因此两个版本的程序调用对方的执行脚本很可能会报错。

从现在来看，Python 2和Python 3都已经非常成熟，因此大多数支持Python 2并且仍然开发和维护的库也开始支持Python 3。

从长远来看，Python 2终究是要被抛弃的，所以迟早是要升级到Python 3的。

但是因为Python 2如此成熟且被广泛应用，所以即使被抛弃也不是短时间内的事情。为了解决Python迁移问题，官方提供了Python 2到Python 3的编码转化工具，具体查阅https://docs.python.org/3/library/2to3.html#to3-reference。

熟悉Python 2的工作者再学习Python 3会非常容易，毕竟二者只是在编码、语法、字符串、字节串、数据类型等方面做了一些变动，而不是语法和程序规范全部变更。有关Python 3的新功能介绍，具体查阅https://docs.python.org/3/whatsnew/3.0.html。

因此，对于如何选择Python版本不应该过于纠结，因为不论选择哪个版本对现在和未来工作的影响都微乎其微。但结合特定场景，笔者还是建议：

·如果想要成熟、可靠且稳定的程序，选择Python 2。

·如果只是想学习或了解一下Python，Python 2和Python 3都可以。

·如果是企业内部应用，具体看企业用的是哪个版本。

·如果你的工作中需要大量的第三方库，那么使用Python 2会让你有更多选择。

·如果你的程序需要在Linux服务器上运行，那么去看看你的Linux服务器自带的Python是什么版本（Linux服务器上自带Python 2的居多）。

·如果上述没有任何一个场景满足你的需求，那么就从Python 2开始吧。由于历史原因，Python 2有更多的应用案例、专业书籍和知识讨论分享社区。

相对于是选Python 2还是选Python 3，选择32位还是64位的版本影响反而更大一些，因为这个涉及与系统平台和第三方应用的兼容和集成。大多数情况下，建议选择32位的Python版本，因为很多软件及Python第三方库默认都是32位版本（尤其是一些比较早期的库）。

当确定了Python的具体版本之后，就可以到Python官网下载对应平台和版本的程序了，Python官网地址为https://www.python.org/downloads/。

本书的案例基于32位Python实现的，版本是Python 2.7.12。

1.2.2 Python IDE

Python自带IDE，可以满足一定的数据开发和测试需求，在交互型的开发和学习上，也有IPython可供选择。但这里建议大家选择另外一个Python IDE工具——PyCharm。

PyCharm带有一整套可以帮助用户在使用Python语言开发时提高效率的工具，比如调试、语法高亮、项目管理、代码跳转、智能提示、自动完成、单元测试、版本控制并可集成IPython、系统终端命令行等。在PyCharm里几乎可以实现所有有关Python的工作。图1-2是PyCharm工具界面截图。

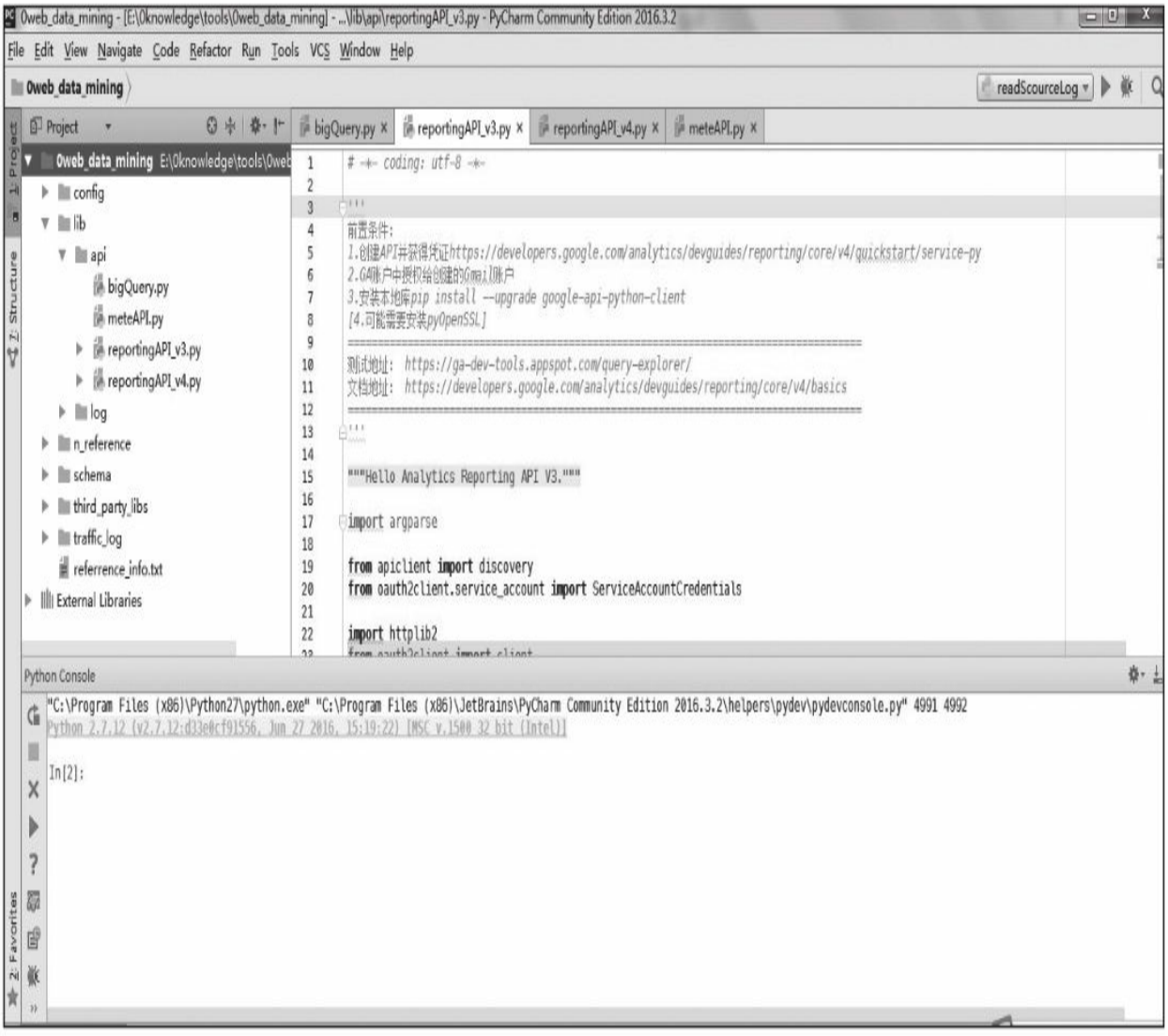


图1-2　PyCharm工具界面截图

有关PyCharm的更多信息可查阅http://www.jetbrains.com/pycharm/download/#section=windows，在该网址下可下载对应操作系统的版本。这里提醒大家，在选择版本时，由于我们以学习为目的，故选择免费的社区版本即可。本书中用到的PyCharm版本是2016.3.2社区版。

1.2.3 Python第三方库

Python第三方库包括交互开发库、科学计算库、机器学习库、自然语言库、数据库连接库、图像处理库、网络爬虫库、图像展示库等。因第三方库较多，故本节介绍的第三方库都是本书中会用到的，其他未用到的库暂时不做介绍，但为大家学习方便，我们会在附录里面提供完整列表。

对于Python第三方库的安装，除了使用setup命令外，大多数都可以通过pip和easy_install命令安装。这里推荐使用pip进行本地或在线安装，因为采用该命令安装2.7.*（准确讲是Python 2≥2.7.9或者Python 3≥3.4）版本时系统会按默认设置自动安装完成。

1.第三方库的安装

（1）使用setup命令从源码安装

每个第三方库都有一个源码文件压缩包，格式为.tar.gz或.zip，例如pandas-0.19.2.tar.gz、numpy-1.12.1.zip。将压缩包从pypi（或其他官方资源）中下载到要安装的服务器或本机并解压（这些步骤都非常简单），然后在系统终端的命令行窗口执行setup命令即可。以“pandas-0.19.2.tar.gz”文件为例说明整个过程：

1）下载源码压缩包文件：不同的系统下载方法不同，最简单的方法是直接打开https://pypi.python.org/pypi/pandas/0.19.2并下载名为“pandas-0.19.2.tar.gz”的文件到本地，然后使用客户端工具复制到服务器。

2）解压文件：打开系统终端的命令行窗口，进入该文件的下载路径（或拷贝路径）并解压，执行如下命令：

```
cd [压缩包文件路径]
tar  -zxvf  pandas-0.19.2.tar.gz [解压后的路径]
```

上述命令执行后，会在制定的[解压后的路径]中解压当前压缩包的内容。

3）执行setup命令：在系统终端的命令行窗口中，进入解压后包含setup.py的路径（通常是[解压后的路径]/pandas-0.19.2），执行如下命令：

```
cd [解压后的包含了setup.py的路径]
python setup.py install
```

上述代码会默认执行完成，如果环境配置正确会有提示安装成功。

离线安装第三方库/包时，不同的库/包可能存在依赖关系，如果在安装之前没有安装和配置好相应的包，那么可能报错。例如安装statsmodels 0.8时，依赖Python≥2.6、Numpy≥1.6、Scipy≥0.11、Pandas≥0.12、Patsy≥0.2.1等，因此大多数情况下，不建议手动离线安装。

（2）使用pip命令从whl文件安装

使用pip安装Python第三方库更加简单，只需先将符合要安装库的系统环境的whl文件下载到服务器或本地，然后在系统终端的命令行窗口输入命令pip install[PackageName.whl]即可。

仍然以上述Pandas安装为例说明整个过程。

1）下载whl文件：不同的系统所需要的whl文件不同，这需根据实际系统版本和Python程序版本而定。以笔者的Windows环境下32位的Python 2版本为例，笔者在https://pypi.python.org/pypi/pandas/0.19.2处下载名为“pandas-0.19.2-cp27-cp27m-win32.whl”的文件到本地。

2）安装whl文件：在系统终端的命令行窗口中进入下载路径（笔者路径为桌面），执行如下命令：

```
cd C:\Users\Administrator\Desktop
pip install pandas-0.19.2-cp27-cp27m-win32.whl
```

命令执行完成之后，也会提示安装成功。

使用pip命令安装包时，需要在系统终端的命令行窗口而非Python或IPython工作界面中执行相应命令。进入Windows终端的命令行窗口的方法是使组合件Win+R，然后在弹出的窗口中输入“cmd”，最后点击“确定”按钮。

3）使用pip进行在线安装：大多数情况下，都建议采用pip进行在线安装，通过pip在线安装可以解决不同包之间的依赖关系（自动下载依赖包）。

在线安装的方法非常简单，只需在系统终端打开命令行窗口，然后输入如下命令：

```
pip install [PackageName]
```

以上述Pandas安装为例，在联网的前提下，只需直接在系统终端命令行输入“pip install pandas”即可完成安装。

pip本身是一个非常强大的第三方库/包管理工具，包括下载、安装、升级、卸载、搜索、查看过期和版本等功能。有关pip的更多信息，可查阅https://pip.pypa.io/en/stable/。

考虑到数据化运营所需的Python相关工具和组件的URL比较多，笔者会在附件的lib文件夹中将所有URL整理为一个名为“lib_url.txt”的文件，同时本书所用到Windows下32位Python 2的第三方安装库（whl文件），也会在该文件夹下，读者有需要可以使用pip命令安装。

2.第三方库简介

（1）交互开发库——IPython

IPython是一个基于Python的交互式shell，比默认的Python shell好用得多，支持变量自动补全、自动缩进、交互式帮助、魔法命令、系统命令等，内置了许多很有用的功能和函数。在本书所说Python第三方库中，若无特殊说明，默认使用IPython作为交互和测试工具。

IPython的安装可直接在系统终端的命令行窗口使用pip install ipython完成。安装成功之后，进入系统终端命令行窗口，输入ipython，回车后进入交互开发界面，如图1-3所示。

图1-3　IPython交互开发界面

本书所用IPython版本是5.3.0。有关IPython的安装和更多信息，可查阅http://ipython.org/。

（2）科学计算库

Numpy

Numpy（Numeric　Python）是Python科学计算的基础工具包，它提供的功能包括：

·快速高效的多维数组ndarray，大多数Python的多维数据组都是基于Numpy进行处理的。

·基于数组整体或元素级别进行科学计算的能力，需要迭代循环。

·比较成熟的（广播）函数库。

·用于整合C、C++和Fortran代码到Python的工具包。

·实用的线性代数、傅里叶变换和随机数生成函数。

·Numpy和稀疏矩阵运算包Scipy配合使用更加方便。

·多种库和算法间进行数据交互的“数据容器”，由低级语言（例如C）编写的库可直接读取Numpy的数据而不必经过转换。

默认情况下，我们可以使用pip install numpy命令对Numpy进行安装，但考虑到我们即将要使用的Scipy 0.19.0中需要依赖于Numpy+mkl，因此我们选择一次性将这两个包一起安装，方法如下：

第一步 从http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy中下载numpy-1.11.3+mkl-cp27-cp27m-win32.whl。

第二步 打开命令行窗口，使用cd[路径]命令进入上述whl文件的下载路径。笔者的下载路径为Windows桌面，路径为C:\Users\Administrator\Desktop。

```
C:\Users\Administrator>cd C:\Users\Administrator\Desktop
```

第三步 使用pip install[本地PackageName].whl命令安装本地下载的包。

```
C:\Users\Administrator\Desktop>pip install numpy-1.11.3+mkl-cp27-cp27m-win32.whl
```

安装成功之后，在IPython中输入import numpy时，若不报错则说明该库已经成功安装并导入。本书中用到的Numpy版本是1.11.3。

查看Numpy（以及其他库）的版本常用的有两种方式：

1）在Python或IPython中导入库后通过__version__属性查看：

```
In [1]: import numpy
In [2]: print (numpy.__version__)
1.12.1
```

2）在命令行窗口（非Python或IPython工作窗口）输入pip　list，系统会返回所有安装的第三方库以及版本列表信息，从中找到Numpy即可：

```
C:\Users\Administrator>pip list
DEPRECATION: The default format will switch to columns in the future. You ca
e       --format=(legacy|columns)       (or       define       a       format=
(legacy|columns) in your pip.con
f under the [list] section) to disable this warning.
appdirs (1.4.3)
asn1crypto (0.22.0)
backports.shutil-get-terminal-size (1.0.0)
beautifulsoup4 (4.5.3)
cffi (1.9.1)
matplotlib (1.5.3)
mysql-connector-python (2.1.5)
nose (1.3.7)
numpy (1.11.3)
pandas (0.19.2)
```

有关Numpy更多信息，请查阅http://www.numpy.org/。

Scipy

Scipy（Scientific Computing Tools for Python）是一组专门解决科学和工程计算不同场景的主题工具包，主要功能包括：

·数值积分和微分方程求解器。

·扩展了有numpy.linal的线性代数历程和矩阵分解功能。

·函数优化器（最小化器）以及根查找方法。

·信号处理工具。

·系数矩阵和系数线性系统求解器。

在其他环境下，安装Scipy时直接使用pip　install　scipy即可，但在Windows　32位Python环境下该命令会报错，原因是在pypi.python.org库

（也就是pip引用的服务器资源库）中找不到32位下的与Python 2对应的Scipy安装包。除了可以源码安装Scipy外，还可通过如下方法进行安装。

第一步　从http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy中下载scipy-0.19.0-cp27-cp27m-win32.whl。

第二步　打开命令行窗口，使用cd[路径]命令进入上述whl文件的下载路径。笔者的下载路径为Windows桌面，路径为C:\Users\Administrator\Desktop。

```
C:\Users\Administrator>cd C:\Users\Administrator\Desktop
```

第三步　使用pip install[本地PackageName].whl命令安装本地下载的包。

```
C:\Users\Administrator\Desktop>pip install scipy-0.19.0-cp27-cp27m-win32.whl
```

安装成功之后，在IPython中输入import scipy时，若不报错则说明该库已经成功安装并导入。

本书中用到的Scipy版本是0.19.0。有关Scipy的更多信息，请查阅https://www.scipy.org/install.html。

Pandas

Pandas（Python Data Analysis Library）是一个用于Python数据分析的库，它的主要作用是进行数据分析。Pandas提供用于进行结构化数据分析的二维表格型数据结构DataFrame，类似于R中的数据框，能提供类似数据库中的切片、切块、聚合、选择子集等精细化操作，为数据分析提供了便捷。另外，Pandas还提供了时间序列功能，用于金融行业的数据分析。

Pandas的安装直接使用pip install pandas命令即可。安装成功之后，在IPython中输入import pandas时，若不报错则说明该库已经成功安装并

导入。

本书中用到的Pandas版本是0.19.2。有关Pandas的更多信息，具体查阅http://pandas.pydata.org/。

Statsmodels

Statsmodels是Python统计建模和计量经济学的工具包，包括一些描述性统计、统计模型估计和统计测试，集成了多种线性回归模型、广义线性回归模型、离散数据分布模型、时间序列分析模型、非参数估计、生存分析、主成分分析、核密度估计以及广泛的统计测试和绘图等功能。

Statsmodels的安装则直接使用pip install statsmodels即可。安装成功之后，在IPython中输入import statsmodels时，若不报错则说明该库已经成功安装并导入。

本书中用到的Statsmodels版本是0.8.0。有关Statsmodels的更多信息，具体查阅http://statsmodels.sourceforge.net/index.html。

（3）机器学习库——scikit-learn

scikit-learn（又称Sklearn）是一个基于Python的机器学习综合库，内置监督式学习和非监督式学习两类机器学习方法，包括各种回归、K近邻、贝叶斯、决策树、混合高斯模型、聚类、分类、流式学习、人工神经网络、集成方法等主流算法，同时支持预置数据集、数据预处理、模型选择和评估等方法，是一个非常完整的机器学习工具库。scikit-learn是Python数据挖掘和机器学习的主要库之一。

scikit-learn缺少了某些常用算法，例如关联规则算法、时间序列算法等。不过结合Pandas和Statsmodels可以实现时间序列算法。关联规则相对简单，pipy上也有很多开源库，当然如果你动手能力强，使用Python自行编写难度也不大。对于这些内容，在后续的模型和案例中，我们会重点介绍。

scikit-learn的安装则直接使用pip install sklearn即可（注意库名称为

sklearn）。安装成功之后，在IPython中输入import sklearn时，若不报错则说明该库已经成功安装并导入。

本书中用到的scikit-learn版本是0.18.1。有关scikit-learn的更多信息，具体查阅http://scikit-learn.org/stable/index.html。

在安装scikit-learn之前一定要确保Numpy（含mkl）、Scipy、Matplotlib按顺序安装，这样才能保证不同库的依赖关系正确建立，否则可能会导致scikit-learn安装或导入报错。

（4）自然语言处理库

结巴分词

由于NLTK本身不支持中文分词，因此在针对中文的处理过程中，我们会引入其他分词工具，例如结巴分词。结巴分词是国内的Python文本处理工具包，分词模式分为三种：精确模式、全模式和搜索引擎模式。结巴分词支持繁体分词、自定义词典等，是非常好的Python中文分词解决方案，可以实现分词、词典管理、关键字抽取、词性标注等。

结巴分词的安装直接使用pip install jieba命令即可。安装成功之后，在IPython中输入import jieba时，若不报错则说明该库已经成功安装并导入。

本书用到的结巴分词的版本是0.38。有关结巴分词的更多信息，具体查阅https://github.com/fxsjy/jieba/。

Gensim

Gensim是一个专业的主题模型（主题模型发掘文字中隐含主题的一种统计建模方法）Python工具包，用来提供可扩展统计语义、分析纯文本语义结构以及检索语义上类似的文档等功能。

Gensim的安装直接使用pip install gensim命令即可。安装成功之后，在IPython中输入import gensim时，若不报错则说明该库已经成功安装并导入。

本书中用到的Gensim版本是1.0.1，具体查阅http://radimrehurek.com/gensim/。

（5）数据库连接库

数据库存储是企业数据存储的基本方式，数据库类型包括MySQL、Oracle、SQL Server、DB2、Sybase等，基于大数据场景下还会包括Hive、HBase、MongoDB、Redis等的数据存储。

MySQL连接库

为了方便读者练习和应用，本书使用MySQL数据库进行数据存储、查询等操作。要使Python连接MySQL，既可以通过MySQL官方连接程序，也可以使用第三方库来实现。

使用MySQL官方程序时，可到https://dev.mysql.com/downloads/connector/python/直接下载对应版本Python的程序，笔者下载的是mysql-connector-python-2.1.5-py2.7-win32.msi。然后直接安装即可，中间没有任何配置。安装成功之后，在IPython中输入import mysql.connector时，若不报错则说明该库已经成功安装并导入。

在选择32位还是64位时，需要注意的是，这里指的是Python的版本，而不是操作系统版本。例如笔者的电脑为64位Windows，但安装的是32位的Python，因此，选择的是mysql-connector-python-2.1.5-py2.7-win32.msi。

使用第三方库MySQL-python时，登录https://pypi.python.org/pypi/MySQL-python/下载对应版本的安装包即可。目前该库已经很长时间没有更新，并且只支持32位的Python 2。安装成功之后，在IPython中输入import MySQLdb（注意大小写）时，若不报错则说明该库已经成功安装并导入。

两种连接方式选择任意一个使用时都没有问题，但考虑到官方连接程序支持的平台和版本更多，因此，这里推荐使用官方连接。

MongoDB连接库

由于不同企业的大数据平台的数据存储不同，并且即使是同一种存储方案，也会由于系统环境和存储组件的版本不同导致适配和连接的差异，本节仅以MongoDB连接包为例进行说明。

MongoDB是由C++语言编写的分布式文件存储的数据库，它是以Key-Value（键值对）形式面向文档存储的非关系型数据库。

Python连接MongoDB可以使用PyMongo（MongoDB官方驱动程序）。通过pip install pymongo对PyMogo进行安装，安装成功之后，在IPython中输入import pymongo时，若不报错则说明该库已经成功安装并导入。

本书中用到的PyMongo版本是3.4.0。有关PyMongo的更多信息，具体查阅http://api.mongodb.com/python/current/tutorial.html。

（6）HTML处理库——Beautiful Soup

网络是企业重要的外部数据来源，因此获取和处理HTML的信息是Python数据接入和处理的重要能力。

Beautiful Soup是网页数据解析和格式化处理工具，它严格意义上来讲不是一个纯抓取类的工具，因为它不具备抓取能力，通常配合Python的urllib、urllib2等库一起使用。

Beautiful Soup的安装直接使用pip install beautifulsoup4（注意后面有个“4”）命令即可。安装成功之后，在IPython中输入import bs4（注意导入的库名跟安装的库名不一致）时，若不报错则说明该库已经成功安装并导入。

本书中用到的Beautiful Soup版本是4.5.3。有关Beautiful Soup的更多信息，具体查阅https://www.crummy.com/software/BeautifulSoup/。

（7）图形展示库——Matplotlib

图形展示是数据可视化的必要内容，在Python中，通常使用

Matplotlib实现图形展示。

Matplotlib是Python的2D绘图库，它以各种硬拷贝格式和跨平台的交互式环境生成出版质量级别的图形，开发者可以仅需要几行代码，便可以生成绘图、直方图、功率谱、条形图、错误图、散点图等。

Matplotlib的安装直接使用pip install matplotlib命令即可。安装成功之后，在IPython中输入import matplotlib时，若不报错则说明该库已经成功安装并导入。

本书中用到的Matplotlib版本是4.5.3，具体查阅http://matplotlib.org/。

（8）图像处理库

图像处理提供针对视频和图像数据的输入、分析、处理和挖掘等功能，Python最常用的图像和视频处理库是PIL和OpenCV。

PIL

PIL（Python Imaging Library）是一个常用的图像输入、处理和分析库，提供了多种数据处理、变换的操作方法和属性。

PIL的安装直接使用pip install PIL（注意大小写）命令即可。安装成功之后，在IPython中输入import PIL（注意大小写）时，若不报错则说明该库已经成功安装并导入。

本书中用到的PIL版本是1.1.7。PIL的安装依赖于Python（=2.7.*），不支持Python 3，有关PIL的更多信息，具体查阅http://www.pythonware.com/products/pil/。

OpenCV

OpenCV是一个强大的图像和视频工作库。它提供了Python、C、C++和Java接口，支持Windows、Linux、Mac OS、iOS和Android。OpenCV的设计效率很高，它以优化的C/C++编写，库可以利用多核处理。除了对图像进行基本处理外，还支持图像数据建模，并预制了多种

图像识别引擎。

OpenCV的安装无法直接使用pip命令实现，需要手动下载OpenCV源文件，然后将特定文件复制到Python第三方库目录下。下面以Windows为例介绍具体过程。

第一步　进入http://opencv.org/opencv-3-2.html，点击“Windows self-extracting archive:sourceforge”，会打开一个新的页面并自动下载一个名为“opencv-3.2.0-vc14.exe”的文件。

第二步　下载完成后，双击该文件，按照提示将OpenCV的源码解压到任意目录。

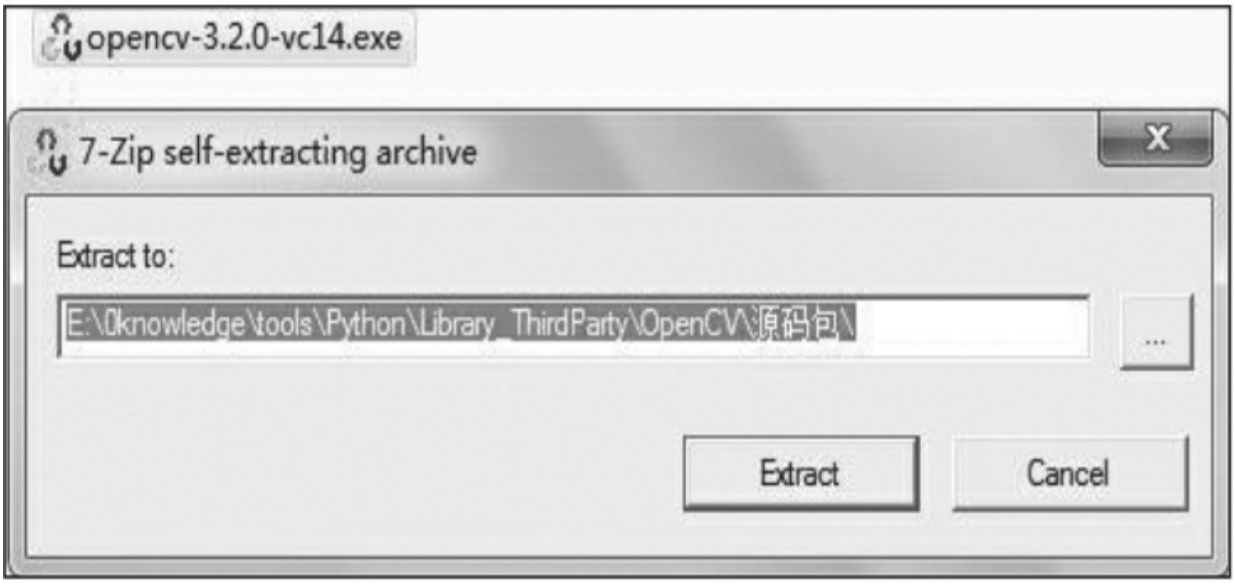


图1-4　解压OpenCV代码到任意目录

第三步　找到解压文件中路径为*\opencv\build\python\2.7\x86中的“cv2.pyd”文件，然后复制到Python的第三方库路径，路径地址为*\Python27\Lib\site-packages。

上述操作完成后，在IPython中输入import　cv2（注意库名）时，若不报错则说明该库已经成功安装并导入。

本书中用到的OpenCV版本是3.2.0。有关OpenCV的更多信息，请查阅http://opencv.org。

（9）其他库

根据实际案例，书中还会安装其他第三方库/包，具体会在场景中说明，在此不一一列出。

1.2.4 数据库和客户端

在本书中，大多数案例数据都会直接从数据库中读取，因为实际应用中的运营数据基本也是在数据库中直接获取并进行初步的数据探查工作，因此数据库是开展数据工作的基础工具。为了提高数据库的操作和使用效率，并能使更多读者快速入门数据库应用，我们会使用客户端工具，通过界面化的方式降低数据库的应用难度。数据库我们选择MySQL，客户端使用Navicat。

关于MySQL和Navicat软件的网络资源非常丰富，请读者自行寻找并下载安装相应版本，笔者的MySQL版本为32位版本5.0.51b。以下是有关MySQL配置过程中的关键点。

1）设置模式，如图1-5所示。

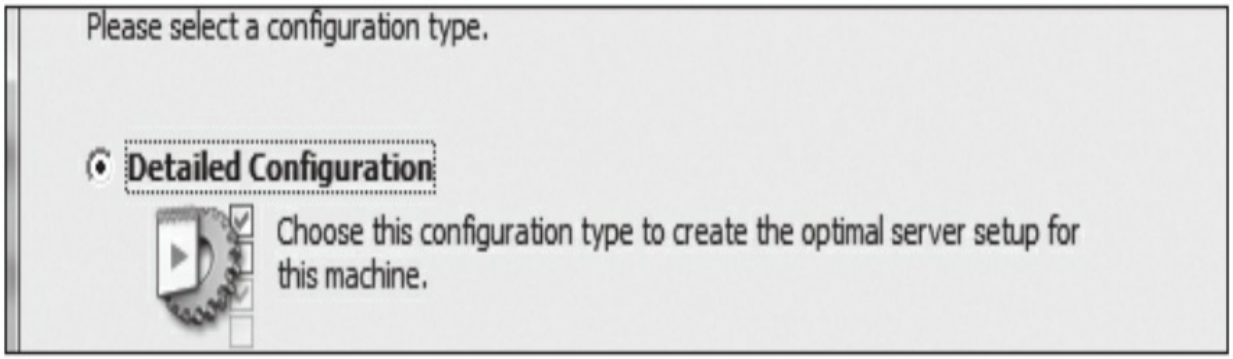


图1-5　设置MySQL模式

选择Detailed Configuration，目的是自己配置所有信息。

2）服务器实例配置，如图1-6所示。

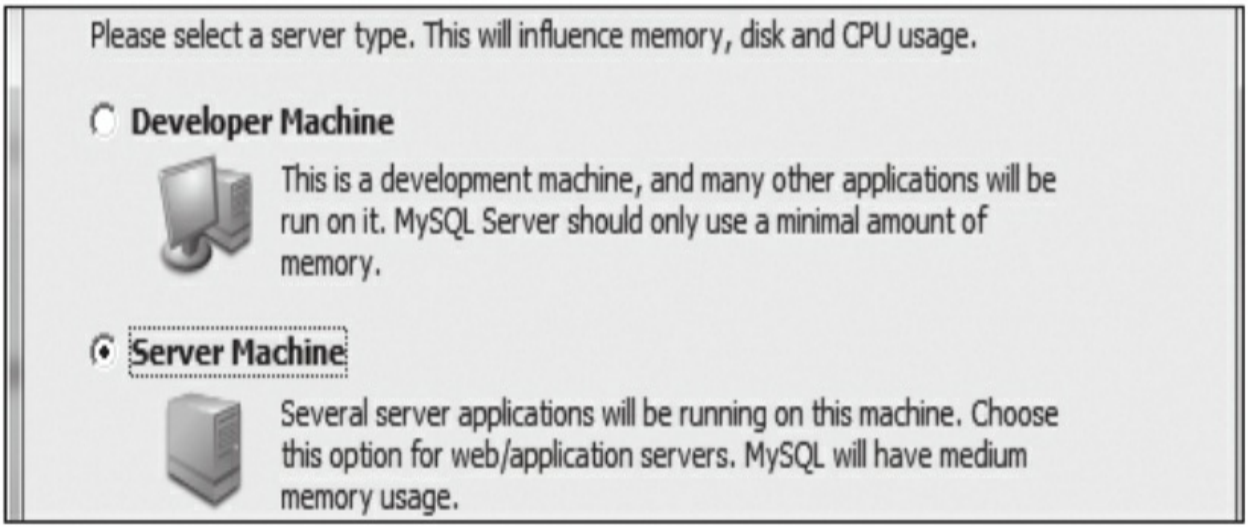


图1-6　配置服务器实例

如果本地电脑是多用途的，那么建议选择Server　Machine，兼顾效率和其他应用；如果优先要保证其他大型应用对资源的占用，那么选择Developer　Machine；如果要优先保障数据库资源，那么选择Dedicated MySQL Server Machine。

3）设置字符集，如图1-7所示。

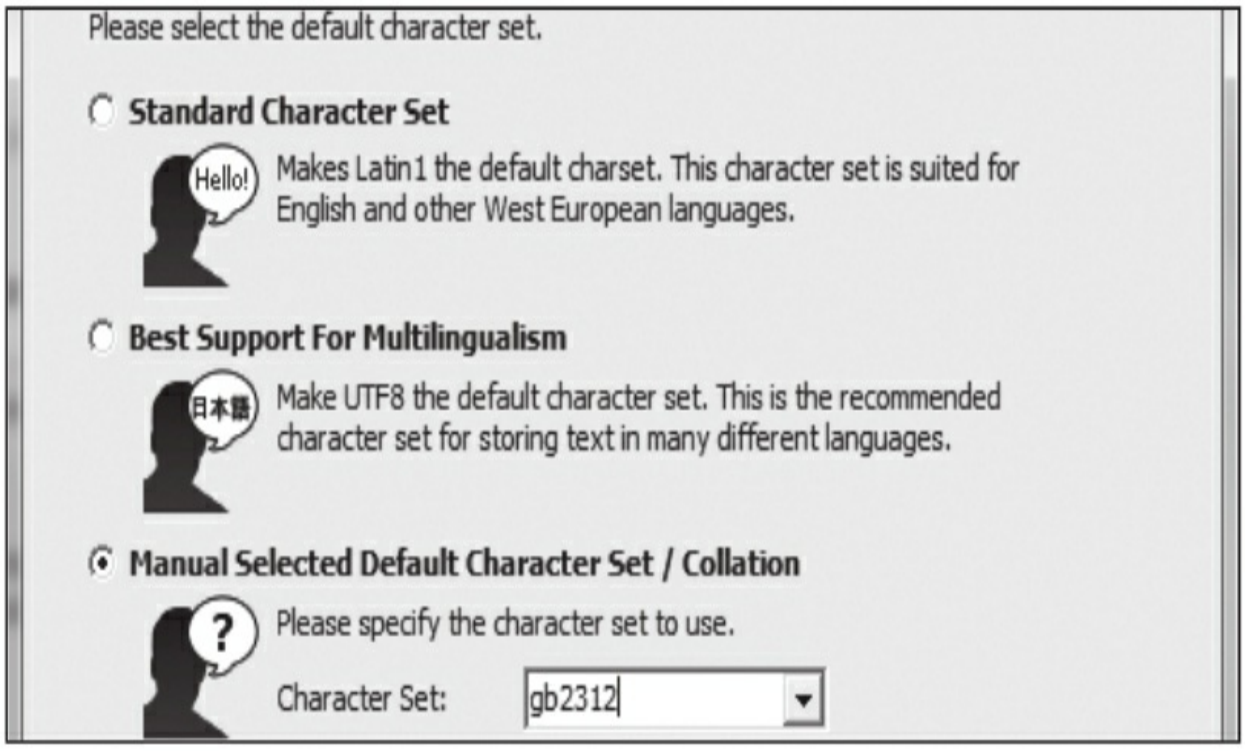


图1-7　设置字符集

这里选择手动设置，数据库字符集为gb2312，目的是兼容中文字符集并不至于使字库容量过大。

数据库安装并配置完成之后，通过Navicat客户端进行连接。方法是：点击顶部菜单“文件→新建连接→MySQL”，在弹出的对象框中配置如下信息：

·连接名：用来识别不同连接的名称。

·主机名/IP地址：本地使用127.0.0.1，远程服务器填写实际IP地址。

·埠（端口）：在配置MySQL时设置的端口，默认为3306（具体取决于配置MySQL时的设置）。

·用户名和密码：在配置MySQL时创建的用户名和密码。

输入完成后，点击“连接测试”，如果信息配置正确并且MySQL服务可用，那么会返回“连接成功”字样。

图1-8　新建数据库连接

有关使用客户端以及通过Python直接操作数据库的具体方法会在后续案例中详细介绍。

1.2.5 SSH远程客户端

对于数据工作而言，如果只是在本机上做数据分析处理和建模工作，通常只能利用有限的个人电脑性能实现有限规模的数据计算。当数据计算量或数据规模很大时，通常会选择在本地进行数据测试，然后到服务器上运行Python程序。此时，我们需要一个能在本机和服务器之间进行数据、信息和指令交互的SSH远程客户端工具。

对大多数数据工作者而言，本地电脑以Windows系统居多，而服务器以Linux系统居多，因此SSH远程客户端工具的主要作用就是连接Windows和Linux。此类工具很多，包括XShell、SecureCRT等，笔者使用的是SecureCRT。

1.3 内容延伸：Python的OCR和TensorFlow

1.3.1 OCR工具：Tesseract-OCR

OCR（Optical Character Recognition，光学字符识别）是一个非常“古老”的话题，原因是这项工作在20世纪90年代就已经广泛流行。但在大数据的背景下，我们要做的不是识别普通的光学字符（光学扫描文字），而是把范围扩大到识别更多领域的文字信息，例如手写识别、工业铭牌文字识别等非标准化、非印刷体、非传统资料录入的文字信息识别。

Tesseract-OCR是一个Google支持的开源OCR图文识别项目，支持超过200种语言（包括中文），并支持自定义训练字符集，支持跨Windows、Linux、Mac OSX等多平台使用。

不同的平台安装和配置Tesseract-OCR差异较大，下面以Windows为例说明其安装过程。

第一步　进入https://sourceforge.net/projects/tesseract-ocr-alt/files/，下载名为“tesser-act-ocr-setup-3.02.02.exe”的安装包。

第二步　下载安装tesseract-ocr-setup-3.02.02.exe。安装成功之后，在系统终端命令行窗口输入tesseract，可直接调用OCR命令。

```
C:\Users\Administrator>tesseract
Usage:tesseract      imagename      outputbase      [-l      lang]      [-
psm pagesegmode] [configfile...
]

pagesegmode values are:
0 = Orientation and script detection (OSD) only.
1 = Automatic page segmentation with OSD.
2 = Automatic page segmentation, but no OSD, or OCR
3 = Fully automatic page segmentation, but no OSD. (Default)
4 = Assume a single column of text of variable sizes.
5 = Assume a single uniform block of vertically aligned text.
6 = Assume a single uniform block of text.
7 = Treat the image as a single text line.
8 = Treat the image as a single word.
9 = Treat the image as a single word in a circle.
10 = Treat the image as a single character.
-l lang and/or -psm pagesegmode must occur before anyconfigfile.
```

```
Single options:
    -v --version: version info
    --list-langs: list available languages for tesseract engine
```

有关Tesseract-OCR的更多信息，具体请查阅https://github.com/tesseract-ocr/tesseract/wiki。

1.3.2 机器学习框架——TensorFlow

TensorFlow是谷歌基于DistBelief进行研发的第二代人工智能学习系统，它使用图模型将复杂的数据结构传输至人工智能神经网中进行分析和处理。它被广泛用于语音识别或图像识别等多项机器深度学习领域。

在TensorFlow出现之前，我们之前介绍的scikit-learn几乎是Python机器学习最流行的工具（或者至少是最流行的工具之一），借助谷歌的强大号召力以及在人工智能领域的技术实力，TensorFlow正在慢慢凸显优势，逐步成为Python领域最具有发展潜力的机器学习框架。

TensorFlow支持跨平台的应用，最新版本已经支持Windows。但遗憾的是TensorFlow只支持X64架构的Windows，这种架构更多集中应用在服务器上，而个人电脑通常是X86架构（包括32位和64位），所以在个人Windows电脑上通常是无法直接安装和使用TensorFlow的（可以在虚拟机上安装学习）。

有关TensorFlow的更多信息，具体查阅www.tensorflow.org。

1.4 第一个用Python实现的数据化运营分析实例——销售预测

1.4.1 案例概述

本节通过一个简单的案例，来介绍如何使用Python进行数据化运营分析。

案例场景：每个销售型公司都有一定的促销费用，促销费用可以带来销售量的显著提升；当给出一定的促销费用时，预计会带来多大的商品销售量？

在“附件-chapter1”中data.txt存储了建模所需的原始数据，get_started_example.py是案例完整代码。以下是原始数据概况：

·来源：生成的模拟数据，非真实数据。

·用途：用来做第一个销售预测案例。

·维度数量：1。

·记录数：100。

·字段变量：第一列是促销费用，第二列是商品销售量。

·数据类型：全部是浮点数值型。

·是否有缺失值：否。

1.4.2 案例过程

下面逐步解析整个分析和实践过程。

第一步 导入库。

本案例中，我们会使用四个库：

·Re：正则表达式，程序中通过该库来实现字符串分割。

·Numpy：数组操作和处理库，程序中用来做格式转换和预处理。

·Sklearn：算法模型库，程序中使用了线性回归方法linear_model。

·Matplotlib：图形展示库，用来在建模前做多个字段关系分析。

代码如下：

```
import re
import numpy
from sklearn import linear_model
from matplotlib import pyplot as plt
```

相关知识点：Python导入库

Python导入库有两种方式：

·导入直接导入库，方法是import[库名]，例如import numpy；对于某些名称比较长的库，我们会使用as方法为其取个别名以方便后续使用，例如import numpy as np。

·导入库中的指定函数，方法是from[库名]import[函数名]，例如from sklearn import linear_model；这种情况下也可以使用as为其取个别名方便后续使用，例如from matplotlib import pyplot as plt。

第二步 导入数据。

本案例中的数据存于txt格式文件中，我们使用Python默认的读取文

件的方法。代码如下：

```
fn = open('data.txt','r')
all_data = fn.readlines()
fn.close()
```

第一段代码fn=open（'data.txt'，'r'）的作用是打开名为“data.txt”的文件，文件模式是只读，并创建一个名为fn的文件对象，后续所有关于该文件的操作都通过fn执行。由于程序文件和数据文件处于同一个目录下，因此无须指定路径；也可以通过相对路径和绝对路径来设置完整路径。

指定相对路径：'../data/data.txt'，含义是“data.txt”位于当前Python工作目录的父级目录中的data文件夹中。

指定绝对路径：'d:/python_data/data/data.txt'，该方式中的绝对路径需要注意使用正斜杠/，而不是Windows默认的反斜杠\；如果一定要使用反斜杠，那么需要写成'd:\\python_data\\data\\data.txt'，用转义字符表示。

在Python中反斜杠作为转义字符存在，使用\\的意思是这是一个反斜杠符号。表1-1列出了Python常用的转义字符。

表1-1　常用转义字符列表

转义字符	描　述	转义字符	描　述
\(行尾时)	续行符，表示一行没有结束	\n	换行符
\\	反斜杠符号	\v	垂直制表符
\'	单引号	\t	水平制表符
\"	双引号	\r	回车符
\a	响铃符，发出系统响铃声	\f	换页符
\b	退格符(Backspace)	\o	八进制数代表的字符
\e	转义符	\x	十六进制数代表的字符
\000	终止符，\000后的字符串全部忽略	\other	其他的字符以普通格式输出

第二段代码all_data=fn.readlines（）的意思是从fn中读取所有的行记录，并保存到一个名为all_data的列表中。我们可以通过all_data[0]来查看该列表的第一个数据，Python中的数据索引都是从0开始，第0个索引值对应第一个值。

```
In[3]:  all_data[0]
Out[3]: '28192.0\t68980.0\n'
```

通过查看第一个数据，我们了解了数据的基本情况，这是一个由数字、水平制表符和回车符组成的字符串，两个数字之间使用\t水平制表符分割，每个字符串以\n回车符结尾。该信息会为后续做数据预处理提供思路。

第三段代码fn.close（）的意思是关闭文件对象的占用。当文件读写完成后，需要及时关闭资源占用。

不仅是文件对象，包括数据库游标、数据库连接等资源，在

使用完成后都需要及时关闭，这样能减少对资源的无效占用。同时，这还能防止当其他功能模块对该对象进行重命名、删除、移动等操作时，不会由于文件对象的占用而报错。

第三步　数据预处理。

在本阶段，主要实现对读取的列表数据进行清洗转换，以满足数据分析展示和数据建模的需要。代码如下：

```
x = []
y = []
for single_data in all_data:
    tmp_data = re.split('\t|\n',single_data)
    x.append(float(tmp_data[0]))
    y.append(float(tmp_data[1]))

x = numpy.array(x).reshape([100,1])
y = numpy.array(y).reshape([100,1])
```

代码x=[]和y=[]的意思是创建两个空列表，目的是用来存放自变量和因变量数据。

代码for single_data in all_data的意思是通过一个for循环每次从列表all_data中读取一条数据，并赋值给single_data。

代码tmp_data=re.split（'\t|\n'，single_data）的意思是，分别使用\t和\n作为分割符，对single_data进行数据分割，分割结果赋值为tmp_data。

代码x.append（float（tmp_data[0]））和y.append（float（tmp_data[1]））的意思是将tmp_data的第一个值追加到列表x中，将tmp_data的第二个值追加到列表y中。在追加之前，我们先将每个数据通过float方法设为浮点型（数值型）。

代码x=numpy.array（x）.reshape（[100，1]）和y=numpy.array（y）.reshape（[100，1]）的意思是将x和y由列表类型转换为数组类型，同时数组的形状是100行1列。

第四步　数据分析。

到现在止我们已经拥有了格式化的数据，但这两列数据集到底应该

使用哪种模型还未可知。因此先通过散点图来观察一下。代码如下：

```
plt.scatter(x,y)
plt.show()
```

代码plt.scatter（x，y）的意思是用一个散点图来展示x和y，plt.show（）的作用是展示图形。代码执行后会弹出如图1-9所示的散点图。

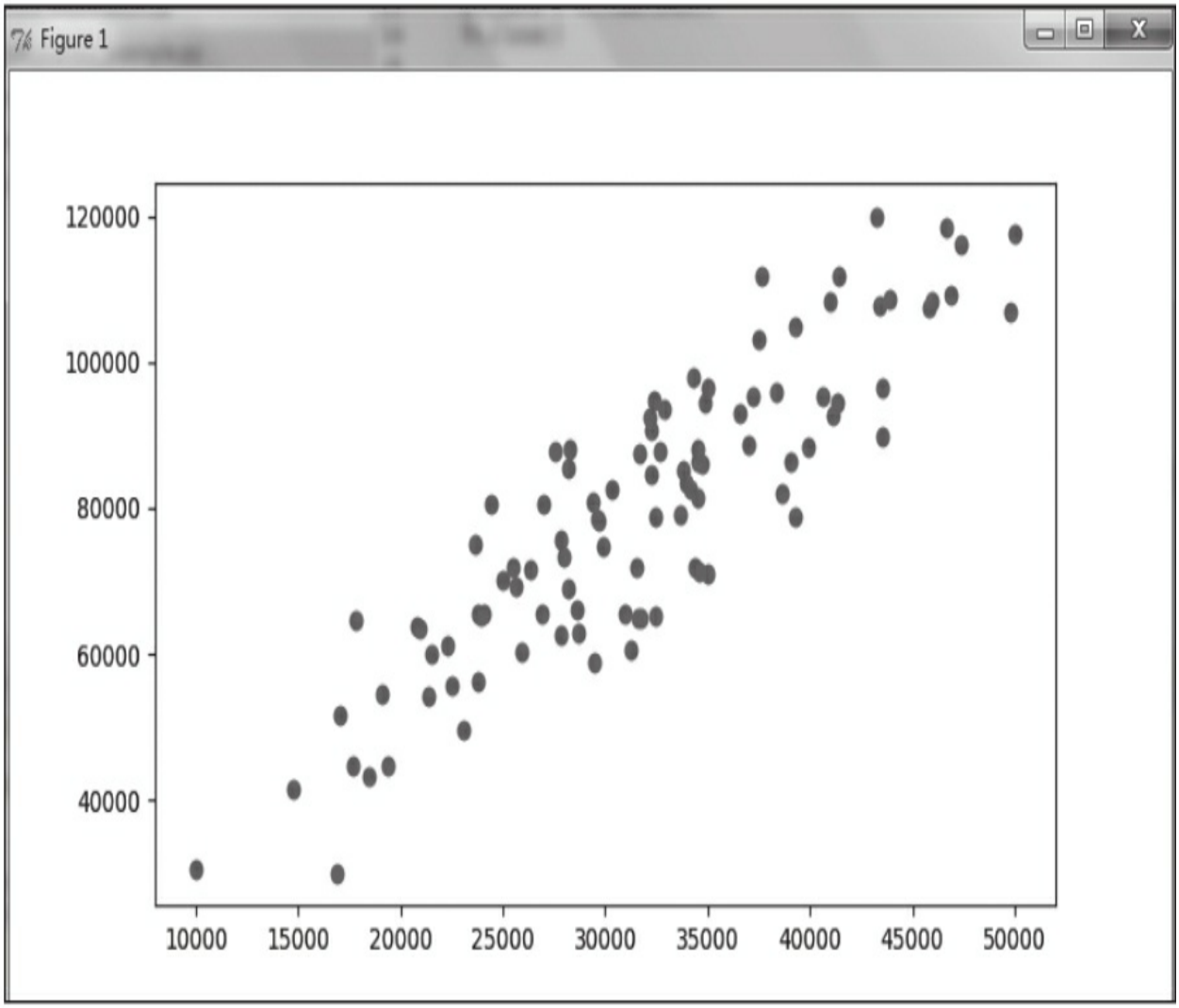


图1-9　散点图

通过散点图发现，x和y的关系呈现明显的线性关系：当x增大时，y增大；当x减小时，y减小。初步判断可以选择线性回归进行模型拟合。

第五步　数据建模。

建模阶段我们使用Sklearn中的线性回归模块实现，代码如下：

```
model = linear_model.LinearRegression()
model.fit(x, y)
```

代码model=linear_model.LinearRegression（）的作用是创建一个模型对象，后续所有的模型操作都基于该对象产生。

代码model.fit（x，y）的作用是将x和y分别作为自变量和因变量输入模型进行训练。

第六步　模型评估。

模型已经创建完成，本阶段进行模型拟合的校验和评估，代码如下：

```
model_coef = model.coef_
model_intercept = model.intercept_
r2 = model.score(x,y)
```

代码model_coef=model.coef_的作用是获取模型的自变量的系数并赋值为model_coef；代码model_intercept=model.intercept_的作用是获取模型的截距并赋值为model_intercept；代码r2=model.score（x，y）获取模型的决定系数R的平方。

通过上述步骤我们可以获得线性回归方程y=model_coef*x+model_inter-cept，即y=2.09463661*x+13175.36904199。该回归方程的决定系数R的平方是0.7876414684758954 5，整体拟合效果不错。

第七步　销售预测。

我们已经拥有了一个可以预测的模型，现在我们给定促销费用（x）为84610，销售预测代码如下：

```
new_x = 84610
pre_y = model.predict(new_x)
print (pre_y)
```

代码new_x=84610的作用是创建促销费用常量，用来做预测时输入。代码pre_y=model.predict（new_x）的作用是对促销费用常量new_x输入模型进行预测，将预测结果赋值为pre_y。代码print（pre_y）的作用是打印输出销售预测值。

代码执行后，会输出[[190402.57234225]]，这就是预测的销售量。为了符合实际销量必须是整数的特点，后续可以对该数据做四舍五入，使round（pre_y）获得预测的整数销量。

相关知识点：如何执行Python代码？

对稍微有些经验的程序员或Python工程师来讲这当然不是什么问题，但对第一次接触Python的读者来讲，这也许是一个需要补充的必要知识点。

对于Python代码，既可以只执行特定代码行/段，也可以执行整个代码文件。

场景1：执行特定代码行/段。

这种方式通常是以调试的方式逐行/模块运行代码，大多应用在单功能、模块的开发、调试和测试上。执行方式如下：

1）在Python命令行中执行：打开系统终端命令行窗口，输入python进入Python命令行界面，然后逐行或模块输入上述Python代码即可。

2）在IPython命令行中执行：在打开系统命令行窗口，输入ipython进入ipython命令行界面，然后逐行或模块输入上述Python代码即可。

3）在PyCharm中执行：PyCharm是本书推荐使用的Python IDE，笔者推荐使用这种方式进行代码功能开发和测试。打开PyCharm程序，第一次需要新建一个项目用来存储和管理所有即将开发的Python资源，Location（位置）指向本书的“附件”根目录，然后点击Create（创建）。如图1-10所示。

创建完成后的PyCharm界面如图1-11所示，除了顶部菜单栏外，下面的内容区分为4个部分，从①到④依次是项目/结构区、代码区、调试区、智能提示区。

·项目/结构区：项目方式展示本项目所有文件、外部连接库，结构方式展示当前文件下所有的函数、变量等详细信息。

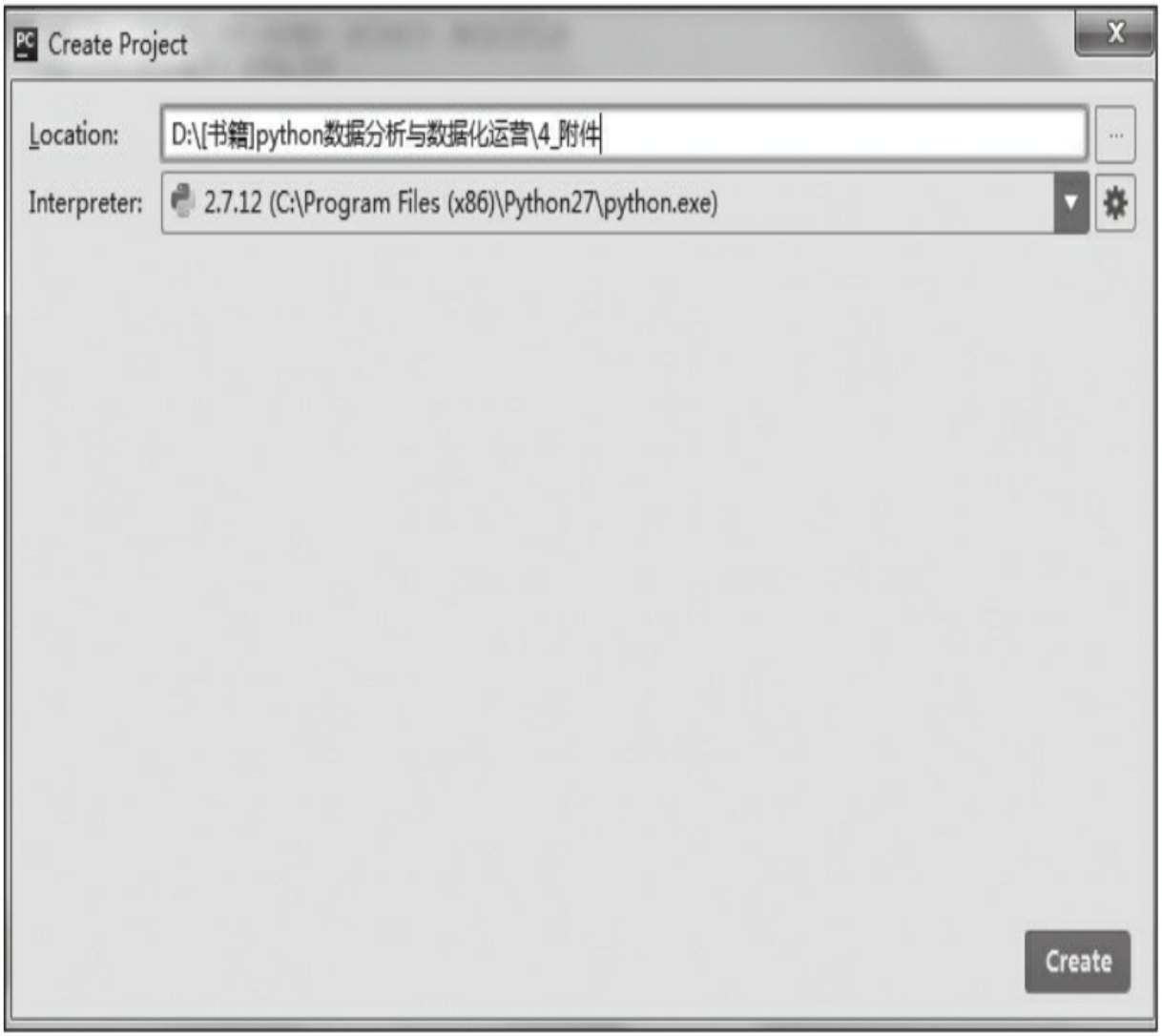


图1-10　在PyCharm中新建Python项目

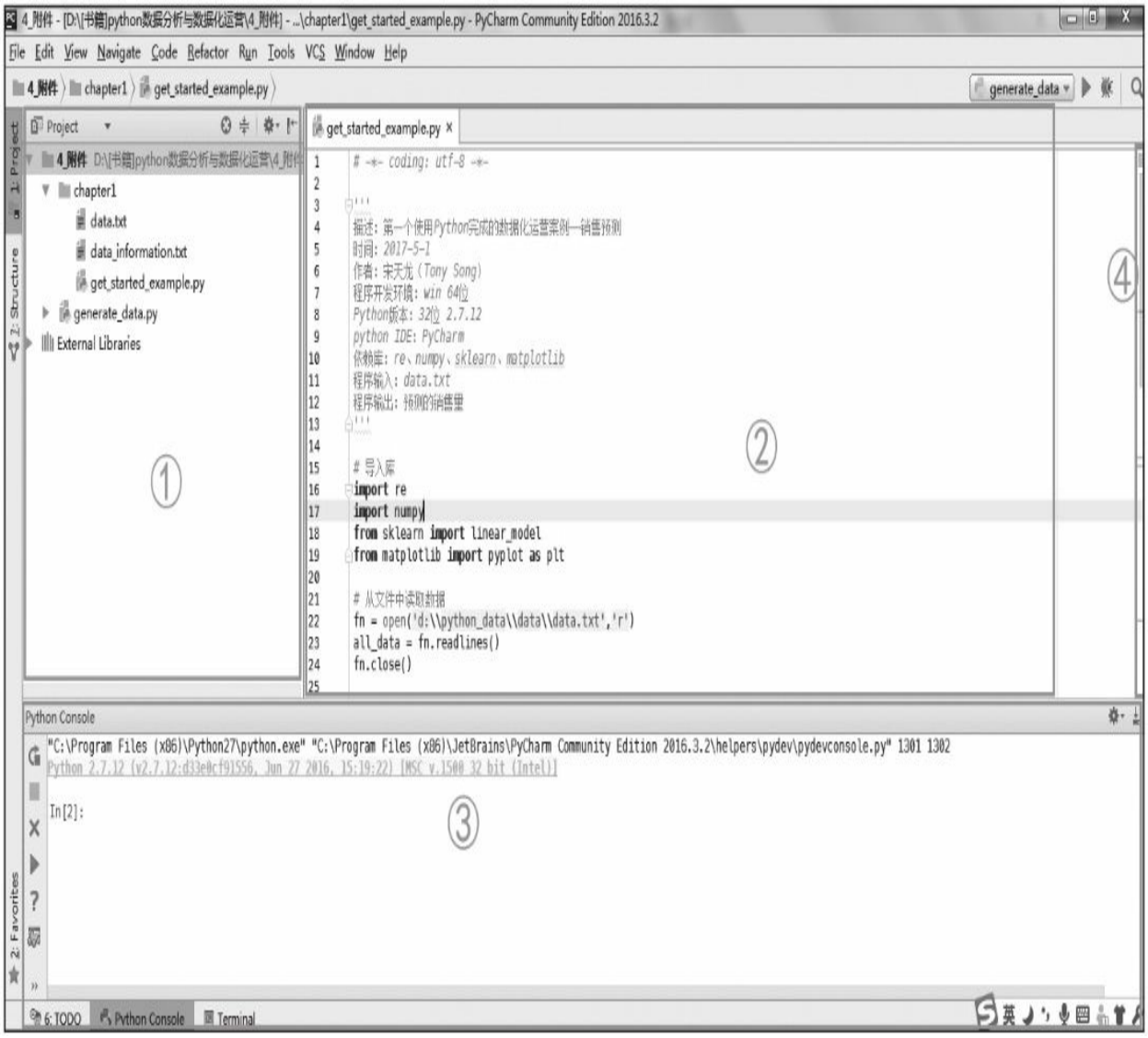


图1-11　PyCharm项目界面概览

·代码区：展示所有程序代码、注释的区域。

·调试区：分为TODO、Console、Terminal三个模块，分别用于提供注释的TODO列表、调试和终端功能。

·智能提示区：给出代码中可能存在的格式、引用、语法等方面的错误或警告信息。

在代码区，用鼠标选中要调试的程序，直接输入组合键ALT+Shift+E，然后在调试区可以看到该代码执行的状态。图1-12所示

是程序调试执行结果。

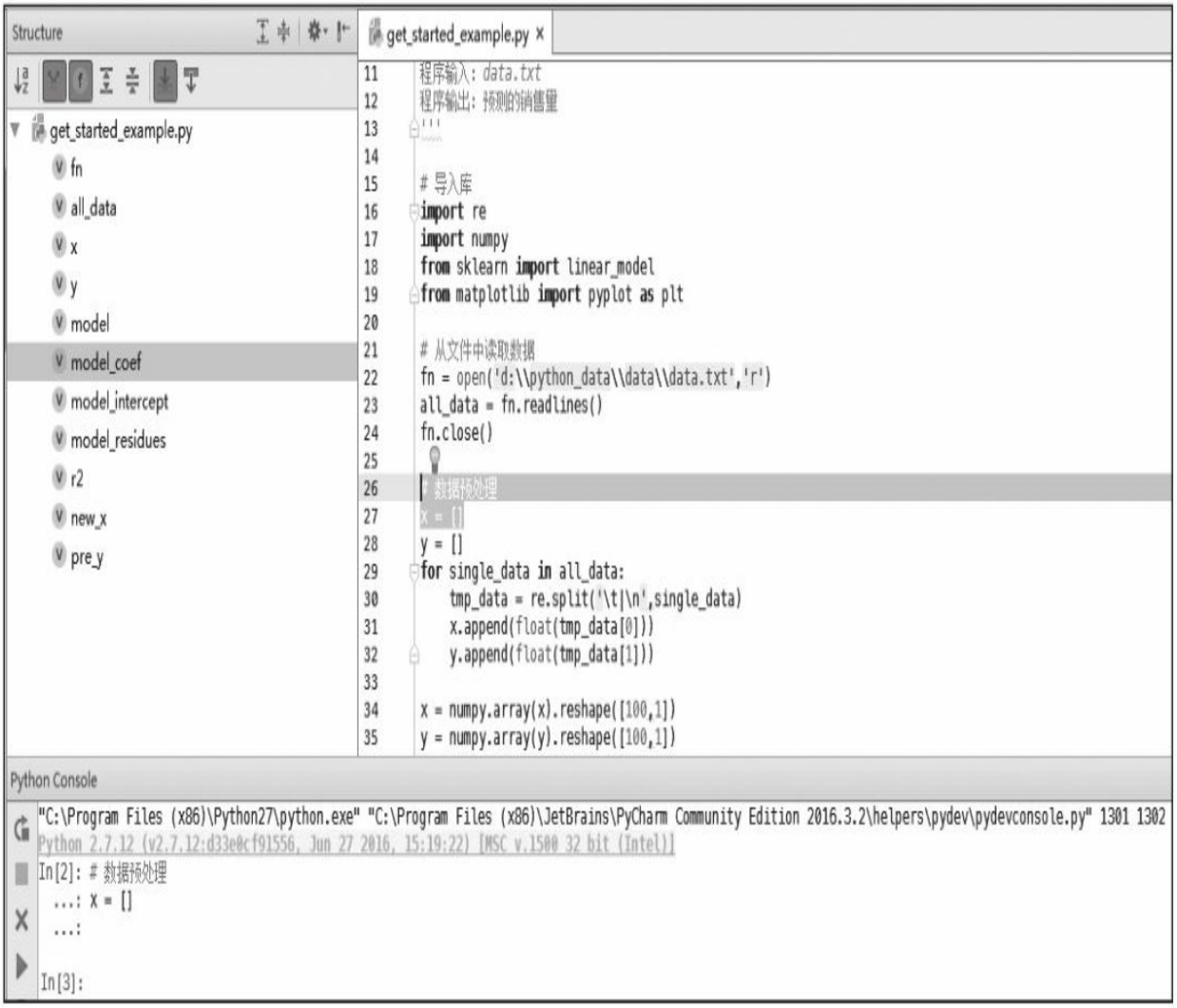


图1-12　程序调试执行结果

场景2：执行整个代码文件。

这种方式通常将Python文件作为整体的运行代码，一般在单个功能或模块开发完成之后，在整体或多功能模块的测试、集成或程序间调用时使用。

1）在系统终端命令行中调用Python命令执行：打开系统命令行窗口，输入python+[python文件名称].py。图1-13为运行本章案例的Python程序文件，执行结果将先显示一个图形文件，关闭图形之后会继续运行显示预测结果。

图1-13　通过系统终端命令行中调用Python命令执行

2）在IPython命令行中执行：打开系统命令行窗口，输入ipython进入ipython命令行界面，然后在交互命令窗口输入run+[python文件名称].py，如图1-14所示。

3）在PyCharm中执行：在PyCharm中新建项目后，双击左侧项目/结构区要执行的Python文件，右侧会显示对应的Python文件的代码内容。输入组合快捷键Alt+Shift+F10或通过顶部菜单栏“Run→Run”，在弹出的窗口中选择要执行的Python文件。如图1-15所示，程序文件执行后，会在底部调试窗口显示执行结果。

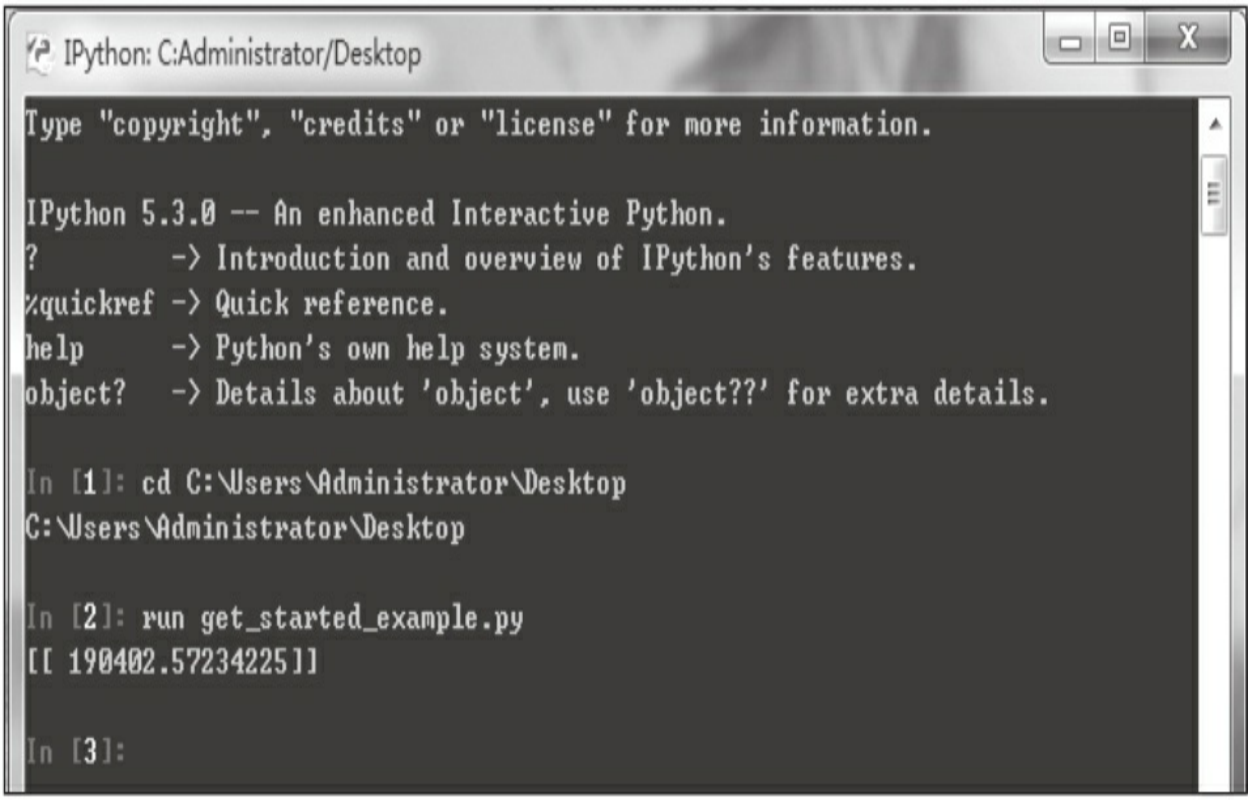


图1-14　在IPython中执行Python文件

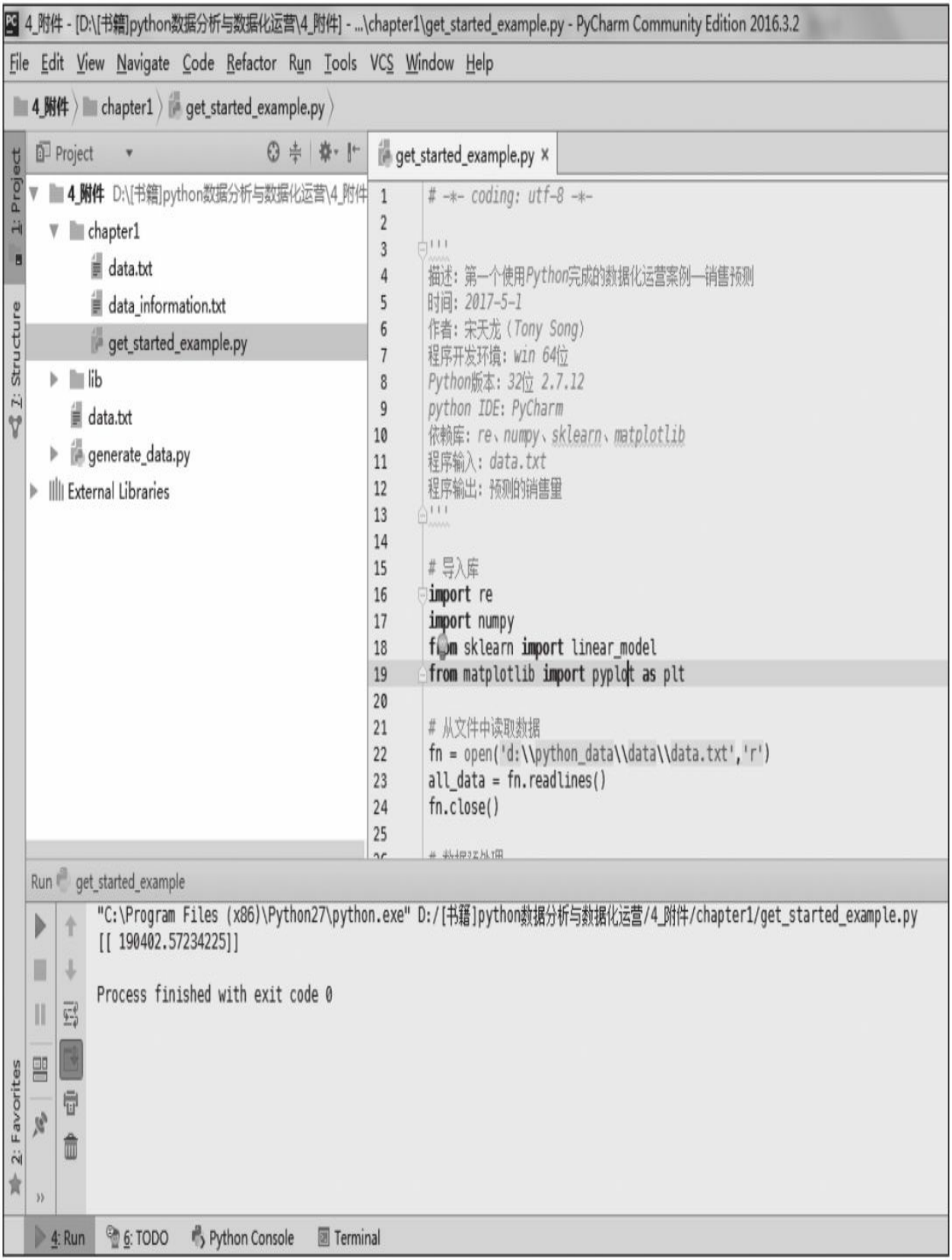


图1-15　在PyCharm中执行Python文件

在PyCharm中已经集成了IPython命令行和系统终端窗口功能，点击“Python Console”后的调试界面，默认调用的是IPython库；点击“Terminal”后的调试界面正是系统终端的命令行界面。读者可直接在此直接操作上述命令，实际上，这也是笔者着重推荐使用PyCharm作为IDE的主要原因之一，在一个工具里面可以直接实现不同的调试场景：

·安装第三方库时使用pip命令或需要从终端调用命令时，可直接点击“Terminal”操作；

·想要使用IPython的友好开发交互和提示功能，可直接在“Python Console”中完成；

·想要完整查看Python项目资源并进行管理，尤其是代码审查等方面，PyCharm可以完美胜任。

1.4.3 案例小结

本案例看似篇幅很长，其实代码本身只有40多行，却实现了导入库、获取数据、数据预处理、数据展示分析、数据建模、模型评估和销售预测7个关键步骤，麻雀虽小五脏俱全。

案例场景虽然简单，但完整地演示了如何从输入数据到输出结果的整个过程，其中，我们用到了以下基础知识：

·Python文件的读取；

·Python基本操作：列表操作（新建、追加）、for循环、变量赋值、字符串分割、数值转换；

·Numpy数组操作：列表转数组、重新设置数组形状；

·使用Matplotlib进行散点图展示；

·使用Sklearn进行线性回归的训练和预测；

·用print输出指定数据。

这是本书的第一个完整案例的目的是引导读者快速进入使用Python进行数据化运营的场景中来，因此笔者并不希望让读者陷入复杂的逻辑和太多知识当中，以下内容仅做拓展思考之用：

·通过散点图初步判断线性回归是比较好的拟合模型，此时应思考，是否有其他回归方法会得到更好的效果，例如广义线性回归、SVR（SVM中的回归）、CART（分类回归树）等？

·通过图形法观察数据模型只适合用于二维数据，如果数据输入的维度超过2个呢？

·本案例中的数据量比较小，如果数据量比较大，假如有1000万条，如何进行数据归约？

·除了案例中的评估指标外，还有哪些指标可以做效果评估？

1.5 本章小结

内容小结：本章从Python和数据化运营的关系、数据化运营所需要的Python相关工具组件入手，介绍了有关Python和数据化运营的理念、关系、流程和工具，并通过一个小案例演示了如何通过Python进行销售预测。

重点知识：有关Python的相关工具部分，这些内容是本书后续所有工作的基础，希望读者能在本机上进行安装、测试和学习。

外部参考：限于篇幅，本章没有对Python和相关工具的基础知识进行讲解，只是对涉及了案例中的部分内容。因此，很多知识需要读者书外“补习”。这些知识除了包含Python基础知识和科学计算以外，还有数据库、Tesseract、TensorFlow等工具，它们将构成数据基础工作和数据延伸工作的基石。作为本章内容的延展，笔者列出了相关资源，希望能为读者提供必要参考。

以下列出了与Python相关的主要的官方参考资源和信息：

·Python官方网站：https://www.python.org/。Python最权威的网站，包含有关Python的帮助手册、新闻、事件、应用、案例、社区等，并提供官方Python所有版本和环境的安装程序和安装包。

·Python pypi第三方库：https://pypi.python.org/pypi。Python使用pip命令安装时，请求的资源就来源于该网站，这里汇聚了第三方Python程序的软件仓库，截至本书完稿时已有101866个软件包。你可以在这里直接查看、下载和评论第三方库。

·Stack Overflow：http://stackoverflow.com/。Stack Overflow是一个与程序相关的IT技术问答网站，用户可以在网站上免费提交、浏览和检索问题。大多数情况下，你的问题都不是第一次出现，所以有问题了不妨先在这里找找答案。

·Python内部帮助文档和信息：在IPython命令行窗口使用help（）和dir（）命令。例如：通过dir（numpy.mean）查找Numpy库下面的mean函数的大部分属性；通过help（numpy.mean）获得该函数的具体介绍、

参数解释、应用举例等详细信息。这是针对特定知识点最为主要的学习参考资源。

大多数情况下，通过上述方法可以了解到Python基本知识，但以下图书资源会帮助你更加深入了解Python及其相关库的工作方式和逻辑，尤其是对于数据挖掘、机器学习等领域的认知：

·《利用Python进行数据分析》（Python for Data Analysis），介绍了Python用于数据分析的几个主要科学计算和展示库Numpy、Pandas、Matplotlib等，书中对这些库的讲解略粗，但全书的逻辑体系完整，适合数据分析和挖掘工作者作为入门读的。

·《Python数据分析与挖掘实战》，以数据工作流的方式展开介绍Python数据应用，书籍的逻辑结构较为完整，后面也有部分案例的介绍，适合对Python有一定了解的数据工作者阅读。

·《机器学习实战》（Machine learning in action），这是使用Python进行机器学习的专业书籍，需要读者具有一定的算法、程序和模型专业知识，适合中高级数据挖掘和建模工程师阅读。

·《集体智慧编程》，这是使用Python进行机器学习的专业书籍，与传统机器学习书籍不同的是，本书没有按照算法分类分别进行阐述，而是从应用的角度分场景介绍，本书需要读者了解工程、算法和模型知识，适合中高级数据挖掘、建模工程师、程序员阅读。

·《Python基础教程（第2版修订版）》这是一本纯介绍Python编程语言的书籍，其中主要围绕每个方法、条件、函数、对象、属性等进行介绍，适合程序员以及想深入了解Python工作原理和逻辑的读者查阅。

数据分析师或挖掘工程师对数据库的应用主要集中在DDL（本机操作）和DML（本机和服务器操作）上，而DCL和TCL涉及相对较少。因此建议读者重点了解前两种语言的相关知识。

·MySQL官方资源：https://dev.mysql.com/doc/。所有有关MySQL的官方信息和知识，在这里都可以找到。

·MySQL第三方教程：http://www.runoob.com/mysql/mysql-

tutorial.html。言简意赅地介绍MySQL的基本用法，按照用法主题分类，并且是中文的，适合作为知识查找工具。

·《深入浅出MySQL（第2版）》，这是一本完整阐述MySQL开发、设计、运维、管理等方面的书籍，内容全面，并且有适合初学者的章节。

有关Tesseract的资源不多，目前主要是官方信息。

·Tesseract wiki：https://github.com/tesseract-ocr/tesseract/wiki。页面右侧按照不同的主题页面展示，可直接点击对应标题查看。

·Tesseract介绍：https://github.com/tesseract-ocr/docs。各种会议和演示的PPT材料和介绍信息。

·Tesseract训练数据集：https://github.com/tesseract-ocr/tessdata。注意页面中是Tesseract 4版本用的数据集，其他版本通过页面底部的入口查看。

·Tesseract语言文件：https://github.com/tesseract-ocr/langdata。按照语言类别归类到文件，用来作为特定的语言做重新训练时的主要过程数据和文件。

TensorFlow作为2015年年底“刚”开源的机器学习框架，其学习资源不多，原因是开源之后即使有大型公司或团队使用，也需要经过一定时间的技术实践和应用。

·TensorFlow官方网站：http://www.tensorflow.org/。要打开这个网站需要一定的工具或技巧。

·TensorFlow中文社区：http://www.tensorfly.cn/。相当于汉化版的官方网站。

·《TensorFlow实战》：这是一本国内为数不多的Tensorflow实战书籍。

·极客学院的TensorFlow官方文档中文

版：http://wiki.jikexueyuan.com/project/tensor-flow-zh/。在TensorFlow刚开源1个多月时，极客学院就组织了很多人进行翻译。

应用实践：读者可以自己手写一个Python工作（比如预测）案例，也许这个过程中会出现很多意想不到的问题，但别担心，总有很多途径可以解决这些问题，并且解决问题的过程正是学习的过程，通过简单的练习可以掌握Python工作的基本原理和方法。

第2章　数据化运营的数据来源

“巧妇难为无米之炊”，对于数据工作者来说数据便是所有数据工作的基础。企业的数据化运营的数据来源复杂，从数据结构类型上来讲，包括结构化和非结构化数据；从数据来源方式来分，既有导出的数据文件、数据库等常见来源，又有流式、API等复杂系统接口和外部资源数据。

本章将从数据类型和数据来源两个方面介绍数据化运营的数据来源，在第三部分我们还会简单介绍有关读取非结构化数据集的知识，包括网页抓取数据、文本、图像、视频、语音等，用来做数据化的整体数据资源的整合。

2.1 数据化运营的数据来源类型

数据化运营的数据来源类型包括数据文件、数据库、API、流式数据、外部公开数据和其他来源等。

2.1.1 数据文件

数据文件就是存储数据的文件，广义上，任何文件中存储的信息都可以称为数据；狭义上，数据文件中以数字或文本形式存储的结构化的数据记录才是数据。本节的数据指的是后者。

结构化数据文件大多来源于数据库，例如从MySQL中导出2017-01-04到2017-10-21的订单明细数据并存储为csv文件；也有来源于系统或工具的工作过程或返回结果，例如Windows版本的Tesseract文字识别后的结果会存储到txt文本文件中。

数据文件常见的数据格式类型包括txt、csv、tsv、xls、xlsx等，也包括xml、html、doc、sql等非常规数据格式。文件格式取决于数据处理需求，也受限于来源系统的导出格式。图2-1所示为MySQL 5.0版本可以导出的数据格式。

图2-1　MySQL 5.0版本支持导出的数据格式类型

大多数情况下，txt（任意指定分隔符）、cvs（以逗号分隔的数据文件）、tsv（以tab制表符分隔的数据文件）是最常用的数据文件格式。当数据文件大小在百兆级别以下时，可以使用Excel等工具打开；数据文件大小在百兆级别时，推荐使用Notepad打开；当数据文件大小在G级别时，推荐使用UltraEdit打开。

2.1.2 数据库

数据库（DataBase）是按照数据结构来组织、存储和管理数据的仓库。数据库广泛应用于CMS（内容管理系统）、CRM（客户关系管理系统）、OA（办公自动化）、ERP（企业资源计划）、财务系统、DSS（决策支持系统）、数据仓库和数据集市、进销存管理、生产管理、仓储管理等各类企业运营事务之中。

数据库的主要应用包括数据的定义、存储、增加、删除、更新、查询等事务型工作，数据传输、同步、抽取、转换、加载等数据清洗工作，数据计算、关联查询、OLAP等分析型工作以及数据权限控制、数据质量维护、异构数据库和多系统通信交互等工作。

数据库按类型分为关系型数据库和非关系型数据库（又称NoSQL数据库）。关系型数据库在企业中非常常见，在传统企业中更为流行，常见的关系型数据库包括DB2、Sybase、Oracle、PostgreSQL、SQL Server、MySQL等；非关系型数据库随着企业经营场景的多样化以及大数据场景的出现，根据应用场景和结构分为以下几类：

·面向高性能并发读写的键值（Key-Value）数据库：优点是具有极高的并发读写性能、查找速度快，典型代表是Redis、Tokyo Cabinet、Voldemort。

·面向海量文档的文档数据库：优点是对数据要求不严格，无须提前定义和维护表结构，典型代表为MongoDB、CouchDB。

·面向可扩展性的列式数据库：优点是查找速度快，可扩展性强，通过分布式扩展来适应数据量的增加以及数据结构的变化，典型代表是Cassandra、HBase、Riak。

·面向图结构的图形数据库（Graph Database）：优点是利用图结构相关算法，满足特定的数据计算需求，例如最短路径搜寻、关系查询等，典型代表是Neo4J、InfoGrid、Infinite Graph。

关系型数据库几乎是企业数据存储的“标配”，因此掌握数据

库的相关操作（主要是DDL和DML两种数据库语言）是每个数据工作者的必备技能之一。而非关系型数据库通常基于大数据平台或大数据场景产生，并不是每个企业都有应用场景，因此该技能通常作为加分项。

2.1.3 API

API（Application Programming Interface）是应用程序编程接口，数据化运营中的API通常分为服务型API和数据型API。

服务型API可以基于预定义的规则，通过调用API实现特定功能。例如，通过调用百度地图JavaScript API可以在网站中构建功能丰富、交互性强的地图应用，这种API下输入的是地理位置数据，从API获得的输出是可视化地图（服务/功能）。

数据型API则通过特定的语法，通过向服务器发送数据请求，返回特定格式的数据（或数据文件）。例如，通过向Google Analytics的Analytics Reporting API V4发送请求来获得符合特定条件的数据记录。

API广泛应用于企业内部和外部多系统和平台交互。API返回的数据格式，大多数情况下是JSON、XML格式。

JSON是一种轻量级的数据交换格式，由流行的JavaScript编程语言创建，广泛应用于Web数据交互。JSON格式简洁、结构清晰，使用键值对（Key:Value）的格式存储数据对象。Key是数据对象的属性，Value是数据对象属性的对应值。例如，“性别”：“男”就是一个Key:Value结构的数据。JSON格式数据示例如下：

```
{
    "category": {
        "name": "电脑",
        "brands": {
            "brand": [
                "DELL",
                "THINKPAD"
            ]
        }
    }
}
```

XML是可扩展标记语言，提供了统一的方法来描述和交换独立于应用程序或供应商的结构化数据，这是一种非常成熟且强大的数据格式。像JSON一样，XML提供了非常好的扩展性，API的创建者可以使用它们创建自己的数据结构。XML格式数据示例如下：

```
<?xml version="1.0" encoding="utf-8"?>
<category>
    <name>电脑</name>
    <brands>
        <brand>DELL</brand>
        <brand>THINKPAD</brand>
    </brands>
</category>
```

2.1.4 流式数据

流式数据指的是实时或接近实时处理的大数据流。常见的流式数据处理使用Spark、Storm和Samza等框架，能在毫秒到秒之间完成作业，用于处理时效性较强的场景，例如在线个性化推荐系统、网站用户实时行为采集和分析、物联网机器日志实时分析、金融实时消费反欺诈、实时异常人员识别等，应用领域集中在实时性较强的互联网、移动互联网、物联网等。

按照数据对象来区别，流式数据可分为两类：

第一类是用户行为数据流。用户行为数据流是围绕“人”产生的数据流，包括用户在网站和APP内部因浏览、搜索、评论、分享、交易以及在外部的微博、微信中操作而产生的数据流。用户行为数据流采集平台可分为Web站、移动站和APP（包含iOS、Android、Windows等）应用。Web站及基于HTML5开发的移动应用都支持JS脚本采集，较早开发的不支持JS的Wap站（现在基本上很少）则采用NoScript方法，即一个像素的硬图片实现数据跟踪。SDK是针对APP提供数据采集的特定方法和框架。这三种方法可以实现目前所有线上用户行为数据采集的需求。

第二类是机器数据流。机器数据流是围绕“物”产生的数据流，包括从机器的生产、制造、应用、监控和管理等过程中产生的所有数据，例如机器运行日志、传感器监控数据、音频采集器数据、监控图像和视频、GPS地理数据等。

机器数据流通常集中在工业4.0、智能工厂等工业的智能运营管理过程中，也出现在物联网、人工智能等人和物的监控、识别、联通、互动等智能化应用中。机器数据流扮演着实时采集目标对象属性、状态、行为、信号等数据的角色。

2.1.5 外部公开数据

外部公开数据指公开的任意第三方都能获取的数据。

数据化运营所需的外部公开数据来源渠道众多，常见的包括：

·政府和相关机构提供的公开数据，例如国家统计局提供的月度CPI数据；

·竞争对手主动公开的数据，例如通过新闻发布会、网络宣传等发布的数据；

·行业协会或相关平台组织提供的统计、资讯数据，例如艾瑞提供的行业研究报告发布的数据；

·第三方的组织或个人披露的与企业运营相关的数据，例如有关竞争对手的供应商、客户等数据。

2.1.6 其他

在某些场景下，企业数据化运营所用数据还会有其他来源，例如通过调研问卷获得的有关产品、客户等方面的数据，从第三方平台直接购买的数据，通过与其他厂商合作所得交互数据等。由于这些场景比较少，并且不是企业主流的数据获取来源，在此不作过多阐述。

2.2 使用Python获取运营数据

使用Python获取数据，目前主要的方法集中在文本文件、Excel文件、关系型和非关系型数据库、API、网页等方面。本节就来介绍具体获取方法。注意，本章的示例代码中，默认Python的工作目录与数据文件所在位置相同，即都是在“附件-chapter2”。如果Python的工作目录不在相同目录下，请先在PyCharm中的Python Console中使用cd方法切换到相同目录；或者直接修改示例代码中的文件为实际路径即可。

2.2.1　从文本文件读取运营数据

1.使用read、readline、readlines读取数据

Python可以读取任意格式的文本数据，使用Python读取文本数据的基本步骤是：

1）定义数据文件；

2）获取文件对象；

3）读取文件内容；

4）关闭文件对象。

（1）定义数据文件

定义数据文件即定义要读取的文件，该步骤不是必须的，可以跟“获取文件对象”整合。但为了后续操作的便捷性、全局数据对象的可维护性以及减少代码冗余，建议读者养成习惯，将数据文件预先赋值给一个对象。

定义文本数据文件的方法是：

```
file_name = [文件名称]
```

示例

```
file_name = 'd:/python_data/data/text.txt'
```

文件名称中可以只写文件名，此时默认Python读取当前工作目录下的文件；也可以加入路径，默认使用斜杠，尤其是Windows下要注意用法。

（2）获取文件对象

获取文件对象的意义是基于数据文件产生对象，后续所有关于该数据文件的操作都基于该对象产生。

语法：

```
file object = open(name [, mode][, buffering])
```

参数：

·name：要读取的文件名称，即上一个环节定义的file_name，必填。

·mode：打开文件的模式，选填，在实际应用中，r、r+、w、w+、a、a+是使用最多的模式。完整mode模式如表2-1所示。

表2-1　Python文件打开模式（mode）

模式 (mode)	描　述
r	以只读方式打开文件。文件的指针将会放在文件的开头。默认打开模式
rb	以二进制格式打开一个文件用于只读
r+	打开一个文件用于读写
rb+	以二进制格式打开一个文件用于读写
w	打开一个文件只用于写入。如果该文件已存在则将其覆盖；如果该文件不存在，则创建新文件
wb	以二进制格式打开一个文件只用于写入。如果该文件已存在则将其覆盖；如果该文件不存在，则创建新文件
w+	打开一个文件用于读写。如果该文件已存在则将其覆盖；如果该文件不存在，则创建新文件
wb+	以二进制格式打开一个文件用于读写。如果该文件已存在则将其覆盖；如果该文件不存在，则创建新文件
a	打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾，也就是说，新的内容将会被写入到已有内容之后；如果该文件不存在，创建新文件进行写入
ab	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾，也就是说，新的内容将会被写入到已有内容之后；如果该文件不存在，创建新文件进行写入
a+	打开一个文件用于读写。如果该文件已存在，文件指针将会放在文件的结尾，文件打开时会是追加模式；如果该文件不存在，创建新文件用于读写
ab+	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾；如果该文件不存在，创建新文件用于读写

·buffering：文件所需的缓冲区大小，选填；0表示无缓冲，1表示线路缓冲。

返回：通过open函数会创建一个文件对象（file object）。

示例：

```
file_name = 'text.txt'
file_object = open(file_name)
```

在“定义数据文件”部分提到可以将前两个环节结合起来，定义的语法是：

```
file_object = open('text.txt')
```

（3）读取文件内容

Python基于文件对象的读取分为3种方法，如表2-2所示。

表2-2　Python读取文件内容的3种方法

方　法	描　述	返回数据
read	读取文件中的全部数据，直到到达定义的 size 字节数上限	内容字符串，所有行合并为一个字符串
readline	读取文件中的一行数据，直到到达定义的 size 字节数上限	内容字符串
readlines	读取文件中的全部数据，直到到达定义的 size 字节数上限	内容列表，每行数据作为列表中的一个对象

示例，在“附件-chapter2”文件夹中有一个名为“text.txt”的数据文件，其中包含2个文本行，数据内容如图2-2所示。

·通过read方法读取该数据文件得到的数据结果：

```
'line1: This is line1\nline2: This is line2'
```

·通过readline方法读取该数据文件得到的数据结果：

```
'line1: This is line1\n'
```

·通过readlines方法读取该数据文件得到的数据结果：

```
['line1: This is line1\n', 'line2: This is line2
```

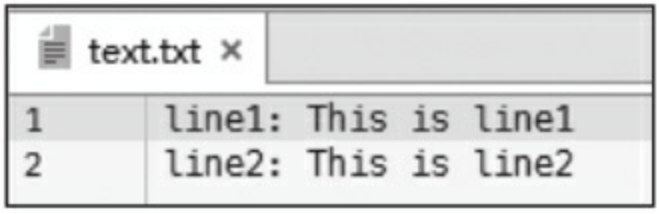


图2-2　示例数据源文件内容

在实际应用中，read方法和readlines方法比较常用，而且二者都能读取全部文件中的数据。二者的区别只是返回的数据类型不同，前者返回字符串，适用于所有行都是完整句子的文本文件，例如大段文字信息；后者返回列表，适用于每行是一个单独的数据记录，例如日志信息。不同的读取方法会直接影响后续基于内容的处理应用；readline由于每次只读取一行数据，因此通常需要配合seek、next等指针操作才能完整遍历读取所有数据记录。

相关知识点：指针

Python文件操作中的指针类似于Word操作中的光标，指针所处的位置就是光标的位置，它决定了Python的读写从哪里开始。默认情况下，当通过open函数打开文件时，文件的指针处于第一个对象的位置。因此，在上述通过readline读取文件内容时，获取的是第一行数据。仍然是上面示例中的数据文件，我们通过如下代码演示基于不同指针位置读取的内容：

```
fn = open('text.txt')# 获得文件对象
print (fn.tell())# 输出指针位置
line1 = fn.readline()# 获得文件第一行数据
print (line1) # 输出第一行数据
print (fn.tell())# 输出指针位置
line2 = fn.readline()# 获得文件第二行数据
```

```
print (line2)# 输出第二行数据
print (fn.tell())# 输出指针位置
fn.close() # 关闭文件对象
```

执行上述代码后，返回如下信息：

```
0
line1: This is line1
22
line2: This is line2
42
```

从返回结果看：当打开文件时，文件指针的位置处于文件开头，输出为0；当读取完第一行之后，文件指针位置处于第一行行末，位置是第22个字符后面（也就是'\n'后面，'\n'是换行符）；当读取完第二行（最后一行）之后，文件指针位置处于第二行行末，位置是第42个字符后面（注意：由于是最后一行，没有换行符'\n'）。

（4）关闭文件对象

每次使用完数据对象之后，需要关闭数据对象。方法是file_object.close（）。

理论上，Python可以读取任意格式的文件，但在这里通常以读取格式化的文本数据文件为主，其中包括txt、csv、tsv等格式的文件，以及有固定分隔符分隔并以通用数据编码和字符集编码（例如utf8、ASCII、GB2312等）存放的无扩展名格式的数据文件。在2.3节中我们会介绍使用Python读取非结构化数据的方法。

2.使用Numpy的loadtxt、load、fromfile读取数据

Numpy读取数据的方法包括loadtxt、load和fromfile等3种，概要描述如表2-3所示。

表2-3　Numpy读取文件的3种方法

方　法	描　述	返回数据
loadtxt	从 txt 文本中读取数据	从文件中读取的数组
load	使用 Numpy 的 load 方法可以读取 Numpy 专用的二进制数据文件，从 npy、npz 或 pickled 文件加载数组或 pickled 对象	从数据文件中读取的数组、元组、字典等
fromfile	使用 Numpy 的 fromfile 方法可以读取简单的文本文件数据以及二进制数	从文件中读取的数据

（1）使用loadtxt方法读取数据文件

Numpy可以读取txt格式的数据文件，数据通常都是一维或二维的。

语法：

```
loadtxt(fname,                                                    dtype=
<type 'float'>, comments='#', delimiter=None, converters= None, skiprows=0,
```

参数：

·fname：文件或字符串，必填，这里指定要读取的文件名称或字符串，支持压缩的数据文件，包括gz和bz格式。

·dtype：数据类型，选填，默认为float（64位双精度浮点数）。Numpy常用类型如表2-4所示。

表2-4　Numpy数据类型

类　型	描　述
bool	用一位存储的布尔类型（值为 TRUE 或 FALSE）
inti	由所在平台决定其精度的整数
int8	整数，大小为一个字节，范围：-128～127
int16	整数，范围：-32 768～32 767
int32	整数，范围：-2^{31}～$2^{31}-1$
int64	整数，范围：-2^{63}～$2^{63}-1$
uint8	无符号整数，0～255
uint16	无符号整数：0～65 535
uint32	无符号整数：0～$2^{32}-1$
uint64	无符号整数：0～$2^{64}-1$
float16	半精度浮点数：16 位，正负号 1 位，指数 5 位，精度 10 位
float32	单精度浮点数：32 位，正负号 1 位，指数 8 位，精度 32 位
float64 / float	双精度浮点数：64 位，正负号 1 位，指数 11 位，精度 52 位
complex64	复数，分别用于两个 32 位浮点数表示实部和虚部
complex128 / complex	复数，分别用两个 64 位浮点数表示实部和虚部

·comments：字符串或字符串组成的列表，是表示注释字符集开始的标志，选填，默认为#。

·delimiter：字符串，选填，用来分割多个列的分隔符，例如逗号、TAB符，默认值为空格。

·converters：字典，选填，用来将特定列的数据转换为字典中对应的函数的浮点型数据，例如将空值转换为0，默认为空。

·skiprows：跳过特定行数据，选填，用来跳过特定前N条记录，例如跳过前1行（可能是标题或注释），默认为0。

·usecols：元组，选填，用来指定要读取数据的列，第一列为0，以此类推，例如（1，3，5），默认为空。

·unpack：布尔型，选填，用来指定是否转置数组，如果为真则转置，默认为False。

·ndmin：整数型，选填，用来指定返回的数组至少包含特定维度的数组，值域为0/1/2，默认为0。

返回：从文件中读取的数组。

示例：在“附件-chapter2”文件夹中有一个名为numpy_data.txt的数据文件，数据为3行5列的矩阵，数据内容如图2-3所示，该示例通过loadtxt方法读取其中的数据。

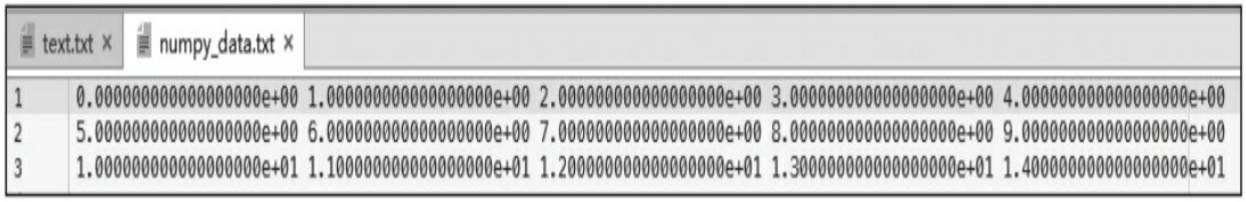


图2-3　示例numpy数据源文件内容

```
import numpy as np  # 导入Numpy库
file_name = 'numpy_data.txt'  # 定义数据文件
data = np.loadtxt(file_name, dtype='float32', delimiter=' ')  # 获取数据
print (data)  # 打印数据
```

上述代码输出的结果如下：

```
[[  0.   1.   2.   3.   4.]
 [  5.   6.   7.   8.   9.]
 [ 10.  11.  12.  13.  14.]]
```

（2）使用load方法读取数据文件

使用Numpy的load方法可以读取Numpy专用的二进制数据文件，从npy、npz或pickled文件加载数组或pickled对象，该文件通常基于Numpy的save或savez等方法产生。

语法：

```
load(file, mmap_mode=None, allow_pickle=True, fix_imports=True, encoding='AS
```

参数：

·file：类文件对象或字符串格式，要读取的文件或字符串，必填，类文件对象需要支持seek（）和read（）方法。

·mmap_mode：内存映射模式，值域为None、'r+'、'r'、'w+'、'c'，选填。

·allow_pickle：布尔型，选填，决定是否允许加载存储在npy文件中的pickled对象数组，默认值为True。

·fix_imports：布尔型，选填，如果为True，pickle将尝试将旧的Python 2名称映射到Python 3中并使用新名称，仅在Python 2生成的pickled文件加载Python 3时才有用，默认值为True。

·encoding：字符串，决定读取Python 2字符串时使用何种编码，选填。

返回：从数据文件中读取的数组、元组、字典等。

示例：我们将在这个示例中先定义一份数据，然后保存为.npy格式的数据文件（该文件也在“附件-chapter2”中，名为load_data.npy），接着使用Numpy的load方法读取并打印输出。代码如下：

```
import numpy as np # 导入nump库
write_data = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])# 定义要存储的数据
np.save('load_data', write_data) # 保存为npy数据文件
read_data = np.load('load_data.npy') #读取npy文件
```

```
print (read_data)# 输出读取的数据
```

上述代码输出的结果如下：

```
[[ 1  2  3  4]
[ 5  6  7  8]
[ 9 10 11 12]]
```

（3）使用fromfile方法读取数据文件

使用Numpy的fromfile方法可以读取简单的文本文件数据以及二进制数据。通常情况下，该方法读取的数据来源于Numpy的tofile方法，即通过Numpy的tofile方法将特定数据保存为文件（默认为二进制数据文件，无论文件扩展名如何定义），然后通过fromfile方法读取该二进制文件。

语法：

```
fromfile(file, dtype=float, count=-1, sep='')
```

参数：

·file：文件或字符串。

·dtype：数据类型，具体参照表2-3。注意数据类型要与文件存储的类型一致。

·count：整数型，读取数据的数量，-1意味着读取所有数据。

·sep：字符串，如果file是一个文本文件，那么该值就是数据间的分隔符。如果为空（""）则意味着file是一个二进制文件，多个空格将按照一个空格处理。

返回：从文件中读取的数据。

示例：我们仍然以“附件-chapter2”文件夹中numpy_data.txt数据文件为例，首先通过tofile方法创建1个二进制文件，然后对该文件进行读

取。

```
import numpy as np # 导入Numpy库
file_name = 'numpy_data.txt' # 定义数据文件
data = np.loadtxt(file_name, dtype='float32', delimiter=' ') # 获取数据
tofile_name = 'binary' # 定义导出二进制文件名
data.tofile(tofile_name) # 导出二进制文件
fromfile_data = np.fromfile(tofile_name, dtype='float32') # 读取二进制文件
print (fromfile_data) # 打印数据
```

上述代码输出的结果如下：

```
[  0.   1.   2.   3.   4.   5.   6.   7.   8.   9.  10.  11.  12.  13.  14.]
```

请务必确保读入文件跟存储文件时的数据类型一致，否则将导致数据报错。有兴趣的读者在使用fromfile导入数据时，可不指定float32格式，看下输出结果。另外，由于使用tofile方法保存的数据会丢失数据形状信息，因此导入时无法重现原始数据矩阵。

3.使用Pandas的read_csv、read_fwf、read_table读取数据

相对于Python默认函数以及Numpy读取文件的方法，Pandas读取数据的方法更加丰富。Pandas读取文本文件数据的常用方法如表2-5所示。

表2-5　Pandas读取数据的常用方法

方　法	描　述	返回数据
read_csv	读取 csv 文件	DataFrame 或 TextParser
read_fwf	读取表格或固定宽度格式的文本行到数据框	DataFrame 或 TextParser
read_table	读取通用分隔符分隔的数据文件到数据框	DataFrame 或 TextParser

（1）使用read_cvs方法读取数据

通过read_csv方法可以读取csv格式的数据文件。

语法：

```
read_csv(filepath_or_buffer, sep=',', delimiter=None, header='infer', names=
```

参数（read_csv的参数众多，以下仅介绍最常用的几个参数）；

·filepath_or_buffer：字符串，要读取的文件对象，必填。

·sep：字符串，分隔符号，选填，默认值为英文逗号','。

·names：类数组，列名，选填，默认值为空。

·skiprows：类字典或整数型，要跳过的行或行数，选填，默认为空。

·nrows：整数型，要读取的前记录总数，选填，默认为空，常用来在大型数据集下做初步探索之用。

·thousands：字符串，千位符符号，选填，默认为空。

·decimal：字符串，小数点符号，选填，默认为点（.），在特定情况下应用，例如欧洲的千位符和小数点跟中国相反，欧洲的“4.321，1”对应着中国的“4，321.1”。

返回：DataFrame或TextParser。

示例：我们以“附件-chapter2”文件夹中csv_data.csv数据文件为例，直接读取文件并打印输出。数据内容如图2-4所示。

```
import pandas as pd  # 导入Pandas库
csv_data          =          pd.read_csv('csv_data.csv',          names=
['col1', 'col2', 'col3', 'col4', 'col5'])  # 读取csv数据
print (csv_data)  # 打印输出数据
```

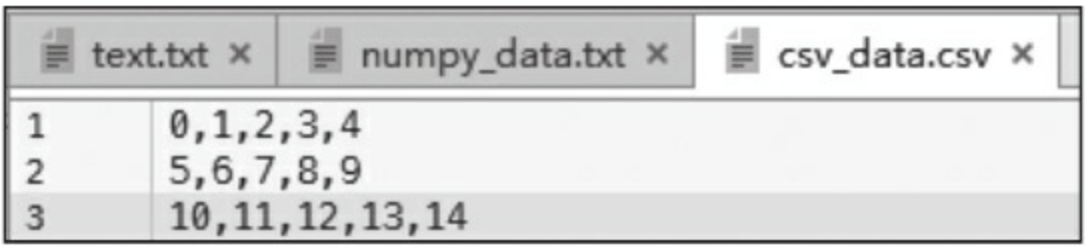


图2-4　数据文件内容

上述代码输出的结果如下：

```
    col1  col2  col3  col4  col5
0     0     1     2     3     4
1     5     6     7     8     9
2    10    11    12    13    14
```

（2）使用read_fwf方法读取数据

通过read_fwf方法可以读取表格或固定宽度格式的文本行到数据框。

语法：

```
read_fwf(filepath_or_buffer, colspecs='infer', widths=None, **kwds)
```

参数：read_fwf跟read_csv一样都具有非常多的参数（只是在语法中前者通过**kwds省略了，查询完整语法请使用help（pd.read_fwf）），并且大多数参数的用法相同。除了read_csv中的常用参数外，以下仅介绍read_fwf特有且常用的参数。

·widths：由整数组成的列表，选填，如果间隔是连续的，可以使用的字段宽度列表而不是“colspecs”。

返回：DataFrame或TextParser。

示例：我们以“附件-chapter2”文件夹中fwf_data数据文件为例，直接读取文件并打印输出。数据内容如图2-5所示。

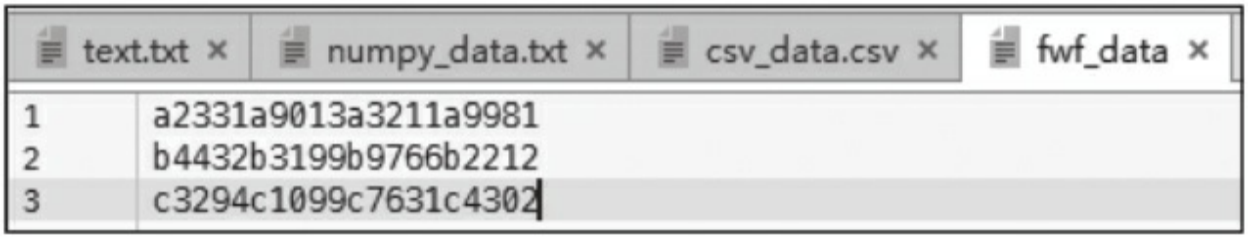


图2-5　数据文件内容

```
import pandas as pd  # 导入Pandas库
fwf_data  =  pd.read_fwf('fwf_data',  widths=[5,  5,  5,  5],  names=
['col1', 'col2', 'col3', 'col4'])  # 读取csv数据
print (fwf_data)  # 打印输出数据
```

上述代码输出的结果如下：

```
    col1   col2   col3   col4
0  a2331  a9013  a3211  a9981
1  b4432  b3199  b9766  b2212
2  c3294  c1099  c7631  c4302
```

（3）使用read_table方法读取数据

通过read_table方法可以读取通用分隔符分隔的数据文件到数据框。

语法：

```
read_table(filepath_or_buffer, sep='\t', delimiter=None, header='infer', nam
```

参数：对于read_table而言，参数与read_csv完全相同。其实read_csv本来就是read_table中分隔符是逗号的一个特例，表现在语法中是read_csv的sep=','（默认）。因此，具体参数请查阅read_csv的参数部分。

返回：DataFrame或TextParser。

我们以“附件-chapter2”文件夹中table_data.txt数据文件为例，直接读取文件并打印输出。数据内容如图2-6所示。

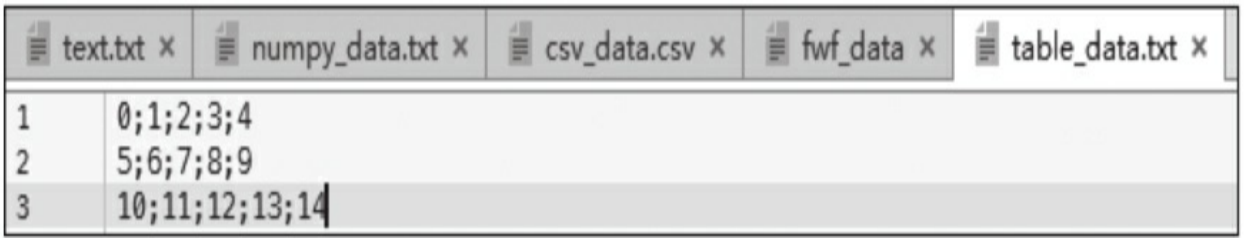


图2-6　数据文件内容

```
import pandas as pd  # 导入Pandas库
table_data     =     pd.read_table('table_data.txt',     sep=';',     names=
['col1', 'col2', 'col3', 'col4', 'col5'])  # 读取csv数据
print (table_data)  # 打印输出数据
```

上述代码输出的结果如下：

```
    col1  col2  col3  col4  col5
0     0     1     2     3     4
1     5     6     7     8     9
2    10    11    12    13    14
```

数据分割（或分列）常用思路分为两种：一种是基于固定宽度，一种是基于分割符号。上述三种方法中，常用的方法是第二和第三种，即read_fwf和read_table方法。

除了上述用于读取文本文件的方法外，Pandas还提供了非常丰富的用于其他场景的数据读取方法，限于篇幅，在此不做更多介绍，仅提供读取列表供有兴趣的读者参考，以及做知识延伸。具体如表2-6所示。

表2-6　Pandas其他数据读取方法

方　法	描　述	返回数据
read_clipboard	读取剪贴板数据，然后将对象传递给 read_table 方法	DataFrame 或 TextParser
read_excel	从 Excel 中读取数据	DataFrame 或 DataFrame 构成的字典
read_gbq	从 Google Bigquery 中读取数据	DataFrame
read_hdf	读取文件中的 Pandas 对象	所选择的数据对象
read_html	读取 HTML 中的表格	由 DataFrame 构成的字典

（续）

方　法	描　述	返回数据
read_json	将 JSON 对象转换为 Pandas 对象	Series 或 DataFrame，具体取决于参数 typ 设置
read_msgpack	从指定文件中加载 msgpack Pandas 对象	文件中的对象类型
read_pickle	从指定文件中加载 pickled Pandas 或其他 pickled 对象	文件中的对象类型
read_sas	读取 XPORT 或 SAS7BDAT 格式的 SAS（统计分析软件）文件	DataFrame 或 SAS7BDATReader 或 XportReader，具体取决于设置
read_sql	读取 SQL 请求或数据库中的表	DataFrame
read_sql_query	从 SQL 请求读取数据	DataFrame
read_sql_table	读取 SQL 数据库中的表	DataFrame
read_stata	读取 Stata（统计分析软件）文件	DataFrame 或 StataReader

4.如何选择最佳读取数据的方法

关于“最佳”方法其实没有固定定义，因此所谓的最佳方法往往跟以下具体因素有关：

·数据源情况：数据源中不同的字段类型首先会制约读取方法的选择，文本、数值、二进制数据都有各自的适应方法约束。

·数据处理目标：读取数据往往是第一步，后续会涉及数据探索、预处理、统计分析等复杂过程，这些复杂过程需要用到哪些方法一定程度上都会受数据源的读取影响，影响最多的点包括格式转换、类型转换、异常值处理、分类汇总等。

·模型数据要求：不同的模型对于数据格式的要求是不同的，应用到实际中不同的工具对于数据的表示方法也有所差异。

·“手感”最好的方法：很多时候，最佳方法往往是对哪个或哪些方法最熟悉，这些所谓的“手感”最好的方法便是最佳方法。

当然，即使使用了不是“最佳”的数据读取方法也不必过于担心，因为Python强大的功能提供了众多可以相互进行转换的方法，从Python存储数据的基本对象类型来看，无非是字符串、列表、字典、元组、数组、矩阵等（当然，不同的库对于这些对象的定义名称可能有所不同，但基本含义相同），不同对象的类型转换都非常容易。但本着少走弯路的原则，在这里笔者还是针对不同的场景提供了较为适合的数据读取方法：

·对于纯文本格式或非格式化、非结构化的数据，通常用于自然语言处理、非结构化文本解析、应用正则表达式等后续应用场景下，Python默认的三种方法更为适合。

·对于结构化的、纯数值型的数据，并且主要用于矩阵计算、数据建模的，使用Numpy的loadtxt方法更为方便，例如本书中使用的sklearn本身就依赖于Numpy。

·对于二进制的数据处理，使用Numpy的load和fromfile方法更为合适。

·对于结构化的、探索性的数据统计和分析场景，使用Pandas方法进行读取效果更佳，因为其提供了类似于R的数据框，可以实现“仿SQL”式的操作方式，对数据进行任意翻转、切片（块等）、关联等都非常方便。

对于结构化的、数值型和文本型组合的数据统计分析场景，使用Pandas更为合适，因为每个数据框中几乎可以装载并处理任意格式的数据。

2.2.2 从Excel获取运营数据

现有的Excel分为两种格式：xls（Excel 97-2003）和xlsx（Excel 2007及以上）。

Python处理Excel文件主要是第三方模块库xlrd、xlwt、pyexcel-xls、xluntils和pyExcel-erator，以及win32com和openpyxl模块，此外Pandas中也带有可以读取Excel文件的模块（read_excel）。

基于扩展知识的目的，我们使用xlrd模块读取Excel数据。

首先安装该库，在系统终端命令行输入命令pip install xlrd。

然后我们以“附件-chapter2”文件夹中demo.xlsx数据文件为例，介绍该库的具体应用。数据概览如图2-7所示。

	A	B	C	D
1	ID_number	Status	Create_Time	Business_City
2	431381198109106573	有效	2016/12/21	深圳市
3	431381198809122734	有效	2016/12/21	深圳市
4	431381197903117478	有效	2016/12/21	深圳市
5	431381197408191515	有效	2016/12/21	深圳市
6	431381197606166011	有效	2016/12/21	深圳市
7	43138119850623339X	有效	2016/12/21	深圳市
8	431381198908223477	有效	2016/12/21	深圳市
9	431381198901176911	有效	2016/12/21	深圳市
10	431381198708275X	有效	2016/12/21	深圳市
11				

图2-7 数据文件内容

```
import xlrd  # 导入库

# 打开文件
xlsx = xlrd.open_workbook('demo.xlsx')
# 查看所有sheet列表
print ('All sheets: %s' % xlsx.sheet_names())
print ('==================================')  # 内容分割线
```

```
# 查看sheet1的数据概况
sheet1 = xlsx.sheets()[0]  # 获得第一张sheet，索引从0开始
sheet1_name = sheet1.name  # 获得名称
sheet1_cols = sheet1.ncols  # 获得列数
sheet1_nrows = sheet1.nrows  # 获得行数
print ('Sheet1 Name: %s\nSheet1 cols: %s\nSheet1 rows: %s') % (sheet1_name,
print ('====================================')  # 内容分割线
# 查看sheet1的特定切片数据
sheet1_nrows4 = sheet1.row_values(4)  # 获得第4行数据
sheet1_cols2 = sheet1.col_values(2)  # 获得第2列数据
cell23 = sheet1.row(2)[3].value  # 查看第3行第4列数据
print ('Row 4: %s\nCol 2: %s\nCell 1: %s\n' % (sheet1_nrows4, sheet1_cols2,
print ('====================================')  # 内容分割线
# 查看sheet1的数据明细
for i in range(sheet1_nrows):  # 逐行打印sheet1数据
    print (sheet1.row_values(i))
```

上述代码中，我们先读取一个Excel文件，再查看所有sheet（工作簿）并输出sheet1相关属性信息；然后查看sheet1中特定数据行、列和元素的信息；最后用循环的方式，依次读取每个数据行并打印输出。

以下是代码执行后打印输出的结果：

```
All sheets: [u'Sheet1']
====================================
Sheet1 Name: Sheet1
Sheet1 cols: 4
Sheet1 rows: 10
====================================
Row 4: [u'431381197408191515', u'\u6709\u6548', 42725.0, u'\u6df1\u5733\u5e0
Col 2: [u'Create_Time', 42725.0, 42725.0, 42725.0, 42725.0, 42725.0, 42725.0
Cell 1: 深圳市
====================================
[u'ID_number', u'Status', u'Create_Time', u'Business_City']
[u'431381198109106573', u'\u6709\u6548', 42725.0, u'\u6df1\u5733\u5e02']
[u'431381198809122734', u'\u6709\u6548', 42725.0, u'\u6df1\u5733\u5e02']
[u'431381197903117478', u'\u6709\u6548', 42725.0, u'\u6df1\u5733\u5e02']
[u'431381197408191515', u'\u6709\u6548', 42725.0, u'\u6df1\u5733\u5e02']
[u'431381197606166011', u'\u6709\u6548', 42725.0, u'\u6df1\u5733\u5e02']
[u'43138119850623339X', u'\u6709\u6548', 42725.0, u'\u6df1\u5733\u5e02']
[u'431381198908223477', u'\u6709\u6548', 42725.0, u'\u6df1\u5733\u5e02']
[u'431381198901176911', u'\u6709\u6548', 42725.0, u'\u6df1\u5733\u5e02']
[u'43138119870827275X', u'\u6709\u6548', 42725.0, u'\u6df1\u5733\u5e02']
```

在上述打印输出的内容中，我们发现第二列、第三列、第四列与原始数据不同。第二列和第四列出现“异常”的原因是将中文编码统一转换为了Unicode编码，便于在不同程序间调用；第三列出现“异常”是由于将日期格式转换为了数值格式。

上述操作只是将数据从Excel中读取出来，基于读取的数据转换为数组便可以进行矩阵计算。由于矩阵计算大多是基于数值型数据实现的，因此上述数据将无法适用于大多数科学计算场景，这点需要注意。

在企业实际场景中，由于Excel本身的限制和适用范围，其无法存储和计算过大（例如千万级的数据记录）的数据量，并且Excel本身也不是为了海量数据的应用而产生的。因此，Excel可用于日常基本数据处理、补充数据来源或者汇总数据，同时也可以作为数据结果展示的载体，这种应用对于大量数值表格的展现效果非常好。

2.2.3　从关系型数据库MySQL读取运营数据

在1.2.4节中我们介绍了安装MySQL和Navicat的方法和关键步骤。这里我们介绍如何利用Python从MySQL中读取数据。

在“附件-chapter2”文件夹中有一个名为order.xlsx的数据文件，我们先把该文件导入MySQL数据库，然后再基于数据库进行相关操作。

第一步　新建一个数据库用于存放要导入的目标数据。

步骤1打开Navicat，左侧导航栏中有我们已经建立好的数据库连接，在第1章中我们已经建立好了本地连接，名为“127.0.0.1”。

步骤2双击该数据库连接，会展示出所有当前连接（对应的用户权限）下可用的数据库，如图2-8所示。

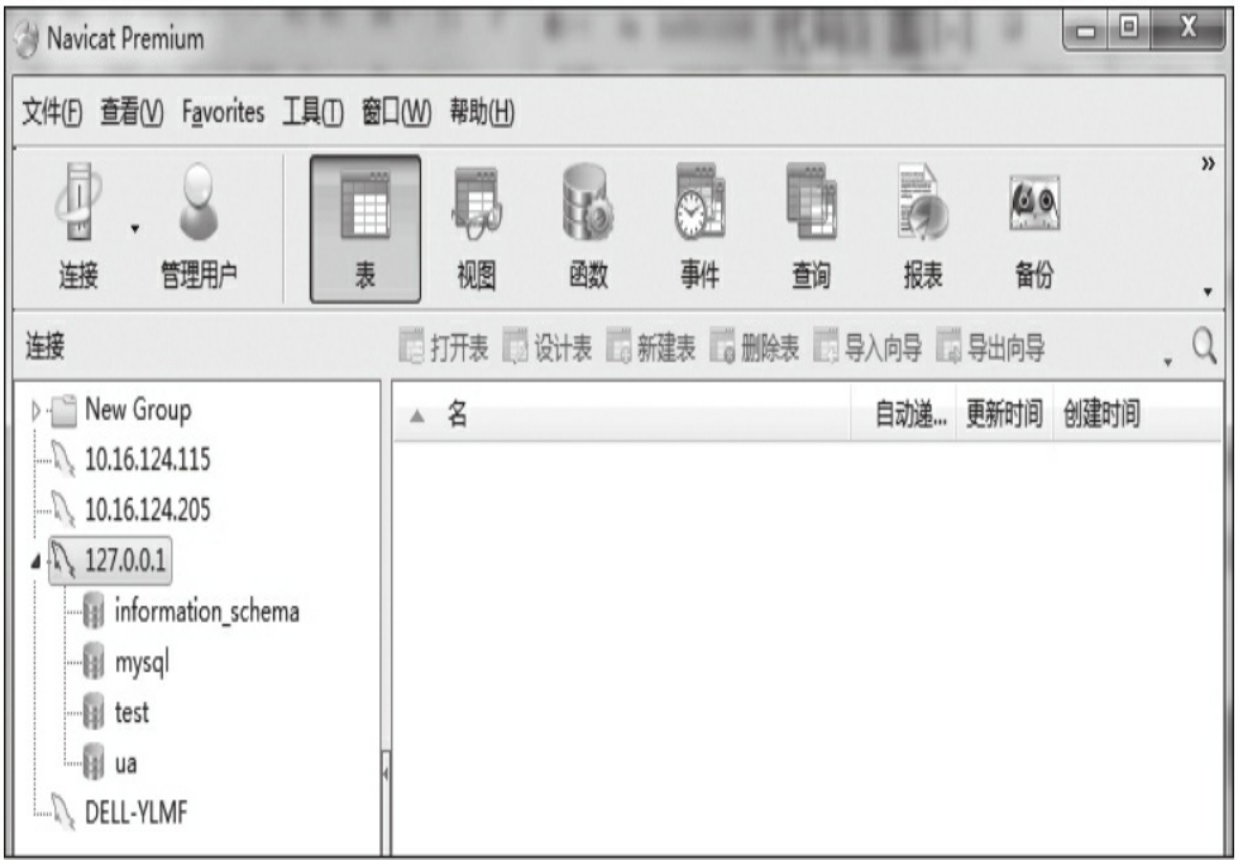


图2-8　127.0.0.1下的数据库

步骤3为了保持不同数据的独立性，我们新建一个数据库，单独存放特定主题的数据。在“127.0.0.1”名称上右击，在弹出的菜单中选择 新建数据库...，然后做如下设置并点击确定。完成之后在左侧的“127.0.0.1”连接下会新增一个刚才新建的数据库“python_data”。

第二步　将Excel数据导入数据库。

双击刚才新建的名为“python_data”的数据库名称，激活数据库连接。此时右侧功能栏中的部分功能已经可用。点击“导入向导”，开始导出Excel数据，如图2-10所示。

步骤1选择数据文件格式： Excel 2007 文件 (*.xlsx)

步骤2选择数据源文件。

步骤3定义附加选项。由于数据文件中第一行是表头，因此数据从第二行开始；日期分隔符与数据文件一致，需要设置为“-”，其他都为默认值。

图2-9　新建数据库设置

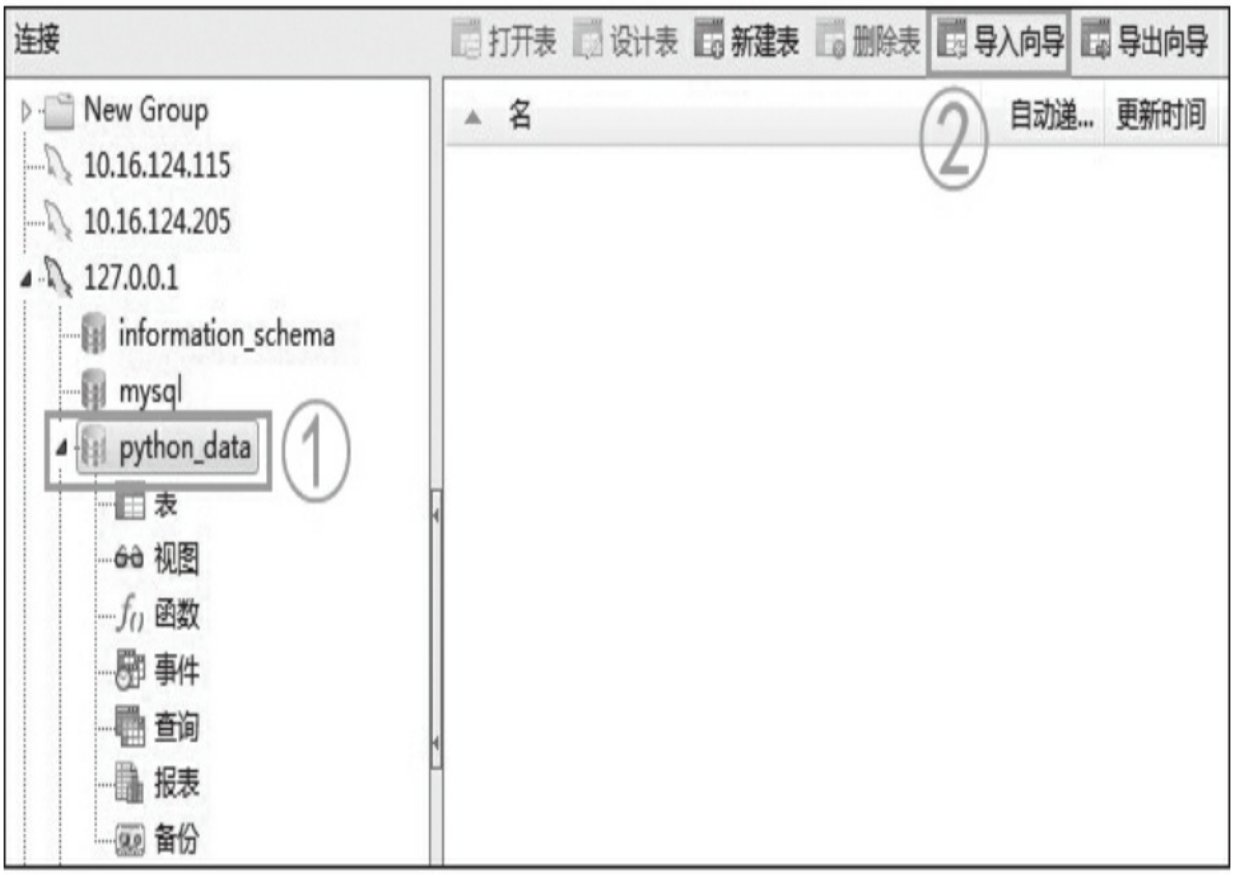


图2-10　创建导入向导

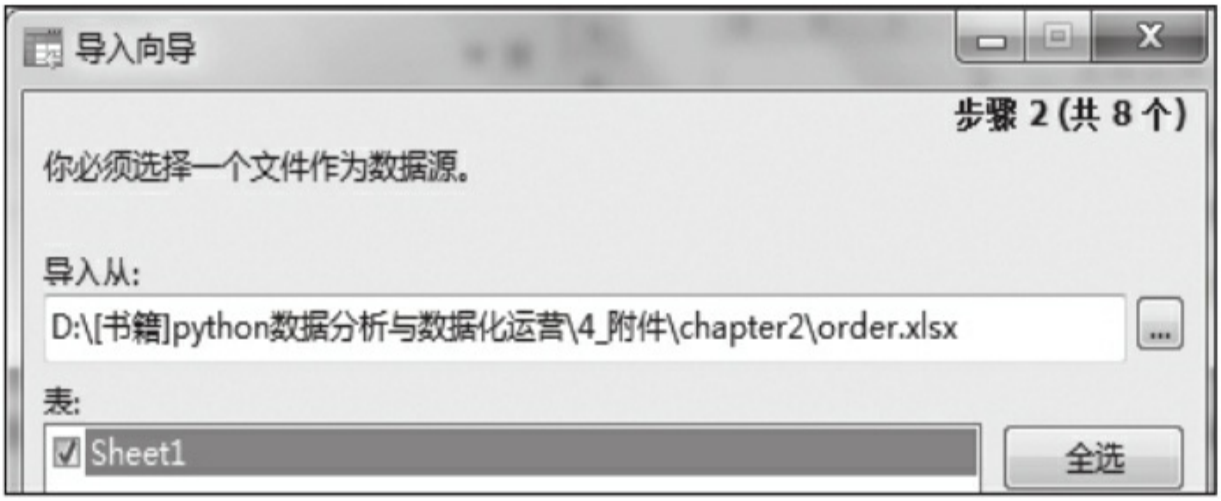


图2-11　设置导入数据源

步骤4设置目标数据表名，此处设置为order。

步骤5设置字段类型、长度、约束等信息。其中主要的设置是将

order_data类型设置为date（日期型），长度为0（不做限制）；
order_tme类型设置为time（时间型），长度为0（不做限制）；
total_amount类型设置为float（浮点型），长度默认为255，如图2-13所示。

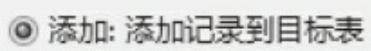
图2-12　设置导入附件选项

	目标栏位	类型	长度	比例
☑	order_id	varchar	255	
☑	order_date	date	0	
☑	order_time	time	0	
☑	status	varchar	255	
☑	user_id	varchar	255	
☑	total_amount	float	255	

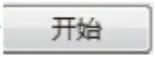
图2-13　设置字段类型等约束

实际上，我们的数据很可能用不了那么多的“存储空间”，因此在设置时存在数据空间冗余，不过没关系，这里我们仅做入门引导之

用。真正涉及数据库表的模型设计有很多学问，这里的设置不会影响我们做数据统计分析之用。

步骤6设置导入模式。由于我们是新创建表，因此设置为 

。

步骤7启动导入任务，点击 

即可。

等待系统导入数据，导入成功后，会在窗口中显示图2-14所示信息，最终的信息中包含“successfully”。点击关闭按钮即可。

至此我们已经成功导入数据，在数据库python_data下新增了一个名为“order”的数据表。我们要初步验证导入的数据是否与原始数据一致。双击打开该表 

。

打开表之后，会弹出信息提示该表没有主键。该信息不影响我们使用该表，点击确定即可，如图2-15所示。

在创建表时，我们完全可以指定主键，例如本案例中的order_id是唯一的，那么可指定该字段作为主键，更可以创建一个自增长ID作为主键。本节的主要内容是介绍如何导入外部数据到数据库。关于数据库建表还有很多话题，在此不进行更多介绍了。

打开order表之后，会显示图2-16所示预览内容。

图2-14　导入成功提示信息

图2-15　提示表没有主键

order @python_data (127.0.0.1)

文件　编辑　查看　窗口　帮助

导入向导　导出向导　筛选向导　网格查看　表单查看　备注　十六进位　图像

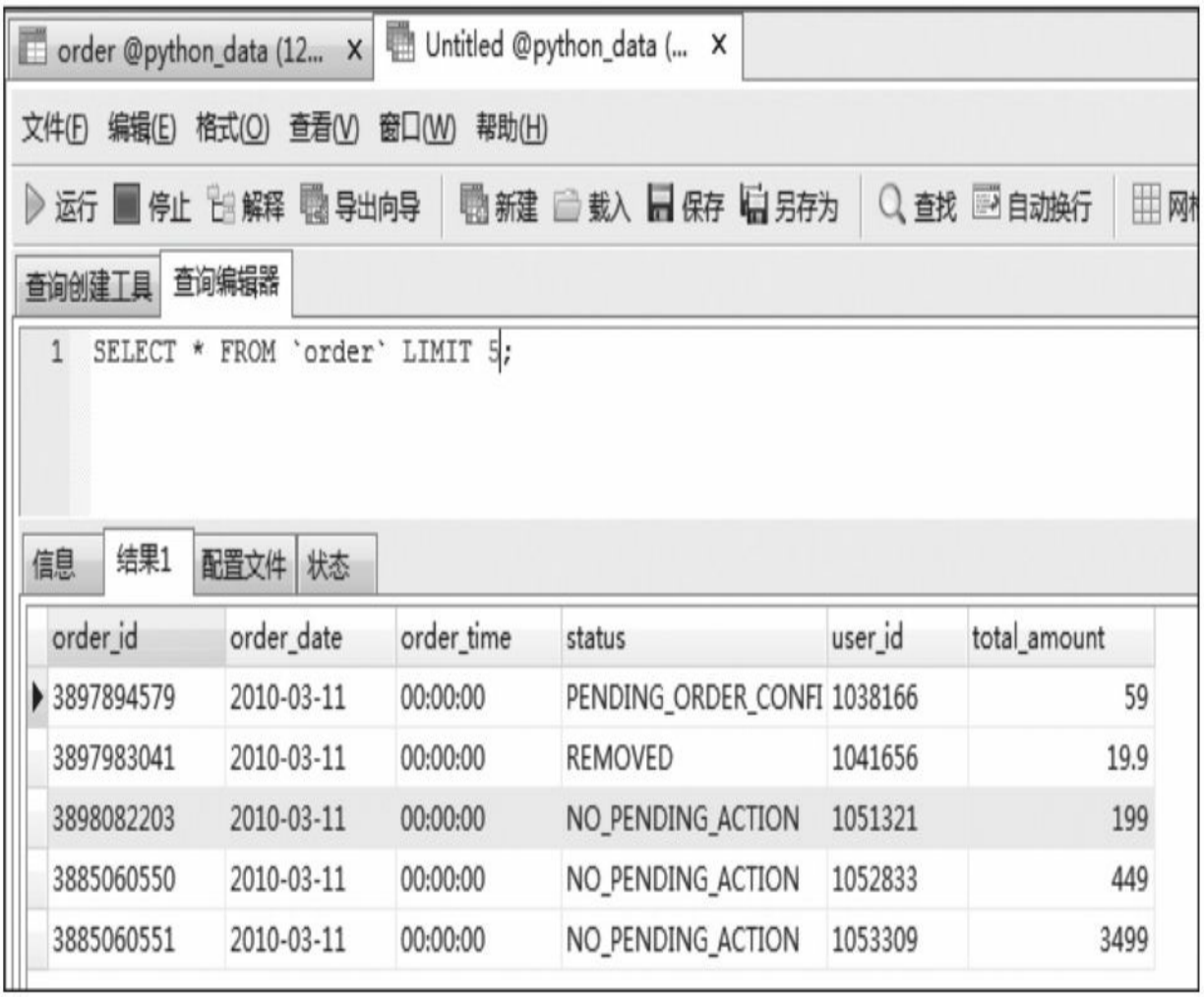
order_id	order_date	order_time	status	user_id	total_amount
3897894579	2010-03-11	00:00:00	PENDING_ORDER_CONFI	1038166	59
3897983041	2010-03-11	00:00:00	REMOVED	1041656	19.9
3898082203	2010-03-11	00:00:00	NO_PENDING_ACTION	1051321	199
3885060550	2010-03-11	00:00:00	NO_PENDING_ACTION	1052833	449
3885060551	2010-03-11	00:00:00	NO_PENDING_ACTION	1053309	3499
3885080549	2010-03-11	00:00:00	REMOVED	1055216	2789
3898206200	2010-03-11	00:00:01	NO_PENDING_ACTION	1036090	59.9
3898042695	2010-03-11	00:00:02	REMOVED	1035236	99
3898265743	2010-03-11	00:00:02	PENDING_ORDER_CONFI	1038613	59
3898176418	2010-03-11	00:00:02	PENDING_ORDER_CONFI	1045960	199
3897899016	2010-03-11	00:00:02	PENDING_ORDER_CONFI	1048950	9

图2-16　数据表概览

在当前窗口，点击“文件→查询表”或直接使用组合键Ctrl+Q，打开SQL窗口，然后输入SELECT*FROM`order`LIMIT 5；点击“运行”或使用

组合键Ctrl+R，返回前5条结果，如图2-17所示。读者可以在此输入其他SQL语句或直接浏览数据表来核对数据导入是否准确。

由于表名order是数据库中的排序关键字，因此在查询表时需要写为`order`，否则会报错。



order_id	order_date	order_time	status	user_id	total_amount
3897894579	2010-03-11	00:00:00	PENDING_ORDER_CONFI	1038166	59
3897983041	2010-03-11	00:00:00	REMOVED	1041656	19.9
3898082203	2010-03-11	00:00:00	NO_PENDING_ACTION	1051321	199
3885060550	2010-03-11	00:00:00	NO_PENDING_ACTION	1052833	449
3885060551	2010-03-11	00:00:00	NO_PENDING_ACTION	1053309	3499

图2-17　运行SQL并返回结果

导入Excel数据主要核对的信息点包括：

·数据记录数是否一致：使用SELECT　COUNT（*）FROM`order`；或在导入成功提示窗口也可以看到）。

·数据字段数量是否一致，导入表中的字段不能多也不能少。

·查看数据内容是否一致，尤其是日期、时间等非字符串字段的数据，很多时候时间型的数据转换会出现0000-00-00，这通常是导入设置的日期分隔符问题。

·数据精准度是否一致，尤其是原始数据是浮点型，对应到数据库字段是否还是浮点型（注意小数位数），问题通常出现在字段类型和位数设置。

·注意字段的最大长度，不同的数值型数据长度位数是不同，例如整数型tinyint、int，浮点型的单精度和双精度浮点等，这对于汇总级别的数据会产生极大影响（因为某些字段的汇总数据可能非常大）。

通过Python连接MySQL需要有Python库来建立连接，本节使用MySQL官方驱动连接程序，更多信息可在1.2.3节中的第5小节中找到。

以下是Python读取数据库数据的基本方法：

```
import mysql.connector  # 导入库

config = {'host': '127.0.0.1',  # 默认127.0.0.1
            'user': 'root',  # 用户名
            'password': '123456',  # 密码
            'port': 3306,  # 端口，默认为3306
            'database': 'python_data',  # 数据库名称
            'charset': 'gb2312'  # 字符编码
            }
cnn = mysql.connector.connect(**config)  # 建立MySQL连接
cursor = cnn.cursor()  # 获得游标
sql = "SELECT * FROM `order`"  # SQL语句
cursor.execute(sql)  # 执行SQL语句
data = cursor.fetchall()  # 通过fetchall方法获得数据
for i in data[:2]:  # 打印输出前2条数据
    print (i)
cursor.close()  # 关闭游标
cnn.close()  # 关闭连接
```

上述代码中，实现了通过Python连接MySQL查询所有的数据，并输出前2条数据的功能。执行结果如下：

```
(u'3897894579', datetime.date(2010, 3, 11), datetime.timedelta(0), u'PENDING
(u'3897983041', datetime.date(2010, 3, 11), datetime.timedelta(0), u'REMOVED
```

在应用MySQL时，除了查询所有数据外，更多时候我们还会应用

更多SQL技巧来帮助我们快速找到目标数据并进行初步处理。以下是常用SQL语法。

1）只查询前N条数据而非全部数据行记录。

示例：只查询前100条数据记录。

```
SELECT * FROM `order` LIMIT 100;
```

LIMIT为限制的数据记录数方法，语法为limit m，n，意思是从第m到第n-1条数据。m可省略，默认值从0开始，上述示例中就是从0开始，取100条数据。例如，要从第11条开始取，取10条可以写为

```
SELECT * FROM `order` LIMIT 11, 10;
```

2）只查询特定列（而非全部列）数据。

示例：只查询total_amount和order_id两列数据。

```
SELECT total_amount, order_id from `order`;
```

选择特定列只需将列名写到表达式中即可，多个列名用英文逗号分隔。另外，还可以使用“表名.字段名”的写法，例如上述表达式可以写成

```
SELECT order.total_amount, order.order_id from `order`;
```

这种写法会出现在表达式中出现多个表的情况下，例如多表关联查询。

3）查询特定列去重后的数据。

示例：查询针对user_id去重后的数据。

```
SELECT DISTINCT user_id FROM `order`;
```

关键词DISTINCT用于返回唯一不同的值，后面接字段名，即要去重的字段。如果后面接多个字段会默认对多个字段同时进行比较，只有多个字段完全相同时才会去重。DISTINCT常用来返回去重后的特定ID，或者与COUNT配合使用，查询特定数据记录的唯一数量。

4）查询带有1个条件的数据。

示例：查询total_amount<100的所有数据。

```
SELECT * FROM `order` WHERE total_amount < 100;
```

WHERE是条件关键字，后面接具体条件。本示例中，由于total_amount为浮点型，因此直接可与数值型比较；如果是字符串型，则比较值需要加引号用来表示一致的字段类型。

5）查询带有多个条件（同时满足）的数据。

示例：查询total_amount<100且status为REMOVED的所有数据。

```
SELECT * FROM `order` WHERE total_amount < 100 and `status` = 'REMOVED';
```

多个条件使用and表示“且”的关系，多个条件必须同时满足。MySQL中字段不区分大小写。由于status也是关键字，因此也需要使用`status`。

6）查询带有多个条件（满足任意一个）的数据。

示例：查询total_amount<100或status为REMOVED的所有数据。

```
SELECT * FROM `order` WHERE total_amount < 100 or `status` = 'REMOVED';
```

多个条件使用or表示“或”的关系，多个条件满足任意一个条件即可。

7）查询特定条件值在某个值域范围内的数据。

示例：查询status的值为REMOVED或NO_PENDING_ACTION或PENDING_ORDER_CONFIRM。

```
SELECT * FROM `order` WHERE `status` in ('REMOVED','NO_PENDING_ACTION','PEND
```

对于特定字段采用“穷举法”列出值域的方法，也可以使用or方法连接多个条件，但使用列表的穷举法会使得表达式更简单。

8）使用正则表达式查询具有复杂条件的数据。

示例：查询user_id以106开头且order_id包含04的所有订单数据。

```
SELECT * FROM `order` WHERE user_id LIKE '103%' and order_id LIKE '%04%';
```

正则表达式通常用在复杂条件中，通过使用关键字LIKE来连接不同的正则语法，LIKE后面接具体的匹配模式，常用的匹配模式包括：

·以特定字符开头：'字符串%'，例如'ABD%'表示以ABD开头的所有匹配；

·以特定字符结尾：'%字符串'，例如'%ABD'表示以ABD结尾的所有匹配；

·包含特定字符：'%字符串%'，例如'%ABD%'表示包含ABD的所有匹配。

关于正则表达式还有更多强大的语法规则，限于篇幅不再展开介绍。

9）将查询到的数据倒叙排列。

示例：将total_amount金额在10～100之间的所有数据按照订单ID倒叙排序。

```
SELECT * FROM `order` WHERE 10 < total_amount < 100 ORDER BY order_id DESC;
```

在查询表达式ORDER　BY的结尾，通过使用DESC来表示倒叙（逆序）排序；省略是默认使用ASC正序（顺序）排序。

10）查询指定列中不包含空值的所有数据。

示例：查询order_id不为空的所有数据。

```
SELECT * FROM `order` WHERE order_id IS NOT NULL;
```

在MySQL中，IS NULL表示为空，IS NOT NULL表示非空。需要注意的是，NULL不代表空字符串或者空格，而是真正意义上的没有任何数据，类似于Python中的None。

上述案例只是展示了如何取数，更多情况下，我们会配合特定函数和方法做数据计算、整合和探索性分析，例如：

·使用MySQL聚合函数求算术平均值、计数、求最大/最小值、做分类汇总等；

·使用MySQL数学函数，用来求绝对值、进行对数计算、求平方根等；

·使用MySQL字符串函数进行字符串分割、组合、截取、匹配、处理等；

·使用MySQL的日期函数进行日期获取、转换、处理等；

·使用MySQL将多个表（2个或2个以上）数据进行关联、匹配和整合。

在Python工作者看来，MySQL的很多工作Python本身也能胜任，为什么还要耗费时间和精力学习如何在MySQL中完成这些数据工作？直接用Python读取数据然后基于Python的相关库进行数据运算岂不更好？

Python的工作强项并不是数据计算，而是其灵活、高效、简易和集成多方的工作方式、效率和效果。MySQL（以及其他关系型数据库）作为成熟的数据存储和集成解决方案，在（关系型）数据本身方面更具

优势，尤其是对于数据结构定义（实体、属性、关系）、关系操作（选择、连接、聚合等）、关系完整性约束（主外键、唯一性等）等方面具有成熟且稳定的应用价值，这对于数据处理至关重要，因此可以说MySQL在结构化数据存储和初步处理工作上比Python更专业。

还有一个更加关键的原因是，如果所有数据工作都由Python完成，那么Python的事务处理在某些情况下一定会面临资源瓶颈、工作效率等问题，届时可能会导致程序崩溃和报错，这将大大降低程序的可靠性以及结果输出的效率，对于海量数据工作更是如此。

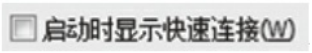
因此，很多时候不是Python不能做，而是在合适的时机选择最合适的工具来完成才是最好的选择。

2.2.4　从非关系型数据库MongoDB读取运营数据

由于MongoDB一般都是企业级数据存储，因此这里我们使用在第1章已经安装过的SSH远程客户端SecureCRT。如果读者在虚拟机或本地安装了MongoDB，也可以使用，操作方法类似。

双击SecureCRT打开程序，点击顶部的“快速连接”按钮，新建连接，如图2-18所示。

图2-18　新建快速连接

然后根据服务器具体配置情况，配置图2-19所示信息，其中主机名和用户名需要单独配置，其他的根据实际情况询问IT或运维管理人员（本案例中都是默认值）。可勾选 

用于以后启动该程序时直接显示快捷入口。

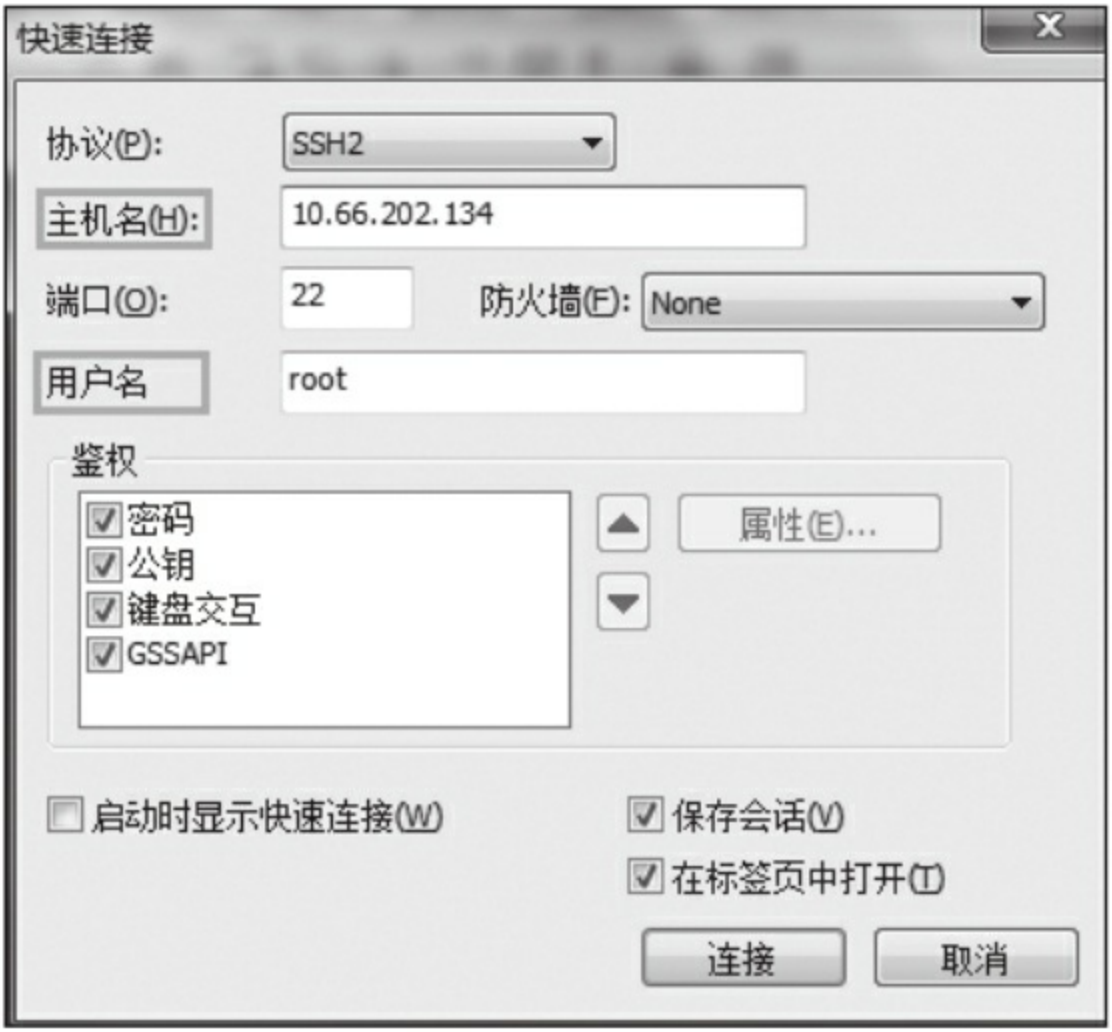


图2-19　填写连接配置信息

点击连接之后，会弹出密码输入窗口，输入密码，可勾选 保存密码(S) 用于以后无须重复输入密码，具体视公司安全规定执行，如图2-20所示。

如果服务器配置信息填写正确，连接服务器成功会显示如图2-21所示。

图2-20　填写密码

图2-21　成功连接服务器

要想通过Python调用MongoDB，需要先安装PyMongoDB，直接使用pip install pymongo即可实现。

注意 不同的服务器环境对于可访问外网的限制有差异，有的服务器可以访问外网，而有的服务器却不能。我们在第1章中介绍的大多数安装方法，默认都是通过pip请求外部网络资源实现的，对于无法直接从服务器连接外网的，请参照1.2.3节中介绍的方法，先将安装包和依赖包下载到本机，然后再复制到服务器上，再通过setup和pip命令安装。

使用Python连接MongoDB之前，需要服务器上的MongoDB处于启

动状态。查看是否成功启动的方法是找一个可以直接连接或访问服务器的浏览器，直接输入“IP:端口”，如果出现“It looks like you are trying to access MongoDB over HTTP on the native driver port.”字样的提示，则说明已经正常启动和运行。例如：笔者的服务器上已经启动了该服务，在浏览器中输入http://10.66.202.134:27017/后提示如图2-22所示信息。

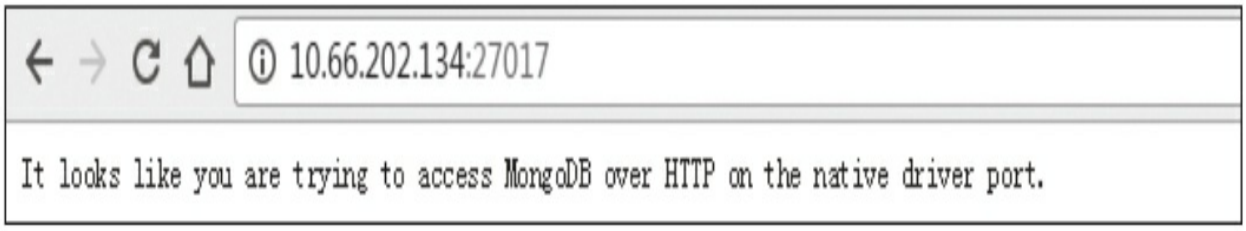


图2-22　MongoDB服务启动后的浏览器返回信息

MongoDB也提供了Windows版本的程序，有兴趣的读者，可登录https://www.mongodb.com/download-center#community下载程序并进行本地部署安装。

要在服务器上进入Python环境，直接在终端窗口输入命令python。在Python命令行窗口中，通过如下方法调用MongoDB数据。

```
from pymongo import MongoClient  # 导入库

client = MongoClient()  # 建立连接
client = MongoClient('10.66.202.134', 27017)  # 环境变量初始化
db = client.test_py  # 选择test_py库
orders = db.ordersets  # 选择orders集合
terms = [{"user": "tony", "id": "31020", "age": "30", "products": ["215120",
04-06"},
          {"user": "lucy", "id": "32210", "age": "29", "products": ["541001"
          "date": "2017-04-06"}]  # 定义一条数据集合用于插入
orders.insert_many(terms)  # 插入数据
print (orders.find_one())  # 获取一文档数据
print ('=======================================')

for i in orders.find():  # 获取所有文档数据并展示
    print (i)
```

上述代码中，我们连接到MongoDB后选择了test_py库下的ordersets集合，然后插入2条数据到集合中，再从集合中查看单独一条以及所有集合数据。

以下是代码执行后打印输出的结果：

```
{u'age': u'30', u'products': [u'215120', u'245101', u'128410'], u'user': u't
04-06', u'_id': ObjectId('58eb1f0b2f76bf26108b4898'), u'id': u'31020'}
========================================
{u'age': u'30', u'products': [u'215120', u'245101', u'128410'], u'user': u't
04-06', u'_id': ObjectId('58eb1f0b2f76bf26108b4898'), u'id': u'31020'}
{u'age': u'29', u'products': [u'541001', u'340740', u'450111'], u'user': u'l
04-06', u'_id': ObjectId('58eb1f0b2f76bf26108b4899'), u'id': u'32210'}
```

除了上述基本查询语句外，PyMongo也提供了类似于MySQL一样的条件过滤。

1）查询特定文档数据。示例，只查询第2条数据。

```
orders.find()[1] # 查询特定索引的数据记录
```

通过增加索引值可以指定查询特定索引的文档数据。注意索引值从0开始，0代表第一条是数据。

2）查询特定范围内的文档数据。示例，值查询第1～2条数据。

```
orders.find()[0:2] # 查询特定范围内的数据记录
```

通过增加索引值可以指定查询特定索引范围的文档数据（切片）。注意索引值为空则默认从0开始，0代表第一条是数据。上述表达式可以写成orders.find（）[:2]。默认不包含索引结束值，例如上述语法中的0:2的实际索引只有0和1。

3）增加查询条件。示例，只查询user为lucy的文档数据。

```
orders.find({"user": "lucy"}) # 所有数据，注意使用迭代方法查看数据
orders.find_one({"user": "lucy"}) # 单条数据
```

作为Key-Value形式的代表应用之一，MongoDB几乎所有的用法都以Key-Vaule的形式存在。过滤条件也不例外，只需在find函数中，以Key-Value的形式设置过滤条件即可，其中Key为过滤维度，Value为对

应值。

4）排序。示例，针对user排序并输出。

```
orders.find({"user": "lucy"}).sort("user") # 所有数据，注意使用迭代方法查看数据
```

通过使用Sort方法设置排序维度。默认是正序，也可以指定为倒叙排列，同时也可以指定多个字段排序规则，例如collection.find（）.sort（[（'field1'，pymongo.ASCENDING），（'field2'，pymongo.DESCENDING）]）。从MongoDB 2.6开始，还提供了按照文本相关性的排序方法，例如collection.find（{'score': {'$meta':'textScore'}}）.sort（[（'score'，{'$meta':'textScore'}）]）。

除了上述基本查询和过滤条件外，PyMongo也提供了用于做数据统计、分析和探索的基本方法，更多信息查阅http://api.mongodb.com/python/current/。

在企业实际应用中，非关系型数据库往往基于“大数据”的场景产生，伴随着海量、实时、多类型等特征。这些数据库通过舍弃了关系型数据库的某些特征和约束，然后在特定方面进行增强，因此才能满足特定应用需求。非关系型数据库由于约束性、规范性、一致性和数据准确性低于关系性数据库，因此常用于实时海量数据读写、非结构化和半结构化信息读写、海量集群扩展、特殊场景应用等。所以在金融、保险、财务、银行等领域内，这种应用比较少；而互联网、移动应用等新兴产业和行业领域则应用较多。

2.2.5 从API获取运营数据

为了更好地让所有读者都能了解从API获取数据的具体过程，本节使用百度免费API作为实际数据来源。百度API提供了众多地图类功能，如基本地图、位置搜索、周边搜索、公交驾车导航、定位服务、地理编码及逆地理编码等。本节使用的是百度Web服务API中的Geocoding API。

Geocoding API用于提供从地址到经纬度坐标或者从经纬度坐标到地址的转换服务，用户可以发送请求且接收JSON、XML的返回数据。该应用可用于对运营数据中的地址相关信息进行解析，从而获得经纬度信息，这些信息可用于进一步基于地理位置进行解析、展示和分析等。

要获得该API，读者需要拥有百度相关账户和AK信息。

第一步 获得百度账户，没有账户的读者可在https://passport.baidu.com/v2/？reg处免费注册获取。

第二步 注册成为百度开放平台开发者，读者可进入http://lbsyun.baidu.com/apicon-sole/key？application=key完成相关注册。该过程非常简单，遵循引导整个过程在5分钟内即可完成，如图2-23所示。

第三步 注册完成之后，会有一个名为“【百度地图开放平台】开发者激活邮件”的验证链接发送到指定（注册时邮箱）邮箱，点击链接即可完成验证，如图2-24所示。

图2-23　提交成功

图2-24　激活验证链接

第四步　点击“申请秘钥”进入创建应用界面，在该应用创建中，我们主要使用Geocoding API v2，其他应用服务根据实际需求勾选。IP白名单区域，如果不做限制，请设置为“0.0.0.0/0”。设置完成后，点击提交，如图2-25所示。

第五步　获得AK秘钥。完成上述步骤之后，会默认跳转到应用列表界面，界面中的“访问应用（AK）”便是该应用的秘钥，如图2-26所示。

1.获取并解析JSON数据

我们先通过Python请求该API来获得JSON格式的数据。本示例的目标是通过给百度API发送一条地理位置数据，返回其经纬度信息。

本节会用到Python第三方库requests，读者需要先通过pip install requests进行安装。完整代码如下：

```
import requests  # 导入库
add = '北京市中关村软件园'  # 定义地址
ak = 'DdOyOKo0VZBgdDFQnyhINKYDGkzBkuQr'  # 创建访问应用时获得的AK
url                =                'http://api.map.baidu.com/geocoder/v2/?
address=%s&output=xml&ak=%s'  # 请求URL
res = requests.get(url % (add, ak))  # 获得返回请求
add_info = res.text  # 返回文本信息
print (add_info)  # 打印输出
```


创建应用
应用名称：
API_For_Python
输入正确
应用类型：
服务端
启用服务：
云存储API
云检索API
Javascript API
Place API v2
Geocoding API v2
IP定位API
路线交通API
静态图API
全景静态图API
坐标转换API
鹰眼API
全景URL API
到达圈
云逆地理编码API
Routematrix API
云地理编码API
时区服务 API
上下车点服务
请求校验方式：
IP白名单校验
IP白名单：
0.0.0.0/0
请输入

图2-25　创建应用

图2-26　获得AK秘钥

在上述代码中，我们先导入一个requests库，该库用来发送网络请求，支持多种发送模式，可以用来做网页内容抓取、自动化网页测试、网页内容采集等。

第二行我们定义了一个地址，该地址通常是运营中的地址类信息，例如通信地址、注册地址、收货地址、联系地址等。

第三行ak是创建访问应用时获得的AK，注意百度每天有6000次的限制，读者可填写自行申请的AK。

第四行定义了一个通过get方法发送的URL请求地址，地址中的address和ak的具体值使用占位符代替，在后面用到时再具体赋值，这种方法经常用到地址、AK有变化的场景中。例如，我们通常会创建多个应用（基于每个账户有限制，以及分开治理应用的考虑），然后从数据库中读取一系列地址用于解析，此时的地址和AK都需动态赋值。

在API请求方法中，最常用的是get和post方法。前者用于向服务器以“明文”（所有参数都在URL中体现）的形式请求数据，常用于普通的页面或服务请求，例如浏览网页、搜索关键字等都是get方法；后者则将“暗语”（所有的数据信息都在HTTP消息中）以键值对的形式

发送，在URL中是看不到具体信息的，常用于数据保密性高的场景，例如登录、注册、订单等表单的处理都是post方法。

第五行通过get方法发送请求到上面定义的URL中，并具体赋值。

第六行将返回对象的文本信息赋值到add_info，返回对象中还包括非常多的相关属性信息，例如状态码、cookie、reason、headers、编码等，这些信息可用于判断是否正确返回、问题原因、状态等信息。

最后一行打印输出返回的文本信息。

以下是代码执行后打印输出的结果：

```
{"status":0,"result":{"location":
{"lng":116.29933765654373,"lat":40.05399730253742},"precise":0,"confidence":
业园区"}}
```

返回结果包括以下字段：

·status：返回结果状态值，成功则返回0。

·location：经纬度坐标，其中lat是纬度值，lng是经度值。

·precise：位置的附加信息，决定是否精确查找。1为精确查找，即准确打点；0为不精确查找，即模糊打点。

·confidence：可信度，描述打点准确度。

·level：百度定义的地址类型。

该结果可以通过JSON进行格式化处理。

```
import json  # 导入库
add_json = json.loads(add_info)  # 加载JSON字符串对象
lat_lng = add_json['result']['location']  # 获得经纬度信息
print (lat_lng)  # 打印输出
```

代码中，我们导入Python自带的JSON库。该库支持Python其他对象

和JSON对象之间的相互转换。

先将获得的地址信息格式转换为JSON字符串（字典类型），然后通过读取字典的Key来获得其Value，由于返回的地址信息中包含字典嵌套，因此这里需要依次读出第一层和第二层信息。读者可通过add_json.items（）来更加直观地观察2层嵌套。

上述代码执行后，返回的结果（字典类型）可进行进一步处理：

```
{u'lat': 40.05399730253742, u'lng': 116.29933765654373}
```

2.获取并解析XML数据

Geocoding API也提供XML格式的返回数据，下面以获得XML格式的数据为例介绍代码过程。

```
import requests  # 导入库
add = '北京市中关村软件园'  # 定义地址
ak = 'DdOyOKo0VZBgdDFQnyhINKYDGkzBkuQr'  # 创建访问应用时获得的AK
url                 =                 'http://api.map.baidu.com/geocoder/v2/?
address=%s&output=xml&ak=%s'  # 请求URL
res = requests.get(url % (add, ak))  # 获得返回请求
add_info = res.text  # 返回文本信息
print (add_info)  # 打印输出
```

上述所有代码与JSON格式的代码完全相同，只在第四行定义URL时，将output的返回值类型设置为XML，执行后返回结果如下：

```
<?xml version="1.0" encoding="utf-8"?>
<GeocoderSearchResponse>
    <status>0</status>
    <result>
        <location>
            <lng>116.299337657</lng>
            <lat>40.0539973025</lat>
        </location>
        <precise>0</precise>
        <confidence>40</confidence>
        <level>工业园区</level>
    </result>
</GeocoderSearchResponse>
```

接着我们通过引入一个XML格式化处理库来从中提取经纬度信

息。关于XML文件的解析，Python默认和第三方的常用库包括xml、libxml2、lxml、xpath等，我们使用Python自带的XML进行处理。

```
# 设置字符编码为utf-8
import sys
reload(sys)
sys.setdefaultencoding('utf-8')
import xml.etree.ElementTree as Etree  # 导入XML中的ElementTree方法
root = Etree.fromstring(add_info)  # 获得XML的根节点
lng = root[1][0][0].text  # 获得lng数据
lat = root[1][0][1].text  # 获得lat数据
print ('lng: %s' % lng)  # 格式化打印输出
print ('lat: %s' % lat)  # 格式化打印输出
```

上述代码中，前4行实现了一个功能，将代码的字符编码设置为utf-8，否则系统会因默认为ASCII，对返回的带有中文字的符串无法识别而报错。

代码第五行导入XML自带的ElementTree方法，该方法可以实现对XML内容的查询、新建、修改等操作，是XML中比较简单容易上手的方法（XML除了ElementTree外，还提供了DOM和SAX方法）。

代码第六行获得从API得到的XML对象并获得根节点。

代码第七、八行分别通过嵌套读取从根节点开始向下的第3层经纬度数据。

代码最后两行格式化打印输出。

上述代码执行后返回结果如下：

```
lng: 116.299337657
lat: 40.0539973025
```

有关百度API的更多信息，具体查阅 http://lbsyun.baidu.com/index.php？title=webapi/guide/webservice-geocoding。

在API应用中，中文的编码处理因细节太多而常常让人头疼。因此，如果可以则应尽量减少直接在API的数据中出现中文字符。在实际

企业应用中，会出现多种API形式，但无论哪种形式，其基本实现思路都是一致的：导入库→定义请求变量→发送请求→获得返回数据→格式化并获得目标数据，因此需要掌握JSON和XML的数据与其他数据的转换方法。

2.3 内容延伸：读取非结构化网页、文本、图像、视频、语音

在前面的章节中，我们介绍的内容是企业常见的数据来源和获取方式，本节将拓展数据来源方式和格式，主要集中在非结构化的网页、文本、图像、视频和语音方面。

2.3.1 从网页中爬取运营数据

要从网页中爬取数据，可使用Python内置标准库或第三方库，例如urllib、urllib2、httplib、httplib2、requests等。本节使用requests方法获取网页数据。

```
import requests # 导入库
url = 'http://www.dataivy.cn/blog/dbscan/' # 定义要抓取的网页地址
res = requests.get(url) # 获得返回请求
html = res.text # 返回文本信息
print (html) # 打印输出网页源代码
```

在代码中，先导入用到的网络请求处理库requests，然后定义一个用来抓取的URL，通过requests的get方法获取URL的返回请求，并通过返回请求的text方法获取内容（源代码），最终打印输出，部分结果如下：

```
<!DOCTYPE html>
<html lang="zh-CN" class="no-js">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width">
    <link rel="profile" href="http://gmpg.org/xfn/11">
......
</body>
</html>
```

从网页中读取的信息其实是网页的源代码，源代码经过浏览器的解析才是我们看到的不同的页面内容和效果。因此，在获取的网页中包含了内容的源代码后，下面要做的是针对源代码的解析。有关该内容会在后面的章节具体说明。

2.3.2 读取非结构化文本数据

非结构化的文本数据指的是文本数据中没有结构化的格式，需要定制化解析才能获取数据，并且每条记录的字段也可能存在差异，这意味着传统的结构化读取方式很难工作。非结构化的日志就是一个典型示例，服务器的日志可由运维工程师自行定义，因此不同公司的日志格式有所不同；另外在网站日志中还可能包含通过页面“埋码”的方式而采集来的用户行为数据，这些都会使日志面临非结构化的解析问题。

在“附件-chapter2”文件夹中有一个名为traffic_log_for_dataivy的日志文件，里面存放了www.dataivy.cn网站一段时间的日志数据，本节示例代码的目的是将日志读取出来。

```
file = 'traffic_log_for_dataivy'
fn = open(file, 'r')  # 打开要读取的日志文件对象
content = fn.readlines()  # 以列表形式读取日志数据
print (content[:2])
fn.close()  # 关闭文件对象
```

上述代码中先定义了一个要读取的非结构化文本文件，然后通过Python标准库open方法以只读模式打开文件，然后通过readlines方法将文件内容按行为单位读取为数据列表，打印输出前2条数据，然后关闭文件对象。执行后，返回如下结果：

```
['120.26.227.125 - - [28/Feb/2017:20:06:51 +0800] "GET / HTTP/1.1" 200 10902
" "curl"\n', '139.129.132.110 - - [28/Feb/2017:20:06:51 +0800] "GET / HTTP/1
" "curl"\n']
```

其实日志文件只是普通文本文件中的一种类型而已，其他的非结构化数据文件都可以以类似的方法读取，即使文件没有任何扩展名。

对于非结构化的文本处理，通常更多地侧重于特定场景，通用性较差，原因就在于非结构化的形式本身变化多样。自然语言理解、文本处理和挖掘、用户日志和机器日志解析等都是该领域中的主要工作。

2.3.3 读取图像数据

Python读取图像通常使用PIL和OpenCV两个库，相对而言，笔者使用后者的情况更多。本节以“附件-chapter2”文件夹中cat.jpg为例进行说明。

1.使用PIL读取图像

Python Imaging Library中包含很多库，常用的是其中的Image，通过使用其中的open方法来读取图像，用法如下：

```
import Image  # 导入库
file = 'cat.jpg'  # 定义图片地址
img = Image.open(file, mode="r")  # 读取文件内容
img.show()  # 展示图像内容
```

其中关键的方法是open，其中的参数包括两个：

·file：文件对象名称，可以是文件名，也可以是图像文件字符串。

·mode：打开模式，默认只能是r模式，否则会报错；当file是图像字符串时，会调用系统的rb模式读取。

图2-27　调用img.show() 展示图像

通过open读取之后会返回一个图像文件对象，后续所有的图像处理都基于该对象进行。上述代码执行后，通过img.show（）会调用系统默认的图像浏览器打开图像并进行查看，如图2-27所示。

该对象包含了很多方法，这些方法可以用来打印输出文件的属性，例如尺寸、格式、色彩模式等。

```
print ('img format: ', img.format)  # 打印图像格式
print ('img size: ', img.size)  # 打印图像尺寸
print ('img mode: ', img.mode)  # 打印图像色彩模式
```

上述代码执行后返回的结果如下：

```
('img format: ', 'JPEG')
('img size: ', (435, 361))
('img mode: ', 'RGB')
```

其中图像的类型是图像本身的格式，例如jpg、gif、png等；图像尺寸是指图像分辨率，示例中的尺寸是435×361（单位是像素）；图像的

模式指的是颜色模式，示例图像是RGB模式。

相关知识点：图像颜色模式

在不同的领域中，图像的色彩模式有多种标准。比较常见的颜色模式包括：

·RGB：自然界中所有的颜色都几乎都可以用红、绿、蓝这三种颜色按不同波长及强度组合得到，这种颜色模式在数字显示领域非常流行。

·CMYK：这是一种工业四色印刷标准，四个字母分别指代青（Cyan）、洋红（Magenta）、黄（Yellow）、黑（Black）。

·HSB：这种模式使用色泽（Hue）、饱和度（Saturation）和亮度（Brightness）来表达颜色的要素，这种模式更多基于人类心理的认识和感觉。

·其他模式：其他模式还包括灰度模式、索引模式、位图模式等，在一定场景下较为常见。

不同的色彩模式之间可以相互转换，例如从RGB模式转换为灰度模式，如图2-28所示。

```
img_gray = img.convert('L')  # 转换为灰度模式
img_gray.show()  # 展示图像
```

除此以外，基于该文件对象也可以进行其他操作，例如图像格式转换、旋转、裁剪、合并、滤波处理、色彩处理、缩略图处理等。限于篇幅，在此不做过多介绍。

2.使用OpenCV读取图像

OpenCV读取和展示图像主要有两类方法：第一种是使用cv库，第二种是使用cv2库。

第一种：使用cv读取图像。

```
import cv2.cv as cv  # 导入库
file = 'cat.jpg'  # 定义图片地址
img = cv.LoadImage(file)  # 加载图像
cv.NamedWindow('a_window', cv.CV_WINDOW_AUTOSIZE)  # 创建一个自适应窗口用于展示图像
cv.ShowImage('a_window', img)  # 展示图像
cv.WaitKey(0)  # 与显示参数配合使用
```

第二种：使用cv2读取图像。

```
import cv2  # 导入库
file = 'cat.jpg'  # 定义图片地址
img = cv2.imread(file)  # 读取图像
cv2.imshow('image', img)  # 展示图像
cv2.waitKey(0)  # 与显示参数配合使用
```

图2-28　灰度图像模式

上述两种方法执行后，都会产生图2-28所示结果。不同的是，图2-28中通过PIL调用的是系统默认的图像显示工具，而在OpenCV中是通过自身创建的图像功能显示图像。

另外，两种方法中都有一个waitKey（）的方法，该方法的作用是键盘绑定函数，其中的参数表示等待毫秒数。执行该方法后，程序将等待特定的毫秒数，看键盘是否有输入，然后返回值对应的ASCII值。如

果其参数为0，则表示无限期等待直到键盘有输入。

笔者通常使用第二种方法读取图像，因为这方法更加简单。其中imread方法细节如下：

语法：

```
cv2.imread(filename[, flags])
```

描述：

读取图像内容，如果图像无法读取则返回空信息，支持的图像格式几乎包括了日常所有场景下的格式，具体包括：

·Windows bitmaps文件：*.bmp、*.dib。

·JPEG文件：*.jpeg、*.jpg、*.jpe。

·JPEG 2000文件：*.jp2。

·PNG文件：*.png。

·WebP文件：*.webp。

·移动图像格式：*.pbm、*.pgm、*.ppm *.pxm、*.pnm。

·Sun rasters文件：*.sr、*.ras。

·TIFF文件：*.tiff、*.tif。

·OpenEXR文件：*.exr。

·Radiance HDR文件：*.hdr、*.pic。

参数：

·filename：必填，字符串，图像地址。

·flags：可选，int型或对应字符串，颜色的读取模式。如果flag>0或者cv2.IMR-EAD_COLOR，读取具有R/G/B三通道的彩色图像；如果flag=0或cv2.IMREAD_GRAYSCALE，读取灰度图像；如果flag<0或cv2.IMREAD_UNCHAN-GED，读取包含Alpha通道的原始图像。

返回：图像内容，如果图像无法读取则返回NULL。

除了使用OpenCV自带的图像展示方法外，OpenCV还经常和Matplotlib配合展示图像，这种场景更加常用。组合使用时可借用Matplotlib的强大图像展示能力进行图像的对比和参照以及不同图像模式的输出。

2.3.4 读取视频数据

Python读取视频最常用的库也是OpenCV。本节以“附件-chapter2”文件夹中Megam-ind.avi视频为例进行说明。如下是一段读取视频内容的代码示例：

```
import cv2  # 导入库

cap = cv2.VideoCapture("tree.avi")  # 获得视频对象
status = cap.isOpened()  # 判断文件是否正确打开

if status:  # 如果正确打开，则获得视频的属性信息
    frame_width = cap.get(3)  # 获得帧宽度
    frame_height = cap.get(4)  # 获得帧高度
    frame_count = cap.get(7)  # 获得总帧数
    frame_fps = cap.get(5)  # 获得帧速率
    print ('frame width: ', frame_width)  # 打印输出
    print ('frame height: ', frame_height)  # 打印输出
    print ('frame count: ', frame_count)  # 打印输出
    print ('frame fps: ', frame_fps)  # 打印输出

success, frame = cap.read()  # 读取视频第一帧
while success:  # 如果读取状态为True
    cv2.imshow('vidoe frame', frame)  # 展示帧图像
    success, frame = cap.read()  # 获取下一帧
     k = cv2.waitKey(1000 / int(frame_fps))  # 每次帧播放延迟一定时间，同时等待输入指令
    if k == 27:  # 如果等待期间检测到按键ESC
        break  # 退出循环
cv2.destroyAllWindows()  # 关闭所有窗口
cap.release()  # 释放视频文件对象
```

上述代码分为4个部分，以空行分隔。

第一部分为前3行，先导入库，然后读取视频文件并获得视频对象，最后获得视频读取状态。其中的关键方法是VideoCapture，用来读取图像。

语法：

```
cv2.VideoCapture(VideoCapture ID|filename|apiPreference)
```

描述：读取视频设备或文件，并创建一个视频对象实例

参数：

·VideoCapture　ID：必填，int型，系统分配的设备对象的ID，默认的设备对象的ID为0。

·Filename：必填。包括如下部分。

·视频文件的名称，字符串，例如abc.avi。目前版本下只支持avi格式。

·序列图像，字符串，例如img_%2d.jpg（图像序列包括img_00.jpg，img_01.jpg，img_02.jpg，...）。

·视频URL地址，字符串，例如protocol://host:port/script_name?script_params|auth。

·apiPreference为int型，后台使用的API。

返回：一个视频对象实例。

第二部分为if循环体内的9行代码，该代码主要用来在判断文件被正确读取的情况下，输出视频文件的整体信息。除了代码中get方法使用的参数值外，OpenCV还支持更多图像属性，如表2-7所示。

表2-7　get方法支持的图像属性

值	属　性	描　述
0	CV_CAP_PROP_POS_MSEC	当前位置（单位为 ms）
1	CV_CAP_PROP_POS_FRAMES	当前位置（单位为帧数，从 0 开始计）
2	CV_CAP_PROP_POS_AVI_RATIO	当前位置（单位为比率，0 表示开始，1 表示结尾）
3	CV_CAP_PROP_FRAME_WIDTH	帧宽度
4	CV_CAP_PROP_FRAME_HEIGHT	帧高度
5	CV_CAP_PROP_FPS	帧速率
6	CV_CAP_PROP_FOURCC	4 字符表示的视频编码（如：M、J、P、G）
7	CV_CAP_PROP_FRAME_COUNT	总帧数
8	CV_CAP_PROP_FORMAT	retrieve(). 调用返回的矩阵格式
9	CV_CAP_PROP_MODE	后端变量指示的当前捕获的模式
10	CV_CAP_PROP_BRIGHTNESS	明亮度（仅用于摄像头）
11	CV_CAP_PROP_CONTRAST	对比度（仅用于摄像头）

（续）

值	属　性	描　述
12	CV_CAP_PROP_SATURATION	饱和度（仅用于摄像头）
13	CV_CAP_PROP_HUE	色调（仅用于摄像头）
14	CV_CAP_PROP_GAIN	增益（仅用于摄像头）
15	CV_CAP_PROP_EXPOSURE	曝光度（仅用于摄像头）
16	CV_CAP_PROP_CONVERT_RGB	是否应该将图像转化为 RGB 图像（布尔值）
17	CV_CAP_PROP_WHITE_BALANCE	白平衡（暂不支持 v2.4.3）

第三部分为具体读取和展示视频的每一帧内容。首先读取视频的第一帧，如果状态为True，则展示图像并读取下一帧，期间通过cv2.waitKey参数做图像延迟控制，同时延迟期间等待系统输入指定；如果输入ESC则退出循环读取帧内容。

相关知识点：动态图像如何产生

我们视觉上看到的视频（或动态图）在计算机中其实是不存在的，计算机中存储的是一幅一幅的图像，在视频里面被称为帧，一帧对应的就是一幅图像。当图像连续播放的速度超过一定阈值间时，由于人类的视觉具有暂留特性（延迟效应），多个暂留图像的叠加便形成了我们看到的动态图像。一般情况下，如果一秒播放超过16帧时，我们就会认为这是一幅动态图像。

在视频中有几个关键名词：

·帧率（FPS）：每秒播放的帧数被定义为帧率，帧率越高，在视觉上认为图像越连贯，就越没有卡顿的现象。常见的帧率包括23.967（电影）、25（PAL电视），示例图像大约为15。帧率与图像清晰度无关，它只是决定了视频的连贯性。

·帧分辨率：帧分辨率基本决定了视频的清晰度（当然除此之外还有视频处理效果、设备播放差异等，这里指的是同等条件下的视频源）。在同样大小的图像中，分辨率越高图像通常就会越清晰。所以形容视频时提到的1080P（1920*1080）、720P（1280*720）其实指的就是分辨率标准。

OpenCV中的图像读取和处理，其实是不包括语音部分的，但从视频文件的组成来讲通常包括序列帧和语音两部分。目前的方式通常是对两部分分开处理。

第四部分为当所有操作结束后，删除所有由OpenCv创建的窗体，释放视频文件对象。

有关OpenCV的更多信息可查阅opencv.org。

2.3.5 读取语音数据

对于语音文件的读取，可以使用Python的audioop、aifc、wav等库实现。但针对语音处理这一细分领域，当前市场上已经具备非常成熟的解决方案，例如科大讯飞、百度语音等，大多数情况下，我们会通过调用其API实现语音分析处理，或者作为分析处理前的预处理。

在具体实现过程中，既可以直接下载SDK做离线应用，也可以使用在线的服务。图2-29所示为科大讯飞的语音服务。

本节将以百度语音API服务应用为例，说明如何通过请求百度语音的API，将语音数据转换为文字信息。

在正式应用百度语音API之前，请先参照2.2.5节中介绍的步骤，建立百度账户以及注册成为百度开发者。基于该条件下，我们继续开通语音识别服务。具体方法如下：

1）进入http://yuyin.baidu.com/app，在弹出的界面中点击要针对哪个应用开通语音识别服务。我们默认使用之前建立的API_For_Python应用。因此，点击该应用的“开通服务”，如图2-30所示。

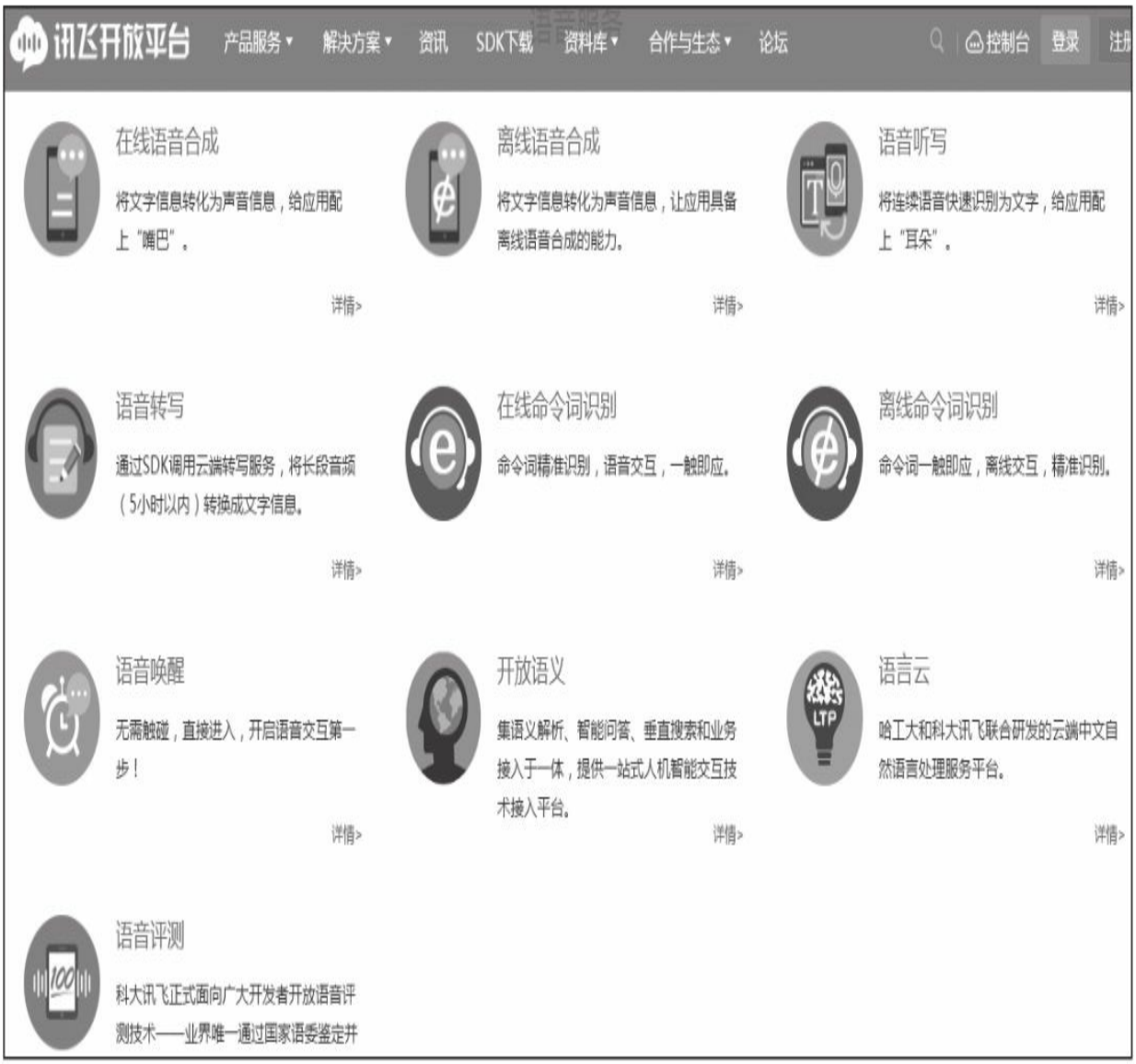


图2-29　科大讯飞语音服务

图2-30　开通服务

2）在弹出的窗口中，点击选择“语音识别”并确定，如图2-31所示。

3）开通成功后系统会提示开通成功，然后点击图2-34中右侧的，会弹出图2-32所示信息。

图2-31　选择开通语音识别服务

上述弹出中的API　Key和Secret　Key为后续语音识别中要使用的信息。

以下为完整代码：

```
# 导入库
import json  # 用来转换JSON字符串
import base64  # 用来做语音文件的Base64编码
import requests  # 用来发送服务器请求

# 获得token
API_Key = 'DdOyOKo0VZBgdDFQnyhINKYDGkzBkuQr'  # 从申请应用的key信息中获得
Secret_Key = 'oiIboc5uLLUmUMPws3m0LUwb00HQidPx'  # 从申请应用的key信息中获得
token_url           =           "https://openapi.baidu.com/oauth/2.0/token?
grant_type=client_credentials&client_id=%s&client_secret=%s"  # 获得token的地
址
res = requests.get(token_url % (API_Key, Secret_Key))  # 发送请求
res_text = res.text  # 获得请求中的文字信息
token = json.loads(res_text)['access_token']  # 提取token信息

# 定义要发送的语音
voice_file = 'baidu_voice_test.pcm'  # 要识别的语音文件
voice_fn = open(voice_file, 'rb')  # 以二进制的方式打开文件
org_voice_data = voice_fn.read()  # 读取文件内容
org_voice_len = len(org_voice_data)  # 获得文件长度
base64_voice_data = base64.b64encode(org_voice_data)  # 将语音内容转换为base64
编码格式

# 发送信息
# 定义要发送的数据主体信息
headers = {'content-type': 'application/json'}  # 定义header信息
payload = {
    "format": "pcm",  # 以具体要识别的语音扩展名为准
    "rate": 8000,  # 支持8000或16000两种采样率
    "channel": 1,  # 固定值，单声道
    "token": token,  # 上述获取的token
    "cuid": "B8-76-3F-41-3E-2B",  # 本机的MAC地址或设备唯一识别标志
    "len": org_voice_len,  # 上述获取的原始文件内容长度
    "speech": base64_voice_data  # 转码后的语音数据
}
data = json.dumps(payload)  # 将数据转换为JSON格式
vop_url = 'http://vop.baidu.com/server_api'  # 语音识别的API
voice_res = requests.post(vop_url, data=data, headers=headers)  # 发送语音识
别请求
api_data = voice_res.text  # 获得语音识别文字返回结果
text_data = json.loads(api_data)['result']
print (api_data)  # 打印输出整体返回结果
print (text_data)  # 打印输出语音识别的文字
```

图2-32　应用key 信息

代码以空行作为分隔，包括4个部分：

第一部分为导入需要的库信息，具体用途见代码注解。

第二部分为获得要使用百度语音识别API的token信息。其中的API_Key和Secret_Key从图2-32所示界面中获得。token_url通过占位符定义出完整字符串，并在请求时发送具体变量数据，从返回的信息中直接读取token便于下面应用中使用。有关获取token的更多信息，具体查阅http://yuyin.baidu.com/docs/asr/56。

在请求获取token时，可使用get或post（推荐使用）两种方法，token的有效期默认为1个月，如果过期需要重新申请。

第三部分主要用于获取和处理语音文件数据。通过最常见的open方法以二进制的方式读取语音数据，然后从获得的语音数据中获取原始数据长度并将原始数据转换为base64编码格式。

百度语音识别API对于要识别的音频源是有要求的：原始PCM的录音参数必须符合8K/16K采样率、16bit位深、单声道，支持的压缩格式有pcm（不压缩）、wav、opus、amr、x-flac。

第四部分为本节内容的主体，发送请求获取语音识别结果。本段落中先定义了发送头信息；然后定义了一个字典，用于存储要发送的Key-Value字符串并将其转换为JSON格式；接着通过post方法以隐示发送的方式进行上传并获得返回结果，最后输出返回结果和其中的语音转文字的信息。该部分内容的细节比较多，具体参见百度语音API开发说明http://yuyin.baidu.com/docs/asr/57。

关于cuid的获取，由于笔者是在本地电脑上测试的，因此使用的是MAC地址。获取MAC地址的方法是：打开系统终端命令行窗口（Win+R，输入cmd并回车），在命令行中输入命令ipconfig/all，在列出的所有连接中找到其中媒体状态不是“媒体已断开”并且属于当前连接的物理地址信息。图2-33所示为笔者电脑MAC信息。

有关语音服务的更多信息可查阅http://www.xfyun.cn/。

上述代码执行后返回如下结果：

```
{"corpus_no":"6409809149574448654","err_msg":"success.","err_no":0,"result":
["百度语音提供技术支持，"],"sn":"83327679891492399988"}
[u'\u767e\u5ea6\u8bed\u97f3\u63d0\u4f9b\u6280\u672f\u652f\u6301\uff0c']
```

系统成功返回的是识别结果，录音的内容是“百度语音提供技术支持”，第二段编码是unicode编码格式的中文。

上述语音识别仅提供了关于语音转为文字的方法，其实语音本身包括非常多的信息，除了相对浅层的生理和物理特征，例如语速、音调、音长、音色、音强等外，还包括更深层次的社会属性，这部分内容需要自然语音理解的深层次应用。目前的语音数据读取后主要应用方向包括：

图2-33　获取MAC地址信息

·语音转文字。这也是广义上语音识别的一种，直接将语音信息转为文字信息，例如微信中就有这个小功能。

·语音识别。语音识别指的是对说话者通过选取语音识别单元、提取语音特征参数、模型训练、模型匹配等阶段实现其角色识别和个体识别的过程，例如通过某段语音识别出是哪个人说的话。

·语音语义理解。在语音识别的基础上，需要对语义特征进行分

析，目的是通过计算得到与语音对应的潜在知识或意图，然后提供对应的响应内容或方法。语音识别和语音理解的差异之处在于，语音识别重在确定语音表达的字面含义，属于表层意义；而语音理解重在挖掘语音的背后含义，属于深层意义。

·语音合成。语音合成就是让计算机能够“开口说话”，这是一种拟人的技术方法。语音合成，又称文本转语音（Text to Speech）技术，它通过机械的、电子的方法将文字信息转变为人类可以听得懂的语音。

·应用集成。经过分析、识别后的信息可以与硬件集成，直接通过语音发送指令。例如通过跟Siri的“沟通”，除了可以进行日常交流外，它还可以告诉你天气情况、帮你设置系统日程、介绍餐厅等。这是智能机器人在模式识别方面的典型应用。

基于上述的复杂应用场景，通常语音后续分析、处理和建模等过程都无法由数据工程师单独完成，还需要大量的语料库素材、社会学、信号工程、语言语法、语音学、自然语音处理、机器学习、知识搜索、知识处理等交叉学科和相关领域人员配合才有可能解开其中的密码。

2.4 本章小结

内容小结：本章的内容较多，主要涉及企业数据化运营可能产生数据的方方面面，包括数据来源的类型、通过不同方式获得运营数据以及对非结构化数据的获取等方面。不同的企业由于其行业和企业背景不同，通常不会全部覆盖其中的所有数据场景，读者可根据自身情况和需求选择。另外，大多数读者所在的企业，应该以结构化的数据为主，内容延伸中的知识作为课外补充和了解即可，真正需要用到这些知识时，再学习和查阅。本书所有示例中的原始代码，在“附件-chapter2”中chapter2_code.py中可以找到，同时在该文件夹下存储了所有示例用到的本地数据。

重点知识：本章需要读者重点掌握2.2节所讲的内容，这里介绍了数据工作者常用的数据来源，其中的从文本文件读取运营数据、从关系型数据库MySQL读取数据最为常用。

外部参考：由于数据来源的获取与数据生产、采集、存储、处理和挖掘工具息息相关，不同的系统和工具之间需要了解更多才有可能更好地利用数据。以下工具或知识是本书以及很多企业中都会用到的，希望读者能进行更深入了解：

·Google BigQuery：作为谷歌在线服务的主要武器之一，BigQuery可以作为云服务应用，也可以作为Google Analytics Premium的细粒度流量数据的获取来源。尤其是基于谷歌广泛的服务体系和应用体系，可以将所有的资源打通，包括数据和服务。

·SAS：做数据挖掘的读者一般都会知道这个工具，SAS是数据挖掘和商业智能领域最为权威和流行的商用工具之一。该工具在很多大型企业内部都有应用，例如国家信息中心、国家统计局、卫生部、中国科学院等，其专业能力可见一斑。

·SQL：作为关系型数据库应用的核心，常用的查询语法需要数据工作者掌握；除了在关系型数据库外，SQL也可以应用到HIVE等大数据工作处理领域，这种通用性（当然语法需要做适当修改）使得SQL几乎在各个企业都有用武之地。

·正则表达式：本书在多个应用示例中都用到了正则表达式，尤其对于非结构化数据，正则表达式几乎是标配知识。

应用实践：本章的内容属于数据工作的第一步，因此希望读者能熟悉不同的数据接入和读取方法。每种数据来源类型建议都逐一进行实践，然后集中精力到现有工作或学习环境中，以达到熟练掌握获取不同数据来源的方法和技巧。

第3章　11条数据化运营不得不知道的数据预处理经验

数据预处理是数据化运营过程中的重要环节，它直接决定了后期所有数据工作的质量和价值输出。从数据预处理的主要内容看，包括数据清洗、转换、归约、聚合、抽样等。本章将摒弃理论和方法说教，直接介绍内容本身可能遇到的问题及应对方法。

3.1 数据清洗：缺失值、异常值和重复值的处理

在数据清洗过程中，主要处理的是缺失值、异常值和重复值。所谓清洗，是对数据集进行丢弃、填充、替换、去重等操作，实现去除异常、纠正错误、补足缺失的目的。

3.1.1 数据列缺失的4种处理方法

数据缺失分为两种：一是行记录的缺失，这种情况又称数据记录丢失；二是数据列值的缺失，即由于各种原因导致的数据记录中某些列的值空缺，不同的数据存储和环境中对于缺失值的表示结果也不同，例如，数据库中是Null，Python返回对象是None，Pandas或Numpy中是NaN。

在极少数情况下，部分缺失值也会使用空字符串来代替，但空字符串绝对不同于缺失值。从对象的实体来看，空字符串是有实体的，实体为字符串类型；而缺失值其实没有实体的，即没有数据类型。

丢失的数据记录通常无法找回，这里重点讨论数据列类型缺失值的处理，通常有四种思路：

1.丢弃

这种方法简单明了，直接删除带有缺失值的行记录（整行删除）或者列字段（整列删除），减少缺失数据记录对总体数据的影响。但丢弃意味着会消减数据特征，以下任意一种场景都不宜采用该方法：

·数据集总体中存在大量的数据记录不完整情况且比例较大，例如超过10%，删除这些带有缺失值的记录意味着将会损失过多有用信息。

·带有缺失值的数据记录大量存在着明显的数据分布规律或特征，例如带有缺失值的数据记录的目标标签（即分类中的Label变量）主要集中于某一类或几类，如果删除这些数据记录将使对应分类的数据样本丢失大量特征信息，导致模型过拟合或分类不准确。

2.补全

相对丢弃而言，补全是更加常用的缺失值处理方式，通过一定的方法将缺失的数据补上，从而形成完整的数据记录对于后续的数据处理、分析和建模至关重要。常用的补全方法包括：

·统计法：对于数值型的数据，使用均值、加权均值、中位数等方法补足；对于分类型数据，使用类别众数最多的值补足。

·模型法：更多时候我们会基于已有的其他字段，将缺失字段作为目标变量进行预测，从而得到最为可能的补全值。如果带有缺失值的列是数值变量，采用回归模型补全；如果是分类变量，则采用分类模型补全。

·专家补全：对于少量且具有重要意义的数据记录，专家补足也是非常重要的一种途径。

·其他方法：例如随机法、特殊值法、多重填补等。

3.真值转换法

某些情况下，我们可能无法得知缺失值的分布规律，并且无法对于缺失值采用上述任何一种方法做处理；或者我们认为数据缺失也是一种规律，不应该轻易对缺失值随意处理，那么还有一种缺失值处理思路——真值转换。

该思路的根本观点是，我们承认缺失值的存在，并且把数据缺失也作为数据分布规律的一部分，这将变量的实际值和缺失值都作为输入维度参与后续数据处理和模型计算。但是变量的实际值可以作为变量值参与模型计算，而缺失值通常无法参与运算，因此需要对缺失值进行真值转换。

以用户性别字段为例，很多数据库集都无法对会员的性别进行补足，但又舍不得将其丢弃，那么我们将选择将其中的值，包括男、女、未知从一个变量的多个值分布状态转换为多个变量的真值分布状态。

·转换前：性别（值域：男、女、未知）。

·转换后：性别_男（值域1或0）、性别_女（值域1或0）、性别_未知（值域1或0）。

然后将这3列新的字段作为输入维度用以替换原来的1个字段参与后续模型计算。有关真值转换的具体方法和知识话题，会在3.2节具体介

绍。

4.不处理

在数据预处理阶段，对于具有缺失值的数据记录不做任何处理，也是一种思路。这种思路主要看后期的数据分析和建模应用，很多模型对于缺失值有容忍度或灵活的处理方法，因此在预处理阶段可以不做处理。常见的能够自动处理缺失值的模型包括：KNN、决策树和随机森林、神经网络和朴素贝叶斯、DBSCAN（基于密度的带有噪声的空间聚类）等。这些模型对于缺失值的处理思路是：

·忽略，缺失值不参与距离计算，例如KNN。

·将缺失值作为分布的一种状态，并参与到建模过程，例如各种决策树及其变体。

·不基于距离做计算，因此基于值的距离做计算，本身的影响就消除，例如DBSCAN。

在数据建模前的数据归约阶段，有一种归约的思路是降维，降维中又有一种直接选择特征的方法。假如我们通过一定方法确定带有缺失值（无论缺少字段的值缺失数量有多少）的字段对于模型的影响非常小，那么我们根本就不需要对缺失值进行处理。因此，后期建模时对字段或特征的重要性判断也是决定是否处理字段缺失值的重要参考因素之一。

对于缺失值的处理思路是先通过一定方法找到缺失值，接着分析缺失值在整体样本中的分布占比以及缺失值是否具有显著的无规律分布特征，然后考虑后续要使用的模型中是否能满足缺失值的自动处理，最后决定采用哪种缺失值处理方法。在选择处理方法时，注意投入的时间、精力和产出价值，毕竟，处理缺失值只是整个数据工作的冰山一角而已。

在数据采集时，可在采集端针对各个字段设置一个默认值。以MySQL为例，在设计数据库表时，可通过default指定每个字段的

默认值，该值必须是常数。在这种情况下，假如原本数据采集时没有采集到数据，字段的值应该为Null，但由于在建立库表时设置了默认值，这会导致“缺失值”看起来非常正常，但本质上还是缺失的。对于这类数据需要尤其注意。

3.1.2 不要轻易抛弃异常数据

异常数据是数据分布的常态，处于特定分布区域或范围之外的数据通常会被定义为异常或“噪音”。产生数据“噪音”的原因很多，例如业务运营操作、数据采集问题、数据同步问题等。对异常数据进行处理前，需要先辨别出到底哪些是真正的数据异常。从数据异常的状态看分为两种：

·一种是“伪异常”，这些异常是由于业务特定运营动作产生，其实是正常反映业务状态，而不是数据本身的异常规律。

·一种是“真异常”，这些异常并不是由于特定的业务动作引起的，而是客观地反映了数据本身分布异常的个案。

大多数数据挖掘或数据工作中，异常值都会在数据的预处理过程中被认为是噪音而剔除，以避免其对总体数据评估和分析挖掘的影响。但在以下几种情况下，无须对异常值做抛弃处理。

1.异常值正常反映了业务运营结果

该场景是由业务部门的特定动作导致的数据分布异常，如果抛弃异常值将导致无法正确反馈业务结果。

例如：公司的A商品正常情况下日销量为1000台左右。由于昨日举行优惠促销活动导致总销量达到10000台，由于后端库存备货不足导致今日销量又下降到100台。在这种情况下，10000台和100台都正确反映了业务运营的结果，而非数据异常。

2.异常检测模型

异常检测模型是针对整体样本中的异常数据进行分析和挖掘以便找到其中的异常个案和规律，这种数据应用围绕异常值展开，因此异常值不能做抛弃处理。

异常检测模型常用于客户异常识别、信用卡欺诈、贷款审批识别、药物变异识别、恶劣气象预测、网络入侵检测、流量作弊检测等。在这

种情况下，异常数据本身是目标数据，如果被处理掉将损失关键信息。

3.包容异常值的数据建模

如果数据算法和模型对异常值不敏感，那么即使不处理异常值也不会对模型本身造成负面影响。例如在决策树中，异常值本身就可以作为一种分裂节点。

除了抛弃和保留，还有一种思路可对异常值进行处理，例如使用其他统计量、预测量进行替换，但不推荐使用这种方法，原因是这会将其中的关键分布特征消除，从而改变原始数据集的分布规律。

3.1.3 数据重复就需要去重吗

数据集中的重复值包括以下两种情况：

·数据值完全相同的多条数据记录。这是最常见的数据重复情况。

·数据主体相同但匹配到的唯一属性值不同。这种情况多见于数据仓库中的变化维度表，同一个事实表的主体会匹配同一个属性的多个值。

去重是重复值处理的主要方法，主要目的是保留能显示特征的唯一数据记录。但当遇到以下几种情况时，请慎重（不建议）执行数据去重。

1.重复的记录用于分析演变规律

以变化维度表为例。例如在商品类别的维度表中，每个商品对应了同1个类别的值应该是唯一的，例如苹果iPhone7属于个人电子消费品，这样才能将所有所有商品分配到唯一类别属性值中。但当所有商品类别的值重构或升级时（大多数情况下随着公司的发展都会这么做），原有的商品可能被分配了类别中的不同值。表3-1展示了这种变化。

表3-1　商品类别归属的变化

商品名称	原有商品类别归属	新商品类别归属
苹果 iPhone7	个人电子消费品	手机数码

此时，我们在数据中使用Full　join做跨重构时间点的类别匹配时，会发现苹果iPhone7会同时匹配到个人电子消费品和手机数码两条记录。对于这种情况，需要根据具体业务需求处理：

·如果跟业务沟通，两条数据需要做整合，那么需要确定一个整合字段用来涵盖2条记录。其实就是将2条数据再次映射到一个类别主体中。

·如果跟业务沟通，需要同时保存2条数据，那么此时不能做任何处理。后续的具体处理根据建模需求而定。

相关知识点：变化维度表

变化维度表是数据仓库中的概念。维度表类似于匹配表，用来存储静态的维度、属性等数据，而这些数据一般都不会改变。但是变与不变是一个相对的概念，随着企业的不断发展，很多时候维度也会发生变化。因此在某个时间内的维度是不变的，而从整体来看维度是变化的。

对于维度的变化，有3种方式进行处理：

·直接覆盖原有值。这种情况下每个唯一ID就只对应一个属性值，这样做虽然简单粗暴也容易实现，但是无法保留历史信息。

·添加新的维度行。此时同一个ID会得到两条匹配记录。

·增加新的属性列。此时不会新增数据行记录，只是在原有的记录中新增一列用于标记不同时期的值。

具体到企业内使用哪种方式，通常是由数据库管理员根据实际情况来决定。

真正的变化维度表或维度表不会以中文做主键，通常都会使用数字或字符串类作为唯一关联ID，本节的示例仅做说明之用。

2.重复的记录用于样本不均衡处理

在开展分类数据建模工作时，样本不均衡是影响分类模型效果的关键因素之一，解决分类方法的一种方法是对少数样本类别做简单过采样，通过随机过采样采取简单复制样本的策略来增加少数类样本。经过这种处理方式后，也会在数据记录中产生相同记录的多条数据。此时，我们不能对其中重复值执行去重操作。

有关样本不均衡的相关内容将在3.4节中介绍。

3.重复的记录用于检测业务规则问题

对于以分析应用为主的数据集而言，存在重复记录不会直接影响实际运营，毕竟数据集主要用来做分析；但对于事务型的数据而言，重复数据可能意味着重大运营规则问题，尤其当这些重复值出现在与企业经营中金钱相关的业务场景中，例如重复的订单、重复的充值、重复的预约项、重复的出库申请等。

这些重复的数据记录通常是由于数据采集、存储、验证和审核机制的不完善等问题导致的，会直接反映到前台生产和运营系统。以重复订单为例，假如前台的提交订单功能不做唯一性约束，那么在一次订单中重复点击提交订单按钮，就会触发多次重复提交订单的申请记录，如果该操作审批通过后，会联动带动运营后端的商品分拣、出库、送货，如果用户接收重复商品则会导致重大损失；如果用户退货则会增加反向订单，并影响物流、配送和仓储相关的各个运营环节，导致运营资源无端消耗、商品损耗增加、仓储物流成本增加等问题。

因此，这些问题必须在前期数据采集和存储时就通过一定机制解决和避免。如果确实产生了此类问题，那么数据工作者或运营工作者可以基于这些重复值来发现规则漏洞，并配合相关部门最大限度降低给企业带来的运营风险。

3.1.4 代码实操：Python数据清洗

1.缺失值处理

对于缺失值的处理，主要配合使用sklearn.preprocessing中的Imputer类、Pandas和Numpy。其中由于Pandas对于数据探索、分析和探查的支持较为良好，因此围绕Pandas的缺失值处理较为常用。

```
import pandas as pd  # 导入Pandas库
import numpy as np  # 导入Numpy库
from sklearn.preprocessing import Imputer   # 导入sklearn.preprocessing中的Imputer库

# 生成缺失数据
df = pd.DataFrame(np.random.randn(6, 4), columns=['col1', 'col2', 'col3', 'col4'])  # 生成一份数据
df.iloc[1:2, 1] = np.nan  # 增加缺失值
df.iloc[4, 3] = np.nan  # 增加缺失值
print (df)

# 查看哪些值缺失
nan_all = df.isnull()  # 获得所有数据框中的N值
print (nan_all)  # 打印输出
# 查看哪些列缺失
nan_col1 = df.isnull().any()  # 获得含有NA的列
nan_col2 = df.isnull().all()  # 获得全部为NA的列
print (nan_col1)  # 打印输出
print (nan_col2)  # 打印输出

# 丢弃缺失值
df2 = df.dropna()  # 直接丢弃含有NA的行记录
print (df2)  # 打印输出

# 使用sklearn将缺失值替换为特定值
nan_model = Imputer(missing_values='NaN', strategy='mean', axis=0)  # 建立替换规则：将值为Nan的缺失值用均值做替换
nan_result = nan_model.fit_transform(df)  # 应用模型规则
print (nan_result)  # 打印输出

# 使用Pandas将缺失值替换为特定值
nan_result_pd1 = df.fillna(method='backfill')  # 用后面的值替换缺失值
nan_result_pd2 = df.fillna(method='bfill', limit=1)  # 用后面的值替换缺失值,限制每列只能替换一个缺失值
nan_result_pd3 = df.fillna(method='pad')  # 用前面的值替换缺失值
nan_result_pd4 = df.fillna(0)  # 用0替换缺失值
nan_result_pd5 = df.fillna({'col2': 1.1, 'col4': 1.2})  # 用不同值替换不同列的缺失值
nan_result_pd6 = df.fillna(df.mean()['col2':'col4'])  # 用平均数代替,选择各自列的均值替换缺失值
# 打印输出
print (nan_result_pd1)  # 打印输出
print (nan_result_pd2)  # 打印输出
print (nan_result_pd3)  # 打印输出
print (nan_result_pd4)  # 打印输出
```

```
print (nan_result_pd5)  # 打印输出
print (nan_result_pd6)  # 打印输出
```

上述代码用空行分为6个部分。

第一部分为导入库，该代码示例中用到了Pandas、Numpy和Sklearn。

第二部分生成缺失数据，通过Pandas生成一个6行4列，列名分别为'col1'、'col2'、'col3'、'col4'的数据框。同时，数据框中增加两个缺失值数据。除了示例中直接通过pd.DataFrame来创建数据框外，还可以使用数据框对象的df.from_records、df.from_dict、df.from_items来从元组记录、字典和键值对对象创建数据框，或使用pandas.read_csv、pandas.read_table、pandas.read_clipboard等方法读取文件或剪贴板创建数据框。该代码段执行后返回了定义的含有缺失值的数据框，结果如下：

```
   col1      col2      col3      col4
0 -0.112415 -0.768180 -0.084859  0.296691
1 -1.777315       NaN -0.166615 -0.628756
2 -0.629461  1.892790 -1.850006  0.157567
3  0.544860 -1.230804  0.836615 -0.945712
4  0.703394 -0.764552 -1.214379       NaN
5  1.928313 -1.376593 -1.557721  0.289643
```

第三部分通过df.null（）方法找到所有数据框中的缺失值（默认缺失值是NaN格式），然后使用any（）或all（）方法来查找含有至少1个或全部缺失值的列，其中any（）方法用来返回指定轴中的任何元素为True，而all（）方法用来返回指定轴的所有元素都为True。该代码段执行后返回如下结果。

1）判断元素是否是缺失值（第2行第2列和第5行第4列）：

```
   col1   col2   col3   col4
0  False  False  False  False
1  False  True   False  False
2  False  False  False  False
3  False  False  False  False
4  False  False  False   True
5  False  False  False  False
```

2）列出至少有一个元素含有缺失值的列（该示例中为col2和col4）：

```
col1    False
col2     True
col3    False
col4     True
dtype:  bool
```

3）列出全部元素含有缺失值的列（该示例中没有）：

```
col1    False
col2    False
col3    False
col4    False
dtype:  bool
```

第四部分通过Pandas默认的dropna（）方法丢弃缺失值，返回无缺失值的数据记录。该代码段执行后返回如下结果（第2行、第5行数据记录被删除）：

```
   col1      col2      col3      col4
0 -0.112415 -0.768180 -0.084859  0.296691
2 -0.629461  1.892790 -1.850006  0.157567
3  0.544860 -1.230804  0.836615 -0.945712
5  1.928313 -1.376593 -1.557721  0.289643
```

第五部分通过Sklearn的数据预处理方法对缺失值进行处理。首先通过Imputer方法创建一个预处理对象，其中strategy为默认缺失值的字符串，默认为NaN；示例中选择缺失值替换方法是均值（默认），还可以选择使用中位数和众数进行替换，即strategy值设置为median或most_frequent；后面的参数axis用来设置输入的轴，默认值为0，即使用列做计算逻辑。然后使用预处理对象的fit_transform方法对df（数据框对象）进行处理，该方法是将fit和transform组合起来使用。代码执行后返回如下结果：

```
[[-0.11241503 -0.76818022 -0.08485904  0.29669147]
 [-1.77731513 -0.44946793 -0.16661458 -0.62875601]
 [-0.62946127  1.89278959 -1.85000643  0.15756702]
 [ 0.54486026 -1.23080434  0.836615   -0.9457117 ]
 [ 0.70339369 -0.76455205 -1.21437918 -0.16611331]
```

```
 [ 1.92831315 -1.37659263 -1.55772092  0.28964265]]
```

代码中的第2行第2列和第5行第4列分别被各自列的均值替换。为了验证我们手动计算各自列的均值，通过使用df['col2'].mean（）和df['col4'].mean（）分别获得这两列的均值，即-0.4494679289032068和-0.16611331259664791，跟Sklearn返回的结果一致。

第六部分使用Pandas做缺失值处理。Pandas对缺失值的处理方法是df.fillna（），该方法中最主要的两个参数是value和method。前者通过固定（或手动指定）的值替换缺失值，后者使用Pandas提供的默认方法替换缺失值，以下是method支持的方法：

·pad和ffill：使用前面的值替换缺失值，示例中nan_result_pd1和nan_result_pd2使用了该方法。

·backfill和bfill：使用后面的值替换缺失值，示例中nan_result_pd3使用了该方法。

·None：无。

在示例中nan_result_pd4、nan_result_pd5、nan_result_pd6分别使用0、不同的值、平均数替换缺失值。需要注意的是，如果要使用不同值替换，需要使用scalar、dict、Series或DataFrame的格式定义。

上述代码执行后返回如下结果。

1）用后面的值（method='backfill'）替换缺失值：

```
    col1      col2      col3      col4
0 -0.112415 -0.768180 -0.084859  0.296691
1 -1.777315  1.892790 -0.166615 -0.628756
2 -0.629461  1.892790 -1.850006  0.157567
3  0.544860 -1.230804  0.836615 -0.945712
4  0.703394 -0.764552 -1.214379  0.289643
5  1.928313 -1.376593 -1.557721  0.289643
```

2）用后面的值（method='bfill'，limit=1）替换缺失值：

```
    col1      col2      col3      col4
```

```
0 -0.112415 -0.768180 -0.084859  0.296691
1 -1.777315  1.892790 -0.166615 -0.628756
2 -0.629461  1.892790 -1.850006  0.157567
3  0.544860 -1.230804  0.836615 -0.945712
4  0.703394 -0.764552 -1.214379  0.289643
5  1.928313 -1.376593 -1.557721  0.289643
```

3）用前面的值替换缺失值（method='pad'）：

```
   col1      col2      col3      col4
0 -0.112415 -0.768180 -0.084859  0.296691
1 -1.777315 -0.768180 -0.166615 -0.628756
2 -0.629461  1.892790 -1.850006  0.157567
3  0.544860 -1.230804  0.836615 -0.945712
4  0.703394 -0.764552 -1.214379 -0.945712
5  1.928313 -1.376593 -1.557721  0.289643
```

4）用0替换缺失值：

```
   col1      col2      col3      col4
0 -0.112415 -0.768180 -0.084859  0.296691
1 -1.777315  0.000000 -0.166615 -0.628756
2 -0.629461  1.892790 -1.850006  0.157567
3  0.544860 -1.230804  0.836615 -0.945712
4  0.703394 -0.764552 -1.214379  0.000000
5  1.928313 -1.376593 -1.557721  0.289643
```

5）手动指定两个缺失值分布为1.1和1.2：

```
   col1      col2      col3      col4
0 -0.112415 -0.768180 -0.084859  0.296691
1 -1.777315  1.100000 -0.166615 -0.628756
2 -0.629461  1.892790 -1.850006  0.157567
3  0.544860 -1.230804  0.836615 -0.945712
4  0.703394 -0.764552 -1.214379  1.200000
5  1.928313 -1.376593 -1.557721  0.289643
```

6）用平均数代替，选择各自列的均值替换缺失值：

```
   col1      col2      col3      col4
0 -0.112415 -0.768180 -0.084859  0.296691
1 -1.777315 -0.449468 -0.166615 -0.628756
2 -0.629461  1.892790 -1.850006  0.157567
3  0.544860 -1.230804  0.836615 -0.945712
4  0.703394 -0.764552 -1.214379 -0.166113
5  1.928313 -1.376593 -1.557721  0.289643
```

以上示例中，直接指定method的方法适用于大多数情况，较为简单直接；但使用value的方法则更为灵活，原因是可以通过函数的形式将缺失值的处理规则写好，然后直接赋值即可。限于篇幅，这里不对所有方法做展开讲解。

另外，如果是直接替换为特定值的应用，也可以考虑使用Pandas的replace功能。本示例的df（原始数据框），可直接使用df.replace（np.nan，0），这种用法更加简单粗暴，但也能达到效果。当然，replace的出现是为了解决各种替换应用的，缺失值只是其中的一种应用而已。

上述过程中，需要考虑的关键点是：缺失值的替换策略，可指定多种方法替换缺失值，具体根据实际需求而定，但大多数情况下均值、众数和中位数的方法较为常用。如果场景固定，也可以使用特定值（例如0）替换。

2.异常值处理

有关异常值的确定有很多规则和方法，这里使用Z标准化得到的阈值作为判断标准：当标准化后的得分超过阈值则为异常。完整代码如下：

```
import pandas as pd  # 导入Pandas库

# 生成异常数据
df = pd.DataFrame({'col1': [1, 120, 3, 5, 2, 12, 13],
                   'col2': [12, 17, 31, 53, 22, 32, 43]})
print (df)  # 打印输出

# 通过Z-Score方法判断异常值
df_zscore = df.copy()  # 复制一个用来存储Z-score得分的数据框
cols = df.columns  # 获得数据框的列名
for col in cols:  # 循环读取每列
    df_col = df[col]  # 得到每列的值
    z_score = (df_col - df_col.mean()) / df_col.std()  # 计算每列的Z-score得分
     df_zscore[col] = z_score.abs() > 2.2  # 判断Z-score得分是否大于2.2，如果是则为True，否则为False
print (df_zscore)  # 打印输出
```

示例代码用空行分为3个部分：

第一部分导入本例需要的Pandas库。

第二部分生成异常数据。直接通过DataFrame创建一个7行2列的数据框，打印输出结果如下：

```
   col1  col2
0     1    12
1   120    17
2     3    31
3     5    53
4     2    22
5    12    32
6    13    43
```

第三部分为缺失值判断过程。本过程中，先通过df.copy（）复制一个原始数据框的副本用来存储Z-Score标准化后的得分，再通过df.columns获得原始数据框的列名，接着通过循环来判断每一列中的异常值。在判断逻辑中，对每一列的数据使用自定义的方法做Z-Score值标准化得分计算，然后跟阈值2.2做比较，如果大于阈值则为异常。标准化的计算还有更多自动化的方法和场景，有关数据标准化的话题，将在3.9节中具体介绍。本段代码返回结果如下：

```
    col1   col2
0  False  False
1   True  False
2  False  False
3  False  False
4  False  False
5  False  False
6  False  False
```

本示例方法中，阈值的设定是确定异常与否的关键，通常当阈值大于2时，已经是相对异常的表现值。

上述过程中，主要需要考虑的关键点是：如何判断异常值。对于有固定业务规则的可直接套用业务规则，而对于没有固定业务规则的，可以采用常见的数学模型进行判断，即基于概率分布的模型（例如正态分布的标准差范围）、基于聚类的方法（例如KMeans）、基于密度的方法（例如LOF）、基于分类的方法（例如KNN）、基于统计的方法（例如分位数法）等，此时异常值的定义带有较强的主观判断色彩，具体需要根据实际情况选择。

3.重复值处理

有关重复值的处理代码示例如下：

```
import pandas as pd  # 导入Pandas库

# 生成重复数据
data1 = ['a', 3]
data2 = ['b', 2]
data3 = ['a', 3]
data4 = ['c', 2]
df = pd.DataFrame([data1, data2, data3, data4], columns=['col1', 'col2'])
print (df)

# 判断重复数据
isDuplicated = df.duplicated()  # 判断重复数据记录
print (isDuplicated)  # 打印输出

# 删除重复值
new_df1 = df.drop_duplicates()  # 删除数据记录中所有列值相同的记录
new_df2 = df.drop_duplicates(['col1'])  # 删除数据记录中col1值相同的记录
new_df3 = df.drop_duplicates(['col2'])  # 删除数据记录中col2值相同的记录
new_df4  =  df.drop_duplicates(['col1',  'col2'])     #  删除数据记录中指定列（col1/col2）值相同的记录
print (new_df1)  # 打印输出
print (new_df2)  # 打印输出
print (new_df3)  # 打印输出
print (new_df4)  # 打印输出
```

上述代码以空行分为4个部分。

第一部分为将数据导入用到的Pandas库。

第二部分生成重复数据，该数据是一个4行2列数据框，数据结果如下：

```
  col1  col2
0    a     3
1    b     2
2    a     3
3    c     2
```

第三部分判断数据记录是否为重复值，返回每条数据记录是否重复的结果，取值为True或False。判断方法为df.duplicated（），该方法中两个主要的参数是subset和keep：

·subset：要判断重复值的列，可以指定特定列或多个列。默认使用全部列。

·keep：当重复时不标记为True的规则，可设置为第一个（first）、最后一个（last）和全部标记为True（False）。默认使用first，即第一个重复值不标记为True。

结果如下：

```
0    False
1    False
2     True
3    False
dtype: bool
```

第四部分为删除重复值的操作。该操作的核心方法是df.drop_duplicates（），该方法的作用是基于指定的规则判断为重复值之后，删除重复值，其参数跟df.duplicated（）完全相同。在该部分方法示例中，依次使用默认规则（全部列相同的数据记录）、col1列相同、col2列相同以及指定col1和col2完全相同四种规则进行去重。返回结果如下：

1）删除数据记录中所有列值相同的记录：

```
  col1  col2
0    a     3
1    b     2
3    c     2
```

2）删除数据记录中col1值相同的记录：

```
  col1  col2
0    a     3
1    b     2
3    c     2
```

3）删除数据记录中col2值相同的记录：

```
  col1  col2
0    a     3
1    b     2
```

4）删除数据记录中指定列（col1和col2）值相同的记录：

```
  col1  col2
0    a     3
1    b     2
3    c     2
```

由于数据是通过随机数产生，因此读者操作的结果可能跟上述示例的数据结果不同。

除了Pandas可用来做重复值判断和处理外，也可以使用Numpy中的unique（）方法，该方法返回其参数数组中所有不同的值，并且按照从小到大的顺序排列。Python自带的内置函数set方法，也能返回唯一元素的集合。

上述过程中，需要考虑的关键点是：如何对重复值进行处理。重复值的判断相对简单，而判断之后如何处理往往不是一个技术特征明显的工作，而是侧重于业务和建模需求的工作。

本小节示例中，主要用了几个知识点：

·通过pd.DataFrame新建数据框；

·通过df.iloc[]来选择特定的列或对象；

·使用Pandas的isnull（）判断值是否为空；

·使用all（）和any（）判断每列是否包含至少1个为True或全部为True的情况；

·使用Pandas的dropna（）直接删除缺失值；

·使用Sklearn.preprocessing中的Imputer方法对缺失值进行填充和替换，支持3种填充方法；

·使用Pandas的fillna填充缺失值，支持更多自定义的值和常用预定义方法；

·通过copy（）获得一个对象副本，常用于原始对象和复制对象同时进行操作的场景；

·通过for循环遍历可迭代的列表值；

·自定义了Z-Score计算公式；

·通过Pandas的duplicated（）判断重复数据记录；

·通过Pandas的drop_duplicates（）删除数据记录，可指定特定列或全部。

3.2 将分类数据和顺序数据转换为标志变量

分类数据和顺序数据是常见的数据类型，这些值主要集中在围绕数据实体的属性和描述的相关字段和变量中。

3.2.1 分类数据和顺序数据是什么

在数据建模过程中，很多算法无法直接处理非数值型的变量。例如KMeans算法用于基于距离的相似度计算，而字符串则无法直接计算距离。另外，即使算法本身支持，很多算法实现包也无法直接基于字符串做矩阵运算，例如Numpy以及基于Numpy的sklearn，虽然这些库允许直接使用和存储字符串型变量，但却无法发挥矩阵计算的优势。这些类型的数据可以分为两类：

1）分类数据：分类数据指某些数据属性只能归于某一类别的非数值型数据，例如性别中的男、女就是分类数据。分类数据中的值没有明显的高、低、大、小等包含等级、顺序、排序、好坏等逻辑的划分，只是用来区分两个或多个具有相同或相当价值的属性。例如：性别中的男和女，颜色中的红、黄和蓝，它们都是相同衡量维度上的不同属性分类而已。

2）顺序数据：顺序数据只能归于某一有序类别的非数值型数据，例如用户的价值度分为高、中、低，学历分为博士、研究生、学士，这些都属于顺序数据。在顺序数据中，有明显的排序规律和逻辑层次的划分。例如：高价值的用户就是比低价值的用户价值高（业务定义该分类时已经赋予了这样的价值含义）。

3.2.2 运用标志方法处理分类和顺序数据

分类数据和顺序数据要参与模型计算，通常都会转化为数值型数据。当然，某些算法是允许这些数据直接参与计算的，例如分类算法中的决策树、关联规则等。将非数值型数据转换为数值型数据的最佳方法是：将所有分类或顺序变量的值域从一列多值的形态转换为多列只包含真值的形态，其中的真值可通过True、False或0、1的方式来表示。这种标志转换的方法有时候也称为真值转换。以用户性别变量为例，原有的用户数据如表3-2所示。

经过转换后的数据如表3-3所示。

表3-2　原有用户数据

用户 ID	用 户 性 别
3566841	男
6541227	女
3512441	女

表3-3　标志转换后的用户数据

用户 ID	用户性别 – 男	用户性别 – 女
3566841	1	0
6541227	0	1
3512441	0	1

为什么不能直接用数字来表示不同的分类和顺序数据，而一定要做标志转换？这是因为在用数字直接表示分类和顺序变量的过程中，无法准确还原不同类别信息之间的差异和相互关联性。例如：

·针对分类数据：性别变量的属性值是男和女，无论用什么值来表

示都无法表达出两个值的价值相等且带有区分的含义。如果用1和2区分，那么1和2本身已经带有距离为1的差异，但实际上二者是不具有这种差异性的，其他任意数字都是如此；如果用相同的数字来表示，则无法达到区分的目的。

·针对顺序数据：学历变量的属性值是博士、研究生和学士，可以用3-2-1来表示顺序和排列关系，那么如何表示三个值之间的差异是3-2-1而不是30-20-10或者1000-100-2呢？因此，任何一个有序数字的排序也都无法准确表达出顺序数据的差异性。

3.2.3 代码实操：Python标志转换

在本示例中，将模拟有两列数据分别出现分类数据和顺序数据的情况，并通过自定义代码以及sklearn代码分别进行标志转换。

```
import pandas as pd  # 导入pandas库
from sklearn.preprocessing import OneHotEncoder  # 导入OneHotEncoder库

# 生成数据
df = pd.DataFrame({'id': [3566841, 6541227, 3512441],
                   'sex': ['male', 'Female', 'Female'],
                   'level': ['high', 'low', 'middle']})
print (df)  # 打印输出原始数据框

# 自定义转换主过程
df_new = df.copy()  # 复制一份新的数据框用来存储转换结果
for col_num, col_name in enumerate(df):  # 循环读出每个列的索引值和列名
    col_data = df[col_name]  # 获得每列数据
    col_dtype = col_data.dtype  # 获得每列dtype类型
    if col_dtype == 'object':  # 如果dtype类型是object（非数值型），执行条件
        df_new = df_new.drop(col_name, 1)  # 删除df数据框中要进行标志转换的列
        value_sets = col_data.unique()  # 获取分类和顺序变量的唯一值域
        for value_unique in value_sets:  # 读取分类和顺序变量中的每个值
            col_name_new = col_name + '_' + value_unique  # 创建新的列名，使用“原标题+值”的方式命名
            col_tmp = df.iloc[:, col_num]  # 获取原始数据列
            new_col = (col_tmp == value_unique)  # 将原始数据列与每个值进行比较，相同为True，否则为False
            df_new[col_name_new] = new_col  # 为最终结果集增加新列值
print (df_new)  # 打印输出转换后的数据框

# 使用sklearn进行标志转换
df2 = pd.DataFrame({'id': [3566841, 6541227, 3512441],
                    'sex': [1, 2, 2],
                    'level': [3, 1, 2]})
id_data = df2.values[:, :1]  # 获得ID列
transform_data = df2.values[:, 1:]  # 指定要转换的列
enc = OneHotEncoder()  # 建立模型对象
df2_new = enc.fit_transform(transform_data).toarray()  # 标志转换
df2_all = pd.concat((pd.DataFrame(id_data), pd.DataFrame(df2_new)), axis=1)  # 合为数据框
print (df2_all)  # 打印输出转换后的数据框
```

该代码段按空行分为4个部分：

第一部分导入库，本示例使用Pandas库和sklearn。

第二部分生成原始数据，数据为3行3列的数据框，分别包含id、sex和level列，其中的id为模拟的用户ID，sex为用户性别（英文），level为用户等级（分别用high、middle和low代表三个等级）。该段代码输出原

始数据框如下：

```
        id   level     sex
0  3566841    high    male
1  6541227     low  Female
2  3512441  middle  Female
```


虽然在Python中可以通过一定的方法来处理中文，但鉴于我们用到的库基本都是外国人开发，对中文的支持不太好，所以不建议在程序中直接使用中文进行计算和建模，除非是基于文本的自然语言和文本挖掘等直接面向中文的主题建模。

第三部分为自定义转换主过程。

步骤1　创建数据框副本。通过copy（）方法创建一个原始数据库的副本，用来存储转换后的数据。该步骤不能省略，原因是新的副本数据框和原始数据框在下面的步骤都要用到。

步骤2　通过循环获得原始数据框的列索引和列名。在for循环中使用enumerate（）方法，返回可供迭代的列索引和列名。然后获得每列数据和对应的dtype数据类型，用来做是否进行标志转换的条件判断。在if表达式中，当数据类型为object时进行转换，转换的核心思路如下：

1）通过drop（）方法删除复制得到的数据框中要进行转换的列，并将结果赋值给df_new，这样每次df_new中就会通过循环不断删除要转换的列，避免数据重复。drop方法的第一个参数是要删除的列名，第二个参数是指定要删除的轴，1表示按列删除。

2）通过unique（）方法获取分类和顺序变量的唯一值域，后续的判断主要针对值域列表进行。

3）通过for循环遍历得到值域中的每个值。通过“原始列名称+值”的形式新建一个列名，这样得到的新列名能保留原始列的含义；由于df_new通过不断循环其本身的索引已经改变，因此需要使用iloc（）方法获得原始数据框的列；通过将原始数据列与值域列表中的每个值进行比较，相同为True，否则为False，并将值赋值到df_new结果数据框中。

4）最后打印输出结果数据框。

上述代码执行后得到的结果如下：

```
        id  level_high  level_low  level_middle  sex_male  sex_Female
0  3566841        True      False         False      True       False
1  6541227       False       True         False     False        True
2  3512441       False      False          True     False        True
```

结果数据框中的True和False可以参与后续数据建模和距离计算中，True的值是1而False的值为0。

第四部分为使用sklearn.preprocessing中的OneHotEncoder方法进行标志转换。

步骤1　创建模拟数据。数据为3行3列的数据框。

很多情况下，原始数据都不会以字符串（第一种方法中的示例）的形式存储，而是将数据转换为数字进行存储，这些数字可通过维度表匹配出对应的字符串。当然，如果是这种数据存储方法，第一种自定义转换过程也可以实现，但需要改变对于分类或顺序数据的识别规则，此时的数据类型就不是object，需要根据实际情况通过指定列名等方法转换。

步骤2　获得ID列并指定要转换的列。获得ID列的目的是用于后续对ID列和转换后的列做拼接，便于数据格式的还原和对照。如果不需要做多个方法的对比或对转换后的数据列不作判别应用，则可以跳过本步骤，以及步骤4中的拼接过程。

步骤3　建立模型对象并进行转换输出。该过程中，主要使用的是OneHotEncoder库中的fit_transform方法直接训练并应用转换，然后使用toarray方法输出为矩阵。如果不使用toarray进行转换，那么输出的数据是一个3行5列的稀疏矩阵。

步骤4　将ID列和转换后的列拼接为完整主体，用于跟原始数据和通过方法1得到的数据做比较。得到的结果如下：

```
         0    0    1    2    3    4
0  3566841  0.0  0.0  1.0  1.0  0.0
1  6541227  1.0  0.0  0.0  0.0  1.0
2  3512441  0.0  1.0  0.0  0.0  1.0
```

上述结果中，由于Sklearn的转换算法是基于Numpy做矩阵计算的，因此无法直接使用字符串做转换。另外，输出的转换列的排序也不是按照预先的逻辑，笔者将通过自定义方法和Sklearn方法得到的结果做比较，二者的数据是一致的，但是列的顺序却是不同的，如图3-1所示。

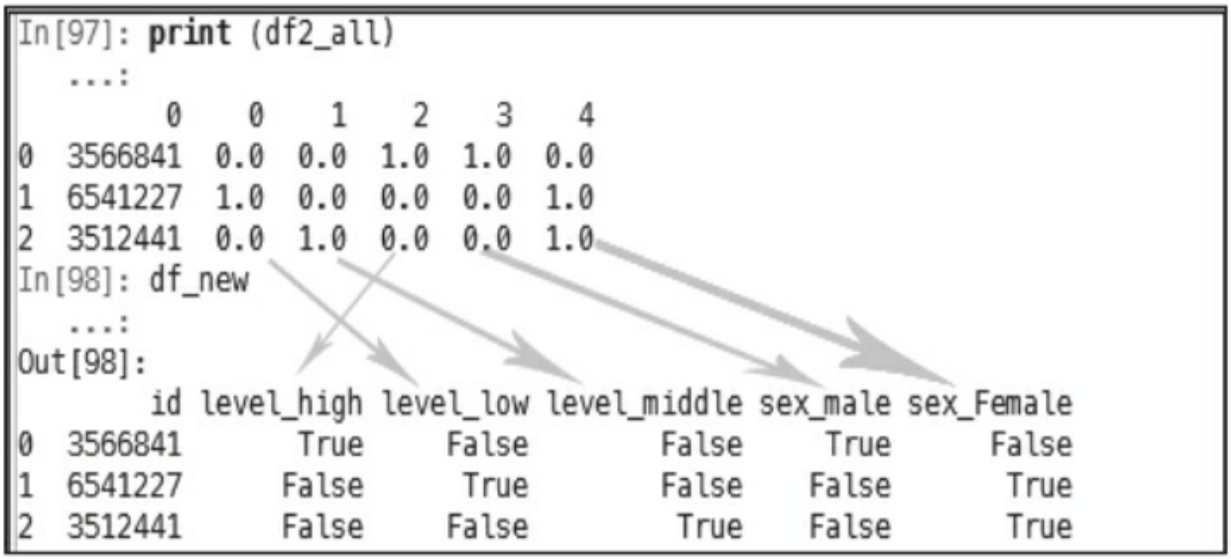


图3-1　自定义转换和sklearn转换的列顺序对照

虽然列的顺序不同，但不会影响数据建模，因为列的顺序对建模不构成任何影响。只是在得到结果后，如果需要做基于特征的解读时，可能会导致无法对应到原始特征变量和转换值。如果是直接面向机器处理的应用则没有任何影响。

上述过程中，需要考虑的关键点是：

·如何判断要转换的数据是分类或顺序数据。

·要在结果数据框中不断删除被转换的原始列并新增转换后的数据列，以防止数据列的重复。

本小节示例中，主要用了以下几个知识点：

·通过pd.DataFrame构建新的数据框；

·通过Pandas中的df[col_name]和iloc[]进行数据切片；

·通过Pandas中的drop（）方法删除特定列，当然也可以用于删除行；

·通过Pandas的dtype获得对象的dtype类型，df.dtypes也能实现所有对象的类型；

·通过unique（）方法获得唯一值；

·通过字符串组合（示例中直接使用的+）创建一个新的字符串；

·直接使用矩阵（Series）对象而无须遍历每个值进行矩阵比较和数值计算；

·通过Pandas的df_new[col_name_new]方法直接新增列值。

3.3 大数据时代的数据降维

数据降维就是降低数据的维度数量，数据降维是维数归约的一个重要课题。

3.3.1 需要数据降维的情况

数据降维可以降低模型的计算量并减少模型运行时间、降低噪音变量信息对于模型结果的影响、便于通过可视化方式展示归约后的维度信息并减少数据存储空间。因此，大多数情况下，当我们面临高维数据时，都需要对数据做降维处理。是否进行降维主要考虑以下方面：

1）维度数量。降维的基本前提是高维，假如模型只有几个维度，那就不一定需要降维，具体取决于维度本身的重要性、共线性以及其他排除关系，而不是出于高维的考虑。

2）建模输出是否必须保留原始维度。某些场景下，我们需要完整保留参与建模的原始维度并在最终建模输出时能够得以分析、解释和应用，这种情况下不能进行转换方式降维，只能选择特征筛选的方式降维。

3）对模型的计算效率与建模时效性有要求。当面临高维数据建模时，数据模型的消耗将呈几何倍数增长，这种增长带来的结果便是运算效率慢、耗时长。如果对建模时间和时效性有要求，那么降维几乎是必要步骤。

4）是否要保留完整数据特征。数据降维的基本出发点是在尽量（或最大化）保留原始数据特征的前提下，降低参与建模的维度数。在降维过程中，无论未被表示出来的特征是噪音还是正常分布，这部分信息都无法参与建模。如果某些场景下需要所有数据集的完整特征，那么通常不选择降维。

数据降维只是处理高维数据的思路和方法之一，除此之外，国内外学者也在研究其他高维数据建模的方法。以高维数据聚类为例，除了可以通过降维来应对之外，其他思路还包括基于超图的聚类、基于子空间的聚类、联合聚类等，这些都是非降维的方法。

3.3.2 基于特征选择的降维

基于特征选择的降维指的是根据一定规则和经验，直接选取原有维度的一部分参与到后续的计算和建模过程，用选择的维度代替所有维度，整个过程不产生新的维度。

基于特征选择的降维方法有4种思路：

·经验法：根据业务专家或数据专家的以往经验、实际数据情况、业务理解程度等进行综合考虑。业务经验依靠的是业务背景，从众多维度特征中选择对结果影响较大的特征；而数据专家则依靠的是数据工作经验，基于数据的基本特征以及对后期数据处理和建模的影响来选择或排除维度，例如去掉缺失值较多的特征。

·测算法：通过不断测试多种维度选择参与计算，通过结果来反复验证和调整并最终找到最佳特征方案。

·基于统计分析的方法：通过相关性分析不同维度间的线性相关性，在相关性高的维度中进行人工去除或筛选；或者通过计算不同维度间的互信息量，找到具有较高互信息量的特征集，然后把其中的一个特征去除或留下。

·机器学习算法：通过机器学习算法得到不同特征的特征值或权重，然后再根据权重来选择较大的特征。图3-2所示是通过CART决策树模型得到不同变量的重要程度，然后根据实际权重值进行选择。

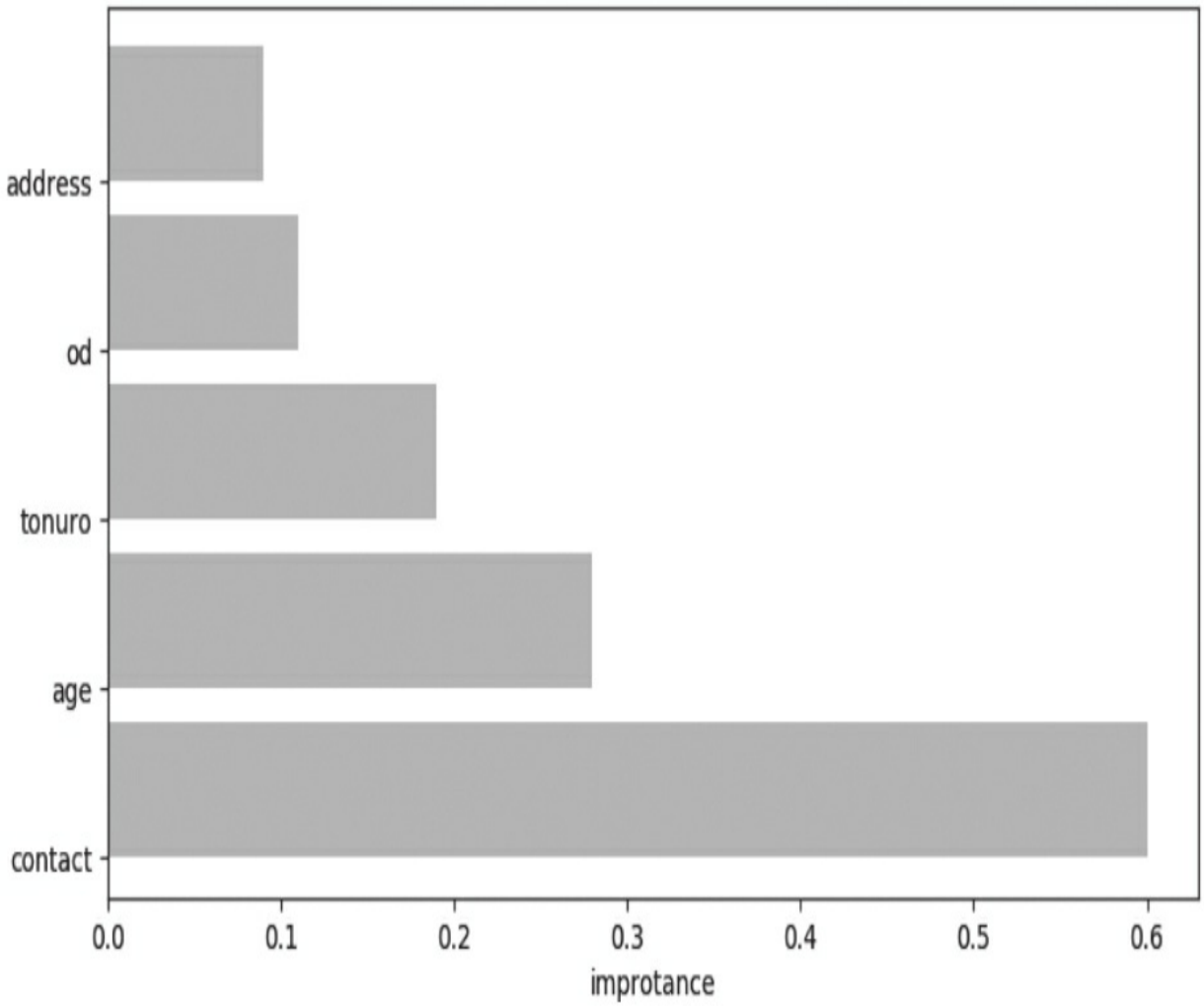


图3-2　变量重要性得分

这种数据降维方法的好处是，在保留了原有维度特征的基础上进行降维，既能满足后续数据处理和建模需求，又能保留维度原本的业务含义，以便于业务理解和应用。对于业务分析型的应用而言，模型的可理解性、可解释性和可应用性在很多时候，其优先级要高于模型本身的准确率、效率等技术类指标。要知道，如果没有业务的理解和支持，再好的数据、模型和算法都无法落地。

例如通过决策树得到的特征规则，可以作为选择用户样本的基础条件，而这些特征规则便是基于输入的维度产生。假如我们在决策树之前将原有维度用表达式（例如PCA的主成分）方法进行转换，那么即使得到了决策树规则，也无法直接应用于业务。

3.3.3 基于维度转换的降维

基于维度转换的降维是按照一定的数学变换方法，把给定的一组相关变量（维度）通过数学模型将高维空间的数据点映射到低维度空间中，然后利用映射后变量的特征来表示原有变量的总体特征。这种方式是一种产生新维度的过程，转换后的维度并非原有维度的本体，而是其综合多个维度转换或映射后的表达式。

例如，假设原始数据集中有10个维度，分别是tenure、cardmon、lncardmon、cardten、lncardten、wiremon、lnwiremon、wireten、lnwireten、hourstv，现在用主成分分析进行降维，降维后选择具有显著性的前3个主成分作为示例：

```
方程式用于 主成分-1
    0.006831 * tenure +
    0.007453 * cardmon +
    0.1861 * lncardmon +
    0.0001897 * cardten +
    0.1338 * lncardten +
    + -4.767
方程式用于 主成分-2
    -0.4433 * lnwiremon +
    -0.0001222 * wireten +
    -0.1354 * lnwireten +
    0.008099 * hourstv +
    + -0.272
方程式用于 主成分-3
    -0.01809 * tenure +
    0.0124 * cardmon +
    0.00002565 * wireten +
    -0.1644 * lnwireten +
    0.03984 * hourstv +
    + -4.076
```

上述转换结果就是主成分分析后提取的3个能基本代表原始10个维度的新“维度”。通过上述结果发现主成分（也就是新的“维度”）是一个多元一次方程，其含义无法直接体现和理解。但对于不关注每个维度业务含义的系统级应用和集成来讲是没有关系的，因为后续算法不需要知道每个变量的具体业务含义。

为了更形象地解释映射关系，现在使用一个序列图来表示整个映射过程。图3-3所示为原始数据集在经过顺时针旋转（映射变换）后的可视化特征变化过程。从图3-3中所示①到⑧的序列中可以看出，原始数

据的分布呈现明显的曲线分布特征，而旋转后的数据样本则呈现直线分布特征。

通过数据维度变换进行降维是非常重要的降维方法，这种降维方法分为线性降维和非线性降维两种，其中常用的代表算法包括独立成分分析（ICA）、主成分分析（PCA）、因子分析（Factor Analysis，FA）、线性判别分析（LDA，又称Fisher线性判别FLD）、局部线性嵌入（LLE）、核主成分分析（Kernel PCA）等。

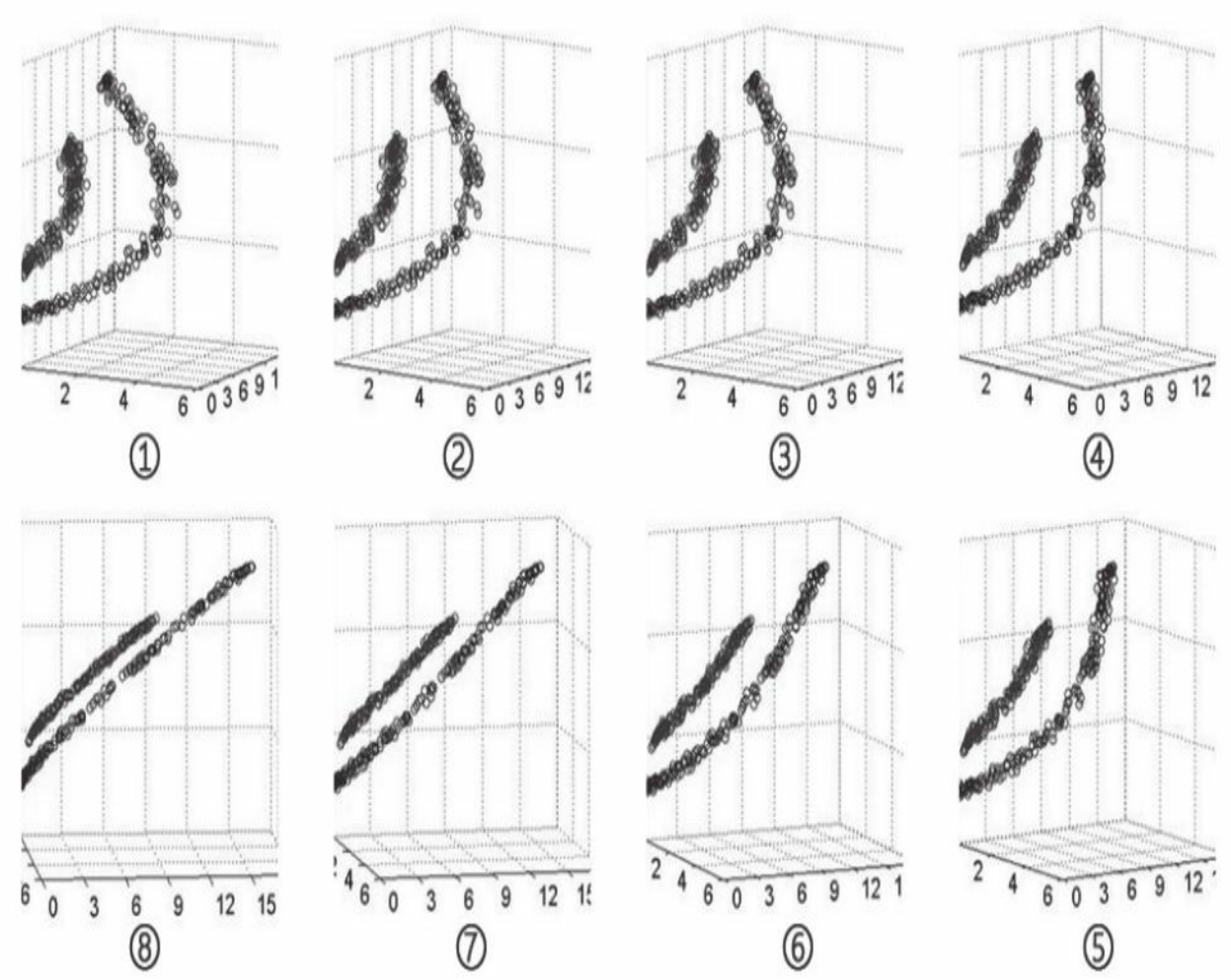


图3-3　数据维度特征转换过程

以下介绍PCA的主要适用场景：

·非监督式的数据集。它是一种非监督式的降维方法，因此适用于不带有标签的数据集；而对于带有标签的数据集则可以采用LDA。

·根据方差自主控制特征数量。最大的主成分的数量会小于或等于特征的数量，这意味着，PCA也可以输出完全相同数量的特征，具体取决于选择特征中解释的方差比例。

·更少的正则化处理。选择较多的主成分将导致较少的平滑，因为我们将能够保留更多的数据特征，从而减少正则化。

·数据量较大的数据集。数据量大包括数据记录多和数据维度多两种情况，PCA对大型数据集的处理效率较高。

·数据分布是位于相同平面上（非曲面），数据中存在线性结构。

在数据挖掘或机器学习中，经常会把数据分为训练集和测试集，也有可能包括应用集，多份数据需要单独降维还是把数据集合起来进行整体降维？对于线性方法（例如PCA）而言，它旨在寻找一个高维空间到低维空间的映射矩阵或映射关系，当映射矩阵找到后便可直接将其应用到其他数据集进行降维（通俗点理解就是直接套用矩阵公式得到降维结果），因此，这种降维方式下可以单独降维；而非线性方法（例如LLE）则需要在保持某种局部结构的条件下实现数据的整体降维，此时需要所有的数据集合到一起然后才能实现数据降维。

3.3.4 代码实操：Python数据降维

本示例中，将分别使用sklearn的DecisionTreeClassifier来判断变量重要性并选择变量，通过PCA进行维度转换。数据源文件data1.txt位于“附件-chapter3”中，默认程序的工作目录为“附件-chapter3”（如果不是，请切换到该目录下，否则会报错“IOError:File data1.txt does not exist”）。完整代码如下：

```
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.decomposition import PCA

# 读取数据文件
data = np.loadtxt('data1.txt')  # 读取文本数据文件
x = data[:, :-1]  # 获得输入的x
y = data[:, -1]  # 获得目标变量y
print (x[0], y[0])  # 打印输出x和y的第一条记录

# 使用sklearn的DecisionTreeClassifier判断变量重要性
model_tree = DecisionTreeClassifier(random_state=0)  # 建立分类决策树模型对象
model_tree.fit(x, y)  # 将数据集的维度和目标变量输入模型
feature_importance = model_tree.feature_importances_  # 获得所有变量的重要性得分
print (feature_importance)  # 打印输出

# 使用sklearn的PCA进行维度转换
model_pca = PCA()  # 建立PCA模型对象
model_pca.fit(x)  # 将数据集输入模型
model_pca.transform(x)  # 对数据集进行转换映射
components = model_pca.components_  # 获得转换后的所有主成分
components_var = model_pca.explained_variance_  # 获得各主成分的方差
components_var_ratio = model_pca.explained_variance_ratio_  # 获得各主成分的方差占比
print (components[:2])  # 打印输出前2个主成分
print (components_var[:2])  # 打印输出前2个主成分的方差
print (components_var_ratio)  # 打印输出所有主成分的方差占比
```

上述代码以空行分为4部分。

第一部分导入库。本程序中涉及Numpy和Sklearn中的DecisionTreeClassifier、PCA。

第二部分导入数据文件。通过Numpy的loadtxt方法读取数据文件得到矩阵，然后对矩阵进行切割，得到输入变量集x和目标变量y，x和y的第一条数据记录为：

```
(array([ 1.88622997,  1.31785876, -0.16480621,  0.56536882, -1.11934542,
       -0.53218995, -0.6843102 ,  1.24149827,  1.00579225,  0.45485041]), 0.
```

第三部分使用Sklearn的DecisionTreeClassifier判断变量重要性。首先建立分类决策树模型对象，在参数中通过random_state来控制随机种子，如果不设置则会导致模型结果有微小差异；然后通过fit方法输入x和y进行训练，在训练模型中通过feature_importances_方法得到变量重要性得分，打印输出结果如下：

```
[ 0.03331054  0.01513967  0.02199713  0.119727    0.47930312  0.04776297
  0.17111746  0.02585441  0.02012725  0.06566044]
```

从变量重要性得分看出，第4/5/7三个变量的重要性最高，分别约为0.12、0.48、0.17，三者得分之和约等于77%，这意味着仅仅这3个变量已经具有非常显著的并且足以代表所有变量参与模型计算的能力。因此，可以选择这3个变量参与后续模型计算。

一般情况下，如果选择的变量重要性总得分接近80%，基本上已经可以解释大部分的特征变化了。变量重要性的总得分为1，值越大重要性越高。

第四部分为使用Sklearn的PCA进行维度转换。该部分的实现思路与DecisionTree-Classifier类似，不同之处在于当创建模型对象之后，我们使用fit方法仅输入x进行PCA训练（y不需要转换），然后通过transform进行转换，fit和tranform可以合并为一步fit_transform（x）。但是考虑到PCA通过fit方法得到的映射关系在后续会用到，因此这里单独使用transform实现对数据集的降维转换。在得到PCA训练模型后，通过模型的components_、explained_variance_、explained_variance_ratio_属性分别获取转换后的所有主成分、各主成分的方差和各主成分的方差占比。打印输出结果如下：

1）前2个主成分：

```
[[  7.18818316e-03   1.41619205e-02   1.00543847e-02   3.65097575e-01
    6.38944537e-01  -1.95750380e-02  -1.73413378e-01  -3.80829974e-02
   -2.87413113e-03  -6.52829504e-01]
 [  1.01307710e-02  -1.95270201e-04  -2.33689543e-02  -6.12915216e-01
```

```
 5.08983971e-01  -2.23429533e-02   6.02958940e-01  -1.49061329e-02
-1.81362216e-02  -3.41623971e-03]]
```

2）前2个主成分对应的方差：

```
[ 4.22180334  2.20928822]
```

3）所有主成分的方差占比

```
[  3.38339364e-01   1.77054475e-01   8.92753857e-02   8.73655166e-02
   8.23542686e-02   8.03329836e-02   7.38094896e-02   7.14685179e-02
   7.52486951e-33   2.60352734e-33]
```

上述结果中，所有主成分的方差占比是选择主成分数量的关键，因为PCA降维的基本思想是根据方差占比来选择主成分的数量。通过该结果可以看出，前6项主成分的方差占比之和components_var_ratio[:5].sum（）约等于77%，取前6项主成分基本可以作为转换后的主成分参与后续模型计算。

在建立PCA模型时，可通过n_components指定要获得的主成分数量，但通常不建议采用这种方式，原因是无法根据各主成分的方差占比判断到底指定多少个主成分最合适。当然，如果对于主成分或转换后变量的数量有强烈约束，那么直接指定components的方法是行之有效的。

上述过程中，需要考虑的关键点是：

·如何根据变量重要性得分选择原始变量；

·如何通过方差占比选择合适的主成分。

本小节示例中，主要用了几个知识点：

·通过Numpy的loadtxt读取数据文件；

·对Numpy矩阵进行切片；

·通过Sklearn的DecisionTreeClassifier建立决策树模型，并获得变量重要性属性；

·通过使用sklearn的PCA建立PCA模型，并对数据集进行维度转换降维，并从PCA模型中获得有关主成分、方差和方差占比属性。

3.4 解决样本类别分布不均衡的问题

所谓的不均衡指的是不同类别的样本量差异非常大。样本类别分布不均衡主要出现在分类相关的建模问题上。样本类别分布不均衡从数据规模上可以分为大数据分布不均衡和小数据分布不均衡两种。

·大数据分布不均衡；这种情况下整体数据规模大，只是其中的小样本类的占比较少。但是从每个特征的分布来看，小样本也覆盖了大部分或全部的特征。例如拥有1000万条记录的数据集中，其中占比50万条的少数分类样本便于属于这种情况。

·小数据分布不均衡；这种情况下整体数据规模小，并且占据少量样本比例的分类数量也少，这会导致特征分布的严重不平衡。例如拥有1000条数据样本的数据集中，其中占有10条样本的分类，其特征无论如何拟合也无法实现完整特征值的覆盖，此时属于严重的数据样本分布不均衡。

样本分布不均衡将导致样本量少的分类所包含的特征过少，并很难从中提取规律；即使得到分类模型，也容易产生过度依赖于有限的数据样本而导致过拟合的问题，当模型应用到新的数据上时，模型的准确性和健壮性将很差。

样本分布不均衡主要在于不同类别间的样本比例差异。以笔者的工作经验看，如果不同分类间的样本量差异达到超过10倍就需要引起警觉并考虑处理该问题，超过20倍就一定要解决了。

3.4.1 哪些运营场景中容易出现样本不均衡

在数据化运营过程中，以下场景会经常产生样本分布不均衡的问题：

·异常检测场景。大多数企业中的异常个案都是少量的，比如恶意刷单、黄牛订单、信用卡欺诈、电力窃电、设备故障等，这些数据样本所占的比例通常是整体样本中很少的一部分，以信用卡欺诈为例，刷实体信用卡的欺诈比例一般都在0.1%以内。

·客户流失场景。大型企业的流失客户相对于整体客户而言通常是少量的，尤其对于具有垄断地位的行业巨擘，例如电信、石油、网络运营商等更是如此。

·罕见事件的分析。罕见事件与异常检测类似，都属于发生个案较少，不同点在于异常检测通常都有预先定义好的规则和逻辑，并且大多数异常事件都会对企业运营造成负面影响，因此针对异常事件的检测和预防非常重要；罕见事件则无法预判，并且也没有明显的积极或消极影响倾向。例如由于某网络大V无意中转发了企业的一条趣味广告导致用户流量明显提升便属于此类。

·发生频率低的事件。这种事件是预期或计划性事件，但是发生频率非常低。例如每年1次的双11盛会一般都会产生较高的销售额，但放到全年来看这一天的销售额占比很可能只有1%不到，尤其对于很少参与活动的公司而言，这种情况更加明显。这就属于典型的低频事件。

3.4.2 通过过抽样和欠抽样解决样本不均衡

抽样是解决样本分布不均衡相对简单且常用的方法，包括过抽样和欠抽样两种。

1）过抽样：又称上采样或（over-sampling），其通过增加分类中少数类样本的数量来实现样本均衡，最直接的方法是简单复制少数类样本形成多条记录。这种方法的缺点是，如果样本特征少则可能导致过拟合的问题。经过改进的过抽样方法通过在少数类中加入随机噪声、干扰数据或通过一定规则产生新的合成样本，例如SMOTE算法。

2）欠抽样：又称下采样（under-sampling），其通过减少分类中多数类样本的数量来实现样本均衡，最直接的方法是随机去掉一些多数类样本来减小多数类的规模，缺点是会丢失多数类样本中的一些重要信息。

总体上，过抽样和欠抽样更适合大数据分布不均衡的情况，尤其是过抽样方法，应用极为广泛。

3.4.3 通过正负样本的惩罚权重解决样本不均衡

通过正负样本的惩罚权重解决样本不均衡的问题的思想：在算法实现过程中，对于分类中不同样本数量的类别分别赋予不同的权重（一般思路分类中的小样本量类别权重高，大样本量类别权重低），然后进行计算和建模。

使用这种方法时需要对样本本身做额外处理，只需在算法模型的参数中进行相应设置即可。很多模型和算法中都有基于类别参数的调整设置，以scikit-learn中的SVM为例，通过在class_weight:{dict，'balanced'}中针对不同类别来手动指定权重。如果使用其默认的方法balanced，那么SVM会将权重设置为与不同类别样本数量呈反比的权重来进行自动均衡处理，计算公式为：

```
n_samples / (n_classes * np.bincount(y))
```

如果算法本身支持，这种思路是更加简单且高效的方法。

3.4.4 通过组合/集成方法解决样本不均衡

组合/集成方法指的是在每次生成训练集时使用所有分类中的小样本量，同时从分类中的大样本量中随机抽取数据来与小样本量合并构成训练集，这样反复多次会得到很多训练集和训练模型。最后在应用时，使用组合方法（例如投票、加权投票等）产生分类预测结果。

例如，数据集中的正、负例的样本分别为100和10000条，比例为1:100。此时可以将负例样本（类别中的大量样本集）随机分为100份（当然也可以分更多），每份100条数据；然后每次形成训练集时使用所有的正样本（100条）和随机抽取的负样本（100条）形成新的数据集。如此反复可以得到100个训练集和对应的训练模型。

这种解决问题的思路类似于随机森林。在随机森林中，虽然每个小决策树的分类能力很弱，但是通过大量的“小树”组合形成的“森林”具有良好的模型预测能力。

如果计算资源充足，并且对于模型的时效性要求不高，这种方法比较合适。

3.4.5 通过特征选择解决样本不均衡

上述几种方法都是基于数据行的操作，通过多种途径可使不同类别的样本数据行记录均衡。除此以外，还可以考虑使用或辅助基于列的特征选择方法。

一般情况下，样本不均衡也会导致特征分布不均衡，但如果小类别样本量具有一定的规模，那么意味着其特征值的分布较为均衡，可通过选择具有显著型的特征配合参与解决样本不均衡问题，也能在一定程度上提高模型效果。

上述几种方法的思路都是基于分类问题的。实际上，这种从大规模数据中寻找罕见数据的情况，也可以使用非监督式的学习方法，例如使用One-class SVM进行异常检测。分类是监督式方法，前期是基于带有标签（Label）的数据进行分类预测的；而采用非监督式方法，则是使用除了标签以外的其他特征进行模型拟合的，这样也能得到异常数据记录。所以，要解决异常检测类的问题，先是考虑整体思路，再考虑方法模型。

3.4.6 代码实操：Python处理样本不均衡

本示例中，我们主要使用的是一个新的专门用于不平衡数据处理的Python包imbalan-ced-learn，读者需要先在系统终端的命令行使用pip install imbalanced-learn进行安装。安装成功后，在Python或IPython命令行窗口通过使用import imblearn（注意导入的库名）检查安装是否正确，示例代码包版本为0.2.1。除此以外，我们还会使用sklearn的SVM在算法中通过调整类别权重来处理样本不均衡问题。本示例使用的数据源文件data2.txt位于“附件-chapter3”中，默认工作目录为“附件-chapter3”（如果不是，请切换到该目录下，否则会报错“IOError:File data2.txt does not exist”）。完整代码如下：

```
import pandas as pd
from imblearn.over_sampling import SMOTE  # 过抽样处理库SMOTE
from imblearn.under_sampling import RandomUnderSampler  # 欠抽样处理库Random-
UnderSampler
from sklearn.svm import SVC  # SVM中的分类算法SVC
from imblearn.ensemble import EasyEnsemble  # 简单集成方法EasyEnsemble

# 导入数据文件
df = pd.read_table('data2.txt', sep=' ', names=
['col1', 'col2', 'col3', 'col4', 'col5', 'label'])  # 读取数据文件
x = df.iloc[:, :-1]  # 切片，得到输入x
y = df.iloc[:, -1]  # 切片，得到标签y
groupby_data_orgianl = df.groupby('label').count()  # 对label做分类汇总
print (groupby_data_orgianl)  # 打印输出原始数据集样本

# 使用SMOTE方法进行过抽样处理
model_smote = SMOTE()  # 建立SMOTE模型对象
x_smote_resampled, y_smote_resampled = model_smote.fit_sample(x, y)  # 输入
数据并作过抽样处理
x_smote_resampled = pd.DataFrame(x_smote_resampled, columns=
['col1', 'col2', 'col3', 'col4', 'col5'])  # 将数据转换为数据框并命名列名
y_smote_resampled = pd.DataFrame(y_smote_resampled, columns=['label'])  # 将
数据转换为数据框并命名列名
smote_resampled = pd.concat([x_smote_resampled, y_smote_resampled], axis=1)
列合并数据框
groupby_data_smote = smote_resampled.groupby('label').count()  # 对label做分
类汇总
print (groupby_data_smote)  # 打印输出经过SMOTE处理后的数据集样本

# 使用RandomUnderSampler方法进行欠抽样处理
model_RandomUnderSampler = RandomUnderSampler() # 建立RandomUnderSampler模型
对象
x_RandomUnderSampler_resampled, y_RandomUnderSampler_resampled = model_Randc
入数据并进行欠抽样处理
x_RandomUnderSampler_resampled = pd.DataFrame(x_RandomUnderSampler_resamplec
['col1','col2','col3','col4','col5']) # 将数据转换为数据框并命名列名
y_RandomUnderSampler_resampled = pd.DataFrame(y_RandomUnderSampler_resamplec
['label']) # 将数据转换为数据框并命名列名
RandomUnderSampler_resampled = pd.concat([x_RandomUnderSampler_resampled, y_
```

```
列合并数据框
groupby_data_RandomUnderSampler = RandomUnderSampler_resampled.groupby('labe
label做分类汇总
print (groupby_data_RandomUnderSampler) # 打印输出经过RandomUnderSampler处理后
的数据集样本分类分布

# 使用SVM的权重调节处理不均衡样本
model_svm = SVC(class_weight='balanced')  # 创建SVC模型对象并指定类别权重
model_svm.fit(x, y)  # 输入x和y并训练模型

# 使用集成方法EasyEnsemble处理不均衡样本
model_EasyEnsemble = EasyEnsemble()  # 建立EasyEnsemble模型对象
x_EasyEnsemble_resampled, y_EasyEnsemble_resampled = model_EasyEnsemble.fit_
入数据并应用集成方法处理
print (x_EasyEnsemble_resampled.shape)  # 打印输出集成方法处理后的x样本集概况
print (y_EasyEnsemble_resampled.shape)  # 打印输出集成方法处理后的y标签集概况

# 抽取其中一份数据做审查
index_num = 1 # 设置抽样样本集索引
x_EasyEnsemble_resampled_t = pd.DataFrame(x_EasyEnsemble_resampled[index_num
['col1','col2','col3','col4','col5']) # 将数据转换为数据框并命名列名
y_EasyEnsemble_resampled_t = pd.DataFrame(y_EasyEnsemble_resampled[index_num
columns=['label']) # 将数据转换为数据框并命名列名
EasyEnsemble_resampled = pd.concat([x_EasyEnsemble_resampled_t, y_EasyEnsemb
列合并数据框
groupby_data_EasyEnsemble = EasyEnsemble_resampled.groupby('label').count()
label做分类汇总
print (groupby_data_EasyEnsemble) # 打印输出经过EasyEnsemble处理后的数据集样本
```

示例代码用空行分为6部分。

第一部分导入库。本示例中用到了第三方库imbalanced-learn实现主要的样本不均衡处理，而Pandas的引入主要用于解释和说明不同处理方法得到的结果集样本的分布情况，sklearn.svm中的SVC主要用于说明SVM如何在算法中自动调整分类权重。

第二部分导入数据文件。该过程中使用Pandas的read_table读取本地文件，为了更好的区别不同的列，通过names指定列名；对数据框做切片分割得到输入的x和目标变量y；通过Pandas的groupby（）方法按照label类做分类汇总，汇总方式使用count（）函数计数。输入原始数据集样本分类分布如下：

```
      col1  col2  col3  col4  col5
label
0.0    942   942   942   942   942
1.0     58    58    58    58    58
```

输出结果显示原始数据集中，正样本（label为1）的数量仅有58

个，占总样本量的5.8%，属于严重不均衡分布。

第三部分使用SMOTE方法进行过抽样处理。该过程中首先建立SMOTE模型对象，并直接使用fit_sample对数据进行过抽样处理，如果要获得有关smote的具体参数信息，可先使用fit（x，y）方法获得模型信息，并得到模型的不同参数和属性；从fit_sample方法分别得到对x和y过抽样处理后的数据集，将两份数据集转换为数据框，然后合并为一个整体数据框；最后通过Pandas提供的groupby（）方法按照label类做分类汇总，汇总方式是使用count（）函数计数。经过SMOTE处理后的数据集样本分类分布如下：

```
       col1   col2  col3  col4  col5
label
0.0     942   942   942   942   942
1.0     942   942   942   942   942
```

通过对比第二部分代码段的原始数据集返回结果发现，该结果中的正样本（label为1）的数量增加，并与负样本数量相同，均为942条，数据分类样本得到平衡。

第四部分使用RandomUnderSampler方法进行欠抽样处理。该过程与第三部分步骤完全相同，在此略过各模块介绍，用途都已在代码备注中注明。经过RandomUnderSampler处理后的数据集样本分类分布如下：

```
       col1   col2  col3  col4  col5
label
0.0      58    58    58    58    58
1.0      58    58    58    58    58
```

通过对比第二部分代码段的原始数据集返回的结果发现，该结果中的负样本（label为0）的数量减少，并跟正样本相同，均为58条，样本得到平衡。

第五部分使用SVM的权重调节处理不均衡样本。该过程主要通过配置SVC中的class_weight参数和值来处理样本权重，该参数可设置为字典、None或字符串balanced三种模式：

·字典：手动指定的不同类别的权重，例如{1:10，0:1}。

·None：代表类别的权重相同。

·balanced：代表算法将自动调整与输入数据中的类频率成反比的权重，具体公式为n_samples/（n_classes*np.bincount（y））， 程序示例中使用了该方法。

经过设置后，算法自动处理样本分类权重，无须用户做其他处理。要对新的数据集做预测，只要调用model_svm模型对象的predict方法即可。

第六部分使用集成方法EasyEnsemble处理不均衡样本。该方法的主要过程与其他imblearn方法类似，不同点在于集成方法返回的数据为三维数据，即将数据在原来的基础上新增了一个维度——“份数”，集成方法返回的数据x和y的形状为（10，116，5）和（10，116）。为了更详细地查看其中每一份数据，抽取其中一份数据做审查，得到的每份数据返回结果如下：

```
      col1   col2  col3  col4  col5
      label
      0.0      58    58    58    58    58
      1.0      58    58    58    58    58
```

通过对比第二部分代码段的原始数据集返回的结果，该结果中的负样本（label为0）的数量减少，并跟正样本相同，均为58条，样本集得到平衡。随后的应用中，可以通过循环读取每一份数据训练模型并得到结果，然后将10（x处理后返回的结果，通过形状返回的元组中的第一个数值，x_EasyEnsemble_resampled.shape[0]）份数据的结果通过一定方法进行汇总。

上述过程中，需要考虑的关键点是：如何针对不同的具体场景选择最合适的样本均衡解决方案。选择过程中既要考虑到每个类别样本的分布情况及总样本情况，又要考虑后续数据建模算法的适应性及整个数据模型计算的数据时效性。

本小节示例中，主要用了以下几个知识点：

·通过Pandas的read_table方法读取文本数据文件，并指定列名；

·对数据框做切片处理；

·通过Pandas提供的groupby（）方法配合count（）做分类汇总；

·使用imblearn.over_sampling中的SMOTE做过抽样处理；

·使用imblearn.under_sampling中的RandomUnderSampler做欠抽样处理；

·使用imblearn.ensemble中的EasyEnsemble做集成处理；

·使用sklearn.svm中的SVC自动调整算法对不同类别的权重设置。

第三方库imblearn提供了非常多的样本不均衡处理方法，限于篇幅此处无法一一介绍，建议读者自行安装并学习不同的用法。

3.5 如何解决运营数据源的冲突问题

多运营数据源冲突指的是来自于多个或同一个系统、环境、平台和工具的具有相同业务逻辑的数据其结果却不同。根据冲突的差异特征，可分为以下几种类型：

·数据类型冲突。同一数据对象的数据格式不同，例如会员注册时间这一字段其存储格式有日期、时间戳两种。

·数据结构冲突。对于同一个数据主体的描述结构有冲突，典型代表是关联主键ID值有不同的逻辑结构，导致后期多源数据匹配和关联变得极为复杂。

·记录粒度不同。对于订单记录的粒度可以存在以订单ID为基础的一条数据中，此时多个商品同时存在商品项目列中；也可以使用每条数据以“订单ID+商品ID”为唯一记录的形式，一个订单可以按商品明细拆分为多条记录。

·数据值域的定义不同。以订单来看，销售系统中可能包括提交订单、审核中、已审核通过、审核不通过四种状态；而库存系统中对于订单状态可能包括提交订单、审核通过、商品分拣、商品包装、商品出库、商品配送、配送成功。这些状态还仅是正常情况下的正向订单状态，即从商家发货到客户手中。几乎每个企业也都存在负向订单，通常产生于退换货的场景下，此时从会员发货到商家手中，这类的订单状态会更多。

·数据值不同。数据值的不同是数据冲突最重要也是最关键的问题所在，也是本节讨论的核心。上述其他的问题大多是格式或定义上的问题，不同的数据源通过ETL过程大多可以解决，但若出现数据值不同的问题却难以判断到底哪份数据是正确的。

这些情况在企业内部运营系统较多、数据源系统复杂时经常出现。需要注意的是，数据的冲突不仅来自于多个不同的系统和工具，有时候也来自于相同（或类似）的系统和工具。

3.5.1 为什么会出现多数据源的冲突

不同（或相同）系统、环境、平台和工具的数据冲突问题需要具体问题具体分析，现以3个实际数据冲突场景为例做详细说明。

1.内部工具与第三方工具数据冲突

为什么自己的网站统计分析工具所得数据跟代理商或广告媒体提供的广告数据有出入，有时差异特别大？

网站分析工具所得数据和广告媒体及代理商提供的数据存在差异是难以避免的，有以下几个关键影响因素：

·对比指标不同：广告媒体或代理商提供的指标一般是广告浏览量或广告点击量，而网站分析工具的指标则包括访问量、进入次数、页面浏览量等，不同的指标计算逻辑不同，虽然都可以统称为流量，但需要具体区分指标。

·测量的时机不同：即使广告媒体或代理商和网站分析工具都是以计数的逻辑进行测量，但前者统计的是用户点击及点击之前的数据，而后者统计的是点击之后的数据。不同的测量时机会导致数据的不一致。

·网络丢包的问题：即使前者采用点击量进行统计，后者采用到达量统计，理论上点击一次就会到达一次，但实际上，网络可能存在丢包的问题，所以二者不会完全相等。这一点无法避免。

·去重机制的问题：很多广告媒体对点击量进行统计时都有去重机制，主要用来屏蔽垃圾点击、恶意点击或蜘蛛点击等作弊流量，而网站分析工具不会主动去重。

·用户中途退出的问题：在很多时候，尤其是用户不小心点击了广告之后，用户的第一反应是立即关掉广告和对应跳转的页面，此时在广告媒体端会统计到数据，但网站分析工具一般无法统计到数据。

·页面跟踪加载问题：即使用户中途没有退出，在加载页面时，可能会由于用户没有等到网站分析工具的页面跟踪代码加载完成（大多数

跟踪代码在页面底部），用户就已经关掉页面，此时网站分析工具仍然无法测量到该数据。

·动机导致的数据夸大：大多数广告媒体是相对客观的，但不排除部分广告媒体存在恶意报量、夸大效果的问题，这点以展示类广告（例如门户、banner、弹窗等）为最。而网站分析工具则会客观统计数据，真实记录每一次的数据。

·其他因素：例如网站分析工具没有正确实施广告跟踪、没进行后台配置、在站外其他广告页面也出现相同广告参数的URL引用等。

综合来看，大多数情况下站外广告媒体或代理商的数据会高于网站分析工具提供的数据。如果按照正常统计逻辑，站外广告媒体或代理商会提供广告曝光量和点击量2个指标（当然也可以基于这两个指标额外提供了一个CTR，即点击通过率，简称点击率），网站分析工具则可以通过统计进入次数来跟点击量做比对，二者的比值通常称为到达率，即用户点击广告之后到达网站的比例。

2.内部同一个业务主体的同一类数据工具的数据测量冲突

为什么网站上不同的监测跟踪代码对同一个指标的监测数据不一致，例如跳出率？

通常，这种场景中由同一个指标产生的数据冲突会受多种因素影响：

·指标定义不同：很多时候不同系统对于同一个指标的定义不同，例如某些网站分析工具对于跳出率的定义是跳出/进入，而不是通用的跳出/访问。

·采集逻辑不同：某些情况下，受限于底层代码实施的差异性，同一个指标采集时会直接受到影响。例如A系统在所有页面都有代码部署，而B系统缺少了帮助中心、活动页等代码从而导致采集到的数据缺失。即使都是相同的页面覆盖，不同的采集逻辑、触发机制、异常处理等采集端的逻辑定义都会影响数据采集本身。

·系统过滤规则不同：复杂的系统都允许用户自定义过滤器，用于

过滤或筛选出特定的数据到账户中，例如白名单、黑名单、IP地址、域名规则等。后端的规律规则不同，必然导致数据值的差异。

·更新时间不同：不同的系统其数据更新周期可能不同。例如某些工具可以在1小时内完成所有数据的全量更新，而其他工具可能需要1～2个小时才能完成。

·监测位置不同：位置对于监测数据的影响主要体现在代码触发时机方面。通常，放在页面顶部的监测代码将比放在底部的监测代码监测到的数据多，这是由于页面默认的加载是自上而下的，处于页面底部的代码会由于页面没有加载完成就关闭或离开页面，而导致无法统计到数据的情况，但此时页面顶部的跟踪代码已经统计到该数据。

3.内部同一个业务主体的不同数据工具的数据测量冲突

为什么销售系统跟会员管理系统统计的订单数据不一致？

二者均为企业内部的订单统计数据，之所以存在差异，通常因为以下几个方面：

·订单来源差异：两个工具对于通常意义下的销售数据来源都有覆盖，例如网站、移动端、线下门店，而对于某些订单来源可能只有特定系统才进行统计。例如，坐席订单产生于呼叫中心，可能不走销售系统；大客户团购可能属于销售系统管理，也反映不到会员管理系统；企业分公司之间的商品分拨以及订单转移属于内部流转，也不走会员管理系统等。

·特殊商品订单跟踪：对实物订单的跟踪比较常见，但某些商品的订单跟踪却有单独的流程和体系，例如虚拟货币、游戏、充值等，这些数据可能无法完整反馈到所有系统中。

·订单状态差异：对于销售系统和会员管理系统而言，二者的订单状态往往不同。销售系统对于订单流转的划分更为详细，基于不同的订单状态查询到的结果可能不一样。

·数据同步问题：销售系统需要与外部第三方平台（例如支付系统）集成时，不同的内部子系统的数据可能无法及时同步，基于不同步

的数据得到的结果会有差异。

·内部系统拆单问题：当企业运营模式复杂时，尤其涉及自营、联营店铺等多种订单形式时，通常在1个客户订单下，会分拆为多个子订单，不同的子订单会对应各自店铺不同的订单。对于销售系统而言，这些信息会与库存系统连接而获得完整信息，而会员管理系统可能无法获得后续拆单信息。当基于主订单和子订单分别（以及混合）查询时得到的结果往往不同。

3.5.2 如何应对多数据源的冲突问题

数据冲突的具体应对没有统一标准，具体取决于后期应用场景和需求，总体上讲应对方法分为3种：

1）消除冲突并形成一份唯一数据。如果是做全局性的汇总统计，那么需要通过一定方法消除冲突以便呈报一份数据；如果同时呈报多份数据往往会让管理者莫衷一是，而且会对数据工作的价值产生怀疑。例如：企业级（非部门或中心级别）的数据日报，每天都要看销售报表，来自销售、会员、物流、呼叫中心、仓储等体系的数据一般都会呈现有关销售额的指标，此时需要整合并统一口径形成唯一的销售额数据。

2）不消除冲突也不作任何处理。如果数据用于数据建模，通常使用的是细粒度的数据而非整体汇总数据，冲突数据记录往往占比很小，因此可以忽略冲突。具体使用哪些数据都不会对后续建模结果造成显著性影响。

3）不消除冲突但是使用全部冲突数据。如果在做基于整体的流程性统计分析时，会用到不同业务流程的不同数据，此时不同的指标之间会具有较好的漏斗转化效应，基于此可以做全流程的漏斗转化分析。此时的漏斗是一个非封闭性的环节，例如广告曝光→用户到达网站→加入购物车→订单提交→形成有效订单→订单配送→订单签收。

即使不消除数据冲突，我们也一定要谨慎对待这种情况，绝不能听之任之，更不能对数据冲突视而不见。对于冲突的数据而言，通常需要关注以下两方面：

1）差异性：对相同实体在相同逻辑下的数据冲突（差异）应该在5%以内，如果条件不具备的可以放宽到10%。尤其对于数据准确度要求较高的订单、支付等更要谨慎分析、严格把关，过大的差异说明在跟踪、采集、同步、计算和统计逻辑上一定有问题。

2）稳定性：由于各种主客观原因，数据冲突（差异）无法完全消除时，需要确保多源数据的差异性相对稳定，不能出现数据差异时高时低甚至相反分布的状态。否则可能意味着其中至少一份数据出现严重问

题，这也是数据校验的基本原则之一。

3.6 数据化运营要抽样还是全量数据

抽样是从整体样本中通过一定的方法选择一部分样本，抽样是数据处理的基本步骤之一，也是科学实验、质量检验、社会调查普遍采用的一种经济有效的工作和研究方法。

3.6.1 什么时候需要抽样

抽样工作在数据获取较少或处理大量数据比较困难的时代非常流行，这主要有以下几方面原因：

·数据计算资源不足。计算机软硬件的限制是导致抽样产生的基本原因之一，尤其是在数据密集的生物、科学工程等领域，不抽样往往无法对海量数据进行计算。

·数据采集限制。很多时候抽样从数据采集端便已经开始，例如做社会调查必须采用抽样方法进行研究，因为根本无法对所有人群做调查。

·时效性要求。抽样带来的以局部反应全局的思路，如果方法正确，可以以极小的数据计算量来实现对整体数据的统计分析，在时效性上会大大增强。

如果存在上述条件限制或有类似强制性要求，那么抽样工作仍然必不可少。但是在当前数据化运营的大背景下，数据计算资源充足、数据采集端可以采集更多的数据并且可以通过多种方式满足时效性的要求。抽样工作是否就没有必要了？其实不是的，即使上述限制条件都满足，还有很多场景依然需要通过抽样方法来解决具体问题：

·通过抽样来实现快速的概念验证。数据工作中可能会包括创新性或常识性项目，对于这类项目进行快速验证、迭代和交付结论往往是概念验证的关键，通过抽样方法带来的不仅是计算效率的提升，还有前期数据准备、数据预处理、算法实现等各个方面的开发，以及服务器、硬件的配套方案的部署等内容的可行性、简单性和可操作性。

·通过抽样来解决样本不均衡问题。在3.4节中，我们提到了通过欠抽样、过抽样以及组合/集成的方法解决不均衡的问题，这个过程就用到了抽样方法。

·无法实现对全部样本覆盖的数据化运营场景。典型场景包括市场研究、客户线下调研分析、产品品质检验、用户电话满意度调查等，这些场景下无法实现对所有样本的采集、分析、处理和建模。

·定性分析的工作需要。在定性分析工作中，通常不需要定量分析时的完整假设、精确数据和复杂统计分析过程，更多的是采用访问、观察和文献法收集资料并通过主观理解和定性分析找到问题答案，该过程中主要依靠人自身的能力而非密集的计算机能力来完成研究工作。如果不使用抽样方法，那么定性分析将很难完成。

3.6.2 如何进行抽样

抽样方法从整体上分为非概率抽样和概率抽样两种。非概率抽样不是按照等概率的原则进行抽样，而是根据人类的主观经验和状态进行判断；概率抽样则是以数学概率论为基础，按照随机的原则进行抽样。本节介绍的抽样方法属于概率抽样。

（1）简单随机抽样

该抽样方法是按等概率原则直接从总中抽取n个样本，这种随机样本方法简单，易于操作；但是它并不能保证样本能完美代表总体，这种抽样的基本前提是所有样本个体都是等概率分布的，但真实情况却是很多数样本都不是或无法判断是否是等概率分布的。在简单随机抽样中，得到的结果是不重复的样本集，还可以使用有放回的简单随机抽样，这样得到的样本集中会存在重复数据。该方法适用于个体分布均匀的场景。

（2）等距抽样

等距抽样是先将总体中的每个个体按顺序编号，然后计算出抽样间隔，再按照固定抽样间隔抽取个体。这种操作方法易于理解、简便易行，但当总体样本的分布呈现明显的分布规律时容易产生偏差，例如增减趋势、周期性规律等。该方法适用于个体分布均匀或呈现明显的均匀分布规律，无明显趋势或周期性规律的数据。

（3）分层抽样

分层抽样是先将所有个体样本按照某种特征划分为几个类别，然后从每个类别中使用随机抽样或等距抽样的方法选择个体组成样本。这种操作方法能明显降低抽样误差，并且便于针对不同类别的数据样本进行单独研究，因此是一种较好的实现方法。该方法适用于带有分类逻辑的属性、标签等特征的数据。

（4）整群抽样

整群抽样是先将所有样本分为几个小群体集，然后随机抽样几个小

群体集来代表总体。这种操作方法跟之前的3种方法的差异点在于该方法抽取的是小群体集，而不是每个数据个体本身。该方法虽然简单易行，但是样本的分布受限于小群体集的划分，抽样误差较大。这种方法适用于小群体集的特征差异比较小，并且对划分小群体集有更高要求。

3.6.3 抽样需要注意的几个问题

1.数据抽样要能反映运营背景

数据能正确反映运营背景看起来非常简单，但实际上需要数据工作者对于运营环节和流程非常熟悉才有可能实现。以下是常见的抽样不能反映运营背景的情况：

·数据时效性问题：例如使用过时的数据（例如1年前的数据）来分析现在的运营状态。

·缺少关键因素数据：没有将运营分析涉及的主要因素所产生的数据放到抽样数据中，导致无法根据主要因素产生有效结论，模型效果差，例如抽样中没有覆盖大型促销活动带来的销售增长。

·不具备业务随机性：有意/无意多抽取或覆盖特定数据场景，使得数据明显趋向于特定分布规律，例如在做社会调查时使用北京市的抽样数据来代表全国。

·没有考虑业务增长性：例如在成长型公司中，公司的发展不都是呈现线性趋势，很多时候会产生指数趋势。这时需要根据这种趋势来使业务满足不同增长阶段的分析需求，而不只是集中于增长爆发区间。

·没有考虑数据来源的多样性：只选择某一来源的数据做抽样，使得数据的分布受限于数据源。例如在做各分公司的销售分析时，仅将北方大区的数据纳入其中做抽样，而忽视了其他大区的数据，其结果必然有所偏颇。

·业务数据可行性问题：很多时候，由于受到经费、权限、职责等方面的限制，在数据抽样方面无法按照数据工作要求来执行，此时要根据运营实际情况调整。这点往往被很多数据工作者忽视。

2.数据抽样要能满足数据分析和建模需求

数据抽样必须兼顾后续的其他数据处理工作，尤其是分析和建模需求。这时需要注意以下几个方面：

（1）抽样样本量的问题

对于大多数数据分析建模而言，数据规模越大，模型拟合结果越准确。但到底如何定义数据量的大小，笔者根据不同类型的数据应用总结为以下几个维度：

·以时间为维度分布的，至少包含一个能满足预测的完整业务周期。例如做月度销售预测的，至少包含12个月的数据；做日销售预测的，至少包含30天的数据，如果一天中包含特定周期，则需要重复多个周期。同时，时间性特征的要充分考虑季节性、波动性、节假日等特殊规律，这些都要尽量包含在抽样数据中。

·做预测（包含分类和回归）分析建模的，需要考虑特征数量和特征值域（非数值型）的分布，通常数据记录数要同时是特征数量和特征值域的100倍以上。例如数据集有5个特征，假如每个特征有2个值域，那么数据记录数需要至少在1000（100×5×2）条以上。

·做关联规则分析建模的，根据关联前后项的数量（每个前项或后项可包含多个要关联的主体，例如品牌+商品+价格关联），每个主体需要至少1000条数据。例如只做单品销售关联，那么单品的销售记录需要在1000条以上；如果要同时做单品+品牌的关联，那么需要至少2000条数据。

·对于异常检测类分析建模的，无论是监督式还是非监督式建模，由于异常数据本来就是小概率分布的，因此异常数据记录一般越多越好。

以上的数据记录数不是固定的，在实际工作时，如果没有特定时间要求，笔者一般会选择一个适中的样本量做分析，此时应综合考虑特征数、特征值域分布数、模型算法适应性、建模需求等；如果是面向机器计算的工作项目，一般会选择尽量多的数据参与计算，而有关算法实时性和效率的问题会让技术和运维人员配合实现，例如提高服务器配置、扩大分布式集群规模、优化底层程序代码、使用实施计算的引擎和机制等。

（2）抽样样本在不同类别中的分布问题

做分类分析建模问题时，不同类别下的数据样本需要均衡分布，有关数据样本均衡分布的更多话题，参见3.4节。

抽样样本能准确代表全部整体特征：

·非数值型的特征值域（例如各值频数相对比例、值域范围等）分布需要与总体一致。

·数值型特征的数据分布区间和各个统计量（如均值、方差、偏度等）需要与整体数据分布区间一致。

·缺失值、异常值、重复值等特殊数据的分布要与整体数据分布一致。

异常检测类数据的处理：

·对于异常检测类的应用要包含全部异常样本。对于异常检测类的分析建模，本来异常数据就非常稀少，因此抽样时要优先将异常数据包含进去。

·对于需要去除非业务因素的数据异常，如果有类别特征需要跟类别特征分布一致；如果没有类别特征，属于非监督式的学习，则需要与整体分布一致。

3.6.4 代码实操：Python数据抽样

本示例中，将使用random包及自定义代码实现抽样处理。数据源文件data2.txt、data3.txt和data4.txt位于“附件-chapter3”中，默认工作目录为“附件-chapter3”（如果不是，请切换到该目录下，否则会报类似“IOError:File data3.txt does not exist”的错误）。完整代码如下：

```
import random  # 导入标准库
import numpy as np  # 导入第三方库

# 简单随机抽样
data = np.loadtxt('data3.txt')  # 导入普通数据文件
data_sample = random.sample(data, 2000)  # 随机抽取2000个样本
print (data_sample[:2])  # 打印输出前2条数据
print (len(data_sample))  # 打印输出抽样样本量

# 等距抽样
data = np.loadtxt('data3.txt')  # 导入普通数据文件
sample_count = 2000  # 指定抽样数量
record_count = data.shape[0]  # 获取最大样本量
width = record_count / sample_count  # 计算抽样间距
data_sample = []  # 初始化空白列表，用来存放抽样结果数据
i = 0  # 自增计数以得到对应索引值
while len(data_sample) <= sample_count and i * width <= record_count - 1:  # 样本量小于等于指定抽样数量并且矩阵索引在有效范围内时
    data_sample.append(data[i * width])  # 新增样本
    i += 1  # 自增长
print (data_sample[:2])  # 打印输出前2条数据
print (len(data_sample))  # 打印输出样本数量

# 分层抽样
# 导入有标签的数据文件
data2 = np.loadtxt('data2.txt')  # 导入带有分层逻辑的数据
each_sample_count = 200  # 定义每个分层的抽样数量
label_data_unique = np.unique(data2[:, -1])  # 定义分层值域
sample_list = []  # 定义空列表，用于存放临时分层数据
sample_data = []  # 定义空列表，用于存放最终抽样数据
sample_dict = {}  # 定义空字典，用来显示各分层样本数量
for label_data in label_data_unique:  # 遍历每个分层标签
    for data_tmp in data2:  # 读取每条数据
        if data_tmp[-1] == label_data:  # 如果数据最后一列等于标签
            sample_list.append(data_tmp)  # 将数据加入到分层数据中
    each_sample_data = random.sample(sample_list, each_sample_count)  # 对每层数据都随机抽样
    sample_data.extend(each_sample_data)  # 将抽样数据追加到总体样本集
    sample_dict[label_data] = len(each_sample_data)  # 样本集统计结果
print (sample_dict)  # 打印输出样本集统计结果

# 整群抽样
data3 = np.loadtxt('data4.txt')  # 导入已经划分好整群的数据集
label_data_unique = np.unique(data3[:, -1])  # 定义整群标签值域
print (label_data_unique)  # 打印输出所有整群标签
sample_label = random.sample(label_data_unique, 2)  # 随机抽取2个整群
sample_data = []  # 定义空列表，用来存储最终抽样数据
```

```
for each_label in sample_label:  # 遍历每个整群标签值域
    for data_tmp in data3:  # 遍历每个样本
        if data_tmp[-1] == each_label:  # 判断样本是否属于抽样整群
            sample_data.append(data_tmp)  # 样本添加到最终抽样数据集
print (sample_label)  # 打印输出样本整群标签
print (len(sample_data))  # 打印输出总抽样数据记录条数
```

整个示例代码用空行分为5部分。

第一部分导入需要的库，这里用到了Python内置标准库random以及第三方库Numpy，前者用于做随机抽样，后者用于读取文件并作数据切片使用。

第二部分实现了简单随机抽样。首先通过Numpy的loadtxt方法读取数据文件；然后使用Random库中的sample方法做数据抽样，指定抽样总体数据为data，抽样数量为2000；最后打印输出前2条数据和总抽样样本量。返回结果如下：

```
[array([ -7.66339337, -10.13735724,   5.57419663,   3.27726596,   5.06304001
```

第三部分实现了等距抽样。

首先使用Numpy的loadtxt方法读取数据文件；然后指定抽样样本量为2000，并通过读取原始数据的形状找到最大样本量边界，这可以用来作为循环的终止条件之一；接着通过最大样本量除以抽样样本量得到抽样间距；

建立一个空列表用于存储最终抽样结果数据，通过一个变量i做循环增长并用来做索引递增，然后进入到抽样条件判断过程：

·当样本量小于等于指定抽样数量并且矩阵索引在有效范围内时，若要做处理，需要注意的是索引从0开始，因此需要用最大数量值减去1作为循环边界，否则会报索引溢出错误。

·通过列表的append方法不断追加通过间距得到的新增样本，在本节后面的方法中还会提到用于列表追加的extend方法，前者用于每次追加1个元素，后者用于批量追加多个元素。

·i+=1指的是每次循环都增加1，可以写成i=i+1。

·最后打印输出前2条数据和抽样样本量。返回结果如下：

```
[array([-3.08057779,  8.09020329,  2.02732982,  2.92353937, -6.06318211]), a
```

第四部分实现了分层抽样。

首先使用Numpy的loadtxt方法导入带有分层逻辑的数据。在该示例中，读取的数据文件中包含了分类标签，放在最后一列。该列分类标签用于做分层抽样的标志。

接着通过unique方法获取分层（分类标签）的值域，用于后续做循环处理。

然后分别定义了用于存放临时分层数据、最终抽样数据、显示各分层样本数量的空列表和空字典。

下面进入正式的主循环过程，实现分层抽样：

·遍历每个分层标签，用来做数据的分层划分，数据一共分为2类标签（0和1）；

·读取每条数据并判断其标签是否属于特定的分层标签，如果是则将数据加入各分层数据列表中。

·当每个分层标签处理完成后会得到该分层标签下的所有数据，此时使用Python内置的Random库的sample方法进行抽样。由于抽样结果是一个列表，因此这里使用extend（而不是append）批量追加到最终抽样数据列表中。然后对每个分层标签得到的样本数量，通过len方法对列表长度进行统计，并打印输出与各个分层对应的样本数量。结果是每个分层都按照指定数量抽取样本，具体如下：

```
{0.0: 200, 1.0: 200}
```

第五部分实现了整群抽样。

首先使用Numpy的loadtxt方法导入已经划分好整群的数据集。在该示例中，读取的数据文件中的最后一列存放了不同整群的标识，整群一共被划分为4个群组，标识分别为0/1/2/3。

接着通过unique方法获取整群标签的值域，用于基于整群的抽样。打印输出结果如下：

```
[ 0.  1.  2.  3.]
```

然后使用Random的sample方法从整群标签中进行抽样，这里定义抽取2个整群。

最后对所有属于抽取到的整群下的数据进行读取和追加并得到最终样本集，打印输出样本集的整群标签和总样本数量的结果如下：

```
[3.0, 1.0]
502
```

由于是随机概率抽样，因此读者使用代码抽取到的样本很可能跟书中示例的不一致，这属于正常现象；另外，读者多次随机抽样程序，也可能得到不一样的结果。

上述过程中，需要考虑的关键点是：如何根据不同的数据特点、建模需求、业务背景综合考虑抽样方法，得到最适合的结果。

本小节示例中，主要用了如下几个知识点：

·使用Numpy的loadtxt方法读取数据文件；

·使用内置标准库Random库中的sample方法做数据抽样；

·对列表通过索引做截取、通过len方法做长度统计、通过append和extend做追加等操作；

·字典赋值操作；

·使用Numpy的unique方法获得唯一值；

·通过for和while循环，遍历一个可迭代的对象；

·if条件语句的使用，尤其是单条件和多条件判断。

3.7 解决运营数据的共线性问题

所谓共线性（又称多重共线性）问题指的是输入的自变量之间存在较高的线性相关度。共线性问题会导致回归模型的稳定性和准确性大大降低，另外，过多无关的维度参与计算也会浪费计算资源和时间。

共线性问题是否常见取决于具体业务场景，常见的具有明显的共线性的维度或变量包括：

·访问量和页面浏览量；

·页面浏览量和访问时间；

·订单量和销售额；

·订单量和转化率；

·促销费用和销售额；

·网络展示广告费用和访客数。

导致出现变量间共线性的原因可能包括：

·数据样本不够，导致共线性存在偶然性，这其实反映了缺少数据对于数据建模的影响，共线性仅仅是影响的一部分。

·多个变量都基于时间有共同或相反的演变趋势，例如春节期间的网络销售量和销售额都相对于正常时间有下降趋势。

·多个变量间存在一定的推移关系，但总体上变量间的趋势一致，只是发生的时间点不一致，例如品牌广告费用和销售额之间，通常是品牌广告先进行大范围的曝光和信息推送，经过一定时间传播之后，销售额才能爆发出来。

·多个变量间存在近似线性的关系。例如，如果用y代表访客数，用x代表展示广告费用，那么二者的关系很可能是y=2*x+b，即每投放1元钱，可以带来大概2～3个访客。

3.7.1 如何检验共线性

共线性一般通过容忍度、方差膨胀因子、特征值这几个特征数据来做判断：

·容忍度（Tolerance）：容忍度是每个自变量作为因变量对其他自变量进行回归建模时得到的残差比例，大小用1减得到的决定系数来表示。容忍度的值介于0.1和1之间，如果值越小，说明这个自变量与其他自变量间越可能存在共线性问题。

·方差膨胀因子（Variance Inflation Factor，VIF）：VIF是容忍度的倒数，值越大则共线性问题越明显，通常以10作为判断边界。当VIF＜10，不存在多重共线性；当10≤VIF＜100，存在较强的多重共线性；当VIF≥100，存在严重多重共线性。

·特征值（Eigenvalue）：该方法实际上就是对自变量进行主成分分析，如果多个维度的特征值等于0，则可能有比较严重的共线性。

除此以外，还可以使用相关系数辅助判断，当相关系数R＞0.8时就表示可能存在较强的相关性，有关相关性的更多话题，会在3.8节中探讨。通常这些方法得到的结果都是相关的，即满足其中某个特征后，其他特征也基本满足。因此可以通过多种方法共同验证。

3.7.2 解决共线性的5种常用方法

解决共线性的5种常用方法如下：

（1）增大样本量

通过增加样本量，来消除由于数据量不足而出现的偶然共线性现象，在可行的前提下这种方法是需要优先考虑的；但即使增加了样本量，也可能无法解决共线性问题，原因是很可能变量间确实存在这个问题。

（2）岭回归法（Ridge Regression）

岭回归分析是一种专用于共线性问题的有偏估计回归方法，实质上是一种改良的最小二乘估计法。它通过放弃最小二乘法的无偏性，以损失部分信息、降低精度为代价来获得更实际和可靠性更强的回归系数。因此岭回归在存在较强共线性的回归应用中较为常用。

（3）逐步回归法（Stepwise Regression）

逐步回归法是每次引入一个自变量并进行统计检验，然后逐步引入其他变量，同时对所有变量的回归系数进行检验。如果原来引入的变量由于后面变量的引入而变得不再显著，那么就将其剔除，逐步得到最优回归方程。

（4）主成分回归（Principal Components Regression）

通过主成分分析，将原始参与建模的变量转换为少数几个主成分，每个主成分是原变量的线性组合，然后基于主成分做回归分析，这样也可以在不丢失重要数据特征的前提下避开共线性问题。

（5）人工去除

直接结合人工经验，对参与回归模型计算的自变量进行删减，也是一个较为常用的方法，但这种方法需要操作者对于业务、模型和数据都有相对深入的理解，这样才有可能做出正确的操作。从专业角度分析，

如果缺少上述三点中的任何一点，那么人工去除的方式都有可能产生偏差，导致结果不准确。

要完全解决共线性问题是不可能的，因为任何事物之间都存在一定的联系，在解决共线性问题的相关话题中，我们只是解决其中严重的共线性问题，而非全部共线性问题。

3.7.3 代码实操：Python处理共线性问题

本示例中，将通过sklearn进行共线性处理。源文件data5.txt位于“附件-chapter3”中，默认工作目录为“附件-chapter3”（如果不是，请切换到该目录下，否则会报错“IOError:File data5.txt does not exist”）。

在自动化工作中（尤其是以Python为代表的智能化数据工作），通常不会通过人工的方法参与算法结果观察、调优、选择等。同样，在解决共线性的方法中，通过程序的方式自动选择或规避是最佳解决方法。在本示例中，将主要使用岭回归和主成分回归实现共线性问题。

完整代码如下：

```
# 导入相关库
import numpy as np
from sklearn.linear_model import Ridge
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression

# 读取数据
data = np.loadtxt('data5.txt', delimiter='\t')  # 读取数据文件
x = data[:, :-1]  # 切分自变量
y = data[:, -1]  # 切分预测变量

# 使用岭回归算法进行回归分析
model_ridge = Ridge(alpha=1.0)  # 建立岭回归模型对象
model_ridge.fit(x, y)  # 输入x/y训练模型
print (model_ridge.coef_)  # 打印输出自变量的系数
print (model_ridge.intercept_)  # 打印输出截距

# 使用主成分回归进行回归分析
model_pca = PCA()  # 建立PCA模型对象
data_pca = model_pca.fit_transform(x)  # 将x进行主成分分析
ratio_cumsm = np.cumsum(model_pca.explained_variance_ratio_)  # 得到所有主成分方差占比的累积数据
print (ratio_cumsm)  # 打印输出所有主成分方差占比累积
rule_index = np.where(ratio_cumsm > 0.8)  # 获取方差占比超过0.8的所有索引值
min_index = rule_index[0][0]  # 获取最小索引值
data_pca_result = data_pca[:, :min_index + 1]  # 根据最小索引值提取主成分
model_liner = LinearRegression()  # 建立回归模型对象
model_liner.fit(data_pca_result, y)  # 输入主成分数据和预测变量y并训练模型
print (model_liner.coef_)  # 打印输出自变量的系数
print (model_liner.intercept_)  # 打印输出截距
```

上述示例代码用空行分为4部分。

第一部分导入相关库。示例中用到了Numpy、Sklearn中的

Ridge（岭回归）、PCA（主成分分析）和LinearRegression（普通线性回归）。由于本书用到的Python库中没有直接集成主成分回归方法，因此这里将通过PCA+LinearRegression的形式组合实现。

第二部分导入数据。使用Pumpy的loadtxt方法读取数据文件，数据文件以tab分隔，共1000条数据，有9个自变量和1个因变量。因变量处于最后一列，使用Numpy矩阵进行数据切分。

第三部分使用岭回归算法进行回归分析。该过程中，先建立岭回归模型对象，指定alpha值为0.1，接着通过fit方法将x和y分别输入模型做训练，然后打印输出回归方程中自变量的系数和截距。结果如下：

```
[  8.50164360e+01  -1.18330186e-03   9.80792921e-04  -8.54201056e-04
   2.10489064e-05   2.20180449e-04  -3.00990875e-06  -9.30084240e-06
  -2.84498824e-08]
-7443.98652868
```

上述结果中包含了各个输入变量的系数及截距，假设因变量为y，自变量为x1/x2/...x9，那么该方程可以写成：

y=85.016436*x1+(-0.001183301 86)*x2+0.000980792921*x3

+(-0.000854201056)*x4+0.000021048906 4*x5+0.000220180449*x6

+(-0.00000300990875)*x7+(-0.0000093008424)*x8+
(-0.0000000284498824)*x9

-7443.98652868

由于其中后面几个变量的系数实在太小，为了清晰展示方程式，因此保留的小数点位数比较多。

第四部分使用主成分回归进行回归分析。该过程主要分为以下步骤：

·先建立PCA模型对象，然后使用fit_transform方法直接进行主成分转换。这里没有指定具体主成分的数量，原因是在主成分的方差贡献率

之前，无法知晓到底选择多少个合适，后续会通过指定阈值的方式自动实现成分选择。

·通过PCA模型对象的explained_variance_ratio_得到各个主成分的方差贡献率（方差占比），然后使用Numpy的cumsum方法计算累积占比，得到如下输出结果：

```
[ 0.9028      0.98570494  0.99957412  0.99995908  0.99999562  0.99999939
  0.99999999  1.          1.        ]
```

直观上分析，前2个主成分基本已经可代表所有成分参与模型计算。但是如果自动化实现，我们不可能每次先中断程序并打印出结果，再使用人工判断。因此我们需要一个阈值，当方差贡献达到阈值时，就自动选择前n个主成分作为最终转换维度。

·通过使用Numpy的where方法找到主成分方差占比累积＞0.8的值索引，通常0.8就已经能代表大部分的数据特征，本示例较为特殊，第一个主成分就已经非常明显，该方法返回的是一个元组，从元组中得到最小的索引值（越往后累积占比越大，索引值也越大，因此第一个符合条件的索引就是我们要取出的最后一个主成分）。

·在获得最小索引值之后，根据最小索引值提取主成分。切片时在索引列值上加1是由于默认是不包含索引值右侧的，例如[0:1]只取出第0列，而不是我们需要的0和1两列。

·接着进入线性回归的环节。跟岭回归的步骤类似，建立回归模型对象并输入x和y做模型训练，最后打印输出回归方程中自变量的系数和截距。结果如下：

```
[  1.26262171e-05]
1058.52726
```

上述结果中包含了输入变量的系数以及截距，假设因变量为y，满足自变量方差占比大于0.9的主成分为第一个主成分，假设其为x1，那么该方程可以写成：

y=0.0000126262171*x1+1058.52726

此时的x1跟岭回归中的x1含义不同：岭回归的x1是原始数据文件中第一个变量，而这里的x1是第一个主成分——原始数据文件中自变量的一个线性组合。

3.8 有关相关性分析的混沌

相关性分析是指对多个具备相关关系的变量进行分析，从而衡量变量间的相关程度或密切程度。相关性可以应用到所有数据的分析过程中，任何事物之间都是存在一定的联系。相关性用R（相关系数）表示，R的取值范围是[-1，1]。

3.8.1 相关和因果是一回事吗

相关性不等于因果，用x1和x2作为两个变量进行解释，相关意味着x1和x2是逻辑上的并列相关关系，而因果联系可以解释为因为x1所以x2（或因为x2所以x1）的逻辑关系，二者是完全不同的。

用一个运营示例来说明二者的关系：做商品促销活动时，通常都会以较低的价格进行销售，以此来实现较高的商品销量；随着商品销售的提升，也给线下物流配送体系带来了更大的压力，在该过程中通常会导致商品破损量的增加。

本案例中，商品低价与破损量增加并不是因果关系，即不能说因为商品价格低所以商品破损量增加；二者的真实关系是基于促销这个大背景下，低价和破损量都是基于促销产生。

相关性的真实价值不是用来分析“为什么”，而是通过相关性来描述无法解释的问题背后真正成因。相关性的真正价值是能知道“是什么”，即无论通过何种因素对结果产生影响，最终出现的规律就是二者会一起增加、降低等。

仍然是上面的案例，通过相关性分析我们可以知道，商品价格低和破损量增加是相伴发生的，这意味着当价格低的时候（通常是做销售活动，也有可能产品质量问题、物流配送问题、包装问题等），我们就想到破损量可能也会增加；但是到底由什么导致的破损量增加，是无法通过相关性得到的。

3.8.2 相关系数低就是不相关吗

R（相关系数）低就是不相关吗？其实不是。

首先，R的取值可以为负，R=-0.8代表的相关性要高于R=0.5，负相关只是意味着两个变量的增长趋势相反。因此需要看R的绝对值来判断相关性的强弱。

其次，即使R的绝对值低，也不一定说明变量间的相关性低，原因是相关性衡量的仅仅是变量间的线性相关关系，变量间除了线性关系外，还包括指数关系、多项式关系、幂关系等，这些非线性相关的相关性不在R（相关性分析）衡量范围之内。

3.8.3 代码实操：Python相关性分析

本示例中，将使用Numpy进行相关性分析。源文件data5.txt位于“附件-chapter3”中，默认工作目录为“附件-chapter3”（如果不是，请切换到该目录下，否则会报错“IOError:File data5.txt does not exist”）。

```
import numpy as np  # 导入库
data = np.loadtxt('data5.txt', delimiter='\t')  # 读取数据文件
x = data[:, :-1]  # 切分自变量
correlation_matrix = np.corrcoef(x, rowvar=0)  # 相关性分析
print (correlation_matrix.round(2))  # 打印输出相关性结果
```

示例中实现过程如下：

1）导入Numpy库；

2）使用Numpy的loadtxt方法读取数据文件，数据文件以tab分隔；

3）矩阵切片，切分出自变量用来做相关性分析；

4）使用Numpy的corrcoef方法做相关性分析，通过参数rowvar=0控制对列做分析；

5）打印输出相关性矩阵，使用round方法保留2位小数，结果如下：

```
[[ 1.   -0.04  0.27 -0.05  0.21 -0.05  0.19 -0.03 -0.02]
 [-0.04  1.   -0.01  0.73 -0.01  0.62  0.    0.48  0.51]
 [ 0.27 -0.01  1.   -0.01  0.72  0.    0.65  0.01  0.02]
 [-0.05  0.73 -0.01  1.    0.01  0.88  0.01  0.7   0.72]
 [ 0.21 -0.01  0.72  0.01  1.    0.02  0.91  0.03  0.03]
 [-0.05  0.62  0.    0.88  0.02  1.    0.03  0.83  0.82]
 [ 0.19  0.    0.65  0.01  0.91  0.03  1.    0.03  0.03]
 [-0.03  0.48  0.01  0.7   0.03  0.83  0.03  1.    0.71]
 [-0.02  0.51  0.02  0.72  0.03  0.82  0.03  0.71  1.  ]]
```

相关性矩阵的左侧和顶部都是相对变量，从左到右、从上到下依次是列1到列9。从结果看出：

·第5列和第7列相关性最高，系数达到0.91；

·第4列和第6列相关性较高，系数达到0.88；

·第8列和第6列相关性较高，系数达到0.83。

上述过程中，需要考虑的关键点是：

·如何理解相关性和因果关系的差异以及如何应用相关性。相关性分析除了可以用来分析不同变量间的相关伴生关系外，也可以用来做多重共线性检验，有关共线性的问题请参照3.7节。

本小节示例中，主要用了如下几个知识点：

·通过Numpy的loadtxt方法读取文本数据文件，并指定分隔符；

·对Numpy矩阵做切块处理；

·使用Numpy中的corrcoef做相关性分析；

·使用round方法保留2位小数。

3.9 标准化，让运营数据落入相同的范围

数据标准化是一个常用的数据预处理操作，目的是处理不同规模和量纲的数据，使其缩放到相同的数据区间和范围，以减少规模、特征、分布差异等对模型的影响。除了用作模型计算，标准化后的数据还具有了直接计算并生成复合指标的意义，是加权指标的必要步骤。

3.9.1 实现中心化和正态分布的Z-Score

Z-Score标准化是基于原始数据的均值和标准差进行的标准化，假设原转换的数据为x，新数据为x'，那么x'=（x-mean）/std，其中mean和std为x所在列的均值和标准差。

这种方法适合大多数类型的数据，也是很多工具的默认标准化方法。标准化之后的数据是以0为均值，方差为1的正态分布。但是Z-Score方法是一种中心化方法，会改变原有数据的分布结构，不适合用于对稀疏数据做处理。

在很多时候，数据集会存在稀疏性特征，表现为标准差小、并有很多元素的值为0，最常见的稀疏数据集是用来做协同过滤的数据集，绝大部分的数据都是0，仅有少部分数据为1。对稀疏数据做标准化，不能采用中心化的方式，否则会破坏稀疏数据的结构。

3.9.2 实现归一化的Max-Min

Max-Min标准化方法是对原始数据进行线性变换，假设原转换的数据为x，新数据为x'，那么x'=（x-min）/（max-min），其中min和max为x所在列的最小值和最大值。

这种标准化方法的应用非常广泛，得到的数据会完全落入[0，1]区间内（Z-Score则没有类似区间），这种方法能使数据归一化而落到一定的区间内，同时还能较好地保持原有数据结构。

3.9.3 用于稀疏数据的MaxAbs

最大值绝对值标准化（MaxAbs）即根据最大值的绝对值进行标准化，假设原转换的数据为x，新数据为x'，那么x'=x/|max|，其中max为x所在列的最大值。

MaxAbs方法跟Max-Min用法类似，也是将数据落入一定区间，但该方法的数据区间为[-1，1]。MaxAbs也具有不破坏原有数据分布结构的特点，因此也可以用于稀疏数据、稀疏的CSR或CSC矩阵。

CSR（Compressed Sparse Row，行压缩）和CSC（Compressed Sparse Column，列压缩）是稀疏矩阵的两种存储格式，这两种稀疏矩阵在scipy.sparse包中应用广泛。除了这两种格式之外，用于存储稀疏矩阵的格式还有COO、CSR、DIA、ELL、HYB等。

3.9.4 针对离群点的RobustScaler

某些情况下，假如数据集中有离群点，我们可以使用Z-Score进行标准化，但是标准化后的数据并不理想，因为异常点的特征往往在标准化之后便容易失去离群特征。此时，可以使用RobustScaler针对离群点做标准化处理，该方法对数据中心化和数据的缩放健壮性有更强的参数控制能力。

3.9.5 代码实操：Python数据标准化处理

本示例中，将使用Numpy、sklearn进行标准化相关处理。源文件data6.txt位于“附件-chapter3”中，默认工作目录为“附件-chapter3”（如果不是，请切换到该目录下，否则会报错“IOError:File data5.txt does not exist”）。

完整代码如下：

```
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt

data = np.loadtxt('data6.txt', delimiter='\t')  # 读取数据

# Z-Score标准化
zscore_scaler = preprocessing.StandardScaler()  # 建立StandardScaler对象
data_scale_1 = zscore_scaler.fit_transform(data)  # StandardScaler标准化处理

# Max-Min标准化
minmax_scaler = preprocessing.MinMaxScaler()  # 建立MinMaxScaler模型对象
data_scale_2 = minmax_scaler.fit_transform(data)  # MinMaxScaler标准化处理

# MaxAbsScaler标准化
maxabsscaler_scaler = preprocessing.MaxAbsScaler()  # 建立MaxAbsScaler对象
data_scale_3 = maxabsscaler_scaler.fit_transform(data)  # MaxAbsScaler标准化
处理

# RobustScaler标准化
robustscalerr_scaler = preprocessing.RobustScaler()  # 建立RobustScaler标准化
对象
data_scale_4 = robustscalerr_scaler.fit_transform(data)  # RobustScaler标准
化标准化处理

# 展示多网格结果
data_list = [data, data_scale_1, data_scale_2, data_scale_3, data_scale_4]
建数据集列表
scalar_list = [15, 10, 15, 10, 15, 10]  # 创建点尺寸列表
color_list = ['black', 'green', 'blue', 'yellow', 'red']  # 创建颜色列表
merker_list = ['o', ',', '+', 's', 'p']  # 创建样式列表
title_list = ['source data', 'zscore_scaler', 'minmax_scaler', 'maxabsscaler
建标题列表
for i, data_single in enumerate(data_list):  # 循环得到索引和每个数值
    plt.subplot(2, 3, i + 1)  # 确定子网格
    plt.scatter(data_single[:, :-1], data_single[:, -1], s=scalar_list[i], m
网格展示散点图
    plt.title(title_list[i])  # 设置自网格标题
plt.suptitle("raw data and standardized data")  # 设置总标题
plt.show()  # 展示图形
```

示例代码用空行分为7个部分。

第一部分导入库，示例中用到Numpy做数据读取、Sklearn中的preprocessing模块做标准化处理、matplotlib.pyplot模块做可视化图形展示。

第二部分使用Numpy的loadt方法导入数据文件，文件以tab分隔。

第三部分进行Z-Score标准化。通过preprocessing.StandardScaler建立模型，然后应用fit_transform方法进行转换。除了StandardScaler类外，还可直接使用preprocessing.scale（X）做标准化处理，二者的应用差别是preprocessing.StandardScaler方法既可以满足一次性的标准化处理，又能转换数据集的特征保存下来，分别通过fit和transform对多个数据集（例如测试集和训练集）做相同规则的转换。该方法针对训练集、测试集和应用集分别应用时非常有效。

第四部分进行Max-Min标准化。通过preprocessing.MinMaxScaler建立模型，然后应用fit_transform方法进行转换。其中的fit_transform方法都可以使用fit然后使用transform做多数据集的应用。该方法也可以使用没有面向对象API的等效函数sklearn.preprocessing.minmax_scale。

在Python中，Max-Min标准化不仅能将数据落入[0，1]的区间，还可以指定最大和最小值范围。通过在建立对象时设置preprocessing.MinMaxScaler（feature_range=（min，max））来自定义标准化后的数据区间。

第五部分进行MaxAbsScaler标准化。通过preprocessing.MaxAbsScaler建立模型，然后应用fit_transform方法进行转换，也可以使用没有面向对象API的等效函数sklearn.pre-processing.maxabs_scale。

第六部分进行RobustScaler标准化。通过preprocessing.RobustScaler建立模型，然后应用fit_transform方法进行转换，也可以使用没有面向对象API的等效函数sklearn.pre-processing.robust_scale。

第七部分展示原始数据和4种标准化的结果。在该部分功能中，将原始数据以及通过上述4种方法得到的结果数据统一对比展示。主要步骤如下：

1）先创建5个列表，用于存储原始数据和4个标准化后的数据、散点尺寸、颜色、样式、标题。

2）通过for循环结合enumerate将索引值和5份数据循环读出；通过plt.subplot为不同的数据设置不同的网格，该网格为2行3列；在每个网格中通过plt.scatter画出散点图，并通过索引将列表中对应的值作为参数传给scatter；然后通过title方法为每个子网格设置标题。

3）最后设置整个图像的总标题并展示图像，如图3-4所示。

上述过程中，需要考虑的关键点是如何根据不同的数据分布特征和应用选择合适的标准化方式，具体来说包含如下几点：

·如果要做中心化处理，并且对数据分布有正态需求，那么使用Z-Score方法；

·如果要进行0-1标准化或要指定标准化后的数据分布范围，那么使用Max-Min标准化或MaxAbs标准化方式是比较好的选择，尤其是前者；

·如果要对稀疏数据进行处理，Max-Min标准化或MaxAbs标准化仍然是理想方法；

·如果要最大限度保留数据集中的异常，那么使用RobustScaler方法更佳。

本小节示例中，主要用了如下几个知识点：

·通过Numpy的loadtxt方法读取文本数据文件，并指定分隔符；

·使用sklearn.preprocessing的StandardScaler方法做Z-Score标准化处理；

·使用sklearn.preprocessing的MinMaxScaler方法做Max-Min标准化处理；

·使用sklearn.preprocessing的MaxAbsScaler方法做最大值绝对值标准化处理；

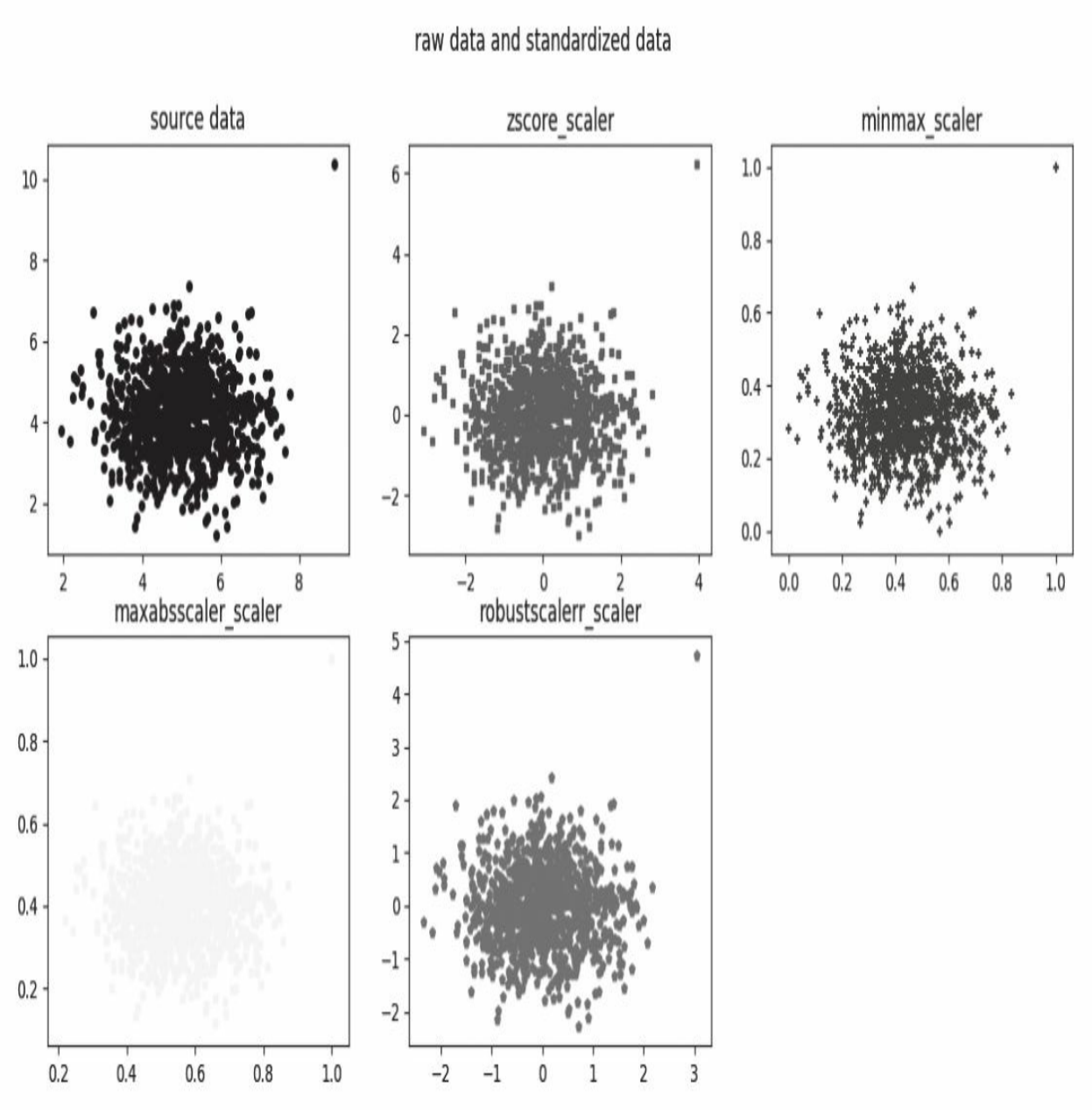


图3-4　不同标准化方法结果

·使用sklearn.preprocessing的RobustScaler方法做针对异常数据的处理；

·通过matplotlib.pyplot画图，并在一幅大图中使用subplot设置多个子网格，使用scatter画出散点图并设置颜色、样式、大小，使用title和suptitle设置子网格和整体图像标题并最终通过show方法展示图像。

3.10　离散化，对运营数据做逻辑分层

所谓离散化，就是把无限空间中有限的个体映射到有限的空间中。数据离散化操作大多是针对连续数据进行的，处理之后的数据值域分布将从连续属性变为离散属性，这种属性一般包含2个或2个以上的值域。离散化处理的必要性：

·节约计算资源，提高计算效率。

·算法模型（尤其是分类模型）的计算需要。虽然很多模型，例如决策树可以支持输入连续型数据，但是决策树本身会先将连续型数据转化为离散型数据，因此离散化转换是一个必要步骤。

·增强模型的稳定性和准确度。数据离散化之后，处于异常状态的数据不会明显突出异常特征，而是会被划分为一个子集中的一部分，因此异常数据对模型的影响会大大降低，尤其是基于距离计算的模型（例如K均值、协同过滤等）效果明显。

·特定数据处理和分析的必要步骤，尤其在图像处理方面应用广泛。大多数图像做特征检测（以及其他基于特征的分析）时，都需要先将图像做二值化处理，二值化也是离散化的一种。

·模型结果应用和部署的需要。如果原始数据的值域分布过多，或值域划分不符合业务逻辑，那么模型结果将很难被业务理解并应用。

离散化通常针对连续数据进行处理，但是在很多情况下也可以针对已经是离散化的数据进行处理。这种场景一般是离散数据本身的划分过于复杂、琐碎甚至不符合业务逻辑，需要进一步做数据聚合或重新划分。

3.10.1 针对时间数据的离散化

针对时间数据的离散化主要用于以时间为主要特征的数据集中和粒度转换，离散化处理后将分散的时间特征转换为更高层次的时间特征。

在带有时间的数据集中，时间可能作为行记录的序列，也可能作为列（维度）记录数据特征。常见的针对时间数据的离散化操作分为两类：

·针对一天中的时间离散化。一般是将时间戳转换为秒、分钟、小时或上下午。

·针对日粒度以上数据的离散化。一般是将日期转化为周数、周几、月、工作日或休息日、季度、年等。

针对时间数据的离散化可以将细粒度的时间序列数据离散化为粗粒度的三类数据：

·离散化为分类数据，例如上午、下午；

·离散化为顺序数据，例如周一、周二、周三；

·离散化为数值型数据，例如一年有52个周，周数是数值型数据。

3.10.2 针对多值离散数据的离散化

针对多值离散数据的离散化指的是要进行离散化处理的数据本身不是数值型数据，而是分类或顺序数据。

例如，用户收入变量的值原来可能划分为10个区间，根据新的建模需求，只需要划分为4个区间，那么就需要对原来的10个区间进行合并。

多值离散数据要进行离散化还有可能是划分的逻辑有问题，需要重新划分，这种问题通常都是由于业务逻辑的变更，导致在原始数据中存在不同历史数据下的不同值域定义。

例如，用户活跃度变量的值，原来分为高价值、中价值和低价值3个类别。根据业务发展的需要，新的用户活跃度变量的值定义为高价值、中价值、低价值和负价值4类。此时需要对不同类别的数据进行统一规则的离散化处理。

3.10.3 针对连续数据的离散化

针对连续数据的离散化是主要的离散化应用，在分类或关联分析中应用尤其广泛，这些算法的结果以类别或属性标识为基础，而非数值标记。例如，分类规则的典型结果逻辑是：

如果变量1=值1并且变量2=值2

那么目标变量（T）

连续数据的离散化结果可以分为两类：一类是将连续数据划分为特定区间的集合，例如{（0，10]，（10，20]，（20，50]，（50，100]}；一类是将连续数据划分为特定类，例如类1、类2、类3；

常见实现针对连续数据离散化的方法包括：

·分位数法：使用四分位、五分位、十分位等分位数进行离散化处理，这种方法简单易行。

·距离区间法：可使用等距区间或自定义区间的方式进行离散化，这种操作更加灵活且能满足自定义需求，另外该方法（尤其是等距区间）可以较好地保持数据原有的分布。

·频率区间法：将数据按照不同数据的频率分布进行排序，然后按照等频率或指定频率离散化，这种方法会把数据变换成均匀分布，好处是各区间的观察值是相同的，不足是已经改变了原有数据的分布状态。

·聚类法：例如使用K均值将样本集分为多个离散化的簇。

·卡方：通过使用基于卡方的离散化方法，找出数据的最佳临近区间并合并，形成较大的区间。

3.10.4 针对连续数据的二值化

在很多场景下，我们可能需要将变量特征进行二值化操作：每个数据点跟阈值比较，大于阈值设置为某一固定值（例如1），小于阈值设置为某一固定值（例如0），然后得到一个只拥有两个值域的二值化数据集。

二值化后的值的设置取决于场景，例如大部分数据的处理可以设置为1或0；在图像处理中则会设置为0或255。有关如何设置没有固定要求，只要满足后续数据和结果的识别、理解和应用即可。

二值化应用的前提是数据集中所有的属性值所代表的含义相同或类似，例如读取图像所获得数据集是颜色值的集合（具体颜色模式取决于读取图像时的模式设置，例如灰度、RGB等），因此每一个数据点都代表一种颜色，此时可对整体数据集做二值化处理。某些情况下，也可能只针对特定列做二值化，这样不同列的属性虽然不同，但同一列内产生的二值化结果却仍然具有比较和分类的意义。

3.10.5 代码实操：Python数据离散化处理

本示例中，将使用Pandas、sklearn进行离散化相关处理。数据源文件data7.txt位于“附件-chapter3”中，默认工作目录为“附件-chapter3”（如果不是，请切换到该目录下，否则会报错“IOError:File data7.txt does not exist”）。

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn import preprocessing

# 读取数据
df = pd.read_table('data7.txt', names=
['id', 'amount', 'income', 'datetime', 'age'])  # 读取数据文件
print (df.head(5))  # 打印输出前5条数据

# 针对时间数据的离散化
for i, signle_data in enumerate(df['datetime']):  # 循环得到索引和对应值
    single_data_tmp = pd.to_datetime(signle_data)  # 将时间转换为datetime格式
    df['datetime'][i] = single_data_tmp.weekday()  # 离散化为周几
print (df.head(5))  # 打印输出前5条数据

# 针对多值离散数据的离散化
map_df = pd.DataFrame([['0-10', '0-40'], ['10-20', '0-40'], ['20-30', '0-
40'], ['30-40', '0-40'], ['40-50', '40-80'], ['50-60', '40-80'], ['60-
70', '40-80'], ['70-80', '40-80'], ['80-90', '>80'], ['>90', '>80']],
                    columns=['age', 'age2'])  # 定义一个要转换的新区间
df_tmp = df.merge(map_df, left_on='age', right_on='age', how='inner')  # 数
据框关联匹配
df = df_tmp.drop('age', 1)  # 丢弃名为age的列
print (df.head(5))  # 打印输出前5条数据

# 针对连续数据的离散化
# 方法1：自定义分箱区间实现离散化
bins = [0, 200, 1000, 5000, 10000]  # 自定义区间边界
df['amount1'] = pd.cut(df['amount'], bins)  # 使用边界做离散化
print (df.head(5))  # 打印输出前5条数据
# 方法2 使用聚类法实现离散化
data = df['amount']  # 获取要聚类的数据，名为amount的列
data_reshape = data.reshape((data.shape[0], 1))  # 转换数据形状
model_kmeans = KMeans(n_clusters=4, random_state=0)  # 创建KMeans模型并指定要
聚类的数量
keames_result = model_kmeans.fit_predict(data_reshape)  # 建模聚类
df['amount2'] = keames_result  # 新离散化的数据合并到原数据框
print (df.head(5))  # 打印输出前5条数据
# 方法3：使用4分位数实现离散化
df['amount3'] = pd.qcut(df['amount'], 4, labels=
['bad', 'medium', 'good', 'awesome'])  # 按4分位数进行分隔
df = df.drop('amount', 1)  # 丢弃名为amount的列
print (df.head(5))  # 打印输出前5条数据

# 针对连续数据的二值化
binarizer_scaler = preprocessing.Binarizer(threshold=df['income'].mean())  #
立Binarizer模型对象
```

```
income_tmp = binarizer_scaler.fit_transform(df['income'])  # Binarizer标准化
转换
income_tmp.resize(df['income'].shape)  # 转换数据形状
df['income'] = income_tmp  # Binarizer标准化转换
print (df.head(5))  # 打印输出前5条数据
```

示例代码用空行分为6个部分。

第一部分导入库。代码中用到了Pandas和Sklearn。前者主要用来做文件读取、切块、时间处理、关联合并和部分离散化操作；后者主要用来做二值化和聚类建模离散化。

第二部分使用Pandas的read_table方法读取数据文件，并指定列名。数据集为100行5列的数据框，包含id、amount、income、datetime和age等5个字段。原始数据前5条数据如下：

```
id  amount  income             datetime     age
0   15093     1390   10.40  2017-04-30 19:24:13   0-10
1   15062     4024    4.68  2017-04-27 22:44:59  70-80
2   15028     6359    3.84  2017-04-27 10:07:55  40-50
3   15012     7759    3.70  2017-04-04 07:28:18  30-40
4   15021      331    4.25  2017-04-08 11:14:00  70-80
```

第三部分针对时间数据的离散化。该过程中，首先通过enumerate方法获得要循环的日期索引和对应值，在每个循环中通过Pandas的to_datetime方法将字符串转换为datetime格式，并直接使用weekday方法获取周几，新获取的周几的数据直接替换原数据框的时间戳。最后打印输出离散化后的结果：

```
id amount  income datetime           age
0   15093    1390    10.40      6   0-10
1   15062    4024     4.68      3  70-80
2   15028    6359     3.84      3  40-50
3   15012    7759     3.70      1  30-40
4   15021     331     4.25      5  70-80
```

从结果中看到，datetime列的值由原来的日期时间格式转换为由0～6组成的数值，0代表周一，6代表周日。

该部分由于to_datetime无法针对整个数据框或Series做整体转

换，因此需要写循环执行。另外，该方法执行效率非常低，笔者的环境下大概需要5秒，如果数据量更大则效率会更低。

第四部分针对多值离散数据的离散化。该过程中先通过dp.Dataframe定义一个新的转换区间，用来将原数据映射到新区间；然后通过merge方法将原数据框和新定义的数据框进行关联，关联的两个key（left_on和left_on）分别是age和age，关联模式为inner（内关联）；接着我们通过drop方法去除原始数据框中的age列，只保留新的转换后的区间。得到如下结果：

```
id  amount  income datetime       age2
0   15093     1390    10.40     6  0-40
1   15064     7952     4.40     0  0-40
2   15080      503     5.72     5  0-40
3   15068     1668     3.19     5  0-40
4   15019     6710     3.20     0  0-40
```

上述返回结果中，age2列是转换后新的列，原来的分类区间已经被映射到新的类别区间。

第五部分是针对连续数据的离散化。该部分包含3种常用方法。

方法1：自定义分箱区间实现离散化。首先自定义一个区间边界列表，用来对数据做划分；然后使用Pandas的cut方法做离散化，并将结果生成一个名为amount1的新列追加到原数据框，打印输出结果如下：

```
id  amount  income datetime     age2        amount1
0    15093    1390   10.40   6  0-40   (1000, 5000]
1    15064    7952    4.40   0  0-40  (5000, 10000]
2    15080     503    5.72   5  0-40    (200, 1000]
3    15068    1668    3.19   5  0-40   (1000, 5000]
4    15019    6710    3.20   0  0-40  (5000, 10000]
```

上述返回结果中，amount1列是转换后产生的列，每行对应的区间左侧是区间开始值（不包含），右侧是区间结束值（包含）；除了显示区间外，cut方法还可以通过自定义labels（值为列表的形式，用来表示不同分类区间的标签）用标签代替上述区间，例如labels=['bad'，'medium'，'good'，'awesome']，那么显示在amount1里面的数据就是对应lables里面的字符串。这里没有将原始amount列删除，原因是下

面的方法中还会用到该列原始数据。

方法2：使用聚类法实现离散化。该过程使用了sklearn.cluster的KMeans算法实现。首先通过指定数据框的列名获得要建模的数据列；然后对数据的形状进行转换，否则算法会认为该数据只有1行（通过Pandas指定列名获得的数据默认都没有列值，例如示例中的data的形状是（100，），因此大多数场景下作为输入变量都需要做形状转换）；接着创建KMeans模型并指定聚类类别数量为4，并设置初始化随机种子为固定值0（否则每次得到的聚类结果很可能基本都不一样），并使用fit_predict方法直接建模输出结果；最后将新的结果追加到原始数据框中，最终打印输出前5条结果如下：

```
   id  amount  income datetime       age2        amount1  amount2
0    15093    1390    10.40    6  0-40   (1000, 5000]         2
1    15064    7952     4.40    0  0-40  (5000, 10000]         1
2    15080     503     5.72    5  0-40    (200, 1000]         2
3    15068    1668     3.19    5  0-40   (1000, 5000]         2
4    15019    6710     3.20    0  0-40  (5000, 10000]         1
```

上述返回结果中，amount2列是转换后产生的列，列的值域是0、1、2，分别代表三类数据。

方法3：使用4分位数实现离散化。该过程中，使用了Pandas的qcut方法指定做4分位数分隔，同时设置不同4分位得到的区间的标签分别为['bad'，'medium'，'good'，'awesome']）；将得到的结果以列名为amount3追加到原始数据框中；然后通过drop方法丢弃名为amount的列，打印输出结果如下：

```
   id  income datetime       age2        amount1  amount2  amount3
0    15093    10.40    6  0-40   (1000, 5000]         2      bad
1    15064     4.40    0  0-40  (5000, 10000]         1  awesome
2    15080     5.72    5  0-40    (200, 1000]         2      bad
3    15068     3.19    5  0-40   (1000, 5000]         2      bad
4    15019     3.20    0  0-40  (5000, 10000]         1  awesome
```

上述返回结果中，amount3列是转换后产生的列，其结果构成与方法1完全相同，差异仅在于区间边界不同。


使用方法1中的cut方法也能实现，二者的区别是cut方法通常是自定义分割区间，而qcut则是应用标准分位数方法。

第六部分做特征二值化处理。先建立Binarizer模型对象，然后使用fit_transform方法进行二值化转换，阈值设置为该列的均值；转换后得到的数据的形状是（1，100），通过resize更改为与原始数据框income列相同的尺寸，并用结果直接替换原始列的值，最后打印输出前5条数据结果如下：

```
id  income datetime      age2        amount1  amount2  amount3
0    15093      1.0   6  0-40   (1000, 5000]        2      bad
1    15064      1.0   0  0-40  (5000, 10000]        1  awesome
2    15080      1.0   5  0-40    (200, 1000]        2      bad
3    15068      0.0   5  0-40   (1000, 5000]        2      bad
4   15019       0.0   0  0-40  (5000, 10000]        1  awesome
```

上述返回结果中，income列的值已经离散为由0和1组成的二值化数据。

上述过程中，需要考虑的关键点是：如何根据不同的数据特点和建模需求选择最合适的离散化方式。因为离散化方式是否合理会直接影响后续数据建模和应用效果。

除了本节介绍的相关类型的转换离散化制约外，不同模型对于离散化的约束如下：

·使用决策树时往往倾向于少量的离散化区间，原因是过多的离散化将使得规则过多受到碎片区间的影响。

·关联规则需要对所有特征一起离散化，原因是关联规则关注的是所有特征的关联关系，如果对每个列单独离散化将失去整体规则性。

本小节示例中，主要用了几个知识点：

·通过Pandas的read_table方法读取文本数据文件，并指定列名；

·通过Pandas的to_datetime方法将字符串转换为datetime格式，使用weekday提取周几的数据；

·使用Pandas的head方法只展示前n条数据；

·通过Pandas的merge方法合并多个数据框，实现类似SQL的数据关联查询；

·使用Pandas的drop方法丢弃特定数据列；

·使用Pandas的cut和qcut方法实现基于自定义区间和分位数方法的数据离散化；

·使用sklearn.cluster的KMeans方法实现聚类分析；

·使用shape方法获取矩阵形状并使用resize方法对矩阵实现形状转换；

·使用sklearn.preprocessing的Binarizer方法做二值化处理。

3.11 数据处理应该考虑哪些运营业务因素

数据处理工作不仅依赖于数据工作者的经验，也需要考虑实际的运营业务因素。这种兼顾两种工作逻辑的工作方式会帮助数据工作者少走弯路并降低数据项目失败的可能性，还有利于提高数据工作的效率和产出效果，真正让运营理解数据、应用数据并驱动业务。

数据处理时应该考虑的运营业务因素包括固定和突发运营周期、运营需求的有效性、交付时要贴合运营落地场景、专家经验、业务需求等变动因素。

3.11.1 考虑固定和突发运营周期

运营业务的周期属性主要表现在两个方面：

·有计划的周期性：运营业务计划的制定都有明显的周期性规律，运营业务的执行也是如此。这种有计划的周期性规律一般包含不同层次的周期。例如，运营一般都会先做年度计划、季度计划，然后分解到月度计划，月度计划再细化到周度和日度计划，层层递进，步步追踪。

·临时或突发周期：除了有规划的运营周期以外，偶然事件的发生也会影响运营业务。例如，由于内部DBA的误操作，导致数据库部分数据被删除，直接影响了公司正常的销售和运营状态。这种情况发生的时间以及产生的影响周期通常都是不固定的。

运营业务的周期性特征对数据的影响：

·有计划的运营周期在数据的选取和分析过程中非常重要，尤其涉及对比（环比、同比等）时，选对具有相同属性的对比周期是形成结论的基础。

·有计划的运营周期对于时间序列特征明显的建模影响很大，包括时间序列、时序关联、隐马尔可夫模型等，这些算法和模型都需要数据带有明显的前后序列状态的关联属性。使用这类算法需要将运营周期的属性与算法和应用的属性相匹配。

·不同周期下产生的数据可能有差异，尤其是对于高速发展的新型公司，不同周期下的数据可能带有明显的线性、指数、二项式以及其他变化特征，甚至可能带有业务因素导致的异常数据点。

·运营过程可能产生突发的数据工作需求，例如针对异常事件的临时性分析，而这些需求很可能由于没有提前针对性地做数据布点、跟踪和采集，而导致数据不完整或根本没有有效数据可进行分析。

·数据工作的整个过程都需要运营业务人员参与，而依赖于运营业务人员参与的时机以及对应的方式和切入点也很重要。例如，在业务正常工作非常繁重甚至无法脱身的情况下，如果需要业务过多地参与数据

工作，必然会形成来自于业务端的重重阻力，此时需要更多数据自动化和程序化的工作模式。

3.11.2 考虑运营需求的有效性

在数据工作项目真正开始前，通常会有多次沟通、反馈、验证和摸底的过程，这些动作的目的是根据业务的需求、数据的实际状态以及数据工作本身的限制来综合考虑运营需求是否有效。数据工作者并不是一定需要承接所有的运营数据需求，他们可能会对某些需求做拒绝或延迟处理，主要原因如下：

·缺少数据：现有数据无法满足运营人员提出的数据分析需求，典型案例是在应对突发事件时的数据分析需求。

·需求不合理：经过验证后，发现运营人员提出的需求不合理，或无法通过数据得出结论。例如运营人员要分析客户对于新产品的预期，这种需求除了市场和客户调研外，其他方法基本无法实现，因为预期本身无法通过数据衡量。

·条件限制：虽然运营人员需求合理，但现有服务器、算法、技术、经验等主客观条件无法实现。例如运营提出要从监控视频中得到整个人在线下店铺内的所有浏览、查看和购买商品的轨迹行为，由于缺少相应的技术和经验而无法实现。

·资源限制：目前数据工作已经满负荷，无法并行开展更多工作。

·低价值需求：运营人员可以自己实现的基本需求。很多时候运营人员会犯懒，基本的取数、查询、统计和分析等工作全部交给数据工作者来实现，对于这些简单且常用的工作内容，由于本身属于数据工作的范畴，数据工作者大多都会接收执行。这些工作当然有价值，但是要真正最大化发挥数据工作者的价值，绝不能只着眼于这些内容。这些基本工作可以通过可视化报表、自动邮件、数据工作文化的培养等逐步从数据工作中剥离，或逐步降低内容比例。这样才能有更多资源应用到潜在规律、预测性和探索性的知识发现中。

对于符合以上特征的数据场景，数据工作者需要慎重考虑是否还要继续投入资源，对于需要拒绝的一定要及时提出，以免导致数据工作项目失败以及数据工作价值度降低。

3.11.3 考虑交付时要贴合运营落地场景

数据处理虽然只是中间过程，并没有到达数据分析、建模、部署及应用等后期阶段，但该阶段的很多工作都会直接影响后期交付和运营落地效果。典型因素包括：

（1）维持原有指标

后期应用时需要原始业务指标（变量）以便于业务理解和应用。如果有类似的需求，那么在数据处理（比如降维）时就不能采用数据转换的方法，要根据实际情况通过多种方式选择维度或不进行降维。

（2）更容易理解的算法模型

某些运营人员可能比较“认真”，会非常关注算法模型的实现过程，如果采用无法解释具体过程的算法（例如神经网络的实现过程）或难以理解的算法（例如SVM中的超平面），那么这类运营人员通常会怀疑数据工作的有效性和正确性。此时，选择更加容易理解的算法模型（例如决策树、线性回归等）则比算法准确度、时效性更重要。在数据处理过程中，要针对这些容易理解的模型做针对性的数据处理。

（3）数据生产和应用环境

如果数据工作项目的结果不是分析、挖掘报告，需要通过程序化的方式执行，那么交付的一般都是代码或脚本。在数据处理程序发布上线时，应该尽量使用生产和应用中既有的模块、环境、库、语言、版本等，减少额外部署、开发和维护的工作量。

3.11.4 不要忽视业务专家经验

本书中多次提到了业务专家经验的重要作用。业务专家经验在数据处理工作中的重要作用体现在以下两个方面。

1.数据工作方向

数据工作方向指的是整个数据工作项目中需要做什么、产出是什么、中间的过程应该向哪个方向考虑等，这些内容侧重于“是什么”。这些内容直接产生于业务专家经验，主要影响的数据工作内容包括：

·数据项目工作目标和需求；

·数据探索和摸底方向；

·数据交付物的形式和规格。

2.数据工作逻辑

数据工作逻辑指的是在数据工作本身方面，业务人员能够为数据工作者提供的价值参考和工作建议，这些内容侧重于“怎么做”。主要影响的数据工作环节包括：

·总体数据周期、规则、条件等的选取；

·数据抽样规则，尤其涉及分层、整群抽样；

·多数据的整合、匹配和关联关系；

·不同数据源和数据间的清洗、转换逻辑；

·重复值、异常值和缺失值的处理逻辑；

·数据离散化的方法选择和区间定义；

·根据变量重要性进行数据变量的选取和降维；

·数据算法和模型选择；

·数据模型的调整、评估和优化。

如果只擅长于运营，这是一个纯业务属性价值点；如果只擅长于数据，这是一个纯数据属性价值点。只有同时具备业务+数据的双重属性，才能成就真正的“分析师”。成功的数据工作一定是数据+运营两条腿走路！

3.11.5 考虑业务需求的变动因素

业务需求的变动主要来源于业务环境的变动或业务需求本身的变动。前者是由于客观环境的变化导致业务需求不得不变动，后者则产生于运营业务本身的主观环境。

在数据工作项目中，变化的需求会影响整个数据工作的所有环节，如果业务需求频繁变动，会给数据工作带有极大困扰甚至可能直接导致数据工作项目的失败。因此，数据处理一定要将业务需求的变动因素考虑进去，涉及客观环境的变化无法预测，而业务主观思想上的变化在很多情况下可以提前做好准备：

（1）充分、有效的沟通

沟通是建立持久、稳定关系的重要方法，在开始数据项目的工作之前，数据工作者千万不要嫌麻烦而略过这一步。多次邀请相关的直接需求业务人员、业务领导、数据提供者（通常对应业务口的数据管理员）进行正式会议的沟通是必要的。同时，为了避免会议沟通上口头表达和理解的误差，一定要在每次会议之后都撰写会议纪要并找相应人员确认作为备案，还要抄送多方领导来让各方都重视并慎重考虑沟通和落实内容。

（2）更完整、更原始的数据集

在数据集的选择和处理时，涵盖更多时间、维度、来源等方面的数据，并放宽甚至去掉业务给出的数据过滤条件，这样可以减少重复取数工作并尽量降低对后续所有处理流程的二次调整影响；对于涉及数据归约的，尽量优先选择直接选取而非转换的形式，这样可以最大限度满足业务对原始数据维度和指标的需求。

（3）可理解性强、规则清晰的算法和模型

在保证一定的模型准确度的情况下，优先选择可理解性强、规则清晰的算法和模型，以降低因业务无法理解而导致的返工或无法落地等情况。

（4）模块化工作方法

在大多数程序式数据工作中，我们发现很多工作内容的基本功能模块是可以复用的，这意味着假如我们第一次需要开发10个模块，第二次可能只需要额外开发5个，其中5个模块复用第一次开发的即可，第三次可能只需要额外开发3个，以此类推，随着项目的增加可复用的功能模块会越来越多。虽然根据不同的场景、数据，同一个模块也会有不同的实现方法、参数等，但这仅仅是对原有模块进行优化和升级，越到后期就越会形成更加通用、完善且具有高复用能力的功能模块。这些功能模块不仅可以用来提高数据项目的工作效率，还能有效应对业务需求的个性化、灵活性的需求变动。

模块化工作方法是笔者倡导的一种有效的工作方式，在Python中，笔者会更多的使用面向函数的编程方法来实现。

（5）建立数据工作流程和机制

无规矩不成方圆。所有的数据工作都应该有对应的流程和机制才能保障其正常运行。对于数据工作流程而言，要建立起从数据需求提出到数据落地的完整流程，其中需要包含应对需求变动的时间、频率、方式、范围、影响的规范，以及审批和授权制度（一定要经相应人员和领导同意），这样制度才有可能落地。当然，只有制度还不够，更关键的还要看执行！

3.12　内容延伸：非结构化数据的预处理

3.12.1　网页数据解析

本节通过一个稍微复杂一点的示例，来演示如何抓取并解析网页数据。之所以说复杂，是因为本节中会出现几个本书中未曾提及的知识和方法，从代码数量来看也会比之前的示例稍微长一点。

本示例中，将使用requests、bs4、re、time库进行网页数据读取、解析和相关处理。

示例的目标是抓取亚马逊中国网站苹果手机和配件的价格，用于做竞争对手的标杆商品价格监控。注意：本示例仅做学习之用。

在抓取和解析网页数据之前，首先要做的是做网页内容分析，包括：

要抓取的内容格式：文本、图像还是其他文件。

·是否存在重定向：重定向往往根据User-Agent来判断，例如手机端、电脑端所看到的页面信息不同。

·是否需要验证：很多网页爬虫都需要用户登录、验证码等。

·目标数据是否具有统一标签规则：要爬取的数据是否具有统一的HTML标签，这决定了后期处理的便捷性。

·2大多数情况下网页爬取都不是只有一个页面，而是多个页面，因此需要了解不同页面的URL规则，尤其是带有条件查询的，需要了解具体参数。

·2根据实际要爬取的数据，分析可能会产生哪些字段，会有哪些冲突、包含关系及关联性影响等。

点击亚马逊中国网站左侧进入二级导航“手机通讯”中的Apple Phone:https://www.amazon.cn/s/ref=sa_menu_digita_l3_siphone？

ie=UTF8&page=1&rh=n%3A665002051%2Cp_89%3AApple%2Cn%3A664

1.要抓取的内容格式分析

笔者的浏览器是Chrome，在当前页面中按快捷键F12，打开开发者工具（或者点击右上角 ⋮，在弹出的菜单中选择更多工具-开发者工具），点击 Elements 切换到查看页面元素视图。点击开发者工具栏左侧的 ，然后点击页面中的商品价格、标题等，多测试几个商品，发现价格是文本格式（不是调用的外部图像）。同样的方法点击商品描述，分析商品标题的特点。商品标题用来存储与某个价格对应的商品，该标题中的关键字可用于识别具体商品型号并与企业自身商品做比较，如图3-5所示。

图3-5　要抓取的内容格式分析

2.是否存在重定向分析

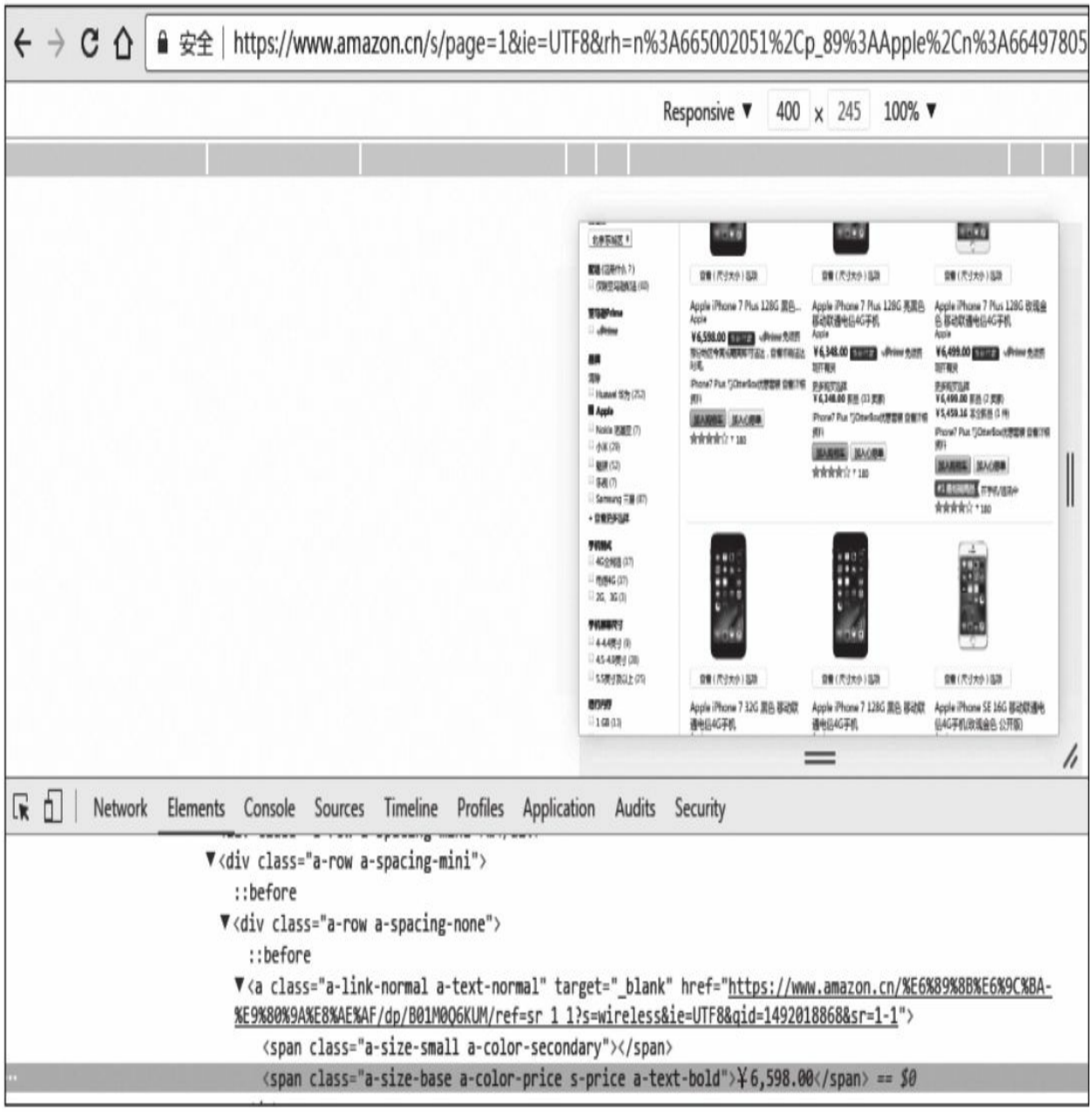
点击开发者工具左侧的第二个图表  ，通过浏览器模拟移动设备环境，然后按F5刷新该页面。原来适配到电脑上的页面现在改为适配移动端，然后按照刚才的方法查看要抓取的数据格式是否仍然相同；除了显示的样式不同可能导致的页面展示和规则不同外，不同平台的商品价格可能存在不同（例如移动端比电脑端便宜10元），如果需要区分平台，那么可以分开抓取。本节仅以电脑上的可视网页为例进行抓取。相关操作如图3-6所示。

3.是否需要验证分析

由于页面中浏览商品时没有任何需要登录、注册等验证信息，因此无须验证可直接访问。

4.目标数据是否具有统一标签规则

通过查看源代码，发现不同的商品通过li标签进行列表展示，每个li下对应一个商品。在li子层级的标签中，类为"a-size-base a-color-price s-price a-text-bold"的span标签包含了价格信息，h2标签总则包含了商品标题，如图3-7所示。因此我们只需要找到这两个特征的标签即可解析出目标数据。


安全 https://www.amazon.cn/s/page=1&ie=UTF8&rh=n%3A665002051%2Cp_89%3AApple%2Cn%3A66497805
Responsive 400 x 245 100%
Network Elements Console Sources Timeline Profiles Application Audits Security
▼
::before
▼
::before
▼<a class="a-link-normal a-text-normal" target="_blank" href="https://www.amazon.cn/%E6%89%8B%E6%9C%BA-%E9%80%9A%E8%AE%AF/dp/B01M0Q6KUM/ref=sr_1_1?s=wireless&ie=UTF8&qid=1492018868&sr=1-1">

￥6,598.00 == $0

图3-6　重定向分析

图3-7　主要商品信息所在的标签

相关知识点：HTML标签

超文本标记语言（HTML）的标签是HTML语言的基本单位，也是设计网页的基本元素。我们查看源代码（在任意页面右击，在弹出的菜单中点击“查看页面源代码”，或使用快捷键Ctrl+U），都能发现HTML标签构成了所有代码的“骨架”。不同的HTML标签的作用不同。我们在爬虫网页内容时“看到”（指的是前台展示的信息）的内容通常都是在body标签里面的。上述示例中几个HTML标签的基本含义：

·li：HTML中的列表，用来展示多个并列的项目信息。

·h2：HTML中的二级标题，一般表示强调，呈现字体加粗增大的效果。

·span：定义一个文字段落。

上面这些不同的标签以及效果（HTML里面被称为样式），可以通

过多种方法在多个地方定义和引用，不同的定义之间会有覆盖效果。因此，标签本身的默认效果可能被更高优先级的样式覆盖而无法显示出来。

5.URL规则分析

在URL规则部分，URL中https://www.amazon.cn/s/page=1&ie=UTF8&rh=n%3A665002051%2Cp_89所包含的page=*用于控制页码，其中*是指示不同的页码，因此我们只需要获得总页面数量，然后依次循环读取数据即可。通过观察页面分析，发现页面中并没有直接显示页面总数量的信息，不过我们可以通过页面左上角的总返回结果数显示： 1-24条， 共81条计算得出。默认我们打开的是第一页，因此有关返回结果的三个数据分布表示第一页商品的起止数以及总商品数，我们用（总商品数/第一页商品数）+1便可以得出总页面数。例如（81/24）+1=4。关于页面这三个数字可以通过“要抓取的内容格式分析”的方法找到页面数量的信息，位于id值为s-result-count的h2标签内。相关页面如图3-8所示。

图3-8　商品数量标签

6.业务常识性分析

本节示例中的商品没有苹果手机的统一型号编码，只有标题的描述信息，因此后期还需要跟实际苹果手机进行匹配，该工作需建立一个匹配表，后期定期维护和增量更新即可。

到此为止我们基本确定了抓取思路：先从打开页面计算得到总页面数量，然后循环读出不同的页面信息；接着在每个页面找到每个商品的标题和价格，并把数据保存到本地文件。

清楚了上述基本情况后，我们开始编写代码，完整代码如下：

```
# 导入库
import requests  # 用于请求
from bs4 import BeautifulSoup  # 用于HTML格式化处理
import re  # 用于HTML配合查找条件
import time  # 用于文件名保存

# 获取总页面数量
def get_total_page_number():
    user_agent = 'Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KH
    headers = {'User-Agent': user_agent}  # 定义头信息
           url  =  'https://www.amazon.cn/s/ref=sa_menu_digita_l3_siphone?
ie=UTF8&page=1&rh=n%3A665002051%2Cp_89%3AApple%2Cn%3A664978051'  # 寻找页码的
URL
    res = requests.get(url, headers=headers)  # 发送请求
    html = res.text  # 获得请求中的返回文本信息
    html_soup = BeautifulSoup(html)  # 建立soup对象，用于处理HTML
      page_number_span = html_soup.find('h2', id="s-result-count")  # 查找
id="s-result-count"的h2标签
    page_number_code = page_number_span.text  # 读出该标签的文本信息
    number_list = re.findall(r'(\w*[0-9]+)\w*', page_number_code)  # 使用正则
表达式解析出文本中的3个数字
    total_page_number = (int(number_list[-1]) / int(number_list[-2])) + 1  #
算得出总页码
    return int(total_page_number)  # 返回页面数字

# 解析单页面
def parse_single_page(i):
       url_part1 = 'https://www.amazon.cn/s/ref=sa_menu_digita_l3_siphone?
ie= UTF8&page=%d' % i  # 定义URL动态前半部分
     url_part2 = '&rh=n%3A665002051%2Cp_89%3AApple%2Cn%3A664978051'  # 定义
URL静态后半部分
    url = url_part1 + url_part2  # 拼接成完整URL
    print ('parse url: %s' % url)  # 输出URL信息
    user_agent = 'Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KH
    headers = {'User-Agent': user_agent}  # 定义头信息，用于发送请求
    res = requests.get(url, headers=headers)  # 发送请求
    html = res.text  # 获得请求中的返回文本信息
    html_soup = BeautifulSoup(html)  # 建立soup对象，用于处理HTML
     tag_list = html_soup.find_all('li', id=re.compile('^result.*'))  # 查找
id以result开始的li标签，返回列表
    for tag_info in tag_list:  # 读取列表中每一个标签（一个标签对应一个商品）
```

```
        # 解析价格
        # print (tag_info)
           price_code = tag_info.find('span', class_="a-size-base a-color-
price s-price a-text-bold")  # 查找价格标签
        if price_code != None:  # 如果非空则继续
            price = price_code.text  # 取出价格标签文字
            # 解析商品标题
            title_code = tag_info.find('h2')  # 查找标题标签
            title = title_code.text  # 取出标题标签文字
            write_data(title, price)  # 每次解析完成写入文件

# 将数据写入文件
def write_data(title, price):
    file_date = time.strftime('%Y-%m-%d', time.localtime(time.time()))  # 当
前日期，用于文件命名
    fn = open('%s.txt' % file_date, 'a+')  # 新建文件对象，以追加模式打开
    content = title + '\t' + price + '\n'  # 写内容，标题和价格以tab分割，末尾增
加换行符
    fn.write(content)  # 写入文件
    fn.close()  # 关闭文件对象

# 解析多页面并写文件
def main():
    total_page_number = get_total_page_number()  # 获得总页面数
    for i in range(1, int(total_page_number) + 1):  # 循环读出每个页面
        parse_single_page(i)

main()
```

上述代码用空行分为6个部分。

第一个部分导入库，具体用途在注释中已经注明。

第二个部分开始我们每个功能都定义为一个函数模块，用于在不同场景下引用。get_total_page_number模块用来计算页面数量。

第三个部分定义了一个用于解析单个URL的函数模块，在定义URL的部分，由于原始URL中有%，这会导致我们新增占位符做外部数字引用时报错，因此应将其分开定义再组合。在解析不同的标签（包括后面的模块）时，我们用到了正则表达式模块，可以非常容易地解析出目标字符。

第四个部分是将每次解析的标题和价格写入文件。文件以追加模式打开，这样每次的数据都会追加到文件尾，而不会覆盖之前的数据，类似于数据库的追加模式操作。在写文件内容时，末尾需要有换行符，否则所有数据都会合并到一行。

第五个部分是通过一个循环来调用并执行多个页面的解析。

第六个部分函数用来执行所有的操作。

上述代码执行后返回结果如下。

1）调试窗口输出信息：

```
parse      url:      https://www.amazon.cn/s/ref=sa_menu_digita_l3_siphone?
ie=UTF8&page=1&rh=n%3A665002051%2Cp_89%3AApple%2Cn%3A664978051
parse      url:      https://www.amazon.cn/s/ref=sa_menu_digita_l3_siphone?
ie=UTF8&page=2&rh=n%3A665002051%2Cp_89%3AApple%2Cn%3A664978051
parse      url:      https://www.amazon.cn/s/ref=sa_menu_digita_l3_siphone?
ie=UTF8&page=3&rh=n%3A665002051%2Cp_89%3AApple%2Cn%3A664978051
parse      url:      https://www.amazon.cn/s/ref=sa_menu_digita_l3_siphone?
ie=UTF8&page=4&rh=n%3A665002051%2Cp_89%3AApple%2Cn%3A664978051
```

2）程序执行目录下产生一个跟运行时间相同的数据文件，图3-9所示是文件中的部分数据。

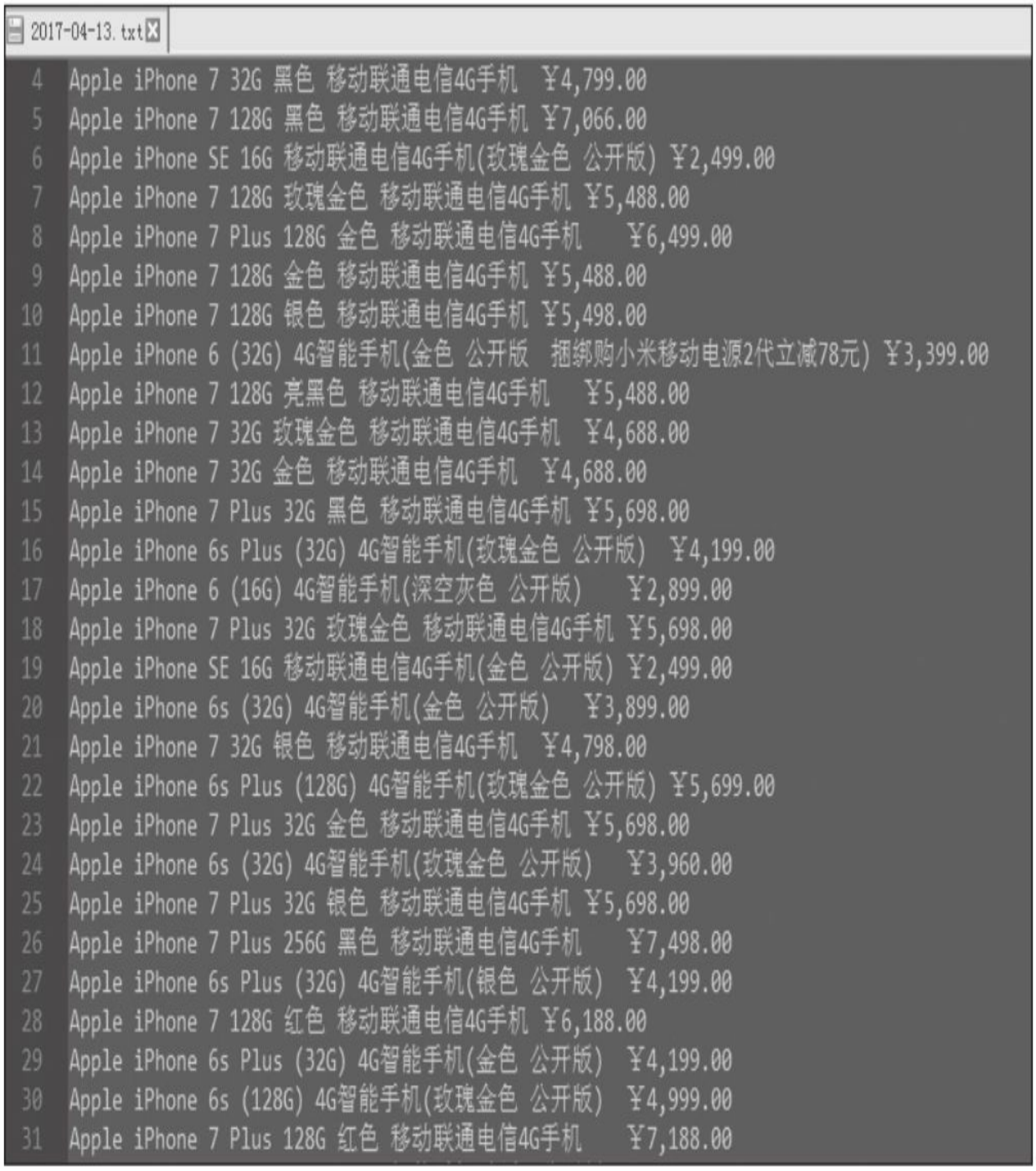


图3-9　爬取数据结果

上述的抓取执行过程中，“遭遇”到了亚马逊的反爬虫应对措施。经过测试，连续执行2次代码，第二次会返回HTML代码，但没有目标数据；连续第3次，HTTP请求直接无法返回数据。

大多数情况下，通过网络爬虫获取数据都是作为辅助方式，原因是现在几乎所有的网站都有防爬虫的意识和方式，这导致数据爬取会受到外部很多因素的影响而导致数据质量低下。基于爬虫的主要工作内容包括舆情监测、市场口碑、用户情绪、市场营销等方面，属于外部属性较强的“附加”工作。这些工作其实都不是公司的核心运营内容，这就会导致这些工作看似有趣并且有价值，但真正对企业来讲价值很难实际体现。

相关知识点：函数

函数是用来形成一个功能的代码段，函数可以用来被其他应用调用。使用函数的好处很多：

·利于维护：代码中有变更时，只需要对特定代码段进行修改，而无须全部修订。

·复用：当某个功能会被很多应用调用时，通过函数可实现一次撰写多次使用的目的。

·清晰化功能设计：当设计功能时，不同的函数模块类似于功能主题，同时基于函数又可以派生库、类、包，通常函数是模块划分的基本单位。

·递归：使用函数可以实现一种特殊的功能——递归，函数内部的功能调用函数自身，实现自循环，这种功能经常被用于有固定规律的场景下，例如求阶乘。

定义函数使用def[函数名]即可，函数可用来执行单独的任务，也可以通过return返回执行结果，用来与其他功能做交互使用。同时，不同的函数间可以通过赋值进行参数传递和调度之用。

上述过程中，需要考虑的关键点是：如何根据不同网页的实际特点，尤其是对于反爬虫的应对来正确读取网页源代码，读取源代码之后的解析往往不是主要问题。

本小节示例中，主要用了如下几个知识点：

·通过requests库发送带有自定义head信息的网络请求；

·通过requests返回对象的text方法获取源代码文本信息；

·使用bs4的BeautifulSoup库配合find方法进行目标标签查找和解析，并通过其text方法获得标签文本信息；

·通过re的正则表达式功能，实现对于特定数字规律的查找；

·通过定义function函数来实现特定功能或返回特定结果；

·通过for循环读取数据列表；

·通过if条件判断对符合条件的记录进行处理；

·对文本文件的读写操作；

·使用time的localtime、time、strftime方法进行日期获取以及格式化操作。

3.12.2 网络用户日志解析

网络用户日志属于非结构化数据的一种，其解析方法根据配合的服务器和跟踪实施的不同而需要自定义模块，本节将用一个示例来演示如何进行日志解析。

本示例中，将使用正则表达式配合自定义函数模块实现日志解析功能。数据源文件traffic_log_for_dataivy位于“附件-chapter3”中，默认工作目录为“附件-chapter3”（如果不是，请切换到该目录下，否则会报错“IOError:File traffic_log_for_dataivy does not exist”）。

背景：我们在这个网站上部署了Google Analytics的代码用来监测用户的行为，由于该工具是SAAS工作模式，数据采集之后直接发送到Google云端服务器。我们通过一定的方式将每次发送给谷歌的数据同时“备份”了一条保存到本地服务器日志。其中日志请求内容的部分以“GET/__ua.gif？”开头便是日志记录。我们的目标是找到这些日志，然后对日志做初步的解析，并存放到本地文件便于后期做进一步数据和应用。

要实现这一目标，基本思路是：先从日志文件夹中读取日志文件列表（本示例中仅有1个文件，因此省略该步骤）；然后依次读取每个日志文件中的数据；接着对日志文件中的每条数据进行判断，符合条件才能成为我们要的目标数据；最后将数据做初步解析并写到文件里面。

本节我们依然以函数的方式来撰写各个功能模块，如下是完整代码。

```
# 去除爬虫功能
def remove_spider_data(single_log):
    # 定义排除爬虫规则集
    exclude_set = [
        'AhrefsBot',
        'archive.org_bot',
        'baiduspider',
        'Baiduspider',
        'bingbot',
        'DeuSu',
        'DotBot',
        'Googlebot',
        'iaskspider',
        'MJ12bot',
```

```
        'msnbot',
        'Slurp',
        'Sogou web spider',
        'Sogou Push Spider',
        'SputnikBot',
        'Yahoo! Slurp China',
        'Yahoo! Slurp',
        'YisouSpider',
        'YodaoBot',
        'bot.html'
    ]
    count = 0  # 初始计数用于计算日志中是否包含爬虫
    for spider in exclude_set:  # 循环读取每个爬虫
        if single_log.find(spider) != -1:  # 如果爬虫出现在日志里面
            count += 1  # count + 1
    if count > 0:  # 如果结果不为0，意味着日志中有爬虫
        return 0  # 返回 0
    else:  # 否则
        return 1  # 返回1

# 读取日志数据
def get_raw_log(file):
    fn_read = open(file, 'r')  # 打开要读取的日志文件对象
    content = fn_read.readlines()  # 以列表形式读取日志数据
    fn_read.close()  # 关闭文件对象
    for single_log in content:  # 循环判断每天记录
          rule1 = single_log.find('GET /__ua.gif?') != -1  # 定义日志规则：含
ua.gif的请求
        rule2 = remove_spider_data(single_log)
        if rule1 == True and rule2 == True:  # 如果同时符合2条规则，则执行
            fn_write = open('ua_data.txt', 'a+')  # 打开要保存的ua日志文件对象
            fn_write.writelines(single_log)  # 写入符合规则的日志
            fn_write.close()  # 关闭文件对象

# 解析每条日志数据
def split_ua(line):
    import re
    # 定义不同日志分割的正则表达式
    ip_rule = r'[\d.]*'  # 定义IP规则，例如203.208.60.230
                time_rule    =    r'\[[^\[\]]*\]'        #    定义时间规则，例如
[02/Mar/2016:14:00:23 +0800]
    request_rule = r'\"[^\"]*\"'  # 定义请求规则
    status_rule = r'\d+'  # 定义返回的状态码规则，例如200
    bytes_rule = r'\d+'  # 返回的字节数，例如326
    refer_rule = r'\"[^\"]*\"'  # 定义refer规则
    user_agent_rule = r'\"[^\"]*\"'  # 定义user agnet规则
    # 原理：主要通过空格和-来区分各不同项目，在各项目内部写各自的匹配表达式
                    log_re_pattern     =     re.compile(r'(%s)\     -\     -
\ (%s)\ (%s)\ (%s)\ (%s)\ (%s)\ (%s)' % (
        ip_rule, time_rule, request_rule, status_rule, bytes_rule, refer_rul
整表达式模式
    matchs = log_re_pattern.match(line)  # 匹配
    if matchs != None:  # 如果不为空
        allGroups = matchs.groups()  # 获得所有匹配的列表
        ip_info = allGroups[0]  # IP信息
        time_info = allGroups[1]  # 时间信息
        request_info = allGroups[2]  # 请求信息
        status_info = allGroups[3]  # 状态信息
        bytes_info = allGroups[4]  # 字节数信息
        refer_info = allGroups[5]  # refer信息
        user_agent_info = allGroups[6]  # user agent信息
```

```
        return ip_info, time_info, request_info, status_info, bytes_info, re

# 主程序
def get_final_data():
    file = 'traffic_log_for_dataivy'  # 定义原始日志
    get_raw_log(file)  # 读取非结构化文本数据
    fn_r = open('ua_data.txt', 'r')  # 打开要读取的ua日志文件
    content = fn_r.readlines()  # 读取ua所有日志记录
    fn_r.close()  # 关闭ua文件对象
    fn_w = open('final_data.txt', 'a+')  # 打开要写入的格式化文件
    for line in content:  # 按行循环
        ip_info, time_info, request_info, status_info, bytes_info, refer_inf
            line)  # 获得分割后的数据
        log_line = ip_info + '!' + time_info + '!' + request_info + '!' + st
                   + '!' + user_agent_info  # 按指定规则组合写入文件
        fn_w.writelines(log_line)  # 写入文件
        fn_w.writelines('\n')  # 写入换行符用于换行
    fn_w.close()  # 关闭写入的文件对象

# 执行程序
get_final_data()  # 执行所有程序
```

上述代码用空行分为5个部分。

第一部分去除爬虫功能。

该函数块的功能是从日志中去除属于爬虫产生的数据。在日志中，很多搜索引擎公司的网络爬虫在抓取页面时会产生日志数据，除了谷歌外，还百度、搜狗、雅虎、有道、必应等都有类似的爬虫（或者称为机器人）。

代码中先定义了一个爬虫列表，所有我们怀疑为爬虫的关键字字段都可加到列表里面，用来做爬虫记录过滤；然后我们通过for循环读取爬虫列表的每个字段，并在每行日志里通过find方法确认爬虫是否存在，如果存在则在计数器上加1（count+=1），一旦count大于0，我们就认为在日志记录里面至少包含1个爬虫信息。最终的返回结果为0代表有爬虫；1代表没有爬虫。

第二部分读取日志数据。

先通过open方法以只读模式打开日志文件，然后通过readlines方法以列表的形式读取日志记录，读取完成之后关闭日志文件对象。

通过一个for循环将列表中的每条日志读取出来，然后定义两条规则用于判断日志是否符合条件：一条规则是日志中必须包

含'GET/__ua.gif？'字符串，这是我们定义Google Analytics日志的标志，另一条规则是日志中不能包含爬虫数据，直接使用在remove_spider_data功能模块的返回值做判断。当在if条件语句中同时满足上述2个条件时，将该条日志记录添加到名为ua_data.txt的新文件中。

该部分代码执行后，会在Python程序当前工作目录下产生一个名为ua_data.txt的新文件。

第三部分解析每条日志数据。

该模块定义了日志下所有字段的分割规则，首先我们针对日志记录中要解析的每个数据字段定义一个正则表达式构成的规则集，以用于目标数据的解析；接着将分散的规则通过re库的compile方法合成一个匹配模式；然后基于匹配模式使用match方法匹配每条数据记录，并由此返回匹配的字段值列表，最后解析出所有定义的字段值并返回给其他函数使用。

第四部分是主要程序模块。

这部分功能的意义是将前文提到的功能整合到该模块中。

先是定义了一个日志文件用于读取日志数据，通过之前定义的get_raw_log模块获取符合要求的日志数据并保存为单独的数据文件。再读取上述保存的日志数据并按行循环，在循环中，我们将每条日志进行分割并以“！”作为最终分隔符写入目标文件。最后执行程序。

上述过程目的是提高每个模块的可读性和可理解性，我们按照功能单独定义，因此“忽略”了总体考虑和最优化规则，此代码有很多地方可以优化：

·import re应该放到全局里面导入。

·从get_raw_log获得的数据，其实不必先写入文件再从文件读取，该过程可以省略（本节之所以先存后读是为了便于读者理解过程），所有程序最终可以只有1个结果文件。

·日志被分割后又进行组合并再次写入文件，实际中不会这样执

行，因为已经格式化的数据可以放到数据库里面。

·分割后的request_info模块的数据还没有解析，原因是没有应用场景需求做支撑，所以暂时不做处理。该字段中每个请求的数据主体（URL中？之后的部分）都是以key-value形式存在的键值对，解析起来非常容易。但是，由于不同的请求里面包含的key（即参数）可能不同，而且系统预定义的key的值域数量非常大（例如il<listIndex>pi<productIndex>cm<metricIndex>的最大笛卡儿为200*200*200=8000000），因此这些数据无法直接分列存储到一张关系型数据库表里面。解决思路有两种：一是将字段按主题拆分，形成数据仓库或数据集市；二是使用非关系型数据仓库，直接支持海量数据（行和列）的扩展。目前这两种方式都有公司在实行。

上述过程中，需要考虑的关键点是：

·如何根据不同的服务器日志配置以及前端代码跟踪实施的具体情况，编写日志过滤规则。

·有关爬虫数据的排除，也是需要额外注意的信息点。

本小节示例中，主要用了以下几个知识点：

·对文本文件的读写操作；

·通过find方法查找符合条件的字符串；

·通过if做多条件判断，并对符合条件的记录做处理；

·通过re的正则表达式功能，实现对于特定字段的查找和匹配；

·通过定义function函数来实现特定功能或返回特定结果；

·通过for循环读取数据列表。

3.12.3 图像的基本预处理

本示例中，将使用OpenCV来做图像基本预处理操作，基本处理内容包括图像缩放、平移、旋转、透视变换、图像色彩模式转换、边缘检测、二值化操作、平滑处理、形态学处理。

数据源文件sudoku.png、j.png位于“附件-chapter3”中，默认工作目录为“附件-chapter3”（如果不是，请切换到该目录下，否则会报类似错“IOError:File sudoku.png does not exist”）。完整代码如下：

```
import cv2  # 导入图像处理库
import numpy as np  # 导入numpy库
from matplotlib import pyplot as plt  # 导入展示库

# 展示图像模块
def img_show(img_name, img):
    cv2.imshow(img_name, img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# 原始图像
img_file = 'sudoku.png'  # 定义原始数据文件
img = cv2.imread(img_file)  # 以彩色模式读取图像文件
rows, cols, ch = img.shape  # 获取图像形状
img_show('raw img', img)  # 展示彩色图像

# 图像缩放
img_scale = cv2.resize(img, None, fx=0.6, fy=0.6, interpolation=cv2.INTER_CU
像缩放
img_show('scale img', img_scale)  # 展示缩放后的图像

# 图像平移
M = np.float32([[1, 0, 100], [0, 1, 50]])  # 定义平移中心
img_transform = cv2.warpAffine(img, M, (cols, rows))  # 平移图像
img_show('transform img', img_transform)  # 展示平移后的图像

# 图像旋转
M = cv2.getRotationMatrix2D((cols / 2, rows / 2), 45, 0.6)  # 定义旋转中心
img_rotation = cv2.warpAffine(img, M, (cols, rows))  # 第一个参数为旋转中心，第
二个为旋转角度，第三个为旋转后的缩放因子
img_show('rotation img', img_rotation)  # 展示旋转后的图像

# 透视变换
pts1 = np.float32([[76, 89], [490, 74], [37, 515], [520, 522]])  # 定义变换前
的四个校准点
pts2 = np.float32([[0, 0], [300, 0], [0, 300], [300, 300]])  # 定义变换后的四
个角点
M = cv2.getPerspectiveTransform(pts1, pts2)  # 定义变换中心点
img_perspective = cv2.warpPerspective(img, M, (300, 300))  # 透视变换
img_show('perspective img', img_perspective)  # 展示透视变换后的图像

# 转换为灰度图像
```

```
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # 图像转灰度
img_show('gray img', img_gray)  # 展示灰度图像

# 边缘检测
img_edges = cv2.Canny(img, 100, 200)  # 检测图像边缘
img_show('edges img', img_edges)  # 展示图像边缘

# 图像二值化
ret, th1 = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)  # 简单阈值
th2 = cv2.adaptiveThreshold(img_gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.T
适应均值阈值
th3 = cv2.adaptiveThreshold(img_gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, c
适应高斯阈值
titles = ['Gray Image', 'Global Thresholding (v = 127)',
          'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']  #
义图像标题
images = [img_gray, th1, th2, th3]  # 定义图像集
for i in xrange(4):
    plt.subplot(2, 2, i + 1), plt.imshow(images[i], 'gray')  # 以灰度模式展示
每个子网格的图像
    plt.title(titles[i])  # 设置每个自网格标题
    plt.xticks([]), plt.yticks([])  # 设置x轴和y轴标题
plt.show()  # 展示图像

# 图像平滑
kernel = np.ones((5, 5), np.float32) / 25  # 设置平滑内核大小
img_smoth_filter2D = cv2.filter2D(img, -1, kernel)  # 2D卷积法
img_smoth_blur = cv2.blur(img, (5, 5))  # 平均法
img_smoth_gaussianblur = cv2.GaussianBlur(img, (5, 5), 0)  # 高斯模糊
img_smoth_medianblur = cv2.medianBlur(img, 5)  # 中值法
titles = ['filter2D', 'blur', 'GaussianBlur', 'medianBlur']  # 定义标题集
images = [img_smoth_filter2D, img_smoth_blur, img_smoth_gaussianblur, img_sm
义图像集
for i in xrange(4):
    plt.subplot(2, 2, i + 1), plt.imshow(images[i], 'gray')  # 以灰度模式展示
每个子网格的图像
    plt.title(titles[i])  # 设置每个自网格标题
    plt.xticks([]), plt.yticks([])  # 设置x轴和y轴标题
plt.show()  # 展示全部图像

# 形态学处理
img2 = cv2.imread('j.png', 0)  # 以灰度模式读取图像
kernel = np.ones((5, 5), np.uint8)  # 设置形态学处理内核大小
erosion = cv2.erode(img2, kernel, iterations=1)  # 腐蚀
dilation = cv2.dilate(img2, kernel, iterations=1)  # 膨胀
plt.subplot(1, 3, 1), plt.imshow(img2, 'gray')  # 设置自网格1图像
plt.subplot(1, 3, 2), plt.imshow(erosion, 'gray')  # 设置自网格2图像
plt.subplot(1, 3, 3), plt.imshow(dilation, 'gray')  # 设置自网格3图像
plt.show()  # 展示全部图像
```

上述代码以空行分为11个部分，涵盖了日常图像处理的常用操作。

第一部分为导入库，本代码中除了OpenCV库外，还有用于定义图像处理的内核的Numpy、用于展示多图图像的Matplotlib。

第二部分定义了一个函数，用来做单图像展示。下面的每个功能模

块，当只有一个图像做展示会直接调用该模块，而无须重复写展示功能代码。cv2.show（）方法必须与cv2.waitKey（）、cv2.destroyAllWindows（）一起使用才能保证图像正常展示及关闭。

第三部分读取原始图像并展示。通过cv2的imread方法以彩色模式读取图像，然后获得彩色图像的长、高和通道形状，最终调用img_show做图像展示，结果如图3-10中所示的①。

第四部分图像缩放处理。直接使用cv2.resize方法设置缩放比例、方法等并展示输出为原图像60%的新图像，结果如图3-10所示的②。

第五部分图像平移处理。先定义图像平移中心，然后使用cv2.warpAffine方法根据平移中心移动图像，移动后的图像，结果如图3-10所示的③。

第六部分图像旋转处理。与图像平移类似，先定义旋转中心，然后使用cv2.warpAffine进行旋转，同时设置旋转角度45度、缩放因子为0.6，结果如图3-10所示的④。

第七部分透视变换处理。先定义变换前的四个校准点，然后定义变换后的四个角点，可用来控制图像大小，接着定义变换中心点并应用cv2.warpPerspective进行透视变换，结果如图3-10所示的⑤。

第八部分转换为灰度图像。使用cv2.cvtColor将BRG模式转为GRAY模式，结果如图3-10所示的⑥。

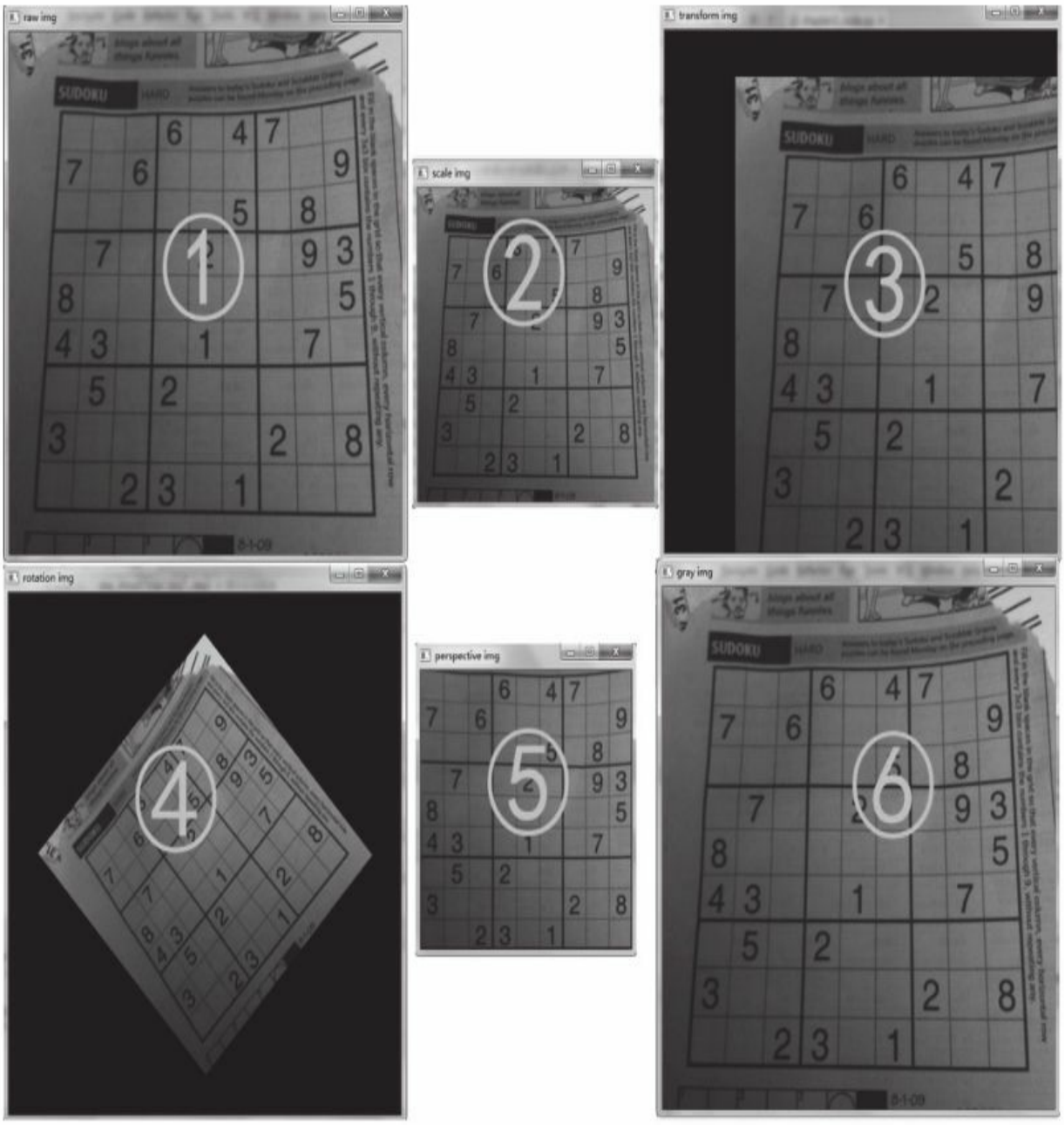


图3-10　原始图像和部分处理后的图像

第九部分边缘检测处理。使用cv2.Canny检测图像边缘，结果如图3-11所示。

第十部分图像二值化处理。这里分别应用了简单阈值、自适应均值阈值、自适应高斯阈值三种方法做二值化处理，并使用Matplotlib做多网格图像，同时展示原始图像和三种阈值下二值化图像处理结果，如图

3-12所示。

第十一部分图像平滑处理。先设置平滑内核大小，然后分别使用cv2.filter2D（3D卷积）、cv2.blur（平均法）、cv2.GaussianBlur（高斯模糊）、cv2.medianBlur（中值法）进行平滑结果对比，如图3-13所示。

第十二部分形态学处理。这里重新以灰度模式读取一个图像，定义处理内核之后，通过cv2.erode和cv2.dilate分别实现腐蚀和膨胀操作。原图和腐蚀、膨胀处理后的图像对比如图3-14所示。

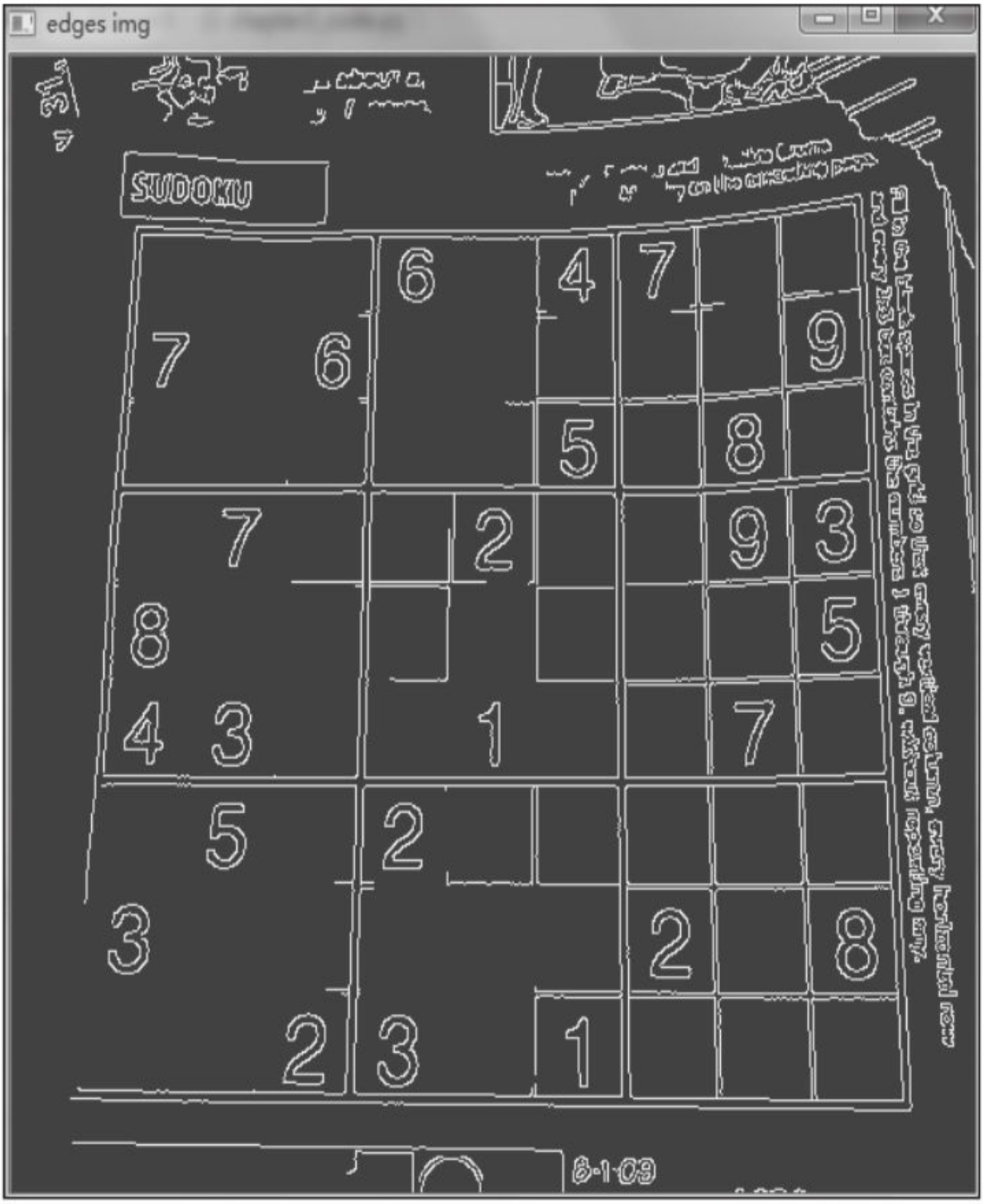


图3-11　边缘检测后的图像

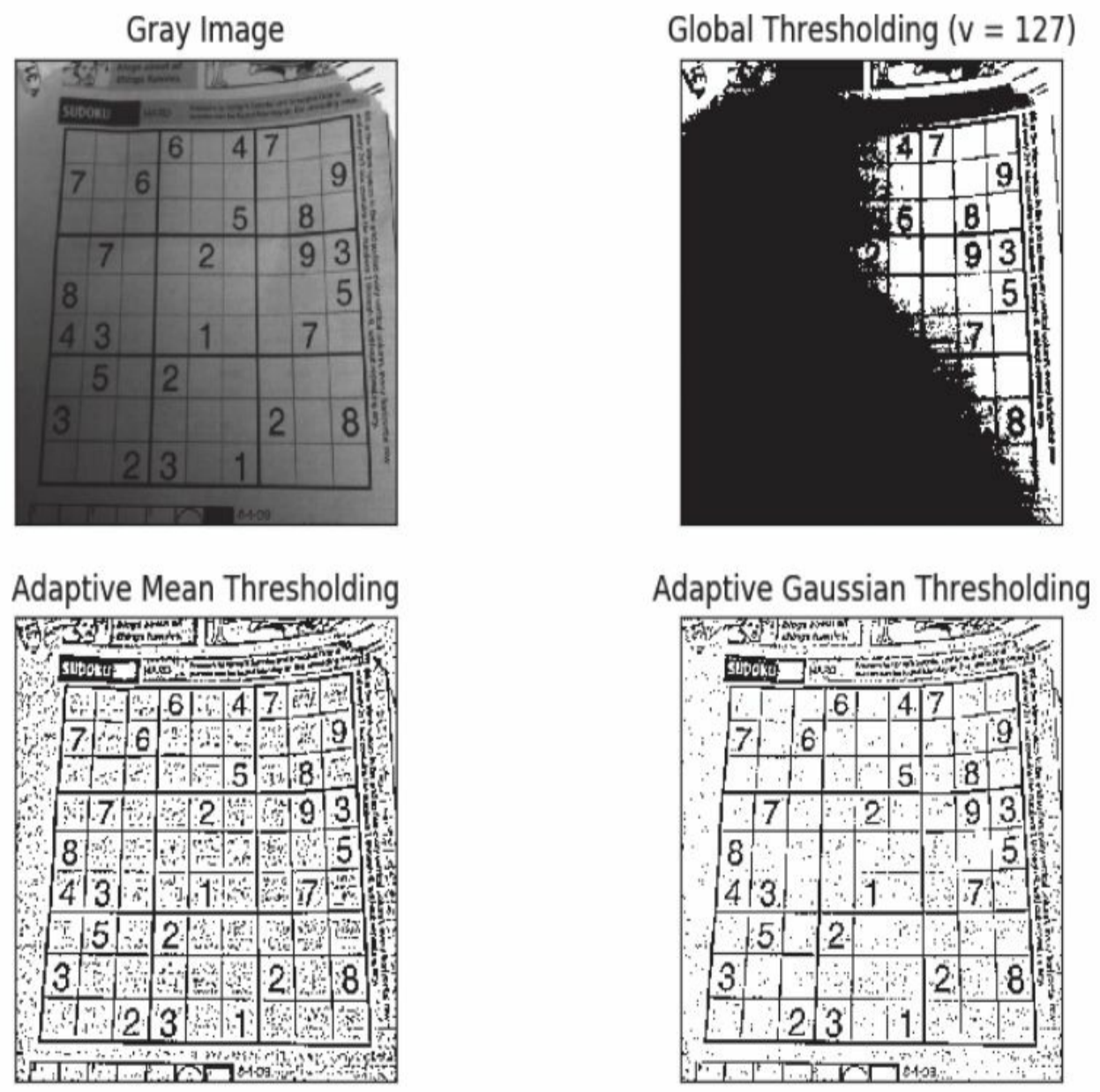


图3-12　原图和多种阈值控制下的二值化结果

filter2D

blur

GaussianBlur

medianBlur

图3-13　多种平滑效果对比

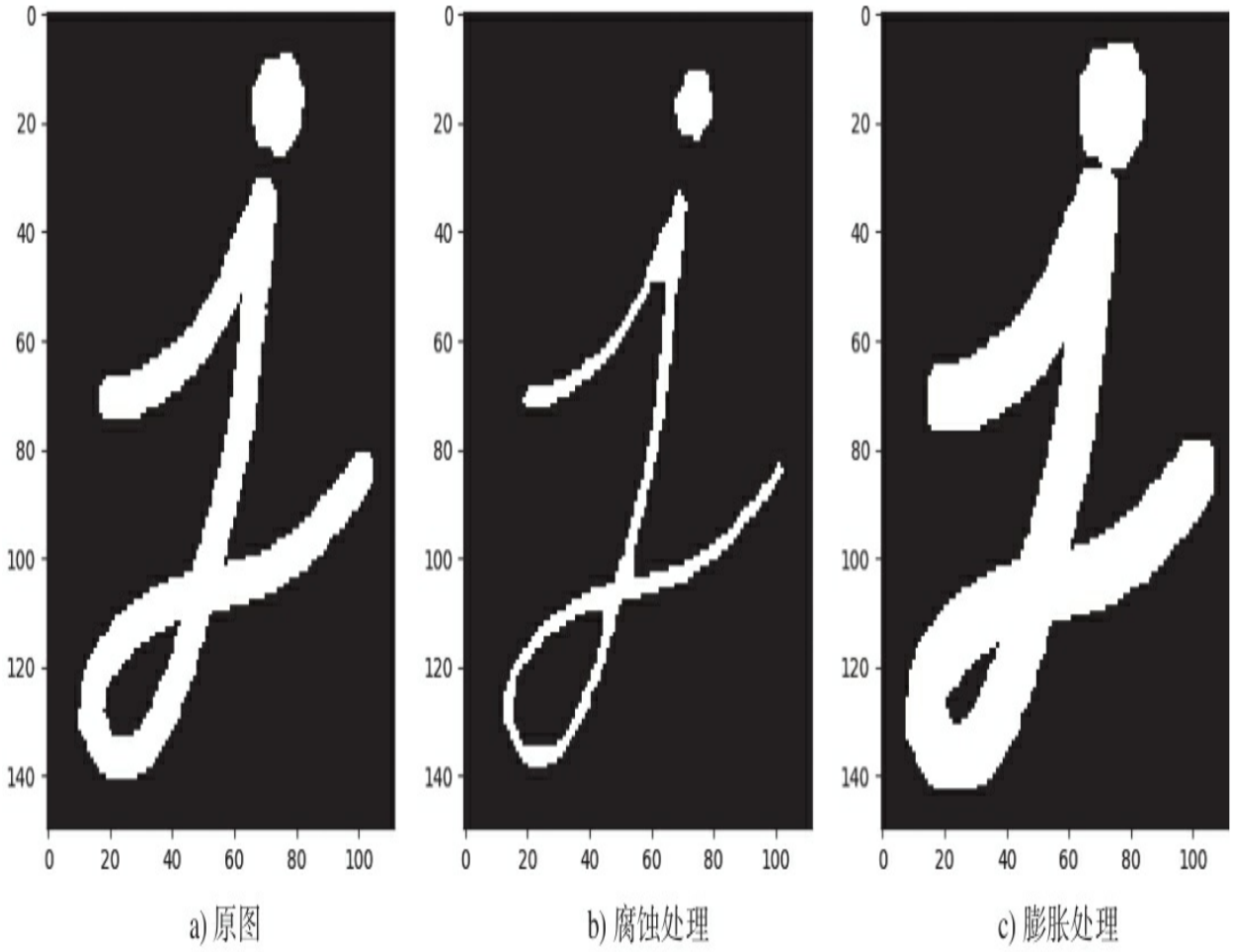


图3-14　原图和腐蚀、膨胀处理效果

上述过程中，需要考虑的关键点是：如何根据不同的图像处理需求，实现图像的基本预处理任务，尤其是对于每种方法下参数的具体设置，需要根据实际情况加以选择。另外，在程序自动化过程中，是不可能依靠人工参与每次边界调整、阈值优化等具体过程的，这往往通过一定专用的参数优化模型来实现。例如，对于透视图像的处理，首先要做的是识别出透视图像的矫正参照点，而该过程就是一个融合了业务场景、图像处理技术、数学知识和计算方法等多学科知识的建模过程。

本小节示例中，主要用了几个知识点：

·通过cv2.imread对图像文件的数据进行读取，并分别以彩色和灰度模式读取图像；

·通过cv2.imshow、cv2.waitKey（）、cv2.destroyAllWindows（）实现图像展示；

·通过Matplotlib库实现多子自网格图的展示；

·通过cv2.resize实现图像缩放；

·通过cv2.warpAffine实现图像平移；

·通过cv2.warpAffine和cv2.getRotationMatrix2D实现图像旋转；

·通过cv2.warpPerspective和cv2.getPerspectiveTransform实现图像透视变换；

·通过cv2.cvtColor实现图像颜色模式转换；

·通过cv2.Canny实现图像边缘检测；

·通过cv2.threshold实现图像二值化处理，并通过简单阈值、自适应均值阈值、自适应高斯阈值等方法寻找最佳阈值；

·通过cv2.filter2D、cv2.blur、cv2.GaussianBlur、cv2.medianBlur等方法实现图像平滑处理；

·通过for循环做数据循环输出；

·通过cv2.erode、cv2.dilate等方法实现图像腐蚀、膨胀等形态学处理。

3.12.4 自然语言文本预处理

与数据库中的结构化数据相比，文本具有有限的结构，某些类型的数据源甚至没有数据结构。因此，文本预处理就是要对半结构化或非结构化的文本进行格式和结构的转换、分解和预处理等，以得到能够用于进一步处理的基础文本。不同环境下，文本所需的预处理工作内容也有所不同，大体上分为以下几个部分：

1.基本处理

根据不同的文本数据来源，可能涉及的基本文本处理包括去除无效标签、编码转换、文档切分、基本纠错、去除空白、大小写统一、去标点符号、去停用词、保留特殊字符等。

·去除无效标签：例如从网页源代码获取的文本信息中包含HTML标签，此时要提取特定标签内容并去掉标签。

·编码转换：不同编码转换对于中文处理具有较大影响，例如UTF-8、UTF-16、GBK、GB2312等之间的转换。

·文档切分：如果要获得单个文档中包含多个文件，需要进行单独切分以将不同的文档拆分出来。

·基本纠错：对于文本中明显的人名、地名等常用语和特定场景用语的错误进行纠正。

·去除空白：文本中可能包含的大量空格、空行等需要去除。

·大小写统一：将文本中的英文统一为大写或小写。

·去标点符号：去除句子中的标点符号、特殊符号等。

·去停用词：常见的停用词包括the、a、an、and、this、those、over、under、above、on等。

·保留特殊字符：某些场景下可能需要只针对汉字、英文或数字进行处理，其他字符都需要过滤掉。

2.分词

分词是将一系列连续的字符串按照一定逻辑分割成单独的词。在英文中，单词之间是以空格作为自然分界符的；而中文只有字、句和段能通过明显的分界符来简单划界，而作为词是没有形式上的分界符的。因此，中文分词要比英语等语种分词困难和复杂。对于复杂的中文分词而言，常用的分词方法包括最大匹配法、逆向最大匹配法、双向匹配法、最佳匹配法、联想-回溯法等。

3.文本转向量（word to vector）

人们通常采用向量空间模型来描述文本向量，即将文档作为行，将分词后得到的单词（单词在向量空间模型里面被称为向量，又称特征、维度或维）作为列，而矩阵的值则是通过词频统计算法得到的值。这种空间向量模型又称文档特征矩阵。其表示方法如表3-4所示。

表3-4　空间向量模型示例

	特征 1	特征 2	特征 3
文档 1	0.015	0.043	0.018
文档 2	0.013	0.081	0.092
文档 3	0.079	0.017	0.018

本示例中，将仅对自然语言文本做分词和word to vector处理，更多有关文本分析的内容，例如词性标注、关键字提取、词频统计、文本聚类、相似关键字分析等会在第4章中介绍。数据源文件text.txt位于“附件-chapter3”中，默认工作目录为“附件-chapter3”（如果不是，请切换到该目录下，否则会报“IOError:File text.txt does not exist”）。完整代码如下：

```
# 导入库
import pandas as pd
import jieba  # 结巴分词
from sklearn.feature_extraction.text import TfidfVectorizer  # 基于TF-IDF的词频转向量库
```

```
# 分词函数
def jieba_cut(string):
    word_list = []  # 建立空列表用于存储分词结果
    seg_list = jieba.cut(string)  # 精确模式分词
    for word in seg_list:  # 循环读取每个分词
        word_list.append(word)  # 分词追加到列表
    return word_list

# 读取自然语言文件
fn = open('text.txt')
string_lines = fn.readlines()
fn.close()

# 中文分词
seg_list = []  # 建立空列表，用于存储所有分词结果
for string_line in string_lines:  # 读取每行数据
    each_list = jieba_cut(string_line)  # 返回每行的分词结果
    seg_list.append(each_list)  # 分词结果添加到结果列表
for i in range(5):  # 打印输出第一行的前5条数据
    print (seg_list[1][i])

# word to vector
stop_words = [u'\n', u'/', u'“', u'“', u'”', u'的', u', ', u'和', u'是', u'随
着', u'对于', u'对', u'等', u'能', u'都', u'。', u'、',
              u'中', u'与', u'在', u'其']  # 自定义要去除的无用词
vectorizer = TfidfVectorizer(stop_words=stop_words, tokenizer=jieba_cut)  #
建词向量模型
X = vectorizer.fit_transform(string_lines)  # 将文本数据转换为向量空间模型
vector = vectorizer.get_feature_names()  # 获得词向量
vector_value = X.toarray()  # 获得词向量值
vector_pd = pd.DataFrame(vector_value, columns=vector)  # 创建用于展示的数据框
print (vector_pd.head(1))  # 打印输出第一条数据
```

上述代码用空行分为5个部分。

第一部分导入库。本代码中用到jieba作为中文分词，sklearn用于word to vector转换，Pandas用于格式化输出。

第二部分建立一个分词函数。该函数用于下面的中文分词。先建立一个空列表用于存储分词结果；使用jieba.cut做中文分词；循环读取每个分词结果并存储到列表中，最后返回。

第三部分读取自然语言文件。使用Python标准方法open读取文本文件，使用read-lines方法读取为列表。

第四部分中文分词。新建一个空列表，用于存储每次分词返回的结果；使用for循环将原文本数据按行读取，并调用分词函数做中文分词，并将结果追加到列表中。最后通过循环打印输出前5条数据，具体如下：

对于

数据

化

运营

和

第六部分word to vector。先定义一个要去除的停用词库，然后使用TfidfVectorizer方法词向量模型，将使用fit_transform方法对输入的分词后的列表做转换。最后通过词向量的get_feature_names获得向量名称，通过转换后的向量使用toarray方法将向量结果转换为数组，再通过数据框做数据格式化。打印输出第一条数据如下：

```
    python     上     下          不断     不曾     专业          业     业务     两
个        严峻 ...   \
0     0.0  0.0  0.0  0.204648  0.0  0.0  0.204648  0.0  0.0  0.204648 ...
     非    非常   非常简单    预测    领域     高    高于     （     ）     ；
0  0.0  0.0   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
[1 rows x 198 columns]
```

在本示例中，没有涉及更多的自然语言文本预处理环节，例如无效标签、编码转换、文档切分、基本纠错、去除空白、大小写统一、去标点符号、去停用词、保留特殊字符等，原因是这些内容都是针对不同案例展开的，而本示例仅做功能演示，应对不同文本时，很难具有通用性和可复制性。另外，由于测试文件为本书的部分文字内容，文本规模本身有限，因此难以提取真正有价值的数据和规律出来，尤其是基于向量化的分词，由于数据量的限制以及作为扩展内容，无法做更多有价值的探索。

上述过程中，需要考虑的关键点是：如何根据不同自然语言的来源特点、应用场景、语言语法、目标应用做综合的文本处理？文本处理中用到的都是针对文字内容的过滤、筛选、去除、替换等主要基于字符串的操作。

本小节示例中，主要用了如下几个知识点：

·对文本文件的读写操作；

·通过join方法实现多字符串的拼接；

·通过encode方法做字符编码转换；

·使用jieba.cut做中文分词，并可设置不同的分词模式；

·通过append方法对列表追加元素；

·通过for循环读取数据列表；

·通过if语句进行条件判断；

·通过sklearn.feature_extraction.text的TfidfVectorizer方法做word to vector处理；

·通过词向量的get_feature_names获得向量名称；

·通过转换后的向量使用toarray方法将向量结果转换为数组；

·使用pandas.DataFrame建立数组并通过head方法输出前n条数据。

3.13 本章小结

内容小结：本章介绍了11条有关数据化运营过程中的数据预处理经验，涵盖了常见的数据清洗、标志转换、数据降维、样本不均衡、数据源冲突、抽样、共线性、相关性分析、数据标准化、数据离散化等内容，并在最后提出了运营业务对于数据处理的影响和应对措施。在扩展内容中简单介绍了有关网页、日志、图像、自然语言的文本预处理工作。本章涉及技术的部分都有对应示例代码，这些代码可在“附件-chapter3”文件夹中的名为chapter3_code.py的文件中找到。

重点知识：客观上讲，本章的每一节内容都非常重要，原因是所有的内容都没有唯一答案，都需要读者根据不同的场景进行判别，然后选择最合适的方法进行处理。因此，掌握每种方法的适用条件以及如何辨别其应用前提是关键。

外部参考：限于篇幅，本书涉及很多内容无法一一展开介绍，以下给出更多外部参考资源供读者学习：

·Python第三方库imblearn提供了非常多的样本不均衡处理方法，尤其是SMOTE、组合、集成方法的应用。读者可在https://github.com/scikit-learn-contrib/imbalanced-learn中找到更多信息。

·本书中多次引用了Sklearn中的processing库，里面还有许多有关数据处理的方法，读者可查阅http://scikit-learn.org/stable/modules/preprocessing.html进一步了解。

·关于数据的预处理，Pandas库真的非常好用，推荐读者深入了解和学习，点击http://pandas.pydata.org/pandas-docs/stable/可查看更多。

应用实践：本章几乎每个小节都带有示例代码，读者可直接使用附件中的示例数据进行模拟操作，以了解实现方法；同时，推荐读者从自己所在环境中找到一些真实数据，针对每个模块进行操作练习。

第4章　跳过运营数据分析和挖掘的“大坑”

在经过第2章的数据获取、第3章的数据预处理之后，终于到达数据分析和挖掘环节了。本章将围绕常见的数据分析和挖掘过程中遇到的问题和困惑展开讲解，希望能帮助读者理清思路并找到应对或解决问题的方法。本章内容涵盖聚类、回归、分类、关联、异常检测、时间序列、路径分析、漏斗分析、归因分析、热力图分析及其他统计分析相关话题；有关聚类、回归、分类、关联、异常检测和时间序列的部分，本章通过Python程序辅助功能实现。

4.1 聚类分析

聚类是数据挖掘和计算的基本任务，是将大量数据集中具有“相似”特征的数据点或样本划分为一个类别。聚类分析的基本思想是“物以类聚、人以群分”，因此大量的数据集中必然存在相似的数据样本，基于这个假设就可以将数据区分出来，并发现不同类的特征。

聚类常用于数据探索或挖掘前期，在没有做先验经验的背景下做的探索性分析，也适用于样本量较大情况下的数据预处理工作。例如针对企业整体的用户特征，在未得到相关知识或经验之前先根据数据本身特点进行用户分群，然后针对不同群体做进一步分析；例如对连续数据做离散化，便于后续做分类分析应用。

常用的聚类算法分为基于划分、层次、密度、网格、统计学、模型等类型的算法，典型算法包括K均值（经典的聚类算法）、DBSCAN、两步聚类、BIRCH、谱聚类等。

聚类分析能解决的问题包括：数据集可以分为几类、每个类别有多少样本量、不同类别中各个变量的强弱关系如何、不同类别的典型特征是什么等；除了划分类别外，聚类还能用于基于类别划分的其他应用，例如图片压缩等。但是，聚类无法提供明确的行动指向，聚类结果更多是为后期挖掘和分析工作提供预处理和参考，无法回答“为什么”和“怎么办”的问题，更无法为用户提供明确的解决问题的规则和条件（例如决策树条件或关联规则）。因此，聚类分析无法真正解决问题。

4.1.1 当心数据异常对聚类结果的影响

K均值（K-Means）是聚类中最常用的方法之一，它基于点与点距离的相似度来计算最佳类别归属。但K均值在应用之前一定要注意两种数据异常：

1）数据的异常值。数据中的异常值能明显改变不同点之间的距离相似度，并且这种影响是非常显著的。因此基于距离相似度的判别模式下，异常值的处理必不可少。

2）数据的异常量纲。不同的维度和变量之间，如果存在数值规模或量纲的差异，那么在做距离之前需要先将变量归一化或标准化。例如跳出率的数值分布区间是[0，1]，订单金额可能是[0，10000000]，而订单数量则是[0，1000]，如果没有归一化或标准化操作，那么相似度将主要受到订单金额的影响。

当然，K均值并不是唯一的聚类方法，如果上述两种条件受某些因素的限制无法实现，那么可以选择其他聚类方法，例如DBSCAN。DBSCAN的全称是Density-Based Spatial Clustering of Applications with Noise，中文含义是“基于密度的带有噪声的空间聚类”。DBSCAN是一个比较有代表性的基于密度的聚类算法，与基于划分和层次的聚类方法不同，它将簇定义为密度相连的点的最大集合，能够把具有足够高密度的区域划分为簇，并可在噪声的空间数据集中发现任意形状的聚类。

DBSCAN算法的出发点是基于密度寻找被低密度区域分离的高密度空间，以此来实现不同数据样本的聚类。跟K均值相比，它具有以下优点：

·原始数据集的分布规律没有明显要求，能适应任意数据集分布形状的空间聚类，因此数据集适用性更广，尤其是对非凸装、圆环形等异形簇分布的识别较好。

·无须指定聚类数量，对结果的先验要求不高。

·由于DBSCAN可区分核心对象、边界点和噪音点，因此对噪声的过滤效果好，能有效应对数据噪点。

但是，由于它直接对整个数据集进行操作且聚类时使用了一个全局性的表征密度的参数，因此也存在几个比较明显的弱点：

·对于高维问题，基于Eps（半径）和MinPts（密度）的定义是个很大问题。

·当簇的密度变化太大时，聚类结果较差。

·当数据量增大时，要求较大的内存支持，I/O消耗也很大。

4.1.2 超大数据量时应该放弃K均值算法

K均值在算法稳定性、效率和准确率（相对于真实标签的判别）上表现非常好，并且在应对大量数据时依然如此。它的算法时间复杂度上界为O（n*k*t），其中n是样本量、k是划分的聚类数、t是迭代次数。当聚类数和迭代次数不变时，K均值的算法消耗时间只跟样本量有关，因此会呈线性增长趋势。

当真正面对海量数据时，使用K均值算法将面临严重的结果延迟，尤其是当K均值被用于做实时性或准实时性的数据预处理、分析和建模时，这种瓶颈效应尤为明显。到底K均值的时间随着样本量的增加会如何变化？笔者利用Python生成一个具有三个分类类别的样本集，每个样本都是二维空间分布（2个维度），样本量从100增长到1000000，步长为1000，我们用折线图来看一下计算时间与样本量的关系，结果如图4-1所示（这幅图耗时2个小时）。

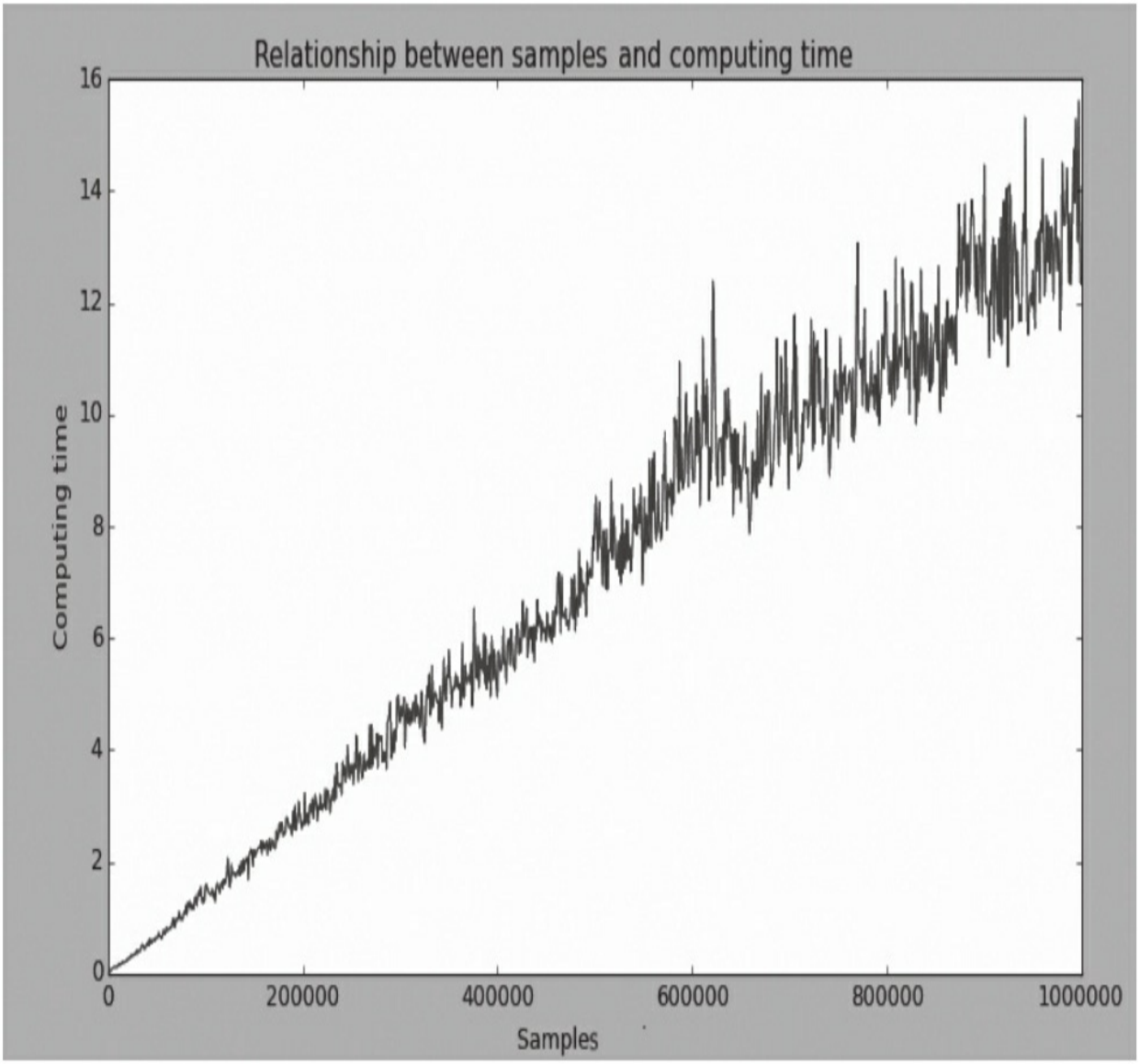


图4-1　K均值算法样本量与计算时间的增长关系

从图4-1所示结果中发现，计算时间跟数据量基本成线性关系。在样本点为二维空间的前提下，当数据量在200000以下时，计算时间基本都在2秒以内；当数据量在1000000时耗时近16秒。由此可以试想，如果样本点有更多样本量或需要更多聚类类别，那么耗时将比上述场景大很多。

针对K均值的这一问题，很多延伸算法出现了，MiniBatchKMeans就是其中一个典型代表。MiniBatchKMeans使用了一个种名为Mini Batch（分批处理）的方法计算数据点之间的距离。

Mini Batch的好处是计算过程中不必使用所有的数据样本，而是从不同类别的样本中抽取一部分样本（而非全部样本）作为代表参与聚类算法过程。由于计算样本量少，所以会相应减少运行时间；但另一方面，由于是抽样方法，抽样样本很难完全代表整体样本的全部特征，因此会带来准确度的下降。MiniBatchKMeans算法的准确度如何，请看下面的实验。在该实验中，我们对30000个样本点分别使用K-Means和Mini Batch KMeans进行聚类，然后对比两种方法得到结果的差异性，如图4-2所示。

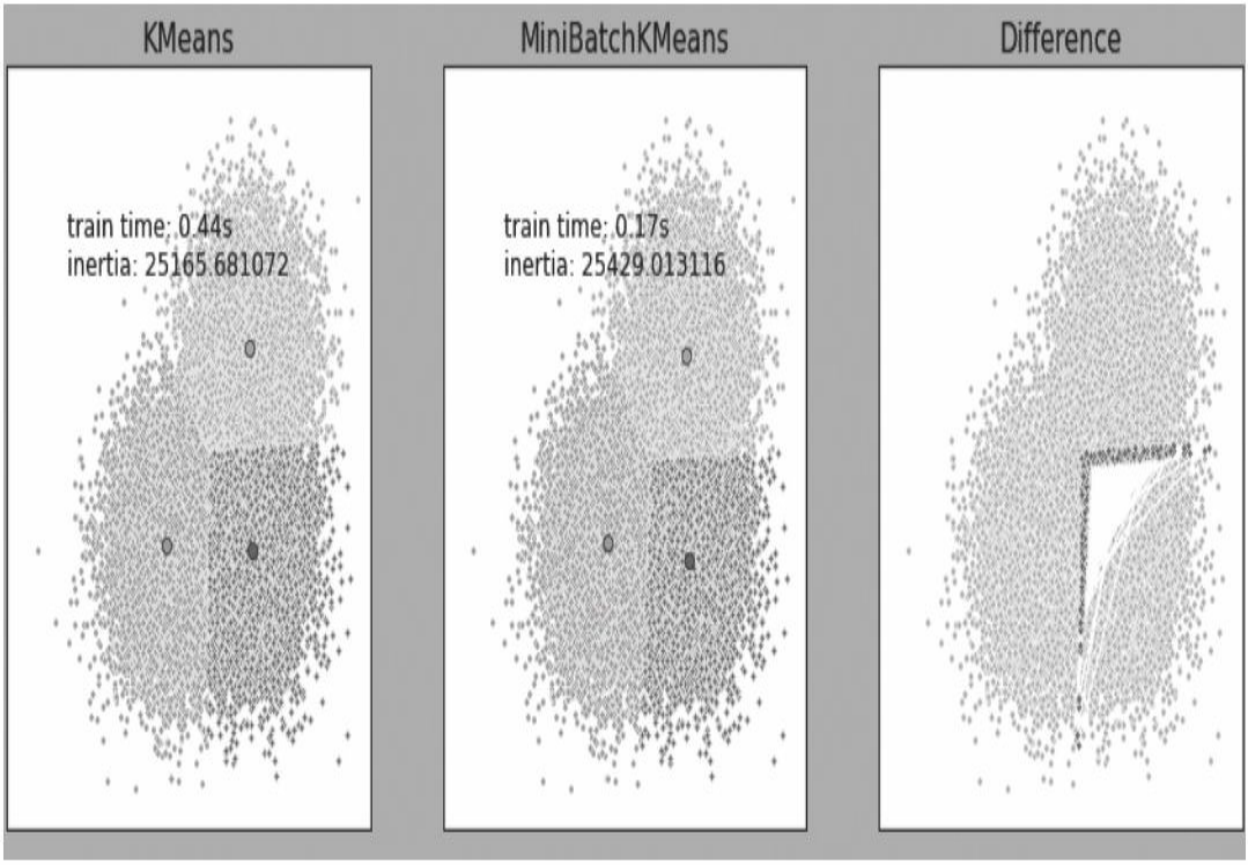


图4-2 K-Means和Mini Batch KMeans聚类结果对比

由图4-2所示结果发现：在30000样本点的基础上，二者的运行时间K-Mean是0.44秒，Mini Batch KMeans是0.17秒，计算时间的差距为2倍多（当然这在预期之内），但聚类结果差异性却很小（右侧样本代表差异分类样本）。

我们看到Mini Batch KMeans在基本保持了K-Means原有较高类别识别率的前提下，其计算效率的提升非常明显。因此，Mini Batch KMeans

是一种能有效应对海量数据，尽量保持聚类准确性并且大幅度降低计算耗时的聚类算法。

4.1.3 聚类不仅是建模的终点，更是重要的中间预处理过程

聚类分析的结果提供了样本集在非监督模式下的类别划分，这种划分得到的结果除了可以做群类别间的差异特征分析、群类别内的关键特征提取、样本群划分等分析功能外，还可以用于很多数据处理的中间过程。

1.图像压缩

图像压缩是用较少的数据量来表示原有的像素矩阵的过程，这个过程称为图像编码。数字图像的显著特点是数据量庞大，需要占用相当大的存储空间，这给图像的存储、计算、传输等带来了不便。因此，现在大多数数字网络下的图像都会经过压缩后再做进一步应用，图像压缩的方法之一便是聚类算法。

在使用聚类算法做图像压缩时，我们会先定义K个颜色数（例如128种颜色、256种颜色），颜色数就是聚类类别的数量；K均值聚类算法会把类似的颜色分别放在K个簇中，然后每个簇使用一种颜色来代替原始颜色，那么结果就是有多少个簇，就生成了由多少种颜色构成的图像，由此实现图像压缩。

2.图像分割

图像分割就是把图像分成若干个特定的、具有独特性质的区域并提出感兴趣的目标技术和过程，这是图像处理和分析的关键步骤。图像分割后提取出的目标可以用于图像语义识别、图像搜索等领域。例如从图像中分割出前景人脸信息，然后做人脸建模和识别。

聚类算法是图像分割方法的一种，其实施的关键是通过不同区域间明显不同的图像色彩特征做聚类，聚类数量就是要分割的区域的数量。

3.图像理解

传统的图像理解大多借助于浅层的视觉特征，如颜色、纹理、形状、轮廓等形成所谓的内容理解，实质上这是计算机对图像内容的理解而非人类的理解。一幅图像中包含的信息和潜在语义远远比视觉特征所

能表达的要丰富得多。图像理解就是要从图像中获得这些深层次的信息，既包括对图像中存在的对象及对象之间空间关系的理解，也包括对隐含在图像背后更为丰富的概念和内容的感受。

在图像理解中，有一种称为基于区域的提取方法。基于区域的提取方法是在图像分割和对象识别的前提下进行的，利用对象模板、场景分类器等，通过识别对象及对象之间的拓扑关系挖掘语义，生成对应的场景语义信息。例如，先以颜色、形状等特征对分割后的图像区域进行聚类，形成少量BLOB；然后通过CMRM模型计算出BLOB与某些关键词共同出现的概率。

4.异常检测

异常检测有多种实施方法，其中常用的方法是基于距离的异常检测方法。该方法包含并拓展了基于统计的思想，即使数据集不满足任何特定分布模型，它仍能有效地发现离群点，特别是当空间维数比较高时，算法的效率比基于密度的方法要高得多。算法具体实现时，首先算出数据样本间的距离（如曼哈顿距离、欧氏距离等），然后对数据做预处理后就可以根据距离的定义来检测异常值。

例如，基于K-Means的聚类可以将离中心点最远的类或者不属于任何一个类的数据点提取出来，然后将其定义为异常值。基于距离的离群检测方法不需要用户拥有任何领域的先验知识，且具有比较直观的识别意义，算法比较容易理解，因此在实际中应用得比较多。

5.数据离散化

聚类算法可以用于数据离散化，该部分内容请查看3.10.3节。

4.1.4 高维数据上无法应用聚类吗

在大数据背景下，数据获取难度和成本非常低，很多高维数据场景，例如电子商务交易数据、Web文本数据日益丰富。在做高维数据聚类时，传统的在低维空间通用的聚类方法运用到高维空间时，通常不能取得令人满意的聚类结果，这主要表现在聚类计算耗时太长、聚类结果相对于真实标签分类的准确性和稳定性都大大降低。

为什么在高维空间下聚类会出现这种问题？

·在面对高维数据时，基于距离的相似度计算效率极低；

·高维空间的大量属性特征使得在所有维上存在簇的可能性非常低；

·由于稀疏性及近邻特性，基于距离的相似度几乎都为0，导致高维的空间中很难存在数据簇。

应对高维数据的聚类主要有2种方法：降维、子空间聚类。

降维是应对高维数据的有效办法，通过特征选择法或维度转换法将高维空间降低或映射到低维空间，直接解决了高维问题。有关降维的内容请参照3.3节。

子空间聚类算法是在高维数据空间中对传统聚类算法的一种扩展，其思想是选取与给定簇密切相关的维，然后在对应的子空间进行聚类。比如谱聚类就是一种子空间聚类方法，由于选择相关维的方法以及评估子空间的方法需要自定义，因此这种方法对操作者的要求较高。

4.1.5 如何选择聚类分析算法

聚类算法有几十种之多，聚类算法的选择，主要参考以下因素：

·如果数据集是高维的，那么选择谱聚类，它是子空间划分的一种。

·如果数据量为中小规模，例如在100万条以内，那么K均值将是比较好的选择；如果数据量超过100万条，那么可以考虑使用Mini Batch KMeans。

·如果数据集中有噪点（离群点），那么使用基于密度的DBSCAN可以有效应对这个问题。

·如果追求更高的分类准确度，那么选择谱聚类将比K均值准确度更好，在Docu-ment clustering using locality preserving indexing中关于K-means和Spectral Clustering应用到TDT2和Reuters-21578两组数据的准确率对比结果证明了这个结论。

4.1.6 代码实操：Python聚类分析

Sklearn中有专门的聚类库cluster，在做聚类时只需导入这个库，便可使用其中多种聚类算法，例如K均值、DBSCAN、谱聚类等。

本示例模拟的是对一份没有任何标签的数据集做聚类分析，以得到不用类别的特征和分布状态等，主要使用Sklearn做聚类、用Matplotlib做图形展示。数据源文件clustring.txt位于“附件-chapter4”中，默认工作目录为“附件-chapter4”（如果不是，请切换到该目录下，否则会报“IOError:File clustring.txt does not exist”）。完整代码如下：

```
# 导入库
import numpy as np  # 导入Numpy库
import matplotlib.pyplot as plt  # 导入Matplotlib库
from sklearn.cluster import KMeans  # 导入sklearn聚类模块
from sklearn import metrics  # 导入sklearn效果评估模块
# 数据准备
raw_data = np.loadtxt('cluster.txt')  # 导入数据文件
X = raw_data[:, :-1]  # 分割要聚类的数据
y_true = raw_data[:, -1]

# 训练聚类模型
n_clusters = 3  # 设置聚类数量
model_kmeans = KMeans(n_clusters=n_clusters, random_state=0)  # 建立聚类模型对象
model_kmeans.fit(X)  # 训练聚类模型
y_pre = model_kmeans.predict(X)  # 预测聚类模型

# 模型效果指标评估
n_samples, n_features = X.shape  # 总样本量,总特征数
inertias = model_kmeans.inertia_  # 样本距离最近的聚类中心的总和
adjusted_rand_s = metrics.adjusted_rand_score(y_true, y_pre)  # 调整后的兰德指数
mutual_info_s = metrics.mutual_info_score(y_true, y_pre)  # 互信息
adjusted_mutual_info_s = metrics.adjusted_mutual_info_score(y_true, y_pre)  整后的互信息
homogeneity_s = metrics.homogeneity_score(y_true, y_pre)  # 同质化得分
completeness_s = metrics.completeness_score(y_true, y_pre)  # 完整性得分
v_measure_s = metrics.v_measure_score(y_true, y_pre)  # V-measure得分
silhouette_s = metrics.silhouette_score(X, y_pre, metric='euclidean')  # 平均轮廓系数
calinski_harabaz_s = metrics.calinski_harabaz_score(X, y_pre)  # Calinski和Harabaz得分
print ('samples: %d \t features: %d' % (n_samples, n_features))  # 打印输出样本量和特征数量
print (70 * '-')  # 打印分隔线
print ('ine\tARI\tMI\tAMI\thomo\tcomp\tv_m\tsilh\tc&h')  # 打印输出指标标题
print ('%d %.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%d' % (
inertias, adjusted_rand_s, mutual_info_s, adjusted_mutual_info_s, homogeneit
silhouette_s, calinski_harabaz_s))  # 打印输出指标值
print (70 * '-')  # 打印分隔线
print ('short name \t full name')  # 打印输出缩写和全名标题
```

```
print ('ine \t inertias')
print ('ARI \t adjusted_rand_s')
print ('MI \t mutual_info_s')
print ('AMI \t adjusted_mutual_info_s')
print ('homo \t homogeneity_s')
print ('comp \t completeness_s')
print ('v_m \t v_measure_s')
print ('silh \t silhouette_s')
print ('c&h \t calinski_harabaz_s')

# 模型效果可视化
centers = model_kmeans.cluster_centers_  # 各类别中心
colors = ['#4EACC5', '#FF9C34', '#4E9A06']  # 设置不同类别的颜色
plt.figure()  # 建立画布
for i in range(n_clusters):  # 循环读类别
    index_sets = np.where(y_pre == i)  # 找到相同类的索引集合
    cluster = X[index_sets]  # 将相同类的数据划分为一个聚类子集
    plt.scatter(cluster[:, 0], cluster[:, 1], c=colors[i], marker='.')  # 展
示聚类子集内的样本点
                                    plt.plot(centers[i][0],         centers[i]
[1], 'o', markerfacecolor=colors[i], markeredgecolor='k', markersize=6)  # 展
示各聚类子集的中心
plt.show()  # 展示图像

# 模型应用
new_X = [1, 3.6]
cluster_label = model_kmeans.predict(new_X)
print ('cluster of new data point is: %d' % cluster_label)
```

上述代码用空行分为6部分。

第一部分导入库。示例中用到了Numpy导入文件、Sklearn做聚类模型应用以及效果评估、Matplotlib展示聚类结果。

第二部分数据准备。使用Numpy的loadtxt方法导入数据文件。数据文件包含1000条样本，3列维度，其中第3列为目标分类标签；在获取原始数据后，使用索引切片对矩阵做切分，将数据分为输入X和标签y_true。

第三部分训练聚类模型。先设置聚类数量为3，并建立聚类模型对象，然后通过fit方法训练模型，通过predict方法做聚类应用得到原始训练集的聚类标签集y_pre（也可以在应用fit方法后直接从聚类对象的labels_属性获得训练集的聚类标签）；从得到的聚类模型中，通过其cluster_centers_属性和inertia_属性得到各类别中心以及样本距离最近的聚类中心的总和。

在sklearn中，几乎所有的算法类库都使用相同的模式做算法训练：①通过算法函数建立算法模型对象，②对模型对象使用fit方法做模型训练，③对模型对象应用predict方法做模型预测。虽然聚类属于非监督式算法，但也使用这种模式，这种通用用法使得sklearn具有极好的易用性。除了fit和predict方法外，算法类库的其他方法还有：score、get_params、set_parames等通用方法；另外，不同的算法库还有特定的方法函数，例如SVM的decision_function、KMeans的transform、PCA的inverse_transform、DecisionTreeClassifier的predict_proba等。

相关知识点：将算法对象保存到硬盘

在某些情况下，我们可能需要将算法对象保存到硬盘，以方便日后再次做模型应用和预测时无需重新训练模型，并且可以保持相同的模型参数和实例来应用到新的数据集，这种用法也称为对象持久化。Python内置标准库cPickle是实现这一过程的有效方法库。

cPickle可以将任意一种类型的Python对象进行序列化/持久化操作，算法模型对象也不例外。cPickle的主要应用方法是dump和load。

dump：将Python对象序列化保存到本地的文件，方法如下：

```
import cPickle
cPickle.dump(model_name,open("my_model_object.pkl","wb"))
```

load：从本地文件读取Python对象并恢复实例对象，方法如下：

```
model_name = cPickle.load(open("my_model_object.pkl","rb"))
```

上述方法中，model_name就是模型的实例化名称，例如本节代码中的model_kmeans，而扩展名为.pkl的文件名称则自定义即可。

第四部分模型效果指标评估。本部分将通过不同的指标来做聚类效果评估。

·inertias：inertias是K均值模型对象的属性，表示样本距离最近的聚

类中心的总和，它是作为在没有真实分类结果标签下的非监督式评估指标。该值越小越好，值越小证明样本在类间的分布越集中，即类内的距离越小。

·adjusted_rand_s：调整后的兰德指数（Adjusted Rand Index），兰德指数通过考虑在预测和真实聚类中在相同或不同聚类中分配的所有样本对和计数对来计算两个聚类之间的相似性度量。调整后的兰德指数通过对兰德指数的调整得到独立于样本量和类别的接近于0的值，其取值范围为[-1，1]，负数代表结果不好，越接近于1越好意味着聚类结果与真实情况越吻合。

·mutual_info_s：互信息（Mutual Information，MI），互信息是一个随机变量中包含的关于另一个随机变量的信息量，在这里指的是相同数据的两个标签之间的相似度的量度，结果是非负值。

·adjusted_mutual_info_s：调整后的互信息（Adjusted Mutual Information，AMI），调整后的互信息是对互信息评分的调整得分。它考虑到对于具有更大数量的聚类群，通常MI较高，而不管实际上是否有更多的信息共享，它通过调整聚类群的概率来纠正这种影响。当两个聚类集相同（即完全匹配）时，AMI返回值为1；随机分区（独立标签）平均预期AMI约为0，也可能为负数。

·homogeneity_s：同质化得分（Homogeneity），如果所有的聚类都只包含属于单个类的成员的数据点，则聚类结果将满足同质性。其取值范围[0，1]值越大意味着聚类结果与真实情况越吻合。

·completeness_s：完整性得分（Completeness），如果作为给定类的成员的所有数据点是相同集群的元素，则聚类结果满足完整性。其取值范围[0，1]，值越大意味着聚类结果与真实情况越吻合。

·v_measure_s：它是同质化和完整性之间的谐波平均值，v=2*（均匀性*完整性）/（均匀性+完整性）。其取值范围[0，1]，值越大意味着聚类结果与真实情况越吻合。

·silhouette_s：轮廓系数（Silhouette），它用来计算所有样本的平均轮廓系数，使用平均群内距离和每个样本的平均最近簇距离来计算，它是一种非监督式评估指标。其最高值为1，最差值为-1，0附近的值表示

重叠的聚类，负值通常表示样本已被分配到错误的集群。

·calinski_harabaz_s：该分数定义为群内离散与簇间离散的比值，它是一种非监督式评估指标。

Sklearn中算法效果评估一共有3种方式：一是算法模型对象的score方法，在创建的算法对象中可直接使用该方法，例如model_tree.score（X_test，y_test）；二是使用交叉验证的模型评估工具的评分参数，例如使用sklearn.model_selection.cross_val_score（）方法返回的得分为交叉检验得分数组；三是metrics库的评估指标，该库可用来对模型选择、分类算法、聚类算法、回归算法、多标签学习、双聚类、配对（Pairwise）算法等主题进行评估，本章用到的大量的算法评估指标都来源于此。

在本部分功能中，先通过shape方法获得输入训练集的形状，包括样本量和特征数量；然后通过模型的inertia_属性获得样本距离最近的聚类中心的总和；接着通过一系列Sklearn中的指标评估模块的函数，对真实类别结果和预测类别结果做对比评估，最后打印输出上述指标。

在打印过程中，通过print方法结合占位符打印输出不同的字段以及对应值，这种方法对于值特别多、且打印值是变量的场景下尤其有效；另外也应用了tab分隔符（\t）来分隔字段内容、使用不同的占位符类型做内容限制（例如浮点型f、整数型d）使内容格式化效果更好；为了区分不同的打印内容，例如总描述、各评估指标、指标缩写，使用分割线来让内容逻辑上更清晰。输出结果如下：

```
samples: 1000       features: 2
------------------------------------------------------------------------------
ineARI        MI       AMI     homo     comp      v_m     silh     c&h
300 0.96     1.03      0.94      0.94      0.94       0.94       0.63
------------------------------------------------------------------------------
short name       full name
ine       inertias
ARI       adjusted_rand_s
MI            mutual_info_s
AMI       adjusted_mutual_info_s
homo       homogeneity_s
comp       completeness_s
v_m       v_measure_s
silh       silhouette_s
c&h       calinski_harabaz_s
```

相关知识点：字符串格式化

在使用print打印方法中，经常会用到字符串格式化操作。常用的格式化符号如表4-1所示：

表4-1　常用格式化符号

格　式	描　述	举　例
%s	字符串	输入：print ('name: %s'%'tony') 输出：name: tony
%d	有符号整数（十进制）	输入：print ('I am %d years old'%30) 输出：I am 30 years old

（续）

格　式	描　述	举　例
%e	浮点数字（科学计数法）	输入：print ('it is %e yuan'% 1 000 000 000) 输出：it is 1.000000e+09 yuan
%E	浮点数字（科学计数法，用E代替e）	输入：print ('it is %E yuan'% 1 000 000 000) 输出：it is 1.000000E+09 yuan
%f	浮点数字（用小数点符号）	输入：print ('the score is %f'% 0.12 612) 输出：the score is 0.126120
%g	浮点数字（根据值的大小采用%e或%f）	输入：print ('the score is %g'% 0.12 612) 输出：the score is 0.12612

在使用print输出时同时输出多个字符串的方法如下：

```
输入：print ('the score is %d ad result is %f' % (1298,1.314))
输出：the score is 1298 ad result is 1.314000
```

对于浮点型数据，如果要保留特定位数的小数，方法如下：

```
输入：print ('the score is %d ad result is %.2f' % (1298,1.314))
输出：the score is 1298 ad result is 1.31
```

其中.2f表示输出的浮点数保留2位四舍五入的小数。

通过上述各个指标，发现K均值的预测结果跟实际结果相似度非常高，证明算法本身的拟合度非常符合实际情况。

第五部分模型效果可视化。先通过K均值模型对象的cluster_centers_方法获得类别中心点；然后定义一个颜色集用来显示不同类别样本的颜色；通过Matplotlib的figure方法建立一块画布，再通过for循环读出每个类别，使用Numpy的where方法得到不同分类下的数据集索引并建立各聚类数据子集，最后使用atplotlib的scatter方法和plot方法做散点图展示聚类子集内的样本点以及中心，如图4-3所示。

相关知识点：scatter和plot方法

scatter方法主要用来做散点图展示，而plot方法主要用来做折线图展示，也可以用于散点图的展示。两个方法的参数基本是通用的。以scatter方法为例，常用参数包括：

```
scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None,
=None, alpha=None, linewidths=None, verts=None, edgecolors=None, hold=None,
=None, **kwargs)
```

·x，y：具有相同长度的一维数据，散点的X和Y轴对应的数据。

·s：数值型，控制点的大小。

·c：字符串，控制点的颜色，常用的颜色字符串为b（blue，蓝色）、c（cyan，品红）、g（green，绿色）、k（black，黑色）、m（magenta，红色）、w（white，白色）、y（yellow，黄色）。

·marker：字符串，控制点的样式，样式列表如表4-2所示。

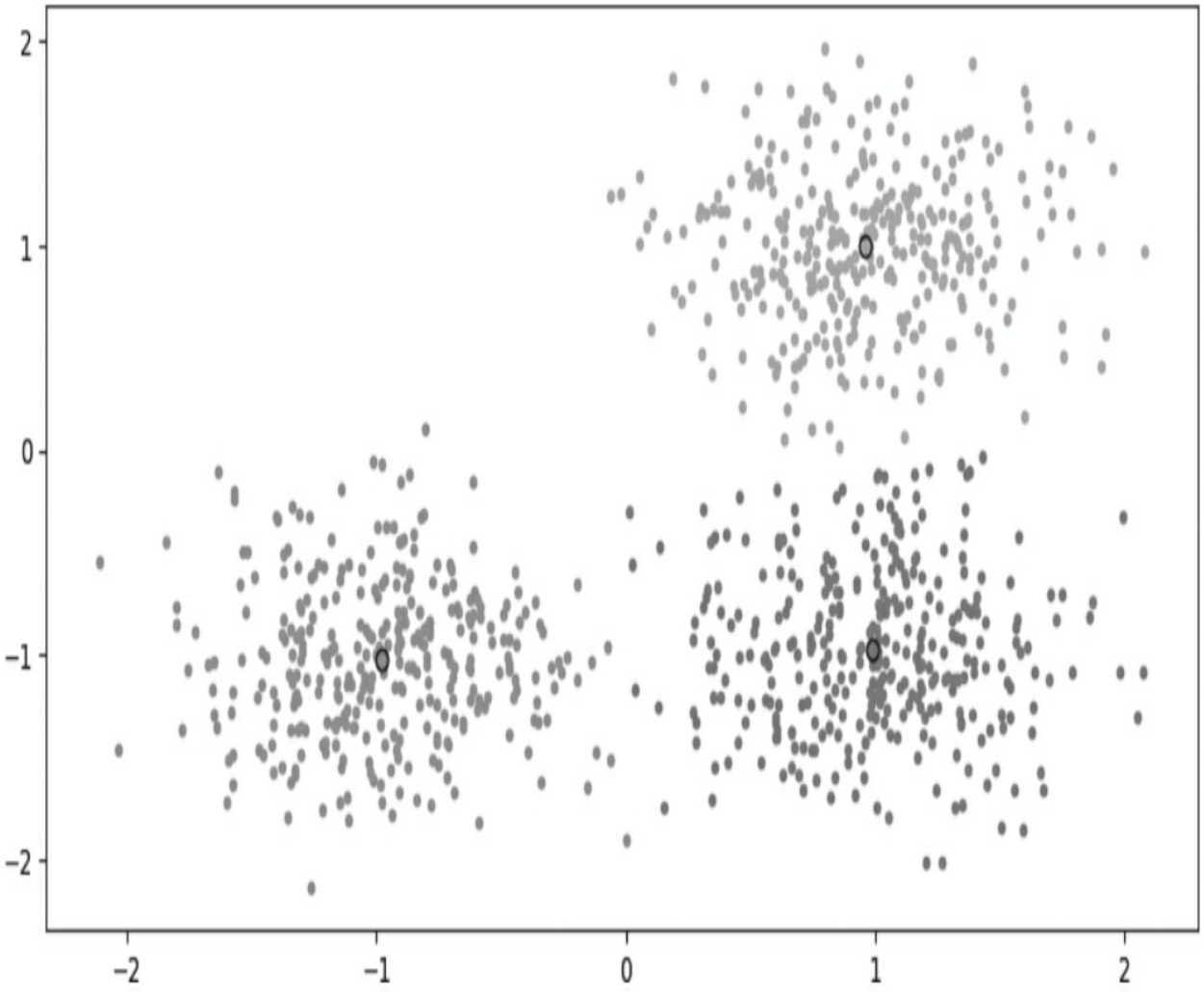


图4-3　聚类结果展示

表4-2　常用marker值列表

marker	描　述	marker	描　述	marker	描　述
.	点	<	右三角形	D	钻石
,	方形的像素	8	八角形	d	细钻石
o	圆圈	s	正方形	x	x号
v	倒三角形	p	五角形	|	竖线
^	正三角形	*	星形	_	横线
<	左三角形	+	加号		

在plot方法的marker值中，还可设定为线条的样式'-'为实线、'--'为虚线、'-.'为点虚线、':'为点虚线。

·norm：float型，控制点图的数据亮度，取值范围[0，1]。

·alpha：float型，控制点图透明度，取值范围[0，1]。

另外，在plot方法中，经常会用到label参数，用来实现多个线条的标签。

从聚类图可以看出，聚类结果点分布跟类别的分布基本一致，表现出了较好的类别间的区隔。如果要做进一步的各类别内的特征分析，可以基于各类别的数据集（在模型可视化部分有“将相同类的数据划分为一个聚类子集”，其中可以直接划分聚类子集），做特征的分析，例如均值、极值、方差、偏度、峰度、异常值、众数、中位数等不同角度的分析。

第六部分模型应用。假如新给到一个数据样本[1，3.6]，我们可以基于训练好的聚类模型预测该样本的类别归属，使用模型的predict方法进行预测。结果如下：

```
cluster of new data point is: 1
```

当然，这个类别的索引值本身没有多大意义，有意义的是，我们可以基于这个索引，提取出对应的类别的所有特征，然后基于这些特征或这个类别下的其他样本来对该样本做分析和应用。例如，当一个用户刚到达网站时，我们会这个用户了解非常少，无法具体给出最合适的信息，此时可以基于聚类算法将这个用户所属的类别进行推测，然后基于类别中的特征（其他用户喜欢的内容、经常浏览的帖子、TOP购买商品等）来给该用户做推荐。

上述过程中，主要需要考虑的关键点是：如何在不确定样本分类数的前提下定义K值，这也是K均值算法的关键问题之一。除此之外还有一个重要的话题是如何对聚类结果进行评估，对于聚类结果的评估主要考虑三个方面：

·一是对于没有任何聚类真实结果指标的，由于无法使用真实数据作对比，只能使用聚类距离指标做评估；

·二是对于有分类真实结果可做对照的（例如本示例），则可以使用真实标签与预测标签的相似、重复、完整性等度量计算，需要注意的是，聚类结果的标签值与其真实类别的标签值没有关系，结果只关注数据点是否属于同一类，而对于类的标签值是1还是2则不关注；

·三是业务类的评估，包括不同类别间的特征是否有显著差异，类内部是否具有能代表类别的显著性特征，不同类别内的样本量分布是否相对均匀等。另外，也可以通过可视化的形式，从聚类结果中发现是否存在异常的类别点等。

在本示例中（也包括本章的其他所有内容）没有涉及到数据预处理的知识，并且笔者也没打算这么做，原因是如果每次都演示从数据读取、预处理、建模、分析、应用、展示等所有场景，那么这本书就真的太厚了，并且知识也会过于重复。笔者更倾向于在不同的章节中介绍不同的内容，即使是完成同一件事情，也尽量通过不同的方法和思路实现，这样可以尽量提供更多的技巧和方法，读者在应用时选择自己最熟悉并且感兴趣的方法使用。

代码实操小结：本小节示例中，主要应用了以下几个知识点：

·通过Numpy的loadtxt读取数据文件。

·对矩阵做切片操作。

·使用sklearn.cluster的KMeans算法完成模型训练，并通过其cluster_centers_、inertia_属性输出聚类结果的中心和总距离。

·使用模型的predict方法做聚类预测并得到聚类类别索引值。

·使用sklearn.metrics中的adjusted_rand_score、mutual_info_score、adjusted_mutual_info_score、homogeneity_score、completeness_score、v_measure_score、silhouette_score、calinski_harabaz_score做基于真实数据结果标签的型效果评估。

·使用matplotlib.pyplot的scatter方法和plot方法展示图形，并设置不同的展示样式。前者主要用来展示散点图，后者主要用来展示趋势图，使用plot方法也可以做散点图展示，而且如果不设置颜色和图形大小，plot方法的展示速度会更快。

·使用print配合占位符输出变量，并使用不同的占位符类型（d/f等）并指定宽度做打印数据格式化。

4.2 回归分析

回归是研究自变量x对因变量y影响的一种数据分析方法。最简单的回归模型是一元线性回归（只包括一个自变量和一个因变量，且二者的关系可用一条直线近似表示），可以表示为$Y=\beta_0+\beta_1 x+\varepsilon$，其中Y为因变量，x为自变量，$\beta_1$为影响系数，$\beta_0$为截距，$\varepsilon$为随机误差。

回归分析是广泛应用的统计分析方法，可用于分析自变量和因变量的影响关系（通过自变量求因变量），也可以分析自变量对因变量的影响方向（正向影响还是负向影响）。回归分析的主要应用场景是进行预测和控制，例如计划制定、KPI制定、目标制定等方面；也可以基于预测的数据与实际数据进行比对和分析，确定事件发展程度并给未来行动提供方向性指导。

常用的回归算法包括线性回归、二项式回归、对数回归、指数回归、核SVM、岭回归、Lasso等。

回归分析的优点是数据模式和结果便于理解，如线性回归用y=ax+b的形式表达，在解释和理解自变量与因变量关系式相对容易；在基于函数公式的业务应用中，可以直接使用代入法求解，因此应用起来比较容易。回归分析的缺点是只能分析少量变量之间的相互关系，无法处理海量变量间的相互作用关系，尤其是变量共同因素对因变量的影响程度。

4.2.1 注意回归自变量之间的共线性问题

在应用回归模型时，应注意识别和解决自变量间的共线性问题。有关该话题，请参照3.7节了解更多信息。

4.2.2 相关系数、判定系数和回归系数之间到底什么关系

在应用回归模型时经常会得到类似的回归方程：y=42.738x+169.94，其中R2=0.5252。如果对这两个变量做相关性分析，还会得到相关系数R=0.72468551874050。

上述结果中，42.738就是自变量x的回归系数，0.5252则是该方程的判定系数，0.72468551874050是两个变量的相关性系数。这三个值之间到底什么关系？

·回归系数：在回归方程中表示自变量x对因变量y影响大小的参数，其绝对值的高低只能说明自变量与因变量之间的联系程度和变化量的比例。

自变量的单位不同会显著影响该系数，这意味着，量级越大的自变量，该系数越高。

·判定系数：是自变量对因变量的方差解释程度的值，计算公式为：回归平方和与总离差平方和之比值。

·相关系数：也称为解释系数，是衡量变量间的相关程度或密切程度的值，其本质是线性相关性的判断。

三者的相互关系：

·判定系数是所有参与模型中自变量的对因变量联合影响程度，而非某个自变量的影响程度。本节示例中只有1个自变量，因此R^2代表的是该自变量本身；如果有x1、x2两个自变量，那么R^2代表的这两个自变量共同影响的结果。假如在线性回归中只有一个自变量，那么判定系数等于相关系数的平方。

回归系数与相关系数的关系：回归系数＞0，相关系数取值在（0，1），说明二者正相关；如果系数小于0，相关系数取值在（-1，0），说明二者负相关。

对于判定系数的高低没有统一的标准，笼统而言，值越高越能代表自变量对因变量的解释作用越大。普通数据通常达到0.6就已经很高，0.6意味着相关系数至少在0.8以上；而对于时间序列数据，判定系数达到0.9则很平常。

4.2.3 判定系数是否意味着相应的因果联系

在3.8节中我们提到了相关性和因果不是一回事，那么判定系数是因果联系吗？其实也不是，以一元线性回归方程（本节开始的方程）举例：

·判定系数是相关系数的平方，既然相关系数不是因果，为什么其平方后就能成为因果？

·判定系数的出发点是用来评估整个模型的拟合优度，直白点就是自变量引起的变动占总变动的百分比。那么二者有相同的变动趋势就意味着是因果关系吗？显然也不是。

当然，需要承认在数学关系上，确实反映了当自变量x变化后，y根据方程也会有相应的变化。但是，二者的关系严格上讲无法判断，原因是我们很可能无法知晓回归中的自变量是否真的是导致因变量发生变化的根本因素。只是从现象上，将二者一起/相关变化的规律，使用回归方程来进行解释。

4.2.4 注意应用回归模型时研究自变量是否产生变化

在应用回归模型做预测时，必须研究对因变量产生影响的自变量是否产生变化，主要考察两个方面：

一是是否有产生了新的对因变量影响更大自变量

在建立回归模型时，需要综合考虑自变量的选择问题。如果遗漏了重要的变量，那么模型很可能无法正确反映实际情况，而且参数估计是有偏的，此时的回归模型及其不稳定且方差较大。同样在应用回归模型时，仍然需要重新评估该问题是否发生。例如，在做用户订单金额预测时是基于正常销售状态下的变量实现的；但当发生大型促销活动且促销活动因素没有被纳入到回归模型中时，原来的回归模型则无法有效预测。

二是原有自变量是否仍然控制在训练模型时的范围之内

如果有自变量的变化超过训练模型的范围，那么原来的经验公式可能无法在新的值域范围下适用，这种情况通常需要重新做研究和建模。例如，假设我们建立了一个回归模型，可以广告投放费用预测广告点击率，在训练回归模型时的广告费用在[0，1000]区间内，如果在广告费用超过1000（例如2000）时，则无法保证最终预测效果是否有效。

判定系数经常作为拟合好坏的主要参照指标，当一个新的指标加入模型后发现模型不变，此时无法根据判定系数来反推该指标的重要性程度，例如该指标无效（或有效）。

4.2.5 如何选择回归分析算法

回归算法按照自变量的个数分为一元回归和多元回归，按照影响是否线性分为线性回归和非线性回归。在面对不同回归方法的选择，注意参考以下因素：

·入门的开始——简单额线性回归。如果是学习为主，那么不需要选择多么强大的模型，基于最小二乘法的普通线性回归最合适；同时，它适合数据集本身结构简单、分布规律有明显线性关系的场景。

·如果自变量数量少或经过降维后得到了可以使用的二维变量（包括预测变量），那么可以直接通过散点图发现自变量和因变量的相互关系，然后选择最佳回归方法。

·如果经过基本判断发现自变量间有较强的共线性关系，那么可以使用对多重共线性（自变量高度相关）能灵活处理的算法，例如岭回归。

·如果数据集噪音较多，推荐使用主成分回归，因为主成分回归通过对参与回归的主成分的合理选择，可以去掉噪音。另外，各个主成分间相互正交，能解决多元线性回归中的共线性问题。这些都能有效地提高模型的抗干扰能力。

·如果在高维度变量下，使用正则化回归方法效果更好，例如Lasso，Ridge和ElasticNet，或者使用逐步回归从中挑选出影响显著的自变量来建立回归模型。

·如果要同时验证多个算法，并想从中选择一个来做做好的拟合，使用交叉检验做多个模型的效果对比，并通过R-square、Adjusted R-square、AIC、BIC以及各种残差、误差项指标做综合评估。

·如果注重模型的可解释性，那么容易理解的线性回归、指数回归、对数回归、二项或多项式回归要比核回归、支持向量回归机等更适合。

·集成或组合回归方法。一旦确认了几个方法，但又不确定如何取

舍时，可以将多个回归模型做集成或组合方法使用，即同时将多个模型的结果通过加权、均值等方式确定最终输出结果值。

4.2.6　代码实操：Python回归分析

sklearn中的回归有多种方法，广义线性回归集中在linear_model库下，例如普通线性回归、Lasso、岭回归等；另外还有其他非线性回归方法，例如核svm、集成方法、贝叶斯回归、K近邻回归、决策树回归等，这些不同回归算法分布在不同的库中。

本示例模拟的是针对一批训练集做多个回归模型的训练和评估，从中选择效果较好的模型并对新数据集做回归预测。本示例主要使用Sklearn的多个回归算法做回归分析、用Matplotlib做图形展示。数据源文件regression.txt位于“附件-chapter4”中，默认工作目录为“附件-chapter4”（如果不是，请cd切换到该目录下，否则会报“IOError:File regression.txt does not exist”）。完整代码如下：

```
# 导入库
import numpy as np  # numpy库
from sklearn.linear_model import BayesianRidge, LinearRegression, ElasticNet
量导入要实现的回归算法
from sklearn.svm import SVR  # SVM中的回归算法
from sklearn.ensemble.gradient_boosting import GradientBoostingRegressor  #
成算法
from sklearn.model_selection import cross_val_score  # 交叉检验
from sklearn.metrics import explained_variance_score, mean_absolute_error, m
量导入指标算法
import pandas as pd  # 导入pandas
import matplotlib.pyplot as plt  # 导入图形展示库

# 数据准备
raw_data = np.loadtxt('regression.txt')  # 读取数据文件
X = raw_data[:, :-1]  # 分割自变量
y = raw_data[:, -1]  # 分割因变量

# 训练回归模型
n_folds = 6  # 设置交叉检验的次数
model_br = BayesianRidge()  # 建立贝叶斯岭回归模型对象
model_lr = LinearRegression()  # 建立普通线性回归模型对象
model_etc = ElasticNet()  # 建立弹性网络回归模型对象
model_svr = SVR()  # 建立支持向量机回归模型对象
model_gbr = GradientBoostingRegressor()  # 建立梯度增强回归模型对象
model_names = ['BayesianRidge', 'LinearRegression', 'ElasticNet', 'SVR', 'GE
同模型的名称列表
model_dic = [model_br, model_lr, model_etc, model_svr, model_gbr]  # 不同回归
模型对象的集合
cv_score_list = []  # 交叉检验结果列表
pre_y_list = []  # 各个回归模型预测的y值列表
for model in model_dic:  # 读出每个回归模型对象
    scores = cross_val_score(model, X, y, cv=n_folds)  # 将每个回归模型导入交叉
检验模型中做训练检验
    cv_score_list.append(scores)  # 将交叉检验结果存入结果列表
```

```
    pre_y_list.append(model.fit(X, y).predict(X))  # 将回归训练中得到的预测y存
入列表

# 模型效果指标评估
n_samples, n_features = X.shape  # 总样本量,总特征数
model_metrics_name = [explained_variance_score, mean_absolute_error, mean_sc
归评估指标对象集
model_metrics_list = []  # 回归评估指标列表
for i in range(5):  # 循环每个模型索引
    tmp_list = []  # 每个内循环的临时结果列表
    for m in model_metrics_name:  # 循环每个指标对象
        tmp_score = m(y, pre_y_list[i])  # 计算每个回归指标结果
        tmp_list.append(tmp_score)  # 将结果存入每个内循环的临时结果列表
    model_metrics_list.append(tmp_list)  # 将结果存入回归评估指标列表
df1 = pd.DataFrame(cv_score_list, index=model_names)  # 建立交叉检验的数据框
df2   =   pd.DataFrame(model_metrics_list,   index=model_names,   columns=
['ev', 'mae', 'mse', 'r2'])  # 建立回归指标的数据框
print ('samples: %d \t features: %d' % (n_samples, n_features))  # 打印输出样
本量和特征数量
print (70 * '-')  # 打印分隔线
print ('cross validation result:')  # 打印输出标题
print (df1)  # 打印输出交叉检验的数据框
print (70 * '-')  # 打印分隔线
print ('regression metrics:')  # 打印输出标题
print (df2)  # 打印输出回归指标的数据框
print (70 * '-')  # 打印分隔线
print ('short name \t full name')  # 打印输出缩写和全名标题
print ('ev \t explained_variance')
print ('mae \t mean_absolute_error')
print ('mse \t mean_squared_error')
print ('r2 \t r2')
print (70 * '-')  # 打印分隔线

# 模型效果可视化
plt.figure()  # 创建画布
plt.plot(np.arange(X.shape[0]), y, color='k', label='true y')  # 画出原始值的
曲线
color_list = ['r', 'b', 'g', 'y', 'c']  # 颜色列表
linestyle_list = ['-', '.', 'o', 'v', '*']  # 样式列表
for i, pre_y in enumerate(pre_y_list):  # 读出通过回归模型预测得到的索引及结果
    plt.plot(np.arange(X.shape[0]), pre_y_list[i], color_list[i], label=mode
出每条预测结果线
plt.title('regression result comparison')  # 标题
plt.legend(loc='upper right')  # 图例位置
plt.ylabel('real and predicted value')  # y轴标题
plt.show()  # 展示图像

# 模型应用
print ('regression prediction')
new_point_set = [[1.05393, 0., 8.14, 0., 0.538, 5.935, 29.3, 4.4986, 4., 307
                 [0.7842, 0., 8.14, 0., 0.538, 5.99, 81.7, 4.2579, 4., 307.,
                 [0.80271, 0., 8.14, 0., 0.538, 5.456, 36.6, 3.7965, 4., 307
                 [0.7258, 0., 8.14, 0., 0.538, 5.727, 69.5, 3.7965, 4., 307.
预测的新数据集
for i, new_point in enumerate(new_point_set):  # 循环读出每个要预测的数据点
    new_pre_y = model_gbr.predict(new_point)  # 使用GBR进行预测
    print ('predict for new point %d is:  %.2f' % (i + 1, new_pre_y))  # 打
印输出每个数据点的预测信息
```

上述代码以空行分为6个部分。

第一部分导入库。本示例中用到的库比较多：Numpy用于文件读写和基本数据处理；sklearn.linear_model中的三个广义线性回归下的回归算法BayesianRidge、Linear-Regre-ssion、ElasticNet，sklearn.svm中的svr回归算法，sklearn.ensemble.gradient_boosting中的GradientBoostingRegressor回归算法；cross_val_score用来做交叉检验；sklearn.metrics下的多个回归模型指标评估方法用来做模型评估；Pandas用来做格式化的数据框，便于数据结果输出；Matplotlib用来做预测结果的展示和对比。这里导入多种回归算法，目的是检验在没有先验经验的条件下，通过多个模型的训练，从中找到最佳拟合算法做后续应用。

第二部分数据准备。使用Numpy的loadtxt方法读取数据文件，并使用索引切片功能从数据集中分割出自变量X和因变量y。自变量X拥有506个样本，13个特征变量。

第三部分训练回归模型。本过程使用交叉检验的方法评估不同模型的训练效果。

先设置交叉检验的次数为6（6折交叉检验），后续会在交叉检验模型训练中用到；

接着分别建立贝叶斯岭回归（BayesianRidge）、普通线性回归（LinearRegression）、弹性网络回归（ElasticNet）、支持向量机回归（SVR）、梯度增强回归（GradientBoost-ingRegressor）模型对象，前3个算法属于广义线性回归，后两个属于支持向量机和梯度增强算法的变体；

然后分别建立不同模型的名称列表、回归模型对象的集合、交叉检验结果空列表、回归模型预测的y值空列表，分别用于后续名称读取、模型代入交叉检验算法、交叉检验结果数据存储和回归预测的结果存储。

最后通过for循环读出每个回归模型对象，对每个回归模型导入交叉检验模型cross_val_score中做训练检验并设置检验6次，并将交叉检验结果通过append方法存入结果列表，将回归训练中得到的预测y通过append方法存入列表。

相关知识点：cross_val_score

cross_val_score是sklearn.model_selection中的交叉检验工具，可以对特定算法模型进行交叉检验。常用参数：

```
sklearn.model_selection.cross_val_score(estimator, X, y=None, groups=None, s
```

estimator：要应用交叉检验的模型对象，该模型在交叉检验中会通过使用fit方法来训练模型

·X，y：交叉检验的数据集X和目标y，其中X至少是2维数据

·cv：要进行交叉检验的模式。如果cv值为空，那么默认使用3折交叉检验；如果cv值是整数，那么使用按照指定的数量做交叉检验。这两种情况下都会调用sklearn.model_selection.StratifiedKFold方法实现。如果cv值是用来交叉检验的生成器对象或可迭代的测试和训练集，那么将使用sklearn.model_selection.KFold方法。

·cross_val_score交叉检验后返回的得分默认调用算法模型的score方法做得分估计，因此不同的算法模型其得分计算方法可能有差异，具体取决于模型本身score方法（scorer（estimator，X，y））的计算逻辑。当然，也可以通过设置scoring参数字符串来手动指定得分计算方法，例如cross_val_score（estimator，X，y，scor-ing='r2'，cv=6）。常用scoring方法如表4-3所示。

表4-3　手动指定的scoring方法

Scoring 方法	应用算法	调 用 函 数	备　注
accuracy	分类	metrics.accuracy_score	
average_precision	分类	metrics.average_precision_score	
f1	分类	metrics.f1_score	二分类
f1_micro	分类	metrics.f1_score	微平均 F1 值
f1_macro	分类	metrics.f1_score	宏平均 F1 值
f1_weighted	分类	metrics.f1_score	加权平均 F1 值
f1_samples	分类	metrics.f1_score	通过多标签样本
neg_log_loss	分类	metrics.log_loss	需要算法对象有 predict_proba 方法支持
precision 等	分类	metrics.precision_score	
recall 等	分类	metrics.recall_score	
roc_auc	分类	metrics.roc_auc_score	
adjusted_rand_score	聚类	metrics.adjusted_rand_score	
neg_mean_absolute_error	回归	metrics.mean_absolute_error	
neg_mean_squared_error	回归	metrics.mean_squared_error	
neg_median_absolute_error	回归	metrics.median_absolute_error	
r2	回归	metrics.r2_score	

第四部分模型效果指标评估。

先通过自变量数据集的shape获得总样本量和总特征数；

·接着创建一个列表用于存储接下来用到的回归模型评估指标对象，explained_variance_score，mean_absolute_error，mean_squared_error，r2_score，这些对象在第一步导入库时已经导入，因此无需做其他处理便可直接使用；创建用于存储不同回归评估指标的列表。

·explained_variance_score：解释回归模型的方差得分，其值取值范围是[0，1]，越接近于1说明自变量越能解释因变量的方差变化，值越小则说明效果越差。

·mean_absolute_error：平均绝对误差（Mean Absolute Error，MAE），用于评估预测结果和真实数据集的接近程度的程度，其值越小说明拟合效果越好。

·mean_squared_error：均方差（Mean squared error，MSE），该指标计算的是拟合数据和原始数据对应样本点的误差的平方和的均值，其值越小说明拟合效果越好。

·r2_score：判定系数，其含义是也是解释回归模型的方差得分，其值取值范围是[0，1]，越接近于1说明自变量越能解释因变量的方差变化，值越小则说明效果越差。

然后通过两层for循环来分别对每个模型和每种回归模型评估方法进行计算。需要注意的是里面有一个用于存储每个内循环的临时结果列表，需要在内循环结束之后，每次将其结果存储到外部循环中的回归评估指标列表。

下面是通过Pandas的Dataframe方法分别建立交叉检验和回归指标数据框，然后打印输出样本量和特征数量、交叉检验和回归指标数据框，并通过打印横线来做逻辑输出切分；为了更好的理解指标的含义，打印输出缩写和全名标题对照表。在打印过程中使用的方法跟聚类中的类似，但是对于格式化的输出，这里使用了Pandas的数据框，这样如果数据量比较多的情况下，更容易控制。输出结果如下：

```
samples: 506       features: 13
----------------------------------------------------------------------
cross validation result:
                        0         1         2         3         4         5
```

```
BayesianRidge     0.662422  0.677079  0.549702  0.776896 -0.139738 -0.024448
LinearRegression  0.642240  0.611521  0.514471  0.785033 -0.143673 -0.015390
ElasticNet        0.582476  0.603773  0.365912  0.625645  0.437122  0.200454
SVR              -0.000799 -0.004447 -1.224386 -0.663773 -0.122252 -1.374062
GBR               0.747920  0.809555  0.767936  0.862972  0.375267  0.559758
----------------------------------------------------------------------------
regression metrics:
                          ev       mae        mse        r2
BayesianRidge     0.731143  3.319204  22.696772  0.731143
LinearRegression  0.740608  3.272945  21.897779  0.740608
ElasticNet        0.686094  3.592915  26.499828  0.686094
SVR               0.173548  5.447960  71.637552  0.151410
GBR               0.975126  1.151773   2.099835  0.975126
----------------------------------------------------------------------------
short name       full name
ev       explained_variance
mae       mean_absolute_error
mse       mean_squared_error
r2       r2
```

通过结果可以看出，增强梯度（GBR）回归是所有模型中拟合效果最好的，表现在能解释97%的方差变化，并且各个误差项的值都是最低的。另外，还有一个重要因素是，梯度增强算法在测试的6次中，其结果相对稳定性较高，这也说明了该算法在应对不同数据集的稳定效果。指标中r2跟ev的值是相同的，原因是r2作为判定系数，本身就是解释方差的比例，含义相同。

第五部分模型效果可视化。

该部分使用Matplotlib将原始数据y和其他5个回归模型预测到的y值通过趋势图进行对比。

先通过figure方法创建画布，然后画出原始数据y的分布，其中画图所需的x值域，使用自变量集X的形状得到一个自增数字列表。

接着画出其他5个回归预测结果展示所需要的颜色列表、样式列表，并通过for集合plot方法循环画出每个预测结果趋势线。

最后设置图形标题、图例位置（右上角）、y轴标题并展示图像，如图4-4所示。

不要忘记设置图像位置，否则即使在plot方法中设置了label值（图例标签值），也会由于缺少图例位置而无法显示图例。

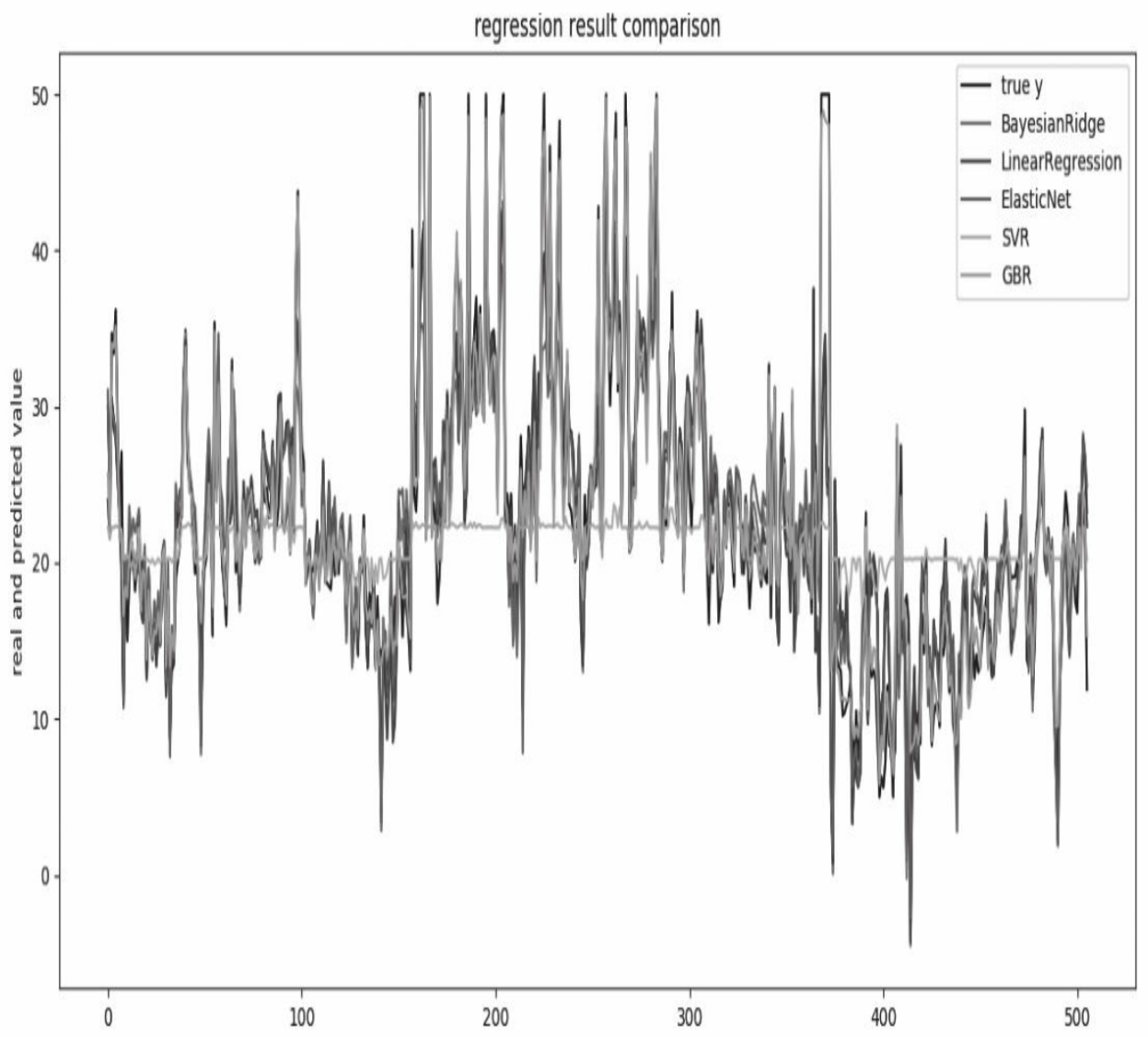


图4-4　不同回归模型预测结果对比

相关知识点：Matplotlib基本样式控制

Matplotlib中对于样式的基本控制如下：

·title：图形标题，通过字符串命名，用法为plt.title（'regression result comparison'）

·legend：设置图例，该方法有众多参数可用，经常使用的是其位置控制参数loc，通过该参数可控制图例显示的位置，值可以直接写定位字符串，也可以写定位代码。例如plt.legend（loc='upper　right'）等价于

plt.legend（loc=1）。常用location位置控制值如下：

表4-4　location位置控制值

定位字符串	定位代码	定位字符串	定位代码
'best'	0	'center left'	6
'upper right'	1	'center right'	7
'upper left'	2	'lower center'	8
'lower left'	3	'upper center'	9
'lower right'	4	'center'	10
'right'	5		

·xlabel：x轴标题，通过字符串命名，用法为plt.xlabel（'real and predicted value'）

·ylabel：y轴标题，通过字符串命名，用法为plt.ylabel（'real and predicted value'）

·xtickets：x轴刻度线标签，通过字符串与x轴刻度对应，用法为：plt.xticks（np.arange（5），（'Tom'，'Dick'，'Harry'，'Sally'，'Sue'））

·ytickets：y轴刻度线标签，通过字符串与y轴刻度对应，用法为：plt.yticks（np.arange（5），（'Tom'，'Dick'，'Harry'，'Sally'，'Sue'））

·xlim：设置x轴的范围，通过设置最大值最小值范围实现，用法为plt.xlim（（1，10））

·ylim：设置y轴的范围，通过设置最大值最小值范围实现，用法为plt.ylim（（1，10））

·text：增加文字到图形中，用法为plt.text（x，y，string），其中x

和y为要添加文字的坐标，string为文字字符串

·suptitle：图像居中主标题，通过字符串命名，用法为plt.suptitle（'classi-fication result'）

·axhline：在图形中增加一条水平线，用法为plt.axhline（y=0，xmin=0，xmax=1），其中y=0控制水平线的垂直位置，xmin和xmax控制水平线长度

·axvline：在图形中增加一条垂直线，用法为plt.axvline（x=0，ymin=0，ymax=1，其中x=0控制水平线的垂直位置，ymin和ymax控制水平线长度

样式控制小技巧：

·关闭坐标轴刻度（不显示刻度）：plt.xticks（[]）、plt.yticks（[]）

·关闭坐标轴：plt.axis（'off'）

·通过legend可批量为不同的图形对象增加图例，例如plt.legend（[s1，s2]，['label1'，'label2']，loc=0），其中s1和s2为不同的图形对象，例如线图、散点图等。

从图中看出，梯度增强算法（青色）的拟合程度跟原始数据（黑色）最接近，这也印证了在上面指标中输出的结果。

第六部分模型应用。基于新给出的包含多个数据点的数据集，我们要预测未来值，直接使用for循环读出每个数据，然后使用最优回归模型梯度增强的predict方法做预测，得到如下结果：

```
predict for new point 1 is:  21.49
predict for new point 2 is:  16.84
predict for new point 3 is:  19.50
predict for new point 4 is:  19.16
```

上述过程中，主要需要考虑的关键点是：如何在样本维度较多的情况下，选择最佳的回归模型，这里我们使用了多个模型，并结合交叉检验的方法找到最优的回归方法。

代码实操小结：本节的代码看似较多，但大多数方法之前已经介绍过，主要应用了以下几个知识点：

·通过Numpy的loadtxt读取数据文件。

·对矩阵做切片操作。

·使用sklearn.linear_model的BayesianRidge、LinearRegression、ElasticNet做广义线性回归分析。

·使用sklearn.svm中的svr、sklearn.ensemble.gradient_boosting中的GradientBoosting-Regressor做非线性回归分析。

·使用模型对象的fit方法训练模型，predict方法做预测，并可以将fit和predict一起使用（即model.fit（X_train）.predict（X_test））；通过列表将模型对象本身做集合以便于迭代循环处理。

·使用sklearn.metrics中的explained_variance_score，mean_absolute_error，mean_squared_error，r2_score做回归模型效果评估。

·对列表通过append追加元素。

·通过矩阵的shape方法获得矩阵形状。

·通过Pandas的DataFrame创建新的数据框，并指定列名、索引名。

·通过enumerate方法结合for循环，读出目标可迭代对象的索引和迭代值，enume-rate方法在需要使用对象索引值非常有效。

·使用Numpy的arange方法创建一个可迭代的连续区间。

·使用matplotlib.pyplot的plot方法展示图形，并设置不同的展示样式，包括颜色、样式、图例等，针对展示图像设置标题、图例位置、y轴标题等。

·使用print方法结合占位符输出特定字符串或变量，使用不同的占位符类型（d/f等）并指定宽度（使用\t做tab分隔）做打印数据格式化。

4.3 分类分析

分类算法通过对已知类别训练集的计算和分析，从中发现类别规则并预测新数据的类别。分类算法是解决分类问题的方法，是数据挖掘、机器学习和模式识别中一个重要的研究领域。分类和回归是解决实际运营问题中非常重要的两种分析和挖掘方法。

常用的分类算法包括朴素贝叶斯、逻辑回归、决策树、随机森林、支持向量机等。

分类的主要用途和场景是“预测”，基于已有的样本预测新样本的所属类别。例如信用评级、风险等级、欺诈预测等；同时，它也是模式识别的重要组成部分，广泛应用到机器翻译，人脸识别、医学诊断、手写字符识别、指纹识别的图像识别、语音识别、视频识别的领域；另外，分类算法也可以用于知识抽取，通过模型找到潜在的规律，帮助业务得到可执行的规则。

4.3.1 防止分类模型的过拟合问题

过拟合通俗点讲就是在做分类训练时模型由于过度学习了训练集的特征，使得训练集的准确率非常高，但将模型应用到新的数据集时准确率却很差。因此，避免过拟合是分类模型中的一个重要任务。分类算法尤其是决策树容易出现过拟合的问题，可通过以下途径避免过度拟合。

·使用更多的数据：导致过拟合的根本原因是训练集和新数据集的特征存在较大差异，导致原本完美拟合的模型无法对新数据集产生良好效果；通过增加数据集，可能会增加训练集和新数据集的特征相似度，这样会使得分类结果在新数据集上表现更好。

·降维：通过维度选择或转换的方式，降低参与分类模型的维度数量，能有效防止原有数据集中的“噪音”对模型的影响，从而达到避免过拟合的目的。

·使用正则化方法：正则化会通过定义不同特征的参数来保证每个特征有一定的效用，不会使某一特征特别重要。例如SVM有个L2正则项参数，可以在目标函数中对模型进行限制，通过最优化过程中尽量追求小的L2值就会提高泛化能力，也就抑制了过拟合的问题。

·使用组合方法：例如，随机森林、adaboost不容易产生过拟合的问题。

4.3.2 使用关联算法做分类分析

在关联算法中，我们会得到不同的关联规则，或者称为频繁规则，这种模式一般用来做关联模式挖掘。但如果转换思路，其实也可以用来做分类分析。

以商品关联规则为例，我们先拿到一张用户订单记录表，表的内容里面包括用户的属性和特征以及用户购买的商品，得到的关联规则结果类似于：面包、牛奶→冰激凌，这个规则中的面包和牛奶属于前项，冰激凌属于后项，我们会认为买了面包和牛奶的客户，还会一并购买冰激凌。

在这个示例中，我们应用的是商品和商品之间的关联；如果前后项指定的不是一个维度的数据，那么就可以实现分类分析：在前项中可以使用属性类特征，例如用户性别、年龄段、学历等，后项仍然是用户购买的商品，那么得到的结果类似于：女、20～30，大学→冰激凌，以及这个规则的支持度（例如12%）、置信度（例如85%）、提升度（例如1.2）。置信度是后项/前项的值，即买了冰激凌的用户中有多少是满足属性值为女、20～30，大学这些条件；换个表达方式是在属性值为女、20～30，大学的人群中，有85%的人购买了冰激凌。这个规则不正类似于决策树得到的if sex='女'and age='20-30'and education='大学'then冰激凌=True吗？

使用关联逻辑并结合支持度、置信度和提升度可以找到其分类信息，实现思路：

从数据集中选择属性特征+目标（可能是商品、内容、广告等），使用关联规则得到频繁项集，频繁项集的前、后项分别是属性（可以是单个或多个属性）和目标。

从得到的频繁项集中，根据支持度、置信度、提升度定义出具有显著性的规则，例如支持度大于45%、置信度大于75%。提升度大于1。

基于提取出来的前后项规则梳理出可用的分类规则以及对应的目标信息，该目标可以是多分类的。

有关关联分析的更多话题，会在4.4节中谈到。

4.3.3 用分类分析来提炼规则、提取变量、处理缺失值

1.分类分析用于提炼应用规则

预测是分类分析的主要应用方向，但将分类用于提炼应用规则，为数据化运营提供规则支持也是其重点应用之一，这种应用相对于其他算法更加具有落地价值。常见的应用场景包括：

·要针对沉默会员做会员重新激活，应该挑选具有什么特征的会员？

·商品A库存积压严重，现在要通过促销活动清仓，选择哪些类型的促销活动更容易实现该目标？

·网站需要大流量广告位来满足VIP商家的精准广告投放，具有哪些特征的广告位更符合VIP商家的客户需求？

从分类算法中提炼特征规则，利用的是在构建算法过程中的分类规则。以决策树为例，决策树的分裂节点是表示局部最优值的显著特征值，每个节点下的特征变量以及对应的值的组合构成了规则。图4-5是利用Sklearn构建的决策树规则的一部分。其中X[0]～X[3]是决策树模型的特征变量，而类似于X[2]<=21891.5则是该特征变量对应显著性分裂值。

2.分类分析用于提取变量特征

从大量的输入变量中获得变量的重要性特征，然后提取权重较高的几个特征是分类分析的重点应用之一。这一应用是数据归约和数据降维的重要方式。在3.3.2节提到了选择特征的方法之一是基于机器学习算法来选择，而分类算法可以作为典型应用方法。具体实现思路是：获取原始数据集并对数据做预处理，将预处理的数据集放到分类算法中进行训练，然后从算法模型中提取特征权重信息。有关具体的程序部分，会在本章4.3.6节中演示。

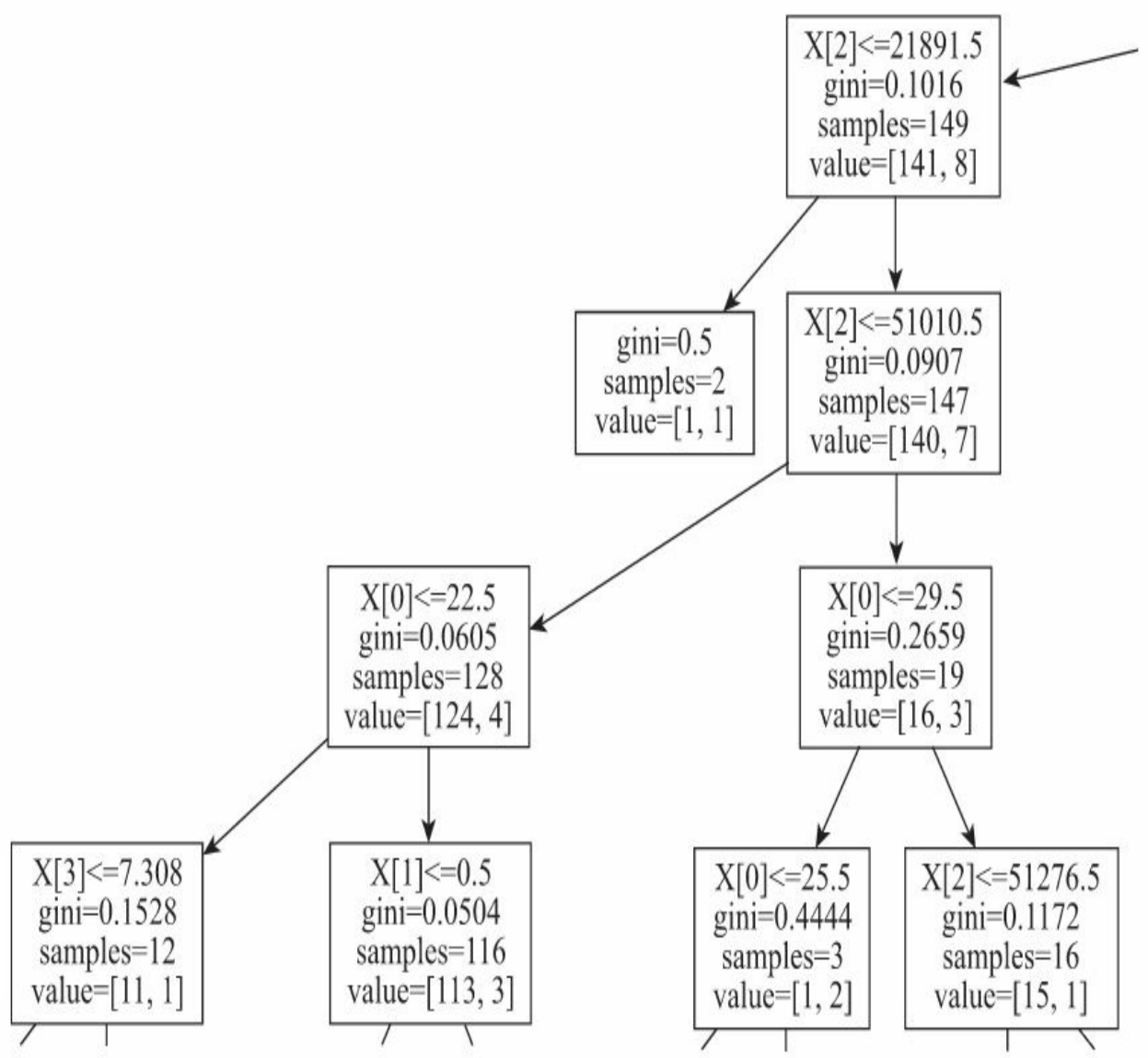


图4-5　决策树规则

3.分类分析用于处理缺失值

在3.1.1节中提到了补全缺失值的方法之一是使用模型法，基于已有的其他字段，将缺失字段作为目标变量进行预测，从而得到较为可能的补全值。如果缺失值是分类变量，则采用分类模型补全。有关分类预测的具体代码部分，也会在4.3.6节中演示。

4.3.4 类别划分-分类算法和聚类算法都是好手

对于大多数初学者而言，聚类和分类两类算法似乎不容易分清楚。聚类和分类的区别：

·学习方式不同：聚类是一种非监督式学习算法，而分类是监督式学习算法。所谓的监督与非监督指的是是否有准确性的指示属性，用来告诉算法模型某种行为是对的还是错的，如果有就是监督式，如果没有就是非监督式。

·对源数据集要求不同：聚类不要求源数据集有预先定义的标签，但分类需要标签作为监督学习的“标准”或“参照”用来训练模型。

·应用场景不同：聚类一般应用于做数据探索性分析、数据降维、数据压缩等探索性、过程性分析和处理，而分类更多的用于预测性分析和使用。

·解读结果不同：聚类算法的结果是将不同的数据集按照各自的典型特征分成不同类别，不同人对聚类的结果解读可能不同；而分类的结果却是一个固定值（例如高、中、低、是、否等），不存在不同解读的情况。

·模型评估指标不同：聚类分析没有所谓的“准确”与否，以及如何准确的相关度量，更多的是基于距离的度量，如果是对带有标签的数据集做聚类则可以做相似度、完整度等方面的评估；而分类模型的指标例如准确率、混淆矩阵、提升率等都有明显的好与坏、提升程度等评估指标。

但是，在实际应用中要实现“划分类别”这一目标，无论二者的差异性有多大，聚类和分类确实都能实现。在4.1.6节的示例中，使用K均值方法得到的聚类结果已经显示了即使在没有标签的情况下做非监督式学习，也能获得良好的划分类别的效果。假如原始数据集带有类别标签，那么选择分类或聚类算法都可以（标签列数据并不是一定要使用）。

假如原始数据集不带有类别标签，那么只能选择使用聚类算法。

K均值这种聚类算法的准确率高的前提能正确划分群体类别，K值的确定不仅是一个数据技巧，而且也需要业务经验来辅助判断，因为即使在数据上做到类间距离最大、类内距离最小，得出的不同类别之间的数据是否具有显著特征，类别间的样本量是否属于一个量级且有利于从业务性的角度进行分析都不是纯数据能“证明”的事情。

有关分类和聚类的应用示例，假如现在公司要对某新会员做促销活动，例如推荐商品、提供个性化信息、推荐最感兴趣的热榜等，并尽量提供该用户感兴趣的内容。如何实现这一需求？

分类：基于现有的会员及其特定类别标签（可选择有代表性或跟实际运营场景最相关的类别标签）做分类模型训练，将该新用户的数据作为新的样本输入模型预测得到该用户所属的目标类别，接着计算该类别下用户最经常购买的商品、经常浏览的信息等并给出推荐内容。

聚类：将新的会员和现有的会员作为一个整体做聚类分析，然后获得该会员所属的聚类类别，进而提取其所在类别下其他会员的经常购买商品、经常浏览信息等并给出推荐内容。

当然，我们还有更多方法给出推荐内容，例如基于协同过滤的方法（用户相似度、查看内容的相似度）、基于关联算法的方法、基于冷启动的方法等。在此不一一进行列举。

4.3.5 如何选择分类分析算法

分类算法作为非常实用且有效的数据化运营支持方法，几乎在所有公司中都能派上用场，如何根据不同场景选择最合适的算法？

·文本分类时，用到的最多的是朴素贝叶斯，例如电子邮件的垃圾邮件识别。

·如果训练集比较小，那么选择高偏差且低方差的分类算法则效果更好，例如朴素贝叶斯、支持向量机，因为这类算法不容易过拟合。

·如果训练集比较大，那么不管选择哪种方法，都不会显著影响分类准确度。

·如果关注的是算法模型的计算时间和模型易用性，那么选择支持向量机、人工神经网络不是好的选择。

·如果重视算法的准确率，那么应选择算法精度较高的方法，例如支持向量机、随机森林。

·如果想得到有关预测结果的概率信息，然后基于预测概率做进一步应用，使用逻辑回归是比较好的选择。

·如果担心离群点或数据不可分，并且需要清晰的决策规则，那么选择决策树。

4.3.6 代码实操：Python分类分析

Sklearn中没有一个专门的分类算法库，分类算法分散在不同的方法库中，例如ensemble、svm、tree等，在使用时需要分别导入不同的库来使用其中的分类算法。

示例模拟的是针对一批带有标签的数据集做分类模型训练，然后使用该模型对新数据集做分类预测；主要使用Sklearn做分类、用Matplotlib做图形展示，数据源文件classi-fication.csv位于“附件-chapter4”中，默认工作目录为“附件-chapter4”（如果不是，请cd切换到该目录下，否则会报“IOError:File classification.csv does not exist”）。

另外，本节会用到两个新的图形和表格展示库：prettytable和pydotplus，以及配合pydotplus的GraphViz程序。

prettytable是用来做表格格式化输出展示的，它的好处是可以非常容易的对行、列进行控制，并且输出带有分割线的可视化table。第一次使用该库需要先通过系统终端命令行窗口（或PyCharm中底部的Terminal窗口）使用pip install prettytable在线安装，安装成功后在Python命令行窗口输入import prettytable，无报错信息则该库已经正确安装。

pydotplus是在决策树规则输出时用到的库，其输出的dot数据可以供GraphViz绘图使用。要能完整使用该库需要先安装GraphViz程序，然后再安装pydotplus。

第一步　安装GraphViz程序。这是一个额外的应用程序，而不是一个Python附属包或程序。读者可登录http://www.graphviz.org/Download.php下载，第一次登录该网站时需要阅读一堆内容须知，阅读完之后可直接点击底部的Agree，然后到达下载程序窗口，在该窗口中按照不同的操作环境选择下载或安装方式。笔者的电脑操作系统环境是Windows，选择的是“graphviz-2.38.msi”。下载完成之后的安装过程没有任何难点。

第二步　安装pydotplus。这是一个从dot数据读取数据格式并保存为可视化图形的库，在系统中打开系统终端命令行窗口（或PyCharm中底部的Terminal窗口）输入pip install pydotplus，几秒钟之内就能完成自

动下载安装过程。

完整代码如下：

```
# 导入库
import numpy as np  # 导入numpy库
from sklearn.model_selection import train_test_split  # 数据分区库
from sklearn import tree  # 导入决策树库
from sklearn.metrics import accuracy_score, auc, confusion_matrix, f1_score,
入指标库
import prettytable  # 导入表格库
import pydotplus  # 导入dot插件库
import matplotlib.pyplot as plt  # 导入图形展示库

# 数据准备
raw_data = np.loadtxt('classification.csv', delimiter=',', skiprows=1, )  #
取数据文件
X = raw_data[:, :-1]  # 分割X
y = raw_data[:, -1]  # 分割y
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, rand
数据分为训练集和测试集

# 训练分类模型
model_tree = tree.DecisionTreeClassifier(random_state=0)  # 建立决策树模型对象
model_tree.fit(X_train, y_train)  # 训练决策树模型
pre_y = model_tree.predict(X_test)  # 使用测试集做模型效果检验

# 输出模型概况
n_samples, n_features = X.shape  # 总样本量,总特征数
print ('samples: %d \t features: %d' % (n_samples, n_features))  # 打印输出样
本量和特征数量
print (70 * '-')  # 打印分隔线

# 混淆矩阵
confusion_m = confusion_matrix(y_test, pre_y)  # 获得混淆矩阵
confusion_matrix_table = prettytable.PrettyTable()  # 创建表格实例
confusion_matrix_table.add_row(confusion_m[0, :])  # 增加第一行数据
confusion_matrix_table.add_row(confusion_m[1, :])  # 增加第二行数据
print ('confusion matrix')
print (confusion_matrix_table)  # 打印输出混淆矩阵

# 核心评估指标
y_score = model_tree.predict_proba(X_test)  # 获得决策树的预测概率
fpr, tpr, thresholds = roc_curve(y_test, y_score[:, 1])  # ROC
auc_s = auc(fpr, tpr)  # AUC
accuracy_s = accuracy_score(y_test, pre_y)  # 准确率
precision_s = precision_score(y_test, pre_y)  # 精确度
recall_s = recall_score(y_test, pre_y)  # 召回率
f1_s = f1_score(y_test, pre_y)  # F1得分
core_metrics = prettytable.PrettyTable()  # 创建表格实例
core_metrics.field_names = ['auc', 'accuracy', 'precision', 'recall', 'f1']
义表格列名
core_metrics.add_row([auc_s, accuracy_s, precision_s, recall_s, f1_s])  # 增
加数据
print ('core metrics')
print (core_metrics)  # 打印输出核心评估指标
# 模型效果可视化
names_list = ['age', 'gender', 'income', 'rfm_score']  # 分类模型维度列表
```

```
color_list = ['r', 'c', 'b', 'g']  # 颜色列表
plt.figure()  # 创建画布
# 子网格1：ROC曲线
plt.subplot(1, 2, 1)  # 第一个子网格
plt.plot(fpr, tpr, label='ROC')  # 画出ROC曲线
plt.plot([0,          1],          [0,          1],          linestyle='--
', color='k', label='random chance')  # 画出随机状态下的准确率线
plt.title('ROC')  # 子网格标题
plt.xlabel('false positive rate')  # X轴标题
plt.ylabel('true positive rate')  # y轴标题
plt.legend(loc=0)
# 子网格2：指标重要性
feature_importance = model_tree.feature_importances_  # 获得指标重要性
plt.subplot(1, 2, 2)  # 第二个子网格
plt.bar(np.arange(feature_importance.shape[0]), feature_importance, tick_lab
出条形图
plt.title('feature importance')  # 子网格标题
plt.xlabel('features')  # x轴标题
plt.ylabel('importance')  # y轴标题
plt.suptitle('classification result')  # 图形总标题
plt.show()  # 展示图形

# 保存决策树规则图为PDF文件
dot_data = tree.export_graphviz(model_tree, out_file=None, max_depth=5, feat
决策树规则生成dot对象
graph = pydotplus.graph_from_dot_data(dot_data)  # 通过pydotplus将决策树规则解
析为图形
graph.write_pdf("tree.pdf")  # 将决策树规则保存为PDF文件

# 模型应用
X_new = [[40, 0, 55616, 0], [17, 0, 55568, 0], [55, 1, 55932, 1]]
print ('classification prediction')
for i, data in enumerate(X_new):
    y_pre_new = model_tree.predict(data)
    print ('classification for %d record is: %d' % (i + 1, y_pre_new))
```

上述代码以空行分为9个部分。

第一部分导入库。本示例中使用了Sklearn的tree库和metrics库，分别用来做分类预测分类指标评估、model_selection库做数据分区，使用Numpy辅助于数据读取和处理，使用prettytable库做展示表格的格式化输出，使用pydotplus来生成决策树规则树形图，使用Matplotlib的pyplot库做图形展示。

第二部分数据准备。使用Numpy的loadtxt方法读取数据文件，指定分隔符以及跳过第一行标题名；然后使用矩阵索引将数据分割为X和y，最后使用sklearn.model_selection的train_test_split方法将数据分割为训练集和测试集，训练集数量占总样本量的70%。

相关知识点：将数据集划分为训练集、测试集和验证集

第二部分中将数据集划分为训练集和测试集两部分，在很多场景中需要将数据集分为训练集、测试集和验证集三部分。Sklearn没有提供直接将数据集分为3种（含3种）以上的方法，我们可以使用Numpy的split方法划分数据集。split参数如下：

```
split(ary, indices_or_sections, axis=0)
```

·ary：要划分的原始数据集。

·indices_or_sections：要划分的数据集数量或自定义索引分区。如果直接使用整数型数值设置分区数量，则按照设置的值做等比例划分；如果设置一个一维的数组，那么将按照设置的数组的索引值做区分划分边界。

·axis：要划分数据集的坐标轴，默认为0。

数据集分割示例：将创建的新数据集通过平均等分和指定分割索引值的方式分为3份。

```
import numpy as np # 导入库
x = np.arange(72).reshape((24,3)) # 创建一个24行3列的新数组
train_set1, test_sets1, val_sets1 = np.split(x, 3) # 将数组平均分为3份
train_set2, test_sets2, val_sets2 = np.split(x, [int(0.6*x.shape[0]), int(0.
训练集,30%测试集,10%验证集
print ('record of each set - equal arrays: ')
print    ('train_set1:    %d,    test_sets1:    %d,    val_sets1:    %d'%
(train_set1.shape[0], test_sets1.shape[0], val_sets1.shape[0]))
print (40*'-')
print ('record of each set - % arrays: ')
print    ('train_set2:    %d,    test_sets2:    %d,    val_sets2:    %d'%
(train_set2.shape[0], test_sets2.shape[0], val_sets2.shape[0]))
```

上述代码执行后，返回如下结果：

```
record of each set - equal arrays:
train_set1: 8, test_sets1: 8, val_sets1: 8
----------------------------------------
record of each set - % arrays:
train_set2: 14, test_sets2: 7, val_sets2: 3
```

第三部分训练分类模型。使用sklearn.tree中的DecisionTreeClassifier

方法建立分类模型并训练，然后基于测试集做数据验证。DecisionTreeClassifier为CART（分类回归树），除了可用于分类，还可以用于回归分析。

相关知识点：tree算法对象中的决策树规则

在决策树算法对象的tree_属性中，存储了所有有关决策树规则的信息（示例中的决策树规则存储在model_tree.tree_中）。最主要的几个属性：

·children_left：子级左侧分类节点。

·children_right：子级右侧分类节点。

·feature：子节点上用来做分裂的特征。

·threshold：子节点上对应特征的分裂阈值。

·values：子节点中包含正例和负例的样本数量。

上述属性配合节点ID、节点层级迭代便能得到如下的规则信息：

```
1 [label="rfm_score <= 7.8375\ngini = 0.1135\nsamples = 14581\nvalue = [1370
```

其中规则开始的1代表节点ID，rfm_score是变量名称，rfm_score<=7.8375是分裂阈值，gini=0.1135是在当前规则下的基尼指数，nsamples是当前节点下的总样本量，nvalue为正例和负例的样本数量。

第四部分输出模型概况。由于分类算法评估内容较多，因此从这里开始将分模块输出内容以便于区分。本部分内容中，通过X的形状获得数据的样本量和特征数量，打印输出结果如下：

```
samples: 21927       features: 4
----------------------------------------------------------------------
```

第五部分输出混淆矩阵。使用sklearn.metrics中的confusion_matrix方

法，通过将测试集的训练结果与真实结果的比较得到混淆矩阵。接下来通过prettytable展示混淆矩阵并输出表格，该库会自动对表格进行样式排版，并通过多种方法指定列表、样式等输出样式：先建立PrettyTable方法表格对象，然后使用add_row方法追加两行数据，打印输出结果如下：

```
confusion matrix
+---------+---------+
| Field 1 | Field 2 |
+---------+---------+
|   5615  |   284   |
|   321   |   359   |
+---------+---------+
```

相关知识点：混淆矩阵（confusion matrix）

混淆矩阵是做分类算法效果评估的基本方法，它是监督式学习中的一种可视化工具，主要用于比较分类结果和实例的真实信息。矩阵中的四个区域，分别代表了TP、FP、FN、TN四个值。如图4-6所示。

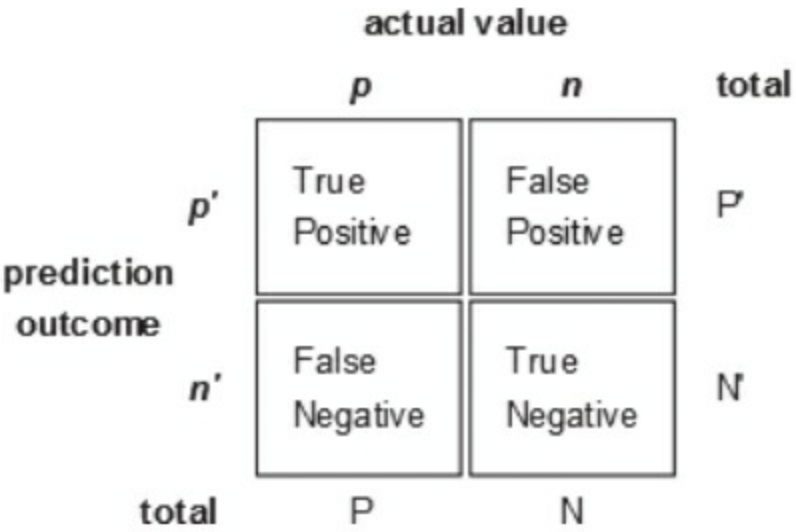


图4-6　混淆矩阵

表示分类正确：

·真正（True Positive，TP）：本来是正例，分类成正例。

·假正（True Negative，TN）：本来是负例，分类成负例。

表示分类错误：

·假负（False Positive，FP）：本来是负例，分类成正例。

·真负（False Negative，FN）：本来是正例，分类成负例。

第六部分输出分类模型核心评估指标。

先通过决策树模型对象的predict_proba方法获得决策树对每个样本点的预测概率，该数据在下面的ROC中用到；输出的概率信息可作为基于阈值调整分类结果输出的关键，例如可自定义阈值来做进一步精细化分类类别控制。

接着通过sklearn.metrics的roc_curve、auc、accuracy_score、precision_score、recall_score、f1_score分别得到AUC、准确率、精确度、召回率、F1得分值。

·auc_s：AUC（Area Under Curve），ROC曲线下的面积。ROC曲线一般位于y=x上方，因此AUC的取值范围一般在0.5和1之间。AUC越大，分类效果越好。

·accuracy_s：准确率（Accuracy），分类模型的预测结果中将正例预测为正例、将负例预测为负例的比例，公式为：A=（TP+TN）/（TP+FN+FP+TN），取值范围[0，1]，值越大说明分类结果越准确。

·precision_s：精确度（Precision），分类模型的预测结果中将正例预测为正例的比例，公式为：P=TP/（TP+FP），取值范围[0，1]，值越大说明分类结果越准确。

·recall_s：召回率（Recall），分类模型的预测结果被正确预测为正例占总的正例的比例，公式为：R=TP/（TP+FN），取值范围[0，1]，值越大说明分类结果越准确。

·f1_s：F1得分（F-score），准确度和召回率的调和均值，公式为：F1=2*（P*R）/（P+R），取值范围[0，1]，值越大说明分类结果越准

确。

上述指标计算完成后仍然通过prettytable的PrettyTable方法创建表格对象，然后使用field_names定义表格的列名，通过add_row方法追加数据，打印输出结果如下：

```
core metrics
+----------------+---------------+--------------+---------------+------
|      auc       |    accuracy    |   precision    |     recall      |
+----------------+---------------+--------------+---------------+------
| 0.749870117567 | 0.908040735674 | 0.55832037325 | 0.527941176471 |0.542705
+--------------+--------------+-------------+--------------+------
```

从上述指标可以看出整个模型效果一般，一方面，在建立模型时，我们没有对决策树剪枝，这会导致决策树的过拟合问题；另一方面，原始数据中，存在明显的样本类别不均衡问题，也没有做任何预处理工作。由于这里仅做算法流程演示，关于调优的部分不展开讲解。

第七部分模型效果可视化，目标是输出变量的重要性以及ROC曲线。建立分类模型维度列表和颜色列表，用于图形展示；然后通过figure方法创建画布。

子网格1：ROC曲线。使用subplot方法定义第一个子网格，其中“1，2，1”表示1行2列的第一个子网格，使用plot方法分别画出模型训练得到的ROC曲线和随机状态下的准确率线，使用title、xlabel、ylabel分别设置子网格标题、X轴和Y轴标题。在使用legend方法设置图例时，使用loc=0来让图表选择最佳位置放置图例。

子网格2：指标重要性。该子网格的位置是1行2列的第二个区域，其设置与子网格1基本相同，差异点在于这里使用了bar方法创建条形图。上述代码返回如4-7所示的图形结果。

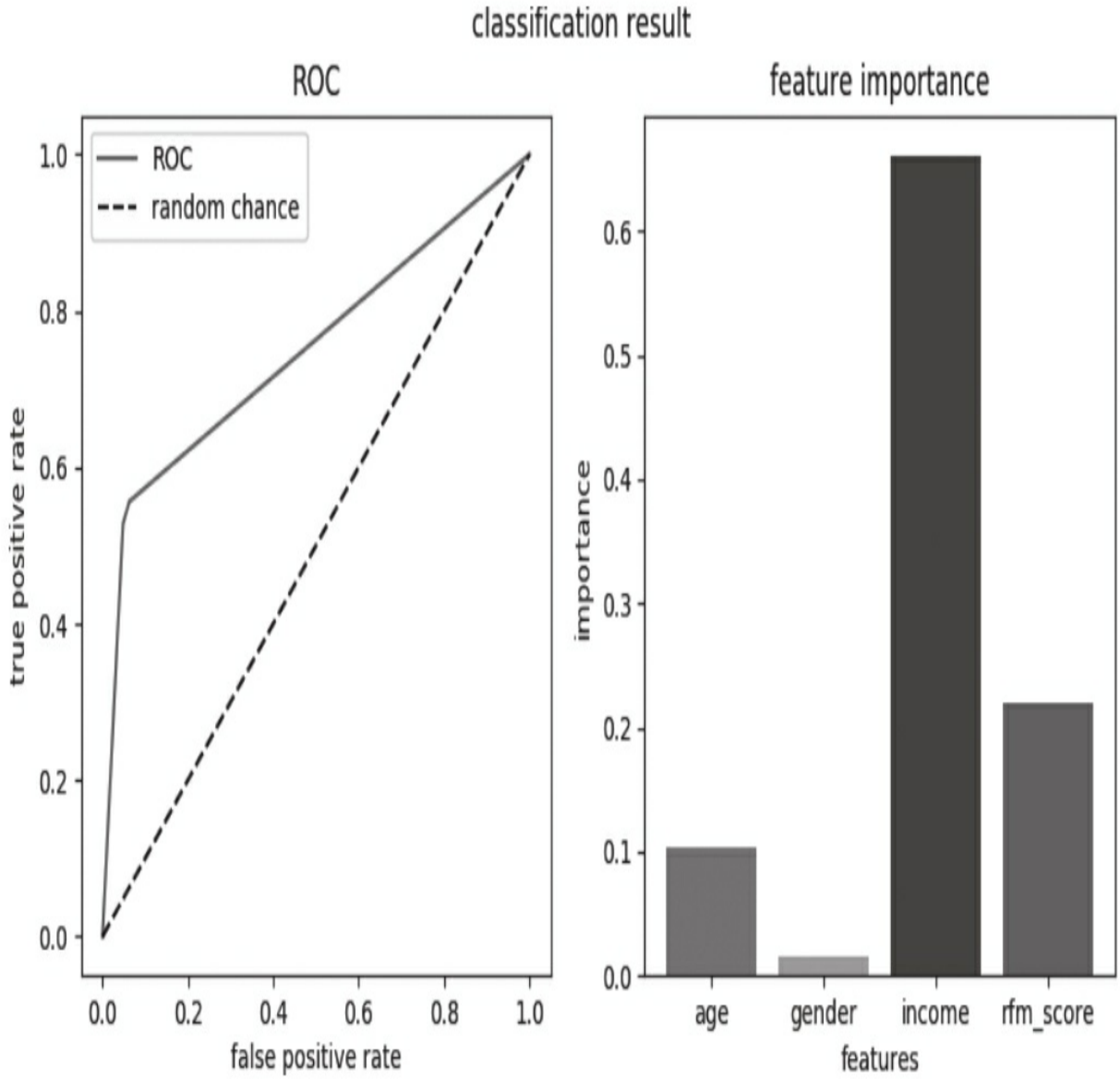


图4-7　分类结果展示

从上述结果可以看出，ROC曲线的面积大于0.5，模型的算法结果比随机抽取的准确率要高，但综合前面我们得到的准确率指标，结果也不是特别理想。在特征重要性中，income变量具有非常高的特征权重，是这几个变量中最重要的变量，其次是rfm_score和age。

第八部分保存决策树规则图为PDF文件。先通过tree库下的export_graphviz方法，将决策树规则生成dot对象，各参数作用如下：

·out_file=None：控制不生成dot文件，否则对象dot_data会为空；

·max_depth：控制导出分类规则的最大深度，防止规则过多导致信息碎片化；

·feature_names：指定决策树规则的每个变量的名称，方便在决策树规则中识别特征名称；

·filled：控制填充，让图形效果更佳美化；

·rounded：控制字体样式。

上述代码执行后，会在python工作目录产生一个新的名为“tree.pdf”的文件，打开PDF文件，部分内容如图4-8所示。

在sklearn的tree库中，有一个特殊的方法tree.export_graphviz，可用来讲树形规则结构转化为DOT格式的数据对象，该方法只存在于tree库中。在dot_data变量的代码中，可去掉out_file=None参数，默认会生成一个名为tree.dot的数据文件，该文件就是上述树形图输出的源数据。

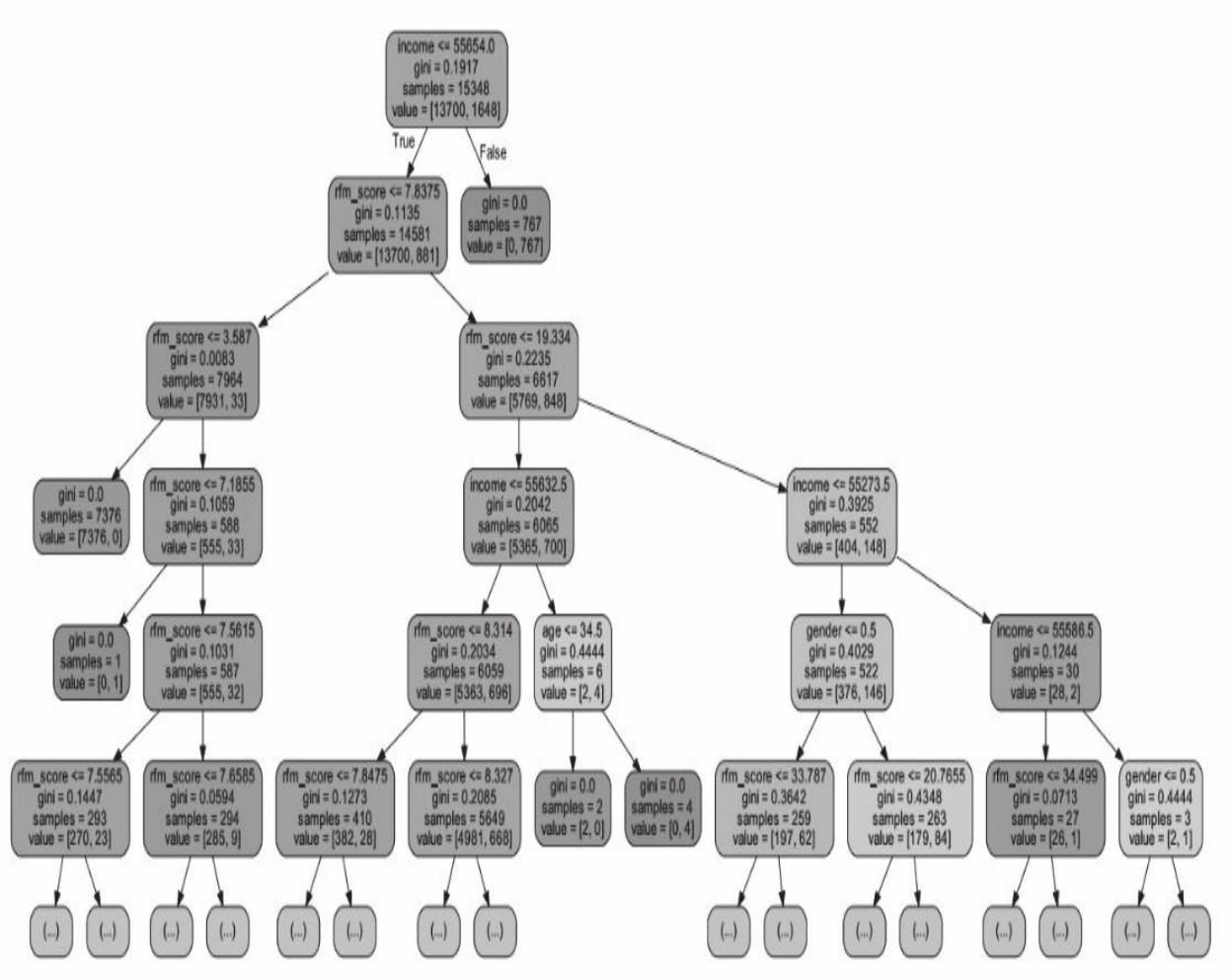


图4-8　决策树规则输出树形图

上述决策树规则显示了当income<=55654时，总样本量为15348，其中负例样本和正例样本分别为13700、1648。当income<=55654且rfm_score<=7.8375时，总样本量有13581，其中负例样本和正例样本分别为13700、881。以此类推可以读出其他的规则。

第九部分模型应用。通过将新的数据集放入模型做分类预测，得到每个数据集的预测类别指标，结果如下：

```
classification prediction
classification for 1 record is: 0
classification for 2 record is: 0
classification for 3 record is: 1
```

上述过程中，主要需要考虑的关键点：

·如何根据不同的应用需求和限制选择最适合的分类算法。

·如何优化算法来提高准确率并降低模型的过拟合程度。

代码实操小结：在本节的代码中，主要用了以下几个知识点：

·通过Numpy的loadtxt读取数据文件。

·对矩阵做切片操作。

·使用sklearn.tree中的DecisionTreeClassifier方法实现分类分析。

·使用sklearn.metrics中的confusion_matrix方法输出混淆矩阵。

·通过prettytable打印输出的表格，自动会对表格进行样式排版，并通过多种方法指定列表、样式等输出样式。

·通过sklearn.metrics的roc_curve、auc、accuracy_score、precision_score、recall_score、f1_score分别得到AUC、准确率、精确度、召回率、F1得分值。sklearn的metrics库中还提供了更多的分类评估指标，感兴趣的读者可自行查阅更多信息。

·通过matplotlib.pyplot展示多网格的图形，使用plot方法展示线图、bar方法展示条形图。

·使用pydotplus配合GraphViz绘制决策树规则树并保存为文件。

·使用模型对象的fit方法训练模型，predict方法做预测，并可以将fit和predict一起使用；通过列表将模型对象本身做集合以便于迭代循环处理。

·通过矩阵的shape方法获得矩阵形状。

·通过enumerate方法结合for循环，读出目标可迭代对象的索引和迭代值，enume-rate方法在需要使用对象索引时非常有效。

·使用Numpy的arange方法创建一个可迭代的连续区间。

·使用matplotlib.pyplot的plot方法展示图形，并设置不同的展示样式，包括颜色、样式、图例等，针对展示图像设置标题、图例位置、y轴标题等。

·使用print方法结合占位符输出特定字符串或变量，使用不同的占位符类型（d/f等）并指定宽度（使用\t做tab分隔）做打印数据格式化。

4.4 关联分析

关联分析通过寻找最能够解释数据变量之间关系的规则，来找出大量多元数据集中有用的关联规则，它是从大量数据中发现多种数据之间关系的一种方法；另外，它也可以基于时间序列对多种数据间的关系进行挖掘。关联分析的典型案例是“啤酒和尿布”的捆绑销售，即买了尿布的用户还会同时买啤酒。

常用的关联算法包括Apriori、FP-Growth、PrefixSpan、SPADE、AprioriAll、Apriori-Some等。

关联规则相对其他数据挖掘模型更加简单，易于业务理解和应用。关联模型的典型应用场景是购物篮分析等，通过分析用户同时购买了哪些商品来分析用户购物习惯。这种策略还会应用于捆绑销售、库存管理、商品促销设计、页面促销设计、货架设计、商品陈列设计、页面内容排版、推荐系统、商品价格策略和基于购买的用户特征分析等。网站分析工具Webtrekk中的关联分析报表便应用了关联规则算法。

4.4.1 频繁规则不一定是有效规则

所谓频繁规则指的是关联结果中支持度和置信度都比较高的规则，而有效规则指的是关联规则真正能促进规则中的前/后项的提升。在做关联结果分析时，频繁规则往往会被“想当然”地认为是有效规则，但实际情况却并非总是如此。

假如，数据集有1000条事务数据用来显示购买苹果和香蕉的订单记录，其中有600个客户的订单记录中包含了苹果，有800个客户的订单记录中包含了香蕉，而有400个客户同时购买了苹果和香蕉。假如我们产生了一条关联规则，用来表示购买了苹果的客户中还有很多人购买香蕉，那么该规则可以表示为：苹果→香蕉。

·支持度：support=400/1000=40%

·置信度：confidence=400/600=67%

如果只是看支持度和置信度，这个规则似乎非常显著的说明了苹果和香蕉之间的频繁关系，买了苹果的客户中有67%的人也会一起购买香蕉。但是，如果忽略购买苹果的事实，只购买香蕉的客户比例会高达是80%（800/1000）！这显示了购买苹果这种条件不会对购买香蕉产生积极的促进作用，反而会阻碍其销售，苹果和香蕉之间是一种负相关的关系。因此，只看支持度和置信度将无法完整体现规则的有效性，通过另一个指标提升度则可以有效应对该问题。

提升度（Lift）指的是应用关联规则和不应用产生结果的比例。在本示例中，Lift=（400）/（800）=0.5（有关联规则的前提下只有400个客户会购买香蕉，没有关联规则的前提下会有800个购买香蕉）。当提升度为1时，说明应用关联规则和不应用关联规则产生相同的结果；当提升度大于1时，说明应用关联规则比不应用关联规则能产生更好的结果；当提升度小于1时，关联规则具有负相关的作用，该规则是无效规则。

在做关联规则评估时，需要综合考虑支持度、置信度和提升度三个指标，支持度和置信度的值越大越好。但需要注意的是，在低频、客单价较高的领域，关联规则会呈现稀疏性，其支持度百分比可能只有几个

百分点。对于提升度的评估，通常关注其值大于1的规则。

4.4.2 不要被啤酒尿布的故事紧固你的思维

关联规则的产生来源于“啤酒和尿布”的故事，这是商品销售间的关联分析。除了这种分析模式外，还能将关联分析应用到更多运营分析场景。

1.相同维度下的关联分析

相同维度下的关联分析指的是关联分析的前后项是相同逻辑的内容维度。例如商品——商品，内容——内容。该模式下常见的应用包括：

（1）网站页面浏览关联分析

网站页面浏览关联分析可以帮助我们找到用户在不同页面（包含广告页、活动页、超市页、单品页、帮助页等）之间的频繁访问关系，以分析用户特定的页面浏览模式。这种频繁模式可用于了解不同页面之间的分流和引流关系，尤其是大型落地页的分析；也可以用来做不同页面间的页面浏览推荐，利于提高用户网站体验和转化率。

（2）广告流量关联分析

广告流量关联分析是针对站外广告投放渠道用户浏览或点击的行为分析，该分析主要用于了解用户的浏览和点击广告的模式，例如点击了A广告之后又点击了B广告；浏览了C广告之后又浏览了D广告。这种站外广告曝光和点击的关联分析可以为广告客户的精准投放提供参考。除了用于站外广告分析，还可以用于网站引流的分析，假如公司通过一组整合传播媒体做活动宣传，那么我们可以分析用户通过宣传渠道到达网站的先后关系和频繁模式，可以从中发觉类似于用户从M广告到达网站之后，还会从N媒体点击到达网站，这对于广告媒体的投放组合和整合营销评估具有重要意义。

（3）用户关键字搜索关联分析

分析用户在站内的搜索关键字是发现用户搜索兴趣并了解用户真实需求的方法之一。通过对用户搜索关键字的关联分析，可以得到类似于搜索了苹果之后又搜索了iPhone，搜索了三星之后又搜索了HTC，这种

关联模型可用于搜索推荐、搜索联想等场景，有助于改善搜索体验，提高客户的目标转化率。

2.跨维度的关联分析

除了可以做相同维度的关联分析，也可以基于不同维度做关联性分析。在4.3.2节中提到了一种基于属性特征+目标的形式做分类应用。除此之外，还可以有以下用法：

（1）不同场景下的关联分析

不同场景下的关联分析指的是发生的事件处于不同的时间下，但通常都在一个约束时间范围内（例如session、会话）。这种模式可以广泛用于分析运营中关注的要素，例如用户浏览商品与购买商品的关联分析、关注产品价格与购买商品价格的关联分析、用户加入购物车与提交订单的关联分析等。通过这种跨事件的关联分析，可以找到用户不同行为模式之间的关系，尤其可以发掘用户的真实需求和关注（潜在）需求之间的关联和差异性，这些信息可用于针对当前用户行为的个性化推荐，并对后续促销活动的价格策略的制定非常有参考价值。

（2）相同场景下的事件关联

相同场景下的事件关联指发生的事件在一个场景下，但属于不同的时间点。例如用户在同一个页面中点击不同功能、选择不同的应用、下载不同的白皮书等。这类信息可以帮助我们了解用户对于功能应用的先后顺序，有利于做产品优化和用户体验提升；对于不同产品功能组合、开发和升级有了更加明确的参考方向，便于针对用户习惯性操作模式做功能迭代；同时针对用户频繁查看和点击的内容，可以采用打包、组合、轮转等策略，帮助客户尽量缩小内容查找空间和时间，也能提升内容曝光度和用户体验度。

4.4.3 被忽略的“负相关”模式真的毫无用武之地吗

在应用关联分析时，我们大多会预先设置支持度和置信度阈值，以缩短算法计算时间并能有效提供具有显著性的强规则，然后通过提升度配合支持度和置信度共同提取有效规则。这种做法在大多数场景下并没有问题。

但是，很多支持度和置信度高而提升度低的强规则也有可利用的空间。当一个规则的支持度和置信度高时，说明这个规则是频繁的；但如果发现规则的提升度低，那么说明规则中的前后项是“互斥”的，即规则中的前后项通常不会一起发生。4.4.1节中的苹果和香蕉的例子就说明了这个问题。

这种规则说明了我们不能把规则中的前后项通过组合、打包或关联的策略展示给客户，它可以作为组合打包的控制条件来优化组合策略。典型应用场景包括：

·在商品销售策略中，不将具有互斥性的商品放到同一个组合购买计划中。

·在站外广告媒体的投放中，不将具有互斥性的多个广告媒体做整合传播或媒体投放。

·在关键字提示信息中，不将具有互斥性的关键字提示给客户。

·在页面推荐的信息流中，不将具有互斥性的信息流展示给用户。

4.4.4 频繁规则只能打包组合应用吗

常见的关联规则基于两种模式产生：基于同一个时间内发生的事件以及基于不同时间下发生的事件。

·基于同一个时间内发生的事件：这种模式发生在同一时间点，例如购买篮分析就是用于在一次购物篮中同时购买的商品，这种场景下使用订单ID做事务型数据的主键。

·基于不同时间下发生的事件：这种模式发生在不同的时间点，但是可以通过特定的主键信息关联，例如用户在不同日期购买了多件商品，这种场景下使用用户ID作为事务型数据的主键。

当通过上述规则分析得到关联结果后，第一反应就是应该把这些商品（或其他项目）放到一起做打包组合应用。例如，将用户常买的商品打包为一个促销方案，或者将用户经常一起浏览的内容作为专栏等。

但除了这种打包组合的思维方式之外，还可以这样考虑应用：既然用户具有较强的发生关联事件关系（例如购买、查看等）的可能性，那么可以基于用户的这种习惯，将前后项内容故意分离开，利用用户主动查找的时机来产生更多价值或完成特定转化目标。

例如：用户经常一起购买啤酒和尿布，我们可以分别将啤酒和尿布陈列在展柜的两端（或者隔开一段距离），然后在用户购买啤酒又去购买尿布的途中，也许会发现别的商品进而产生兴趣，从而实现更多的销售。

在用户完成两件事情的期间，不仅能促进商品销售，还能用于展示更多的广告和促销活动、提供更多的个性化内容等。这种方式可以将用户的更多兴趣激发出来。

但是，这种模式不是在所有关联规则下都能生效，需要具备一定的条件：

·关联规则必须是强规则且有效规则。规则不频繁或无效则说明用户并不具有这种频繁习惯去完成两件事情，更何况需要花费更多的时间

和精力。因此，刚需内容是最好的，其次是强规则且有效规则。

·发生关联的前后项之间需要有非常强的完成动机。强动机是促进用户完成两件事情的主要原因，例如，在做大型促销活动时，需要用户完成多个步骤才能获得抄底价商品的购买资格就属于这类应用。

·不能过多降低用户体验。用户体验在这个过程中是辅助于用户完成事件的重要因素，尤其对于关注用户体验的客户群来说，该因素更加重要。如果在用户完成不同事件间设置的环节或周期过长，会使得用户体验度的极度下降，这样会直接导致用户的放弃。因此，能在保证甚至提升客户一定的体验度的前期下，设计完成环节非常重要。

4.4.5 关联规则的序列模式

基于不同时间下发生的频繁规则，可以使用关联规则中的序列模型来分析。

序列模式相较于普通关联模式最大的区别是不同的事件之间具有明显的时间区隔，以及先后的序列发生关系，能得到类似于“完成某个事件之后会在特定的时间周期内完成其他事件”的结论，例如购买了冰箱的客户会在3个月内购买洗衣机的结论。这是一种预测性分析的模式，能够将事件发生的时间和对象提取出来，因此更加适合对基于时间下的数据化运营的应用需求。

常见的运营应用场景：

·客户购买行为预测：基于用户上次的购买时间和商品信息，推断用户下次购物的时间和订单商品。

·Web访问模式预测：基于用户上次浏览页面的时间和页面信息，推断用户下次最可能浏览的页面。

·流量来源预测：基于用户上次网站到达时间和到访信息，推断用户下次最可能从哪些媒体渠道进入网站。

·关键字搜索预测：基于用户上次搜索关键字的时间和关键字，推断用户下次最可能搜索哪些关键字。

能实现序列模式的关联算法包括：

·AprioriAll：基于哈希树的序列关联算法，它与Apriori算法的执行过程是一样的，不同点在于候选集的产生，需要区分最后两个元素的前后顺序。

·AprioriSome：该算法是对AprioriAll算法的改进。

·CARMA（Continuous Association Rule Mining Algorithm）：CARMA是一种比较新的关联规则算法，能够处理在线连续交易流数

据。

·GSP（Generalized Sequential Patterns）：基于水平存储结构和哈希树遍历操作的序列关联算法，它具有类似于Apriori算法的实现步骤，主要区别在于产生候选序列模式。

·SPADE（Sequential PAttern Discovery using Equivalence classes）：基于垂直存储结构和格理论连接操作的序列关联算法，它是一种改进的GSP算法。

·FreeSpan：频繁模式投影的序列关联算法，利用频繁项递归地将序列数据库投影到更小的投影数据库集中，在每个投影数据库中生成子序列片断，是一种分治思想的算法。

·PrefixSpan（Prefix-Projected Pattern Growth）：基于前缀树的序列关联算法，从Free-Span中推导演化而来。

4.4.6 代码实操：Python关联分析

对于Python，目前的流行数据科学计算库中尚没有一个广泛应用的关联算法库，因此本节我们无法直接使用流行库来做关联分析（在关联分析这个问题上，Python确实比R要逊色）。

Orange有关联算法应用，但它需要先下载和安装orange程序，再通过python库做调用，还不是完全的Python第三方扩算法库。

不过，得益于Python的开放性，我们可以在本地自定义一个算法包，然后直接导入Python程序中做关联分析。事实上，在很多运营分析中，我们都需要通过导入本地第三方包来解决特定问题，例如：

·现有Python流行库中没有我们需要的算法，本示例就是如此。

·企业内部根据实际运营需求开发了很多Python程序，这些程序需要集成到Python工作环境中。

·企业内部的数据工作环境中，有预定义好的类和库，直接引用即可实现特定功能，无需重复“造轮子”。

·现有的Python流行库中的算法存在某些问题和缺陷，在企业内部已经经过一定的优化和提升，而这些优化只能在企业内部应用。

示例模拟的是针对一批事务型的订单交易数据集做关联分析，以得到不同商品之间的关联规则；在本节示例代码的附件中，有一个名为apriori.py的关联算法包，该程序的基础版本来源于网络，并经过二次开发和修正以满足特定需求，它将作为本示例关键算法的主要工具包；数据源文件association.txt位于“附件-chapter4”中，默认工作目录为“附件-chapter4”（如果不是，请cd切换到该目录下，否则会报“IOError:File association.txt does not exist”）。

另外，对于关联分析的关系结果展示，我们会用到在4.3节中的GraphViz，同时需要一个封装的Python API第三方包——GraphViz，通过该包调用GraphViz自定义画图。读者可以在系统终端的命令行窗口通

过pip install graphviz安装。以下是完整代码：

```
import sys
sys.path.append('../chapter4')
import pandas as pd
from graphviz import Digraph
import apriori

# 定义数据文件
fileName = 'association.txt'

# 通过调用自定义的apriori做关联分析
minS = 0.1  # 定义最小支持度阈值
minC = 0.38  # 定义最小置信度阈值
dataSet = apriori.createData(fileName)  # 获取格式化的数据集
L, suppData = apriori.apriori(dataSet, minSupport=minS)  # 计算得到满足最小支持度的规则
rules = apriori.generateRules(fileName, L, suppData, minConf=minC)  # 计算满足最小置信度的规则

# 关联结果报表评估
model_summary = 'data record: {1} \nassociation rules count: {0}'  # 展示数据集记录数和满足阈值定义的规则数量
print (model_summary.format(len(rules), len(dataSet)))  # 使用str.format做格式化输出
df                =                pd.DataFrame(rules,                columns=['item1', 'itme2', 'instance', 'support', 'confidence', 'lift'])  # 创建频繁规则数据框
df_lift = df[df['lift'] > 1.0]  # 只选择提升度>1的规则
print (df_lift.sort('instance', ascending=False))  # 打印排序后的数据框

# 关联结果图形展示
dot = Digraph()  # 创建有向图
graph_data = df_lift[['item1', 'itme2', 'instance']]  # 切分画图用的前项、后项和实例数数据
for each_data in graph_data.values:  # 循环读出每条规则
    node1, node2, weight = each_data  # 分割每条数据画图用的前项、后项和实例数
    node1 = str(node1)  # 转化为字符串
    node2 = str(node2)  # 转化为字符串
    label = '%s' % weight  # 创建一个标签用于展示实例数
    dot.node(node1, node1, shape='record')  # 增加节点（规则中的前项）
    dot.edge(node1, node2, label=label, constraint='true')  # 增加有向边
dot.render('apriori', view=True)  # 保存规则为pdf文件
```

上述代码以空行分为5部分。

第一部分导入库。本示例中用到了sys、Pandas、GraphViz和apriori。

·sys：用来为Python增加自定义库的工作路径，使用sys.path.append方法将chapter4目录（自定义库apriori所在的目录）将自定义库目录追加到程序中。

·Pandas：用来将关联规则的结果通过数据框表格展示出来。

·GraphViz：用来创建一个关系图展示不同项目的关联关系。

·apriori：为自定义的关联算法，在通过sys.path.append操作之后可直接跟普通第三方库一样使用import方法导入。需要注意的是导入自定义的本地库的代码位置必须在sys.path.append方法之后，否则程序会无法找到自定义包。

相关知识：按照约定俗成的位置顺序导入不同类型的库

Python的工作库有三种类型，分别是Python内置标准库、第三方库和自定义库。

·内置标准库是Python安装之后自带的库和包，例如re、string、datetime等，读者可在https://docs.python.org/2/library/index.html查看所有2.*（本书程序所用的版本）版本的内置库。

·第三方库是除了Python程序之外的，第三方主体开发的流行库，例如Sklearn、Numpy、Pandas等，一般是直接通过网络相关资源下载到本地安装或者使用pip命令在线安装。

·自定义库是本地的相关程序，一般是自定义的功能模块。

Python中导入不同类型的库，约定俗成的位置顺序是：

·Python标准库。

·Python第三方库。

·自定义库。

例如：

```
import re # Python标准库
import numpy as np # Python第三方库
import apriori # 自定义库
```

如果这三类库比较多，通常在不同类型的库之间以空行隔开。这样做的好处是不同的类型库之间的关系清晰，易于管理和维护，尤其是大型程序导入的应用库多且需要多人协同运维时非常必要。本书中由于示例代码一般都比较短，并且需要以空行做功能讲解，因此没有通过空行的形式区分不同类型的库。

第二部分定义数据文件。数据文件的格式是以逗号分隔的订单合并的事务型记录，每条记录是一个订单，每个订单中的多个项目在同一行。

第三部分通过调用自定义的apriori做关联分析。

先定义了最小支持度和置信度阈值，除了可以用来筛选特定条件的规则外，还能有效提高程序效率。对于两个阈值的定义，如果在时间允许的条件下，笔者一般都先设置较低的阈值，后期再做进一步筛选。

接着通过apriori库的createData方法获得数据，返回的是列表类型的数据，每个列表内嵌套商品项目列表。

然后使用apriori库的apriori方法计算得到满足最小支持度的规则，返回的是所有满足条件的频繁项集的列表和所有候选项集的支持度数据列表。

最后使用apriori库的generateRules方法计算满足最小置信度的规则，返回同时满足最小支持度和最小置信度的频繁规则。

第四部分关联结果报表评估。先定义一个结果概览字符串，并使用str.format对字符串用占位符做格式化输出。打印输出结果如下：

```
data record: 100
association rules count: 13
```

原始数据集有100条数据记录，得到的符合条件的频繁规则有13条。

相关知识点：使用str.format对字符串做格式化处理

在做格式化输出时，我们之前用到了print直接输出、Pandas的数据框、使用print配合%占位符等方法。该部分示例用到一种新的方法：str.format。与%占位符类似，str.format使用{}做占位符。这是一个针对字符串格式化输出的利器，相比较之前的几种方法，str.format使用灵活、方便，能通过多种途径获取要输出的变量内容。

str.format占位符所指示的格式化语法为：

```
[[fill][align][sign][#][0][width][,][.precision][type]
```

其中：

·[fill]：任何要填充到占位符的字符串。

·[align]：对齐方式，通过<、>、=、^可设置为左对齐、右对齐、强制将变量值放到符号之后数字之前以及强制居中对齐。

·[sign]：符号选项，仅对数字类型有效，通过+表示一个符号应该用于正数和负数；-表示一个符号只能用于负号；空格表示在正数时应使用前导空格，负号使用负号。

·[＃]：选项仅对整数有效，仅适用于二进制，八进制或十六进制输出。如果存在，它指定输出将分别以0b，0o或0x为前缀。

·[width]：定义最小字段宽度的十进制整数。如果未指定，则字段宽度将由内容确定。

·[，]：表示使用逗号作为千分位分隔符。

·[.precision]：用来控制精度，这是一个十进制数，表示在使用f和F格式化的浮点值的小数点后面应显示多少个数字。

·[type]：表示以何种形式展示数据，常用的类型如表4-5所示。

表4-5　常用type类型

type 值	描　述	type 值	描　述
s	字符串型	x	十六进制
b	二进制	e	指数心事
c	unicode 字符串	f	精度为 6 位的浮点数
d	十进制整数	%	百分数
o	八进制		

格式化数据示例

示例一：通过位置映射获得变量信息输出数据：

```
print ('number:{1},count:{0}'.format(10,20)) # 输入
```

其中"number:{1}，count:{0}"为原始字符串，{1}代表format变量中的第2个变量值，{0}代表format变量中的第1个变量值。字符串的占位符与format参数不要求前后位置一一对应，只要索引位置对应即可。上述代码执行后输出如下结果：

```
number:20,count:10 # 输出
```

示例二：通过列表/元组/字符串的位置映射形式输出一组数据：

```
str_key = 'I am: {0[1]}\nYou are: {0[2]}\nHe is: {0[0]}'
str_value = ('Lucy','Tony','Lilei')
print (str_key.format(str_value))
```

其中{0[1]}代表输出的是元组的第二个值，以此类推{0[2]}和{0[0]}分别代表元组的第三个和第一个值。format的参数变量可以是元组也可以是列表，只要是能迭代读出的序列就可以。除了列表和元素，还可以使用分割后的字符串（如果觉得字符串本身也有意义，也可以直接使用

字符串），字符串位置跟fomat参数的位置无关，只跟位置索引有关。例如：

```
str_key = 'I am: {0[1]}\nYou are: {0[2]}\nHe is：{0[0]}'
str_value = 'Lucy,Tony,Lilei'
print (str_key.format(str_value.split(',')))
```

这段代码返回的结果与上述代码结果一致，结果如下：

```
I am: Tony
You are: Lilei
He is：Lucy
```

示例三：通过列表/元组/字符串的形式多次输出一组数据：

```
str_key = 'I am: {0[1]}\nYou are: {0[1]}\nHe is：{0[0]}'
str_value = 'Lucy,Tony'
print (str_key.format(str_value.split(',')))
```

上述代码中，在占位符中可多次调用format参数中的索引。字符串可以多次映射到format参数的位置，与其参数的数量无关。返回结果如下：

```
I am: Tony
You are: Tony
He is：Lucy
```

示例四：通过关键字参数获得变量信息：

```
str_value = {'age':30, 'name':'tony'}
str_key='I am: {name}, I am {age} years old'
print (str_key.format(**str_value))
```

上述代码中，先定义一个字典，字典的key为关键字参数，然后在format参数中，通过**+字典名称的方式批量传入参数信息。返回结果如下：

```
I am: tony, I am 30 years old
```

示例五：通过表达式结合数字传递变量信息：

```
print ('rate is: {:.2%}'.format(10./21))
```

上述代码中，在format参数中传入一个表达式来传值。占位符表示的是保留2位小数的百分数。结果如下：

```
rate is: 47.62%
```

示例六：样式居中对齐：

```
print ('{:*^40}'.format('This is a title'))
```

上述代码中，强制字符串居中对齐，总长度为40位，不足的部分以*补齐。返回结果如下：

```
************This is a title*************
```

接着创建一个频繁规则数据框，用来存在返回的规则数据，包括前项、后项、实例数、支持度、置信度、提升度，然后只选择数据框中lift>1的数据记录，最后使用sort方法对数据按照实例数排序输出。结果如下：

```
             item1          item2  instance  support  confidence    lift
0   (17288980167)  (17092020299)        14     0.14      0.4828  1.3410
1   (17092020299)  (17288980167)        14     0.14      0.3889  1.3410
3   (38657641492)  (17092020299)        14     0.14      0.4118  1.1438
4   (17092020299)  (38657641492)        14     0.14      0.3889  1.1438
7   (38728350298)  (38660935334)        14     0.14      0.5000  1.3158
8   (13943415375)  (38660935334)        14     0.14      0.4516  1.1885
9   (38722052311)  (38660935334)        14     0.14      0.3889  1.0234
2   (38657641492)  (38660935334)        13     0.13      0.3824  1.0062
5   (36989298167)  (38660935334)        13     0.13      0.4483  1.1797
6   (13943415375)  (17092020299)        13     0.13      0.4194  1.1649
12  (38657641492)  (38722052311)        13     0.13      0.3824  1.0621
10      (4002591)  (38728350298)        11     0.11      0.4231  1.5110
11  (38728350298)      (4002591)        11     0.11      0.3929  1.5110
```

第五部分关联结果图形展示。本部分调用GraphViz实现画关系图。在第一部分的导入库操作中，我们直接导入了GraphViz的有向图库digraph，除此之外还有一个Graph的无向图库。

所谓有向和无向指的是不同节点之间是否具有方向性。有向图库digraph可以表示有顺序、流转等逻辑的图形，例如流程图、指向图等；无向图库Graph则只表示不同节点间的相互关系，而没有方向指向性，例如热力图等。

首先通过digraph方法创建一个有向图对象用来画图。然后切分画图用的前项、后项和实例数数据。通过for循环读出每条规则数据并做画图处理：

·分割每条数据画图用的前项、后项和实例数；

·将前后项转化为字符串类型；

·创建一个标签用于展示实例数，通过占位符的形式动态赋值；

·使用node方法为有向图对象添加节点，其中的三个参数分别代表节点名称、节点标签和节点的图形样式；

·为该节点增加一条有向线条，用来表示该节点与哪个其他节点产生关系，其中的四个参数分别为线条开始的节点、线条终止的节点、线条标签（展示每个有向线条所代表的规则的实例数），以及是否根据图形自动调整位置。

最后使用render方法将画完的图保存为apriori的文件，系统默认会保留为PDF格式的文件。通过view=true来控制保存完成之后自动打开文件。得到如图4-9所示的规则。

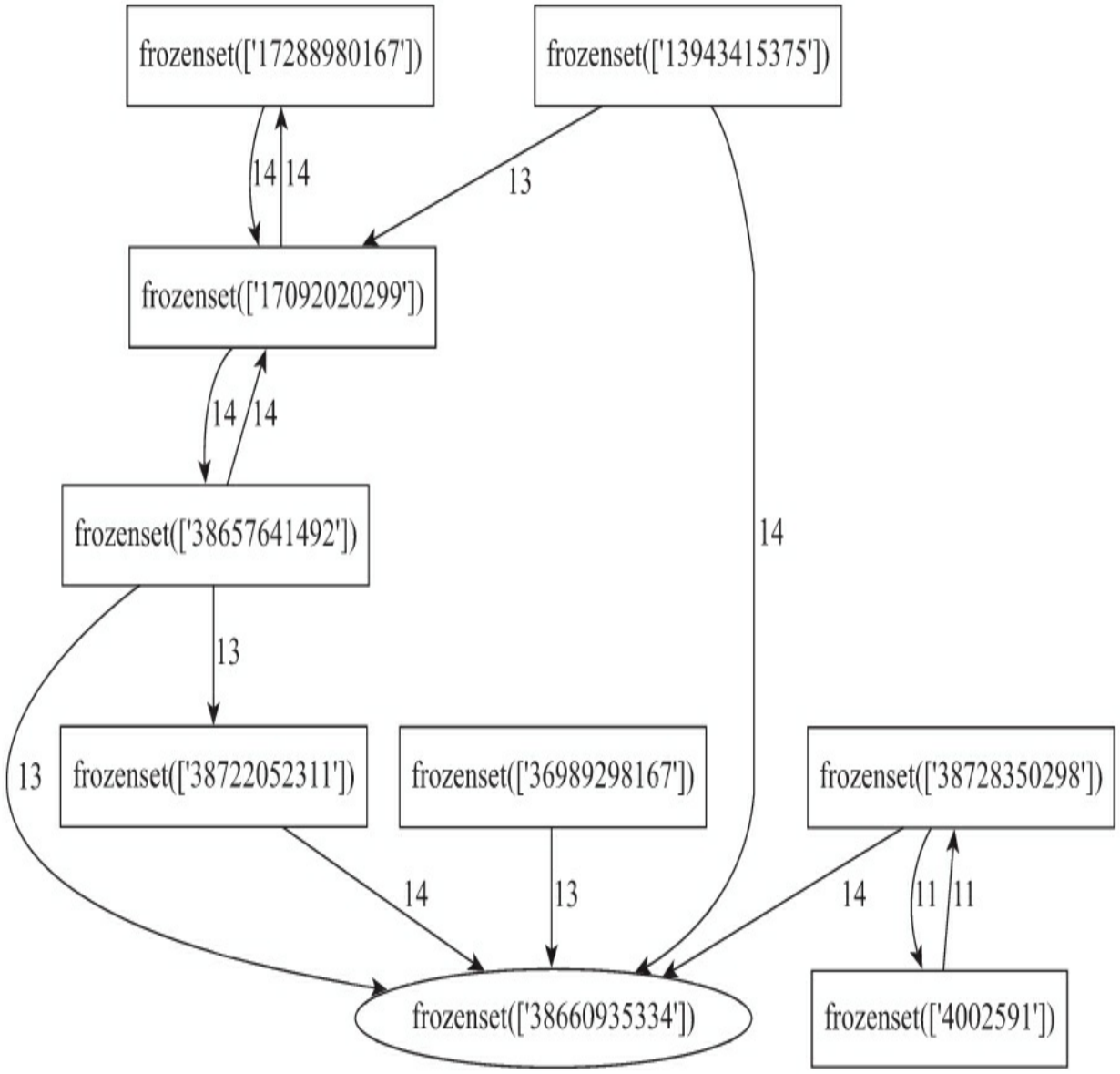


图4-9　关联结果关系图

上述过程中，关联算法本身由本地导入，因此不涉及算法的应用，本示例新增了一些知识：

·如何定义本地库路径并导入自定义库；

·如何通过GraphViz画出关系图。

代码实操小结：在本节的代码中，主要用了以下几个知识点：

·通过sys.path.append方法将本地自定义库的目录追加到程序中并导入自定义库apriori；

·通过调用自定义库的功能实现关联分析并得到关联规则结果集；

·使用str.format方法做输出格式化操作；

·通过Pandas的dataframe创建数据库并指定列名；

·对数据库进行切片分割，并按照指定列值做数据记录筛选；

·使用sort方法对数据框进行排序；

·使用GraphViz的digraph画有向图，并分别定义了节点、连接线并设置了自定义信息；

·将通过GraphViz画出的图形保存为本地PDF格式的文件。

4.5 异常检测分析

数据集中的异常数据通常被认为是异常点、离群点或孤立点，特点是这些数据的特征与大多数数据不一致，呈现出“异常”的特点，检测这些数据的方法称为异常检测。在大多数数据分析和挖掘工作中，异常值都会被当作“噪音”剔除。但在某些情况下，如果数据工作的目标就是围绕异常值展开，那么异常值会成为数据工作的焦点。

“噪音”的出现有多种原因，例如业务操作的影响（典型案例：网站广告费用增加10倍，导致流量激增）、数据采集问题（典型案例：数据缺失、不全、溢出、格式匹配等问题）、数据同步问题（异构数据库同步过程中的丢失、连接错误等导致的数据异常），在对离群点进行挖掘分析之前，需要从中区分出真正的“离群数据”，将“垃圾数据”去掉。

常用的异常检测方法分为基于统计的异常检测方法（如基于泊松分布、正态分布等分布规律找到异常分布点）、基于距离的异常检测方法（如基于K均值找到离所有分类最远的点）、基于密度的离群检测方法（LOF就是用于识别基于密度的局部异常值的算法）、基于偏移的异常点检测方法、基于时间序列的异常点监测方法等。

异常值检测常用于异常订单识别、风险客户预警、黄牛识别、贷款风险识别、欺诈检测、技术入侵等针对个体的分析场景。

4.5.1 异常检测中的“新奇检测”模式

异常检测根据原始数据集的不同可分为离群点检测和新奇检测两类模式。

1.离群点检测（Outlier Detection）

大多数场景下我们定义的异常数据都属于离群点数据，离群点检测的训练数据集包含“离群点”数据，对这些数据训练完成之后再在新的数据集中寻找异常数据。

2.新奇检测（Novelty Detection）

所谓新奇检测是识别新的或未知数据模式和规律的检测方法，这些规律和知识在已有机器学习系统的训练集中没有被发掘出来。新奇检测的前提是已知训练数据集是“纯净”的，未被真正的“噪音”数据或真实的“离群点”污染，然后针对这些数据训练完成之后再对新的数据做训练以寻找新奇数据的模式。

新奇检测主要应用于新的模式、主题、趋势的探索和识别，包括信号处理、计算机视觉、模式识别、智能机器人等技术方向，应用领域例如潜在疾病的探索、新物种的发现、新传播主题的获取等。

新奇检测和异常检测有关，一开始的新奇点往往都以一种离群的方式出现在数据中，这种离群方式一般会被认为是离群点，因此二者的检测和识别模式非常类似。但是，当经过一段时间之后，新奇数据一旦被证实为正常模式，例如将新的疾病识别为一种普通疾病，那么新奇模式将被合并到正常模式之中，就不再属于异常点的范畴。

4.5.2 将数据异常与业务异常相分离

数据异常反映的只是数据层面的离群分布，而这些离群分布未必都是业务意义上的“异常”，很多时候数据的异常是对业务特殊运营状态的反映，因此属于数据异常但业务正常的范畴。有关这部分内容已经在3.1.2节中介绍过，在此不再赘述。

4.5.3 面临维度灾难时，异常检测可能会失效

当原始数据集的维度非常多时，普通的异常检测方法可能会失效，原因是随着维度的增加，数据之间的相似程度将严重受到维度数量的影响，因此这种过程将很难奏效。

解决高维空间下的异常检测问题，通常有三种思路：

·扩展现有的离群点检测模式。

·发现子空间中的离群点。

·对高维数据进行建模。

在异常检测面对高维数据集时，跟聚类遇到的问题非常类似，有关该话题的更多信息请查看4.1.4节。

4.5.4 异常检测的结果能说明异常吗

在做异常检测分析时，输出的结果是用户是否异常的标签，例如1、-1，这种标签只是客观上基于数据相似度或密度的识别结果。但是，即使在业务也没有任何特殊动作（即由异常业务引发的真实数据反应）导致的“假异常”的前提下，仍然无法判断结果是否真的异常。

在销售型公司中，黄牛常用来代表哪些通过一定的渠道和方法大量倒卖公司的商品，并从中获利的人。对黄牛的检测和识别属于销售公司的主要分析项目之一。通过多种算法做出的异常检测结果会给出属于黄牛的信息列表，但列表中的异常检测结果并一定就符合实际情况，还需要业务介入做进一步筛查和审核。

因此，在大多数场景下通过非监督式方法实现的异常检测的结果只是用来缩小排查范围，为业务的执行提供更加精准和高效的执行目标而已。

要实现异常检测不只有非监督式一种方法，如果有历史对于黄牛等异常信息的标记，还可以通过监督式（分类算法）进行预测性检测分析。

4.5.5 代码实操：Python异常检测分析

sklearn中提供了one-class SVM和EllipticEnvelope两种方法用于异常检测，前者基于libsvm实现的非监督式异常检测方法，可用于做高维度分布的评估；后者只能做基于高斯分布数据集的异常检测。

本节示例模拟的是针对一批没有任何标签的原始数据做异常检测模型训练，然后通过新的测试集来发现新数据集中的异常数据；本示例会使用svm来做异常检测分析，数据源文件outlier.txt位于“附件-chapter4”中，默认工作目录为“附件-chapter4”（如果不是，请cd切换到该目录下，否则会报“IOError:File outlier.txt does not exist”）。完整代码如下：

```
# 导入库
from sklearn.svm import OneClassSVM  # 导入OneClassSVM
import numpy as np  # 导入numpy库
import matplotlib.pyplot as plt  # 导入Matplotlib
from mpl_toolkits.mplot3d import Axes3D  # 导入3D样式库

# 数据准备
raw_data = np.loadtxt('outlier.txt', delimiter=' ')  # 读取数据
train_set = raw_data[:900, :]  # 训练集
test_set = raw_data[:100, :]  # 测试集

# 异常数据检测
model_onecalsssvm = OneClassSVM(nu=0.1, kernel="rbf", random_state=0)  # 创
建异常检测算法模型对象
model_onecalsssvm.fit(train_set)  # 训练模型
pre_test_outliers = model_onecalsssvm.predict(test_set)  # 异常检测

# 异常结果统计
toal_test_data = np.hstack((test_set, pre_test_outliers.reshape(test_set.sha
测试集和检测结果合并
normal_test_data = toal_test_data[toal_test_data[:, -1] == 1]  # 获得异常检测
结果中正常数据集
outlier_test_data = toal_test_data[toal_test_data[:, -1] == -1]  # 获得异常检
测结果中异常数据
n_test_outliers = outlier_test_data.shape[1]  # 获得异常的结果数量
total_count_test = toal_test_data.shape[0]  # 获得测试集样本量
print ('outliers: {0}/{1}'.format(n_test_outliers, total_count_test))  # 输
出异常的结果数量
print ('{:*^60}'.format(' all result data (limit 5) '))  # 打印标题
print (toal_test_data[:5])  # 打印输出前5条合并后的数据集

# 异常检测结果展示
plt.style.use('ggplot')  # 使用ggplot样式库
fig = plt.figure()  # 创建画布对象
ax = Axes3D(fig)  # 将画布转换为3D类型
s1 = ax.scatter(normal_test_data[:, 0], normal_test_data[:, 1], normal_test_
                marker='o')  # 画出正常样本点
```

```
    s2 = ax.scatter(outlier_test_data[:, 0], outlier_test_data[:, 1], outlier_te
                    marker='o')  # 画出异常样本点
    ax.w_xaxis.set_ticklabels([])  # 隐藏x轴标签，只保留刻度线
    ax.w_yaxis.set_ticklabels([])  # 隐藏y轴标签，只保留刻度线
    ax.w_zaxis.set_ticklabels([])  # 隐藏z轴标签，只保留刻度线
    ax.legend([s1, s2], ['normal points', 'outliers'], loc=0)  # 设置两类样本点的
    图例
    plt.title('novelty detection')  # 设置图像标题
    plt.show()  # 展示图像
```

上述代码以空行分为5个部分。

第一部分导入库。本示例使用OneClassSVM做异常检测，用numpy做数据导入和基本预处理，用Matplotlib做图像展示，用Axes3D将图形转换为3D类型。

第二部分数据准备。使用Numpy的loadtxt方法导入数据，数据集中的训练集和测试集直接切分为两部分。

第三部分异常数据监测。使用OneClassSVM做异常检测分析，先创建异常检测算法模型对象，然后使用fit方法对训练集做模型训练，最后应用测试集做新的异常数据的检测。

相关知识点：OneClassSVM

one-class SVM用于异常检测，它的基本原理是在给定的一组样本中，检测数据集的边界以便于区分新的数据点是否属于该类。它是基于密度检测方法的一种，属于无监督学习算法，拟合过程由于不存在数据标签列，因此只需要输入一个矩阵X即可。

one-class SVM属于SVM的一种，可用于高维数据的异常检测。one-class SVM提供了linear、poly、rbf、sigmoid和precomputed检测内核可供使用，甚至你可以自定义一个内核算法来调用。one-class SVM本质上还是一种分类，但这种分类与传统的“分类”意义不同，one-class SVM的分类是将数据分为“正常数据”和“异常数据”（用+1和-1表示），而传统的分类是将正常数据按照不同的特征分为几个不同的类别。

第四部分异常结果统计。本部分中先使用np.hastck将测试集的输入与预测的检测结果合并，从而得到一份完整的数据集；然后基于该数据集将数据划分为预测后的正常数据集和异常数据集，接着使用shape形

状方法获得异常的结果数量和测试的总样本量，并打印输出异常数据的数量以及整体测试集的带有预测结果标签的前5条数据。这里使用了之前介绍过的str.format方法，得到如下结果：

```
outliers: 6/100
**************** all result data (limit 5) *****************
[[ 0.03685295  0.0343899   0.09197858 -0.01026255 -0.00814121 -1.        ]
 [-0.0011522   0.02174971 -0.02040125  0.00986554 -0.03447136  1.        ]
 [-0.01258645  0.04736393  0.01110832 -0.01156876 -0.02334062  1.        ]
 [-0.02837847  0.04398011  0.00126378  0.02313849  0.00542565  1.        ]
 [ 0.02222529  0.00715191 -0.03713534 -0.02938668 -0.09915368 -1.        ]]
```

第五部分异常检测结果展示。在展示结果的部分使用了3D展示图形类型。

首先通过plt.style.use方法加载预定义的样式库ggplot。

相关知识点：matplotlib.pyplot的样式库

在使用Matplotlib画图时，可指定图像的样式。默认情况下，通过print plt.style.available方法可以得到可用的预定义样式库：

```
[u'seaborn-darkgrid',    u'seaborn-notebook',    u'classic',    u'seaborn-
ticks',            u'grayscale',            u'bmh',            u'seaborn-
talk',   u'dark_background',   u'ggplot',   u'fivethirtyeight',   u'seaborn-
colorblind',     u'seaborn-deep',     u'seaborn-whitegrid',     u'seaborn-
bright',  u'seaborn-poster',  u'seaborn-muted',  u'seaborn-paper',  u'seaborn-
white',  u'seaborn-pastel',  u'seaborn-dark',  u'seaborn',  u'seaborn-dark-
palette']
```

列表中的不同的样式库（每个字符串代表一个样式库），可使用plt.style.use（）方法指定来实现不同的样式风格。同时，也可以自己设计一个样式库，然后加载进来使用。

然后创建一个画布对象fig，并将该对象通过Axes3D方法转换为3D类型的对象ax。

接着在ax对象上通过scatter分别画出正常和异常的样本点，这里在画图的时候由于是三维图像，需要同时指定x、y和z轴的数据（对二维数据只需要指定x和y即可）。

使用w_xaxis.set_ticklabels、w_yaxis.set_ticklabels、w_zaxis.set_ticklabel方法隐藏x、y、z轴标签，只保留刻度线。

使用legend方法设置图例标签以及位置。图例位置自适应，图例标签值的设置可通过scatter中的legend参数设置。

使用title方法设置标题并通过show方法展示图像，结果如图4-10所示。

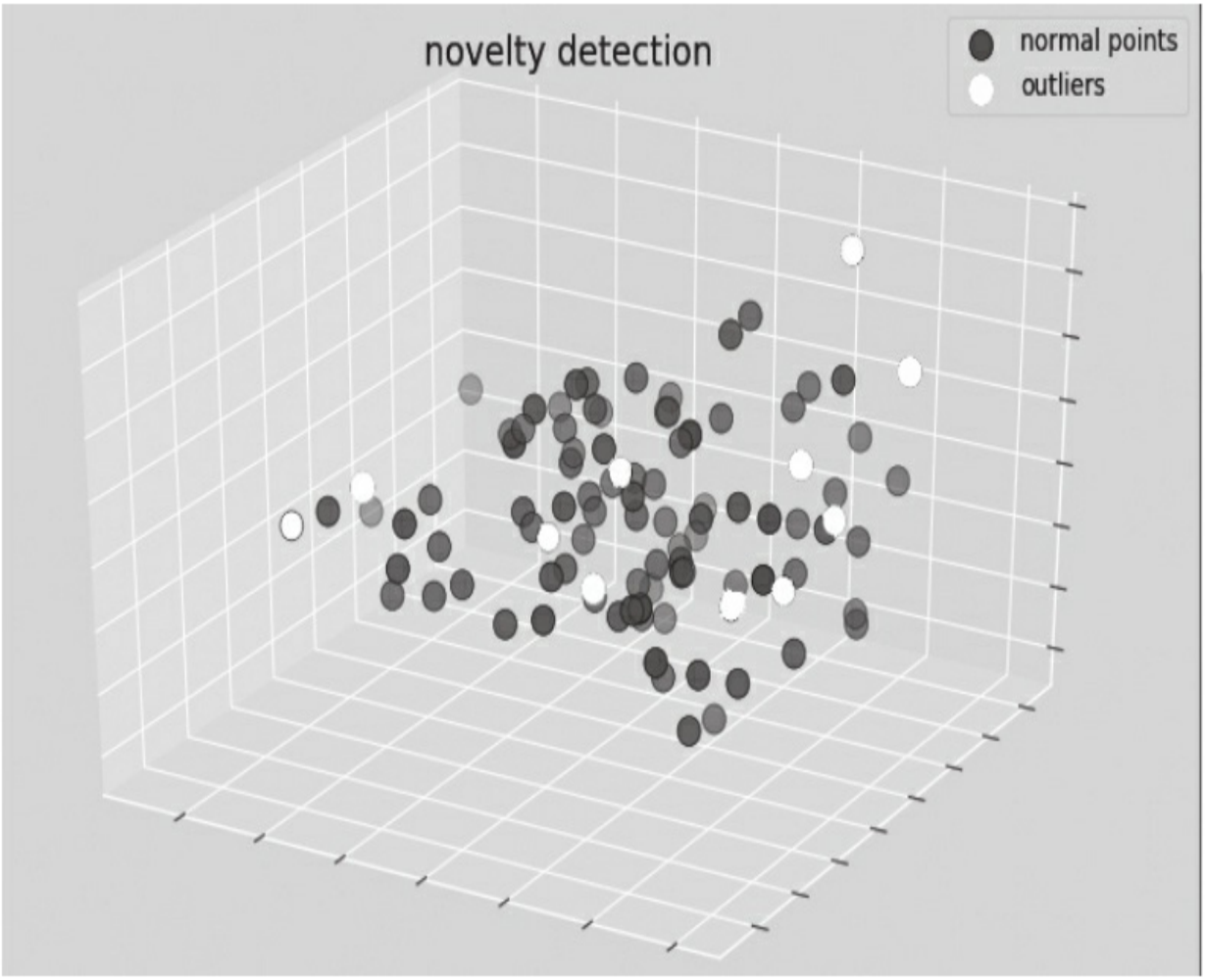


图4-10　异常检测3D图

图形中的空心圆色代表异常点，实心圆代表正常点。该图形可直接通过鼠标拖拽以显示不同3D视角下的数据分布，这在某些区域的点比较集中时非常有用。

上述过程中，主要需要考虑的关键点是：

·注意区分one-class SVM和EllipticEnvelope的应用数据集的限制并根据不同实际情况选择算法；

·如果要做新奇检测或要适应更多的数据集分布，那么使用one-class SVM，但需要注意新奇检测的训练数据必须是不包含噪音的干净数据，否则新奇点将可能无法监测出来。

代码实操小结：在本节的代码中，主要用了以下几个知识点：

·通过Numpy的loadtxt读取数据文件；

·对矩阵做切片操作；

·使用sklearn.svm中的OneClassSVM方法实现异常检测分析，并使用fit方法针对训练集，使用predict方法针对测试集应用；

·使用Numpy的hstack方法将矩阵按列合并，得到新的矩阵；

·通过对矩阵中特定列的值进行判断，直接选择或切分数据集；

·通过shape方法获得矩阵的形状；

·通过print方法结合str.format做格式化的输出；

·通过plt.style.use方法使用Matplotlib的预定义库样式；

·通过mpl_toolkits.mplot3d的Axes3D方法做3D类型图像转换；

·通过matplotlib.pyplot的scatter方法画散点图，并通过展示图形，设置不同的展示样式，包括颜色、样式、图例等，针对并隐藏坐标轴标签、设置图例和标签、设置标题等。

4.6 时间序列分析

时间序列是用来研究数据随时间变化趋势而变化的一类算法，它是一种常用的预测性分析方法。它的基本出发点是事物的发展都具有连续性，都是按照它本身固有的规律进行的。在一定条件下，只要规律赖以发生的条件不产生质的变化，则事物在未来的基本发展趋势仍然会延续下去。

从本质上看，时间序列算法是利用统计技术与方法，从预测指标的连续型规律中找出演变模式并建立数学模型，对预测指标的未来发展趋势做出定量估计。

时间序列的常用算法包括移动平均（MA，Moving Average）、指数平滑（ES，Exponen-tial Smoothing）、差分自回归移动平均模型（ARIMA，Auto-regressive Integrated Moving Average Model）三大主要类别，每个类别又可以细分和延伸出多种算法。

时间序列可以解决在只有时间（序列项）而没有其他可控变量下对未来数据的预测问题，常用于经济预测、股市预测、天气预测等偏宏观或没有可控自变量的场景下。

4.6.1 如果有自变量，为什么还要用时间序列

时间序列通常用于在没有自变量可用的条件下做预测性分析，但在某些场景下，即使有自变量仍然需要用到时间序列。下面以笔者经历的一个案例说明该问题。

项目的背景是要做地区用电量预测，但现有的数据可用的只有三份数据，一份是地区每5分钟一个点的量测值数据，一份是该地区每隔1小时的天气数据，另外还有季节性数据例如节假日、工作休息日等。

表面上看，可以将天气和季节性数据作为自变量、将用电量数据作为因变量建立回归模型，但是从本质上考虑，地区的用电情况虽然会受到天气和季节性变化的影响（例如夏天空调），但这种影响对于整个地区的用电影响非常小，因为该地区的用户大户是企业客户、工业用电客户，这些客户的用电量往往更多地受到产销、政策和经济环境这些大因素的影响，而跟季节性本身的关系非常小。因此，这种回归模型很可能无法满足用户预测的需求。

为了验证这一猜想，我们使用了多种回归模型做用电量预测，无一例外地发现效果非常差；经过跟业务部门的沟通也证实了我们的猜想是对的。

经过后续的分析讨论，我们认为用电量趋势中一定会存在随着时间变化而变化的隐形规律，而天气和季节性因素则可以作为外部影响因素对结果加以调整。基于这样的假设，我们使用了ARIMA+SVR（svm中的回归模型器）结合起来做用电量预测，基本思路是：先通过ARIMA得到下一个时间点t的预测用电量值x1、预测值上下限x2、预测样本量x3；然后将这三个特征，再加上天气和季节性因素例如温度x4、湿度x5、风力x6、节气x7等几十个特征作为输入变量，来预测t时刻点的真实用电量。

结果证明，这种方法比单独使用任何一种回归和时间序列得到的预测结果更符合实际情况。ARIMA基于其中隐含的主要规律预测出用电的“主要部分”，而不同的天气和季节性因素则作为外部因素来增加对结果的修正，弥补了无法通过时间规律反映出的“次要部分”和“波动部分”。

该案例说明了即使在有自变量的前期下，也可以使用回归方法来解释其中的“主要规律”，然后通过其他方法对结果加以修正来实现更加精准的预测。在实际运营分析中，可能也存在类似的场景：在要分析的主题中无法确定主要变量因素，或即使确定了主要变量但是无法获得其数据，那么可采用这种方法来实现。

4.6.2 时间序列不适合商业环境复杂的企业

在国内变化多端的商业环境中，很多因素都会影响企业的运营，而这些因素中的大多数都无法通过时间性规律反映出来，因此时间序列很难在这样的场景下真正发挥效果。常见的对时间序列影响较大的因素包括：

·融资、并购、收购：当新的资源进入企业后，原来的企业发展模式都会面临重构。

·人工造势：很多行业巨擘都有造势的能力，例如天猫双11、京东618等，这些事件都跟时间没有太大关系，但确实也能极大影响个别日期的数据状态。

·恶意商业活动：当企业被某些恶意商业活动攻击时，会产生很多无法预知的信息，例如恶意流量、黄牛订单、活动刷单、内部订单等。

·广告活动：广告活动对流量和销售的影响几乎是决定性的，有关用户、销售、流量等方面的分析跟时间规律几乎没有关系，只关乎有多少广告费用。

·促销活动：促销活动对于销售型公司而言，几乎是运营必备，而且现在基本都演变为“没有促销就没有销售”的状态。

·人为因素：每个企业总有一些不可预知的人为因素，能产生一些意想不到的结果，例如删除数据库、商品调价错误、内部信息泄露、升级系统导致的程序崩溃等导致数据异常。

·系统问题：在很多企业内部，IT服务系统自身出现了问题，这也会直接影响运营结果，例如高并发下服务器响应慢、服务器宕机、系统内部错误等都将导致数据异常。

·竞争对手影响：在一定周期内的市场容量是稳定的，当竞争对手采取一定的活动后，会给自身企业来带负面影响。例如竞争对手在本企业店庆之前做促销活动，必然会吸引一部分用户去消费，尤其是对于两个网站重叠的会员来讲更是如此，被竞争对手提前透支了消费能力之

后，本企业再做促销活动就很难达到预期效果。

·宏观政策影响：宏观政策调整对企业运营的影响非常大，在很多情况下会成为主要因素。例如，北京的商品房调控政策出来之后，二手房的成交率下降70%。

·企业经营策略的转变：很多企业尤其是中小企业都定期制定或修订经营策略，这种经营模式的变革对企业运营的影响非常大。

因此，实际运营分析中，只使用时间序列做预测性的分析场景相对较少。

4.6.3 时间序列预测的整合、横向和纵向模式

当数据集规模较大、数据粒度丰富时，我们可以选择多种时间序列的预测模式。时间序列的数据粒度可分为秒、分、小时、天、周、月、季度、年等，不同的粒度都可以用来做时间序列预测。

假如有3650天（完整10年）以每分钟时间戳为粒度的数据，那么这份数据集会有5256000（60分钟*24小时*365天*10年）条记录。基于该数据集预测今天的总数据，可以考虑的时间序列模式有三种：

·整合模式：将历史数据按每天数据进行汇总，得到按天为粒度的共计3650条时间序列数据；然后应用所有数据集做时间序列分析，预测的时间序列项目为1（天）。

·横向模式：将历史数据按每小时做汇总，得到按小时为粒度87600条时间序列数据；然后将要预测的1天划分为24小时（即24个预测点），这样会形成24个预测模型，每个模型只预测对应的小时点数据，预测的时间序列项目为1（小时）；最后将预测得到的24个小时点的数据求和得到今天的总数据。

·纵向模式：这种模式的粒度更细，无需对历史数据做任何形式的汇总，这样有525600条时间序列数据。将要预测的1天划分为1440个分钟点（60分钟*24小时），这样会形成1440个预测模型，每个模型只预测对应的分钟点的数据，预测的时间序列项目为1（分钟）；最后将预测得到的1440个分钟点的数据求和得到今天的总数据。

这三种方式的训练集都是3650条，即按天为粒度的数据。横向模式和纵向模式由于做更细的粒度切分，因此需要更多的模型，这意味着需要更多的时间来做训练和预测。当第一种整合模式不能满足需求或者预测结果不佳时，可以尝试后两种模式。

4.6.4 代码实操：Python时间序列分析

Python的科学计算和数据挖掘相关库中，Pandas和statsmodels都提供了时间序列相关分析功能，本示例使用的是statsmodels。有关时间序列算法的选择，实际场景中最常用的是ARIMA或ARMA，因此本示例将使用ARIMA/ARMA来做时间序列分析。

对于这两种时间序列方法而言，应用的难点是如何根据不同的场景判断参数值。本示例将设置判断阈值，通过自动化的程序方式来完成自动的ARIMA/ARMA的参数（p、d、q）选择以及模型训练，降低时间序列算法应用的难度。

示例中模拟的是针对具有时间序列特征的数据集做未来时间序列的预测，数据源文件time_series.txt位于“附件-chapter4”中，默认工作目录为“附件-chapter4”（如果不是，请cd切换到该目录下，否则会报“IOError:File time_series.txt does not exist”）。完整代码如下：

```
# 导入库
import pandas as pd  # Pandas库
import numpy as np  # Numpy库
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf  # acf和pacf展示库
from statsmodels.tsa.stattools import adfuller  # adf检验库
from statsmodels.stats.diagnostic import acorr_ljungbox  # 随机性检验库
from statsmodels.tsa.arima_model import ARMA  # ARMA库
import matplotlib.pyplot as plt  # Matplotlib图形展示库
import prettytable  # 导入表格库

# 多次用到的表格
def pre_table(table_name, table_rows):
    '''
    :param table_name: 表格名称，字符串列表
    :param table_rows: 表格内容，嵌套列表
    :return: 展示表格对象
    '''
    table = prettytable.PrettyTable()  # 创建表格实例
    table.field_names = table_name  # 定义表格列名
    for i in table_rows:  # 循环读多条数据
        table.add_row(i)  # 增加数据
    return table

# 数据平稳处理
def get_best_log(ts, max_log=5, rule1=True, rule2=True):
    '''
    :param ts: 时间序列数据，Series类型
    :param max_log: 最大log处理的次数，int型
    :param rule1: rule1规则布尔值，布尔型
    :param rule2: rule2规则布尔值，布尔型
```

```
    :return: 达到平稳处理的最佳次数值和处理后的时间序列
    '''
    if rule1 and rule2:  # 如果两个规则同时满足
        return 0, ts  # 直接返回0和原始时间序列数据
    else:  # 只要有一个规则不满足
        for i in range(1, max_log):  # 循环做log处理
            ts = np.log(ts)  # log处理
            lbvalue, pvalue1 = acorr_ljungbox(ts, lags=1)  # 白噪声检验结果
            adf, pvalue2, usedlag, nobs, critical_values, icbest = adfuller(
检验
            rule_1 = (adf < critical_values['1%'] and adf < critical_values[
定性检验
            rule_2 = (pvalue2 < 0.05)  # 白噪声检验
            rule_3 = (i < 5)
            if rule_1 and rule_2 and rule_3:  # 如果同时满足条件
                print ('The best log n is: {0}'.format(i))  # 打印输出最佳次数
                return i, ts  # 返回最佳次数和处理后的时间序列

# 还原经过平稳处理的数据
def recover_log(ts, log_n):
    '''
    :param ts: 经过log方法平稳处理的时间序列，Series类型
    :param log_n: log方法处理的次数，int型
    :return: 还原后的时间序列
    '''
    for i in range(1, log_n + 1):  # 循环多次
        ts = np.exp(ts)  # log方法还原
    return ts  # 返回时间序列

# 平稳性检验
def adf_val(ts, ts_title, acf_title, pacf_title):
    '''
    :param ts: 时间序列数据，Series类型
    :param ts_title: 时间序列图的标题名称，字符串
    :param acf_title: acf图的标题名称，字符串
    :param pacf_title: pacf图的标题名称，字符串
    :return: adf值、adf的p值、三种状态的检验值
    '''
    plt.figure()
    plt.plot(ts)  # 时间序列图
    plt.title(ts_title)  # 时间序列标题
    plt.show()
    plot_acf(ts, lags=20, title=acf_title).show()  # 自相关检测
    plot_pacf(ts, lags=20, title=pacf_title).show()  # 偏相关检测
    adf, pvalue, usedlag, nobs, critical_values, icbest = adfuller(ts)  # 平
稳性检验
    table_name = ['adf', 'pvalue', 'usedlag', 'nobs', 'critical_values', 'ic
格列名列表
    table_rows = [[adf, pvalue, usedlag, nobs, critical_values, icbest]]  #
格行数据，嵌套列表
    adf_table = pre_table(table_name, table_rows)  # 获得平稳性展示表格对象
    print ('stochastic score')  # 打印标题
    print (adf_table)  # 打印展示表格
     return adf, pvalue, critical_values,  # 返回adf值、adf的p值、三种状态的检验
值

# 白噪声（随机性）检验
def acorr_val(ts):
    '''
    :param ts: 时间序列数据，Series类型
    :return: 白噪声检验的P值和展示数据表格对象
```

```
    '''
    lbvalue, pvalue = acorr_ljungbox(ts, lags=1)  # 白噪声检验结果
    table_name = ['lbvalue', 'pvalue']  # 表格列名列表
    table_rows = [[lbvalue, pvalue]]  # 表格行数据，嵌套列表
    acorr_ljungbox_table = pre_table(table_name, table_rows)  # 获得白噪声检验
展示表格对象
    print ('stationarity score')  # 打印标题
    print (acorr_ljungbox_table)  # 打印展示表格
    return pvalue  # 返回白噪声检验的P值和展示数据表格对象

# arma最优模型训练
def arma_fit(ts):
    '''
    :param ts: 时间序列数据，Series类型
    :return: 最优状态下的p值、q值、arma模型对象、pdq数据框和展示参数表格对象
    '''
    max_count = int(len(ts) / 10)  # 最大循环次数最大定义为记录数的10%
    bic = float('inf')  # 初始值为正无穷
    tmp_score = []  # 临时p、q、aic、bic和hqic的值的列表
    for tmp_p in range(max_count + 1):  # p循环max_count+1次
        for tmp_q in range(max_count + 1):  # q循环max_count+1次
            model = ARMA(ts, order=(tmp_p, tmp_q))  # 创建ARMA模型对象
            try:
                 results_ARMA = model.fit(disp=-1, method='css')  # ARMA模型
训练
            except:
                continue  # 遇到报错继续
            finally:
                tmp_aic = results_ARMA.aic  # 模型的获得aic
                tmp_bic = results_ARMA.bic  # 模型的获得bic
                tmp_hqic = results_ARMA.hqic  # 模型的获得hqic
                tmp_score.append([tmp_p, tmp_q, tmp_aic, tmp_bic, tmp_hqic])
加每个模型的训练参数和结果
                 if tmp_bic < bic:  # 如果模型bic小于最小值，那么获得最优模型ARMA
的下列参数：
                    p = tmp_p  # 最优模型ARMA的p值
                    q = tmp_q  # 最优模型ARMA的q值
                    model_arma = results_ARMA  # 最优模型ARMA的模型对象
                    aic = tmp_bic  # 最优模型ARMA的aic
                    bic = tmp_bic  # 最优模型ARMA的bic
                    hqic = tmp_bic  # 最优模型ARMA的hqic
    pdq_metrix = np.array(tmp_score)  # 将嵌套列表转换为矩阵
                  pdq_pd     =     pd.DataFrame(pdq_metrix,     columns=
['p', 'q', 'aic', 'bic', 'hqic'])  # 基于矩阵创建数据框
    table_name = ['p', 'q', 'aic', 'bic', 'hqic']  # 表格列名列表
    table_rows = [[p, q, aic, bic, hqic]]  # 表格行数据，嵌套列表
     parameter_table = pre_table(table_name, table_rows)  # 获得最佳ARMA模型结
果展示表格对象
    print ('each p/q traning record')  # 打印标题
      print  (pdq_pd)   #  打印输出每次ARMA拟合结果，包含p、d、q以及对应的AIC、BIC、
HQIC
    print ('best p and q')  # 打印标题
    print (parameter_table)  # 输出最佳ARMA模型结果展示表格对象
    return model_arma  # 最优状态下的arma模型对象

# 模型训练和效果评估
def train_test(model_arma, ts, log_n, rule1=True, rule2=True):
    '''
    :param model_arma: 最优ARMA模型对象
    :param ts: 时间序列数据，Series类型
    :param log_n: 平稳性处理的log的次数，int型
```

```
    :param rule1: rule1规则布尔值，布尔型
    :param rule2: rule2规则布尔值，布尔型
    :return: 还原后的时间序列
    '''
    train_predict = model_arma.predict()  # 得到训练集的预测时间序列
    if not (rule1 and rule2):  # 如果两个条件有任意一个不满足
        train_predict = recover_log(train_predict, log_n)  # 恢复平稳性处理前的真实时间序列值
        ts = recover_log(ts, log_n)  # 时间序列还原处理
    ts_data_new = ts[train_predict.index]  # 将原始时间序列数据的长度与预测的周期对齐
    RMSE = np.sqrt(np.sum((train_predict - ts_data_new) ** 2) / ts_data_new.RMSE
    # 对比训练集的预测和真实数据
    plt.figure()  # 创建画布
    train_predict.plot(label='predicted data', style='--')  # 以虚线展示预测数据
    ts_data_new.plot(label='raw data')  # 以实线展示原始数据
    plt.legend(loc='best')  # 设置图例位置
    plt.title('raw data and predicted data with RMSE of %.2f' % RMSE)  # 设置标题
    plt.show()  # 展示图像
    return ts  # 返回还原后的时间序列

# 预测未来指定时间项的数据
def predict_data(model_arma, ts, log_n, start, end, rule1=True, rule2=True):
    '''
    :param model_arma: 最优ARMA模型对象
    :param ts: 时间序列数据，Series类型
    :param log_n: 平稳性处理的log的次数，int型
    :param start: 要预测数据的开始时间索引
    :param end: 要预测数据的结束时间索引
    :param rule1: rule1规则布尔值，布尔型
    :param rule2: rule2规则布尔值，布尔型
    :return: 无
    '''
    predict_ts = model_arma.predict(start=start, end=end)  # 预测未来指定时间项的数据
    print ('-----------predict data----------')  # 打印标题
    if not (rule1 and rule2):  # 如果两个条件有任意一个不满足
        predict_ts = recover_log(predict_ts, log_n)  # 还原数据
    print (predict_ts)  # 展示预测数据
    # 展示预测趋势
    plt.figure()  # 创建画布
    ts.plot(label='raw time series')  # 设置推向标签
    predict_ts.plot(label='predicted data', style='--')  # 以虚线展示预测数据
    plt.legend(loc='best')  # 设置图例位置
    plt.title('predicted time series')  # 设置标题
    plt.show()  # 展示图像

# 读取数据
date_parse = lambda dates: pd.datetime.strptime(dates, '%m-%d-%Y')  # 创建解析列的功能对象
df = pd.read_table('time_series.txt', delimiter='\t', index_col='date', date
取数据
ts_data = df['number'].astype('float32')  # 将列转换为float32类型
print ('data summary')  # 打印标题
print (ts_data.describe())  # 打印输出时间序列数据概况
# 原始数据检验
adf, pvalue1, critical_values = adf_val(ts_data, 'raw time series', 'raw acf
定性检验
```

```
pvalue2 = acorr_val(ts_data)  # 白噪声检验
# 创建用于区分是否进行平稳性处理的规则
rule1 = (adf < critical_values['1%'] and adf < critical_values['5%'] and adf
    '10%'] and pvalue1 < 0.01)  # 稳定性检验
rule2 = (pvalue2[0,] < 0.05)  # 白噪声检验
# 对时间序列做稳定性处理
log_n, ts_data = get_best_log(ts_data, max_log=5, rule1=rule1, rule2=rule2)
# 再次做检验
adf, pvalue1, critical_values = adf_val(ts_data, 'final time series', 'final
定性检验
pvalue2 = acorr_val(ts_data)  # 白噪声检验
# 训练最佳ARMA模型并输出相关参数和对象
model_arma = arma_fit(ts_data)
# 模型训练和效果评估
ts_data = train_test(model_arma, ts_data, log_n, rule1=rule1, rule2=rule2)
# 模型预测应用
start = '1991-07-28'  # 设置预测开始的时间索引
end = '1991-08-02'  # 设置预测结束的时间索引
predict_data(model_arma, ts_data, log_n, start, end, rule1=rule1, rule2=rule
测时间序列数据
```

上述代码以空行分为10个部分。

1）第一部分导入库。statsmodels库里面的plot_acf、plot_pacf用于acf和pacf图形检验分析，adfuller、acorr_ljungbox用于adf检验和随机性检验，ARMA库做时间序列分析，Matplotlib和prettytable做图形和表格格式化输出。

ARIMA相对于ARMA多了一步差分的过程，但是由于statsmodels中的ARIMA对差分的支持不太好，最多只能多2阶差分，其实用性会大打折扣（其实2阶差分已经可以满足90%以上的场景了）。所以，在实际应用中，我们会手动做数据平稳性处理，然后将平稳数据传给ARMA模型做训练，完成类似ARIMA的时间序列分析。

2）第二部分定义一个名为pre_table的函数用来展示格式化表格数据。该函数的输入有2个参数：

·table_name：表格名称，字符串列表；

·table_rows：表格内容，嵌套列表，考虑到在添加内容时，可能会有批量添加的模式，因此批量添加的数据记录行需要转换为嵌套列表的形式，例如[[1，2，3，4]，[2，2，3，4]]为要添加的两条数据记录；即使只有一条数据记录，也要写成嵌套列表的形式，例如[[1，2，3，

4]]。

返回一个包含内容的可打印对象。

函数功能中，先通过prettytable.PrettyTable创建表格对象实例，然后通过实例的field_names方法增加表格名称，通过一个for循环将table_rows中的多个嵌套表格内容读出，并使用表格实例的add_row方法增加行数据。

在实际应用中，只要是用到2次或2次以上的功能，或者完成一项功能需要的代码为比较大的模块，笔者都会写成函数来调用，这样一方面能将降低代码冗余，另一方面也能增加代码的可维护性、可理解性和可编辑性。

相关知识点：Python中函数的Doc String

从本示例开始，笔者会尽量在每个函数中使用Doc String做函数注释。Doc String是文档字符串，内容一般是对象的描述信息。Doc Strings是一个重要的工具，它能帮助程序读者快速了解程序的功能、用法和注意事项，使得程序更加简单易懂；尤其当程序需要多部门流通使用，或者在大型项目交付时，通常都会通过Doc Strings来增强代码的可理解性。

Doc String使用三个引号（单引号和双引号都可以）作为标识，通常包括以下几部分内容：

·函数功能简述：概述该函数或模块的主要功能。

·参数信息：详细介绍每个参数含义、类型和用法。

·返回：该模块会返回的内容、类型等信息。

·相关信息：跟该函数模块相关的其他函数或信息。

·报错：该函数会在哪些场景下报哪些错误。

·注意事项：其他有关该函数模块的使用注意和信息等。

·应用示例：通过简单的几个例子说明该函数或模块的用法。

例如，下面是numpy.dot的文档字符串信息：

```
dot(...)
    dot(a, b, out=None)

    Dot product of two arrays.

    For 2-D arrays it is equivalent to matrix multiplication, and for 1-D
    arrays to inner product of vectors (without complex conjugation). For
    N dimensions it is a sum product over the last axis of `a` and
    the second-to-last of `b`::

        dot(a, b)[i,j,k,m] = sum(a[i,j,:] * b[k,:,m])

    Parameters
    ----------
    a : array_like
        First argument.
    b : array_like
        Second argument.
    out : ndarray, optional
        Output argument. This must have the exact kind that would be returne
        if it was not used. In particular, it must have the right type, must
                                                                        C-
contiguous, and its dtype must be the dtype that would be returned
        for `dot(a,b)`. This is a performance feature. Therefore, if these
        conditions are not met, an exception is raised, instead of attemptir
        to be flexible.

    Returns
    -------
    output : ndarray
        Returns the dot product of `a` and `b`.  If `a` and `b` are both
        scalars or both 1-D arrays then a scalar is returned; otherwise
        an array is returned.
        If `out` is given, then it is returned.

    Raises
    ------
    ValueError
        If the last dimension of `a` is not the same size as
        the second-to-last dimension of `b`.

    See Also
    --------
    vdot : Complex-conjugating dot product.
    tensordot : Sum products over arbitrary axes.
    einsum : Einstein summation convention.
    matmul : '@' operator as method with out parameter.

    Examples
    --------
    >>> np.dot(3, 4)
    12

    Neither argument is complex-conjugated:
```

```
>>> np.dot([2j, 3j], [2j, 3j])
(-13+0j)

For 2-D arrays it is the matrix product:

>>> a = [[1, 0], [0, 1]]
>>> b = [[4, 1], [2, 2]]
>>> np.dot(a, b)
array([[4, 1],
       [2, 2]])

>>> a = np.arange(3*4*5*6).reshape((3,4,5,6))
>>> b = np.arange(3*4*5*6)[::-1].reshape((5,4,6,3))
>>> np.dot(a, b)[2,3,2,1,2,2]
499128
>>> sum(a[2,3,2,:] * b[1,2,:,2])
499128
```

不同模块间一般以空行分隔，用于区分不同内容。在本书中，由于代码会在书中进行讲解，因此不会完整包含上述内容，但是函数功能简述、参数信息和返回这三部分关键信息都在有相应提示。

上述注释信息经常在那个地方可以查询或使用到：

·help（）方法获取帮助信息。在使用Python的过程中，当遇到函数功能不清楚的情况时，通常会使用help（funciton_name）来查找其帮助信息，其中funciton_name就是函数的名称，而通过help（）方法调用的正是Doc String中的内容。

·使用funciton_name.__doc__方法输出函数的帮助信息。除了上述使用help方法查询模块功能外，也可以使用函数的.__doc__方法获取帮助信息，该帮助信息也是获取的Doc String内容，例如function_name.__doc__。

·生成Python相关帮助文档。在做项目交付时都需要完整的文档交付，其中涉及函数功能的部分，可以使用相关工具，例如pydoc、epydoc、doxygen、sphinx来直接生成对应的帮助文档，这些自动化工具可以提取这些信息来生成文档。

最简单的应用示例

在本示例代码中，当执行了pre_table的函数模块后，可在python交互窗口直接输入：help（pre_table），会返回如下帮助信息：

```
Help on function pre_table in module __main__:
pre_table(table_name, table_rows)
    :param table_name: 表格名称，字符串列表
    :param table_rows: 表格内容，嵌套列表
    :return: 展示表格对象
```

这是我们在注释中写的帮助内容。无论是出于可维护、可交付还是可理解的目的，都建议读者在正式的函数编程时养成写标准注释的好习惯。

3）第三部分定义一个名为get_best_log的函数用于数据平稳处理。该函数的输入参数有5个：

·ts：时间序列数据，Series类型。

·max_log：最大log处理的次数，默认值为5，int型。

·rule1：rule1规则布尔值，默认值为True，布尔型。

·rule2：rule2规则布尔值，默认值为True，布尔型。

返回：达到平稳处理的最佳log处理次数和处理后的时间序列。

该函数功能中，先通过rule1和rule2条件判断输入的两个规则是否同时满足，如果同时满足则说明该时间序列是平稳的，无需做任何平稳性处理；否则使用对数方法（numpy.log）做平稳性处理：

通过for循环按照指定的最大处理次数做循环处理，一般情况下做2次log处理就能满足90%的时间序列平稳性。

每次处理后使用acorr_ljungbox做白噪声检验、使用adfuller方法做ADF检验，如果二者检验的结果能够符合两个规则的阈值设定并且i的值在5以内，那么返回平稳处理的次数以及处理后的时间序列；否则终止程序。平稳处理的次数用于后期数据的还原，处理后的时间序列用于ARMA模型中的时间序列的输入数据集。

相关知识点：时间序列数据的平稳性

平稳性是做时间序列分析的前提条件，所谓平稳通俗理解就是数据没有随着时间呈现明显的趋势和规律，例如剧烈波动、递增、递减等，而是相对均匀且随机地分布在均值附近。在ARIMA模型中的I就是对数据做差分以实现数据的平稳，而ARIMA关键参数p、d、q中的d即时间序列成为平稳时所做的差分次数。

如何判断时间序列数据是否需要平稳性处理？一般有三种方法：

·观察法：通过输出时间序列图发现数据是否平稳。本示例中的adf_val函数便含有该方法。

·自相关和偏相关法：通过观察自相关和偏相关的系数分析数据是否平稳。

·ADF检验：通过ADF检验得到的显著性水平分析数据是否平稳。本示例中的adf_val函数便有含有该方法。

实现数据的平稳有以下几种方法：

·对数法：对数处理可以减小数据的波动，使其线性规律更加明显。但需要注意的是对数处理只能针对时间序列的值大于0的数据集。

·差分法：一般来说，非纯随机的时间序列经一阶差分或者二阶差分之后就会变得平稳（statsmodels的ARIMA模型自动支持数据差分，但最大为2阶差分）。

·平滑法：根据平滑技术的不同，可分为移动平均法和指数平均法，这两个都是最基本的时间序列方法。

·分解法：将时序数据分离成不同的成分，包括成长期趋势、季节趋势和随机成分等。

4）第四部分定义了一个名为recover_log的函数，目的是还原经过平稳处理的数据。在经过平稳性处理之后，原有的时间序列训练集已经被更改，通过ARMA得到的预测数据也是经过平稳处理后的数据，该数据要使用必须经过还原。该函数的参数：

·ts：经过log方法平稳处理的时间序列，Series类型。

·log_n：log方法处理的次数，int型。

返回：还原后的时间序列。

该函数的功能中直接通过for循环通过平稳性处理得到的最佳处理次数，使用Numpy的exp方法做逆对数处理得到原始时间序列值。

5）第五部分定义了一个名为adf_val的函数，用于平稳性检验。该平稳性检验实现了时间序列图、acf图、pacf图、ADF检验数据等多种检验方法的数据和图形输出。函数的参数：

·ts：时间序列数据，Series类型。

·ts_title：自定义的时间序列图的标题名称，字符串。

·acf_title：自定义的acf图的标题名称，字符串。

·pacf_title：自定义的pacf图的标题名称，字符串。

返回：adf值、adf的p值、三种状态（1%、5%、10%）的检验值，这些值是用来构成rule1的判断规则的参数值。

该函数的功能中，先调用Matplotlib的plot方法直接输出时间序列图，然后使用statsmodels.graphics.tsaplots的plot_acf和plot_pacf方法分别输出自相关和偏相关检测图，使用adfuller方法做平稳性检验；接着通过定义要输出的表格的列名列表和行记录列表，调用pre_table方法形成格式化表格并打印输出。

6）第六部分定义了一个名为acorr_val的函数，用于白噪声（随机性）检验。函数参数：

·ts：时间序列数据，Series类型。

返回：白噪声检验的P值，用来做rule2判断规则的参数值。

函数参数：

函数功能中，使用acorr_ljungbox返回白噪声检验结果，然后调用pre_table将数据格式化为表格形式并打印出来。

相关知识点：白噪声检验

白噪声（white noise）检验也称为随机性检验，用于检验时间序列的各项数值之间是否具有任何相关关系，白噪声分布是应用时间序列分析的前提。检测时间序列是否属于随机性分布，可通过图形法和Ljung Box法检验。

·图形法：时间序列应该是围绕均值随机性上下分布的状态。

·Ljung Box法：是对时间序列是否存在滞后相关的一种统计检验，判断序列总体的相关性或者说随机性是否存在。

白噪声检验通常和数据平稳性检验是协同进行数据检验的，这意味着如果平稳性检验通过，白噪声检验一般也会通过。

7）第七部分定义了一个名为arma_fit的函数，用于做arma最优模型训练。该模型实现的是基于给定的时间序列数据，自动寻找BIC最小时的p和q的阶数，并形成训练好的ARMA模型。函数参数：

·ts：时间序列数据，Series类型。

返回：最优状态下arma模型对象，用于后续模型训练效果评估和预测应用。需要注意的是，输入的ts时间序列数据是已经经过平稳性处理，并满足平稳性和白噪音检验的时间序列数据。

函数实现功能中，先定义了几个值。默认p和q阶数的最大值为记录数的10%；判断最佳ARMA模型的方法是BIC方法，即选择最小的BIC值时的q和p的值，因此通过float（'inf'）来定义一个BIC值为正无穷，后续模型得到BIC值更小时选择对应的参数信息；tmp_score为临时p、q、aic、bic和hqic的值的列表，该列表将嵌套每次训练形成的评估参数结果集。

接下来的循环主体，将通过p和q的嵌套循环实现。

·最内层循环中，先通过ARMA方法基于输入的时间序列和循环得到的p、q值创建模型对象。

·然后应用模型对象的fit方法做模型训练，fit中的参数disp用来控制不打印收敛信息，mothed控制最大似然的计算方法是条件平方和似然度最大化。

·在应用fit过程中，由于多次循环可能会报错导致循环中止，因此使用了try、except和finally进行控制，这三个语句的含义为执行try中的程序，如果遇到错误则执行except中的程序，最终无论是否报错都要执行finally中的程序。在finally语句中，通过fit后返回的结果类的aic、bic和hqic三个参数值，然后将p、q、aic、bic、hqic以列表的形式追加到临时列表中；如果判断bic的值比最小值要小，那么认为该值下的模型为更优模型，将其p、q、aic、bic和hqic的值存储下来。

·最后将临时列表tmp_score转换为矩阵，并基于该矩阵创建用于展示每次循环数据的详细数据；然后基于最优模型的数据创建格式化展示列表并分别输出详细数据和最优模型对象。

statsmodels中的ARMA通过应用fit方法后，返回的对象是一个ARMAResults类，而不再是原有的model（ARMA模型对象）本身，这一点跟sklearn不同，因此无法直接通过model本身获取模型的参数和评估指标等信息，而要通过ARMAResults来获取。

相关知识点：statsmodels中的ARMA

statsmodels常用的时间序列模型包括AR、ARMA和ARIMA，并都集中在statsmodels.tsa.arima_model下面。ARMA（p，q）是AR（p）和MA（q）模型的组合，关于p和q的选择，一种方法是观察自相关图ACF和偏相关图PACF，通过截尾、拖尾等信息分析应用的模型以及适应的p和q的阶数；另一种方法是通过循环的方法，遍历所有条件并通过AIC、BIC值自动确定阶数。

ARMA（以及statsmodels中的其他算法）跟sklearn中的算法类似，也都有方法和属性两种应用方式。常用的ARMA方法：

·fit：使用卡尔曼滤波器的最大似然法拟合ARMA模型。

·from_formula：基于给定的公式和数据框创建模型。

·geterrors：基于给定的参数获取ARMA进程的错误。

·hessian：基于给定的参数计算Hessian信息。

·loglike：计算ARMA模型的对数似然。

·loglike_css：条件求和方差似然函数。

·predict：应用ARMA模型做预测。

·score：得分函数。

8）第八部分定义了一个名为train_test的函数，用于时间序列的模型训练和效果评估。该函数可将训练的时间序列内预测值与实际值做比较，并输出RMSE（均方根误差）和时间序列对比图。函数参数：

·model_arma：最优ARMA模型对象。

·ts：时间序列数据，Series类型。

·log_n：平稳性处理的log的次数，int型。

·rule1：rule1规则布尔值，布尔型。

·rule2：rule2规则布尔值，布尔型。

返回：还原后的时间序列，该时间序列用于在时间序列外做预测后的图形展示。

函数功能实现中，通过最优模型的predict方法做时间序列内的数据预测，如果数据经过平稳性处理，则将数据做还原处理；由于预测数据比原始时间序列数据的索引少，因此将二者匹配为相同索引下的数据量用于数据比较；通过自定义的公式计算RMSE；最后通过Matplotlib分别画出预测数据（虚线表示）、实际数据（实线表示）。

9）第九部分定义了一个名为predict_data的函数，用来预测未来指定时间项的数据。

函数参数：

·model_arma：最优ARMA模型对象。

·ts：时间序列数据，Series类型。

·log_n：平稳性处理的log的次数，int型。

·start：要预测数据的开始时间索引。

·end：要预测数据的结束时间索引。

·rule1：rule1规则布尔值，布尔型。

·rule2：rule2规则布尔值，布尔型。

返回：无。

该函数功能的实现中，通过最优模型的predict方法，指定时间序列外的开始和终止时间索引来定义要预测的时间区间；如果预测的结果经过平稳性处理，那么将结果做还原处理；最后通过Matplotlib分别将原始时间序列（实现）和预测的时间序列（虚线）输出到一个图像中。

10）第十部分为整个程序的应用部分。分为以下几个步骤：

步骤1　读取数据

先通过lambda函数定义了一个用于将日期解析出来的功能项，功能项的核心是使用lambda表达式配合pandas.datetime中的strptime方法将字符串解析为日期格式，该功能项会在Pandas读取数据文件时使用，日期的格式按照源文件的格式来标记，这样能准确解析源文件日期。

相关知识点：lambda表达式

lambda表达式是一个匿名函数，它其实是一个简易版的函数，通常

用于函数功能和逻辑比较简单且没有重复性使用的场景。lambda函数可以接收任意多个参数并且默认返回单个表达式的值，其函数的语法只包含一个语句：

```
lambda [arg1 [,arg2,.....argn]]:expression
```

其中arg1、arg2等是参数，expression是表达式主体部分。以下是几种常用的表达式示例。

示例一　单个参数求平方，普通的函数式可能会这样定义：

```
def square_x(x):
    x = x ** 2
    return x
```

使用lambda表达式只需要写成：square_x=lambda　　x:x**2。输入square_x（2）返回结果为4。

示例二　多个参数复合运算，普通的函数可能这样定义：

```
def mul_var(x,y):
    result = x+y+x**2+y-3
    return result
```

使用lambda表达式只需要写成：mul_var=lambda　x，y:x+y+x**2+y-3。输入mul_var（2，3）返回结果为9。

示例三　条件判断计算，普通的函数可能这样定义：

```
def filter_function(x,y):
    if x >1 and y < 10:
        return x+y
    else:
        return x-y
```

使用lambda表达式只需要写成：filter_function=lambda　x，y:x+y　if x>1 and y<10 else x-y。输入filter_function（1，7）返回结果为-6。这里需要注意if..else的语法和def定义的函数不同。

有关lambda使用的几个注意点：

·不要使用lambda定义复杂的功能（例如嵌套），虽然它也能实现。

·lambda表达式虽然精简，但使用如果包含过多条件，会降低代码的可读性和可理解性。

·lambda定义的函数是一次性的，不能重复使用。

·不是所有的def功能都可以使用lambda重写，例如，在Python2.7.*中无法在表达式中使用print方法。

·lambda默认已经包含return功能，如果再增加return方法会报错。

相关知识点：使用strftime将字符串解析为日期格式

Pandas中的datetime库是处理与日期相关主要功能库，其中的strftime用来将字符串解析为日期格式，与strptime方法对应的是strftime方法，用来将日期转换为字符串。关于日期的格式，Pandas采用的是Python默认的日期格式，完整见https://docs.python.org/2/library/datetime.html#strftime-and-strptime-behavior。常用的日期格式字符串如表4-6所示。

表4-6　日期字符串

字 符 串	日 期 格 式	示 例
%y	2 个数字表示的年份	99
%Y	4 个数字表示的年份	1999
%b	月份的简写	Jan
%B	月份的全写	January
%m	月份（[01,12]）	10
%a	星期的简写	Mon
%A	星期的全写	Monday
%w	周几（[0, 6]）0 表示周一，以此类推	5
%U	当年的第几周，星期日作为周的第一天（[00,53]）	30
%W	当年的第几周，星期一作为周的第一天（[00,53]）	30
%d	日是这个月的第几天，范围 [01,31]	12
%p	AM 或 PM	AM
%H	小时（24 小时制，[00, 23]）	20
%I	小时（12 小时制，[01, 12]）	12
%j	日在年中的天数 [001,366]	360
%M	分钟（[00,59]）	59
%S	秒（[00,59]）	59
%f	微秒（[000000,999999]）	0

在使用Pandas的read_table读取数据文件时，通过index_col指定date列为索引列，代替默认的数值索引；通过date_parser参数指定刚才定义的日期解析功能项，由于date_parser默认的日期解析格式为YYYY-MM-DD HH:MM:SS，跟源数据格式不符，因此需要人工自定义格式。

最后应用astype方法将时间序列项的格式类型转换为float32，并通过describe方法输出时间序列概况，结果如下：

```
data summary
count    149.000000
mean     164.382553
std       75.097740
min       47.000000
25%      100.000000
50%      156.000000
75%      201.000000
max      400.000000
Name: number, dtype: float64
```

describe方法输出的数据包括数据记录数、均值、标准差、最小值、最大值、25%、50%和75%分位数。

步骤2　原始数据检验

这里直接调用已经定义好的稳定性检验和白噪声检验函数，并传入原始时间序列数据，返回结果如图4-11所示。

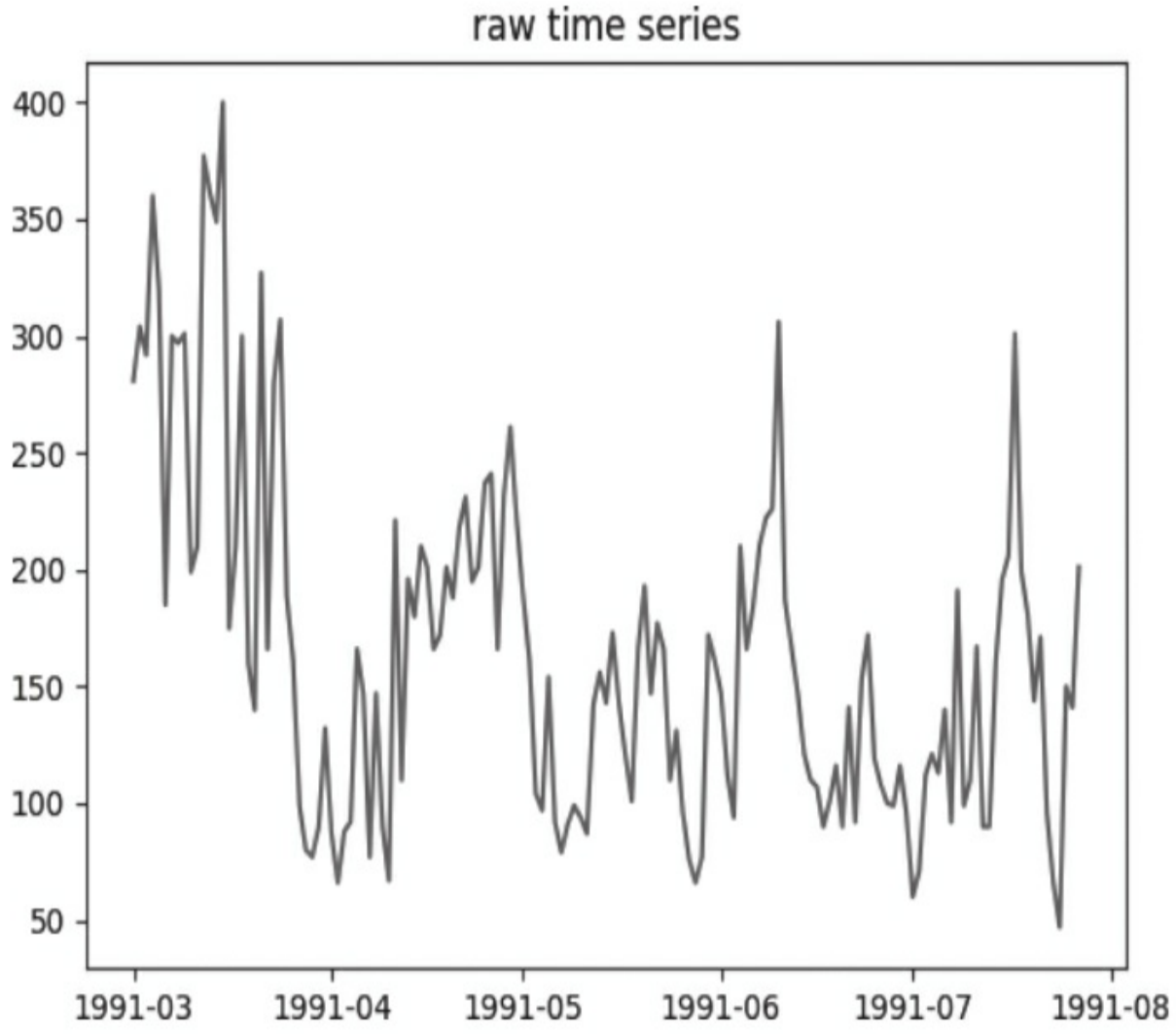


图4-11　原始时间序列图

由原始时间序列图发现，4月份之前的数据波动较为明显，但4月份之后的数据基本处于平稳波动的状态。

在ACF图中通过配置参数lags=20（20个滞后点）来更清晰地展示数据的分布，前20个滞后点基本已经可以看出明显趋势。

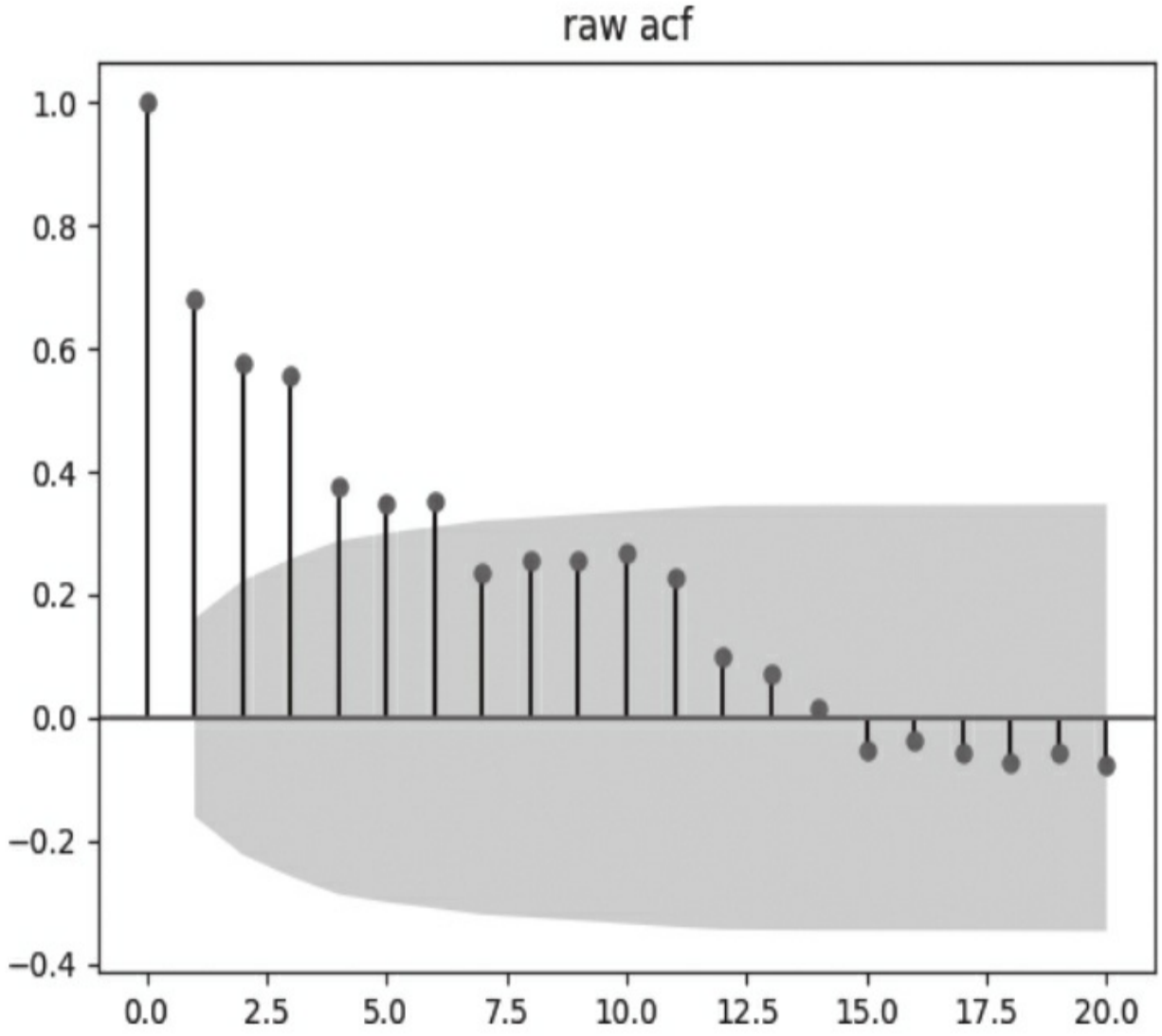


图4-12　原始ACF检验图

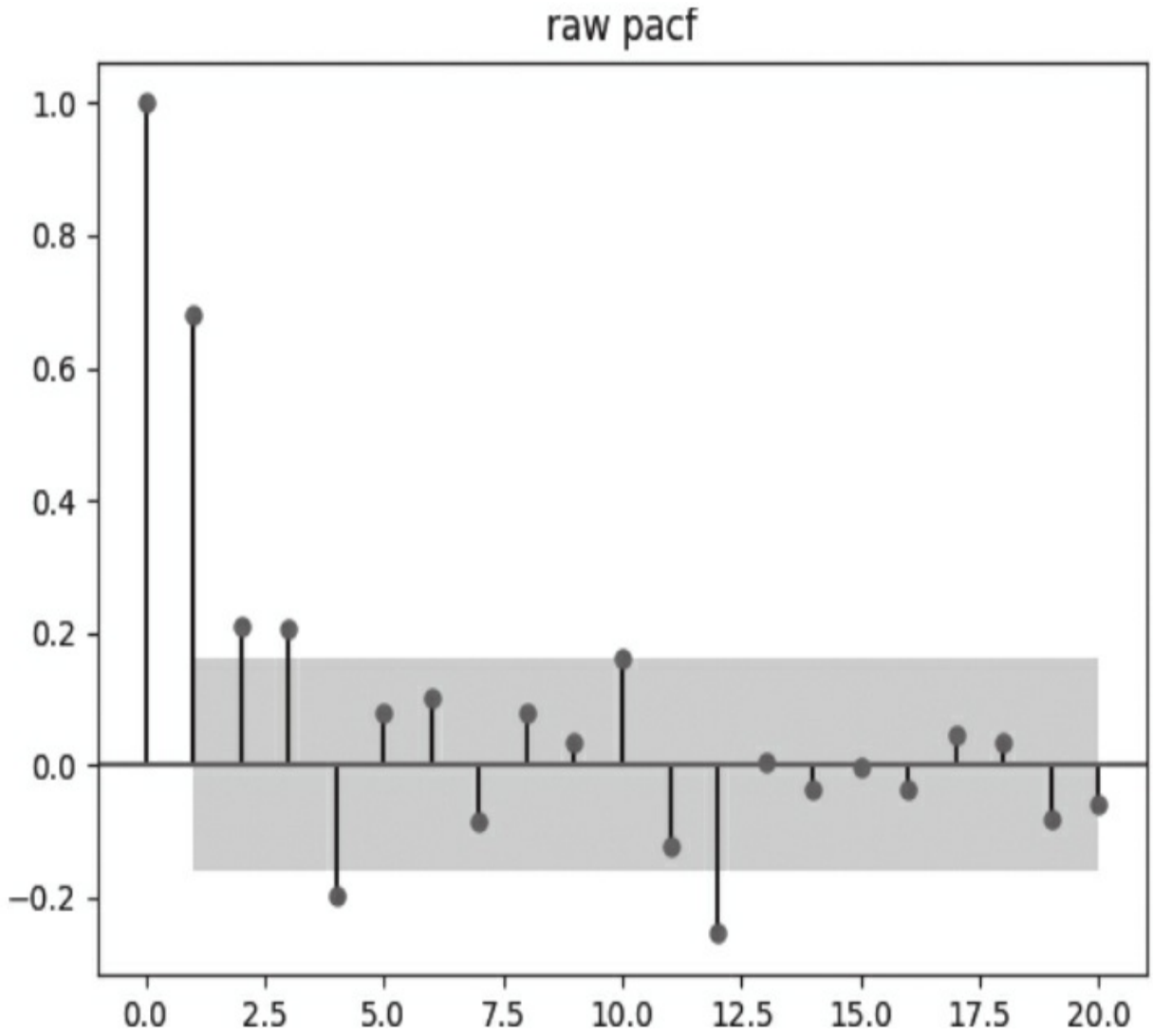


图4-13　原始PACF检验图

ADF稳定性检验结果：stochastic score。

```
+----------------+-----------------+---------+--------+
----------------+--
|      adf       |       pvalue    | usedlag |   nobs |
 critical_values                              | icbest |
+----------------+-----------------+---------+--------+
---------------+--
| -3.76427781964 | 0.0032946903803 |    11   | 137  |
 {'5%': -2.8828782366015093, '1%': -3.4790073553689438, '10%': -2.5781488587
 1405.38466046 |
+----------------+-----------------+---------+------+
---------------+--
```

白噪声检验结果：stationarity score

```
+----------------+--------------------+
|    lbvalue     |       pvalue       |
+----------------+--------------------+
| [ 70.22298485] | [  5.29656862e-17] |
+----------------+--------------------+
```

如何通过上述指标分析数据稳定且随机分布？如果是时间序列具有稳定性，那么ADF的值（adf）需要小于critical_values中的1%、5%和10%的三个值，且P值（pvalue）一般小于0.01。在白噪声检验结果中，P值需要小于0.05即说明数据是随机分布的。从数据结果看，示例数据恰好能满足条件的定义。

输出的ACF和PACF图更多的用于辅助判断p和q的阶数，通过分析截尾和拖尾的状态来人工判断p和q的值。这里不使用这种方法。

步骤3　创建用于区分是否进行平稳性处理的规则。该规则使用调用步骤2中返回的结果，在条件的定义中，比较关键的是ADF的P值和白噪声检验的P值的阈值定义，这两个值的定义标准在步骤2的返回结果中已经给出。

步骤4　对时间序列做稳定性处理。在该步骤中，如果数据符合平稳性，那么不需要做平稳性处理；否则需要处理直到同时满足2个条件的阈值要求。由于示例数据满足稳定性和白噪声检验，因此直接返回0和原始数据集。

步骤5　再次做检验，包括平稳性和白噪声检验。由于没有做任何处理，返回的结果仍然与步骤2相同。

步骤6　训练最佳ARMA模型并输出相关参数和对象。在该步骤中，命令中交互窗口会有警告信息，忽略该信息即可。该过程将需要一点时间来完成。部分输出如下结果：

```
each p/q traning record
        p     q           aic           bic          hqic
0     0.0   0.0  1712.839862  1718.847755  1715.280770
1     0.0   1.0  1653.446702  1662.458541  1657.108063
2     0.0   2.0  1648.868523  1660.884308  1653.750338
3     0.0   3.0  1618.676555  1633.696287  1624.778824
4     0.0   4.0  1619.720071  1637.743748  1627.042793
5     0.0   5.0  1619.720071  1637.743748  1627.042793
```

```
6     0.0   6.0  1601.914727  1625.946297  1611.678357
...
223  14.0  13.0  1430.080724  1514.333692  1464.318768
224  14.0  14.0  1430.080724  1514.333692  1464.318768
[225 rows x 5 columns]
```

该数据是每次训练时取不同p和q计算对应的AIC、BIC和HQIC的值，可通过该数据分析不同训练的结果。

```
best p and q
+----+---+---------------+---------------+---------------+
| p  | q |      aic      |      bic      |      hqic     |
+----+---+---------------+---------------+---------------+
| 14 | 1 | 1467.49034744 | 1467.49034744 | 1467.49034744 |
+----+---+---------------+---------------+---------------+
```

ARMA最优模型下的p和q的值，满足BIC最小的原则。

步骤7　模型训练和效果评估。通过最优的ARMA模型对训练集做时间序列内的预测，并与原始时间序列数据做对比，输入预测值与实际值的时间序列对比趋势图以及RMSE输出预测结果。

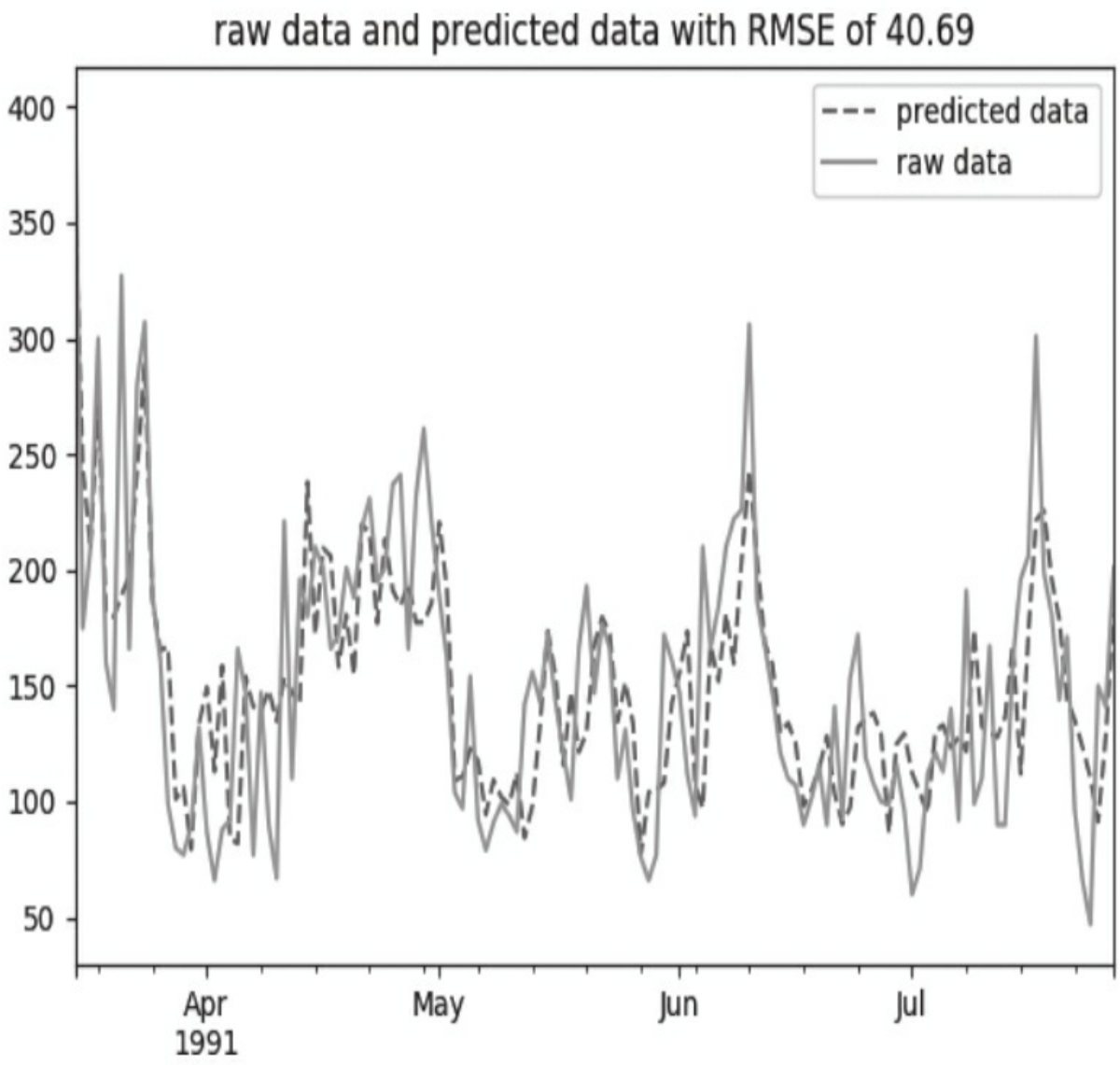


图4-14　ARMA预测结果与实际结果对比

从对比图中发现，预测值与实际值的数据对比差异不大，并且趋势分布能比较好地还原实际数据分布规律。

步骤8　模型预测应用。这里设置了预测起止时间分别是1991-07-28、1991-08-02，调用predict_data做预测分析。得到结果如下：

```
----------predict data----------
1991-07-28    183.093636
1991-07-29    125.896794
1991-07-30    135.165396
1991-07-31    143.513278
1991-08-01    111.555943
```

```
1991-08-02    113.530398
Freq: D, dtype: float64
```

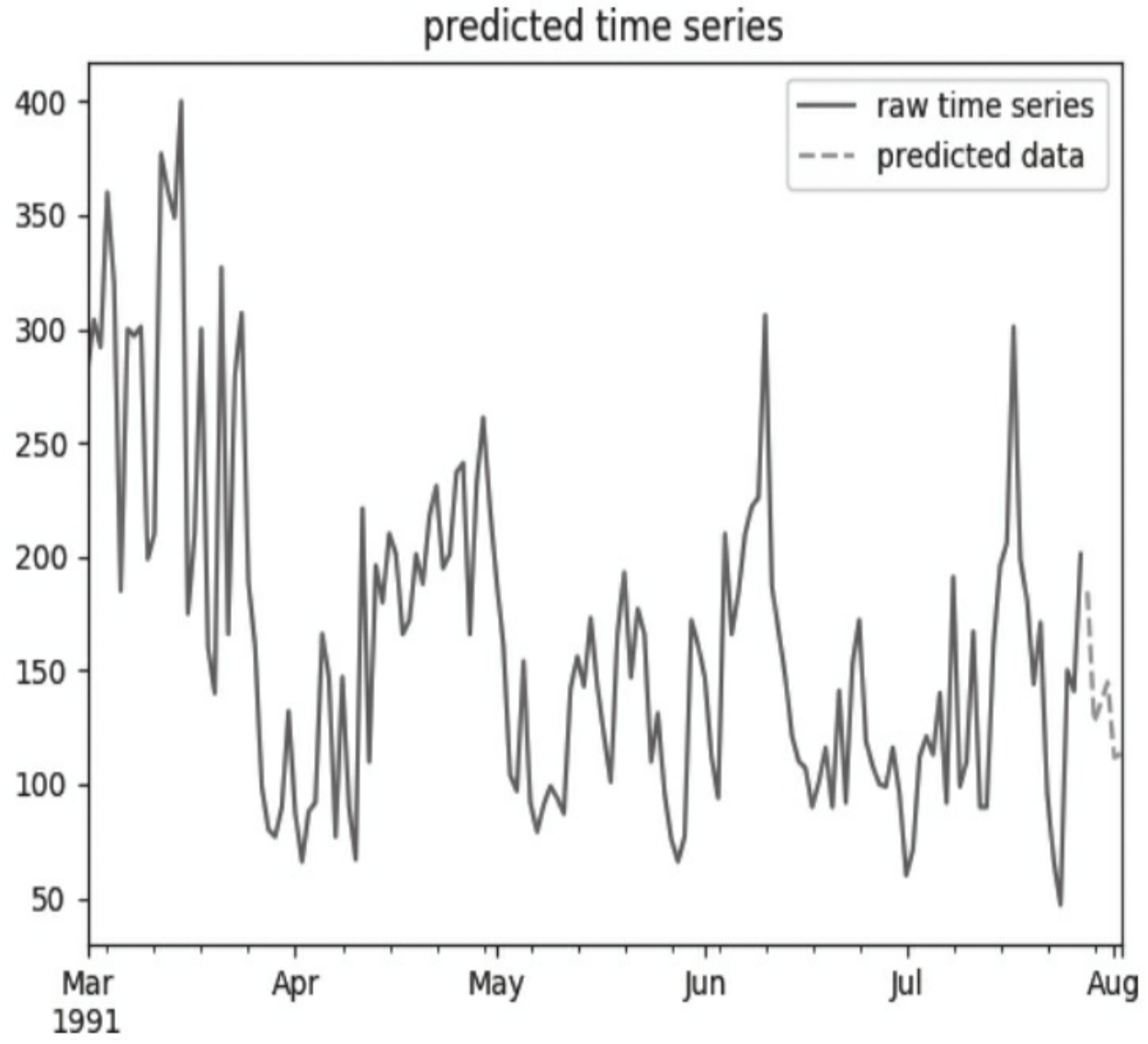


图4-15　ARMA预测结果与实际结果合成

上面输出的数据表格和趋势图，显示了未来预测的结果和趋势，整个图像趋势没有明显的异常，较为符合实际情况。

上述过程中，主要需要考虑的关键点是：

·如何做数据平稳性处理。

·如何通过ACF和PACF图做p和q阶数评估（当然采用AIC、BIC是一

种自动化的方法）。

·在做数据平稳性处理之后，注意还原数据，否则预测数据将难以应用。

·ARMA模型在做最优模型训练后，可直接返回模型对象给其他程序调用，而无需重复训练。

代码实操小结：在本节的代码中，主要使用了statsmodels的相关库和方法做时间序列的分析处理和预测应用，知识点如下：

·通过def函数式的方法定义不同的功能模块实现不同的功能，函数中包含参数、功能和返回信息。

·为函数写Doc String。

·使用prettytable库中的add_row、field_names等方法创建格式化表格并输出。

·通过for循环结合多条件语句做循环处理。

·使用Numpy的log、exp、sum等方法做基本数据处理。

·使用statsmodels库里面的plot_acf、plot_pacf做acf和pacf图形检验分析，adfuller、acorr_ljungbox做adf检验和随机性检验。

·使用Matplolib做折线图输出，并设置不同的线条样式。

·使用ACI、BIC等方法做最优ARMA模型p和q阶数的选择。

·使用自定义方法计算RMSE。

·使用ARMA的fit方法做模型训练，predict方法做时间序列预测。

·使用lambda表达式实现基本功能处理。

·使用pandas.datetime中的strptime将字符串转换为日期格式。

·通过Pandas的read_csv方法读取数据文件并指定分隔符、索引列和日期解析功能对象。

·使用Pandas中的describe方法输出数据框的基本概述信息。

4.7 路径、漏斗、归因和热力图分析

路径分析、漏斗分析、归因分析和热力图分析原本是网站数据分析的常用分析方法，但随着认知计算、机器学习、深度学习等方法的应用，原本很难衡量的线下用户行为正在被识别、分析、关联、打通，使得这些方法也可以应用到线下客户行为和转化分析。

1.漏斗分析

漏斗分析是网站分析的基本方法，很多强大的工具支持全站页面、事件、目标之间的混合漏斗分析，通过漏斗查看特定目标的完成和流失情况。根据漏斗的封闭性可分为封闭型漏斗和开放型漏斗。

封闭型漏斗指漏斗从第一环节开始后最后的环节，数据从上一环节开始依次“漏”下来，不存在其他进入途径。典型的封闭型漏斗是购物车流程，通常情况下从加入购物车开始，用户依次进入结算和提交订单，由此形成加入购物车→结算→提交订单的完整闭环，该过程中不可能从其他环节直接进入。

开放型漏斗指漏斗的各个环节都有可能存在其他入口，整个漏斗不封闭。典型的开放型漏斗是全站购物流程漏斗，通常该漏斗是到达着陆页→查看产品页→加入购物车。在整个过程中，用户查看产品页和加入购物车可能从任何一个具备该功能的入口进入，而不一定是从着陆页开始。

漏斗分析的典型应用场景是分析站内流程，如注册流程、购物车流程等；除了可以做针对多页面的流程分析外，还可以做单页面的多个步骤分析，如表单分析、注册分析等。

2.路径分析

路径分析也是网站分析的基本方法，借助于网站数据的可跟踪和可监测特征，所有用户行为都处于可分析的状态。路径分析不仅可以基于页面产生，还可以基于目标路径、事件路径等数据主体产生。

页面路径常用于分析不同页面引流和前后路径关系，如用户从活动

页落地后如何分流、典型客户的路径特征、客户网站访问动线、页面广告资源挖掘、站内多页面流程设计优化等。大多数网站分析系统只能提供基于流量（通常是PV）的单维度路径，有些强大的分析系统或插件能实现三维路径分析。这些分析可以为站内流程优化、流量引导和分配等提供决策建议。典型应用包括：

·活动主会场/网站主页面如何导流？

·用户是否按照“预期”流程行动？

·购买“手机的用户”的浏览习惯是怎样的？

·渠道A集中访问了某条路径，是否是“恶意流量”？

3.归因分析

归因分析很多时候也叫订单转化归因或归因模型，主要用于评估多个参与转化的主体如何分配贡献大小。出现归因的基本条件是某些转化没有特定的归属，因此无法直接判断到底是由哪些因素产生。

以订单转化为例，归因分析用来衡量在用户从第一次进入网站到最后一次进入网站成单时，所有来源渠道对订单的贡献作用。传统的网站分析工具把订单归因为最后一次来源渠道（在此不考虑渠道覆盖规则），但这种订单归因的分配模式忽视了其他渠道对于该订单的“转化支持”作用。在实际运营业务中，SEM品牌词流量、直接输入流量、网址导航直接进入网站的流量质量都非常高，原因是用户认知度、认可度和忠诚度比较高。但如果因此只投放这些“收口”渠道而忽视其他渠道，这些“收口”渠道效果是否还能持续？

除了传统的归因于最后进入的渠道的方法外，还有其他的归因方法：归因于最初进入的渠道、线性平均归因、随时间衰减归因、根据位置的综合归因等。

下面以一个实例说明不同归因模型下各个渠道的订单贡献情况。案例说明：用户打算购买在某网站购买商品，第一天从Sina Banner进入浏览了该网站某个活动；第二天在微博上看到该活动的推广博文点击进入网站，并详细看了其中某个活动单品；第三天该用户在搜索引擎中搜索

了该单品，并点击进入该网站继续查看；第四天用户在其他网站看到有该网站的合作推广单品，点击进入该网站但仍未成单；第五天该用户最终搜索品牌关键字，点击品牌区进入网站完成订单。用户整个订单周期内访问路径如图4-16所示。

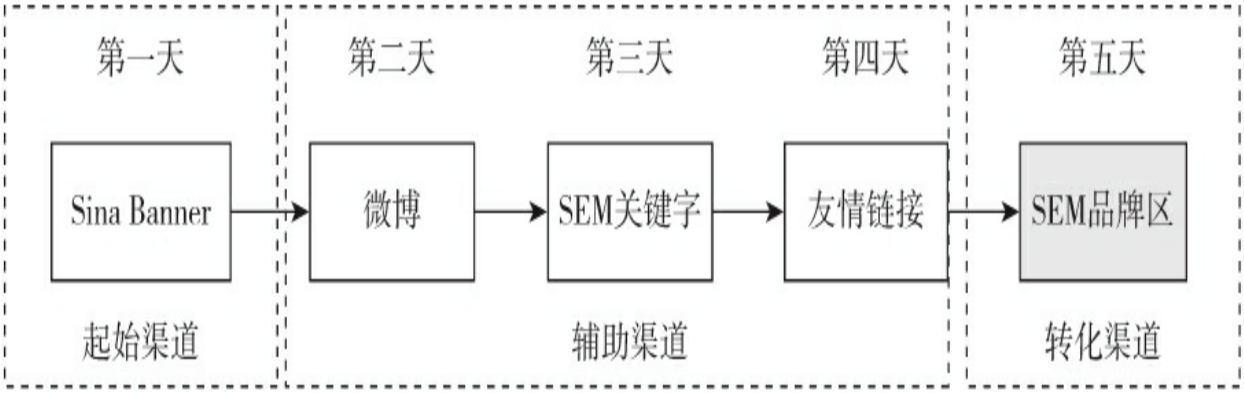


图4-16　用户整个订单周期内访问路径

不同的订单归因模型，各个渠道贡献如下：

·归因于最后进入的渠道：最终互动模型将100%的转化价值归功于客户在进行购买或转化之前与之互动的最后一个渠道。案例中，SEM品牌区订单贡献为100%，其他渠道订单贡献为0。适用场合：广告和推广活动的目的是在购买时吸引用户，或者企业业务主要参与的销售周期不涉及观望阶段。

·归因于最初进入的渠道：将100%的转化价值归功于客户与之互动的第一个渠道。案例中，Sina　Banner订单贡献为100%，其他渠道订单贡献为0。适用场景：广告或推广旨在建立最初的认知度，品牌并不为人熟知，企业前期的推广重点放在品牌曝光下，那么首次进行品牌展示的媒介是重点关注媒介。

·线性平均归因：将功劳平均分配给转化路径中的每个渠道。案例中，每个渠道的订单贡献都是25%。适用场景：如果广告或推广活动的目的是在整个销售周期内保持与客户的联系并维持品牌的认知度，则适合使用此模型。在这种情况下，每个接触点在客户考虑的过程中都同等重要。

·随时间衰减归因：如果销售周期中涉及的考虑阶段较短，那么更

适合时间衰减模式，该模型向最接近转化发生时间的互动分配最多的功劳。案例中，不同渠道的订单贡献作用与其最后接触的时间相关，渠道位置离订单转化越近，订单贡献作用越大，因此各个渠道的订单贡献作用依次为：SEM品牌区>友情链接>SEM关键字>微博>Sina Banner。适用场合：如果投放短期的促销广告活动，可能希望将更多的功劳分配给促销期间的产生互动的媒介，在这种情况下，与接近转化的接触点相比，一周之前发生的互动只有很少的价值。通常企业大促销情况下这种模型较为合适，“时间衰减”模式能够适当地将功劳分配给促成转化前一两天的接触点。

·根据位置的综合归因：根据位置综合归因的模式结合了以上全部模型因素，根据不同渠道在整个订单周期内位置进行权重分配。案例中不同渠道的订单贡献根据设置而定，Google Analytics将权重划分为最终进入渠道、中间辅助渠道、最终转化渠道三类，Webtrekk将渠道归因细分到五种位置：第一进入渠道、第二渠道、中间渠道、倒数第二渠道、最后渠道。对位置的定义越详细，可以细分的维度和视角越多。

由此可见，不同的归因模型下，不同渠道对于订单的贡献评估结果是不同的。

4.热力图分析

热力图分析是网站分析的重要方法，该方法的主要作用是分析单个页面内的点击分布热力图是单页面用户体验分析的重要途径，通过热力图可以直观反映用户对于页面内容喜好程度。热力图可分为基于链接的热力图和基于像素的热力图。

基于链接的热力图反映了页面内每个链接的点击情况，这种热力图更容易用数据的形式分析页面不用功能间的点击分布，是一种理性的分析方法。链接热力图的问题在于如果页面内不同位置存在相同链接，则会导致同一链接在不同位置的点击数据归为一致，比如页面顶部和底部都有一个指向首页的链接，两个链接的数据相同。

基于像素的热力图反映了页面内每个点击位置（任何位置而不限于链接）的点击情况，像素热力图相较于链接热力图更容易发现在非链接位置上的用户点击习惯，如用户习惯性的点击位置、特殊页面位置喜好等。像素热力图对于新广告资源挖掘、用户喜好和习惯分析、网页功能

分析具有重要指导意义；但其缺点是只能“感性”地看图分析，没有直接的数据支撑，并且热力图无法反应页面内容的变化，无法分辨同一位置不同内容的点击数据。

热力图分析常用于提取单页面内重要访问特征，如点击集中度、功能使用率等情况，通过发现用户集中点击区域为业务行动指明方向。

4.7.1 不要轻易相信用户的页面访问路径

通过页面间的路径数据图可以看到用户从某个页面开始到其他页面的流量分布；同样，也可以分析从某个页面开始，其之前的流量都来自于哪些页面或路径。但是用户真的是按照这种方式浏览页面的吗？很多时候在路径结果数据中会出现这样的问题：页面下一级路径中出现了该页面上没有的点击链接，例如链接C只存在于B页面上，但发现很多用户从A页面进入了C页面。

页面路径的识别标志不是以页面是否包含跳转链接为判定依据的，而是根据不同页面的时间戳确定的。这种判定方法准确讲其实是时间路径，而非页面路径。例如，2个不同的tab标签，分别各自打开一个页面，这2个页面会成为上下游页面，虽然这2个页面可能没有任何链接关系。

示例：用户在10:00打开了A页面，然后点击A页面的一个链接并在浏览器的新标签卡中打开B页面，接着用户点击标签卡返回A页面，又点击了A页面中的链接到达C页面，此时形成的路径是A→B→C。该报告显示了用户是依次从A到达B再到达C页面，但实际上C页面是通过A点击点击进入，而非B页面点击进入的。因此，路径分析在这种场景下就会失效。

4.7.2 如何将路径应用于更多用户行为模式的挖掘？

除了将路径应用于分析页面访问行为，还可以用于站外广告渠道路径分析、用户关键字搜索路径分析等。

1.将路径分析用于站外广告渠道分析

通过站外广告渠道路径分析不仅可以分析用户是从哪个渠道来的，更可以分析用户在该渠道之前是从哪个渠道来的，之后又会从哪个渠道进入网站，在最终形成的转化路径中，到底哪些路径是用户最常使用的“转化偏好路径”。例如有的用户习惯于从A→B进入网站成单，有的用户习惯于A→C→B成单，还有的用户习惯于A→A→A→B成单。不同的渠道进入模式决定了在广告运营的各个方面存在特定的优化点，该信息可用于以下业务场景：

·整合营销传播是否适用于企业推广。如果数据发现大部分用户只通过一个渠道进入网站就可以成单，而对其他渠道的交叉访问依赖关系较弱，是否间接说明渠道整合程度可能较低或目前不具有太高的重要性。

·拓展渠道评估的视野。在做渠道效果评估时，除了评估渠道的CPC、ROI、转化率、订单成本等指标外，还需要综合分析该渠道对其他渠道的“辅助”引流和订单贡献作用。

·渠道落地细节优化。在转化路径中，不同推广渠道是否具有明显的位置特点或集中趋势？如展示类、广告类渠道明显处于用户转化路径的前期，距离订单转化点较远；SEM和直接输入明显处于转化的末端，距离订单转化点较近。在广告投放时，是否可以综合考虑各个渠道的媒体排期？针对不同渠道间的交叉访问，是否存在流量交叉覆盖或需要关联投放？整个广告活动的前、中、后期推广策略，如何根据路径的渠道顺序、转化时间进行优化？

2.将路径分析用于用户关键字搜索分析

当用户在网站内通过搜索引擎多次搜索相关内容时，基于不同的搜索词会产生搜索关键字路径信息。该信息显示了用户在不同情况下先后

搜索的关键字路径，这些信息可以应用到业务关键字调整策略中。例如：

·某个关键字虽然长期参与用户访问路径，但对订单的转化效果贡献不大，此时可以考虑该关键字的投入产出比的调整；

·路径越长意味着关键字点击次数越多，同样或相近水平的回报收入，必然选择路径渠道少即点击次数少就完成订单的关键字路径；

·结合关键字路径转化周期，在关键字调优时提供时间上下线参考依据；

·转化路径可以基于关键字、广告组或广告计划进行分析，该模型可以综合评估每个关键字、广告组、广告计划在订单转化过程中的作用，前端引流、中间支持还是后端收口，在费用控制下可以优先投放具有收口作用的关键字；

·某个广告活动投放周期可以根据用户访问行为进行判断，除了参考促销活动周期、用户转化周期外，不同路径长度下的转化周期也会纳入考量范围。

4.7.3 为什么很多数据都显示多渠道路径的价值很小？

在做多渠道路径分析时，经常会发现用户真正通过多个渠道交叉进入网站并完成转化的路径占比很小，似乎数据显示了用户并没有那么多的跨渠道访问后完成转化的习惯。在企业实际数据表现中，这并不是一个特例。

根据笔者接触到的日均UV在100万以上的几家企业，涵盖了快消、家电、保险等行业，发现该规律普遍成立。为什么多渠道整合传播对于用户转化的意义如此有限，或者说整合传播的意义没有传统宣传中的那么明显？主要因素包括以下几个方面：

·Cookie删除机制。虽然比例较小，但是删除Cookie仍然意味着渠道路径信息无法关联。

·媒体碎片化严重。多渠道访问转化意味着用户需要在不同渠道间重复访问，即用户分别访问了A、B之后，才能形成由A到B的路径，而媒体碎片化导致这种重合概率变小。

·企业投放媒体较少。投放媒体较少会直接导致数据样本量不够，无法覆盖用户重复访问的渠道，因此无法得出有意义的结论。

·用户忠诚度和访问习惯。高质量的媒体用户忠诚度较高，这些入口习惯会让这些渠道成为对转化最具有价值的投放媒体，因此用户不会通过访问多个媒体后完成转化，而仅仅访问一个高质量的媒体便能形成转换。

·跨设备追踪、跨平台追踪问题。之前用户跨设备、跨平台访问无法关联，导致用户多设备、多平台间的关联失效，例如，用户在手机上看到广告和从PC端完成购买是无法关联的。即使是目前基于用户真实注册或登录ID的跨设备、平台追踪也无法从本质上关联匿名用户信息。当用户的目的只是浏览信息时，一般是不会特意登录或注册的，而没有登录或注册唯一ID就无法关联到所有设备或平台。

4.7.4 点击热力图真的反映了用户的点击喜好？

当用户在网站上点击时，无论页面内是否有链接，都会被网站跟踪分析工具捕获点击的位置（水平和垂直坐标），然后基于每个像素的点击形成页面点击热力图。这些点击通常都会集中到有内容交互的页面功能，例如表单、按钮、播放器、链接等元素上。

但是，点击热力图有时并不是基于交互元素产生，在很多空白的内容区域也会产生大量的用户点击。这些点击通常不是因为用户真的喜欢某些功能而点击，而是习惯性动作才导致了很多非内容或功能区域点击聚集。

以图4-17为例，点击热力图显示了用户在填写表单时的点击焦点。通过像素热力图分析发现，首页游戏充值表单的右侧（线框区域）点击热度很高，但该区域并没有如此多的功能交互项，因此不可能发生基于特定的内容或兴趣产生的点击。这些位置很大可能是用户习惯性的点击右侧，以完成表单选择或作为滚动滑轮的前置动作。

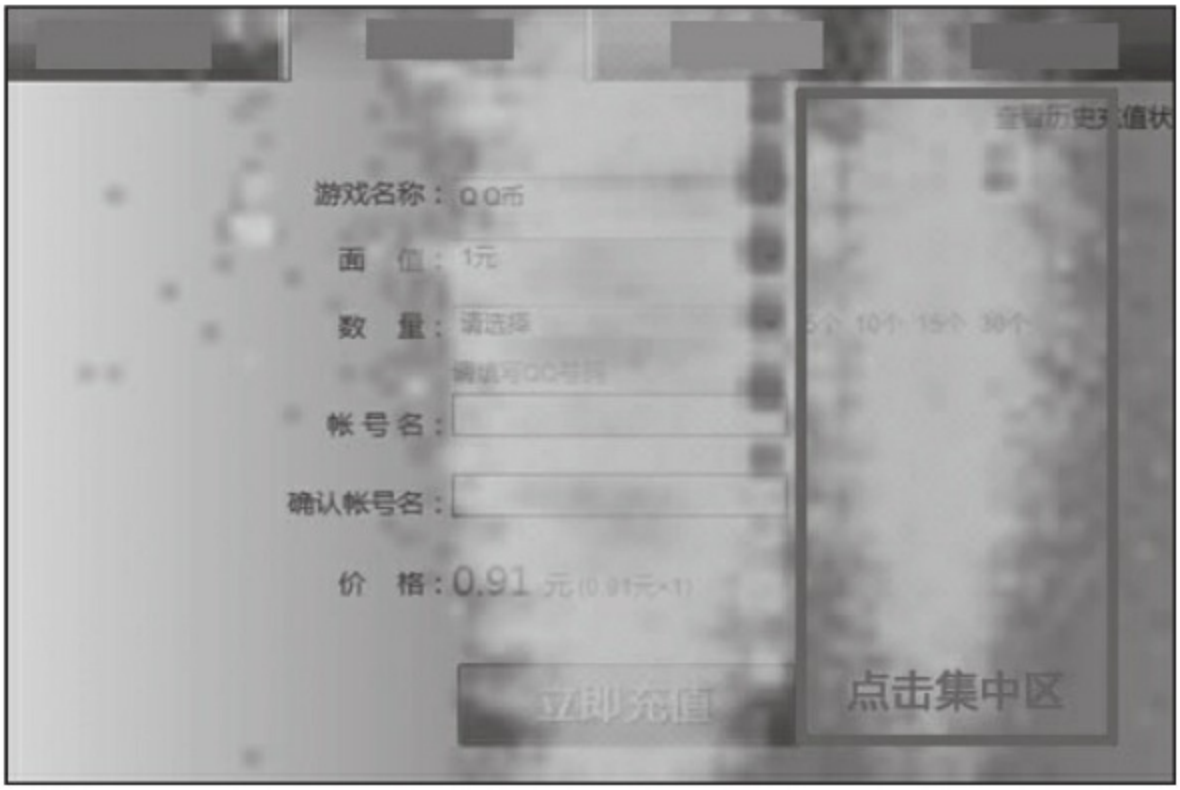


图4-17　用户点击热力图

4.7.5 为什么归因分析主要存在于线上的转化行为

归因分析产生于线上网站数据分析主题中，主要原因是在于线上转化行为的归属模糊性。以订单转化归因为例，在线下实体销售网络中，商品都隶属于某个分公司、分销商或门店，具有明显的归属关系。当商品售卖出去之后，可以通过商品归属分析商品是通过哪些门店、促销展柜等卖出去的。

但是，线上的商品却没有明显属于哪个“门店”的逻辑，相对于线下门店或商品陈列的位置，线上对应了多种商品展示位置，例如首页、列表页、超市页、活动页等，所有的位置都可以展示并允许用户点击甚至购买商品。当用户从某个位置点击商品之后，还可能会通过其他位置再次点击相同的商品，此时如果发生订单转化，就很难判断到底是哪个位置导致了用户的订单转化行为或卖出了商品。以下两种场景显示了转化归因的难点：

场景1　用户点击了多个商品陈列位置并购买了其中的商品。

假如商品P1在首页、超市页和列表页分别有K1、K2、K3三个商品展位。用户依次进入这3个页面并点击了K1、K2、K3这三个坑位后最终购买了该商品。此时，用户由于点击了3个坑位，无法判断到底哪个坑位产生的转化贡献最显著，更无法判断到底是那个展位卖出的商品。

场景2　用户点击了多个商品陈列位置但购买了其他商品。

假如商品P1在首页有一个展示坑位K1，当用户点击该商品P1后到达商品详情页；此时用户发现商品详情页上有关联推荐商品P2便进行了点击，在P2页面用户又发现了P3商品并到达商品P3的详情页，最终用户下单购买了P3。此时，用户从首页坑位K1到达P1页面，然后到达P2页面，最后到达P3页面完成购买。虽然用户开始点击的商品跟最终购买的商品不同，但不可否认的是，前两次点击对于用户购买P3确实也产生了重要作用，如果没有前两步的“引荐”作用，用户可能无法发现P3更不可能购买。

归因分析的出现恰好用来解决线上用户复杂的访问和购买场景下，如何分配所有参与转化的主体如何分配“贡献度”的问题。

4.7.6 漏斗分析和路径分析有什么区别

漏斗分析和路径分析都是用来衡量有关多个主体的流程性、序列性的关系，因此二者很容易混淆，但这两种分析方法具有很大的差异性：

·分析目标不同。漏斗分析用于测量有特定完成目标的场景，例如购物车转化、注册转化等；而路径分析侧重点分析开放性的流程，通常没有特定完成的目标。

·不同环节的关系不同。漏斗分析只能看到从上一级节点到下一级节点的转化关系以及到外部的流失节点；但是路径分析可以形象的展示某个节点跟其他所有节点之间的关系而不限定于是否转化或流失。

·衡量主体之间的逻辑不同。漏斗分析的前后不同序列节点之后，往往带有明显的转化关系，通常会用从上一级到下一级还“剩”多少来表示这种关系；但路径分析的不同层级和节点之间是一种平行的关系，不存在明显的前后序列或转化关系，并且路径分析的不同主体之间还可以相互转换互相成为双向上下游，而漏斗分析只能是单向的上下游关系。

·表示结果不同。漏斗分析通常用完成率、流失率等指标评估，而路径则使用事件的比例来评估，例如页面浏览量占比、访问量占比等。

·应用主体不同。漏斗分析可以用来分析不同类型的数据之间的转化关系，例如事件、页面、行为，例如停留超过30秒→浏览首页→点击加入购物车→提交订单；但是路径分析通常只用来分析相同维度下的主体，例如页面路径、广告渠道路径、关键字路径等，很少会做交叉类的分析。

有关网站分析的具体分析实践，主流的方法是通过网站工具来实现，而非使用R或Python。很多公司内部也有使用Python、R基于日志来做网站分析，甚至自己开发网站分析工具。但几乎所有公司内部自己开发的具有网站数据分析功能的工具，其分析能力、实时性、可用性、灵活性等都无法与免费的Google Analytics相提并论，更不用说与Webtrekk、Adobe Analytics等商用分析工具比较。Python等开源工具可以用来做网站分析，但并不是大多数场景下网站分析的最优选择。本节

提到的漏斗、路径、归因、热力图，以及其他更深入的多维下钻、转化和自定义跟踪配置、实时数据分析等都是如此。有关Python应用到网站数据分析的更多话题，会在后面第7章讲到。

4.8 其他数据分析和挖掘的忠告

4.8.1 不要忘记数据质量的验证

数据质量是所有数据工作中最基础但也是最容易被忽视的一个环节，以下是几个简单的数据质量较差的场景：

·三个数据系统中同样定义的“销售额”指标数据不一致。

·数据库中“邮箱”字段80%为空。

·数据库中“性别”字段中某些值为10。

·数据库中“产品名称”字段出现乱码。

以上问题反映的是数据质量差的现象，导致这些问题的既有公司内部原因又有外部原因。内部原因包括数据采集方式错误、数据验证机制不全面、数据同步不及时、ETL过程错误、数据提取错误等。外部原因包括用户填写信息不规范、用户数据采集环境客观差异等。

要有效应对数据质量的问题，必须在工作过程中进行数据质量验证。什么是数据质量验证？

·理解数据来源、数据统计和收集逻辑，数据入库处理逻辑；

·理解数据在数据仓库中存放细节，包括字段类型、小数点位数、取值范围，规则约束等；

·明确数据的取数逻辑，尤其是过程中是否对数据有转换或重新定义；

·第一时间对数据做数据审查，包括数据有效性验证、取值范围、空值和异常值验证，是否与原始数据原则一致等。

这些工作完成之后才是数据分析。但可惜的是大多数数据分析师都不关注数据质量问题，甚至对数据的理解仅限于看到数据的字面意义。

4.8.2 不要忽视数据的落地性

无论数据分析的服务对象是具有决策权的领导层还是执行权的业务层，数据的价值都只存在于辅助决策或者数据驱动中。但部分数据分析师的数据报告却让业务方觉得没有价值，表现为：

·分析过程明显不符合业务操作实际；

·结论明显是错的；

·建议方向性很对，但都是人人都知道的大道理，具体执行缺乏落地点；

·建议方向性很明确，也有具体执行建议，但是业务不能执行。

在以上问题中，前两条问题的原因是数据分析的基本数据能力和业务基本常识不足，这是一定需要避免的问题；而后两条问题的原因更多的在于信息不对称。从数据分析师的主观问题分析，根源可能有以下几方面：

·数据分析师不懂业务操作流程，凭自己的理解去猜测业务流程；

·数据分析师不了解目前业务的困难点和紧迫点，想要驱动的是业务的“次要”关注点；

·数据分析师不了解业务的实际能力与权限，尤其是公司大环境下的实施制约因素。

4.8.3 不要把数据陈列当作数据结论

把数据陈列当作数据结论是指数据报告中的结论全部都是由数字组成的简单陈述，通俗点讲就是“读数”。这种问题常见于日常报告，如日报、周报、月报等常规性报告，报告内容以阶段性总结和汇总为主，报告中没有深度分析的内容。

将报告中的数据简单陈列出来的情况通常称为数据事实，数据事实与数据结论的区别在于：数据事实是将数据陈列，不涉及好、坏、优、劣的定性；而数据结论需要将数据事实结合业务目标和实际情况定性为好、坏、优、劣等。数据事实与数据结论的联系在于：数据事实和数据结论是日常总结性报告中不可缺少的两个部分，前者以数据的形式直接反映结果，后者从数据分析的角度定性该结果并阐述了该结果的影响。

举例：表4-7是一份公司日报内容的一部分。

表4-7　某日报数据

日期	订单量	商品销售量	重复购买率	新客户比例	人均商品数量
2014-04-22	8745	33 101	62	46	3.8
2014-04-21	6570	21 438	43	39	3.3
环比 %	33%	54%	42%	18%	16%

在该报告中的数据结论定义可能有如下两种。

第一种　数据事实

昨日（2014-04-22）公司订单量8745，环比增长33%；商品销售量33101，环比增长54%；重复购买率62%，环比提高42%；新客户比例46%，环比提高18%；人均商品数量3.8，环比提高16%。

大多数的日常报告的结论可能就是类似于以上的数字陈述，报告中不存在任何结论。这种工作通常不能被视为一份报告，而是一个电子表

格或数据，可以通过系统报表自动实现。

第二种　数据结论

那真正的数据报告结论是什么？以表4-7的数据为例，结论可能是：

昨日公司整体销售状况环比前日大幅度提升。——这是一个总的结论。

订单量8745，环比增长33%；商品销售量33101，环比增长54%。——数据陈述。

公司订单量和商品销售量增长比例较大且超出正常波动范围，需要相关部门A、部门B（具体负责的部门）关注。——数据结论定义为增长，增长状态为超过正常范围。

重复购买率62%，环比提高42%。——数据陈述。

意味着用户重复购买的次数大大提高，且提高的比例超过了正常波动的上限范围，这是一个积极信号。——数据结论定义为增长，增长状态为超过正常范围。

人均商品数量3.8，环比提高16%。——数据陈述。

属于正常波动范围。——定性结论。

完整的日报结论部分为：

昨日公司整体销售状况环比前日大幅度提升。公司订单量和商品销售量增长比例较大且超出正常波动范围，需要相关部门A、部门B（具体负责的部门）关注。网站访问量和页面浏览量增长比例较大且超出正常波动范围，需要相关部门关注。重复购买率62%，环比提高42%，意味着用户重复购买的次数大大提高，且提高的比例超过了正常波动的上限范围，这是一个积极信号。人均商品数量3.8，环比提高16%，属于正常波动范围。

以上结论只是一个示例，在实际业务中会根据汇报对象的层次、理

解水平、对数据的认识程度以及听汇报的习惯重新组织语言和格式等；另外，由于没有对其他数据进行相关性分析，无法提炼出某些指标的直接对接和负责部门，实际应用中需要根据当前部门分工以及工作重点，将各个指标的负责部门联系起来，以便产生数据驱动效应。

4.8.4 数据结论不要产生于单一指标

数据结论产生于单一指标指当前结论的来源是某个指标，而非全面的数据指标。这是普遍存在于分析报告中的错误，原因是单一指标通常无法全面衡量某一业务的整体效果。比如，昨日全站订单量提升20%并不意味着全站销售效果提升，还要根据客单价、实际妥投率等做综合评估。

举例：在做网站数据分析时，经常会使用三个流量质量评估指标——跳出率、新访问占比和访问深度，当这三个指标的数值都环比上升时，反映的趋势并不完全一致。

·跳出率的提升是业务不希望看到的结果，该结果意味着流量质量不高（暂且不论是站外流量质量问题还是站内落地页设计问题）。

·新访问占比虽然有提升，但无法判断提升对业务是好还是坏，原因是企业不同阶段及业务不同目标决定了该目标的评估取向。如果业务目标是扩大品牌认知，那么需要提高对新用户的覆盖度，此时数据目标是提高新访问占比；如果业务目标是增加老用户回访体验活动，那么需要提高老用户访问占比。

·访问深度的提升是一个积极的效果，意味着用户浏览的页面数量增加，但是如果访问深度太深也可能是一个不好的信号，可能说明用户迷失在页面上无法快速找到意向内容。

假设只有三个网站流量质量评估指标（实际情况中不只三个），其中任何一个都无法全面说明网站质量情况。

4.8.5 数据分析不要预设价值立场

所谓预设价值立场指的是在做数据分析之前就已经先入为主的有了某种价值判断。

数据有没有立场？

数据的公正客观在大多数人看来是与生俱来的，因为数据的存在就是客观的。数据的存在的确是客观的，但数据的分析和应用的主体是“人”，不同人对同一数据的分析结果可能不同，这取决于数据从业者的立场。

这会影响什么？

我们对数据存在的初始期望是希望数据能客观地反馈业务结果，并服务于业务优化和改进。如果对数据的分析解读不客观、不公正，那么结果必然有失公允，基于数据的决策将面临风险。

为什么会这样？

数据从业者的立场决定了数据的立场，这种立场受两方面因素影响：

一是数据从业者在公司所处的角色：如果数据从业者在企业组织架构中位于采销中心之下，在对公司级数据进行整理并汇报采销相关数据时，出于自我中心或其他因素的保护意识，可能会出现不客观的结果，比如只报喜不报忧、甚至颠倒是非。

二是数据从业者基本的价值观：任何人都有基本的认知价值观，对数据从业者而言，如果在拿到一个案例之后，先有了结果偏向，那么整个分析和挖掘过程必然会只选择与其结果一致性的样本和方法进行验证，这可能直接导致客观数据结果扭曲。

举例：假如某次活动时网站的转化率是1.2%，要对此指标做数据分析。

数据分析的第一步是定性结果，1.2%的转化率是好还是坏？有比较才能区分好坏，如何选择比较方法？常用的比较分析方法有环比、占比、定基比、横向比、纵向比等，每一种对比方法又可以选择不同时间进行对比，如昨日、上周今日、上月今日等。不同比较方法、不同时间的对比结果可能存在差异甚至是截然相反。如何在符合统计学基本前提下做结果定性？

假设第一步定性工作完成，该活动转化率是好的结果。下一步需要分析为什么好？到底是谁的“功劳”？电商网站做大型促销活动时存在一个普遍规律：只要价格足够低，无论用户体验多差、网站UI多烂、送货速度多慢、客服态度多差，这些都不会太影响转化率。这意味着，无论企业营销、网站运营工作效果如何，只要能保证页面正常访问，所有节点的转化率结果都会特别好。此时，各个业务部门对转化率的影响各占多少权重？

假设数据分析师排除万难，通过复杂模型算法计算出各个业务节点的贡献，作为数据分析师，如何跟领导汇报并解读各个业务节点的真实贡献，是实话实说还是含糊其辞，甚至是颠倒是非？

客观、公正是数据从业者的职业要求和个人素质之一，任何基于数据的决策项目都要求从业者秉着客观、公正的态度去对待，否则数据工作不仅没有价值，反而会误导决策，这是非常危险的事情。

4.8.6 不要忽视数据与业务的需求冲突问题

在数据分析挖掘工作开始前，都需要有一段时间积累数据，目的是尽量获得能符合实际业务返回的完整数据集，基于这种数据所得出的结论才有可能是客观、公正的。数据需求的严谨性主要体现在数据采集阶段。数据采集阶段要求数据样本量必须具备在一定周期内相对稳定的特征，并且这种特征能在后期数据处理中排除异常值波动的影响，进而得到完整、真实反馈业务效果的数据。

数据采集通常会受两方面影响，一是数据采集单位效率，即每天能采集多少数据；二是周期，即使数据单位采集效果很高，也不能只使用很短的周期数据进行分析，因为当期的数据可能存在异常值，而且该异常值不通过数据对应分析是无法验证和剔除的。因此，数据采集阶段通常至少需要一周的数据采集数据，如果采集效率低，则需求时间更长。

但在业务方看来，如此“长”的时间通常是无法忍受的。业务方通常想要在较大业务动作后立即反馈结果进行优化矫正，但我们看到业务方的这种“短、快、全”的需求直接与数据需求的严谨性产生冲突。这种冲突的场景有：

·某站内广告在首页焦点图的A位置只放3天，3天后马上下架换新素材。

·某站内UED部门做产品体验提升，每周做一次产品方案优化。

这几种业务场景从客观上直接导致数据需求严谨性的缺失，因此会对数据质量和后期的分析挖掘产生一定影响。

数据严谨性并不意味着数据结果的产生一定需要很长时间，时间长短取决于业务需求下数据对时间和数据样本的要求。通常实时数据、即席报表都能以很快的速度反馈业务关键节点，以帮助业务做及时调整，比如某渠道推广效果、站内某活动实时效果等。但某些长期、对全局性有影响的关键业务节点需要更慎重的决策支持以避免数据决策失误，比如首页改版、购物车改进等。

4.9 内容延伸：非结构化数据的分析与挖掘

在之前的内容中，我们主要介绍了常用的数据分析和挖掘方法，大多数是针对结构化数据（或者已经被转换为结构化的数据）开展的。本节将介绍一些针对非结构化数据的分析和挖掘方法，主要侧重于文本处理领域。

4.9.1 词频统计

词频统计是针对文档中的词出现频率的统计分析。通常词频统计采用的方法是TF-IDF方法，但本节使用的只是基于分词后的词语统计出现的次数。有关TF-IDF的应用，会在下面的章节中提到。

示例模拟的是从一篇文章中提取关键字并作词频统计分析，主要用到的技术是中文分词、词频统计和词云展示。数据源文件article1.txt位于“附件-chapter4”中，默认工作目录为“附件-chapter4”（如果不是，请cd切换到该目录下，否则会报“IOError:File article1.txt does not exist”）。完整代码如下：

```
# 导入库
import re  # 正则表达式库
import collections  # 词频统计库
import numpy as np  # numpy库
import jieba  # 结巴分词
import wordcloud  # 词云展示库
from PIL import Image  # 图像处理库
import matplotlib.pyplot as plt  # 图像展示库

# 读取文本文件
fn = open('article1.txt')  # 以只读方式打开文件
string_data = fn.read()  # 使用read方法读取整段文本
fn.close()  # 关闭文件对象

# 文本预处理
pattern = re.compile(u'\t|\n|\.|-|一|:|;|\)|\(|\?|"')  # 建立正则表达式匹配模式
string_data = re.sub(pattern, '', string_data)  # 将符合模式的字符串替换掉

# 文本分词
seg_list_exact = jieba.cut(string_data, cut_all=False)  # 精确模式分词[默认模式]
object_list = []  # 建立空列表用于存储分词结果
remove_words    =    [u'的',    u'，',    u'和',    u'是',    u'随着',    u'对于',  ' ',  u'对',  u'等',  u'能',  u'都',  u'。',  u'、',  u'中',  u'与',  u'在',  u'其',  以',    u'进行',    u'有',    u'更',    u'需要',    u'提供',    u'多',    u'能力',    u'通过',    u'会',    u'不同',    u'一个',    u'这个',    u'我们',    u'将',    u'并',    u'同时',  u'看',  u'如果',  u'但',  u'到',  u'非常',  u'—',  u'如何',  u'包括',  u'这']  # 自定义去除词库
# remove_words = [] #空去除词列表，用于跟关键字提取做效果对比
for word in seg_list_exact:  # 迭代读出每个分词对象
    if word not in remove_words:  # 如果不在去除词库中
        object_list.append(word)  # 分词追加到列表

# 词频统计
word_counts = collections.Counter(object_list)  # 对分词做词频统计
word_counts_top5 = word_counts.most_common(5)  # 获取前10个频率最高的词
for w, c in word_counts_top5:  # 分别读出每条词和出现从次数
    print w, c  # 打印输出
```

```
# 词频展示
mask = np.array(Image.open('wordcloud.jpg'))  # 定义词频背景
wc = wordcloud.WordCloud(
     font_path='C:/Windows/Fonts/simhei.ttf',   # 设置字体格式，不设置将无法显示中
文
    mask=mask,  # 设置背景图
    max_words=200,  # 设置最大显示的词数
    max_font_size=100  # 设置字体最大值
)
wc.generate_from_frequencies(word_counts)  # 从字典生成词云
image_colors = wordcloud.ImageColorGenerator(mask)  # 从背景图建立颜色方案
wc.recolor(color_func=image_colors)  # 将词云颜色设置为背景图方案
plt.imshow(wc)  # 显示词云
plt.axis('off')  # 关闭坐标轴
plt.show()  # 显示图像
```

上述代码以空行分为6个部分。

第一部分：导入库。示例中用到了正则表达式库re，用来做字符串匹配替换；词频统计库collections用来基于分词列表做词频统计；Numpy库配合PIL的Image做背景图像的读取和转换；jieba用来做中文分词；wordcloud用来做根据词频的词云图像展示；Matplotlib配合wordcloud做图像展示。

第二部分：读取文本文件。该部分使用Python标准文件读取方法，先通过默认的open方法以只读方式打开文本文件，然后使用read方法将所有文本信息读取为整个字符串，最后关闭文件对象。

第三部分：文本预处理。该部分主要功能是去除文本数据中无效的字符，先使用re.compile方法建立正则表达式匹配模式，然后使用re.sub方法将符合匹配模式的字符串替换掉（去掉）。

第四部分：文本分词。通过jieba.cut以默认的精确分词模式做中文分词，然后分别建立空列表用于接下来的循环分词的追加，建立remove_words用来过滤掉一些停用词、助词、语气词等无关词汇，接着使用for循环读取结巴分词的每个结果，先判断是否属于要过滤的词，如果不属于则追加到要使用的分词列表中。

上述几个部分在第三章都有相关介绍，这里不再做详细用法说明。

第五部分：词频统计。这里使用collections.Counter方法对分词列表做统计，然后提取词频统计结果中词频最高的5个词，打印输出结果如

下：

数据113

分析48

功能47

Adobe 45

Analytics 37

上述结果的第一列是分词后的词语，第二列是对应出现的次数（注：该demo示例没有排除所有无关词，因此在接下来的词云展示中仍然会出现部分无关词汇）。在真实使用场景下，通常会建立去除词的列表并定期做更新维护，这样越到后期该列表会越有效。

第六部分：词频展示。这里使用了wordcloud库做基于自定义图案的词云展示。

先使用Image的open方法读取一个文件，返回结果是一个文件对象实例，通过numpy.array将其转换为数组才能在wordcloud做应用，该步骤建立了一个自定义展示图案，同时也是自定义图案配色的来源。

接着通过wordcloud.WordCloud方法建立一个词云对象，并详细定义展示细节。

·font_path：展示词云需要的字体库，中文下需要自定义该参数，否则中文将无法正常显示；

·mask：设置词云展示的图案；

·max_words：设置最大展示词云的数量；

·max_font_size：控制字体最大值。

对词云对象使用generate_from_frequencies方法，从词频统计结果中获取词频数据。

使用wordcloud.ImageColorGenerator方法自定义展示颜色方案，并应用到wc.recolor方法中，该步骤可以改变默认词云的颜色方案，用来自适应图像本身的区域和对应颜色方案。

最后通过plt的imshow方法展示词云，设置关闭坐标轴并展示图像。词云的展示以Python默认Logo样式和颜色方案为基准（原图形文件可参看附件4中的wordcloud.jpg文件），原图形文件和生成的词云结果如图4-18所示。

上述过程中，主要需要考虑的关键点是如何将词频形象的展示出来。除了传统的基于关键字数量的条形图、柱形图展示以外，可以应用词云展示方案，并可以基于不同的图案和颜色做展示配置。

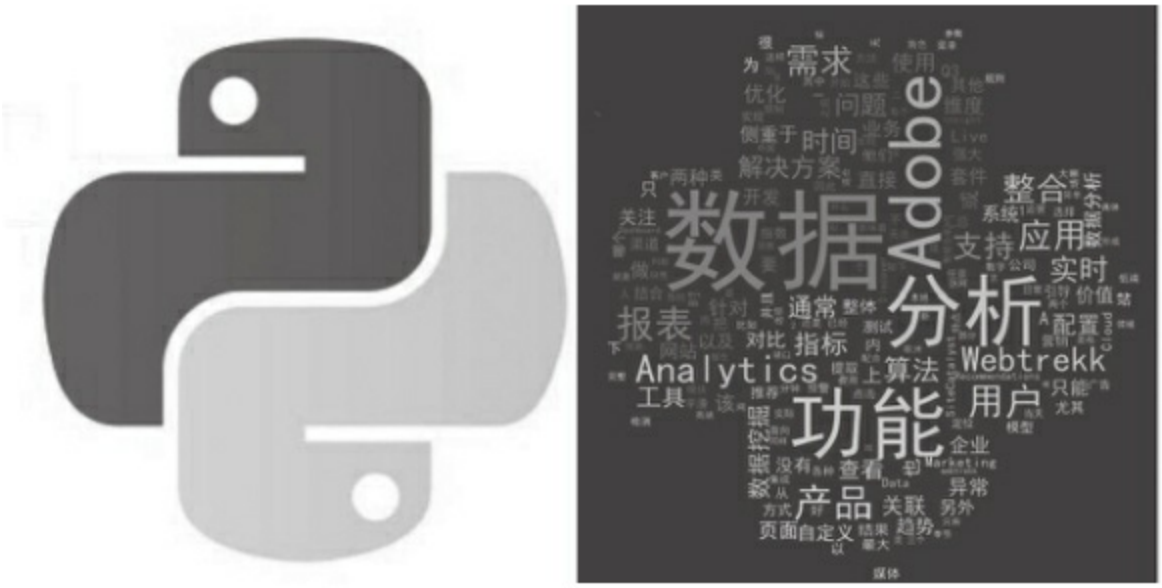


图4-18　词云展示

如果使用自定义背景图案做词云展示时，背景图案应尽量简单、明了，同时要颜色对比明显且有大面积的色块，不要有复杂的图案和线条，否则词云展示效果不好。

代码实操小结：在本节的代码中，主要使用了以下几个知识点：

·使用正则表达式库re.compile做多模式的匹配以及使用re.sub做模式

替换；

·使用jieba.cut做中文分词；

·使用列表的append方法做列表追加；

·使用for循环配合if条件语言做循环处理；

·使用collections库collections.Counter方法做词频统计，并基于词频统计结果的most_common方法提取词频最高的n的词；

·使用Image.open读取图像；

·使用numpy.array做数据转换；

·使用wordcloud做词云展示，包括自定义词云背景、字体、最大显示的词语数量、字体最大值、自定义配合方案等。

4.9.2 词性标注

所谓词性标注是给每个词语确定一个词性分类。例如，“我爱Python数据分析与数据化运营”分词后的结果可能是“我|爱|Python|数据分析|与|数据|化|运营”，其中每个词都有专属的分类：

·我：代词

·爱：动词

·Python：英语

·数据分析：习用语

·与：介词

·数据：名词

·化：名词

·运营：名动词，具有名词功能的动词

在运营中，有很多场景需要做词性标注，然后基于标注的词性可以做进一步应用。例如统计竞争对手新闻稿的主要词语分布、分词结果筛选和过滤、配合文章标签的提取等。

本示例中，模拟的是从一篇文章中提取关键字并作词性标注，主要用到的技术是中文分词和标注。数据源文件article1.txt位于“附件-chapter4”中，默认工作目录为“附件-chapter4”（如果不是，请cd切换到该目录下，否则会报“IOError:File article1.txt does not exist”）。完整代码如下：

```
# 导入库
import jieba.posseg as pseg
import pandas as pd

# 读取文本文件
fn = open('article1.txt')  # 以只读方式打开文件
string_data = fn.read()  # 使用read方法读取整段文本
```

```
fn.close()  # 关闭文件对象

# 分词+词性标注
words = pseg.cut(string_data)  # 分词
words_list = []  # 空列表用于存储分词和词性分类
for word in words:  # 循环得到每个分词
    words_list.append((word.word, word.flag))  # 将分词和词性分类追加到列表
words_pd = pd.DataFrame(words_list, columns=['word', 'type'])  # 创建结果数据
框
print (words_pd.head(4))  # 展示结果前4条

# 词性分类汇总-两列分类
words_gb = words_pd.groupby(['type', 'word'])['word'].count()
print (words_gb.head(4))

# 词性分类汇总-单列分类
words_gb2 = words_pd.groupby('type').count()
words_gb2 = words_gb2.sort_values(by='word', ascending=False)
print (words_gb2.head(4))

# 选择特定类型词语做展示
words_pd_index = words_pd['type'].isin(['n', 'eng'])
words_pd_select = words_pd[words_pd_index]
print (words_pd_select.head(4))
```

上述代码以空行分为6个部分。

第一部分：导入库。本示例中用到了jieba.posseg直接做带有词性标注的分词，用pandas做数据结构化展示。

第二部分：分词+词性标注。使用pseg.cut（结巴分词的jieba.posseg）将读取的文本内容做分词；创建一个空列表用于存储分词和词性分类；然后通过一个循环读取每个分词，将分词和词性分类追加到列表并创建结果数据框，前4条数据如下：

```
        word type
0      Adobe  eng
1                x
2  Analytics  eng
3         和   c
```

数据框中的word列是分词结果，type是分词标注的类型。结巴分词的词性标注方法采用和ictclas兼容的标记法。常用的分类如表4-8所示。

表4-8　常用结巴分词词性分类

一级分类	二级分类	名　称	描　述
a		形容词	取英语形容词 adjective 的第 1 个字母
	ad	副形词	直接作状语的形容词。形容词代码 a 和副词代码 d 并在一起
	ag	形语素	形容词性语素。形容词代码为 a，语素代码 g 并在一起
	an	名形词	具有名词功能的形容词。形容词代码 a 和名词代码 n 并在一起
b		区别词	取汉字“别”的声母
c		连词	取英语连词 conjunction 的第 1 个字母
d		副词	因其第 1 个字母已用于形容词，所以取 adverb 的第 2 个字母
	dg	副语素	副词性语素。副词代码为 d，语素代码 g 前面置以 d
e		叹词	取英语叹词 exclamation 的第 1 个字母
f		方位词	取汉字“方”的声母
g		语素	绝大多数语素都能作为合成词的“词根”，取汉字“根”的声母
h		前接成分	取英语 head 的第 1 个字母
i		成语	取英语成语 idiom 的第 1 个字母
j		简称略语	取汉字“简”的声母
k		后接成分	
l		习用语	习用语尚未成为成语，有点“临时性”，取“临”的声母
m		数词	取英语 numeral 的第 3 个字母，n，u 已有他用
n		名词	取英语名词 noun 的第 1 个字母
	ng	名语素	名词性语素。名词代码为 n，语素代码 g 前面置以 n
	nr	人名	名词代码 n 和“人 (ren)”的声母并在一起
	ns	地名	名词代码 n 和处所词代码 s 并在一起
	nt	机构团体	“团”的声母为 t，名词代码 n 和 t 并在一起
	nz	其他专名	“专”的声母的第 1 个字母为 z，名词代码 n 和 z 并在一起
o		拟声词	取英语拟声词 onomatopoeia 的第 1 个字母
p		介词	取英语介词 prepositional 的第 1 个字母
q		量词	取英语 quantity 的第 1 个字母
r		代词	取英语代词 pronoun 的第 2 个字母，因 p 已用于介词

(续)

一级分类	二级分类	名　称	描　述
s		处所词	取英语 space 的第 1 个字母
t		时间词	取英语 time 的第 1 个字母
	tg	时语素	时间词性语素。时间词代码为 t，在语素的代码 g 前面置以 t
u		助词	取英语助词 auxiliary 的第 2 个字母，因 a 已用于形容词
v		动词	取英语动词 verb 的第一个字母
	vd	副动词	直接作状语的动词。动词和副词的代码并在一起
	vg	动语素	动词性语素。动词代码为 v，在语素的代码 g 前面置以 V
	vn	名动词	指具有名词功能的动词。动词和名词的代码并在一起
x		非语素字	非语素字只是一个符号，字母 x 通常用于代表未知数、符号
y		语气词	取汉字“语”的声母
z		状态词	取汉字“状”的声母的前一个字母

第四部分：词性分类汇总——两列分类。使用Pandas的grouby方法对'type'和'word'这两列同时做分类汇总，汇总列为'word'，汇总方式为计数。得到的前4条结果如下：

```
type  word
a     不同      14
      不足       2
      不通       1
      严谨       2
Name: word, dtype: int64
```

第五部分：词性分类汇总——单列分类。仍然使用Pandas的grouby

方法对'type'做分类汇总，汇总列为'type'，汇总方式为计数；然后对汇总结果使用sort_values方法对word列做逆序排序，得到的前4条结果如下：

```
      word
type
x      994
n      981
v      834
eng    295
```

第六部分：选择特定类型词语做展示。代码中通过使用isin方法从列表中选择了名词（n）和英文（eng），然后得到新的数据框，前4条数据如下：

```
         word type
0       Adobe  eng
2   Analytics  eng
4    Webtrekk  eng
9          领域    n
```

上述过程中，所有步骤相对简单，没有技术上和理解上的难点。

代码实操小结：在本节的代码中，主要使用了以下几个知识点：

·使用jieba.posseg做带有词性标注的分词，并通过循环得到每个分词的词语和类别结果；

·使用列表的append方法追加元素；

·使用Pandas的Dataframe方法基于列表建立数据框；

·使用数据框的groupby方法做分类汇总，并可指定分类汇总的列以及汇总方式；

·使用数据框的head方法展示前n条数据；

·使用数据框的isin从数据框中查找符合列表元素的数据索引；

·使用数据框的sort_values方法对数据数据框排序，并指定排序规则。

4.9.3 关键字提取

关键字提取是从文本中提取跟内容最相关的词语，关键字抽取的结果经常使用的场景是文档检索、文章标签编辑等，也经常用在文本聚类、文本分类、关键字摘要等方面。

本示例中，模拟的是从一篇文章中提取关键字，主要用到的技术是中文关键字提取。数据源文件article1.txt位于“附件-chapter4”中，默认工作目录为“附件-chapter4”（如果不是，请cd切换到该目录下，否则会报“IOError:File article1.txt does not exist”）。完整代码如下：

```
# 导入库
import jieba.analyse  # 导入关键字提取库
import pandas as pd  # 导入pandas

# 读取文本数据
fn = open('article1.txt')  # 以只读方式打开文件
string_data = fn.read()  # 使用read方法读取整段文本
fn.close()  # 关闭文件对象

# 关键字提取
tags_pairs = jieba.analyse.extract_tags(string_data, topK=5, withWeight=True
['ns', 'n', 'vn', 'v', 'nr'], withFlag=True)  # 提取关键字标签
tags_list = []  # 空列表用来存储拆分后的三个值
for i in tags_pairs:  # 打印标签、分组和TF-IDF权重
    tags_list.append((i[0].word, i[0].flag, i[1]))  # 拆分三个字段值
tags_pd = pd.DataFrame(tags_list, columns=['word', 'flag', 'weight'])  # 创
建数据框
print (tags_pd)  # 打印数据框
```

上述代码以空行分为3部分。

第一部分：导入库。本示例用到了关键字提取库jieba.analyse，使用Pandas做格式化输出。

第二部分：读取文本数据。使用read方法读取文件的内容为字符串。

第三部分：关键字提取。使用jieba.analyse.extract_tags方法，提取前5个关键字标签，以下是各参数信息：

·string_data：要提取的文本对象；

·topK：提取的关键字数量，不指定则提取全部；

·withWeight：设置为True指定输出词对应的IF-IDF权重；

·allowPOS：只允许提取符合特定词语类别的标签，具体类别见4.9.2节中的完整表格；

·withFlag设置为True指定输出的关键字标签带有词语类别信息。

关键字提取的结果是一个字典，每一个元素都由一个关键字和词语类别组成的元组以及权重值组成。接下来将结果解析出来：先创建一个空列表用于存储数据，然后通过循环将每个列表元素的关键字标签、分组和TF-IDF权重以元组的形式追加到列表中；最后建立一个数据框用来存储和展示数据，结果如下：

	word	flag	weight
0	数据	n	0.313395
1	报表	n	0.163367
2	功能	n	0.150263
3	分析	vn	0.134857
4	用户	n	0.126633

相关知识点：TF-IDF

TF-IDF（term frequency–inverse document frequency）是一种针对关键字的统计分析方法，用来评估关键字或词语对于文档、语料库和文件集合的重要程度。关键字的重要程度跟它在文档中出现的次数成正比，但同时跟它出现的频率呈反比。这种加权的形式能有效避免常用词对于关键字搜索结果的影响，提高搜索结果与关键字之间的相关性。

TF-IDF的主要思想是：如果某个关键字在一篇文档中出现的频率（TF，Term Frequ-ency）高，并且在其他文档中很少出现，那么认为该关键字具有良好的区分不同文档的能力，也就越重要。

在4.9.1节中，我们使用了基本的频率（TF，Term Frequency）方法，这会导致这样的问题：出现频率最高的往往是文档中的助词、介词、标签符号等，例如的、逗号、句号等。因此，在做关键字相关处理之前基本上都要有停用词处理的步骤。在4.9.1节对应的代码附件中，读

者可取消第567行代码注释#remove_words=[]#空去除词列表，用于跟关键字提取做效果对比，再次运行其代码查看前5个关键字最高的结果如下：

```
的291
，241
123
。117
数据113
```

因此，TF-IDF是做词频统计的基本思路和方法，也是做词语向量化、以及基于文本向量的聚类、分类等应用时的基本方法。

上述过程中，所有步骤都比较简单，没有技术上和理解上的难点。

代码实操小结：在本节的代码中，主要使用了以下几个知识点：

·使用关键字提取库jieba.analyse.extract_tags提取关键字，并可设置关键字提取类别、权重、数量等；

·使用Pandas做格式化输出；

·使用Python标准open方法读取文件并使用read方法从文件对象读取内容为字符串；

·使用列表的append方法追加元素；

·使用Pandas的Dataframe方法基于列表建立数据框。

4.9.4 文本聚类

文本聚类就是要在一堆文档中，找出哪些文档具有较高的相似性，然后可以针对这些相似性文档的聚合进行类别划分。文本聚类应用场景：提供大规模文档集进行类别划分并提取公共内容的概括和总览；找到潜在的各个文档间的相似度以进行相似度判别、类别修正，以减少浏览相似文档和信息的时间和精力。

通常，聚类分析（也包括其他算法）大多是针对数值型做计算的，K均值这类基于聚类的算法要求只有数值型变量才能得到距离相似度。对于文本聚类而言，由于不同文本集出现的全部都是文字内容，因此无法直接针对这些文本进行聚类。

要实现文本聚类，除了要进行必要的文本数据清洗和预处理外，还有两个前置条件：

·分词。分词是实现聚类条件的第一个必要步骤，分词之后会得到不同文本集中已经分割好的单词（也包括中文词语）。

·word to vector。也称为文本转向量、词转向量，目的是将不同文本集的单词集合，转换为向量集合，然后通过向量空间模型建立向量矩阵。

完成上述两个基本步骤之后，可以基于向量矩阵做文本聚类分析、情感分析、词频统计、相似词频等。有关中文分词和word to vector的更多介绍，请参照3.12.4节。

本示例中，模拟的是从某商品的60个商品评论中提取用户的评价关键字，然后对关键字进行聚类，找到特定类别的关键字特征。数据源文件comment.txt位于“附件-chapter4”中，默认工作目录为“附件-chapter4”（如果不是，请cd切换到该目录下，否则会报“IOError:File comment.txt does not exist”）。完整代码如下：

```
# 导入库
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer  # 基于TF-IDF的词
```

```
频转向量库
from sklearn.cluster import KMeans
import jieba.posseg as pseg

# 中文分词
def jieba_cut(comment):
    word_list = []  # 建立空列表用于存储分词结果
    seg_list = pseg.cut(comment)  # 精确模式分词[默认模式]
    for word in seg_list:
        if word.flag in ['a', 'ag', 'an']:  # 只选择形容词
            word_list.append(word.word)  # 分词追加到列表
    return word_list

# 读取数据文件
fn = open('comment.txt')
comment_list = fn.readlines()  # 读取文件内容为列表
fn.close()

# word to vector
stop_words = [u'…', u'。', u', ', u'? ', u'! ', u'+', u' ', u'、', u': ', u';
']  # 定义停用词
vectorizer = TfidfVectorizer(stop_words=stop_words, tokenizer=jieba_cut, use
建词向量模型
X = vectorizer.fit_transform(comment_list)  # 将评论关键字列表转换为词向量空间模
型

# K均值聚类
model_kmeans = KMeans(n_clusters=3)  # 创建聚类模型对象
model_kmeans.fit(X)  # 训练模型

# 聚类结果汇总
cluster_labels = model_kmeans.labels_  # 聚类标签结果
word_vectors = vectorizer.get_feature_names()  # 词向量
word_values = X.toarray()  # 向量值
comment_matrix = np.hstack((word_values, cluster_labels.reshape(word_values.
向量值和标签值合并为新的矩阵
word_vectors.append('cluster_labels')  # 将新的聚类标签列表追加到词向量后面
comment_pd = pd.DataFrame(comment_matrix, columns=word_vectors)  # 创建包含词
向量和聚类标签的数据框
print (comment_pd.head(1))  # 打印输出数据框第1条数据

# 聚类结果分析
comment_cluster1 = comment_pd[comment_pd['cluster_labels'] == 2].drop('clust
择聚类标签值为2的数据，并删除最后一列
word_importance = np.sum(comment_cluster1, axis=0)  # 按照词向量做汇总统计
print (word_importance.sort_values(ascending=False)[:5])  # 按汇总统计的值做逆
序排序并打印输出前5个词
```

上述代码以空行分为7个部分。

第一部分：导入库。本示例用到的Numpy、Pandas都是基本处理库，用到了Sklearn做word to vector处理以及K均值聚类，使用结巴分词做中文分词。

第二部分：定义了一个函数jieba_cut用来对每一条文本数据（每个

评论）做中文分词，该函数将作为sklearn.feature_extraction.text.TfidfVectorizer的分词器对中文分词。实现该功能时先定义一个空列表用于存储分词结果数据，接着使用jieba.posseg的cut方法进行中文分词，然后使用for循环只将属于特定分类（形容词）的关键字加入分词列表中，返回分词列表。

第三部分：读取数据文件。由于原始数据中存储的是每一条都是一个评论，因此需要将每一条作为列表的一个值。使用Python标准open方法打开文件并使用readlines方法将内容读取为列表。

第四部分：word to vector（文本转向量或词语转向量）。先定义一批要去除的词语列表，定义列表时使用u来表示是unicode字符串；然后使用sklearn.feature_extraction.text的TfidfVectorizer方法创建词语转向量的对象，使用fit_transform方法将评论关键字列表转换为词向量空间模型。TfidfVectorizer方法参数如下：

·stop_words：指定为自定义的去除词的列表，不指定默认会使用英文的停用词列表。

·tokenizer：用来设置定义的分词器，这里是在上面自定义的结巴分词。默认的分词器对于英文下工作良好，但对于中文来讲效果不佳。

·use_idf：设置为True指定TF-IDF方法做词频转向量。

第五部分：使用K均值聚类。先使用KMeans方法创建聚类对象，设置聚类数为3；然后使用fit方法训练聚类模式。

第六部分：聚类结果汇总。

先获取要做汇总的三类值：聚类标签、词向量和向量值。使用聚类模型对象的labels_方法获得聚类标签，使用词语转向量的get_feature_names方法获得向量列表，将转换后的词向量应用toarray方法转化为数组。

建立数据框用于聚类标签、词向量和向量值。使用Numpy的hstack将向量值和向量标签按列合并，得到新的矩阵；通过列表的append方法将聚类标签的列名追加到词向量列表中；通过Pandas的DataFrame方法

基于新的矩阵创建数据框，数据框第一行数据如下：

一般不厚不爽不贵不错不高便宜具体准确凹...蓝诚意\

```
      一般   不厚   不爽   不贵   不错   不高   便宜   具体   准
确   凹      ...        蓝   诚意  \
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0       ...        0.0  0.
     贵   足   透亮   重要   难         高   麻烦  cluster_labels
0  0.0  0.0  0.0  0.0  0.0  0.392535  0.0             1.0
[1 rows x 64 columns]
```

使用hstck和vstack可以实现将两个矩阵做基于行或基于列的合并，这在矩阵处理中经常用到。

数据框包含60行（60个评论）、64列（其中63个词向量和1个聚类标签）。

第七部分：聚类结果分析。本段将从结果中抽取聚类标签为2的评论，发现其评论的显著性词语。使用数据框得到聚类标签为2的数据，并将得到的数据框删除最后一列聚类标签；将得到的新数据框按列最数值汇总，并按逆序排序，得到新关键字按总和排序，前5条数据如下：

```
好      4.890181
流畅     0.808409
清晰     0.685972
着急     0.641769
慢      0.508659
dtype: float64
```

通过数据发现，聚类标签为2的评论中，对于商品的评论关键词倾向是积极的，主要关键字是好、流畅、清晰等。

上述过程中，主要需要注意的关键点：

·中文分词需要自定义分词器，结巴分词是比较好的分词工具；

·将评论列表转换为评论——向量空间模型，然后做文本聚类。

代码实操小结：在本节的代码中，主要使用了以下几个知识点：

·使用sklearn.feature_extraction.text的TfidfVectorizer方法将文本转换为向量；

·使用文本转向量模型的get_feature_names方法获得向量列表；

·使用sklearn.cluster的KMeans方法对文本向量做聚类分析；

·使用聚类模型对象的labels_方法获得训练数据的聚类标签；

·使用jieba.posseg的cut方法将中文文本切分为带有分类词性标签的关键字；

·使用Python标准open方法读取文件并使用readlines方法从文件对象读取内容为列表；

·使用列表的append方法追加元素；

·使用Pandas的Dataframe方法基于列表建立数据框；

·使用Pandas的sort_values方法排序；

·使用数据框的head方法只显示前n条数据；

·使用toarray方法将对象转换为数组；

·使用Numpy的hstack方法将两个矩阵做列合并；

·使用Numpy的sum方法沿特定轴做汇总统计；

·通过Numpy的shape方法获得矩阵形状，并使用reshape方法做矩阵形状转换。

4.10 本章小结

内容小结：本章的内容侧重于实际经验的讲解，其中每个部分都凝结了笔者的工作体会，尤其是关于应用误区以及注意事项等内容，需要读者多加留意。除了常用的统计分析方法外，笔者还在内容延伸中拓展了部分文本分析和挖掘方法，而对于其他领域的分析和挖掘，例如图像、视频、语音等也都有特定的分析、挖掘和建模方法，限于篇幅无法一一介绍。内容中涉及技术的部分都有对应示例代码，该代码可在“附件-chapter4”中的名为chapter4_code.py的文件中找到。

重点知识：本章几乎在每个部分内容中都有“小结”，其中已经注明了重点知识和需要引起注意的知识点。

外部参考：本章中提到了一些知识需要读者通过外部途径做深入学习和了解。

尽管本章写的是有关分析和挖掘的内容，但几乎没有一点知识性和理论性的内容，如果读者有兴趣对于这些理论性内容，可以参考以下几本书：《数据挖掘概念与技术（原书第3版）》（《Data Mining Concepts and Techniques，Third Edition》），这本书是关于数据挖掘方面的专业著作；《深入浅出数据分析》，这是一本写的非常有趣的入门级的数据分析方法和理论书籍。

在本书中大量用到了占位符做格式化输出、变量定义等应用，有关占位符配合print打印输出还有更多内容，例如字符串类型、对其方式、格式化指示符等，请读者额外了解“Python print格式化输出”。

在做分类应用画树形图时用到了Graphviz，这是贝尔实验室开发的一个开源的工具包，它可以用于将结构信息表示为抽象图形和网络图。本章中展示决策树的规则应用只是其图形绘制能力的冰山一角。除了树图外，还可以绘制流程图、网络图等复杂关系和知识图谱，更多详细信息请参照http://www.graphviz.org。

有关非结构化处理还有更多领域的应用，例如文本分析和挖掘方向的提取相似关键字、文本摘要、文章相似度、文本分类等，网络应用领域的最短路径、PageRank、传播聚类，图像识别领域人脸识别、前后景

分割、角点检测等知识需要读者书外了解。

应用实践：lambda表达式是一个非常好的解决“小问题”的功能方法，读者可写几个简单应用。

尝试对历史应用的项目进行反思和总结，看是否有本章中提到的问题而没有注意到。

自动化和工程化能有效提高重复性工作的效率，建议读者对现有手头工作做梳理，将重复工作中常用的经验和方法固定下来，通过程序的形式减少人工参与度，然后有更多时间参与到更多的学习和项目实践中。

第5章　会员数据化运营

从本章开始，我们将开始介绍数据化运营的具体应用，包括会员数据化运营、商品数据化运营、流量数据化运营、内容数据化运营。

本章将从会员数据化运营的概述、关键指标、应用场景、数据分析模型、分析小技巧、分析大实话以及实际案例几个方面展开，逐步透析有关会员数据化运营的方方面面。

5.1 会员数据化运营概述

会员数据化运营几乎是所有企业的必备运营工作，企业要生存必须有会员（客户），无论企业处于发展周期的哪个阶段、企业规模如何、企业性质如何皆如此。会员数据化运营辅助于客户关系管理（CRM），可以用来解决以下几方面问题：

·会员的生命周期状态是什么；

·会员的核心诉求是什么；

·会员的转化习惯和路径是什么；

·会员的价值如何；

·如何扩大市场覆盖、获得更多的新会员；

·如何更好地维系老会员；

·应该在什么时间、采取何种措施、针对哪些会员做哪些运营活动；

·在特定运营目标下，应该如何制定会员管理策略，包括行为管理、体验管理、增值服务、信息管理、营销管理、客户关怀等。

传统的会员数据化运营更多聚焦在线上和线下的注册、购买会员，线下会员在注册转化之前的用户数据无法获取。在大数据技术支持下，这种状态正在发生变化，借助于人工智能、深度学习等关键技术，可以将线上相对成熟的人脸识别、路径追踪等方面的应用正逐步扩展到线下，这为线下会员完整生命周期的数据跟踪、识别和分析提供了基础。

5.2 会员数据化运营关键指标

会员数据化运营的关键指标包括会员整体指标、营销指标、活跃度指标、价值度指标、终生价值指标和异动指标。

5.2.1 会员整体指标

1.注册会员数

注册会员数是指企业已经成为注册会员的数量。根据注册时间周期的不同，又可以细分为累积注册会员数、新增注册会员数等。

2.激活会员数

激活会员相对于注册会员有一个特定的激活动作，该动作往往决定了用户是否真的成为企业会员，常见的代表性动作包括：点击确认链接、手机验证、身份验证等。

激活会员数是指已经注册的会员中有多少会员已经激活。根据激活时间周期的不同，又可以细分为累积激活会员数、新增激活会员数等。

激活会员数可以延伸出相对转化率指标：会员激活率。会员激活率指的是注册会员中已经完成激活的会员比例。

3.购买会员数

购买会员是真正给企业带来利润的群体。购买会员数指有过购买行为的会员数量（企业也可以根据自身转化定义为其他要素，如付费会员数）。根据购买时间周期的不同，又可以细分为累积购买会员数、新增购买会员数等。

购买会员数可以延伸出相对转化率指标：

·注册——购买转化率：从注册到购买的会员转化比例；

·激活——购买转化率：从激活到购买的会员转化比例。

在企业内部，如果转化周期和步骤较长，还会细分更多的转化状态指标，例如妥投会员、充值会员等。

5.2.2　会员营销指标

1.可营销会员数

可营销会员数是指整体会员中可通过一定方式进行会员营销以满足企业特定需求的会员数量。会员可营销的方式包括：手机号、邮箱、QQ、微信等具有可识别并可接触的信息点，具备这些信息中的任何一种便形成可营销会员。

2.营销费用

会员营销费用一般包括营销媒介费用、优惠券费用和积分兑换费用三种。

（1）营销媒介费用

营销媒介费用是特定营销媒介而产生的费用，例如短信费用、会员渠道推广费用、电子邮件费用等。

（2）优惠券费用

优惠券根据不同的使用条件和金额可以划分成多种，如30元红券、50元店铺券等，企业促销时申请的优惠券费用是会员营销费用的重要组成部分。

（3）积分兑换费用

大部分网站都有会员积分系统，会员积分通常可以兑换成金额使用。如网站的积分兑换比例为20：1，即每20个积分可以兑换1元钱。在促销活动时，除了前期投入的广告费用、促销优惠券费用外，还会包含两种情况的积分费用：一部分是积分可以直接兑换成人民币来支付订单，另一部分是订单生成后会赠送一定数量的积分又形成可供兑换的金额（对企业来说是费用）。这两种情况的积分兑换都构成会员营销费用。

3.营销收入

会员营销收入是通过会员营销渠道和会员相关运营活动产生的费用，包括电子邮件、短信、会员通知、线下二维码、特定会员优惠码等。

在评估会员营销活动产生的收入时，直接通过特定有标记的渠道或促销码而成交的数据能清晰分辨出来。但是如果用户没有直接通过会员营销的接触点形成转化，那么这种收入是无法评估的。例如会员看到短信之后，又通过其他渠道形成订单。出现这种问题的根源是收入转化没有特定的标识符号用来做会员营销活动区分，因此，在做会员营销时一定要尽量让用户有特定的标志，这样才能区分营销效果。

4.用券会员/金额/订单比例

会员营销时大多数情况下都会使用优惠券，这不仅是促销销售的一种方式，也是识别不同会员订单来源的重要途径。关于用券类指标包括：

·用券会员比例：使用优惠券下单的会员占总下单会员的比例；

·用券金额比例：使用优惠券下单的订单金额占总下单金额的比例；

·用券订单比例：使用优惠券下单量占总下单量的比例。

除此以外，还包括基于用券数据产生的用券用户平均订单金额、用券用户复购率等相关指标。

5.营销费率

营销费率是会员营销费用占营销收入的比例。营销费率分析的目的是监督营销费用的支出情况，确保其不超出计划指标。

6.每注册/订单/会员收入

监控会员营销的单位收入是评估收益效率的重要指标，包括：

·每注册收入：每个注册用户带来多少收入；

·每订单收入：每个订单带来多少收入；

·每会员收入：每个会员带来多少收入。

7.每注册/订单/会员成本

单位成本的考量是精细化业务动作的关键指标之一，包括：

·每注册成本：每个获得一个注册用户需要多少成本；

·每订单成本：每个获得一个订单需要多少成本；

·每会员成本：每个获得一个会员需要多少成本。

除了上述单位成本指标外，还可能包括其他类型的成本，例如每挽回一个流失客户成本、每完成一个特定目标成本（例如下载企业白皮书）、每单位线索成本（例如获得一个联系方式）等。

5.2.3　会员活跃度指标

1.整体会员活跃度

整体会员活跃度用来评价当前所有会员的活跃度情况，通常以会员动作或关键指标作为会员是否活跃的标识（如是否登录）。在此介绍一个会员活跃度矩阵，通过业务定义的关键因素来判断整体会员活跃度（因素及权重可根据企业自身实际情况定义）。

表5-1列出了所有会员关键动作节点和指标因素，并标识了每个因素的取值范围及权重。当用户登录/注册后（标识会员的前期条件），所有会员的行为都会被记录下来，形成会员数据日志。对每个会员的活跃度数据加权处理后求和得到整体会员活跃度得分。

整体会员活跃度=∑（注册×1+登录×1+验证×1+等级数×1+积分×1+…+/商品评价×1）

表5-1　用户活跃度定义矩阵

行为编码	取值范围	行　为	行为类型	权　重
Y/N	1	注册	账户行为	1
Y/N	1	登录	账户行为	1
Y/N	≥1	EMAIL 验证 / 手机验证 / 支付密码验证	账户行为	1
等级数	≥1	升级会员	账户行为	1
使用次数	≥1	使用积分	账户行为	1
使用次数	≥1	使用优惠券	账户行为	1
订阅次数	≥1	订阅信息	互动行为	1
访问页面数	≥1	访问页面	互动行为	1

(续)

行为编码	取值范围	行　为	行为类型	权　重
搜索次数	≥1	搜索	互动行为	2
查看多少次	≥1	查看商品	互动行为	2
次数	≥1	页面咨询	互动行为	1
次数	≥1	收藏商品	互动行为	2
次数	≥1	商品比价	互动行为	2
次数	≥1	到货通知	互动行为	1
次数	≥1	页面纠错	互动行为	1
次数	≥1	加入购物车	订单行为	1
次数	≥1	在线下单	订单行为	1
次数	≥1	取消订单	订单行为	-1
次数	≥1	换货订单	订单行为	-1
次数	≥1	退货订单	订单行为	-1
次数	≥1	订单完成	订单行为	1
次数	≥1	参与活动	订单行为	1
次数	≥1	商品评价	分享行为	1/0/-1

举例：网站有两个用户，其中一个会员完成了1次注册、1次邮件验证（新会员默认是1级会员）、查看2次商品并有1次收藏行为；另一个会员是老会员（假设为2级会员），完成了1次登录、1次页面咨询和1次退货订单，那么该网站（在假设只有2个会员的基础上）的用户活跃度

为：新会员（1×1+1×1+2×2+1×2+1×1）+老会员（1×1+1×1-1×1+2×1）＝12。通过对每个用户的活跃度以及网站整体活跃度的积累可以发现网站用户活跃度变化趋势。

2.每日/每周/每月活跃用户数

活跃用户中“活跃”的定义在不同公司有不同的方法。最早活跃用户的概念从APP中产生，指的是每日应用上的仍然启用的用户数量。活跃用户根据活跃周期的不同可以定义为：

·每日活跃用户（Daily Active Users daily，DAU）：每天活跃用户的数量。

·每周活跃用户（Weekly Active Users daily，WAU）：每周活跃用户的数量。

·每月活跃用户（Monthly Active Users daily，MAU）：每月活跃用户的数量。

这三个指标在对应的时间周期内重复，即当有用户多次完成事件时会在周期内只计算一次。下面以一个示例说明每日活跃用户和每周活跃用户的差异，如表5-2所示。

表5-2显示了在不同日期内，每日活跃用户的数量等于完成活跃动作或行为的用户数，而每周活跃用户数在1周内对用户做去重。

表5-2　用户活跃度定义

日	活 跃 用 户	每日活跃用户	每周活跃用户
第 1 天	A	1	1
第 2 天	A 和 B	2	2
第 3 天		0	2
第 4 天	A	1	2
第 5 天		0	2
第 6 天	A	1	2
第 7 天	A	1	2

5.2.4 会员价值度指标

1.会员价值分群

会员价值分群是以用户价值为出发点，通过特定模型或方法将会员分为几个群体或层级。常见的分群结果例如：高、中、低；钻石、黄金、白银、青铜等。

会员价值分群并不是一个真正的指标，而是给用户打标签，该标签用来显示用户的状态、层次和价值区分等。

2.复购率

复购率是一定周期内购买2次或2次以上的会员比例。不同公司对复购率的定义有所差异，基本定义逻辑分为三种，现以1个月为周期说明复购的定义：

·第一种：1个月内购买2次或2次以上的会员；

·第二种：1个月内购买2次或2次以上，以及1个月之前有购买行为，在1个月之内又产生购买行为（可能是1次）的会员；

·第三种：1个月之前有购买行为，1个月之内又有购买行为的会员。

以上三种定义可根据自身情况调整，同时1个月的时间周期也可根据商品或服务销售频次进行重新定义。

3.消费频次

消费频次跟复购相关，二者都是重复消费指标。消费频次是将用户的消费频率，按照次数做统计，统计结果是在一定周期内消费了不同次数，例如2次、3～5次、6～10次、11次以上。该指标可以有效分析用户对于企业的消费黏性。

4.最近一次购买时间

最近一次购买时间的含义就是字面意义，该指标也可以作为会员消费价值黏性的评估因素。如果会员距离上次的购买或消费时间过长，那么意味着用户可以在沉默或将要流失甚至已经流失的阶段，此时应该采取措施挽回用户。

5.最近一次购买金额

最近一次购买金额和最近一次购买时间类似，该指标衡量的是用户最近一次购买或消费时的订单，该金额越大说明用户最近一次的消费能力越高。根据二八法则，20%的老会员会贡献80%的消费金额。

5.2.5 会员终生价值指标

1.会员生命周期价值/订单量/平均订单价值

会员生命周期指标是从用户成为企业会员开始到现在的总数据统计值，该指标与任何时间周期无关，衡量的是用户完整生命周期内的价值。包括：

·会员生命周期价值（Customer Lifetime Value，CLV）：用户整个生命周期内下单金额总和。

·会员生命周期订单量：用户整个生命周期内下单量总和。

·会员生命周期平均订单价值：用户整个生命周期内下单金额/下单量。

会员生命周期相关指标由于突破了时间的限制，能从整体上获得会员的宏观状态，因此是重要到宏观价值衡量指标。

2.会员生命周期转化率

会员生命周期转化率指会员在完整生命周期内完成的订单和到达网站/企业/门店的次数比例，该指标衡量了用户是否具有较高的转化率。例如：用户一次达到网站100次，但是只有1次消费，那么会员生命周期转化率为1%。

3.会员生命周期剩余价值

会员生命周期剩余价值是一类预测性的指标，用来预测用户在生命周期内还能产生多少价值。该指标可以细分出很多相关指标，例如：

·预期未来30天的会员转化率；

·预期生命周期剩余订单价值；

·预期7天内下单数量；

·预计下1个订单的订单金额；

·下一次购买的商品名称。

这种预测性的指标通常会基于特定的算法和模型做训练，然后预测未来的数据，其中回归和分类是主要预测性应用方法，某些情况下也可以使用关联算法。

5.2.6　会员异动指标

1.会员流失率

会员流失指会员不再购买或消费企业相关业务、商品和服务，会员流失率指流失的会员数量与全部会员数量间的比例。会员流失是一个正常现象，没有任何一个企业能做到不让一个会员流失。但是会员流失意味着企业会失去相应的利润来源，因此需要从2个方面重点关注该指标：

·会员流失率的数值。正常情况下会员流失率应该是一个比较小的比例，不同行业有不同的基准，各企业要制定符合行业特定的基准作为参考。

·会员流失率的走向。尽管会员流失不可避免，但我们仍然希望流失用户的比例越小越好，因此需要关注流失率的走向。比较好的状态是流失率处于平稳或下降状态，如果出现流失率的上升则需要引起警惕。

2.会员异动比

会员异动比指新增会员与流失会员之间的比例关系，计算公式为：会员异动比=新增购买会员/流失会员。

如果会员异动比等于1，说明企业一定周期内新增会员与流失会员数相等；如果大于1，说明新增会员多于流失会员，这是良好的发展状态；如果小于1，说明会员增长不如流失快，企业面临会员枯竭危机。

5.3 会员数据化运营应用场景

会员数据化运营主要应用于会员营销和会员关怀两方面。

5.3.1 会员营销

数据化运营应用于会员营销主要体现在以下几个方面：

·以信息化的方式建立基于会员的客户关系管理系统，促进所有会员数据的信息化；

·通过特定方法将普通用户拓展企业会员，并提高新会员留存率；

·基于用户历史消费记录，挖掘出用户潜在消费需求及消费热点；

·基于历史数据，为会员营销活动提供策略指导和建议，促进精准营销活动的开展；

·从会员营销结果中寻找异常订单或转化，作为黄牛或VIP客户识别的参考；

·挖掘会员传播关系，找到口碑传播效应的关键节点。

5.3.2 会员关怀

数据化运营应用于会员关怀主要体现在以下几个方面：

·为预警事件设置阈值，自动触发应急处理机制；

·分析会员行为，为会员提供个性化、精准化和差异化服务；

·通过会员喜好分析，提高客户忠诚度、活跃度和黏性；

·通过会员分析，预防会员流失并找到挽回已经流失会员的方法；

·基于会员群体行为，更好的划分会员群体属性并挖掘群体性特征；

·基于群体用户和内容相似度，发现有价值的会员互动方式；

·基于会员生命周期的关怀管理，促进用户终生价值最大化。

5.4 会员数据化运营分析模型

在会员数据化运营分析模型中，将主要介绍会员细分模型、会员价值度模型、会员活跃度模型、会员流失预测模型、会员特征分析模型和营销响应预测模型。

5.4.1 会员细分模型

会员细分模型是将整体会员划分为不同的细分群体或类别，然后基于细分群体做管理、营销和关怀。会员细分模型常用于在整体会员的宏观性分析以及探索性分析，通过细分建立初步认知为下一步的分析和应用提供基本认知；会员细分也是做精准营销的基本前提。

常用的细分模型包括：基于属性的方法、ABC分类法、聚类法等。

（1）基于属性的方法

会员细分可以基于现有会员属性，常用的细分属性包括：会员地域（例如北京、上海、武汉等）、产品类别（例如大家电、3C数码、图书等）、会员类别（例如大客户、普通客户、VIP客户等）、会员性别（例如男、女、未知）、会员消费等级（例如高价值会员、中价值会员、低价值会员）、会员等级（例如钻石、黄金、白银）等。这种细分方法可以直接利用现有会员数据库数据，无需做二次开发和计算，是一种比较简单且粗犷的方法。

（2）ABC分类法

ABC分类法（Activity Based Classification）是根据事物的主要特征做分类排列，从而实现区别对待、区别管理的一种方法。ABC法则是帕累托二八法则衍生出来的一种法则。不同的是，二八法则强调的是抓住关键，ABC法则强调的是分清主次，并将管理对象划分为A、B、C三类。

在ABC分析法中先将目标数据列倒叙排序，然后做累积百分比统计，最后将得到的累积百分比按照下面的比例值划分为A、B、C三类：

·A类因素：发生累计频率为0%～80%，是主要影响因素。

·B类因素：发生累计频率为80%～90%，是次要影响因素。

·C类因素：发生累计频率为90%～100%，是一般影响因素。

下面以示例数据说明如何使用ABC分类法对会员做细分。

步骤1　先建立一个二维表格数据，数据中包括会员ID和订单金额（或其他关键指标）两列。

步骤2　二维表格数据按照订单金额做倒叙排序。

步骤3　对订单金额列做累积百分比统计。

步骤4　按照A、B、C划分标准将会员划分为不同的分类，得到如表5-3所示数据。

表5-3　用户活跃度定义

用户 ID	订单金额	订单金额累积百分比	ABC 分类
228355	55960719	29.44%	A
260835	28382998	44.36%	A
231664	22995042	56.46%	A
204075	20582024	67.29%	A
189610	16093666	75.75%	A
205016	14543748	83.40%	B
327963	14393922	90.97%	C
189334	13752004	98.21%	C
234680	3382530	99.99%	C
266283	28119	100.00%	C

（3）聚类法

使用聚类法做会员分群是常用的非监督式方法，该方法无需任何先验经验，只需要指定要划分的群体数量即可。有关聚类分析的具体操作方法，请查看4.1.6节。

5.4.2 会员价值度模型

会员价值度用来评估用户的价值情况，是区分会员价值的重要模型和参考依据，也是衡量不同营销效果的关键指标之一。价值度模型一般基于交易行为产生，衡量的是有实体转化价值的行为。常用的价值度模型是RFM。

RFM模型是根据会员最近一次购买时间R（Recency）、购买频率F（Frequency）、购买金额M（Monetary）计算得出RFM得分，通过这三个维度来评估客户的订单活跃价值，常用来做客户分群或价值区分。该模型常用于电子商务（即交易类）企业的会员分析。

RFM模型基于一个固定时间点来做模型分析，因此今天做的RFM得分跟7天前做的结果可能不一样，原因是每个客户在不同的时间节点所得到的数据不同。以下是RFM模型的基本实现过程：

步骤1　设置要做计算时的截止时间节点（例如2017-5-30），用来做基于该时间的数据选取和计算。

步骤2　在会员数据库中，以今天为时间界限向前推固定周期（例如1年），得到包含每个会员的会员ID、订单时间、订单金额的原始数据集，一个会员可能会产生多条订单记录。

步骤3　数据预计算。从订单时间中找到各个会员距离截止时间节点最近的订单时间作为最近购买时间；以会员ID为维度统计每个用户的订单数量作为购买频率，将用户多个订单的订单金额求和得到总订单金额。由此得到R、F、M三个原始数据量。

步骤4　R、F、M分区。对于F和M变量来讲，值越大代表购买频率越高、订单金额越高；但对R来讲值越小代表离截止时间节点越近，因此值越好。对R、F、M分别使用五分位（三分位也可以，分位数越多划分的越详细）法做数据分区，需要注意的是，对于R来讲需要倒过来划分，离截止时间越近的值划分越大。这样就得到每个用户的R、F、M三个变量的分位数值。

步骤5　将三个值组合或相加得到总的RFM得分。对于RFM总得分

的计算有两种方式，一种是直接将三个值拼接到一起，例如RFM得分为312、333、132；另一种是直接将三个值相加求得一个新的汇总值，例如RFM得分为6、9、6。

在得到不同会员的RFM的之后，根据步骤5产生的两种结果有不用的应用思路：

思路1：基于三个维度值做用户群体划分和解读，对用户的价值度做分析。例如得分为212的会员往往购买频率较低，针对购买频率低的客户定期发送促销活动邮件；针对得分为321的会员虽然购买频率高但是订单金额低等，这些客户往往具有较高的购买黏性，可以考虑通过关联或搭配销售的方式提升订单金额。

思路2：基于RFM的汇总得分评估所有会员的价值度价值，并可以做价值度排名；同时，该得分还可以作为输入维度跟其他维度一起作为其他数据分析和挖掘模型的输入变量，为分析建模提供基础。

上述示例中模型的三个维度权重是相同的，可以根据不同企业的需求为RFM设置不同权重值，然后通过加权的形式得到符合运营需求的得分。

5.4.3 会员活跃度模型

会员活跃度用来评估用户的活跃度情况，是会员状态分析的基本模型之一。在5.2.3节的整体活跃度指标中介绍了一种基于加权统计的方法，在此再介绍另一种活跃度模型——RFE模型。

RFE模型基于用户的普通行为（非转化或交易行为）产生，它跟RFM类似都是使用三个维度做价值评估。RFE模型是根据会员最近一次访问时间R（Recency）、访问频率F（Frequency）和页面互动度E（Engagements）计算得出的RFE得分。其中：

·最近一次访问时间R（Recency）：会员最近一次访问或到达网站的时间。

·访问频率F（Frequency）：用户在特定时间周期内访问或到达的频率。

·页面互动度E（Engagements）：互动度的定义可以根据不同企业的交互情况而定，例如可以定义为页面浏览量、下载量、视频播放数量等。

在RFE模型中，由于不要求用户发生交易，因此可以做未发生登录、注册等匿名用户的行为价值分析，也可以做实名用户分析。该模型常用来做用户活跃分群或价值区分，可用于内容型（例如论坛、新闻、资讯等）企业的会员分析。

RFM和RFE模型的实现思路相同，仅仅是计算指标发生变化。对于RFE的数据来源，可以从企业自己监控的用户行为日志获取，也可以从第三方网站分析工具获得。

在得到用户的RFE得分之后，跟RFM类似也可以有两种应用思路：

思路1：基于三个维度值做用户群体划分和解读，对用户的活跃度度做分析。RFE得分为313的会员说明其访问频率低，但是每次访问时的交互都非常不错，此时重点要做用户回访频率的提升，例如通过活动邀请、精准广告投放、会员活动推荐等提升回访频率。

思路2：基于RFE的汇总得分评估所有会员的活跃度价值，并可以做活跃度排名；同时，该得分还可以作为输入维度跟其他维度一起作为其他数据分析和挖掘模型的输入变量，为分析建模提供基础。

无论是RFM和RFE都不要忽略不同的消费频率、品类和周期对于结果的影响性。例如大家电的更换周期可能是2年、手机的更换频率是1年、日用消费品的周期却是7天，由于不同品类的差异性很大，最终得到的得分结果没有必然的可比性，例如偏向于购买大家电品类的RFM得分为113属于“正常现象”，因为大家电的购买属性决定了这就是一个长周期、低频、大金额的行为。

5.4.4 会员流失预测模型

会员流失预测模型用来预测会员是否流失，是做会员生命周期管理的重要预防性应用。做会员流失模型的关键因素之一是要定义好“流失”，即处于何种状态、具备哪些特征的会员属于流失会员；另外，流失也可能区分是永久性流失还是临时性流失。常见的属于流失的状态定义示例：

·会员已经退订公司的促销活动；

·会员打电话要求将自己的信息加入通知黑名单；

·会员已经连续6个月没有登录过网站；

·针对会员发送的关怀激励活动中没有任何有效反馈和互动；

·会员最近1年内没有任何订单。

上述流失状态可以归为两类：一类是会员有明确的表达，不再希望接收到公司的相关信息；另一类是会员没有明确的表示，但是在业务关注的主要领域内，没有得到有效反馈。

会员流失预测模型的实现方法属于分类算法，常用算法包括逻辑回归、支持向量机、随机森林等，有关这些算法的具体选择问题，请参照4.3.5节。

在做会员流失预警模型时，需要注意以下几个问题：

·流失会员的样本分类一定是少数类，需要注意处理样本不均衡问题。

·对于流失会员的预测结果，得到概率性的输出可以结合流失预测标签一起应用，因为业务方可以基于概率再结合业务经验做判断。

·对于参与训练模型的维度变量的选择，一定要结合业务经验，因为业务方对于特定场景的判断是影响训练模型和应用结果的关键因素之一。

·输入的维度变量中一定要包含发生转化前的行为数据，假如业务定义为最近6个月没有订单的客户为流失客户，那么在做预测模型时需要将用户的匿名访问、登录、页面浏览、搜索、活动咨询等转化前的数据考虑在内，而不能只考虑订单转化本身。

·会员流失预警模型不是一次性的，而是周期性监视和运行的，例如每天、每周或至少是每月。

通过会员流失模型得到每个会员是否属于流失标签后，可以将该结果给到会员运营人员，运营人员一般会根据业务经验做二次审查和确认，然后再通过会员挽回、激励等机制提升会员的忠诚度，延缓或防止会员流失。而关于如何挽回以及激励的问题，通常也是需要数据参与来帮助运营人员制定相应的策略，例如在合适的时间、以恰当的方式提供个性化的内容给特定会员，这些都需要数据的支持。

5.4.5　会员特征分析模型

会员特征分析模型是针对现有会员做特征分析。会员特征分析模型提供的结果可能是模糊的，也可能是明确的。例如：

·明确的特征，它提供了业务所要行动的细节要素，是一种具有极高落地价值的数据分析工作。

·模糊的特征，它指数据分析结果未提供详细的动作因素，仅指明了下一步行动方向或目标。

会员特征分析主要应用于以下两种业务场景。

第一种是在没有任何前期经验或特定目标下触发，希望通过整体特征分析了解会员全貌。在这种模式下，可以通过一定方法先将用户划分为几个类别，然后再做基于类别的特征分析，常用实现方法和应用包括：

·聚类：通过聚类将用户划分为几个群组，然后再分析不同群组的典型特征和群组间的差异性。例如：公司的总体会员具有哪些特征？模型结果：通过聚类方法将会员划分为3类，然后每个类别都有各自显著性特征，会员部门可根据不同类别做特定分析并指定群体性策略。

·统计分析：先将整体用户做统计分析，包括描述性统计、频数分布等，了解整体数据概括。

第二种是有明确的业务方向，希望找到能达到事件目标的会员特征，用于做进一步的会员运营。对于这类分析模型，常用的实现方法和应用包括：

·分类：利用分类规则例如决策树找到符合目标的关键变量以及对应的变量值，进而确定会员特征。例如，收入＞5400元，最近购买时间是5个月之前，总订单金额在4300元以下的会员最可能购买商品。

·关联：使用关联规则找到不同属性、项目间的关联发生或序列发生关系，然后将会员的属性特征（频繁项集）提供给运营。例如：购买

X商品的客户一般是来自于上海、购物频率为1周3次、客单价为100元以下。

·异常检测：使用非监督式的异常检测方法，从一堆数据中找到异常数据样本，然后将这些数据样本特征提供给运营做进一步确认和审查。例如：异常客户的特征往往是每次订单的商品数量超过4件、地域集中在江苏和浙江、一般拥有超过3个以上的子账户。

会员特征分析模型输出的上述两类结果，第一类结果往往作为辅助于、启发性和提示性结果，用于为运营提供进一步业务动作的思考，这种一般开始于数据工作项目的开始或业务方对数据主题的先验经验不足的情况下；第二类结果则可以作为运营下一步动作的直接“触点”。

5.4.6 营销响应预测模型

营销响应预测模型是针对营销活动展开的，通常在做会员营销活动之前，通过营销响应预测模型分析找到可能响应活动的会员特征以及整体响应的用户比例、数量和可能带来的销售额。这在会员营销之前的有关策略制定的辅助价值非常明显。

营销响应预测模型的实施采用的一般是分类算法，常见算法包括，常用算法包括逻辑回归、支持向量机、随机森林等，有关这些算法的具体选择问题，请参照4.3.5节。

在做营销响应模型之前，需要先收集训练所需的数据集。

步骤1　从所有会员上随机选择一定量的会员样本，具体数量要根据企业实际情况而定，一般情况下，至少要有1000条数据以上（同时要兼顾总体会员数量）才能满足模型训练的需要。

步骤2　然后针对选择的会员样本通过一定媒介和渠道发送营销活动信息，例如手机短信、电子邮件等。需要注意的是，一定要记录好营销活动发送的时间、频率、信息等关键运营要素，这些需要跟后期的实施保持一致。

步骤3　收集营销活动数据。在收集数据时需要注意数据收集的周期，通常情况下，一般电子邮件的有效周期为1～7天，时间过短可能无法被用户看到；手机短信的有效期一般是1天左右，时间太长用户一般会忽略。

经过上述步骤收集到分类所需的样本集之后，接着就需要通过分类模型做营销响应预测，这是典型的二分类问题。在做营销响应模型训练时，也需要注意在5.4.4节提到的问题，二者在很多方面都有共通性。

通过营销响应预测模型得到的结果一般包括两个方向：

·基于模型找到最可能产生购买转化行为的会员规则特征。例如最近一次购买时间在3个月以内、会员等级为3级以上、总订单金额大于3000、订单量大于10的客户。通过这些条件直接从数据库中筛选对应的

会员列表，并可以对该会员列表发送营销活动。

·基于模型预测可能产生的订单转化数量、转化率（例如选择10000个客户，会有4000个客户产生转化），以及有转化客户的客单价（通过训练样本集选择有转化客户，然后用订单金额/会员量计算得到）大体计算出此次发送会员能得到的营销收入。这些信息可以作为此次营销活动计划提报的数据量化指标和资源申请的数据支持。

5.5 会员数据化运营分析小技巧

在做会员数据化运营分析时，有一些小技巧和小方法，将在本节中简单介绍。

5.5.1 使用留存分析新用户质量

用户留存指的是新会员/用户在经过一定时间之后，仍然具有访问、登录、使用或转化等特定属性和行为，留存用户占当时新用户的比例就是留存率。留存率按照不同的周期分为三类，以登录行为认定的留存为例：

第一种　日留存，日留存又可以细分为以下几种：

·次日留存率：（当天新增的用户中，第2天还登录的用户数）/第一天新增总用户数

·第3日留存率：（第一天新增用户中，第3天还有登录的用户数）/第一天新增总用户数

·第7日留存率：（第一天新增用户中，第7天还有登录的用户数）/第一天新增总用户数

·第14日留存率：（第一天新增用户中，第14天还有登录的用户数）/第一天新增总用户数

·第30日留存率：（第一天新增用户中，第30天还有登录的用户数）/第一天新增总用户数

第二种　周留存，以周度为单位的留存率，指的是每个周相对于第一个周的新增用户中，仍然还有登录的用户数。

第三种　月留存，以月度为单位的留存率，指的是每个月相对于第一个周的新增用户中，仍然还有登录的用户数。

留存率是针对新用户的，其结果是一个矩阵式半面报告（只有一半有数据），每个数据记录行是日期、列为对应的不同时间周期下的留存率。正常情况下，留存率会随着时间周期的推移而逐渐降低，如图5-1所示。

在应用留存分析时，需要注意以下几个问题：

·区别应用不同留存周期。日留存用来做短期结果、周留存用来看中期效果、月留存用来看长期效果。

·在留存中注意观察和分析衰减比率，正常情况下的留存会随着时间逐渐衰减，这种衰减趋势可能呈线性、指数性甚至二项式等不同趋势，通过衰减数据可以得到衰减模型，这些模型可以用来做衰减异常检测，以发现哪些时间的衰减存在异常（通常是过渡衰减）。

·注意分析运营活动对于留存的影响，在没有较大的运营活动时，留存的衰减会存在一定规律；但当运营采取一定活动时可能会导致留存率的提升，而这种提升应该是预期内的。如果留存中没有反映出提升趋势，需要多方面总结运营活动效果。

用户月度留存报告-%	留存% (month 0)	留存% (month 1)	留存% (month 2)	留存% (month 3)	留存% (month 4)	留存% (month 5)	留存% (month 6)	留存% (month 7)	留存% (month 8)	留存% (month 9)	留存% (month 10)	留存% (month 11)
2016-05	100 %	4.72 %	2.94 %	2.46 %	1.86 %	1.57 %	1.81 %	1.86 %	1.75 %	1.98 %	1.60 %	0.90 %
2016-06	100 %	9.72 %	8.07 %	7.64 %	4.79 %	5.16 %	5.44 %	4.59 %	4.65 %	4.76 %	2.80 %	- %
2016-07	100 %	3.20 %	2.74 %	2.39 %	1.84 %	2.41 %	1.72 %	2.05 %	1.93 %	1.08 %	- %	- %
2016-08	100 %	2.42 %	1.83 %	1.62 %	1.79 %	1.72 %	1.78 %	1.36 %	0.50 %	- %	- %	- %
2016-09	100 %	8.35 %	5.83 %	6.73 %	5.85 %	5.87 %	5.09 %	2.00 %	- %	- %	- %	- %
2016-10	100 %	5.56 %	4.01 %	3.65 %	3.71 %	3.75 %	1.45 %	- %	- %	- %	- %	- %
2016-11	100 %	4.59 %	3.11 %	3.21 %	2.05 %	0.70 %	- %	- %	- %	- %	- %	- %
2016-12	100 %	3.68 %	3.82 %	3.47 %	0.75 %	- %	- %	- %	- %	- %	- %	- %
2017-01	100 %	10.01 %	8.30 %	3.26 %	- %	- %	- %	- %	- %	- %	- %	- %
2017-02	100 %	11.64 %	3.33 %	- %	- %	- %	- %	- %	- %	- %	- %	- %
2017-03	100 %	3.69 %	- %	- %	- %	- %	- %	- %	- %	- %	- %	- %
2017-04	100 %	- %	- %	- %	- %	- %	- %	- %	- %	- %	- %	- %

图5-1　月度留存报告

5.5.2 使用AARRR做APP用户生命周期分析

AARRR是Acquisition、Activation、Retention、Revenue、Refer五个单词的缩写，分别对应用户生命周期中的5个环节：获取用户、提高活跃度、提高留存率、获取收入、自传播。

获取用户：获取用户是第一步，解决的是从哪里能带来更多的用户和会员。该部分的数据评估维度一般包括两个层次：用户数量和用户质量。

·用户数量：用户数量是获得用户的数量，关于用户数量的统计有多种维度和方法。例如以设备唯一ID作为唯一用户标识；以匿名Cookid作为唯一用户识别标志；以用户实名ID作为唯一用户标志，例如会员ID、电子邮件、手机号等。

·用户质量：用户质量的评估会涉及多个方面，例如访问频率、启动次数、浏览深度、停留时间、目标转化率、付费转化率、收入等单维度指标以及活跃度、价值度等复合模型指标。

提高活跃度：提升活跃度是将用户引入后持续关注的问题。分析活跃的数据评估维度包括每日/每周/每月活跃用户数三种，关于该部分内容请见5.2.3节。在做活跃度对比时，除了企业内部基于不同时间的对比，还需要横向跟行业内的其他企业做标杆对比，这样才能了解企业活跃度在行业内的水平。

提高留存率：提升留存率意味着会有更多的客户沉淀下来。有关留存率的具体问题，请见“5.5.1使用留存分析分析新客户质量”。

获取收入：获取收入是运营的根据目标。有关收入的数据支撑，一般会通过以下几个指标来做分析：

·付费用户数/比例：产生收入的用户数及其占整体的比例。

·ARPU：平均每用户收入，衡量每个用户的付费能力。

·新增付费用户：新产生的付费用户数。

·付费转化周期：用户从免费到付费的转化时间，时间越短越好。

·重复消费比例：具有2次及以上消费的用户比例。

·消费金额在特定金额以上的客户：通常用来分析VIP客户或大客户，例如消费金额在10000以上。

·消费分级：根据不同的消费数据做消费分级，用来整体划分会员群体。

自传播：如果用户使用产品时体验良好，那么可能会产生传播效应，把产品推荐给其他朋友、亲戚等使用。自传播主要分为两个方面：

·产品引导传播：这种一般是在产品内部有推送信息提示用户做产品传播，可能同步需要配合不同的激励措施以达到推动传播的效果。

·用户主动传播：用户自发地将相关产品应用截图、数据、产品等分项到社交圈。

对这两种传播都需要在产品应用内部为不同的用户和传播场景做标记，以便于下一步分析，常用标记数据包括：分享内容、主动传播/被动传播、激励措施等。

上述所有的数据采集一般通过三种方式配合实现：

·APP行为检测：对于APP产品应用内部的用户行为，通过埋点的方式将不同用户行为记录下来。

·APP交易检测：一般的APP内部也可以做交易检测，但更多（以及更准确）的还是通过企业内部的销售系统做数据收集。

·APP外部检测：主要用来检测用户在APP之外的信息，例如分享、评论、转发等。

5.5.3 借助动态数据流关注会员状态的轮转

在做会员状态分析时，通常得到的是某个时间点状态。但是在不同时间周期下，会员的状态会发生改变。基于动态的时间周期，可以有效分析用户的状态轮转变化，可以从整个周期的视角发现会员状态的全貌。

例如，通过价值度模型定义好不同的价值度群体标准，然后每个客户都有一个价值度标签，基于一个时间节点开始，向后可以分析所有用户在不同时间周期下的价值度的变迁和轮转情况，效果如图5-2的桑基图所示。

在桑基图中，可以看到不同价值度之下用户的状态变化，同时可以结合动态时间轴做特定时间周期的分析。这种分析相对于柱形图或趋势图的数据统计，可以明显看到不同状态的用户轮转变换的比例和大小，而不仅是一个结果值，这种中间的转换状态才是动态分析的重点。

图5-2　用户活跃度状态轮转

5.5.4 使用协同过滤算法为新会员分析推送个性化信息

在针对新会员的运营过程中，由于新会员的数据量少，因此无法根据其历史数据针对性的提供个性化、精准化的运营活动。此时，除了可以采用TOP榜（例如TOP浏览商品、TOP销售商品等）的固定推送机制外，还可以基于相似会员的喜好来做推荐。

基于相似会员的信息挖掘有很多种方法，在此介绍基于用户的协同过滤算法。协同过滤（Collaborative Filtering，CF）是利用集体智慧的一个典型方法，常被用于分辨特定对象（通常是人）可能感兴趣的项目（项目可能是商品、资讯、书籍、音乐、帖子等），这些感兴趣的内容来源于其他类似人群的兴趣和爱好，然后被作为推荐内容推荐给特定对象。基于用户的协同过滤是利用的人群的相似度为基础做推荐。

协同过滤主要解决的问题是当客户进入某个领域后，什么内容或项目是他/她可能感兴趣的东西，然后以用户的兴趣为出发点推荐他/她可能感兴趣的内容，以此来提高用户体验、用户交互频率提升、订单转化效果、销售利润提升等。

协同过滤目前主要用于电子商务网站、兴趣部落网站、知识性网站、话题型网站、社交性网站的个性化项目推荐。协同过滤推荐的场景通常发生在，当客户对内容进行打分的前提下，例如内容评分、综合评价等。

举例：如图5-3所示是基于用户的协同过滤机制。假设用户A喜欢物品A、C，用户B喜欢物品B，用户C喜欢物品A、C、D；从用户的历史喜好信息中，我们可以发现用户A和用户C的口味和偏好比较类似，因此可以将物品D推荐给用户A。

使用基于用户的协同过滤算法时，主要步骤如下：

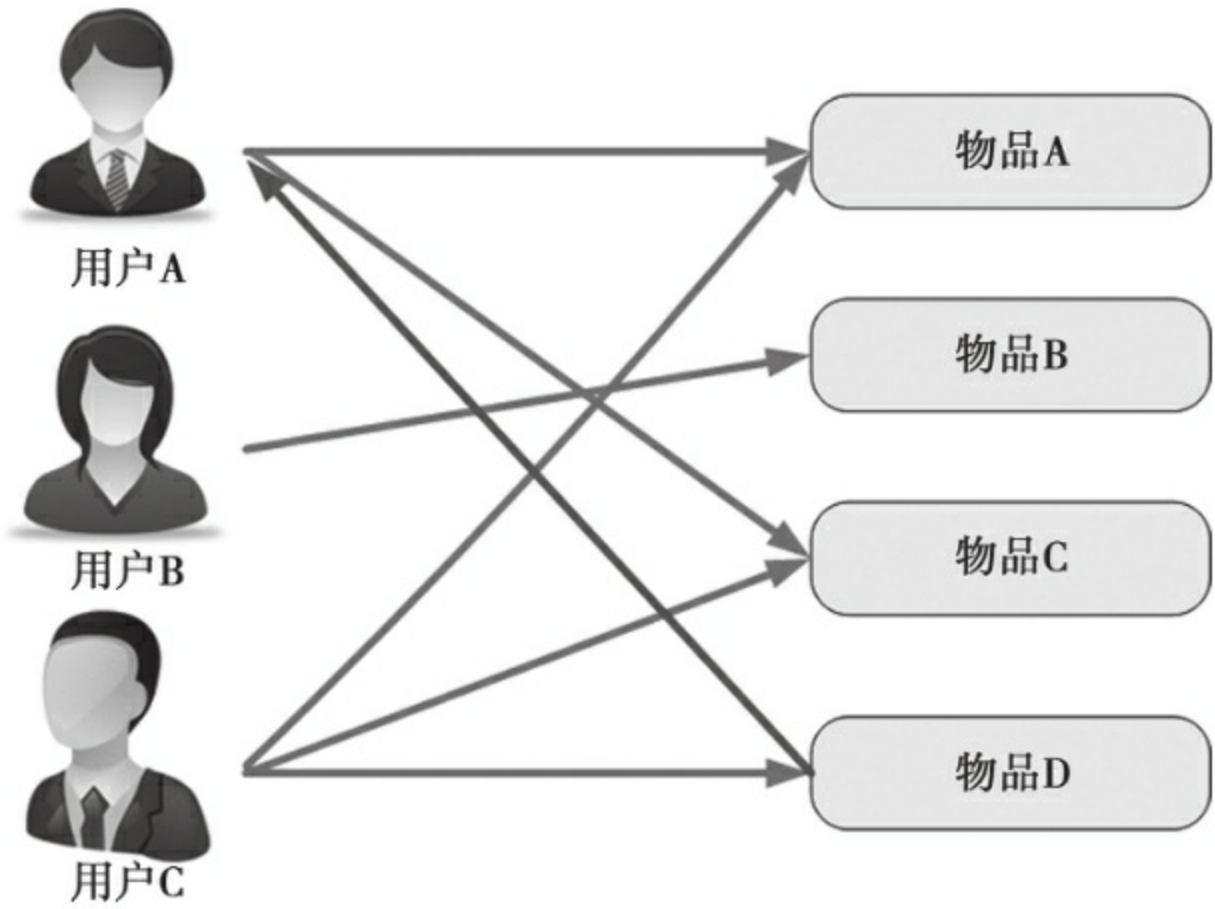


图5-3　基于用户的协同过滤算法

步骤1　收集用户数据。一般情况下对会员数据的采集包括属性、行为等多个方面，例如性别、年龄、地域、订单量、购买商品、广告渠道来源、订单金额等。

步骤2　会员相似度计算。以新会员为目标对象，计算其他所有会员跟新会员的相似度并获得相似度得分。关于相似度的算法有很多种，例如：

·基于几何距离的“相似度”算法：欧式距离、曼哈顿距离、切比雪夫距离等。

·基于非几何距离的“相似度”算法：余弦距离、汉明距离、杰卡德相似系数、相关系数等。

步骤3　将其他所有会员跟目标新会员的相似性得分按倒序排序，

得分最高者为最相似的会员。但是实际情况是我们会选择有K个最相似的用户，使用类似于K近邻的方法。例如，选择最相似的5个会员。

步骤4　选择最相似K个会员的目标推荐信息。在选择目标信息的过程中，对于不同会员的信息统计有多种方法，例如普通计数、加权汇总等，一般使用加权汇总的方式。例如选择最相似的5个会员最经常浏览的商品（假设是针对商品浏览做推荐），他们各自最经常浏览的商品列表如下：

·会员1：PID1234

·会员2：PID1255

·会员3：PID1256

·会员4：PID1255

·会员5：PID1234

同时，我们会有一个针对不同相似度会员的权重表，用来将相似度排名和会员的数量结合起来，形成如表5-4所示的数据。

基于表5-4的结果对商品ID做汇总统计，得到如表5-5所示的结果。

表5-4　用户活跃度定义

会员 ID	商品 ID	权重	会员 ID	商品 ID	权重
会员 1	PID1234	0.3	会员 4	PID1255	0.15
会员 2	PID1255	0.25	会员 5	PID1234	0.1
会员 3	PID1256	0.2			

表5-5　推荐结果数据

商品 ID	商品得分	商品 ID	商品得分	商品 ID	商品得分
PID1234	0.4	PID1255	0.4	PID1256	0.2

基于上述结果，可为该新会员推荐商品ID为PID1234或PID1255商品。

协同过滤属于个性化推荐算法的一种，并不是个性化推荐的全部，除了协同过滤算法外，还有多种算法可以在不同场景下满足不同的推荐需求，例如关联算法、KNN、TOPN、深度学习等算法。另外，推荐算法也只是推荐系统中的一部分，除了推荐算法模块外，还可能包括实时数据接入和流处理、DMP数据管理、用户行为建模和画像标签库、项目标签库、冷启动场景规则、异常场景规则、强制人工干预规则、模型和数据修正模块、场景引擎模块、投放引擎模块、机器学习算法库和挖掘模块、推荐规则模块、推荐效果分析模块等。

5.6 会员数据化运营分析的“大实话”

在会员数据化运营过程中，存在很多看似普遍的现象，但却蕴含了更加深刻的规律。

5.6.1 企业“不差钱”，还有必要做会员精准营销吗

“不差钱”是会员部门营销的普遍现状，这源于企业内部对于会员管理和营销的支持和信任，认为把钱花在会员运营上要比其他运营上更有价值。例如：每次做营销活动时选择给所有会员通发、做促销活动时提供优惠券给所有的会员。

当会员运营部门确实“不差钱”时，还有必要做精准营销吗？答案是需要。原因包括以下三个方面。

·更好的用户体验：会员精准营销首先带来的是个性化、精准化的信息，它能根据不同的会员提送不同的信息或者只针对有可能产生兴趣的会员推送信息，这首先是一种体验和关怀，而跟金额无关。

·更高的销售回报：会员精准营销在实施时，能够根据不同的会员提出不同的营销策略，对于价格敏感性会员推送折扣类活动、优惠券折扣或低价商品以促进其订单转化，对于忠诚类的非价格敏感性客户推送高价值商品、非折扣类活动甚至定制化服务来提高客单价和收入水平。通过对不同人群的区分对待处理能有效提升销售收入。

·更低的成本支出：会员精准营销可以有效节省运营成本，主要体现在营销成本和促销成本。通过精准选择最有可能转化的客户而非全部客户，能有效节省营销费用；在活动推送、优惠券发放时只针对价格敏感性、优惠券激励型客户发送，能有效降低优惠券成本。

5.6.2 用户满意度取决于期望和给予的匹配程度

用户满意度调查和分析是用户研究的重要内容，对用户来讲，其满意度到底取决于什么？根本上取决于用户期望与企业提供的价格、服务、商品、信息等的相对匹配程度，而不是企业提供的绝对程度。

以苹果iPhone 6手机为例，如果企业给出的销售价格是4056，假如用户只有两种满意度可供选择（满意或不满意），那么这两种可能行结论是：

·可能性1满意。由于用户通过其他大型网店、销售商城、官网得到的价格区间在4200～4400之间，此时用户的预期价格区间已经形成。企业给出的价格远低于用户的预期，因此比较满意。

·可能性2不满意。跟可能性1相反，假如用户的心理价格预期在4056以下，那么企业给出的价格将不符合用户预期，因此用户会不满意，此时绝对价值不是关键因素。

这反映了很多用户购买商品还是一种占便宜的心态，对实际运营有哪些意义？还是以价格为例，有以下应用场景：

·在商品描述中有原价、促销价、折扣价等多个价格，通过对比能够有效突出最终成交的折扣价。

·通过历史价格趋势对比，让用户感觉到现有价格已经比历史价格低，以此加强现有价格比较低的心理预期。

·在做促销活动时，通过与其他大型标杆销售企业（尤其是价格高的企业）的商品价格对比，来突出自身的价格优势。

·在原来商品销售价格的基础上，通过满减、满返等方式让用户在下单时看到更便宜的成交价格，促进商品的购物车转化。

·在特大型活动中，很多企业都会先把价格调高，然后再利用很大的折扣来降低价格，这样一升一降感觉价格会有巨大折扣，实际上可能仅仅比调价前便宜了一点而已，甚至很多商品都是借着活动的名义提价

销售。

通过以上方法，让用户感觉便宜（而不是因为商品的确实便宜）而促进商品销售转化。

5.6.3 用户不购买就是流失了吗

在CRM中，我们经常会定义用户订单间隔超过一定阈值（例如12个月）就认为用户已经流失。这种规则定义在用户信息匮乏的时代比较流行，但在大数据运营背景下，我们可以获取到用户的行为和信息，这些信息不仅局限于订单，更包括用户订单之前的多种线上和线下数据。包括：

·用户线下逛店数据：包括什么时间、哪个门店、看了哪些商品、停留多长时间等；

·用户线上行为数据：包括用户的访问频率、页面浏览量、访问深度、停留时间、搜索词、商品查看等。

因此，当用户没有发生购买行为时，并不意味着用户已经没有其他任何行为了，用户可能还在经常性登录账户、浏览网站商品、搜索相关信息等，这些信息意味着用户还有转化的可能性，只是由于某些原因没有发生购买（当然也可以基于这些数据做路径、漏斗等关键节点分析，找到用户流失的方向和原因）。

实现上述分析的基本前提是将用户的CRM数据跟线上用户行为数据、线下的监视数据打通，里面涉及的主要环节是非结构化数据的采集、处理、识别。线上的用户行为数据的采集、处理和识别相对容易实现，很多第三方开源、免费或付费工具都提供这类服务；但对于线下用户行为的识别中会涉及人脸信息、进店路径、商品识别、动作识别等关键机器学习方向的应用。目前仅仅对于标准状态下（例如在特定区域内使用脸部正面图像）的人脸识别支持性较好，而对于复杂门店、多姿势、多动作、多光线下的识别效果仍然有很多提升空间。

5.6.4 来自调研问卷的用户信息可信吗

对于很多无法直接采集到的用户信息，企业会选择使用调研问卷的方式收集。典型的调研问卷包括市场调研、满意度调研、基本信息调研等。通过调研问卷获得用户信息数据可靠吗？——未必，主要原因包括以下几个方面：

原因1：答案的顺序会影响用户习惯性的选择。

在调研问卷的答案设计中，答案的顺序是影响选择结果的重要因素。很多用户会习惯性的选择第一个或最后一个答案（尤其是第一个）。因此，很多在线的调研系统都会支持答案的随机顺序分布，以降低习惯性选择对结果的影响。

原因2：问卷问题的措辞会影响用户对问题的理解。

在问答式的题目中，很多题目可能会题意不清楚或表达方式过于烦琐，导致用户对问题的理解不清楚甚至误解，此时做出的答案将无法还原事实真相。

原因3：问卷题目的先后顺序会影响用户的选择判断。

大多数人都有自我防备的心理，这种心理在做调研问卷时仍然存在，因此如果一开始就问一些过于敏感或隐私的问题，很可能会招致用户的反感和抵触。如果将敏感问题放到最后，被调查者的戒备心理已大大减弱，更愿意提供真实信息。

原因4：问题题目过多会导致用户失去耐心。

用户填写调研问卷的耐心是有一定限度的，尤其当调研没有太多激励措施时尤为如此。一般情况下用户习惯于在一页内完成，并且时间尽量短、题目尽量少；如果时间过长或题目过多会导致用户失去耐心，中途可能放弃问答，即使最终都勉强答完，其结果可信度也大打折扣。

原因5：用户自我尊重的心理活动会促使用户选择积极的答案。

在调研问卷中问到有关用户的心理薄弱点时，用户往往会有一种自我尊重和自我保护的心理活动。例如，假如用户非常在意自己的学历太低，那么用户可能在实际作答时会倾向于选择更高的学历。同样的，在涉及金钱、收入、房产等与社会尊重密切相关的信息时都可能产生这类问题。

原因6：不同的调研问卷激励措施会导致不同的调研倾向。

很多企业在收集调研问卷时会提供一定的激励措施，而不同的激励措施下会产生不同的调研问卷结果。当激励越大时，会有越多的用户参与并且为了获取激励而仔细答题，这是一种收入和回报成正比的心态；当激励越小时，就会有越少的人原因参与，并且在答题时也可能不仔细，尤其面对较多的题目时该问题更加严重。

原因7：调研方式会直接影响调研的结论。

不同公司有不同样本选择和覆盖范围，例如互联网调研覆盖的是上网用户（上网用户也会区分在不同媒体上投放问卷而面对不同的媒体人员）、线下调研覆盖的是线下人群（线下区域的选择会成为覆盖人员的关键）。即使同一个企业选择不同的调研方式也会产生不同的结果。

当然，针对上述问题我们也不是束手无策，可以通过一定的方法对已经收集到的数据做验证。例如：

用户所选地域与IP的匹配关系。原始数据中用户一般都会选择所在地域，同时调研工具也会采集其IP地址，通过IP地址库可以匹配到用户的“网络”地域信息，将这两个信息进行对比即可了解用户所选地域是否正确。

通过IP确认用户地理位置的方法需要企业具备较高精度的IP地址库，否则匹配后的信息可能产生极大的误导性。

问题间的继承关系。调研问卷中可以通过一定的问题关系做数据排查，例如“您在以下哪些网上商城购买过产品？（多选）”“以下网上商城中您最常在哪家购买产品？）单选）”。假如在第二个问题选择了答案A，说明用户最经常在A网站购物；但如果在第一个问题中没有选择

A，即没有在A网站购过物，那么两道题的结论存在矛盾关系。这种情况下需要谨慎对待该用户的调研结果。

问题间的继承关系还表现在强制跳转，常见于当用户选择不同答案时跳转到不同的题目。典型应用场景是针对不同选项的用户做分群调研，例如购物会员与非购物会员的调研。假如题目之间出现明显的冲突，说明数据采集出现问题。

5.6.5 不要盲目相信二八法则

二八法则也称为二八定律、帕累托法则（定律）等，该法则用来形容20%的人占据了80%的事实。例如世界上20%的人口占据了80%的财富，企业中20%的老客户贡献了80%的企业利润等。该法则经常用于社会学、管理学等领域。

在会员管理中，如果80%的利润来源于20%的老客户，那么就应该把会员管理的焦点、资源和注意力放在20%的老客户身上，而不是平均分配到所有会员。这是一种抓主、次的正确思想。

在网络时代，这种规律正在被“长尾效应”改变。原因是随着信息越来越多、海量商品出现以及获取对称信息的成本越来越低，原来集中在少数对象或只能由少数对象产生的活动，已经可以被大多数对象产生。例如：

·80%的商品销售额、利润和现金流来源于80%的商品SKU。

·80%的利润和订单来自80%的客户。

·80%的页面浏览量分布在80%的页面。

·80%的搜索量分布在80%的搜索过程中。

上述的20%和80%都是模糊的量化词，20%代表的是少数，可以是10%、30%等；80%代表的是大多数，可以是60%、90%等。这些现象在企业经营模式越复杂、商品SDK越多、企业会员量越大的情况下尤为明显。

5.7 案例：基于RFM的用户价值度分析

5.7.1 案例背景

用户价值细分是了解用户价值度的重要途径，而销售型公司中对于订单交易尤为关注，因此基于订单交易的价值度模型将更适合运营需求。

对于用户价值度模型而言，由于用户的状态是动态变化的，因此一般需要定期更新，业务方的主要需求是至少每周更新一次。由于要兼顾历史状态变化，因此在每次更新时都需要保存历史数据，不同时间点下的数据将通过日期区分。

每次模型结果的数据，一部分是要给运营直接做分析，一部分是要“回吐”到数据库中，作为其他数据建模的基本数据维度，因此数据的输出需要有本地文件和写数据库两种方式。

本节案例的输入源数据sales.csv和源代码chapter5_code1.py位于“附件-chapter5”中，默认工作目录为“附件-chapter5”（如果不是，请cd切换到该目录下，否则会报“IO-Error:File sales.csv does not exist”）。程序输出RFM得分数据写入本地文件sales_rfm_score.csv和MySQL数据库的目标数据表（sales_rfm_score）。

5.7.2 案例主要应用技术

本案例没有使用现有成熟模型包，而是通过Python代码手动实现RFM模型，主要用到的库包括time、numpy、pandas和mysql.connector。有关RFM模型更多内容，请查看5.4.3节。

5.7.3 案例数据

案例数据是某企业2016年的部分抽样数据，数据来源于销售系统，主要是用户订单记录。以下是数据概况：

·特征变量数：4。

·数据记录数：86135。

·是否有NA值：有。

·是否有异常值：有。

以下是本数据集的4个特征变量，包括：

·USERID：用户ID，每个用户的ID唯一，由纯数字组成。

·ORDERDATE：订单日期，格式为YYYY-MM-DD，例如2016-01-01。

·ORDERID：订单ID，每个订单的ID唯一，由纯数字组成。

·AMOUNTINFO：订单金额，浮点型数据。

5.7.4 案例过程

步骤1 导入用到的库。

```
import time  # 导入时间库
import numpy as np  # 导入numpy库
import pandas as pd  # 导入pandas库
import mysql.connector  # 导入mysql连接库
```

该案例用到了四个库：time、numpy、pandas、mysql.connector。

·time：用来记录插入数据库时的当前时间。

·numpy：用来做基本数据处理等。

·pandas：有关日期转换、数据格式化处理、主要RFM计算过程等。

·mysql.connector：数据库连接工具，读写MySQL数据库。

步骤2 读取原始数据。

```
dtypes = {'ORDERDATE': object, 'ORDERID': object, 'AMOUNTINFO': np.float32}
置每列数据类型
raw_data = pd.read_csv('sales.csv', dtype=dtypes, index_col='USERID')  # 读
取数据文件
```

dtypes定义的字典用于使用pd.read_csv读取数据时对数据框数据类型的自定义，而非系统默认类型。本案例中将ORDERDATE（订单时间）和ORDERID（订单ID）都设置为object（字符串对象），AMOUNTINFO（订单金额）设置为浮点型。

使用pd.read_csv读取数据文件过程中，默认的csv以逗号作为分隔符，因此无需指定分隔符；设置参数dtype为上述自定义的类型字典，同时指定列USERID（用户ID）为索引列，代替默认的数值索引。

步骤3 数据审查和校验，主要包括数据概览、缺失值审查等。

```
# 数据概览
print ('Data Overview:')
print (raw_data.head(4))  # 打印原始数据前4条
print ('-' * 30)
print ('Data DESC:')
print (raw_data.describe())  # 打印原始数据基本描述性信息
print ('-' * 60)
```

在数据概览部分，通过数据框的head方法输出前4条数据，通过数据框的describe方法输出其基本描述性统计信息。返回结果如下：

```
Data Overview:
         ORDERDATE     ORDERID  AMOUNTINFO
USERID
142074  2016-01-01  4196439032      9399.0
56927   2016-01-01  4198324983      8799.0
87058   2016-01-01  4191287379      6899.0
136104  2016-01-01  4198508313      5999.0
------------------------------
Data DESC:
         AMOUNTINFO
count  86127.000000
mean     744.705261
std     1425.211182
min        0.500000
25%       13.000000
50%       59.000000
75%      629.000000
max    30999.000000
------------------------------------------------------------
```

如何通过上述结果得到有效信息？

在Data Overview部分，主要查看不同数据列的数据格式，尤其是有特定转换操作之后是否符合源数据文件的格式或得到目标转换要求；另外，数据的长度、组成规律、类型等是否与真实数据一致。

在Data DESC部分主要分析数值型字段（本案例中是AMOUNTINFO）的数值分布规律，包括记录数、极值、标准差、分位数结果等，可用于数据集的适用模型、极值的处理等后续计算的辅助判断依据。

在本案例中，发现数据的极值相差非常大，并且标准差也很大，说明数据波动非常明显。另外，最大值和最小值似乎有些奇怪：最大值竟然有超过30000元、最小值却只有0.5元，这两种状态都非常异常。经过与业务方沟通后确认，最大值的订单金额有效，为某客户一次性购买多

个大家电商品；而订单金额为0.5元的订单都属于促销优惠券生成的订单，这些订单用来为用户消费时提供优惠券，没有实际意义，因此这些数据需要去掉。除了这些0.5元的订单，所有低于1元的订单均有这个问题。

```
# 缺失值审查
na_cols = raw_data.isnull().any(axis=0)  # 查看每一列是否具有缺失值
print ('NA Cols:')
print (na_cols)  # 查看具有缺失值的列
print ('-' * 30)
na_lines = raw_data.isnull().any(axis=1)  # 查看每一行是否具有缺失值
print ('NA Recors:')
print ('Total number of NA lines is: {0}'.format(na_lines.sum()))  # 查看具有缺失值的行总记录数
print (raw_data[na_lines])  # 只查看具有缺失值的行信息
print ('-' * 60)
```

缺失值对于后续的计算会产生重大影响，因此这里需要确认数据中是否含有缺失数据。先通过raw_data.isnull（）.any（axis=0）来判断所有列是否含有缺失信息，其中的isnull用来查看是否有缺失值，any用来判断数据记录中的任何一个位置出现缺失值都会计算在内，参数axis=0以列为基础做查看。打印结果如下：

```
NA Cols:
ORDERDATE      True
ORDERID       False
AMOUNTINFO     True
```

返回结果显示了ODERDATE和AMOUNTINFO都有缺失。下一步结合每一行看下到底有多少条数据出现缺失。这次使用的方法跟判断列缺失一致，仅仅是axis设为1用来表示按行查看。在na_lines中，结果由True和False组成，因此可以直接做数值计算，这里使用sum函数统计一共有多少行为含有NA，含有NA的总记录数和详细行记录打印结果如下：

```
NA Recors:
Total number of NA lines is: 10
         ORDERDATE     ORDERID  AMOUNTINFO
USERID
75849   2016-01-01  4197103430         NaN
103714         NaN  4136159682       189.0
155209  2016-01-01  4177940815         NaN
139877         NaN  4111956196         6.3
```

```
54599   2016-01-01  4119525205         NaN
65456   2016-01-02  4195643356         NaN
122134  2016-09-21  3826649773         NaN
116995  2016-10-24  3981569421         NaN
98888   2016-12-06  3814398698         NaN
145951  2016-12-29  4139830098         NaN
```

由返回结果可知，一共有10条数据含有NA值，这些数据在整体样本集中占比非常小，因此这里可以直接删除。

步骤4　数据预处理准备工作，包括数据异常、格式转换和处理。

```
# 异常值处理
sales_data = raw_data.dropna()  # 丢弃带有缺失值的行记录
sales_data = sales_data[sales_data['AMOUNTINFO'] > 1]  # 丢弃订单金额<=1的记录
```

对于异常值的处理，直接使用数据框的dropna方法删除；对于订单金额<=1的数据，我们也直接丢弃，只选择订单金额>1的数据记录。

```
# 日期格式转换
sales_data['ORDERDATE'] = pd.to_datetime(sales_data['ORDERDATE'], format='%Y
%m-%d')  # 将字符串转换为日期格式
print ('Raw Dtypes:')
print (sales_data.dtypes)  # 打印输出数据框所有列的数据类型
print ('-' * 60)
```

日期转换的目的是实现基于时间间隔的计算，这样才能算出R距离指定日期的天数。这一操作没有在读取数据转换（在读取时的具体转换方法请参考4.6.4节），这里使用pd.to_datetime方法将ORDERDATE列转换为Pandas的日期类型（pd.datetime类型），format参数以原始数据字符串的格式来写，只有格式对应上才能实现解析。解析完成后，使用数据框的dtypes打印输出Dtypes类型如下：

```
Raw Dtypes:
ORDERDATE     datetime64[ns]
ORDERID               object
AMOUNTINFO           float32
dtype: object
```

接下来需要分别计算R、F、M三个原始变量的数值，主要使用的方式是数据框的groupby方法。

```
# 数据转换
recency_value = sales_data['ORDERDATE'].groupby(sales_data.index).max()  # 计
算原始最近一次订单时间
frequency_value = sales_data['ORDERDATE'].groupby(sales_data.index).count()
算原始订单频率
monetary_value = sales_data['AMOUNTINFO'].groupby(sales_data.index).sum()  #
算原始订单总金额
```

这三行代码都是以原始数据框的索引为主键（以用户ID作为汇总维度）分别对ORDERDATE求最大值、对ORDERDATE做计数统计、对AMOUNTINFO求和，得分R、F、M三个指标的原始值。

步骤5　计算RFM得分。

```
# 分别计算R、F、M得分
deadline_date = pd.datetime(2017, 01, 01)  # 指定一个时间节点，用于计算其他时间与
该时间的距离
r_interval = (deadline_date - recency_value).dt.days  # 计算R间隔
r_score = pd.cut(r_interval, 5, labels=[5, 4, 3, 2, 1])  # 计算R得分
f_score = pd.cut(frequency_value, 5, labels=[1, 2, 3, 4, 5])  # 计算F得分
m_score = pd.cut(monetary_value, 5, labels=[1, 2, 3, 4, 5])  # 计算M得分
```

首先指定一个时间节点，用于计算其他时间与该时间的距离，这是计算R的基础。这里定义了2017-01-01，通过数据框相减得到时间间隔天数对象，并对该对象使用dt.days方法获得天的数值。

下面对得到的R、F、M三个变量值使用分位数法做区间划分，这里使用了pd.cut方法，默认设置为5份，同时通过labels标签指定区间标志。主要注意的是对R（最近购买时间）而言，数值越大意味着离指定日期越远，因此其区间划分后的值应该越小，所以该标签列表顺序与其他两个相反。

dt是pandas中Series时间序列datetime类属性的访问对象，除了代码中用到的days（天）以外，还包括：date、dayofweek、dayofyear、days_in_month、freq、hour、micro-second、minute、month、quarter、second、time、tz、week、weekday、weekday_name、weekofyear、year等。这些是提取Series时间数据的常用方法。

在得到RFM各自得分后，将三维维度数据合并为一个数据框，便于

在一起做展示以及后续的RFM总得分计算。

```
# R、F、M数据合并
rfm_list = [r_score, f_score, m_score]  # 将r、f、m三个维度组成列表
rfm_cols = ['r_score', 'f_score', 'm_score']  # 设置r、f、m三个维度列名
rfm_pd = pd.DataFrame(np.array(rfm_list).transpose(), dtype=np.int32, columr
立r、f、m数据框
print ('RFM Score Overview:')
print (rfm_pd.head(4))
print ('-' * 60)
```

先建立R、F、M三个维度的值列表和名称列表，用于生成数据框时指定数据和标签。然后使用pd.DataFrame建立数据框。使用np.array将R、F、M生成的值列表转换为矩阵，此时的矩阵形状是（3，59676），不符合我们需要的三列的需求，因此使用transpose方法对矩阵做转置处理，该方法也可以简写为T（注意大小写），即np.array（rfm_list）.transpose（）等价于np.array（rfm_list）.T；由于R、F、M的值域是[1，5]的整数，因此创建数据框时指定数据类型为整数型，通过dtype=np.int32实现，该方法也可以写成dtype='int32'，二者是等价的；然后设置列名并指定索引列为用户ID，由于R、F、M三个Series的索引都相同，因此这里随便指定为frequency_value.index。最后打印前4条结果如下：

```
RFM Score Overview:
        r_score  f_score  m_score
USERID
51220         4        1        1
51221         2        1        1
51224         3        1        1
51225         4        1        1
------------------------------------------------------------
```

完成R、F、M数据框创建后，可以基于该数据框计算RFM总得分，这里使用两种方法：基于三个维度加权的RFM总得分以及基于组成的总得分。

```
# 计算RFM总得分
# 方法一：加权得分
rfm_pd['rfm_wscore'] = rfm_pd['r_score'] * 0.6 + rfm_pd['f_score'] * 0.3 + r
# 方法二：RFM组合
rfm_pd_tmp = rfm_pd.copy()
rfm_pd_tmp['r_score'] = rfm_pd_tmp['r_score'].astype('string')
rfm_pd_tmp['f_score'] = rfm_pd_tmp['f_score'].astype('string')
```

```
rfm_pd_tmp['m_score'] = rfm_pd_tmp['m_score'].astype('string')
rfm_pd['rfm_comb'] = rfm_pd_tmp['r_score'].str.cat(rfm_pd_tmp['f_score']).st
```

在方法一中，直接取出数据框的三列（R、F、M）通过乘以特定权重值得到总得分。由于业务方更关注活跃度，认为访问的邻近度最重要，因此R的权重比较高，设置为0.6；其次是访问频率F设置为0.3；订单金额M则设置为0.1。然后基于不同的列直接做加权相加，而无需通过循环读出各个元素再做计算。

在设定权重值时需要与业务部门沟通，主要看业务关注哪个方面，再设置对应维度的值。而对于权重值的分配一般情况下总权重的和为1，这样划分更清晰并便于解释和管理。

在方法二中，我们要将R、F、M三个值组合，需要用到字符串的组合。由于原始数据框中为了做加权计算而设置为数值型，因此这里需要转换为字符串型。为了不影响原始数据，通过copy方法得到一份副本，然后将副本的三列使用astype方法将数值型转换为字符串型。

在设置数据类型时一般有两种思路：一种是在创建数据框时通过dtype指定（原始数据框就是这种方法），第二种是在创建好的数据框中使用astype方法将特定栏位转换为目标类型。

在最后的合并过程中，使用了Pandas的字符串处理库str中的cat方法做字符串合并，该方法可以将右侧的数据合并到左侧，在连续使用两个str.cat方法后得到总的R、F、M字符串组合。

相关知识点：使用str.cat方法将字符串合并

关于字符串的合并，之前在不同章节中主要用到了四种方法：

·使用+（加号）组合字符串：例如输入X='a'+'b'能得到X的值是'ab'。

·使用%占位符做字符串组合，主要用在变量输出上，例如输入X='I am%s'%'Tony'，能得到X的值是'I am Tony'。

·使用.join方法将多个可迭代的对象合并，例如输入X=''.join（['I'，'am'，'tony']），得到X的值是'I am tony'。

·使用.format做占位符将多个字符串合并，跟%用法类似但更强大，主要用在变量输出上，例如输入：X='I am{1}and{0}years old'.format（30，'Tony'），得到X的值是'I am Tony and 30 years old'。

本节使用了pandas自带的字符串组合方法，该放在在str库中，它用于字符串对象合并。该方法语法如下：cat（self，others=None，sep=None，na_rep=None），参数：

·others：要合并的另外一个对象（右侧对象），如果为空则将左侧对象组合。

·sep：合并的分隔符，默认为空，可自定义，例如'，'、'；'等。

·na_rep：如果遇到NA（缺失值）时如果处理，默认为忽略。

需要注意的是：该方法用于对Series做组合，而不能是数据框，适用于一维数据或字符串。

示例：

将左侧对象组合

输入：pd.Series（['a'，'b'，'c']）.str.cat（sep='；'）

输出：'a；b；c'

将左侧对象和右侧对象组合

输入：pd.Series（['a'，'b'，'c']）.str.cat（['A'，'B'，'C']，sep='；'）

输出：

```
0    a;A
1    b;B
2    c;C
```

步骤6　打印输出和保存结果，该步骤中我们将数据打印出来，并分别保存为本地csv数据文件以及将数据存入MySQL数据库。

经过上述计算过程，已经得到R、F、M结果，这里我们将其打印出来。

```
# 打印结果
print ('Final RFM Scores Overview:')
print (rfm_pd.head(4))  # 打印数据前4项结果
print ('-' * 30)
print ('Final RFM Scores DESC:')
print (rfm_pd.describe())
```

使用head方法指定打印前4条结果以及RFM得分描述性统计结果如下：

```
Final RFM Scores Overview:
        r_score  f_score  m_score  rfm_wscore rfm_comb
USERID
51220         4        1        1         2.8      411
51221         2        1        1         1.6      211
51224         3        1        1         2.2      311
51225         4        1        1         2.8      411
------------------------------
Final RFM Scores DESC:
            r_score       f_score       m_score    rfm_wscore
count  59676.000000  59676.000000  59676.000000  59676.000000
mean       3.299970      1.013439      1.000134      2.384027
std        1.402166      0.116017      0.018307      0.845380
min        1.000000      1.000000      1.000000      1.000000
25%        2.000000      1.000000      1.000000      1.600000
50%        3.000000      1.000000      1.000000      2.200000
75%        5.000000      1.000000      1.000000      3.400000
max        5.000000      5.000000      5.000000      5.000000
```

该过程主要验证数据输出是否符合预期，主要是R、F、M的组合和加权计算；使用描述性统计简单看下数值型得分的区间分布（重点是极值）是否在预期的[1，5]之间。分析发现上述数据没问题。

如果原始数据集中有NA值而没有经过处理，那么在使用分位数方法时会默认忽视NA的分位，即值为NA的数据点仍然是NA。这会导致在做后期加权得分和组合时都会遇到问题。通过描述性统计也能发现该问题是否存在。

接着将RFM得分文件保存到本地，便于业务做进一步分析应用。

```
# 保存RFM得分到本地文件
rfm_pd.to_csv('sales_rfm_score.csv')  # 保存数据为csv
```

直接使用数据框对象的to_csv方法将数据保存为csv格式的数据文件。除了保存为csv格式的文件，数据框对象还可以支持保存为excel等文件。另外也支持sql、json、stata、pickle、sparse、hdf、html等对象的输出。保存到本地csv文件的部分数据截图如图5-4所示。

由于RFM结果要作为其他模型结果的输入维度，因此需要将RFM得分写入数据框。这里使用的是MySQL官方连接程序实现。

```
# 设置要写库的数据库连接信息
table_name = 'sales_rfm_score'  # 要写库的表名
# 数据库基本信息
config = {'host': '127.0.0.1',  # 默认127.0.0.1
          'user': 'root',  # 用户名
          'password': '123456',  # 密码
          'port': 3306,  # 端口，默认为3306
          'database': 'python_data',  # 数据库名称
          'charset': 'gb2312'  # 字符编码
          }
```

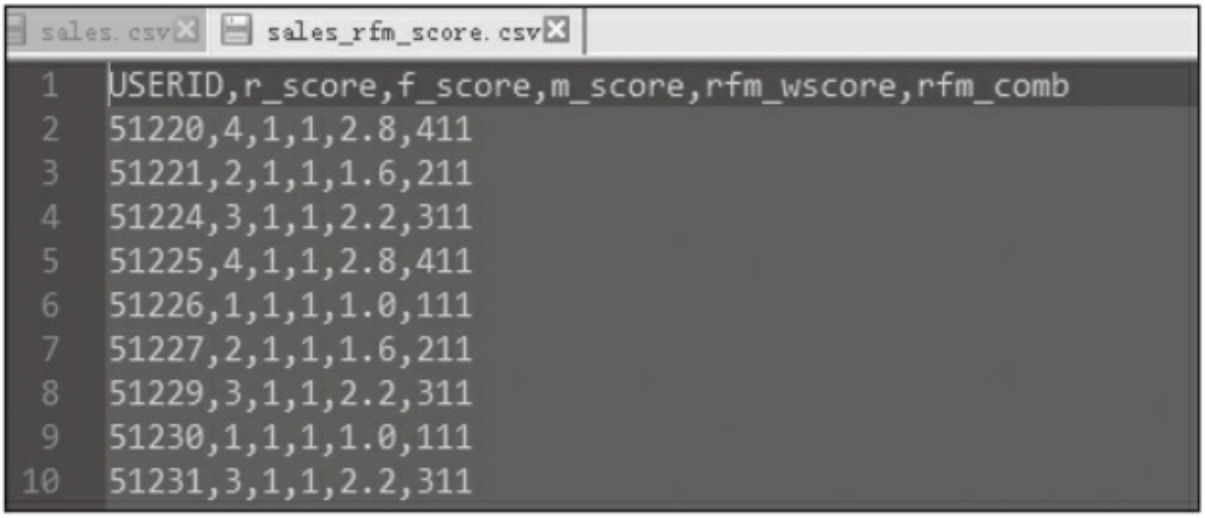


图5-4　本地csv文件结果

首先建立要写入数据库的基本信息，通过table_name='sales_rfm_score'定义要写入的表的名称，该变量会在后续表

识别和写入时用到。config定义了一个用于写入数据库的详细配置信息定义，包括主机host、用户user、密码password、端口port、要写入的数据表所处的数据库名database、字符集charset。这些信息在2.2.3节中已经详细介绍过，在此不再赘述。

配置信息定义完成后，下面建立数据库连接。

```
con = mysql.connector.connect(**config)  # 建立mysql连接
cursor = con.cursor()  # 获得游标
```

使用mysql.connector.connect方法连接数据库，后续的获得游标和写库都需要该有效连接；使用连接的con.cursor方法获得游标，后续的查询等操作都基于该对象实现。

在将数据写入数据库之前，需要先判断数据库中是否存在要写入的表。如果不存在需要新建数据表。

```
# 查找数据库是否存在目标表，如果没有则新建
cursor.execute("show tables")  #
table_object = cursor.fetchall()  # 通过fetchall方法获得所有数据
table_list = []  # 创建库列表
for t in table_object:  # 循环读出所有库
    table_list.append(t[0])  # 每个每个库追加到列表
if not table_name in table_list:  # 如果目标表没有创建
    cursor.execute('''
    CREATE TABLE %s (
    userid                VARCHAR(20),
    r_score                int(2),
    f_score               int(2),
    m_score               int(2),
    rfm_wscore               DECIMAL(10,2),
    rfm_comb               VARCHAR(10),
    insert_date                VARCHAR(20)
    )ENGINE=InnoDB DEFAULT CHARSET=gb2312
    ''' % table_name)  # 创建新表
```

使用游标的execute方法执行一个sql语句，由于在config中我们定义了数据库名，因此这里直接使用"show　tables"来显示当前数据库下所有的表列表，而不是先使用use[数据库]，然后再"show tables"。

游标执行后获得的对象是没有数据的，使用fetchall方法抓取全部数据并返回，但返回的是一个数据列表对象[（u'order'，），（u'test'，）]，列表中的每个元素都是一个元组，而数据表名处于元组

的第一个值。因此接着通过一个循环将对象读取出来，得到数据表列表table_list。

使用if条件语句判断要写入的数据表table_name是否在数据表列表中，如果该表不存在则新建。这里仍然使用游标的execute方法执行一段SQL语句，只是由于创建表的SQL语句较长，使用'''来表示多个段落的字符串。

相关知识点：Python的字符串标识符

Python的字符串标识符有三种，分别是'（单引号）、"（双引号）、'''（三个连续单引号）、"""（三个连续双引号）。这四种表示符号（注意都是英文状态下）在大多数场景下的作用是相同的，但是都要求成对出现。例如：'ABC'等价于"ABC"，也等价于'''ABC'''。

但在某些场景下他们的用途是有区分的，主要体现在两个方面：

·当字符串中出现子集字符串时，需要使用不同类型的字符串做区分，例如'I said:"You can go！"'，这时由于在外层字符串使用了'（单引号），在内层就不能再使用相同的表示方法了，所以使用双引号。

·当字符串非常长（通常超过1行）、或者字符串步长但是需要分段显示时，就需要用到'''（连续三个单引号）或"""（三个连续双引号）来表示，上述代码就是这种应用。除此以外，这种长段落还经常用到字典定义、格式化段落文本输出、复杂功能的SQL语句（例如SQL嵌套、关联查询）等。

经过上述步骤，能保证要写入的数据表已经存在，并且跟要写入的数据类型和格式相匹配。接下来开始写库操作。

```
# 将数据写入数据库
user_id = rfm_pd.index  # 索引列
rfm_wscore = rfm_pd['rfm_wscore']  # RFM加权得分列
rfm_comb = rfm_pd['rfm_comb']  # RFM组合得分列
timestamp = time.strftime('%Y-%m-%d', time.localtime(time.time()))  # 写库日期
print ('Begin to insert data into table {0}...'.format(table_name))  # 输出开始写库的提示信息
for i in range(rfm_pd.shape[0]):  # 设置循环次数并依次循环
    insert_sql = "INSERT INTO `%s` VALUES ('%s',%s,%s,%s,%s,'%s','%s')" % \
                 (table_name, user_id[i], r_score.iloc[i], f_score.iloc[i],
```

```
库SQL依据
    cursor.execute(insert_sql)  # 执行SQL语句，execute函数里面要用双引号
    con.commit()  # 提交命令
```

写库时用到了7列数据，分别是userid、r_score、f_score、m_score、rfm_wscore、rfm_comb、insert_date。其中：

·userid可以从数据框的索引中取出，用于标识不同的用户，使用数据框的index获得索引列用户ID数据框；

·r_score、f_score、m_score三列值在“分别计算R、F、M得分”时已经产生；

·rfm_wscore、rfm_comb可以分别使用数据框读取列名以获得RFM加权得分和组合得分数据；

·insert_date用来记录每次写库时的日期，该日期可以作为变化状态的判断依据，可应用于分析不同时间和周期下用户的活跃度变化，例如5.5.3节中的那种应用。而在跟其他模型做数据集成应用时，可根据实际情况取需要的时间对应的状态，例如取最新状态可取insert_date中的最大值；

在获取insert_date写库日期时使用了3个时间函数：

·time.time（）：获取当前系统时间下相对于Epoch时间（1970-01-0100:00:00 UTC），以浮点型数据的形式表示，例如1495770037.572。

·time.localtime（）：将时间转换为本地时间，是一个包含了年、月、日、小时、分钟、秒、一周中第几天、一年中第几天等数据的元组。例如将上述时间转换结果为time.struct_time（tm_year=2017，tm_mon=5，tm_mday=26，tm_hour=11，tm_min=40，tm_sec=37，tm_wday=4，tm_yday=146，tm_isdst=0）。

·time.strftime（）：将不同形式的时间转换为特定格式和时间表示方法，该方法在本书中多次用到，例如将上述结果转换为'2017-05-26'。

经过上述功能转换后的日期是YYYY-MM-DD的格式，例如2017-05-26。接下来的写库操作利用for循环，根据数据框的记录做读出索引

值，然后使用游标的执行方法运行SQL；由于我们要做批量写入操作，因此需要使用on.commit方法提交命令。

写入操作完成后，关闭游标和数据库连接。

```
        cursor.close()  # 关闭游标
        con.close()  # 关闭数据库连接
        print ('Finish inserting, total records is: %d' % (i + 1))  # 打印写库结果
```

得到总的写入数据库记录数如下：

```
        Finish inserting, total records is: 59676
```

可使用Navicat打开对应的数据库表，查看结果是否符合预期，以及总记录数是否一致，如图5-5所示。

5.7.5　案例数据结论

由于在RFM划分时，将区间划分为5份，因此可以将这5份区间分别定义了：高、中、一般、差和非常差5个级别，分别对应到R、F、M中的5/4/3/2/1。

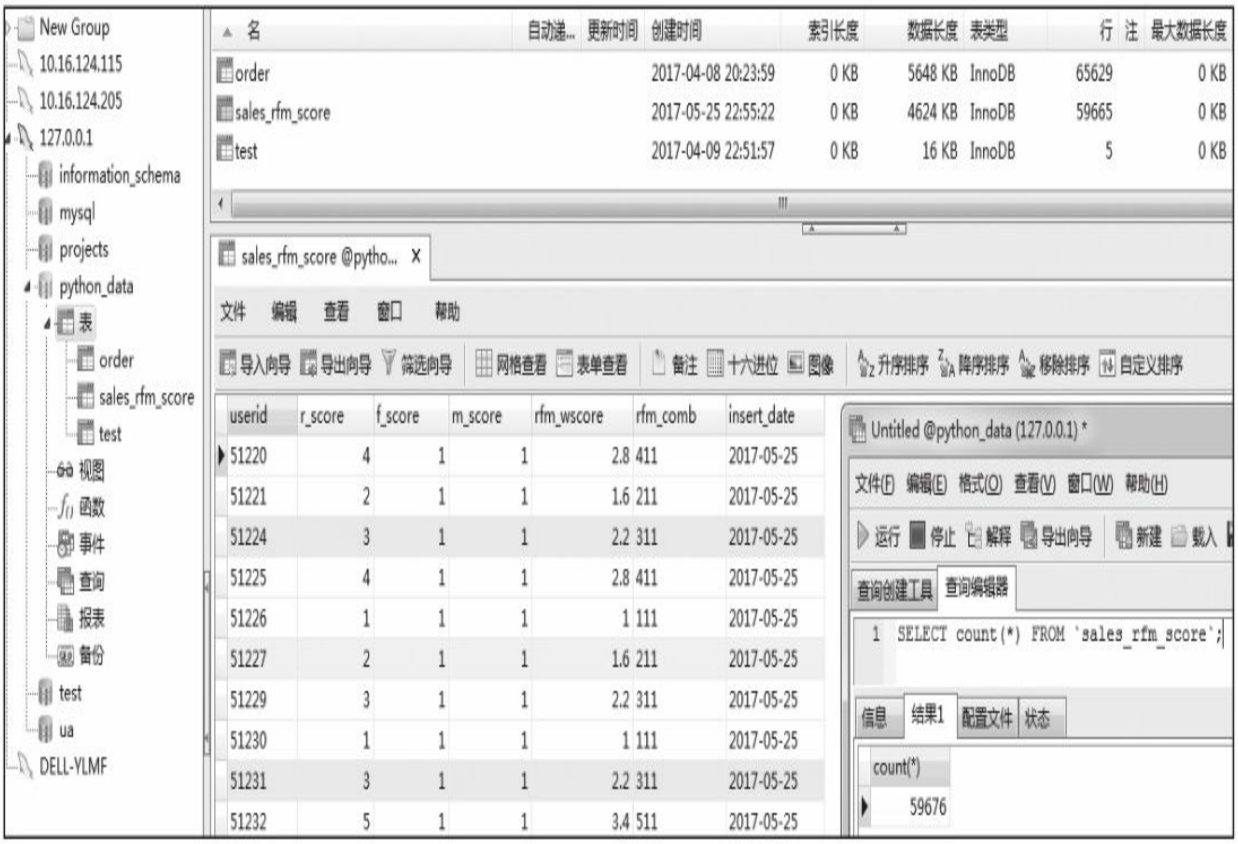


图5-5　使用Navicat查看结果

基于RFM得分业务方得到这样的结论：

·公司的会员中99%以上的客户消费状态都不容乐观，主要体现在消费频率低R、消费总金额低M。——经过分析，这里主要由于其中有一个用户（ID为74270）消费金额非常高，导致做5分位时收到最大值的影响，区间向大值域区偏移。

·公司中有一些典型客户的整个贡献特征明显，重点是RFM得分为555的用户（ID为74270），该用户不仅影响了订单金额高，而且其频率

和购买新鲜度和消费频率都非常高，应该引起会员管理部门的重点关注。

·本周表现处于一般水平以上的用户的比例（R、F、M三个维度得分均在3以上的用户数）相对上周环比增长了1.3%。这种良好趋势体现了活跃度的提升。

·本周低价值（R、F、M得分为111以上）用户名单中，新增了1221个新用户，这些新用户的列表已经被取出。

5.7.6 案例应用和部署

针对上述得到的分析结论，会员部门采取了以下措施：

·将ID为74270的用户加入大客户名单，并实现重点关怀和管理。

·上周的会员工作中有关低价值群体的用户并未减少，反而新增了一些，需要会员部门重点关注和处理。目前最主要的还是通过会员渠道拉动会员再次访问网站并订单，防止客户的流失和沉默，而订单的金额是次要因素。

录入数据库的RFM得分数据已经应用到其他数据模型中，成为建模输入的关键维度特征之一。同时，该模型已经作为定时任何每周一早晨上班前运行一次。

5.7.7 案例注意点

本案例中有以下几个点需要读者重点关注：

对于R、F、M区间的划分是一个离散化的过程，具体需要划分为几个区间需要跟业务方确认。本案例是划分为5个区间，实际上一般不会比这个区间更多，3～5是比较好的选择。

R、F、M的组合加权中关于权重的确认，不同时期的业务关注点可能不同，因此权重值可能是需要调整的；另外，权重值的打分一般使用专家打分法，即让业务来做权重打分，如果没有业务打分，那么大多数情况下三者的权重从大到小依次是：R＞F＞M。

本案例中有一个ID为74270的用户，由于其各方面特征非常显著，因此整个RFM得分的划分区间都受其影响。这种极值的影响（包括案例开始时极小值的处理）需要跟业务部门沟通确认才能进行处理，此次的极大值的出现是业务认可的有效数据，因此没有去除；如果有其他极值可能需要特别处理，否则会影响区间划分。本例中虽然影响划分，但恰好发现了该用户的价值。

虽然订单数据库中的数据质量相对较高，但可能由于采集问题、数据库同步、ETL、查询、误操作等问题还是会导致NA值的出现，NA值的处理非常重要。

R、F、M三个维度的处理包括计算、离散化、组合、转换之前都需要注意其数据类型和格式，尤其是有关时间项的转换操作提前完成。

5.7.8 案例引申思考

利用RFM模型划分用户群体并作价值度分析是统计分析非常基础且有效的方法。该模型几乎用不到任何专业的统计分析和挖掘知识，只需要具有基本的数据清洗、处理和转换技能即可完成，因此几乎是各个企业都会用到的模型。更重要的是，该模型原理简单，业务方在理解和应用起来非常容易入手，大大提高了部署落地的可能性。

对于本案例的实施，读者还可以从以下几方面做引申思考：

·问题一　按照R、F、M每个区间划分为5份的原则，实际上出来的总得分可能性是53=125种，为什么结果中只有14种组合得分？

·问题二　如果不同周期下R、F、M的权重会改变，其运营结果是否具有可比性和连续性？

·问题三　RFM模型可以作为模型分析方法，也可以作为数据预处理方法，基于不同的维度通过计算组合得分或加权得分的方式获得新的数据，这是一种数据降维的有效方法。使用组合得分方法和加权得分方法得到的两种降维方法在后续应用中有哪些不同？

笔者关于引申思考的想法：

·问题一　在案例注意点中也提到了，由于受到ID为74270的用户行为影响，其极大值导致整个区间划分非常大，很多区间内没有数据。以F值为例，该用户的F原始值为14（一年中有14个订单），其他用户最高的只有6个订单，因此在做区间划分时，在7～13内没有一个用户符合条件，也就不会产生该得分区间的用户统计量。

·问题二　权重的变化影响的仅仅是加权得分项，而不影响组合得分项，因此其他建模应用中没有任何影响。在分析过程中，这种改变不可避免地会使加权得分产生规则发生变化，需要在业务做分析时加以提醒。

·问题三　首先需要明确的是RFM的两种总得分应该只选择其中一种加以应用，因为二者具有高度相关性。由于加权RFM得分是一个连续

型变量，更适合直接参与到后续回归建模中；组合RFM得分是一个分类变量，更适合应用到分类建模中，同时需要注意的是该分类需要做标志转换，具体过程参见“3.2将分类数据和顺序数据转换为标志变量”。

5.8 案例：基于AdaBoost的营销响应预测

5.8.1 案例背景

该案例介绍了有关会员营销预测的实际应用。会员部门在做会员营销时，希望能通过数据预测在下一次营销活动时，响应活动会员的具体名单和响应概率，以此来制定针对性的营销策略。

本案例是一个常规性应用模型，业务部门希望通过建立好的模型周期性自动执行，也满足营销需要。同时，业务部门还希望基于现有的辅助决策平台将会员数据筛选和查看功能跟该模型结合起来应用。

本节案例的输入源数据order.xlsx和源代码chapter5_code2.py位于“附件-chapter5”中，默认工作目录为“附件-chapter5”（如果不是，请cd切换到该目录下，否则会报“IOError:File order.xlsx does not exist”）。程序的输出预测数据写入本地文件order_predict_result.xlsx。

5.8.2 案例主要应用技术

本案例用到的主要技术包括：

·数据预处理：二值化标志转换OneHotEncoder、基于方差分析的特征选择的数据降维SelectPercentile结合f_classif。

·数据建模：管道方法Pipe、交叉检验cross_val_score配合StratifiedKFold和自定义得分计算方法、集成分类算法AdaBoostClassifier。

主要用到的库包括：time、Numpy、Pandas和Sklearn，其中Sklearn是数据建模的核心库。

本案例的技术应用重点是通过管道方法将多个数据处理环节结合起来，形成处理管道对象，然后针对该对象做交叉检验并得到不同参数下的检验结果，辅助于参数值设置。

5.8.3 案例数据

案例数据位于order.xlsx中，包括2个sheet：sheet1为本次案例的训练集，sheet2为本次案例的预测集。以下是数据概况：

·特征变量数：13。

·数据记录数：sheet1中的训练集数据记录数为39999，sheet2中的预测集数据记录数为8843。

·是否有NA值：有。

·是否有异常值：无。

以下是本数据集的13个特征变量，包括：

·age：年龄，整数型变量。

·total_pageviews：总页面浏览量，整数型变量。

·edu：教育程度，分类型变量，值域[1，10]。

·edu_ages：受教育年限，整数型变量。

·user_level：用户等级，分类型变量，值域[1，7]。

·industry：用户行业划分，分类型变量，值域[1，15]。

·value_level：用户价值度分类，分类型变量，值域[1，6]。

·act_level：用户活跃度分类，分类型变量，值域[1，5]。

·sex：性别，值域为1或0。

·blue_money：历史订单的蓝券用券订单金额（优惠券的一种），整数型变量。

·red_money：历史订单的红券用券订单金额（优惠券的一种），整数型变量。

·work_hours：工作时间长度，整数型变量。

·region：地区，分类型变量，值域[1，41]。

目标变量response，1代表用户有响应，0代表用户未响应。

5.8.4 案例过程

步骤1 导入库。

```
import time  # 导入自带时间库
import numpy as np  # numpy库
import pandas as pd  # pandas库
from sklearn.preprocessing import OneHotEncoder  # 导入OneHotEncoder库
from sklearn.model_selection import StratifiedKFold, cross_val_score  # 导入交叉检验算法
from sklearn.feature_selection import SelectPercentile, f_classif  # 导入特征选择方法库
from sklearn.ensemble import AdaBoostClassifier  # 导入集成算法
from sklearn.pipeline import Pipeline  # 导入Pipeline库
from sklearn.metrics import accuracy_score  # 准确率指标
```

本案例中用到的库相对较多，主要包括time时间库用来记录不同算法参数下模型的运行时间；numpy和pandas是常用的数据读取、展示、处理和数据保存库；在sklearn中用到以下库：

·OneHotEncoder：将分类变量和顺序变量转换为二值化标志变量。

·StratifiedKFold，cross_val_score：用来做交叉检验，前者用来将数据分为训练集和测试集；后者用来交叉检验。这里选择的StratifiedKFold能够有效结合分类样本标签做数据集分割，而不是完全的随机选择和分割。这种方式在应对分类样本不均衡时尤为有效。

·SelectPercentile，f_classif：前者用来做特征选择的数量控制，后者用来确定特征选择的得分计算标准。

·AdaBoostClassifier是集成算法，用来做分类模型训练。

·Pipeline：是一个“管道”，目的是将不同的环节结合起来应用，这常用于多个流程和环节的反复性操作。例如本案例中，将特征选择和集成算法结合起来形成一个“管道”对象，然后针对该对象训练不同参数下对应交叉检验的结果。

·accuracy_score：准确率评估指标，用于分类算法。

除了上述导入库或包以外，还有关于使用pandas写Excel需要的附属

库。Pandas支持直接将数据框写入Excel，支持xls（97～2003）和xlsx（2007以上）两种格式，前者需要xlwt库支持，后者需要openpyxl支持。如果读者之前没有安装过这两个库，需要在系统终端命令行窗口使用pip install xlwt和pip install openpyxl完成安装。这两个库无需导入，在Pandas写入Excel时会自动调用。

步骤2　基本状态审查，包括基本状态查看、缺失值审查、类样本均衡审查等。

基本状态查看，用于查看数据集的记录数、维度数、前2条数据、描述性统计和数据类型等。

```
def set_summary(df):
    '''
    查看数据集的记录数、维度数、前2条数据、描述性统计和数据类型
    :param df: 数据框
    :return: 无
    '''
    print ('Data Overview')
    print ('Records: {0}\tDimension{1}'.format(df.shape[0], (df.shape[1] - 1
印数据集X形状
    print ('-' * 30)
    print (df.head(2))  # 打印前2条数据
    print ('-' * 30)
    print ('Data DESC')
    print (df.describe())  # 打印数据基本描述性信息
    print ('Data Dtypes')
    print (df.dtypes)  # 打印数据类型
    print ('-' * 60)
```

由于函数功能非常简单，因此不做详细描述，主要用法如下：

·在format函数中，df的形状由于只记录训练集X的特征数量，因此最后会减去1。

·head方法用来显示指定数量（N）的前N条数据。

·describe方法用来显示数据最小值、均值、标准差、25%、50%、75%分位数数据以及最大值。

·dtypes显示当前所有列的数据类型，输出结果可以用来辅助判断接下来要做的数据转换、处理等操作对数据类型的要求。

缺失值审查，用来查看数据集的缺失数据列、行记录数。

```
def na_summary(df):
    '''
    查看数据集的缺失数据列、行记录数
    :param df: 数据框
    :return: 无
    '''
    na_cols = df.isnull().any(axis=0)  # 每一列是否具有缺失值
    print ('NA Cols:')
    print (na_cols)  # 查看具有缺失值的列
    print ('-' * 30)
    print ('valid records for each Cols:')
    print (df.count())  # 查看每一列有效值（非NA）的记录数
    print ('-' * 30)
    na_lines = df.isnull().any(axis=1)  # 查看每一行是否具有缺失值
    print ('Total number of NA lines is: {0}'.format(na_lines.sum()))  # 查看具有缺失值的行总记录数
    print ('-' * 30)
```

本部分有以下几个重点方法：

·使用df.isnull（）.any（）判断指定轴是否含有缺失值，当设置axis=0时以列为判断依据，当设置axis=0时以行为判断依据。

·使用df.count（）统计每个特征有效值（非NA）的记录数，通过该方法可以获知每个特征缺失数据的情况是否严重。

·由于na_lines返回的是一个包含True/False的列表，因此可以使用sum（）方法统计其中缺失值的数量，统计时缺失值（True）会当1做统计，而False会作为0参与计算。

类样本均衡审查，查看每个类的样本量分布。如果样本分布不均衡则需要做样本均衡处理。有关样本均衡的具体方法请参见3.4节。

```
def label_summary(df):
    '''
    查看每个类的样本量分布
    :param df: 数据框
    :return: 无
    '''
    print ('Labesl samples count:')
    print (df['value_level'].groupby(df['response']).count())  # 以response为分类汇总维度对value_level列计数统计
    print ('-' * 60)
```

这里直接使用数据框的groupby方法以response（目标变量）为维度对value_level做计数统计。

步骤3　数据预处理。

经过步骤2的基本审查，对于数据的状态已经了然于心，在该步骤中需要做相应的数据预处理。该部分包含多个函数。

第一个函数变量类型转换，用来将分类和顺序变量数据类型转换为整数型。为什么要转换为整数型，而不是其他类型，例如字符串等？这是因为在接下来的数据变换过程中，我们会使用在3.2节中介绍的OneHotEncoder方法，该方法要求分类变量和顺序变量的值都必须是整数，如果是字符串等其他类型将报错。如果原始数据变量中存在字符型值，例如性别是M/F，那么需要先在该步骤中对这些值做替换即可。

```
def type_con(df):
    '''
    转换目标列的数据为特定数据类型
    :param df: 数据框
    :return: 类型转换后的数据框
    '''
    var_list = {'edu': 'int32',
                'user_level': 'int32',
                'industry': 'int32',
                'value_level': 'int32',
                'act_level': 'int32',
                'sex': 'int32',
                'region': 'int32'
                }  # 字典：定义要转换的列及其数据类型
    for var, type in var_list.iteritems():  # 循环读出列名和对应的数据类型
        df[var] = df[var].astype(type)  # 数据类型转换
    print ('Data Dtypes')
    print (df.dtypes)  # 打印数据类型
    print ('-' * 30)
    return df
```

在功能实现中，先定义了一个字典，字典的Key和Value分别是变量特征名和对应的数据类型，这里数值类型定义为int32；接着使用for循环读出字典的可迭代对象iteritems属性值，每次读出的属性值是包括Key和Value的键值对；然后应用astype对特征变量类型对转换，最后打印输出转换后的数据框的数据类型。

第二个函数是NA值替换，用来将不同列的缺失值替换为指定数值。我们在3.1节讲过很多种缺失值替换方法，这里使用的是根据每一

列自动指定缺失值的方法。

```
def na_replace(df):
    '''
    将数据集中的NA值使用自定义方法替换
    :param df: 数据框
    :return: NA值替换后的数据框
    '''
    na_rules = {'age': df['age'].mean(),
                'total_pageviews': df['total_pageviews'].mean(),
                'edu': df['edu'].median(),
                'edu_ages': df['edu_ages'].median(),
                'user_level': df['user_level'].median(),
                'industry': df['user_level'].median(),
                'act_level': df['act_level'].median(),
                'sex': df['sex'].median(),
                'red_money': df['red_money'].mean(),
                'region': df['region'].median()
                }  # 字典：定义各个列数据转换方法
    df = df.fillna(na_rules)  # 使用指定方法填充缺失值
    print ('Check NA exists:')
    print (df.isnull().any().sum())  # 查找是否还有缺失值
    print ('-' * 30)
    return df
```

功能实现过程中，先定义一个包括变量维度名称和对应的缺失值替换方法的字典，这里主要用到了均值mean和中位数median，其中均值用于数值型变量，中位数用于字符串变量；接着用df.fillna方法使用自定义的每列的不同方法批量替换缺失值；然后查看数据框中是否还存在缺失值并打印输出。

第三个函数做二值化的标志转换，主要用于将分类变量和顺序变量转换为二值化的标志0和1为值域的变量。在转换过程中，由于涉及训练集（及测试集）和预测集两种状态，训练集需要使用转换对象的fit方法训练，然后使用训练好的模型分别对训练集和预测集做转换，因此需要区分不同阶段。

```
def symbol_con(df, enc_object=None, train=True):
    '''
    将分类和顺序变量转换为二值化的标志变量
    :param df: 数据框
    :param enc_transform: sklearn的标志转换对象，训练阶段设置默认值为None；预测阶段
使用从训练阶段获得的转换对象
    :param train: 是否为训练阶段的判断状态，训练阶段为True，预测阶段为False
    :return: 标志转换后的数据框、标志转换对象（如果是训练阶段）
    '''
    convert_cols = ['edu', 'user_level', 'industry', 'value_level', 'act_lev
择要做标志转换的列名
    df_con = df[convert_cols]  # 选择要做标志转换的数据
```

```
    df_org = df[['age', 'total_pageviews', 'edu_ages', 'blue_money', 'red_mo
置不做标志转换的列
    if train == True:  # 如果处于训练阶段
        enc = OneHotEncoder()  # 建立标志转换模型对象
        enc.fit(df_con)  # 训练模型
         df_con_new = enc.transform(df_con).toarray()  # 转换数据并输出为数组格
式
         new_matrix = np.hstack((df_con_new, df_org))  # 将未转换的数据与转换后
的数据合并
        return new_matrix, enc
    else:
         df_con_new = enc_object.transform(df_con).toarray()  # 使用训练阶段获
得的转换对象转换数据并输出为数组格式
         new_matrix = np.hstack((df_con_new, df_org))  # 将未转换的数据与转换后
的数据合并
        return new_matrix
```

该函数的参数除了数据框外，还有两个参数：

·enc_transform：Sklearn的标志转换对象，该对象只有应用fit方法之后才能形成，因此训练阶段为空；在预测集应用阶段，直接使用该参数传入训练集时的对象即可。

·train：是否为训练阶段的判断状态，训练阶段为True，预测阶段为False。根据不同的状态判断是否需要做fit，并返回不同的参数对象。

在实现过程中：

先通过convert_cols定义要转换的特征变量名称列表，然后形成数据框df_con。

再使用df_org定义出无需转换的数据矩阵（数据框的值，使用values方法获取），df_org是一个Numpy矩阵而非Pandas数据框，接下来我们要使用Numpy方法做数据矩阵合并，原因是使用OneHotEncoder转换后的对象也是Numpy矩阵，这样就不需要再做转换了。

接下来要根据train的状态做判断，当状态为True时，先使用OneHotEncoder方法建立转化对象enc，然后应用fit方法做训练，再使用transform方法做转换并将结果使用toarray方法转换为数组并赋值给df_con_new。这里将fit和transform方法分开操作目的是enc对象要在fit之后传给预测集使用，如果直接使用fit_transform则无法实现。最后通过numpy的hstack方法将两个矩阵合并。当状态为False时，直接使用从训练阶段获得的对象enc做transform，然后将结果合并返回。

相关知识点：使用Numpy的hstack和vstack做矩阵合并

在之前的章节中我们使用了Pandas的merge和contact方法将多个数据框合并。在本节中使用了Numpy的hstack做矩阵合并。hstack是将矩阵以列为单位做合并，与之相对应的是以行为单位做合并的方法vstack，这两个方法的参数都是一个元组。

假如b=a=np.arange（6）.reshape（2，3），a和b都是2行3列矩阵：

当使用numpy.hstack（（a，b））做列合并时，合并后的矩阵是2行6列，此时该方法等价于numpy.concatenate（（a，b），axis=1）

当使用numpy.vstack（（a，b））做行合并时，合并后的矩阵是4行3列，此时该方法等价于numpy.concatenate（（a，b），axis=0）

由于需要对数据转换的方式保持一致，因此只能使用一次fit方法，分别对训练集和预测集做转换。在应用fit方法时，是针对训练数据进行的，这就要求训练集的数据值分布要比较完整，这样预测集应用才有效。例如，假如变量X1的训练集值域是1/2/3，预测集的值域是1/2，那么使用测试集的fit方法形成的对象应用到预测集做转换没有问题；但如果变量X的训练集值域是1/2，预测集的值域是1/2/3，由于应用fit方法时没有3这个值域分布，从而会导致应用到预测值时无法转换出来（简单点说就是训练集中没有这个值域，因此转换模式超过训练范畴）。

第四个函数是获得最佳模型参数，用于按照指定的参数训练方法得到每次交叉检验不同评估指标的分类模型结果。交叉检验是做模型效果评估的最佳方法，分类模型中一般都会使用该方法。在该函数中，我们通过交叉检验的方法，自定义了几种不同的检验指标，然后对模型的不同参数做训练并得到不同评估指标的结果，通过每次的输出结果以及运行时间的评估，从中找到比较合适的参数。

```
def get_best_model(X, y):
    '''
    结合交叉检验得到不同参数下的分类模型结果
    :param X: 输入X（特征变量）
    :param y: 预测y（目标变量）
    :return: 特征选择模型对象
```

```
    '''
    transform = SelectPercentile(f_classif, percentile=50)  # 使用f_classif
方法选择特征最明显的50%数量的特征
    model_adaboost = AdaBoostClassifier()  # 建立AdaBoostClassifier模型对象
                                model_pipe        =        Pipeline(steps=
[('ANOVA', transform), ('model_adaboost', model_adaboost)])  # 建立由特征选择
和分类模型构成的"管道"对象
    cv = StratifiedKFold(5)  # 设置交叉检验次数
    n_estimators = [20, 50, 80, 100]  # 设置模型参数列表
    score_methods = ['accuracy', 'f1', 'precision', 'recall', 'roc_auc']  #
置交叉检验指标
    mean_list = list()  # 建立空列表用于存放不同参数方法、交叉检验评估指标的均值列表
    std_list = list()  # 建立空列表用于存放不同参数方法、交叉检验评估指标的标准差列表
    for parameter in n_estimators:  # 循环读出每个参数值
        t1 = time.time()  # 记录训练开始的时间
         score_list = list()  # 建立空列表用于存放不同交叉检验下各个评估指标的详细数
据
        print ('set parameters: %s' % parameter)  # 打印当前模型使用的参数
        for score_method in score_methods:  # 循环读出每个交叉检验指标
            model_pipe.set_params(model_adaboost__n_estimators=parameter)  #
过"管道"设置分类模型参数
            score_tmp = cross_val_score(model_pipe, X, y, scoring=score_meth
用交叉检验计算指定指标的得分
            score_list.append(score_tmp)  # 将交叉检验得分存储到列表
        score_matrix = pd.DataFrame(np.array(score_list), index=score_method
交叉检验详细数据转换为矩阵
         score_mean = score_matrix.mean(axis=1).rename('mean')  # 计算每个评估
指标的均值
        score_std = score_matrix.std(axis=1).rename('std')  # 计算每个评估指标
的标准差
        score_pd = pd.concat([score_matrix, score_mean, score_std], axis=1)
原始详细数据和均值、标准差合并
        mean_list.append(score_mean)  # 将每个参数得到的各指标均值追加到列表
        std_list.append(score_std)  # 将每个参数得到的各指标标准差追加到列表
         print (score_pd.round(2))  # 打印每个参数得到的交叉检验指标数据，只保留2位
小数
        print ('-' * 60)
        t2 = time.time()  # 计算每个参数下算法用时
        tt = t2 - t1  # 计算时间间隔
        print ('time: %s' % str(tt))  # 打印时间间隔
    mean_matrix = np.array(mean_list).T  # 建立所有参数得到的交叉检验的均值矩阵
    std_matrix = np.array(std_list).T  # 建立所有参数得到的交叉检验的标准差矩阵
    mean_pd = pd.DataFrame(mean_matrix, index=score_methods, columns=n_esti-
mators)  # 将均值矩阵转换为数据框
    std_pd = pd.DataFrame(std_matrix, index=score_methods, columns=n_estimat
均值标准差转换为数据框
    print ('Mean values for each parameter:')
    print (mean_pd)  # 打印输出均值矩阵
    print ('Std values for each parameter:')
    print (std_pd)  # 打印输出标准差矩阵
    print ('-' * 60)
    return transform
```

在该函数的实现过程中，先要做数据降维，目的是降低数据计算量。这里使用SelectPercentile方法结合f_classif评估选择特征最明显的50%数量的特征。SelectPercentile方法用来按照指定方法选择特定数量或比例的特征变量，f_classif是计算数据集的方差P值方法。这两种方法

结合起来后形成的transform模型对象，该对象目前没有做任何操作，只是一个空对象而已。

这里的数据降维使用了特征选择方法，而不是像PCA、LDA等方法做数据转换。在实际应用过程中，由于考虑到业务方可能对结果的特征重要性要做分析，因此这里不使用转换方法。除了f_classif方法外，sklearn.feature_selection这对分类模型还提供了chi2、f_classif、f_regression、mutual_info_classif、mutual_info_regression等评估指标用于分类和回归检验。而SelectPercentile方法能够按照上述指定的评分方法对所有特征变量做评估，然后选择设置好的特征数量或比例。

然后通过AdaBoostClassifier方法建立分类模型对象，这两个模型对象要作为下面“管道”的两个步骤。

相关知识点：使用集成算法做模型训练

AdaBoost是一种非常流行的集成算法，其核心思想是针对同一个训练集训练不同的分类器（弱分类器），然后把这些弱分类器集合起来，构成一个强分类器。sklearn提供两类集成算法：

第一类是使用平均方法的集成算法，这类算法的基本原理是构建一些小的模型器然后基于每个模型器的结果做均值计算得到最终结果。这种组合方法单个模型的效果更好，因为其方差小。典型代表是Bagging方法和Random Forest。

第二类是使用提升方法的集成算法，在这种集成方法中，每个模型器顺序参与模型评估，并试图降低组合模型器的偏差。这是一种组合多个模型器来形成强大模型器的有效方法。典型代表是AdaBoost、Gradient Tree Boosting等。

集成算法相对于单个算法的显而易见的好处是其准确率较高，且鲁棒性强。我们在“4.2回归分析”中使用过GradientBoostingRegressor做回归预测，已经见识过它的效果。

接着使用Pipeline方法建立一个“管道”，目的是将上述两个过程（特征选择和分类算法）合并起来做交叉检验。该步骤实际上简化了交叉检

验的过程，后续只需要对管道对象做参数设置、训练和预测即可，而无需分别配置特征选择和分类训练过程。

相关知识点：使用Pipeline构建组合评估器

sklearn中的Pipeline（翻译为管道）是一个复合评估器，用来将多个具有上下逻辑环节的过程连接起来形成一个复合对象。

在数据处理和计算算法过程中，经常会用到数据标准化、数据转换、特征降维等方法，然后再对处理后的数据集做模型训练和评估，这个过程可以使用pipeline（例如sklearn.pipeline.Pipeline）形成具有序列步骤的组合评估器。

Pipeline（steps）只有一个参数“steps”，该参数是一个由名称和模型对象组成的元组的列表。这在这个列表中，不同的元组之间是有明确的先后关系，并且最后一个元组一定是一个评估算法。

当使用pipe组合对象时，可应用的方法包括fit、predict、fit_predict、fit_transform等。当对其使用predict（以及包含predict的其他方法，例如predict_proba、fit_predict等）会自动调用最后一个评估算法的predict方法做预测。

Pipeline（steps）方法本身是用来为交叉检验提供算法对象不同参数下的效果，进而得到最优模型效果参数的方法。

由于本案例中使用了很多新的用法，因此可能涉及的相关知识点有点多，对案例内容的连贯性产生影响。接着回到本书内容，使用StratifiedKFold方法设置5次交叉检验使用的训练集和测试集；n_estimators是Adaboost的n_estimators参数用到的值列表，score_methods是交叉检验指标列表，有关更多自定义交叉检验的方法请参照4.2.6节中的表4-3；然后分别建立mean_list和std_list用于用于存放不同参数方法、交叉检验评估指标的均值和标准差。

接下来进入到循环阶段。这里的训练包括2层逻辑，外层逻辑是按照不同的n_estimators参数值做算法交叉检验，内层逻辑是按照不同的交叉检验指标做交叉检验。

外层循环中，使用for循环遍历每个n_estimators参数，使用t1记录一个该参数下开始的时间戳，用来记录不同参数对应的算法运行时间；然后定义一个score_list空列表，用于记录每次n_estimators参数下的算法交叉检验结果。

内层循环中，使用for循环遍历score_methods中的每个交叉检验得分评估方法，设置组合模型器（“管道”）对象model_pipe的参数，通过步骤名称+__（两个下划线）+参数对象的方法设置，这里的步骤名称是在建立管道是定义的字符串名称，__（两个下划线用来表示连接对应步骤对象的参数），后面接具体参数的名称，例如model_adaboost__n_estimators=parameter的意思是设置model_adaboost的n_estimators参数的值为parameter（parameter为从外层循环读取的值）。接着使用cross_val_score方法建立交叉检验过程，分别设置模型对象为管道，数据集X和y以及交叉检验方法评分方法和交叉检验方法cv。最后将得到的结果使用append追加到score_list列表，获得每个参数下的评估结果。

在内层循环结束后，将每个参数下交叉检验详细数据转换为矩阵score_matrix，目的是便于接下来基于不同的交叉检验指标计算均值和标准差，并格式化为Pandas数据框做输出。然后分别基于score_matrix的行（每行代表一个指标，每列代表一次交叉检验结果）计算均值和标准差。接着使用Pandas的concat方法做Pandas对象合并，得到一个包含原始详细交叉检验指标数据和均值、标准差的总体数据集。最后，将每个参数下的均值和标准差数据使用append方法追加到mean_list和std_list，便于评估不同参数的效果；使用round方法打印输出保留两位小数的数据框，并得到最后时间戳并输出时间间隔。

步骤4　数据应用。

上面定义完各种功能函数后，本部分开始正式应用。

```
# 加载数据集
raw_data = pd.read_excel('order.xlsx', sheetname=0)  # 读出Excel的第一个sheet
X = raw_data.drop('response', axis=1)  # 分割X
y = raw_data['response']  # 分割y
```

使用Pandas的read_excel方法读取第一个sheet数据，由于从Excel读

取数据会比较慢，这里可能稍微要等一点时间（大概5秒钟），然后分割出X和y用于做后续处理。

```
# 数据审查和预处理
set_summary(raw_data) # 基本状态查看
```

基本状态查看结果如下：

```
Data Overview
Records: 39999 Dimension13
------------------------------
    age  total_pageviews  edu  edu_ages  user_level  industry  value_level
0  39.0          77516.0  1.0      13.0         1.0       1.0            1
1  50.0          83311.0  1.0      13.0         2.0       2.0            2
   act_level  sex  blue_money  red_money  work_hours  region  response
0        1.0  1.0        2174        0.0          40     1.0         0
1        1.0  1.0           0        0.0          13     1.0         0
------------------------------
Data DESC
                age  total_pageviews           edu      edu_ages  \
count  39998.000000     3.999800e+04  39998.000000  39998.000000
mean      38.589654     1.895136e+05      2.511626     10.076754
std       13.663490     1.053109e+05      1.638110      2.573384
min       17.000000     1.228500e+04      1.000000      1.000000
25%       28.000000     1.175282e+05      2.000000      9.000000
50%       37.000000     1.783410e+05      2.000000     10.000000
75%       48.000000     2.372685e+05      2.000000     12.000000
max       90.000000     1.484705e+06     10.000000     16.000000
......
Data Dtypes
age                float64
total_pageviews    float64
edu                float64
edu_ages           float64
user_level         float64
industry           float64
value_level          int64
act_level          float64
sex                float64
blue_money           int64
red_money          float64
work_hours           int64
region             float64
response             int64
dtype: object
```

上述结果中的特征数量、数据记录数、前2条数据样本查看以及数值型数据的描述性统计结果都比较简单。这里重点说下Dtypes输出，从输出结果可以看到，要做分类的特征包括edu、user_level、industry、act_level、sex、region都会识别为浮点型，而后面要做的二值化标志转

换需要转换为int型，这里的结果为下面做转换提供了参考依据。

```
na_summary(raw_data) # 缺失值审查
```

以下是缺失值审查结果：

```
NA Cols:
age                  True
total_pageviews      True
edu                  True
edu_ages             True
user_level           True
industry             True
value_level         False
act_level            True
sex                  True
blue_money          False
red_money            True
work_hours          False
region               True
response            False
dtype: bool
------------------------------
valid records for each Cols:
age                 39998
total_pageviews     39998
edu                 39998
edu_ages            39998
user_level          39998
industry            39997
value_level         39999
act_level           39998
sex                 39998
blue_money          39999
red_money           39998
work_hours          39999
region              39997
response            39999
dtype: int64
------------------------------
Total number of NA lines is: 12
```

从NA Cols的结果可以看到除了value_level、blue_money和work_hours外，其他特征都有缺失值；从valid records for each Cols结果中分析发现，这些列的缺失值不多，完整数据记录是39999条而数据记录缺失最多的特征也只有2条而已；从Total number of NA lines is得到总的具有缺失值的记录是12条，由此我们判断缺失值情况不严重。

```
label_summary(raw_data) # 类样本均衡审查
```

以下是样本均衡审查结果：

```
Labesl samples count:
response
0    30415
1     9584
Name: value_level, dtype: int64
```

从结果中发现，二分类的样本分布相对均匀，没有出现量级上的巨大差异，因此无需做类样本均衡处理。

接下来先做缺失值（NA）替换，为什么要先做替换而不是先做数据类型转换？这是因为NA数据无法转换为整数型，转换必须针对非NA数据才行。

```
X_t1 = na_replace(X) # 替换缺失值
```

缺失值替换之后，检查含有NA值的样本数量为0。

```
Check NA exists:
0
```

接着做数据类型转换。

```
X_t2 = type_con(X_t1) # 数据类型转换
```

数据类型转换之后的结果符合预期：

```
Data Dtypes
age                float64
total_pageviews    float64
edu                  int32
edu_ages           float64
user_level           int32
industry             int32
value_level          int32
act_level            int32
sex                  int32
blue_money           int64
red_money          float64
```

```
work_hours           int64
region               int32
dtype: object
```

标志转换没有打印输出结果，只有功能项。读者可以自行通过X.shape和X_new.shape打印输出发现原始数据集X和转换后的数据集X_new的特征变量数量分别为13和92。

```
X_new,enc = symbol_con(X_t2,enc_object=None,train=True) # 将分类和顺序数据转换为标志
```

下面是分类模型训练过程。

```
# 分类模型训练
transform = get_best_model(X_new, y)  # 获得最佳分类模型参数信息
transform.fit(X_new, y)  # 应用特征选择对象选择要参与建模的特征变量
X_final = transform.transform(X_new)  # 获得具有显着性特征的特征变量
final_model = AdaBoostClassifier(n_estimators=100)   # 从打印的参数均值和标准差信息中确定参数并建立分类模型对象
final_model.fit(X_final, y)  # 训练模型
```

首先通过get_best_model获得最佳分类模型参数信息并返回转换对象，该过程会非常耗时，原因是我们在设置不同的算法参数时的同时又设置了不同的交叉检验指标。笔者的工作环境下大概需要3分钟的时间完成该步骤。该步骤输出每个参数下的数据示例如下：

```
set parameters: 100
              0     1     2     3     4  mean   std
accuracy   0.86  0.86  0.87  0.87  0.86  0.86  0.00
f1         0.67  0.67  0.69  0.70  0.69  0.68  0.01
precision  0.76  0.77  0.78  0.79  0.76  0.77  0.01
recall     0.60  0.60  0.61  0.63  0.62  0.61  0.01
roc_auc    0.92  0.92  0.92  0.92  0.92  0.92  0.00
------------------------------------------------------------
time: 110.470000029
```

当设置n_estimators=100时，意味着Adaboost会使用的最大单一评估模型（小模型）的数量为100个。该步骤耗时将近2分钟。

这里设置的n_estimators的值指的是会使用小模型器的最大数量，这意味着当模型结果已经足够好时，将不会使用更多的模型器，

因此当继续加大该参数的数量时，可能会无法产生更好的结果。

然后对比不同参数下的各交叉检验的均值和标准差，结果如下：

```
Mean values for each parameter:
                20        50        80        100
accuracy   0.853946  0.859922  0.862647  0.863672
f1         0.656329  0.672173  0.679723  0.682772
precision  0.753126  0.764994  0.770094  0.771560
recall     0.582011  0.599542  0.608411  0.612376
roc_auc    0.908324  0.914994  0.918613  0.919941
Std values for each parameter:
                20        50        80        100
accuracy   0.005260  0.004636  0.004463  0.004834
f1         0.009512  0.012836  0.011527  0.011929
precision  0.023295  0.010208  0.011250  0.012753
recall     0.013767  0.016128  0.013591  0.013184
roc_auc    0.003201  0.002865  0.002629  0.002652
```

但是对比各方面的评估指标，我们发现参数设置为80时的结果相对于100基本一致（其实小数点后面还有更多的位数能够体现出更多小模型器的结果会更好，但这种效果提升已经不明显）。

然后使用tranform对象对数据集X_new做特征选择得到最终建模特征数据X_final，最后使用AdaBoostClassifier模型以及最佳参数80做算法训练。

经过上述步骤，我们完成了模型训练并确定了最佳模型Adaboost的参数，接下来对新数据集做预测。

```
# 新数据集做预测
new_data = pd.read_excel('order.xlsx', sheetname=1)  # 读取要预测的数据集
final_reponse = new_data['final_response']  # 获取最终的目标变量值
new_data = new_data.drop('final_response', axis=1)  # 获得预测的输入变量X
set_summary(new_data)  # 基本状态查看
na_summary(new_data)  # 缺失值审查
new_X_t1 = na_replace(new_data)  # 替换缺失值
new_X_t2 = type_con(new_X_t1)  # 数据类型转换
new_X_t3 = symbol_con(new_X_t2, enc_object=enc, train=False)  # 将分类和顺序数据转换为标志
new_X_final = transform.transform(new_X_t3)  # 对数据集做特征选择
```

先使用跟训练集相同的方法做数据读取，这里通过sheetname=1设置读取第二个sheet。


使用read_excel方法读取Excel时，可以通过参数sheetname定义要读取哪个sheet的数据，其值可以定义为4种，数值、字符串、数值和字符串的混合列表以及Noen。其中数值代表sheet的索引值，0代表第一张sheet；字符串代表要读取的sheet名称；数值和字符串的混合允许将索引值和sheet名混合起来应用，例如[0，1，"Sheet5"]；而None则默认读取第1张sheet的内容。

由于上述过程跟训练集相同，在此不作赘述。其中一个final_reponse变量的用途会在下文讲到。

接下来做预测。

```
# 输出预测值以及预测概率
predict_labels = pd.DataFrame(final_model.predict(new_X_final), columns= ['l
得预测标签
predict_labels_pro = pd.DataFrame(final_model.predict_proba(new_X_final), cc
得预测概率
predict_pd = pd.concat((new_data, predict_labels, predict_labels_pro), axis=
预测标签、预测数据和原始数据X合并
print ('Predict info')
print (predict_pd.head(2))  # 打印前2条结果
print ('-' * 60)
```

先使用final_model（训练后的AdaBoostClassifier模型对象）的predict方法做分类预测，得到的结果转换为pandas数据框；然后读取final_model的predict_proba属性获得其预测为正例和负例的概率值并转换为数据框，最后将原始数据预测集、预测标签、预测概率数据框按列合并，打印前2条数据，如下所示：

```
Predict info
   age  total_pageviews  edu  edu_ages  user_level  industry  value_level  \
0   61           243019   10         1         2.0       7.0            2
1   33           215596    4         5         2.0       7.0            2
   act_level  sex  blue_money  red_money  work_hours  region  labels  \
0          1    1           0          0          40     1.0       0
1          5    1           0          0          40     6.0       0
       pro1      pro2
0  0.504053  0.495947
1  0.507486  0.492514
```

预测得到的最终结果，我们写到一个Excel中便于业务做对照和进一步整理分析。

```
# 将预测结果写入Excel
writer = pd.ExcelWriter('order_predict_result.xlsx')  # 创建写入文件对象
predict_pd.to_excel(writer, 'Sheet1')  # 将数据写入sheet1
writer.save()  # 保存文件
```

先使用pandas的ExcelWriter方法创建一个写入文件，名称为order_predict_result.xlsx；然后直接使用目标数据框predict_pd的to_excel方法对文件对象写入数据到sheet1中，最后保存。

使用pandas的写入Excel功能时，需要先根据写入Excel扩展名的不同安装xlwt或openpyxl。具体参照本节开始的导入库部分的介绍。

后续与实际效果的比较：

由于本案例在运营活动结束后统计了最终结果，其值位于final_reponse中，笔者将预测结果与实际结果进行了对比：

```
print ('final accuracy: {0}'.format(accuracy_score(final_reponse, predict_la
```

输出结果为：final accuracy:0.862490105168。准确率跟做交叉检验时非常一致，这也说明了该模型的鲁棒性非常强。

5.8.5 案例数据结论

由于本案例是一个预测性质的应用，其结果直接给到业务部门，因此没有分析型的数据结论。从最终的预测结果与真实应用结果的对照来看，业务部门对86.2%的准确率表示基本满意，符合预期。

5.8.6 案例应用和部署

本案例的结果给到业务部门后，前期业务部门有以下动作：

·制定了营销响应率不低于80%的KPI作为本次营销活动的绩效考核目标。

·针对80%的基准线结合已经产生订单的历史数据算出了相应订单金额、订单数量等基本数据，作为本次活动的预期收益；同时跟此次活动的预算相结合，制定了ROI目标（但不作为考核）。

·基于预期的订单金额和订单数量，以及关联的用券数量和金额，向公司申请了对应的优惠券用于促销用户购买转化。

由于该模型是一个常规性应用，基本上是每月一次的频率，而在应用过程中一般没有特殊参数调整或算法变动。因此，笔者将该模型经过优化并配合其他部门做调整后，由后端部门将其封装为固定程序执行。主要调整内容包括：

·数据源从Excel调整为从MySQL数据库直接取数，有关MySQL取数的具体方法请参照2.2.3节的内容。

·数据源的周期，默认取最近1年的数据，数据根据不同取数时间自动滚动。

·预测集由于是变动的，需要业务部门每次根据不同的情况选择特定的会员列表。会员部门已经可以通过辅助决策平台自助做会员筛选，然后将选择列表加入到模型中作为预测集。

·模型的运行以业务部门的触发为条件，当业务部门选择会员列表加入模型对象后便触发执行操作，而非自动执行。

·模型由于在服务器上执行，因此数据结果生成后直接Excel链接的形成提供给会员部门，会员部门可直接下载。

·在从数据库取出数据后，跟DBA（数据主管人员）反复沟通了关

于数据基本状态的问题，然后针对性地优化了关于数据查缺补全的策略，包括空值、异常值、缺失值等的数据清洗操作和步骤。

·经过多次测试，在最耗时的“获得最佳模型参数”阶段已经基本确认了最佳参数，无需每次运行，因此该步骤在后期已经省略，直接应用模型和最佳参数。

5.8.7 案例注意点

本案例中有以下几个点需要读者重点关注：

由于涉及二值化标志转换的问题，如果训练集比较小，无法完全覆盖完整的值域分布，那么笨案例中的二值化转化将无法应用。

本案例中的特征选择的数量是影响分类结果评估准确与否的关键因素之一，特征数量越多最终分类准确率越高的可能性越大（还要取决于分类算法的可靠性），如图5-6所示是在digits数据集结合方差分析和SVM的管道在不同特征数量下的准确率对比。

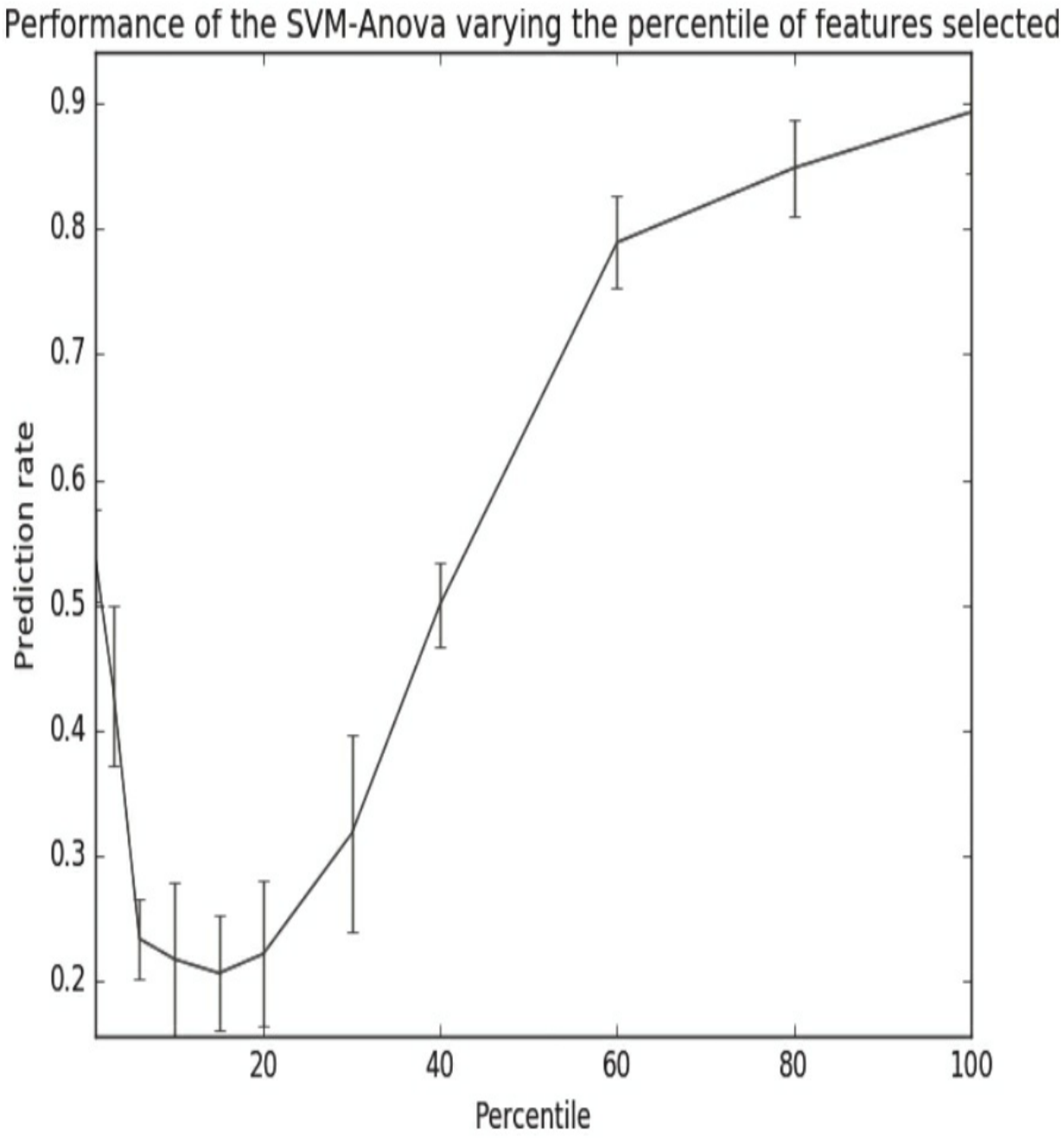


图5-6　特征比例对于准确率的影响

Adaboost应用时小分类器的数量（n_estimators参数值）将严重影响模型计算时间，因此对于有时间性要求的场景和应用要谨慎设置该参数（调低数值）或选择其他更快的方法。

交叉检验的几个指标具有非常高的相关性，相关系数超过0.99，因此读者在实际应用中选择其中一个使用即可，无需多次“重复”证明。

5.8.8 案例引申思考

案例中使用的“管道”用来组合多个序列步骤，但实际上组合方法其实有更多用途，例如：

·在构建模型过程中，假如希望使用多个算法协同完成工作，例如做分类时将SVM、随机森林、Adaboost组合起来使用，形成一个更强大的“组合模型”，此时也可以使用pipeline（例如sklearn.pipeline.make_pipeline）做组合。

·在做数据降维时，我们可能希望使用多个方法将不同方法评估得到的特征组合起来来时，此时可以使用pipeline（例如pipeline.FeatureUnion）构建一个组合特征选择器，这是非常有效的一种方法。

总体上，基于特征数量（比例或绝对数量值）来选择特征的方法存在一定风险性，主要原因是无法确定多少特征为最佳数量。在这方面，使用PCA的方法的可靠性要好于特征选择方法，因此PCA这类方法可以基于解释方差的比例选择主成分的数量，而非固定主成分。

5.9 本章小结

内容小结：本章的会员数据化运营讲到的知识不多，但都围绕实际读者最可能用到的场景展开，可落地性和应用性较强，这也是后面几章的显著性特征。

重点知识：本章最重要的几个内容是会员数据化运营分析模型、会员数据化运营分析小技巧以及最后的两个实际案例，尤其是案例中提到了一些之前没有介绍过的知识，这些知识都是日常应用中的常见方法，希望读者能够熟练掌握。具体包括：

·将数据保存为csv、Excel文件以及写回数据库；

·使用自定义方法建立RFM模型；

·使用“管道”方法Pipeline建立组合模型器；

·使用集成方法Adaboost做分类训练和预测，并测试了不同分类器数量下的性能；

·使用交叉检验方法cross_val_score以及自定义的多种评估指标（accuracy、f1、pre-cision、recall、'roc_auc）做模型效果分析。

外部参考：本节中涉及新的相关知识点中笔者认为有必要进一步学习和了解的部分。

·有关sklearn.pipeline（管道）的更多用法，包括特征组合、模型组合等请参照http://scikit-learn.org/stable/modules/classes.html#module-sklearn.pipeline。

·有关sklearn更多集成方法的介绍，请详见http://scikit-learn.org/stable/modules/ensemble.html。

·协同过滤以及个性化推荐是一个大命题，如果要写这个恐怕需要一本书，有关这方面的内容请查看《推荐系统》（《Recommender systems:An introduction》）。

应用实践：本章中的模型和小技巧是笔者做会员分析时的“利器”，而且每个模型笔者也都介绍了实施方法，希望读者能在充分了解后结合自身情况加以应用和实践。尤其需要注意的是，同一个模型应用到不同的公司和场景下会有不同的效果，以案例2的营销响应预测为例，86%其实不算是一个“特别出彩”的结果，但是如果结合到更多的数据集以及更多的数据特征变量，那么会使得该模型的效果得到显著性的提升；同样的应用其他模型也会很多优化和提升点可以挖掘。这些需要读者在实践中才能深度体会，仅凭书中的只言片语难以洞察到。

第6章　商品数据化运营

商品运营是销售型公司的核心工作之一，本章将围绕商品数据化运营展开，内容包括概述、关键指标、应用场景、分析模型、分析小技巧、分析“大实话”以及应用案例。

6.1 商品数据化运营概述

本章的商品指的是狭义上的实物商品，不包括有偿服务、虚拟商品等。商品跟产品的概念在很多场合下可以互用，但在互联网领域，产品也可用来表示与用户交互的载体，例如APP、网站等。这类产品的概念不是本章所指的商品范畴之内。

数据在商品运营过程中扮演了非常重要的角色，几乎随着商品工作的销售产生便已经开始，它涵盖了商品运营的方方面面。从销售预测到库存管理、从商品结构优化到动销管理、捆绑策略到关联组合等各方面都需要数据支持；在海量商品数据和复杂用户购物需求的背景下，通过数据来发现销售规律已经成为商品运营的关键。

6.2 商品数据化运营关键指标

商品数据化运营的关键指标包括销售类指标、促销活动类指标、供应链类指标。这里列出了每类中最常用的指标。

6.2.1 销售类指标

1.订单量/商品销售量

订单量指用户提交订单的数量，计算逻辑去重后的订单ID的数量。

商品销售量又称销售件数，指销售商品的数量。

订单量用来衡量唯一订单的数量，而商品销量用来衡量商品的总数量。例如订单编号为A901的订单中包含A和B两种商品，A商品数量为1，B商品数量为2，那么该订单的商品销售量总计为3，而订单量为1。

2.订单金额/商品销售金额

订单金额为用户提交订单时的金额，又称为应付金额。订单金额是用户真正应该支付的金额。计算公式为：

订单金额=商品销售金额+运费-优惠凭证金额-其他折扣（如满减）

其中：运费指未满足免邮费的订单需要支付的配送费用，优惠凭证金额指通过优惠券、积分兑换、会员卡等可作为金额使用的抵充金额，其他折扣信息包括满减信息（如满1000减200）等。

商品销售额是指商品销售的金额，商品销售额与订单金额的区别在于没有计算任何其他费用或优惠金额。计算公式为：

商品销售额=商品销售单价×销售数量

订单金额和商品销售金额都会作为商品总销售收入的评估指标，前者主要侧重于用户实际付款，后者侧重于总收入。

3.每订单金额/客单价/件单价

每订单金额指平均每个订单的金额，它是以订单为单位计算。公式：

每订单金额=订单金额/订单量

客单价指平均每个用户的下单金额，是以用户为计算单位。公式：

每订单金额=订单金额/订单用户量

件单价又称每件商品价值，指平均每个商品的销售价格。公式：

件单价=订单金额/商品销售量

这三个指标用来评估单位对象的价值产出，分别侧重于订单个体、用户个体和商品个体。

4.订单转化率

订单转化率是电子商务网站最重要的评估指标之一，大多数网站分析工具的计算公式为：订单转化率=订单量/总访问量或订单转化率=订单量/总UV量，这种计算方式实际上不科学，原因是它衡量的不是人的转化比例。假如一个用户下多个订单或网站存在订单拆分的情况，会使订单量大大高于实际订单人数，导致订单转化率虚高。正确的计算方法为：

订单转化率=产生订单的访问量/总访问量或产生订单的UV/总UV量

这里没有使用用户ID作为分子和分母，原因是大多数用户在到达网站时不会主动登录，只有在有订单或必须用户登录的条件下才能获取用户ID信息，无法统计到的用户ID便不会加入到分母后，导致数据缺失。

5.支付转化率

支付转化率是衡量用户支付转化的数据指标，是用户完成购物的重要步骤，更是企业产生真实销售价值的关键。计算公式为：

支付转化率=完成支付的客户数/需要支付的客户数

支付转化率是针对选择先款后货客户的转化评估指标，因此它只能评估订单用户中的一部分。对于选择货到付款的用户无需该指标评估，用户会在配送验证的时候支付。另外，由于每个订单都对应真实的客

户，因此这里客户数计算支付转化率。

6.有效订单量/有效订单金额/有效商品销售量/有效商品销售额

当用户提交订单后，其订单状态不一定有效，原因是销售公司都有订单审核过程会导致部分订单无效，并且很多用户下单后也会主动取消订单或由于超出特定订单规则外而被动取消。不同公司对有效的定义会有差别，但基本逻辑类似，有效订单中会去除审核未通过、作废、主动取消等无效状态。

·有效状态下的订单数量为有效订单量。

·有效状态下的订单金额。

·有效状态下的商品销售量为有效商品销售量。

·订单状态为有效的商品销售金额。

根据上述指标还可以延伸后有效件单价、有效订单状态率、有效客单价等。

7.订单有效率/废单率

订单有效率是用来衡量订单有效比例的重要指标，计算公式为：

订单有效率=有效订单量/订单量

订单有效率从订单发生时随着时间推移开始下降，直到所有订单完成妥投才处于稳定状态。大多数电子商务企业的订单有效率在60%以上，如果低于该数据值可能说明订单中包含大量作弊或无效订单。

与订单有效率相对的指标是“废单率”，废单率是所有订单中作废的订单比例，计算公式为：

废单率=1–订单有效率

8.毛利/毛利率

毛利是衡量自营商品利润最重要的指标之一，公式：

毛利=商品妥投销售额–商品批次进货成本

这里的毛利仅指销售毛利，即通过商品进销差价计算的毛利，没有考虑商品促销费用、配送费用、活动推广费用及其他摊销费用；另外，公式中使用“商品批次进货成本”计算毛利，原因是相同的商品在不同批次下进货成本可能不同，因此需要使用相应批次的进货成本。

毛利率是考察自营商品盈利情况另外一个最重要的指标，毛利和毛利率综合反映了商品的盈利规模和盈利能力。毛利率计算公式为：

毛利率=毛利/商品妥投销售额

所有的毛利计算基本都是以妥投状态为计算准则。

6.2.2 促销活动指标

1.每订单成本/每有效订单成本

每订单成本指完成每个订单需要的成本。公式：

每订单成本=费用/订单量

对于公式中的费用，不同部门有不同的费用支出情况。比如针对广告部门的费用通常只包含广告费用：

每订单成本=广告费用/订单量

对于运营类部门的费用可能只包含促销类费用，如优惠券费用。广告部门中存在一种按订单付费的合作方式CPS，这种方式基于成交订单而支付给推广媒介一定比例的佣金或返点。这也属于订单成本的一种。

每订单成本核算的是每个“毛”订单成本，订单中包含了所有状态（包含无效状态），只适合评估部门级别业务效果或做企业的初级评估指标。

每有效订单成本与每订单成本计算逻辑相似，不同点在于该指标只计算所有订单中有效订单的部分。计算公式为：

每有效订单成本=费用/有效订单量

该指标中仅包含有效订单状态的订单成本，是针对企业级别的真实评估指标。

由于有效订单也不一定最终妥投，其中有些是进行中的状态，因此最终产生价值的订单属于妥投订单，通过妥投订单计算得到的成本才是最终订单成本。

2.每优惠券收益/每积分兑换收益

每优惠券收益指每张优惠券能带来的收益。公式：

每优惠券收益=优惠券带来的订单成交金额/优惠券数量

由于企业往往发送不同的优惠券类型以及面值，通常需要在此基础上分别计算每种类型和币值优惠券带来的收益水平。

积分与优惠券类似，都是用来衡量优惠促销对销售的拉动情况。公式：

积分兑换收益=使用积分兑换的订单成交金额/积分兑换量

在实际业务中，由于用户往往可以在同一个订单中同时使用积分和优惠券，因此可能会出现订单贡献重复计算的情况。

3.活动直接收入/活动间接收入

活动直接收入指单纯通过促销活动带来的收入，用户购买的订单均属于促销活动商品。

活动间接收入指通过促销活动带来的用户购买了非活动商品的收入情况。通常计算活动间接收入的逻辑是该用户通过促销活动引入且订单属于非活动商品，通过促销活动引入可通过定义用户落地页是活动页面加以区分，订单属于非活动商品可通过参与活动商品列表进行拆分。

4.活动收入贡献

活动收入贡献包含活动直接收入贡献和间接收入贡献的总和，用活动收入贡献总金额除以全站订单成交金额得出活动收入贡献占比，公式：

活动收入贡献占比=（活动直接收入+活动间接收入）/全站订单成交金额

当然，除了可以用订单成交金额计算外，还可以使用订单量、商品销售量等计算活动贡献情况，其逻辑相同。

5.活动拉升比例

活动拉升比例指活动对全站销售的拉升情况，可以指销量拉升、销售额拉升、订单量拉升等。

活动拉升比例通常不能使用活动贡献占比来评估，原因是活动促销期间本来应该通过正常流程和渠道购物的用户反而会通过促销渠道下单。最简单的计算方法是用活动期间的收入与非活动期间的收入进行对比，计算公式为：

活动拉升比例=（活动期间收入/非活动期间收入）-1

在通常情况下，在计算收入拉升比例会发现收入拉升效果不如订单量和销量明显，原因是通常促销客单价较低，影响收入提升效果。

6.2.3 供应链指标

1.库存可用天数

库存可用天数反映了当前库存可以满足供应的天数，是仓库备货能力的一个体现。公式：

库存可用天数=库存商品数量/期内每日商品销售数量

库存可用天数越长代表可用时间越多，但过长的可用天数可能意味着商品滞销，因此库存可用天数需要保持在一定范围内。

库存可用天数通常会按照时间划分，不同商品的可用天数需要根据库存周转天数来定义，假如商品库存周转天数是30天，那么可以将库存天数划分为7天以内、8～14天、15～30天、30天以上等。

2.库存量

库存量是指一定周期内全部库存商品的数量。库存量的定义中包括多种状态的商品，例如正常可售卖商品、已被订购但未发货商品、残次商品、调拨未出库商品、调拨未入库商品等。因此，某些情况下，可能出现商品有库存但无法销售的情况。

企业通常会定义安全库存量、最低库存量和最高库存量，目的是保证商品在一定程度上可满足用户购买需求，同时不至于造成商品积压。如果低于最低库存量可能造成商品缺货，高于最高库存量可能造成商品滞销。

安全库存量=每日商品销量×正常到货时间（天）+P

最低库存量=每日商品销量×紧急到货时间（天）+P

最高库存量=每日商品销量×最长到货时间（天）+P

其中的P为调节参数，包含企业销售任务、节假日因素、仓储运维等因素。

3.库龄

商品从进入仓库时便开始产生库龄，一般意义上的库龄指的是商品库存时间。公式：

库龄=出库时间-入库时间

仓储系统中一般按照先进先出、先进先销的原则出库，因此同一个商品的库龄要按照其相应进货批次的时间计算。

库龄通常会按照时间划分不同区间，如1～30天库龄、31～60天库龄、61～90天库龄等，不同商品周转天数不同，划分时间段也有所差异。库龄时间过长意味着商品进入滞销阶段。

4.滞销金额

滞销指商品周转天数超过其应该售卖的周期，导致无法销售出去的情况。

滞销金额可以衍生滞销金额占比、滞销SKU占比、滞销商品销量占比等指标，各指标计算逻辑类似，以滞销金额占比为例，公式：

销售金额占比=滞销金额/库存金额

滞销一方面会造成资金积压影响资金流动，另一方面会造成产品过季、过保质期或淘汰周期而导致产品损毁或下市。

5.缺货率

缺货是相对于滞销的另一个极端，缺货意味着库存商品无法满足用户购买需求。缺货率计算公式为：

缺货率＝缺货商品数量/顾客订货数量

缺货率同样可以衍生其他指标定义：

缺货金额=缺货商品数量×缺货商品单价

缺货商品数量=顾客订货数量–库存商品数量

6.残次数量/残次金额/残次占比

残次指的是由于商品库存、搬运、装卸、物流、销售等因素造成的商品外包装损坏、产品损坏、附件丢失等影响商品二次销售的情况。

残次数量指残次商品的数量。

残次金额指残次商品的进货成本，公式：

残次金额=残次商品批次进货单价×残次商品数量

残次占比用来衡量残次商品在整个仓库中的比例，计算方法为：

残次金额占比=残次商品金额/库存商品金额

残次数量占比=残次商品数量/库存商品数量

7.库存周转天数

库存周转天数是用时间表示库存的周转速度，指的是从商品进货开始到最终完成销售或损毁所经历的天数。天数越少说明周转越快，公式：

库存周转天数=360/库存周转率

其中：

库存周转率=年销售商品金额/年平均库存商品金额

6.3 商品数据化运营应用场景

商品数据化运营的主要应用场景包括销售预测、库存分析、市场分析和促销分析。

6.3.1 销售预测

无论企业规模如何，销售预测都是销售管理非常重要的一环，也是做计划、管理、预算和目标的基础。销售预测主要应用的是销售预测场景，通过对历史数据的分析预测未来一段时间内企业可能产生的销售额、销售量或订单金额等。典型场景例如：

·未来一周会产生多少商品销售量？

·如果给销售部门50000元促销费用，预期能带来多少订单？

·下个月估计能产生多少毛收入？

通过销售预测能得到未来关于目标的预测数值，该预测值既可以用于评估相应的资源投入，也可以针对性的做商品销售策划。

6.3.2 库存分析

库存分析是商品动销分析的关键点之一，也是商品销售的基础和前提。库存分析主要用来解决以下几类问题：

·当前的商品结构如何？是否具有合理的广度和深度组合？

·库存中的滞销商品金额有多少，滞销时间有多久？

·当前M商品的可用天数是否能满足销售预期，需要补货多少？

·平均商品库龄是多少，如何提升商品周转并降低库龄？

·如何设置安全库存警戒线？

·如何管理季节性库存商品，来满足季节性促销活动？

·如何找到大龄库存商品并合理安排销售周期，防止商品过期、过季？

·如何找到商品的最佳库存位置，以实现更高效率的分拣、包装和出库？

库存分析的关键是找到脱销和滞销的平衡点，能在不变的库存状态下，最大化满足商品周转并降低库存资金占用。

6.3.3 市场分析

商品数据化运营中的市场分析主要应用于对商品所在市场的规模、特点、容量、性质、趋势等方面的宏观分析，主要侧重于解决以下几方面问题：

·公司要打造新产品，该产品的市场容量有多少？预期年销售量有多少？

·用户对于产品的关注点在哪里？最满意和最不满意的点都分别是哪些？

·新产品要上线，售价应该定在多少会比较合适？

·产品C的市场竞争对手是谁？它们各自具备哪些优势和不足？

·P手机产品现售价4800元，预计3个月后售价是多少？

·北京的用户对于商品的预期需求跟上海地区有哪些差异？

商品市场分析既可以侧重于单个商品，也能侧重于品牌、品类等更高聚合的维度，并且能从宏观角度评估所有商品本身以及所处市场的优劣得失。

6.3.4 促销分析

促销分析是商品数据化运营应用最为广泛的场景之一，现在几乎每个企业都形成了以促销带销售的运营节奏。数据对于促销分析的主要应用包括以下几类场景：

·制定打包和组合策略，使得用户单次购买商品金额最大化？

·制定商品向上销售策略，购买了家电的用户下次还可能购买什么？

·促销资源分析，明日商品活动的目标销售额是5000万，预计需要多少促销费用？

·精准商品销售或推介，企业目前有10000件商品需要清仓处理，如何快速销售出去？

·恶意促销订单、作弊订单的检测和分析，当前订单中有哪些是疑似黄牛的订单？

·促销方式分析，不同的促销方式下，哪种最有利于销售提升并能使总体销售最大化，而不是全部商品都做促销？

·商品定价，针对M商品应该制定促销价是多少能满足销售额最大化的需求？

·商品陈列分析，如何摆放不同的商品能促销连带销售的最大化？

·组合方式分析，大型活动时应该如何将不同的促销方式和折扣手段结合起来，以产生最大的活动收入？

促销分析涵盖的策略制定、实时监测、后期分析等各个场景都是商品运营非常关注的环节，也是数据产生可量化价值的主要场景。

6.4 商品数据化运营分析模型

本节将介绍几个常用的商品分析模型，包括商品价格敏感度模型、新产品市场定位模型、销售预测模型、商品关联销售模型、异常订单检测模型、商品规划的最优组合。

6.4.1 商品价格敏感度模型

商品价格敏感度模型是指通过研究找到用户对于价格是否敏感以及敏感程度的价格杠杆。利用价格敏感度模型可以辅助于销售定价，促销活动的折扣方式、参考价格、价格变动幅度等方面的参考。例如：

·促销活动时是否应该包含M2商品。

·当商品M3提价100元时，订单量会如何变化。

·在商品详情页的参考价格应该定为多少才能让客户感觉到已经降价并触发下单动作。

·满减、满返、跨品类用券等哪些方式最适合M4商品。

商品价格敏感度分析可以通过两种方式实现。

（1）调研问卷法

通过调研问卷的形式针对关注的品类或商品做调研分析是比较通用的一种方法。这种方法可以获得品类详细信息，并且可以通过问卷设置不同的关注信息点，收集到的信息更符合实际需求。

但是，当面临新的价格敏感度分析需求时，通常都需要重新开展调研分析工作。这种方式实施起来周期较长且反馈结果较慢，另外，当要收集的商品信息较多时，可能很难获得完整数据。

（2）数据建模法

通过数据建模的方式建立商品价格和销售量之间的关系模型是研究价格敏感度的有效方法。这种方法实施起来相对简单：

首先，收集不同价格下的销售数据。价格敏感度模型需要有基于不同价格下的销售数据产生，因此需要商品运营部门针对性的做调价。这种调价动作根据需求的不同，可能是长期的，也可能是短期的。长期的调价是一种“自然状态”，因为在一个较长周期内商品会经历生命周期的不同阶段，并结合商品促销、打折等运营工作产生多种价格和销售数

据；而短期的调价更多的是为了采集数据而产生。

其次，数据建模分析。商品价格敏感度模型关注的主要是价格和销量之间的关系，可以用回归方法来解决。在回归方法中，自变量中除了价格外，还需要包含其他两类信息：

·商品信息，商品品类、上市时间、同期竞争对手价格、是否参与促销活动、促销方式、折扣力度、通用属性等。

·客户信息，客户性别、年龄、收入、学历、会员级别、历史订单量、品类偏好度、活跃度、价值度等。

之所以要将大量的商品信息和客户信息加入到回归模型中，是因为如果只针对价格和商品销售量做回归，那么价格本身能解释的商品销售量变化可能会非常有限，销量的变化还可能受到其他很多因素的影响，因此要在控制这些干扰因素的前提下做回归模型。

关于回归方法的选择，具体请参照4.2.5节的内容。

6.4.2 新产品市场定位模型

新产品市场定位分析用于当企业新生产或策划一款产品时，需要根据市场上现有的竞争对手产品情况做定位分析。该分析的目的是评估新产品与哪些产品能形成竞品关系，可以针对性地找到与竞品的差异性和优势，例如功能特点、使用周期、产品质量等，从而应用到产品定价、市场宣传、渠道推广等方面。

新产品市场定位分析可以通过基于相似度的方法实现。例如，使用非监督式的KNN（K近邻），模型的核心是通过对新产品的数据与现有数据的比较，发现跟新产品相似的其他产品。通过KNN实现新产品市场定位分析的步骤如下：

步骤1　数据准备。先准备好要训练的数据集，由于这不是一个分类应用，因此数据集中只包含不同竞品的特征变量即可，无需目标变量。

步骤2　数据预处理。预处理过程根据数据集情况可能包括二值化标志转换、缺失值处理、异常值处理、数据标准化等。需要注意的是，由于是基于距离的计算，分类和顺序变量需要做二值化转换，异常值（包括量纲和值的异常）都会对相似度产生重大影响。

步骤3　建立KNN模型并训练模型。直接使用NearestNeighbors方法建立模型后使用fit方法做训练。

步骤4　找到新产品最近的K个相似产品。使用KNN模型的kneighbors方法获得指定数量的K个近邻。

如下是一段简单但包含了核心步骤的示例：

```
from sklearn.neighbors import NearestNeighbors # 导入NearestNeighbors库
X = [[0., 0.1, 0.6], [0., 1.5, 0.3], [1.2, 1.6, 0.5]] # 定义训练集，训练集包含3条记录，
    每个记录包含3个特征变量
neigh = NearestNeighbors(n_neighbors=1) # 建立非监督式的KNN模型对象
neigh.fit(X) # 训练模型对象
new_X = [[1., 1., 1.]] # 要预测的新产品数据
print(neigh.kneighbors(new_X)) # 打印输出新产品最相似的训练集产品
```

上述代码执行后返回如下信息：

```
(array([[ 0.80622577]]), array([[2]]))
```

其中第一个数字是与新产品数据最相似的产品的距离，第二个数字是对应最相似产品记录的索引值（注意索引值从0开始，2表达第三个）。

6.4.3 销售预测模型

销售预测模型根据历史的销售数据来预测未来可能产生的销售情况。该模型常用于促销活动前的费用申请、目标制定、活动策略等的辅助支持。

·销售预测模型通常要得到的结果为未来会产生多少销售量、收入、订单量等具体数值，可通过时间序列、回归和分类三种方法实现。

·基于时间序列做销售预测。使用时间序列做销售预测的方法常用于没有太多可用的自变量的场景下，只能基于历史的销售数据做预测性分析。有关时间序列的更多话题，具体请参照4.6节的内容。

·基于回归做销售预测。基于可控的特征变量建立回归模型来预测未来的销售情况是更常用的方法，有关回归模型的更多内容，请具体参照4.2节。

·基于分类做销售预测。分类方法是针对每个销售客户产生的是否购买的预测分类，然后再基于能产生购买的预测分类做客单价、订单量和收入的分析。这是一种对于具体数值的变通实现思路。有关分类分析的更多内容，具体参照4.3节。

6.4.4 商品关联销售模型

商品关联销售模型主要用来解决哪些商品可以一起售卖或不能一起打包组合的问题。关联销售是商品销售的常态，也是促进单次销售收入和拉升复购效果的有效手段。

商品关联销售模型的实现方式是关联类算法，包括Apriori、FP-Growth、PrefixSpan、SPADE、AprioriAll、AprioriSome等，主要实现的是基于一次订单内的交叉销售以及基于时间序列的关联销售。

关联销售算法的实现步骤上与普通的监督式和监督式算法略有不同，原因是关联分析对于数据集的要求不同。一般包括三种数据源格式：

·第一种是事务型交易数据，典型的数据格式是每个数据行以订单ID或客户ID作为关联分析的参照维度，如果同一个订单内有多少个商品，那么将会有多个数据行记录。如表6-1所示。

表6-1 事务型交易数据

订单 ID	购买商品
2501540	苹果
3265100	香蕉
3964910	牛奶
3964910	香蕉
1850010	苹果

·第二种是合并后的交易数据，数据格式是每个数据行以订单ID或客户ID作为分析的参照维度，如果同一个订单内有多个商品，那么多个商品会被合并到一条记录中。如表6-2所示。

表6-2 合并后的交易数据

订单 ID	购买商品
56241501540	苹果
12463265100	苹果，香蕉
34239564910	牛奶，香蕉
39612335210	香蕉
15865809010	苹果，草莓，香蕉

·第三种是真值表格数据，每个数据行是每个订单ID或客户ID，列是每个要关联项目的是否购买值，通常以T或F来表示。如表6-3所示。

表6-3　真值表格数据

订单 ID	苹果	香蕉	牛奶	草莓
56241501540	T	F	F	F
12463265100	T	T	F	F
34239564910	F	T	T	F
39612335210	F	T	F	F
15865809010	T	T	F	T

以上三种数据格式中，第一种和第二种常见于企业内部的源数据环境或数据仓库，第三种需要经过ETL处理得到，很多第三方工具也可以提供这种数据形式。如果企业内不具备能够直接做关联分析的数据，则需要做对应处理。有关关联分析的更过内容，具体参照“4.4关联分析”。

6.4.5 异常订单检测

异常订单检测用来识别在订单（尤其是促销活动中的订单）中的异常状态，目标找到非普通用户的订单记录，例如黄牛订单、恶意订单、商家刷单等。

黄牛订单会大量削减促销对普通用户的吸引程度，使得促销权益和利益被一小部分人获取，而非给到目标会员。

恶意订单则更加危险，很多竞争对手间会通常这种方式在促销活动中，将大量的商品库存通过订单的方式锁定，然后再活动结束后通过取消、退货等方式释放库存。这种方式将使促销活动由于无法真正卖出商品而无法实现促销的目的，同时还会消耗公司大量的人力、物力，是各个公司都非常反感的恶意竞争方式。

商家刷单是一种常见的用于提升商家排名的方式，通常由商家来安排内部或关联人员大量购买商品，以形成商家流量和销售提升的目的。

异常订单检测主要基于两类方法实现：

·一类是基于监督式的分类算法。将历史已经识别出来的真实异常订单数据通过分类模型（例如SVM、随机森林等）做训练，然后应用新数据做分类预测，看预测结果是否属于异常订单。

·一类是基于非监督式的算法。通过非监督式算法（例如OneClassSVM）基于历史的数据做训练，然后针对新的数据做判别，找到存在异常可能性标签的订单列表。

这两类方法是常见的算法应用，具体查看4.3节和4.5节的内容。

6.4.6 商品规划的最优组合

在做商品促销或广告宣传时，通常企业会面临多种组合策略，它是在一定限制条件下考虑通过何种组合策略来实现最大或最小目标。此时，可以考虑使用线性规划方法。

线性规划（Linear programming，LP）是运筹学中研究比较早、方式相对成熟且实用性非常强的研究领域，主要用来辅助人们进行科学管理，目标是合理地利用有限的人力、物力、财力等资源作出的最优决策。解决简单线性规划问题的最直接的方法是图解法，即借助直线与平面区域的交点求解直线在y轴上的截距的最大值或最小值。

在做线性规划时涉及几个概念：

·未知数：影响决策主要变量或因素。

·约束条件：解决线性规划问题时已知的并须遵守的前提条件。

·目标函数：用来表示未知数与目标变量关系的函数，线性规划中一般是线性函数。

·可行域：满足优化问题约束条件的解叫作可行解，由所有可行解组成的集合叫作可行域。

·最优解：满足目标函数最大化或最小化目标的最优的解。

实现线性规划的基本步骤如下：

步骤1 找到影响目标的主要因素，它们是规划中的未知数。

步骤2 基于未知数确定线性约束条件。

步骤3 由未知数和目标之间的关系确定目标函数。

步骤4 找到直角坐标系中的可行域。

步骤5 在可行域内求目标函数的最优解及最优值。

为了能清晰地表达上述概念和步骤，在此通过一个简单的示例演示该过程。假设公司有P1和P2两种商品，当推广P1商品时，每次费用为60元；当推广P2商品时，每次费用为30元。现在公司有1800元预算可以用来做P1和P2商品推广，其中受到两种商品尺寸和品类的限制，P1商品最多只能投放20次，P2商品最多只能投放40次，并且两种商品的总投放次数不超过45次。已知每次推广P1和P2的商品分别能获得单品毛利为40元和30元，问：如何安排P1和P2的商品投放次数才能达到销售毛利最大化目标？

为了解决问题，我们假设P1和P2两种商品的投放次数分别是X_1和X_2，最大化销售毛利为z，此时：

$$
\begin{cases}
\text{目标函数 } z = 40X_1 + 30X_2 \\
\text{约束条件} \begin{cases}
60X_1+30X_2 \leqslant 1800 \text{（总放到金额不超过 1800 元）} \\
X_1 \leqslant 20 \text{（P1 商品投放次数不超过 20 次）} \\
X_2 \leqslant 40 \text{（P2 商品投放不超过 40 次）} \\
X_1 > 0 \text{（投放次数不能小于 0）} \\
X_2 > 0 \text{（投放次数不能小于 0）} \\
X_1+X_2 \leqslant 45 \text{（P1 和 P2 的总投放次数不超过 45 次）}
\end{cases}
\end{cases}
$$

由于这是一个简单二维变量，因此可以先画出直角坐标图和可行域，然后基于目标函数找到最优解位置，如图6-1所示。

通过图可以发现最优解是目标函数与$X_1+X_2=45$和$60X_1+30X_2=1800$的交点，求解两个函数的解用到的是九年义务教育阶段基本数学知识。

$$
\begin{cases}
X_1+X_2=45 \text{（等式 1）} \\
60X_1+30X_2=1800 \text{（等式 2）}
\end{cases}
$$

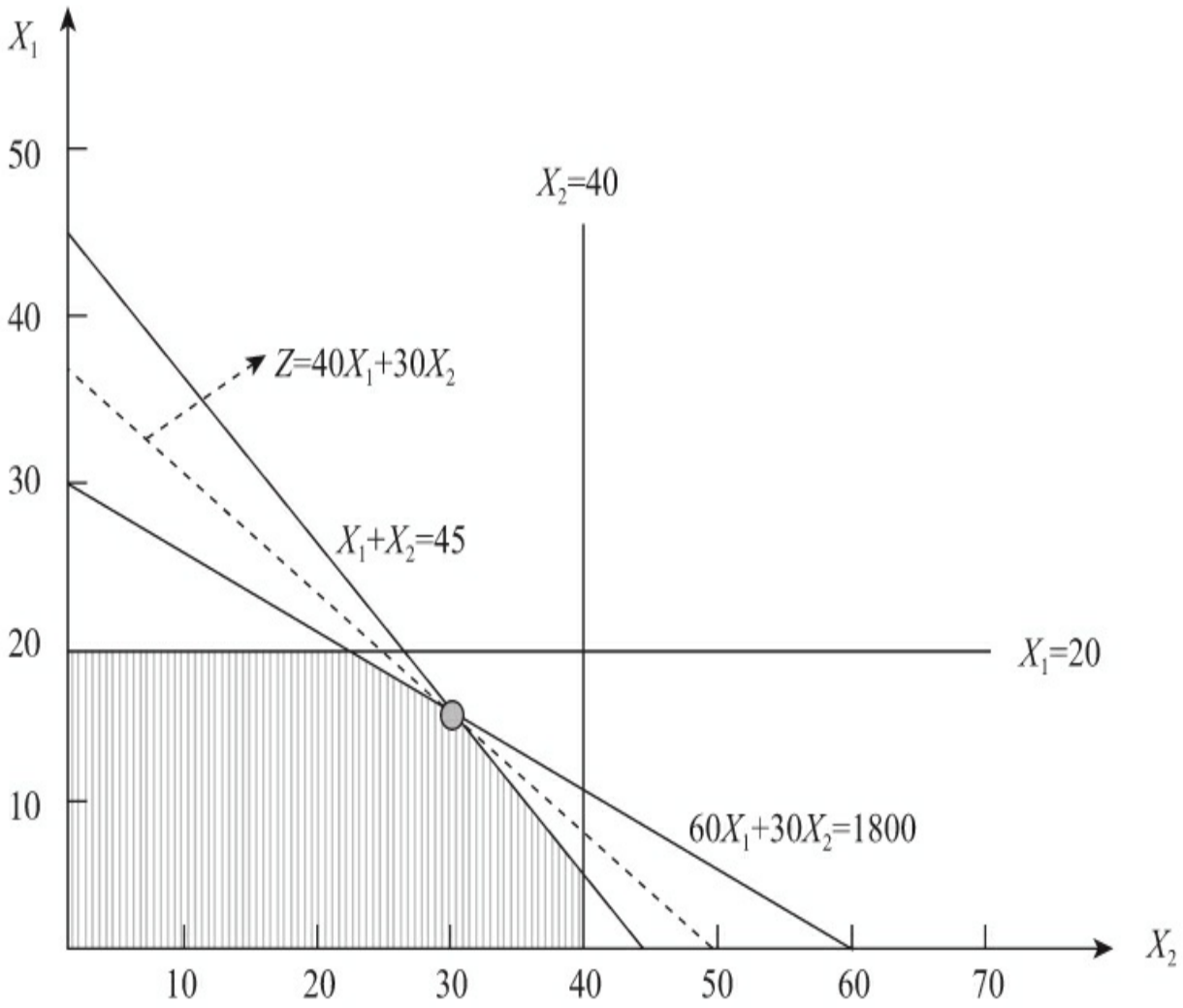


图6-1　线性规划可行域及最优解

步骤1　将等式1做转换：$X_1=45-X_2$

然后将转换后的X_1代入等式2，并依次求解：

步骤2　$60（45-X_2）+30X_2=1800$

步骤3　$2700-60X_2+30X_2=1800$

步骤4　$2700-30X_2=1800$

步骤5　$30X_2=900$

步骤6　$X_2=30$

步骤7　$X_1=45-30=15$

步骤8　然后将X_1和X_2代入目标函数：

$z=40X_1+30X_2=40\times15+30\times30=1500$

如果线性规划中有多个变量，那么我们无法通过图形的方式直接发现最优值的位置，此时可以借助Python的线性规划库来完成线性求解工作，包括scipy.optimize.linprog、pulp等。

6.5 商品数据化运营分析小技巧

本节中我们将介绍一些商品运营分析中的小技巧，包括层次分析法、假设检验、BCG矩阵分析分析法和4P分析等。

6.5.1 使用层次分析法将定量与定性分析结合

层次分析法（Analytic Hierarchy Process，AHP）是将与决策总目标相关的细分元素进行分解，在此基础上做层次权重的方法，这是一种将定性分析和定量分析相结合的方法。

这种方法可以广泛用于多种将定性与定量分析结合的商品运营决策场景中，例如：

·商品资源位的决策中，到底哪些商品应该放到最好的位置上？

·商品流量导航的入口，哪些品类应该放在最明显的位置？

·公司在天猫上争取到了几个资源位，应该投放哪些商品？

·商品贡献价值衡量，除了销量、利润等数字可衡量的因素以外，增加商品标杆、商品品牌价值的提升、企业战略合作商品、投资者关注品类等因素，如何选出贡献最大的商品？

层次分析法是不仅能用于不确定性和主观信息的整合，还可以将以往经验和洞察应用到权重评估过程中，这种基于层次划分的方法，可以使评估人员能有效衡量不同指标间的相对重要性。

层次分析法的完整步骤如下：

步骤1　构建层次分析结构，确定决策目标，然后对影响决策目标的因素归类并建立一个多层次结构。

步骤2　构造决策因素判断矩阵，比较同一层次中各因素关于上一层次的同一个因素的相对重要性，构造成对比较矩阵，该矩阵是一个正互反矩阵，然后做一致性检验。

步骤3　构造方案判断矩阵，然后一致性检验。

步骤4　计算决策因素权重，并结合对比方案矩阵计算各个方案的总得分并排序，得到决策方案。

使用APH做决策的核心是确定不同方案在不同因素上的加权得分总和，然后对综合做排序找到得分最大的方案。

为了能使该过程更通俗易懂，这里使用稍微简化一点的方法来说明AHP决策过程，忽略其中的一致性检验过程。假设现在有3个商品要做资源推广，现在考虑选择商品的因素包括：引入流量能力、转化率、毛利价值、品牌价值、标杆价值。其中前三个指标可以通过数字衡量的，后两个指标则是相对感性且无法具体量化的。

构建层次分析结构。分析结构分为3层，目标是选择最优商品，判断维度是引入流量能力、转化率、毛利价值、品牌价值、标杆价值，最终方案是从商品A、B、C中选择最优商品。

构造决策因素判断矩阵。通过两两对比做重要性评估时，一般采用1～9标度法，即最重要为9，最不重要为1/9，如果为1则说明同等重要。由表6-4的得分，看到左上到右下的对角线是1，沿对角线的值是对称互为倒数。

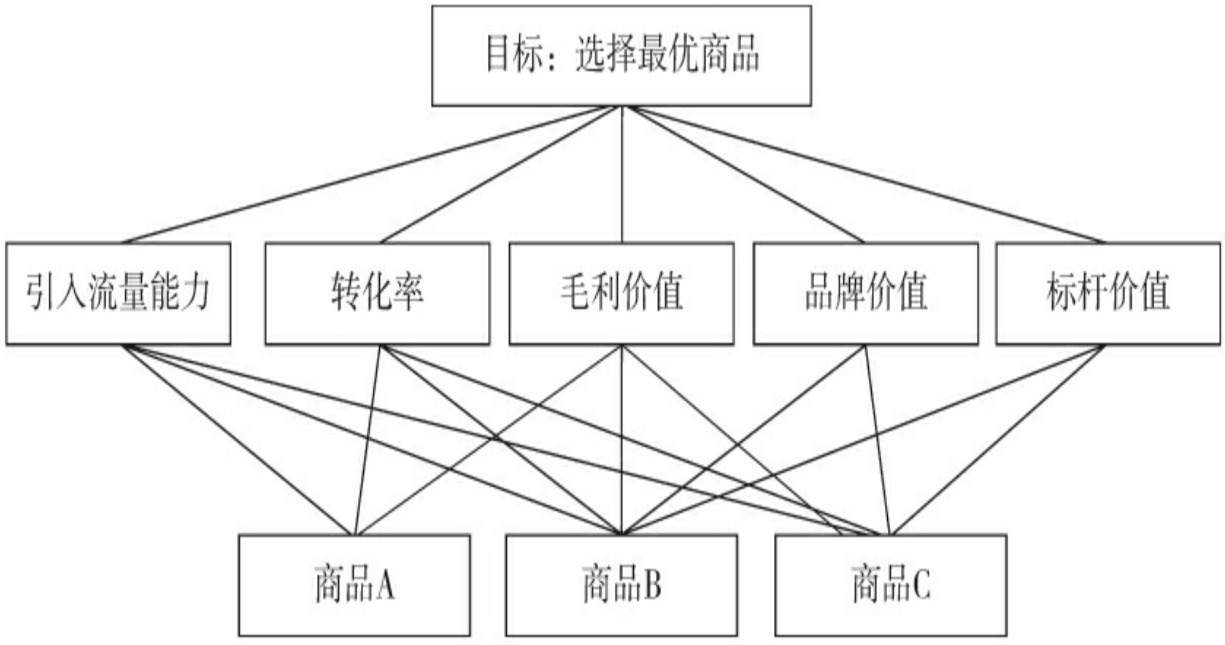


图6-2　决策层次分析结构

表6-4　决策因素重要性对比矩阵

	引入流量能力	转化率	毛利价值	品牌价值	标杆价值
引入流量能力	1	2	7	5	5
转化率	1/2	1	4	3	3
毛利价值	1/7	1/4	1	1/2	1/3
品牌价值	1/5	1/3	2	1	1
标杆价值	1/5	1/3	3	1	1

指标权重计算，这里采用规范列平均法（和法）。先对表6-4的每一列做标准化，然后将标准化后的值沿行求均值，得到每个决策因素的权重，标准化方法这里选择0-1标准化，最终得分如表6-5所示。

表6-5　决策因素标准化得分和权重矩阵

	引入流量能力	转化率	毛利价值	品牌价值	标杆价值	均值
引入流量能力	1.00	1.00	1.00	1.00	1.00	1.00
转化率	0.42	0.43	0.50	0.56	0.57	0.49
毛利价值	0.00	0.00	0.00	0.00	0.00	0.00
品牌价值	0.07	0.05	0.17	0.11	0.14	0.11
标杆价值	0.07	0.05	0.33	0.11	0.14	0.14

构造方案判断矩阵。对每个商品在不同决策因素下的相对重要性对比，形成如表6-6～表6-10所示的数据。

表6-6　不同商品引入流量能力对比矩阵

	商品 A	商品 B	商品 C
商品 A	1	1/3	1/8
商品 B	3	1	1/3
商品 C	8	3	1

表6-7　不同商品转化率对比矩阵

	商品 A	商品 B	商品 C
商品 A	1	2	5
商品 B	1/2	1	2
商品 C	1/5	1/2	1

表6-8　不不同商品毛利价值对比矩阵

	商品 A	商品 B	商品 C
商品 A	1	1	3
商品 B	1	1	3
商品 C	1/3	1/3	1

表6-9　不同商品品牌价值对比矩阵

	商品 A	商品 B	商品 C
商品 A	1	3	4
商品 B	1/3	1	1/3
商品 C	1/4	3	1

表6-10　不同商品标杆价值对比矩阵

	商品A	商品B	商品C
商品A	1	4	1/4
商品B	1/4	1	1/3
商品C	4	3	1

接着按照指标权重计算中的方法，计算每个方案在每个指标下的得分。具体过程省略，这里只给出具体结果，如表6-11所示。

表6-11　方案在不同决策因素下的得分矩阵

	引入流量能力	转化率	毛利价值	品牌价值	标杆价值
商品A	0.00	1.00	1.00	1.00	0.40
商品B	0.26	0.32	1.00	0.04	0.04
商品C	1.00	0.00	0.00	0.39	0.89

总排序权重计算及决策。现在我们已经有了三套方案（商品A、B、C）分别在不同因素下的得分以及各得分的权重，接下来要做的就是将表6-5中不同因素的权重均值与表6-11中不同商品在各个因素上的权重相乘并得到最终加权的总得分。过程如下：

商品A=1.00×0.00+0.49×1.00+0.00×1.00+0.11×1.00+0.14×0.40=0.656

商品B=1.00×0.26+0.49×0.32+0.00×1.00+0.11×0.04+0.14×0.04=0.4268

商品C=1.00×1.00+0.49×0.00+0.00×0.00+0.11×0.39+0.14×0.89=1.1675

由此，我们看到商品C的总得分是最高的，主要取胜的原因是其引入流量的能力明显。上述只是简单的过程，并没有做一致性检验，追求严谨性的读者可以将一致性检验加入到决策因素和方案对比矩阵中做一致性验证。

层次分析法非常好用且实用，但也存在一定缺点：

·不能使用太多的决策变量，否则计算起来会比较费时。

·决策变量间应该具有相对独立的特征，不能存在高度线性相关关系，否则得到的计算会出现偏差。

6.5.2 通过假设检验做促销拉动分析

在做运营效果评估时，往往第一步是判断运营效果是否显著，这一点对于促销活动尤其是对于销售的拉动分析尤为重要。一般情况下，企业做促销活动都会同时产生销售额提升的现象，但是这种现象是由于促销活动的拉动导致还是自然波动，需要通过数据来论证，不能盲目的看到销售提升就下结论是促销活动导致的。

在做这种结论定性时，可以通过多种假设检验的方式做显著性分析。

假设检验（Hypothesis Testing）是数理统计中常用的方法，它实现的是根据一定假设条件由样本推断总体。假设检验的基本实现思路是：根据研究课题的需要对研究总体作某种假设，记作H0；选取合适的统计样本，其选取结果要使得在假设H0成立时，其分布情况已知；然后由实测的样本，计算出统计量的值，并根据预先给定的显著性水平进行检验，作出拒绝或接受假设H0的判断。常用的假设检验方法有卡方检验、T检验法、F检验法、U检验法、Z检验法、方差分析法等。

·卡方检验属于非参数假设检验，适用于布尔型或二项分布数据，基于两个概率间的比较，早期用于生产企业的产品合格率等，在网站分析中可以用于目标转化率等有比率度量的比较分析。

·T检验属于参数假设检验，所以它适用的范围是数值型的数据。T检验的需要总体样本符合正态分布特征，主要用于样本含量较小（例如n＜30），总体标准差未知的数据。T检验广泛应用于医学统计等领域。

·F检验又叫方差齐性检验。在判断两种样本检验方差是否相等时，需要使用F检验做验证。因此，它是一种基础的检验方法。至于两组数据之间是否存在系统误差，则在进行F检验并确定它们的精密度没有显著性差异之后，再进行T检验。

·U检验是在大样本（n＞30）的情况下，检验随机变量的数学期望是否等于某一已知值的一种假设检验方法。U检验适用于样本量n较大且符合正态分布的情况。

·Z检验一般用于大样本（即样本容量大于30）平均值差异性检验的方法，比较两个平均数的差异是否显著。

·方差分析法常用来分析多个（2个以上）群体中的计量型数据，以便比较变异的意义和分析其来源。方差分析要求观察样本的分布符合正态分布或近似正态分布，并且各组数据之间的方差具有齐性。

在做假设检验时，经常用到的Python库是scipy中的stats，里面包含了各种参数和非参数检验方法以及用于评估数据分布模型的检验方法。

6.5.3 使用BCG矩阵做商品结构分析

BCG矩阵又称为波士顿矩阵，它是波士顿咨询集团提出的一种分析方法。该方法常用来分析如何将企业有限的资源有效地分配到合理的产品结构中去，以保证企业利益的最大化，因此这是一种宏观分析方法。

BCG矩阵认为决定商品结构的基本因素有两个：市场引力和企业实力。

·市场引力指的是企业销售量、增长率、目标市场容量、竞争对手强弱等市场因素，其中最能反映市场引力的指标是销售增长率，这是决定商品结构的外部因素。

·企业实力包括市场占有率，资金、技术等方面的能力，其中市场占有率是决定企业商品结构的内在因素。

基于销售增长率和市场占有率这两个维度，BCG矩阵将商品组合分为4种类型：

·明星型商品（Stars）：指高增长率，高市场份额的商品，这类商品可能成为企业的现金牛商品。

·问题型商品（Question Marks）：指高增长率，低市场份额的商品。这类商品虽然市场增长较快，但是一般利润率较低，所需资金支持和资源投入较大。

·现金牛型商品（Cash Cows）：指低增长率，高市场份额商品，这类商品能帮助企业回收资金并加速资金流转，可以支持其他商品和业务的发展，是公司发展的基石和后盾。

·瘦狗型商品（Dogs）：指低增长率，低市场份额的商品。这类商品对企业来讲一般处于亏损或利润贡献比较差的阶段，通常很难为企业带来收益。

在企业中应用这种分析方法做商品结构分析，基本步骤如下：

步骤1　确定企业销售增长率和市场占有率。关于销售增长率的计算可以用本年销售额/上年销售额-1，而市场占有率需要通过市场分析、第三方报告、行业协会等方式获取。

步骤2　绘制四象限图，确定企业不同商品的销售增长率和市场占有率均值，然后基于均值划分为4个象限。

步骤3　将企业内不同品类的名称按照实际销售增长率和市场占有率以散点的方法填到各个象限中。为了增加更多可用于分析的特征，还可以用散点的大小来表示销售额或利润额等其他主要贡献因素。此时可以得到如图6-3所示的矩阵信息。

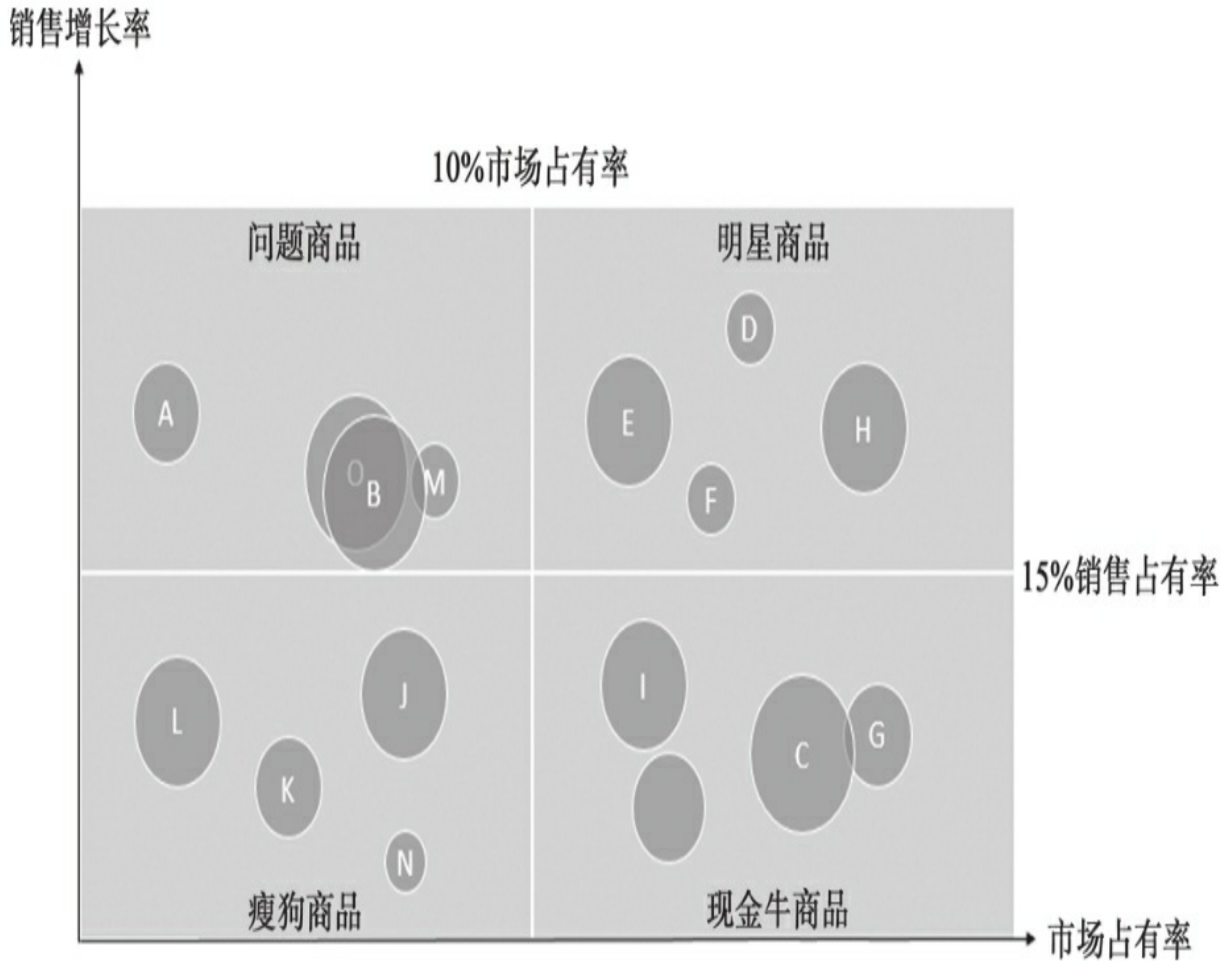


图6-3　BCG矩阵示例

如何对矩阵信息做分析？

·针对明星型商品品类，应当积极扩大经济规模和市场机会，以长远利益为目标，提高市场占有率，加强竞争地位，例如采用事业部形式直接管理。

·针对问题型商品品类，应当重点考虑市场占有的问题，包括市场营销、客户维系、新产品引入等方式来扩大市场面。

·针对现金牛型品类，需要重点维持现状，因此单独的事业部以及营销体系是重要措施。

·针对瘦狗型品类，需要考虑市场撤出、资源整合到其他商品业务或做结构化改革，以降低其对企业的负面效应。

BCG矩阵的优势是能够将不同的商品或业务放到一个平面做对比，并且基于公司的整体触发可以考虑资源间的最优配置和调整方向。在通过矩阵做输出表示时，其具有非常好的可理解性和应用性，是企业战略研究的重要方法支撑。

6.5.4　巧用4P分析建立完善的商品运营分析结构

4P是营销学的一种组合概念，包括产品（product）、价格（price）、渠道（place）、促销（promotion）。

·产品：包括有关商品的客观实体要素，例如质量、外观、样式、品牌、类别、属性、规格等。

·价格：有关商品的价值要素，例如基本价格、促销价格、付款价格、毛利、利润率等。

·渠道：有关商品的分销要素，包括分销渠道、供应商、物流、库存、经销商等。

·促销：有关商品的促销类要素，包括广告、营销、促销活动、宣传、推广等。

4P理论几乎涵盖了所有商品在市场营销中的关键可控要素，是企业内商品运营的“抓手”和落地点。实际上商品运营所做的所有内容都脱离不了4P理论的内容。

在做商品数据化运营分析过程中，如果在没有成型可用的思路之前，可以考虑从4P的角度去搭建商品运营分析的基本思路，这是非常有效的初始化打破僵局的方法。

当然，经典4P理论都是围绕商品环节的，在实际销售过程中除了商品本身的要素外，还有更多跟商品运营相关的其他要素，包括：

·人员：人员是做商品运营（也包括所有运营）的基石，任何运营活动的结果都跟人挂钩，因此可以从人员的角度分析对运营效果的影响。

·资源：资源的支持力度是导致商品运营结果差异的重要因素，不同的促销经费、资源位、站外广告位、活动时间、内部推介等资源支持下，运营效果差异很大。

·流程和机制：商品运营过程中的流程也是影响运营结果的主要因素，很多情况下甚至会成为关键因素，包括固定性流程（例如每月一次的××活动）、突发或异常应对机制（例如异常订单的隔离）、临时性流程（临时提报活动的影响）等。

活用4P理论能够在商品运营分析中，有效的整合运营的主要要素，并建立完善的分析结构。

跟4P概念相对应的还有4C，消费者（Consumer）、成本（Cost）、便利（Convenience）、沟通（Communication）。4C跟4P的最大不同是，4P是以商品为出发点考虑营销基本面，而4C则是以消费者为导向的组合方法。随着概念的发展，后来又延伸出4S、4R、4V、4I等不同营销理论。

6.6 商品数据化运营分析的“大实话”

本节的大实话，将解释几个常见的商品数据化运营中的潜在规律和知识，这些知识会增加读者对运营本质的理解。

6.6.1 为什么很多企业会以低于进价的价格大量销售商品

在商品销售中，我们经常会看到一些价格非常低，甚至还包邮或免运费的商品，并且这些商品的销量还非常大，例如5元全国包邮。如此低价的大量“倾销”商品，是否是一种“赔本赚吆喝”的买卖？

这种现象非常常见，但除了极少数情况外，这其实不是一种“赔本赚吆喝”的行为，这类行为的动机主要包括以下几种：

（1）扩大市场占有率

当企业想要扩大某类商品的市场占有率时，最佳途径是通过低价快速占领市场，等具备一定条件之后再恢复正常价格，通常这类条件包括：已经将竞争对手扼杀、已经建立绝对的市场地位之后、用户的消费习惯已经养成等。现在很多企业在市场营销策略上都会如此应用，例如滴滴打车最开始的补贴行为、运营商对新客户的巨额补贴、京东最开始的“多、快、好、省”等都是如此。当然，这类行为需要企业具备一定的资金规模，能够承受短期（或比竞争对手更长期）的亏损；或者可以从外部获得资金注入以满足市场发展需求。

（2）树立低价商品形象

很多时候商家会将爆款商品的价格压得很低，这类商品通常是销售领域中具有标杆性质的商品，例如苹果中的iPhone就是如此。由于消费者无法获知所有商品的价格，只能通过对某些具有标杆性质的商品价格来感知商家的平均价格水平，因此将这类标杆商品价格压低之后，消费者会由于标杆商品价格低而产生对商家所有商品都会比较低的认知，这是一种常见的销售策略。

（3）获得更有力的店铺排名

很多第三方经营平台的商品和店铺排名，都会受到商品销售量、点击量、浏览量、咨询量、评论量等诸多因素影响，而这些因素最基础的因素是“引流”，即首先要有人看到商品或店铺，然后才有可能形成浏览、转化、咨询等进一步动作。因此低价是吸引用户点击和浏览的最佳途径之一，商家自然希望通过低价来获得更多的用户流量，然后在做好

后端销售和服务的基础上来提升店铺排名，最终增加店铺销售。

（4）新产品推广

当企业推广新产品时会经常使用免费试用或低价销售的策略获得用户关注，这对于新领域内的商品尤为重要。免费或低价意味着用户无需付出任何商品购买成本或只需付出一点点成本便能获得商品体验，是一种比较好的推广方式。当然，商品低价甚至免费并不适合所有商品，因为除了商品购买成本外，消费者体验还会受到其他因素影响，例如商品应用场景、使用成本、使用习惯、使用周期等多个因素的影响，总体而言，商品使用的综合成本越低、所需时间越少、使用限制越少，越能获得更多的用户体验意愿。

（5）去库存

在6.2.3节我们提到了一些非常重要的供应链指标，包括库龄、滞销金额、残次数量等指标，这些指标都对应的运营场景一般都是商品无法或很难销售出去。对于这类包括过季、滞销、残次等因素导致的商品，企业一般会采用多种方式做促销，低价促销是其中的一种，低价相对于原始商品销售价格可能会更低，但却能将“静态”库存转换为“动态”现金流，这种价值不是低价所能一次性衡量的。举个例子，例如商品A积压在库存中的时间为1年，假设此时商品A以低于进货价100元的价格“亏损”卖出，那么商品回收的资金在接下来的1年内可能获得200年的净利润；如果商品A仍然不作低价销售，那么接下来的1年内商品本身的进货成本、库存成本、管理成本等综合成本仍然存在，甚至还会面临无法销售出去的问题，此时会导致更多的亏损。

（6）薄利多销

很多我们看到的“低价”商品，其实相对于其进货价来讲是不亏损的。以利润率非常高的家纺类产品为例，假设某品牌家纺的四件套日常销售价格为500元，即使以250元“对价”销售，该四件套也不会亏损，因为毛利润一般都远超过50%。很多行业的商品毛利润非常高，即使以低价销售，企业仍然有利可图。此外，企业中都存在着边际效应规律，当销售规模越来越大时，其单位成本会越来越低，这会表现在运营的各个方面，例如库存、送货、销售、促销等。当企业通过低价的方式将商品大量销售出去后，其边际成本会一般会比较低，单位商品的利润会通过

多销而补足甚至提升。

（7）组合销售策略

在商品结构中，并不是所有的商品都承担“毛利”任务。在6.5.3节中我们提到了四类商品，只有“现金牛”是主要负责产生毛利的，而其他的商品则可能分别承担了引流、梳理商品形象、扩大市场占有率、新模式探索、合作伙伴联盟共赢等多种角色，其中低价商品即使相比于批次进货价处于亏损状态，在整体销售组合策略的支持下，公司的整体利润仍然为正。

6.6.2　促销活动真的是在促进商品销售吗

以促销带销售是绝大多数企业商品运营的基本思路，并且很多消费者也被“惯坏了”，如果不是非常着急买，那就等企业做活动的时候再买。

图6-4显示了某企业常规性月度活动的订单量按日分布图（图中部分数据已经做处理）。

从图中可以看到在企业做促销活动的每月18日，订单量确实会有非常大的增长。但是，当看一整个月的订单分布时却明显发现，在18日前后的时间内的订单量都比月初要低不少。在18日之前的时间，由于企业在做促销活动时都有领券和活动宣传，经常性的告知用户优惠券只能在18日使用并且18日会有更多优惠活动，此时的消费能力被暂时性的“禁锢”；等到18日当天则全部释放出来，形成了当日的“巨大”增长；而到18日之后，由于消费需求大多都已经释放，之后的订单都属于常规性订单，因此订单比较少。

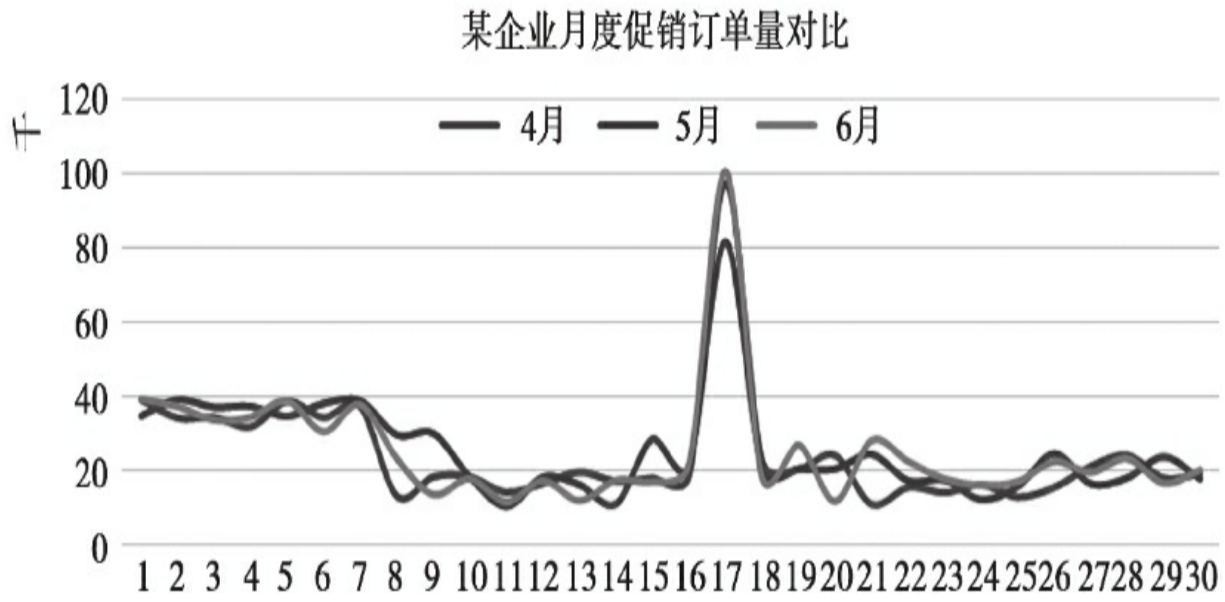


图6-4　某企业常规性月度活动订单量对比图

这种情况下，如果我们把18日的订单分散到每日，其实会发现做活动会比做做促销活动的效果好不了太多。既然如此，为什么企业会要做

促销活动，可能有以下因素影响：

·如果本企业不作促销活动，那么用户的消费需求会被竞争对手的促销活动吸引走，此时必然会降低本企业的订单和销量。

·很多用户已经养成了“促销订单”的习惯，要改变这种习惯只靠一个企业是不可能的。

·促销是增加用户活跃度和忠诚度的必要方式之一，没有促销活动作为载体，企业会减少与用户的沟通机会，久而久之会导致用户流失。

·企业市场声量的考虑，促销活动必然意味着市场宣传，这种宣传可以吸引用户关注、增加投资者的信心、提高促销时的商品销量.如果没有这类宣传，那么企业将慢慢在市场上失去关注度。

·销售方式单一的无奈选择，很多企业的运营方式单一，缺乏综合性、强黏性和组合性的运营策略，只能依靠促销来销售。

当然，除了这些基本运营层面的因素，很多行业巨擘也试图通过“大事件”来彰显企业对于社会的贡献力和影响力。通过类似于全国性的活动，企业能够“轻而易举”的获得社会总销售额超过1%的流水贡献，并能广泛影响商品销售产业链的生产、加工、制造、物流、库存、运输、分销等各个环节。这种“无声胜有声”的说服力，会极大增加企业在跟政府、资本市场、供应商、合作伙伴等的谈判地位和谈判能力。

6.6.3 用户关注的商品就是要买的商品吗

在做用户关注度分析时，我们可以通过用户浏览的信息分析出用户到底关注哪些商品。但是，如果通过关联分析来对用户的关注（或最经常浏览的商品）和购买商品做关联性分析（具体见4.4.2节），却会发现二者其实经常会不一致，而且结果通常是用户关注的商品会比购买的商品更“高端”，表现在价格更高、质量更号、品牌认知度更高等方面。

为什么很多用户关注的商品跟实际购买的商品有出入？这主要受到以下因素的影响：

（1）用户希望从更“高端”的商品上获得更多信息

一般情况下，更“高端”的商品在质量、性能、品质等方面会更胜一筹，用户可以从这类商品中发现一些更高价值的产品属性点，然后将这些产品属性点应用到实际要购买的商品上。例如用户可能只需要购买一款2000元左右的Android手机，但在做产品对比时用户需要了解哪些属性是某些商品区别于其他商品的显著性特征，除了查询相关知识外，常用的方法是查看更“高端”的商品会具有哪些特别属性，此时可能会看下华为P10、三星S8等。

（2）用户的心理预期是购买更好的商品

在购买商品时，用户往往会有一个目标购买商品的价值基准，以及围绕价值基准上下可波动的价值范围。如图6-5所示。

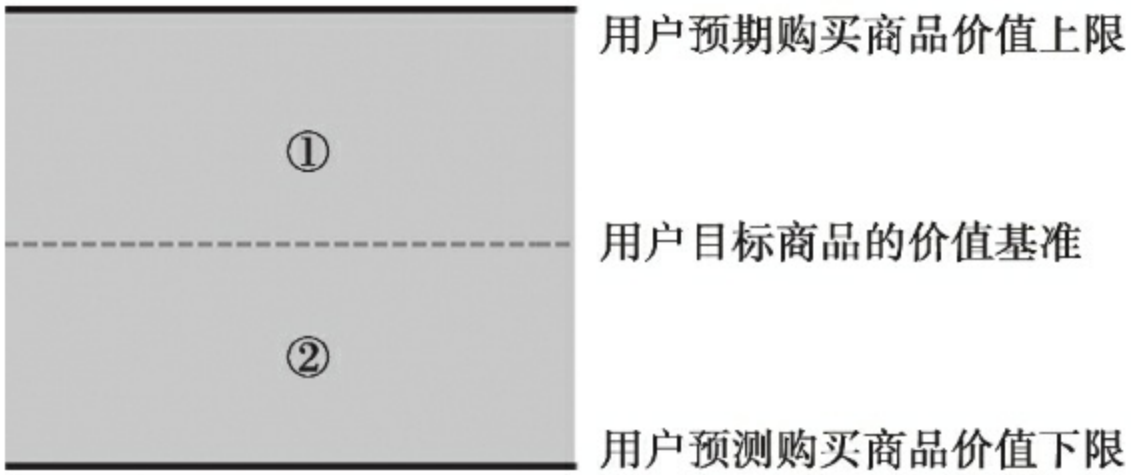


图6-5　用户预期商品价值

围绕目标商品的价值基准，用户的预期价值允许上下一定范围内的波动。通常，在具备对应购买能力的前提下，用户会倾向于购买更好的商品以获得更多的商品价值。因此，符合区间①内的商品便是用户经常性浏览的商品。

（3）用户的购买力不支持目标商品价值

在某些情况下，也会存在用户关注的商品价格高于实际购买力的情况，这种情况主要分为两种因素：

因素一　现阶段没有如此高的购买力，但是在未来可预期的时间内可以达到，因此提前做“功课”，等到具备购买力时“一举拿下”，这种针对二次购买需求的场景经常发生。

因素二　要购买的商品的价格突然高于自身购买力，或由于某些原因导致计划资金临时性占用而无法完成购买。例如商品临时调价、资金临时用作别的用途等。

6.6.4 提供的选择过多其实不利于商品销售

在互联网时代，信息的获取成本越来越低。消费者在购买商品时不再是信息匮乏，而是信息泛滥。海量信息在满足消费者个性化需求的同时，也会对用户销售转化产生负面影响。例如，当用户在要购买13.3英寸的MAC笔记本时，可能会面临以下情况：

·不同年份的机型，现在有16款还有17款的。

·不同的套餐，有官方套餐还有各种增加了不同配置、不同配件的套餐。

·不同的来源，有国行、港行以及各种来自其他平台的非正规渠道机器。

·不同的配置，包括内存、硬盘等规格不同。

·不同的卖家，有官方商城、个人店铺、第三方大卖场还有大型销售连锁商等。

面对如此多的选项，缺乏实际购买经验的消费者如果要仔细甄别需要花费大量的时间和精力做调查研究，对大部分用户来讲是不现实的，这会导致很多用户会直接放弃购买或者转而购买其他需要投入资源更少的商品上。

另外，提供的选择过多也可能会将用户关注的信息淹没，用户无法快速找到关注的商品信息，也会阻碍商品销售转化的产生。

这些问题对于具有海量商品信息的大型商场、在线商城等非常突出，个性化推荐的产生便可以解决这一问题。个性化推荐可以有效针对用户的潜在关注点针对性地给出用户可能关注或喜欢的商品，从而大大降低用户购买时的综合成本（包括时间和精力），从而提高商品转化。因此，从这个角度来讲，个性化推荐通过解决商品信息不对称的问题来达到销售提升的目的。

6.7 案例：基于超参数优化的Gradient Boosting的销售预测

6.7.1 案例背景

商品销售预测几乎是每个运营部门的必备数据支持项目，无论是大型促销活动还是单品营销都是如此。本案例就是针对某单品做的订单量预测应用。

本节案例的输入源数据products_sales.txt和源代码chapter6_code1.py位于“附件-chapter6”中，默认工作目录为“附件-chapter6”（如果不是，请cd切换到该目录下，否则会报“IOError:File products_sales.txt does not exist”）。程序的输出预测数据直接打印输出，没有写入文件。

6.7.2 案例主要应用技术

本案例用到的主要技术包括：

·基本预处理：包括缺失值填充、异常值处理以及特征变量的相关性校验。

·数据建模：超参数交叉检验和优化方法GridSearchCV、集成回归方法Gradient-BoostingRegressor。

·图形展示：使用Matplotlib做折线图展示。

主要用到的库包括：Numpy、Pandas、Sklearn、Matplotlib，其中Sklearn是数据建模的核心库。

本案例的技术应用重点是设置集成回归算法的不同参数值，通过GridSearchCV方法对每个参数值（或值对）做遍历，从指定的交叉检验得分中寻找最优值并得到最优值下的集成回归算法模型。这是一种更为自动化的参数优化方法。

6.7.3 案例数据

案例数据是某企业的运营活动数据，其中一些数据来源于每次销售系统，有些来源于每次运营系统，还有一些来源于手动运营记录。以下是数据概况：

·特征变量数：10。

·数据记录数：731。

·是否有NA值：有。

·是否有异常值：有。

以下是本数据集的10个特征变量，包括：

·limit_infor：是否有限购字样信息提示，1代表有，0代表没有。

·campaign_type：促销活动类型，分类型变量，值域为[0，6]代表7种不同类型的促销活动，例如单品活动、跨店铺活动、综合性活动、3C大品类活动等。

·campaign_level：促销活动重要性，分类型变量，值域为[0，1]，分别代表促销活动本身的不重要或重要性程度。

·product_level：产品重要性分级，分类型变量，值域为[1，3]，分别代表运营部门对于商品重要性的分级。

·resource_amount：促销资源位数量，整数型变量，代表每次该商品在参加促销活动时有多少个资源位入口。

·email_rate：发送电子邮件中包含该商品的比例，浮点型变量，值域[0，1]，值越大代表包含该商品的电子邮件越多。

·price：单品价格，整数型变量，代码商品在不同阶段的实际销售价格。

·discount_rate：折扣率，浮点型变量，值域[0，1]，值越大代表折扣力度越大。

·hour_resouces：在促销活动中展示的小时数，整数型变量，值越大代表展示的时间越长。

·campaign_fee：该单品的促销费用，整数型变量，值越大代表用于该单品的综合促销费用越高，这里面包含促销费用、广告费用、优惠券费用等综合摊派的费用。

目标变量：orders，代表该单品在每次活动中形成的订单量。

6.7.4 案例过程

步骤1 导入库。

```
import numpy as np  # 导入numpy库
import pandas as pd  # 导入pandas库
from sklearn.ensemble import GradientBoostingRegressor  # 集成方法回归库
from sklearn.model_selection import GridSearchCV  # 导入交叉检验库
import matplotlib.pyplot as plt  # 导入图形展示库
```

本案例主要用到了以下库：

·numpy：基本数据处理。

·pandas：数据读取、审查、异常值处理、缺失值处理、相关性分析等。

·GradientBoostingRegressor：集成方法GradientBoosting的回归库，本案例核心库之一。

·GridSearchCV：对超参数进行交叉检验优化的库。

·matplotlib：画折线图。

步骤2 读取数据，该步骤使用pandas的read_table方法读取txt文件，指定数据分隔符为逗号。

```
raw_data = pd.read_table('products_sales.txt', delimiter=',')
```

步骤3 数据审查和校验，该步骤包含了数据概览、类型分布、描述性统计值域分布、缺失值审查、相关性分析4个部分。

第一部分数据概览，目的是查看基本概况、分布规律等。

```
print ('{:*^60}'.format('Data overview:'))
print (raw_data.tail(2))  # 打印原始数据后2条
print ('{:*^60}'.format('Data dtypes:'))
print (raw_data.dtypes)  # 打印数据类型
```

```
print ('{:*^60}'.format('Data DESC:'))
print (raw_data.describe().round(1).T)  # 打印原始数据基本描述性信息
```

该部分的内容在之前很多章节都有介绍，重点介绍其中两个略有差异的应用点：

·raw_data.tail（）用来展示最后2条数据，跟raw_data.head（）方法相对应。

·使用round（）方法保留1位小数，这样展示的内容更加简洁。

·通过.T的方法做矩阵转置操作，该操作等同于transpose（），转置的目的也是展示更加简洁。

上述代码执行后输入如下结果：

基本数据审查结果，发现数据读取格式跟源数据相同。

```
**********Data overview:**********
     limit_infor  campaign_type  campaign_level  product_level  \
729            0              6               0              1
730            0              6               0              1
     resource_amount  email_rate  price  discount_rate  hour_resouces  \
729                8         0.8  150.0           0.87            987
730                9         0.8  149.0           0.84           1448
     campaign_fee  orders
729          2298    3285
730          3392    4840
```

数据类型结果，数据类型可以正确读取。

```
**********Data dtypes:**********
limit_infor          int64
campaign_type        int64
campaign_level       int64
product_level        int64
resource_amount      int64
email_rate         float64
price              float64
discount_rate      float64
hour_resouces        int64
campaign_fee         int64
orders               int64
dtype: object
```

数据描述性统计结果：

```
************************Data DESC:************************
                  count    mean     std    min     25%     50%     75%      m
limit_infor       731.0     0.0     0.4    0.0     0.0     0.0     0.0     10
campaign_type     731.0     3.0     2.0    0.0     1.0     3.0     5.0      6
campaign_level    731.0     0.7     0.5    0.0     0.0     1.0     1.0      1
product_level     731.0     1.4     0.5    1.0     1.0     1.0     2.0      3
resource_amount   731.0     5.0     1.8    1.0     3.0     5.0     7.0      9
email_rate        731.0     0.5     0.2    0.1     0.3     0.5     0.6      0
price             729.0   162.8    14.3  100.0   152.0   163.0   173.0    197
discount_rate     731.0     0.8     0.1    0.5     0.8     0.8     0.9      1
hour_resouces     731.0   848.2   686.6    2.0   315.5   713.0  1096.0   3410
campaign_fee      731.0  3696.4  1908.6   20.0  2497.0  3662.0  4795.5  33380
orders            731.0  4531.1  1932.5   22.0  3199.0  4563.0  6011.5   8714
```

在描述性统计结果中，以行为单位查看每个特征变量的基本信息，发现以下异常点：

·limit_infor的值域应该是0或1，但是最大值却出现了10。

·campaign_fee的标准差非常大，说明分布中存在异常点，并且最大值是33380，严重超过均值。

·price的记录数只有729，而其他特征变量的有效记录是731，说明该列有2个缺失值。

第二部分查看值域分布，目的是查看分类变量的各个变量的值域分布。在第一部分中发现了属于分类变量中出现了异常分布值，这里需要做下进一步验证。

```
col_names = ['limit_infor', 'campaign_type', 'campaign_level', 'product_leve
义要查看的列
for col_name in col_names:  # 循环读取每个列
    unque_value = np.sort(raw_data[col_name].unique())  # 获得列唯一值
    print ('{:*^50}'.format('{1} unique
values:{0}').format(unque_value, col_name))  # 打印输出
```

实现过程中先定义了要查看的分类变量，然后通过for循环读取每个特征名称，并使用dataframe的unique方法获取唯一值，然后使用Numpy的sort方法排序并打印输出。上述执行输入结果如下：

```
**************limit_infor unique values:[ 0  1 10]***************
**************campaign_type unique values:[0 1 2 3 4 5 6]***************
**************campaign_level unique values:[0 1]***************
**************product_level unique values:[1 2 3]***************
```

从结果看出，limit_infor确实存在值域分布问题，其他的分类变量则正常。

相关知识点：使用多种方法做数据框切片

Pandas提供了多种方法可以用来做数据框切片，本书的示例中用到的主要是基于指定列名的方法。下面以limit_infor为例说明选择不同列（特征）的具体用法：

1）通过列名方法选择。用法：raw_data['limit_infor']，这是最常用的选择方法，这种方法无需知晓目标列的具体索引值，只需知晓列名即可，可充分利用Pandas的良好交互优势。当然，也可以使用raw_data[[0]]来选取，其选择的结果是一个矩阵，而前者则是一个Series（列表）。

2）通过属性方法选择。用法：raw_data.limit_infor，该方法的效果跟raw_data['limit_infor']是等同的。

3）通过loc方法选择。用法：raw_data.loc[:，'limit_infor']。loc方法也是通过列名标签来获取列的常用方法，通过对行和列的限制可以选择特定数据块、片和单个数据体。例如通过raw_data.loc[1:10，'limit_infor']选择该列第2到第11个值。

4）通过iloc方法选择。用法：raw_data.iloc[:，0]。iloc方法也可以实现选择列，跟loc的用法类似，都可以做灵活的数据选取。但iloc的初衷是用于选择行数据，并且选择是通过列索引实现，而非标签实现。

5）通过ix方法选择。用法：raw_data.ix[:，'limit_infor']或raw_data.ix[:，0]。ix方法可以看做是loc和iloc的结合体，并且由于它结合了二者的用法，在做列选取时即可使用索引值，也可以使用列名标签。因此是实用性最广的灵活选择方法。

综上来看，在选择方法时，以下是主要区别点：

·如果单纯做全列选取，raw_data['limit_infor']和raw_data.limit_infor最常用。

·如果要做多条件的灵活选取，使用ix则更加适合，其次选择loc和iloc方法。

第三部分缺失值审查，目的是验证在第一部分中得到的关于price缺失的结论，同时再次确认下其他列的缺失情况。

```
na_cols = raw_data.isnull().any(axis=0)  # 查看每一列是否具有缺失值
print ('{:*^60}'.format('NA Cols:'))
print (na_cols)  # 查看具有缺失值的列
na_lines = raw_data.isnull().any(axis=1)  # 查看每一行是否具有缺失值
print ('Total number of NA lines is: {0}'.format(na_lines.sum()))  # 查看具有缺失值的行总记录数
```

实现过程跟之前的章节基本一致，主要方法用途如下：

·isnull（）：用于判断是否存在缺失值，存在则为True，否则为False。

·any（axis=0）：用于判断以列为单位的缺失值，如果一列中存在任何一个缺失值则为True。

·any（axis=1）：用于判断以行为单位的缺失值，如果一行中存在任何一个缺失值则为True

·sum（）：用于对指定对象求和，True作为1、False作为0参与计算。

上述代码执行后返回结果输出结果也验证了第一部分的结论（price有两条缺失数据），具体如下：

```
*************************NA Cols:*************************
limit_infor        False
campaign_type      False
campaign_level     False
product_level      False
resource_amount    False
email_rate         False
price               True
discount_rate      False
```

```
hour_resouces       False
campaign_fee        False
orders              False
dtype: bool
Total number of NA lines is: 2
```

第五部分相关性分析，主要用来做回归的共线性问题检验。如果存在比较严重的线性问题则需要使用特定算法或做降维处理。

```
print ('{:*^60}'.format('Correlation Analyze:'))
short_name = ['li', 'ct', 'cl', 'pl', 'ra', 'er', 'price', 'dr', 'hr', 'cf',
long_name = raw_data.columns
name_dict = dict(zip(long_name, short_name))
print (raw_data.corr().round(2).rename(index=name_dict, columns=name_dict))
出所有输入特征变量以及预测变量的相关性矩阵
print (name_dict)
```

该方法主要使用了pandas的corr（）方法做所有特征变量的相关性分析，使用round（2）保留2位小数。在输出时，由于原变量名称过长，因此这里使用rename方法对索引和列名做修改，先定义一个缩写列表，使用columns方法获取原始变量名，然后使用dict结合zip方法将两个列表组成字典。最后输出相关性矩阵和变量名对应信息如下：

```
********************Correlation Analyze:********************
          li    ct    cl    pl    ra    er  price    dr    hr    cf  orders
li      1.00 -0.03 -0.08 -0.04  0.05  0.04  -0.02  0.00  0.01 -0.04   -0.02
ct     -0.03  1.00  0.04  0.03  0.01 -0.01  -0.05 -0.01  0.06  0.06    0.06
cl     -0.08  0.04  1.00  0.06  0.05  0.05   0.02  0.02 -0.52  0.26    0.05
pl     -0.04  0.03  0.06  1.00 -0.12 -0.12   0.59 -0.04 -0.25 -0.23   -0.30
ra      0.05  0.01  0.05 -0.12  1.00  0.98   0.13  0.15  0.54  0.46    0.62
er      0.04 -0.01  0.05 -0.12  0.98  1.00   0.14  0.18  0.54  0.47    0.63
price  -0.02 -0.05  0.02  0.59  0.13  0.14   1.00  0.25 -0.08 -0.11   -0.10
dr      0.00 -0.01  0.02 -0.04  0.15  0.18   0.25  1.00  0.17  0.19    0.23
hr      0.01  0.06 -0.52 -0.25  0.54  0.54  -0.08  0.17  1.00  0.32    0.66
cf     -0.04  0.06  0.26 -0.23  0.46  0.47  -0.11  0.19  0.32  1.00    0.76
orders -0.02  0.06  0.05 -0.30  0.62  0.63  -0.10  0.23  0.66  0.76    1.00
{'campaign_type': 'ct', 'campaign_fee': 'cf', 'email_rate': 'er', 'hour_resc
```

从相关性矩阵中发现，er（'email_rate）和ra（resource_amount）具有非常高的相关性，相关系数达到0.98，通常意味着我们需要对此做处理，更多共线性的问题请参见3.7节。

步骤4　数据预处理，包括异常值处理、分割数据集两部分。

第一部分异常值处理。根据步骤3得到的结论，包括price中的缺失

值、limit_infor中值域为10的记录以及campaign_fee为33380的记录都要做针对性处理。

```
sales_data = raw_data.fillna(raw_data['price'].mean())  # 缺失值替换为均值
# sales_data = raw_data.drop('email_rate',axis=1) # 丢弃缺失值
sales_data = sales_data[sales_data['limit_infor'].isin((0, 1))]  # 只保留促销
值为0和1的记录
sales_data['campaign_fee'] = sales_data['campaign_fee'].replace(33380, sales
异常极大值替换为均值
print ('{:*^60}'.format('transformed data:'))
print (sales_data.describe().round(2).T.rename(index=name_dict))  # 打印处理
完成数据基本描述性信息
```

实现过程中，各个主要功能点如下：

·通过raw_data['price'].mean（）获得price列的均值，使用raw_data.fillna方法填充缺失值。

·对limit_infor中值域为10的记录做丢弃，使用isin（（0，1）方法只获取值为0或1的数据记录。

·campaign_fee值为33380的记录，使用replace方法将其替换为本列均值。

·打印输出时，使用describe（）获得矩阵描述性统计结果，使用round（2）保留两位小数，使用.T做矩阵转置，使用rename（index=name_dict）将索引设置为缩写值。

上述过程完成后，打印输出结果如下：

```
**********transformed data:**********
        count     mean      std     min      25%      50%      75%      max
li      730.0     0.03     0.17    0.00     0.00     0.00     0.00     1.00
ct      730.0     3.00     2.01    0.00     1.00     3.00     5.00     6.00
cl      730.0     0.68     0.47    0.00     0.00     1.00     1.00     1.00
pl      730.0     1.40     0.55    1.00     1.00     1.00     2.00     3.00
ra      730.0     4.95     1.84    1.00     3.00     5.00     7.00     9.00
er      730.0     0.47     0.16    0.08     0.34     0.48     0.61     0.84
price   730.0   162.82    14.26  100.00   152.00   163.00   173.00   197.00
dr      730.0     0.81     0.08    0.49     0.77     0.82     0.87     0.98
hr      730.0   848.51   687.03    2.00   315.25   717.00  1096.50  3410.00
cf       730.0  3655.61  1561.27   20.00  2495.00  3660.00  4783.25  6946.00
orders  730.0  4531.27  1933.85   22.00  3196.50  4566.00  6021.25  8714.00
```

第二部分分割数据集X和y：

```
X = sales_data.ix[:, :-1]  # 分割X
y = sales_data.ix[:, -1]  # 分割y
```

该部分使用前面介绍过的ix方法，对数据集做切分，形成输入自变量X和因变量y。上述过程中，我们提到了一个共线性的问题还没有解决，这个问题留在后面的模型优化里面具体解释。

步骤5　模型最优化参数训练及检验，这是本案例实现的核心功能，目标是通过交叉检验自动得到最优结果时的模型参数以及模型对象。包括模型最优化参数训练及检验、获取最佳训练模型两部分。

第一部分模型最优化参数训练及检验：

```
model_gbr = GradientBoostingRegressor()  # 建立GradientBoostingRegressor回归对象
parameters = {'loss': ['ls', 'lad', 'huber', 'quantile'],
              'min_samples_leaf': [1, 2, 3, 4, 5],
              'alpha': [0.1, 0.3, 0.6, 0.9]}  # 定义要优化的参数信息
model_gs = GridSearchCV(estimator=model_gbr, param_grid=parameters, cv=5)  # 立交叉检验模型对象
model_gs.fit(X, y)  # 训练交叉检验模型
print ('Best score is:', model_gs.best_score_)  # 获得交叉检验模型得出的最优得分
print ('Best parameter is:', model_gs.best_params_)  # 获得交叉检验模型得出的最优参数
```

实现过程中的主要功能点如下：

先建立GradientBoostingRegressor模型对象，然后设置交叉检验时用到的模型可选参数，也就是GradientBoostingRegressor中的的调节参数，这里用到了三个参数：

·loss：loos是GradientBoostingRegressor的损失函数，主要包括四种ls、lad、huber、quantile。ls（Least　squares），默认方法，是基于最小二乘法方法的基本方法，也是普通线性回归的基本方法；lad（Least　absolute　deviation）是用于回归的鲁棒损失函数，它可以降低异常值和数据噪音对回归模型的影响；huber是一个结合ls和lad的损失函数，它使用alpha来控制对异常值的灵敏度；quantile是分位数回归的损失函数，使用alpha来指定分位数用于预测间隔。

·min_samples_leaf：作为叶子节点的最小样本数。如果设置为数字，那么将指定对应数量的样本，如果设置为浮点数，则指定为总样本量的百分比。

·alpha：用于huber或quantile的调节参数。

这里设置了loss的所有可用损失函数，以字符串列表的形式定义；min_samples_leaf则是具体值作为定义，而非样本比例；alpha定义了常用的覆盖了0到1之间的间隔值。这里定义为列表或元组都可以，只要是能实现迭代读取即可。

再使用GridSearchCV建立交叉检验模型对象，设置参数如下：

·estimator：用来设置要训练的模型器，本示例中是集成回归对象model_gbr。

·param_grid：用来设置要检验的所有参数列表值，本示例定义的值放在parameters中。该值是一个字典或者字典的列表。

·cv：用来定义交叉检验的方法。如果为空，则使用默认的3折交叉检验；如果为数字，则定义为交叉检验的次数，这里自定义为5可以获得更准确的结论；如果设置为一个对象，那么这是一个自定义的交叉检验方法；如果设置为可迭代的训练集和测试集对象，那么将循环读取分割好的数据集做交叉检验。我们在第五章的第二个案例中，就是用了自定义的交叉检验方法。

除此以外，GridSearchCV还提供了大量可供设置的参数，例如自定义交叉检验得分标准，可以通过scoring来定义，关于自定义交叉检验得分标准的内容，请参见第五章第二个案例中的get_best_model部分。

交叉检验模型对象创建完成之后，使用fit方法做训练，然后分别通过训练后的对象的best_score_和best_params_属性获取最佳结果的得分以及最佳参数组合。返回结果如下：

```
('Best score is:', 0.93144793519798985)
('Best parameter is:', {'alpha': 0.9, 'min_samples_leaf': 3, 'loss': 'huber'
```

程序通过交叉检验验证了所有可用的组合之后，得到了最优的得分以及参数值的组合。如果读者想要了解每次交叉检验的详细结果，可以使用model_gs.cv_results_获取，该方法可以获取所有组合下的运行时间、得分、排名等具体信息。

第二部分获取最佳训练模型，我们已经知道了最佳模型的参数，下面通过设置最佳模型做数据集训练。

```
model_best = model_gs.best_estimator_  # 获得交叉检验模型得出的最优模型对象
model_best.fit(X, y)  # 训练最优模型
plt.style.use("ggplot")  # 应用ggplot自带样式库
plt.figure()  # 建立画布对象
plt.plot(np.arange(X.shape[0]), y, label='true y')  # 画出原始变量的曲线
plt.plot(np.arange(X.shape[0]), model_best.predict(X), label='predicted y')
出预测变量曲线
plt.legend(loc=0)  # 设置图例位置
plt.show()  # 展示图像
```

我们并不打算“手动”将参数填充到模型对象中，而是直接通过交叉检验的训练对象的属性直接获取。使用model_gs.best_estimator_获得交叉检验模型得出的最优模型并建立对象model_best，对模型通过fit方法做训练。

为了验证效果，我们通过折线图来展示预测值与实际值的拟合程度：先使用Matplotlib自带的ggplot样式库，这样免去自己设置图形信息的步骤；然后建立画布并分别画出原始变量曲线和预测变量曲线，设置图例位置为自动调整之后展示图像，这些步骤之前都多次提到，图形结果如图6-6所示。

从图形输出结果看，预测值与真实值的拟合程度比较高。

步骤6　新数据集预测

业务方给出一个新的数据标准，即limit_infor=1，campaign_type=1，campaign_level=0，product_level=1，resource_amount=15，email_rate=0.5，price=177，discount_rate=0.66，hour_resouces=101，campaign_fee=798，针对该条数据做预测。

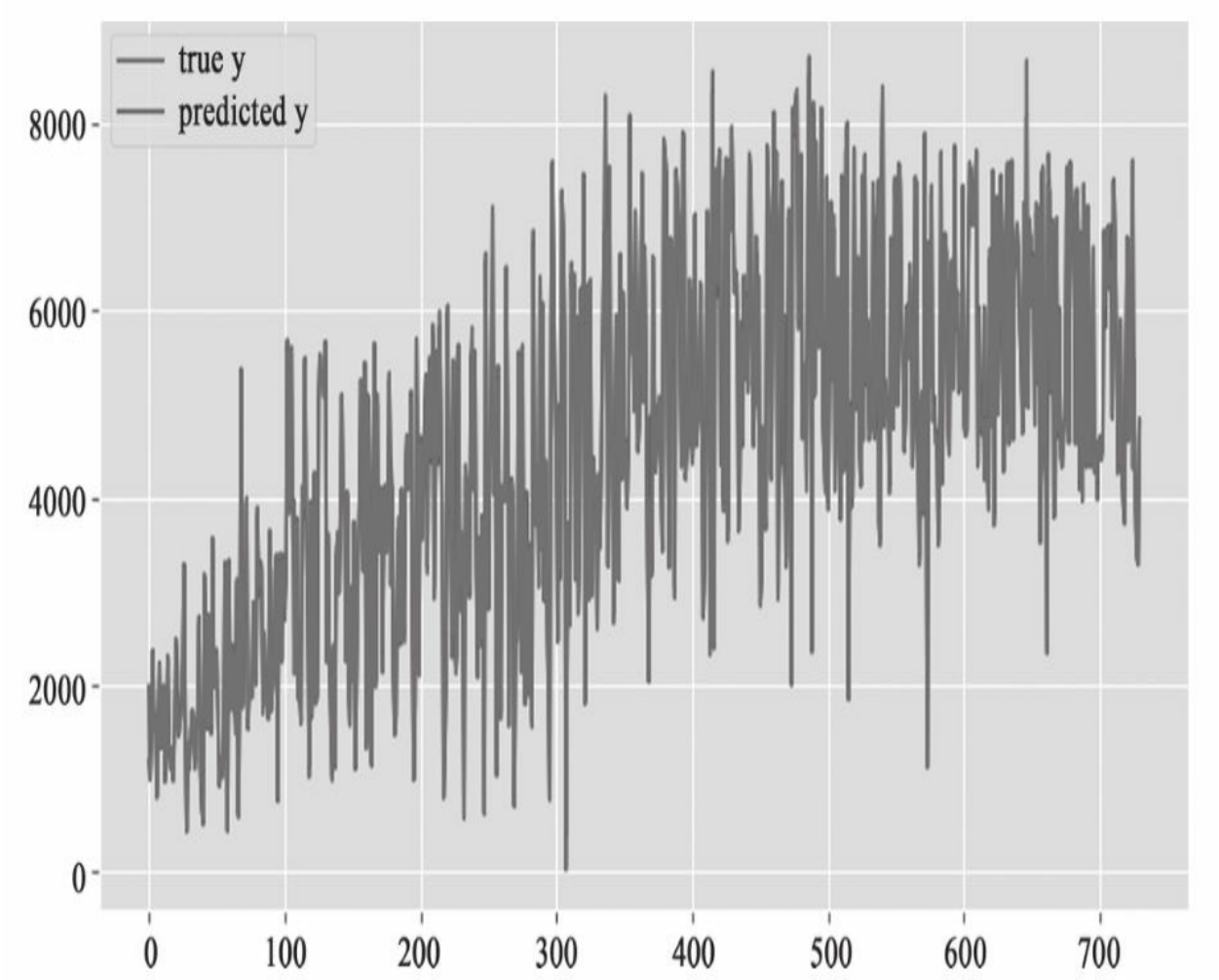


图6-6　预测值与真实值对比

```
New_X = np.array([[1, 1, 0, 1, 15, 0.5, 177, 0.66, 101, 798]])  # 要预测的新数据记录
print ('{:*^60}'.format('Predicted orders:'))
print (model_best.predict(New_X).round(0))  # 打印输出预测值
```

该数据定义为数组New_X，然后将该数据使用model_best（最佳模型对象）的predict方法做预测，结果不保留小数位数。结果如下：

```
********************Predicted orders:**********************
[ 779.]
```

由此得到预测的订单量为779。

将浮点型数据保存为浮点型，可以使用round（）方法，也可以使用int方法。以本次预测应用为例，model_best.predict（New_X）输出的原始结果为778.51283186。如果使用round（0）则得到四舍五入的结果779，如果实现int（model_best.predict（New_X））则输出结果为778，说明int方法是对数据做截取，只保留整数部分。二者对于预测结果的影响不大，但是对于过程中的预处理则会产生非常大的影响。

6.7.5 案例数据结论

本案例是一个应用型的模型示例，通过业务方给出的训练集以及预测自变量得到最终订单量预测值。

从交叉检验得到的最佳模型的得分为0.93，该得分由GradientBoostingRegressor的score产生，其值是预测的决定系数R2，该得分越高意味着自变量对因变量的解释能力越强，最高得分为1，5次交叉检验0.93的得分已经能够充分说明回归模型的预测能力比较强且效果相对稳定。

6.7.6 案例应用和部署

该模型给到业务部门后，业务部门针对该单品策划了一个跨品类活动，然后分别针对输入变量值落实运营要素。

对于本模型来讲，由于很多数据都是业务人工收集，难免这个过程会出现一些数据异常，而这些数据异常大多难以预料。例如：resource_amount和hour_resouces需要根据运营系统的记录做手动汇总整理，campaign_fee需要跟多个部门做沟通然后汇总多方来源。另外，也由于该模型每次运行的时间不长，并且总数据量也不大，因此每次人工运行以便应对临时性、突发性的数据问题。

6.7.7 案例注意点

本案例的应用核心是通过自动化优化方法，从众多指定的参数集合中通过交叉检验得到最优模型以及参数组合。该过程中有以下需要重点注意的内容：

·本案例中的异常值比较多，异常值的处理必不可少，即使GradientBoosting-Regressor的损失函数对异常值不敏感，通常也建议读者做预先处理。

·虽然本案例是一个自动优化的实现思路，但参数值域需要分析师人工指定，对于有固定值域的参数（例如loss）而言，这不会有什么影响，最多使用所有值域做训练优化；但对于有开放性值的参数而言，仍然是一个多次尝试的过程。例如min_samples_leaf，笔者分别尝试指定比例（浮点型）和整数型才找到相对较好的测试方案。

6.7.8 案例引申思考

在这个案例中，我们可以思考两个问题：

问题1：

本案例中，不同的自变量之间其实存在量纲的差异，针对这种情况是否需要先对每一列做标准化然后再做回归分析？

问题2：

本案例没有对具有高相关性的变量做任何处理，这是否得当？

笔者对这两个问题的见解如下：

问题1：

对于回归分析而言，是否做标准化取决于具体场景。在本案例中，回归分析的目的是做预测，因此无需做标准化；如果要做特征重要性的分析，就必须要做标准化。标准化的目的是去除量纲对于回归系数的影响，如果不做标准化，本案例中的因变量将主要受campaign_fee的影响（该变量系数的绝对值最大）。

问题2：

本案例应用的是集成方法，具体参数通过交叉检验得到的是使用huber损失函数做回归评估，它已经能够兼顾（一定程度上解决共线性）的问题。因为在应用GradientBoosting-Regressor方法中，有一个参数learning_rate是用来通过正则化的方法控制梯度下降过程的步长，它通过收缩正常化（learning_rate<1.0）来提高模型性能。可以在代码文件中取消65行的注释，该行代码会将email_rate列去掉，然后再做模型最优化评估，会发现得分结果仍然是0.93（小数点后面几位会有差异，但已经非常不明显）。

6.8 案例：基于LogisticRegression、RandomForest、Bagging概率投票组合模型的异常检测

6.8.1 案例背景

异常检测在之前介绍过，可以通过监督式和非监督式两类算法实现。本案例是使用监督式算法中的分类算法实现的异常检测应用。

本节案例的输入源数据new_abnormal_orders.csv、abnormal_orders.txt和源代码chapter6_code2.py位于“附件-chapter6”中，默认工作目录为“附件-chapter6”（如果不是，请cd切换到该目录下，否则会报"IOError:File new_abnormal_orders.csv does not exist"错误）。程序的输出预测数据直接打印输出，没有写入文件。

6.8.2 案例主要应用技术

本案例用到的主要技术包括：

·基本预处理：使用DictVectorizer将字符串分类变量转换为数值型变量、使用SMOTE对不均衡样本做过抽样处理。

·数据建模：基于cross_val_score的交叉检验，基于LogisticRegression、RandomForest、Bagging概率投票组合模型做分类。

主要用到的库包括：Numpy、Pandas、Sklearn、Imblearn，其中Sklearn是数据建模的核心库。

本案例的技术应用重点有两部分：

一部分是将原始数据集中的字符串分类转换为数值分类，便于参与建模运算；

一部分是通过概率投票方法，基于LogisticRegression、RandomForest、Bagging三个分类器建立一个组合分类器，用于实现组合投票和分类预测。

6.8.3 案例数据

案例数据是某企业的部分订单，以下是数据概况：

·特征变量数：13。

·数据记录数：134190。

·是否有NA值：有。

·是否有异常值：有。

以下是本数据集的13个特征变量的详细说明：

·order_id：订单ID，数字组合而成，例如4283851335。

·order_date：订单日期，格式为YYYY-MM-DD，例如2013-10-17。

·order_time：订单日期，格式为HH:MM:SS，例如12:54:44。

·cat：商品一级类别，字符串型，包含中文和英文。

·attribution：商品所属的渠道来源，字符串型，包含中文和英文。

·pro_id：商品ID，数字组合而成。

·pro_brand：商品品牌，字符串型，包含中文和英文。

·total_money：商品销售金额，浮点型。

·total_quantity：商品销售数量，整数型。

·order_source：订单来源，从哪个渠道形成的销售，字符串型，包含中文和英文。

·pay_type：支付类型，字符串型，包含中文和英文。

·use_id：用户ID，由数字和字母等组成的字符串。

·city：用户下订单时的城市，字符串型，中文。

目标变量：abnormal_label，代表该订单记录是否是异常订单。

6.8.4 案例过程

步骤1 导入库。

```
import sys
reload(sys)
sys.setdefaultencoding('utf-8')
import numpy as np  # numpy库
import pandas as pd  # pandas库
from sklearn.feature_extraction import DictVectorizer  # 数值分类转整数分类库
from imblearn.over_sampling import SMOTE  # 过抽样处理库SMOTE
from sklearn.model_selection import StratifiedKFold, cross_val_score  # 导入
交叉检验算法
from sklearn.linear_model import LogisticRegression  # 导入逻辑回归库
from sklearn.ensemble import VotingClassifier, RandomForestClassifier, Baggi
Classifier  # 三种集成分类库和投票方法库
```

本案例主要用到了以下库：

·sys：导入sys并设置字符编码为utf-8，目的是显示中文不乱码。

·Numpy：基本数据处理。

·Pandas：数据读取和审查，异常值处理，缺失值审查和处理等。

·DictVectorizer：用于将定义好的字符串值和数值键值对做转换。

·SMOTE：用于处理样本不均衡的过抽样处理库。

·StratifiedKFold：适用于有标签数据集的交叉检验数据集划分方法。

·cross_val_score：通过交叉检验方法做模型效果评估。

·LogisticRegression、RandomForestClassifier、BaggingClassifier：逻辑回归、随机森林、Bagging方法的分类库。

·VotingClassifier：用于分类的投票组合模型方法库。

步骤2 数据审查预处理函数。

基本状态查看，查看数据集后2条数据、数据类型、描述性统计。

```
def set_summary(df):
    '''
查看数据集后2条数据、数据类型、描述性统计
    :param df: 数据框
    :return: 无
    '''
    print ('{:*^60}'.format('Data overview:'))
    print (df.tail(2))  # 打印原始数据后2条
    print ('{:*^60}'.format('Data dtypes:'))
    print (df.dtypes)  # 打印数据类型
    print ('{:*^60}'.format('Data DESC:'))
    print (df.describe().round(2).T)  # 打印原始数据基本描述性信息
```

本函数主要功能定义如下：

·使用tail（2）方法预览数据集最后2条数据，与head（2）方法相对应。

·使用dtypes打印输出所有列数据类型

·通过describe（）.round（2）.T做描述性统计，并保留2位小数，最后做数据转置（目的是便于查看）。

缺失值审查，查看数据集的缺失数据列、行记录数

```
def na_summary(df):
    '''
查看数据集的缺失数据列、行记录数
    :param df: 数据框
    :return: 无
    '''
    na_cols = df.isnull().any(axis=0)  # 查看每一列是否具有缺失值
    print ('{:*^60}'.format('NA Cols:'))
    print (na_cols)  # 查看具有缺失值的列
    na_lines = df.isnull().any(axis=1)  # 查看每一行是否具有缺失值
    print ('Total number of NA lines is: {0}'.format(na_lines.sum()))  # 查看具有缺失值的行总记录数
```

本函数主要功能定义如下：

·isnull（）.any（）查看指定轴是否具有缺失值，axis=0指定沿列查看，axis=1指定按行查看。

·sum（）对数据中所有为True的数据求和。

类样本均衡审查，查看每个类的样本量分布

```
def label_samples_summary(df):
    '''
查看每个类的样本量分布
    :param df: 数据框
    :return: 无
    '''
    print ('{:*^60}'.format('Labesl samples count:'))
    print (df.ix[:, 1].groupby(df.ix[:, -1]).count())
```

本函数主要功能定义如下：

·df.ix[:，1].groupby（df.ix[:，-1]）.count（）使用.ix方法指定以最后一列为维度，对第一列做计数统计。

字符串分类转整数分类，用于将分类变量中的字符串转换为数值索引分类，这是本书新提到的用于预处理的函数。在很多情况下，数据库中存储的是字符串型的数据（例如性别是M/F），为了能够进行矩阵计算，需要把这些字符串型的分类变量转换为数值型分类变量（例如将M/F转换为0/1）。

```
def str2int(set, convert_object, unique_object, training=True):
    '''
用于将分类变量中的字符串转换为数值索引分类
    :param set: 数据集
     :param convert_object:  DictVectorizer转换对象，当training为True时为空；当training为False时则使用从训练阶段得到的对象
     :param unique_object: 唯一值列表，当training为True时为空；当training为False时则使用从训练阶段得到的唯一值列表
    :param training: 是否为训练阶段
     :return: 训练阶段返回model_dvtransform,unique_list,traing_part_data；预测应用阶段返回predict_part_data
    '''
    convert_cols = ['cat', 'attribution', 'pro_id', 'pro_brand', 'order_sour义要转换的列
    final_convert_matrix = set[convert_cols]  # 获得要转换的数据集合
    lines = set.shape[0]  # 获得总记录数
    dict_list = []  # 总空列表，用于存放字符串与对应索引组成的字典
    if training == True:  # 如果是训练阶段
        unique_list = []  # 总唯一值列表，用于存储每个列的唯一值列表
        for col_name in convert_cols:  # 循环读取每个列名
            cols_unqiue_value = set[col_name].unique().tolist()  # 获取列的唯一值列表
            unique_list.append(cols_unqiue_value)  # 将唯一值列表追加到总列表
        for line_index in range(lines):  # 读取每行索引
            each_record = final_convert_matrix.iloc[line_index]  # 获得每行数
```

```
据，是一个Series
                for each_index, each_data in enumerate(each_record):  # 读取
Series每行对应的索引值
                    list_value = unique_list[each_index]  # 读取该行索引对应到总唯
一值列表列索引下的数据(其实相当于原来的列转置成了行，目的是查找唯一值在列表中的位置)
                    each_record[each_index] = list_value.index(each_data)  # 获
得每个值对应到总唯一值列表中的索引
                each_dict = dict(zip(convert_cols, each_record))  # 将每个值和对应
的索引组合成字典
                dict_list.append(each_dict)  # 将字典追加到总列表
            model_dvtransform = DictVectorizer(sparse=False, dtype=np.int64)  #
立转换模型对象
            model_dvtransform.fit(dict_list)  # 应用分类转换训练
            traing_part_data = model_dvtransform.transform(dict_list)  # 转换训练
集
            return model_dvtransform, unique_list, traing_part_data
        else:  # 如果是预测阶段
            for line_index in range(lines):  # 读取每行索引
                each_record = final_convert_matrix.iloc[line_index]  # 获得每行数
据，是一个Series
                for each_index, each_data in enumerate(each_record):  # 读取
Series每行对应的索引值
                    list_value = unique_object[each_index]  # 读取该行索引对应到总
唯一值列表列索引下的数据(其实相当于原来的列转置成了行，目的是查找唯一值在列表中的位置)
                    each_record[each_index] = list_value.index(each_data)  # 获
得每个值对应到总唯一值列表中的索引
                each_dict = dict(zip(convert_cols, each_record))  # 将每个值和对应
的索引组合成字典
                dict_list.append(each_dict)  # 将字典追加到总列表
            predict_part_data = convert_object.transform(dict_list)  # 转换预测集
            return predict_part_data
```

本函数的功能定义略显复杂，主要包括以下过程：

首先定义要转换的数据，通过convert_cols定义转换的列名，并在此基础上新建转换数据集final_convert_matrix。

接下来获取总记录数，便于按行做循环。建立一个空列表用于存储字符串与对应索引的字典。由于我们要使用sklearn的DictVectorizer方法做转换，它要求转换对象是一个由字典组成的列表，字典的键值对是字符串及其对应的数字映射，我们的目的是从唯一字符集中取出其value和对应的index作为该键值对的组合。

通过training做条件判断，然后分别应用不同计算逻辑。训练阶段比预测集的应用阶段主要多了两部分内容：唯一值的列表，训练好的DictVectorizer对象。

当处于训练阶段时（training==True），先建立unique_list，用于存储每个列的唯一值列表。

再使用for循环读取每个列名，单独获取该列数据并使用unique方法获取唯一值，由于该唯一值的结果是一个数组，因此需要使用tolist方法转换为列表，便于后期应用。最后将每个列的唯一值列表追加到总的唯一值列表中。

获得所有列的唯一值组合后，下面开始将每条记录的具体值与其在唯一值列表中的索引做映射。例如唯一值结果是['1'，'2'，'3'，'4'，'5']，如果该列中有一个值为'4'，那么需要将该字符串转换为其对应的索引值3（索引从0开始）。

由于转换是按行实现的，因此使用for循环读取每行索引，然后使用iloc方法获取对应行数据，该数据是一个Series格式的列表；下面要做的是将每个列表中的value映射到唯一值列表中的index。

使用一个for循环结合enumerate读取列表的每个值及对应索引，索引结合unique_list[each_index]用于从唯一值总列表中找到原始所处的列的唯一列表，值用于从unique_list[each_index]中匹配出值对应的索引，使用的是列表的index（value）方法，得到的索引再替换掉原始字符串数据。该子循环结束后，每条记录已经是转换为数值型分类的列表，使用dict结合zip方法将其与列名转换为字典。

上述循环完成后，我们已经在dict_list中存储了所有的数据记录，每天的数据记录以一个字典的形式存储，整个数据集是由字典组成的列表。

下面开始做转换。先建立DictVectorizer转换模型对象，这里设置了2个参数：sparse=False指定转换后的数据集是一个数组，否则默认为压缩后的稀疏矩阵，这样设置的原因是后续很多步骤和模型都不支持直接基于压缩后的稀疏矩阵做转换和建模；dtype=np.int64用于设置转换后的数据类型是整数型，否则默认是浮点型。使用fit方法做训练，使用tranform方法做转换应用。最后返回转换模型对象，唯一值总列表和转换后的数据。

相关知识点：使用DictVectorizer将字符串映射为指定数值变量

在之前的章节我们提到过使用OneHotEncoder方法将数值型变量转换为二值化的标志变量，DictVectorizer可以看作是OneHotEncoder方法

的上游步骤，它可以基于用户指定的字符串和任意数值的映射列表（通过字典的形式做映射），对字符串做转换。

假如现在有一个数组，如表6-12所示。

表6-12　原始字符串数据

sex	edu
M	University
M	Middle_School
F	Workers University

我们可以将每条记录转换为字典的形式（参照本节的方法），上述数据可以表示为：[{'sex':1，'edu':1}，{'sex':1，'edu':10}，{'sex':2，'edu':5}]

上述每个变量的唯一值所对应的数值可以任意指定，例如education中的University可以设置为1、1000、1312等任意数字。这也是这种指定方法的灵活性所在，在本案例中我们设置为值所在列表的索引。

使用DictVectorizer的转换具体过程为：

```
from sklearn.feature_extraction import DictVectorizer
model_dv = DictVectorizer(sparse=False)
data =[{'sex':1,'edu':1},{'sex':1,'edu':10},{'sex':2,'edu':5}]
data_new = model_dv.fit_transform(data)
print (model_dv.feature_names_)
print (data_new)
```

上述输出结果为：

```
['edu', 'sex']
[[  1.   1.]
 [ 10.   1.]
 [  5.   2.]]
```

当处于预测阶段时（training==False），由于在训练阶段已经获取了唯一值总列表以及转换模型对象，这里只需要做转换即可。整个过程中，相比于训练阶段主要少了两个步骤：

·无需再次计算唯一值总列表。

·无需对DictVectorizer模型对象应用fit方法，在tranform时，直接使用训练阶段的对象。

时间属性拓展，用于将日期和时间数据拓展出其他属性，例如星期几、周几、小时、分钟等。由于纯时间数据无法作为有效数据集参与分类模型计算，因此这里将其转换出多种属性。

```
def datetime2int(set):
    '''
将日期和时间数据拓展出其他属性，例如星期几、周几、小时、分钟等。
    :param set: 数据集
    :return: 拓展后的属性矩阵
    '''
    date_set = map(lambda dates: pd.datetime.strptime(dates, '%Y-%m-%d'),
                   set['order_date'])  # 将set中的order_date列转换为特定日期格式
    weekday_data = map(lambda data: data.weekday(), date_set)  # 周几
    daysinmonth_data = map(lambda data: data.day, date_set)  # 当月几号
    month_data = map(lambda data: data.month, date_set)  # 月份

    time_set = map(lambda times: pd.datetime.strptime(times, '%H:%M:%S'),
                   set['order_time'])  # 将set中的order_time列转换为特定时间格式
    second_data = map(lambda data: data.second, time_set)  # 秒
    minute_data = map(lambda data: data.minute, time_set)  # 分钟
    hour_data = map(lambda data: data.hour, time_set)  # 小时

    final_set = []  # 列表，用于将上述拓展属性组合起来
    final_set.extend((weekday_data, daysinmonth_data, month_data, second_dat
属性列表批量组合
    final_matrix = np.array(final_set).T  # 转换为矩阵并转置
    return final_matrix
```

本函数的功能是使用map配合lambda为指定的数据集应用特定功能，具体包括以下过程：

先通过date_set=map（lambda dates:pd.datetime.strptime（dates，'%Y-%m-%d'），set['order_date']）将数据集中的order_date字符串转换为指定日期格式的数据。有关日期格式的转换，我们之前已经分别介绍了在导入数据时先定义一个匿名函数date_parse=lambda dates:pd.datetime.strptime（dates，'%m-%d-%Y'），再

配合导入参数date_parser=date_parse做解析的方法以及直接使用pd.to_datetime（）为指定列做转换的方法。这里介绍的方法应用格式类似，只是用了不同的实现过程。接下来分别通过weekday（）、day、month获取转换后日期数据的周几、当月几号和月份。

再使用相同的方法对数据集中的order_time类做转换，然后分别应用second、minute、hour获得其秒、分钟、小时数据。

上述两个转换过程完成后，通过列表的extend方法批量添加到一个空列表中，并转换为数组后再转置，最后返回该数组。

相关知识点：使用map配合lambda实现自定义功能应用

lambda匿名函数在4.6节中已经介绍过，在此重点介绍map函数。

map是Python内置的标准函数，它的作用是为一个序列对象的每个元素应用指定的函数功能，然后返回一个列表。语法：map（function，sequence[，sequence，...]）

其中：

·function可以是任意函数，一般情况下是针对单个元素实现的具体映射功能。

·sequence是要应用的列表。

map（）函数的特点是小巧灵活，再配合lambda可以应用到很多小型的迭代过程中，例如日期格式转换、大小写转换、对每个元素做切分、数值计算等以个体为单位的功能中。

样本均衡，使用SMOTE方法对不均衡样本做过抽样处理。

```
def sample_balance(X, y):
    '''
使用SMOTE方法对不均衡样本做过抽样处理
    :param X: 输入特征变量X
    :param y: 目标变量y
    :return: 均衡后的X和y
    '''
    model_smote = SMOTE()  # 建立SMOTE模型对象
    x_smote_resampled, y_smote_resampled = model_smote.fit_sample(X, y)  # 转
```

```
入数据并做过抽样处理
    return x_smote_resampled, y_smote_resampled
```

本函数的定义过程比较简单：先建立SMOTE（）模型对象，然后使用其fit_sample方法做训练后过抽样处理，最后返回处理后的数据集。

步骤3　读取数据，从这里开始进入到正式的数据应用过程。

```
# 定义特殊字段数据格式
dtypes = {'order_id': np.object,
          'pro_id': np.object,
          'use_id': np.object}
raw_data = pd.read_table('abnormal_orders.txt', delimiter=',', dtype=dtypes)
取数据集
```

定义的dtypes用来为特定列指定数据类型为字符串型，该列表在训练数据和预测数据读取时都会用到。使用Pandas的read_table方法读取训练集并指定分隔符和指定列的数据类型。

步骤4　数据审查。

基本状态查看，执行set_summary（raw_data），从返回的结果中分析：

通过基本状态输出的概览，未发现数据有录入异常。

```
**********Data overview:**********
          order_id  order_date order_time     cat attribution      pro_id  \
134188      4285770012      2013-09-19         23:55:06                家居日
用              GO  1000335947
134189     4285770056     2013-05-20       23:58:59            生活电器和厨卫电
器       GO  1000009280
       pro_brand  total_money  total_quantity order_source pay_type  \
134188       炊大师         79.0               1           抢购合并支付
134189        海尔        799.0               1           抢购合并支付
          use_id city  abnormal_label
134188      shukun  东莞市               0
134189  544975322_  海口市               0
```

通过数据类型输出，发现我们指定的几个列以及其他包含有字符串的列都被识别为字符串型，其他类型识别正常。

```
**********Data dtypes:**********
order_id           object
order_date         object
order_time         object
cat                object
attribution        object
pro_id             object
pro_brand          object
total_money       float64
total_quantity      int64
order_source       object
pay_type           object
use_id             object
city               object
abnormal_label      int64
dtype: object
```

通过描述性统计，发现total_money和total_quantity中存在了极大值。但是我们选择不做任何处理，因为本节的主题就是针对异常值的分类检测。

```
**********Data DESC:**********
                   count    mean      std  min   25%   50%    75%       max
total_money     134189.0  660.11  2901.21  0.5  29.0  98.4  372.0  766000.0
total_quantity  134190.0    1.20     3.23  1.0   1.0   1.0    1.0    1000.0
abnormal_label  134190.0    0.21     0.41  0.0   0.0   0.0    0.0       1.0
```

在使用describe做描述性统计时，默认情况下，只针对数值型数据做统计，如果要对全部数据做统计，那么需要使用describe（include='all'）。

缺失值审查，执行na_summary（raw_data），从返回的结果中分析有三列值存在缺失，总的缺失记录为1429，占总体样本量的1%（1429/134189）左右。

```
**********NA Cols:**********
order_id          False
order_date        False
order_time        False
cat                True
attribution       False
pro_id            False
pro_brand          True
total_money        True
total_quantity    False
order_source      False
pay_type          False
```

```
use_id            False
city               True
abnormal_label    False
dtype: bool
Total number of NA lines is: 1429
```

类样本分布审查，执行label_samples_summary（raw_data），从结果中发现数据存在一定程度的不均衡，异常值记录（label为1）与非异常值的比例为1:3.7左右。该结果可以处理也可以不处理，这里我们选择处理。

```
******************Labesl samples count:********************
abnormal_label
0    105733
1     28457
Name: order_date, dtype: int64
```

步骤5　数据预处理，经过上面的基本分析，我们对特定问题基本有初步的判断，该步骤着手进行处理。

```
drop_na_set = raw_data.dropna()  # 丢弃带有NA值的数据行
X_raw = drop_na_set.ix[:, 1:-1]  # 分割输入变量X，并丢弃订单ID列和最后一列目标变量
y_raw = drop_na_set.ix[:, -1]    # 分割目标变量y
model_dvtransform, unique_object, str2int_data = str2int(X_raw, None, None,
符串分类转整数型分类
datetime2int_data = datetime2int(X_raw)  # 拓展日期时间属性
combine_set = np.hstack((str2int_data, datetime2int_data))  # 合并转换后的分类
和拓展后的日期数据集
constant_set = X_raw[['total_money', 'total_quantity']]  # 原始连续数据变量
X_combine = np.hstack((combine_set, constant_set))  # 再次合并数据集
X, y = sample_balance(X_combine, y_raw)  # 样本均衡处理
```

丢弃NA值：由于样本量足够大，因此我们在处理中会选择丢弃缺失值，这是一种“大数据”量下的缺失值问题。使用drop方法丢弃，形成不含有NA值的drop_na_set。

分割输入变量X：在drop_na_set基础上，使用ix方法获取从第二列开始到倒数第二列的数据，形成输入变量集合X_raw。第一列为订单ID，该列用于区别每个订单，因此该唯一区别值不具有规律特征；最后一列是目标变量y。

分割目标变量y：在drop_na_set基础上，使用ix方法获取最后一列数据，形成目标数据集y_raw。

字符串分类转整数型分类：直接调用str2int方法对X做转换，返回model_dvtransform、unique_object、str2int_data分别是训练后的DictVectorizer对象、唯一值总列表和转换后的数值型分类。

拓展日期时间属性：直接调用datetime2int方法对X做转换，形成结果集datetime2int_data。

合并转换后的分类和拓展后的日期数据集：使用Numpy的hastck方法将分类和拓展后的日期数据集沿列合并，形成合并数据集combine_set。

将原始数据集中的total_money和total_quantity列数据集提取出来，然后再次使用numpy的hstack方法与combine_set做合并，至此形成了完整的输入变量集X_combine。

最后调用sample_balance函数对X_combine和y_raw做过抽样处理，形成最终结果集X和y。

步骤6　组合分类模型交叉检验，这是本节内容的核心。

```
model_rf = RandomForestClassifier(n_estimators=20, random_state=0)  # 随机森林分类模型对象
model_lr = LogisticRegression(random_state=0)  # 逻辑回归分类模型对象
model_BagC = BaggingClassifier(n_estimators=20, random_state=0)  # Bagging分类模型对象
estimators = [('randomforest', model_rf), ('Logistic', model_lr), ('bagging'
立组合评估器列表
model_vot = VotingClassifier(estimators=estimators, voting='soft', weights=
[0.9, 1.2, 1.1], n_jobs=-1)  # 建立组合评估模型
cv = StratifiedKFold(8)  # 设置交叉检验方法
cv_score = cross_val_score(model_vot, X, y, cv=cv)  # 交叉检验
print ('{:*^60}'.format('Cross val socres:'))
print (cv_score)  # 打印每次交叉检验得分
print ('Mean scores is: %.2f' % cv_score.mean())  # 打印平均交叉检验得分
model_vot.fit(X, y)  # 模型训练
```

建立多个分类模型对象。通过RandomForestClassifier方法建立随机森林分类模型对象model_rf，设置分类器数量为20，目的是希望通过更多的分类器达到更好的分类精度；设置随机状态为0，目的是控制每次随机的结果相同。然后，按照类似的步骤分别建立逻辑回归分类模型对象model_lr、Bagging分类模型对象model_BagC。


RandomForest、Bagging以及之前我们用到的AdaBoost、Gradient Boosting都是常用的集成方法，除这些外，sklearn.ensemble中还提供了extra-trees、Isolation Forest等多种集成方法。这些集成方法大多数都既有分类器又有回归器，意味着可以用于分类和回归。

建立一个由模型对象名称和模型对象组合的元组的列表estimators。其中：对象名称为了区分和识别使用，任意字符串都可以；模型对象是上面建立的三个分类器对象。该列表用于组合投票模型器的参数设置。

使用VotingClassifier方法建立一个基于投票方法的组合分类模型器，具体参数如下：

·estimators：模型组合为上面建立的estimators列表。

·voting：投放方法设置为soft，意味着使用每个分类器的概率做投票统计，最终按投票概率选出；还可以设置为hard，意味着通过每个分类器的label按得票最多的label做预测输出。

·weights：设置三个分类器对应的投票权重，这样可以将分类概率和权重做加权求和。

·n_jobs：设置为-1，意味着计算时使用所有的CPU。


在设置voting参数时，如果设置为soft，要求每个模型器必须都支持predict_proba方法，否则只能使用hard方法。例如，使用SVC（SVM的分类器）时，就只能设置为hard。

使用StratifiedKFold（8）设置一个8折交叉检验方法，这是一个按照目标变量的样本比例进行随机抽样的方法，尤其适合分类算法的交叉检验。

通过cross_val_score方法做交叉检验模型，设置的参数分别为上述的组合分类模型器和交叉检验方法。打印输出每次交叉检验的结果以及均值如下：

```
**********Cross val socres:**********
[ 0.77178407  0.91971669  0.97473201  0.9724732   0.92316233  0.90960257
  0.91006203  0.91538403]
Mean scores is: 0.91
```

从交叉检验结果看出，8次交叉检验除了第一次结果略差以外，其他7次都比较稳定，整体交叉检验得分（准确率）达到91%，说明其准确率和鲁棒性相对不错。

在检验之后，我们直接对组合分类模型器model_vot应用fit方法做模型训练，形成可用于预测的模型对象。

步骤7　对新数据集做预测，该部分跟训练集时的思路相同，仅少了分割目标变量y和样本均衡两个环节，在此不做赘述。

```
X_raw_data = pd.read_csv('new_abnormal_orders.csv', dtype=dtypes)  # 读取要预测的数据集
X_raw_new = X_raw_data.ix[:, 1:]  # 分割输入变量X，并丢弃订单ID列和最后一列目标变量
str2int_data_new = str2int(X_raw_new, model_dvtransform, unique_object, trai
符串分类转整数型分类
datetime2int_data_new = datetime2int(X_raw_new)  # 日期时间转换
combine_set_new = np.hstack((str2int_data_new, datetime2int_data_new))  # 合并转换后的分类和拓展后的日期数据集
constant_set_new = X_raw_new[['total_money', 'total_quantity']]  # 原始连续数据变量
X_combine_new = np.hstack((combine_set_new, constant_set_new))  # 再次合并数据集
y_predict = model_vot.predict(X_combine_new)  # 预测结果
print ('{:*^60}'.format('Predicted Labesls:'))
print (y_predict)  # 打印预测值。
```

最后使用model_vot的predict方法做预测并打印结果值如下：

```
**********Predicted Labesls:**********
[1 0 0 0 0 0 0]
```

6.8.5 案例数据结论

在该案例中，91%的准确率是一个比较高的结果，多数据集的鲁棒性表现也很突出。这首先得益于各个分类评估器本身的性能比较稳定，尤其是集成方法的随机森林和Bagging方法；其次是基于预测概率的投票方法配合经验上的权重分配，会使得经验与数据完美结合，也会产生相互叠加效应，在正确配置的前提下，会进一步增强组合模型的分类准确率。

6.8.6 案例应用和部署

对于该类型的案例，通常情况下的应用分类有两种情况：

（1）异常订单的分析

包括异常订单的主要特征、品类集中度、重点客户等，尤其是可以将异常订单联系人加入黑名单，以降低其对公司正常运营的干扰。

（2）订单的实时检测

订单实时检测后会将结果为异常的订单记录发送到审核部门，然后由审核部门做进一步审查。这时需要额外做两部分工作：

一部分是将训练过程与预测过程分离，目的是训练阶段与预测阶段形成两个并行进程。实时检测的前端属于预测阶段的实时调用，仅应用本案例的步骤7，这个过程中还有一步非常重要的内容是将训练过程中用到的模型对象（包括DictVectorizer对象和VotingClassifier对象等）持久化保存到服务器硬盘，这样做的好处是在预测应用调用不同的模型对象时需一次引用，多次调用，将功能分离的同时又会大大提高运行效率。

一部分是在训练阶段做增量更新，而不必每次都跑所有数据，这样也能提高实时运算效率。

限于篇幅，本书暂时不过多介绍订单的实时检测中提到的两部分内容。

6.8.7 案例注意点

在本案例中有几个关键点需要读者注意：

（1）关于耗时

以笔者的工作环境，完整运行一次大约需要需要15分钟时间，这里面主要有两个耗时的环节：

·训练阶段的字符串分类转整数型分类str2int，该过程需要对矩阵中的每个值做映射。

·训练阶段的交叉检验，该过程由于是8折交叉检验并且里面有2个集成方法，再加上使用组合投票的分类器的应用，导致整个交叉检验耗时较长。

因此，如果读者更侧重于效率的话，那么这种基于组合投票以及集成分类方法的实现思路将不是优先选择。

（2）关于输入特征变量

本案例中应用了两个特殊字段pro_id和use_id，这两类ID一般作为关联主键或者数据去重唯一ID，而很少用于模型训练本身。这里使用的原因是希望能从中找到是否异常订单也会集中在某些品类或某些客户上。笔者经过测试，如果把这两个维度去掉，整个模型的准确率会下降到70%以下。

（3）关于样本均衡

由于本案例中的两类数据差异并没有特别大（例如1:10甚至更大），因此均衡处理不是必须的。本案例中，由于运营对于异常的定义比较宽松，因此才会形成大量异常名单。实际上异常检测在很多情况下的记录是比较少的，因此样本均衡操作通常必不可少。

6.8.8 案例引申思考

（1）更高的模型准确率或更低的模型误差

本案例的核心建模思路是通过组合多个单一或集成模型器的方法，实现更高层次的手动集成分类模型。我们在6.7节中提到了自动参数优化方法，该方法可以与本案例的组合模型器相结合，这样会进一步提高模型效果。另外，我们在5.8节也介绍过通过交叉检验得到不同参数下的模型效果值，并手动做最优参数配置的方法。这三种方法，尤其是前两种在追求高准确率或低误差的应用场景中非常有效。

（2）有关二值标志转换的问题

虽然本案例中没有用到二值标志转换操作，但由于用到了两个具有特殊意义的字段pro_id和、use_id，在此讨论下如果使用了这类唯一性特别强或分类特别多的变量，会遇到哪些问题。这类字段通常会产生海量的唯一值，如果对其做二值化标志转换，那么会出现两个主要问题：

第一，海量的商品ID和用户ID会使得数据稀疏特征非常明显，数据将主要受到这两个特征所转换的海量特征的影响，而使得整个数据集的类别划分效果变得非常差。

第二，如果通过现有的二值化转换方法OneHotEncoder来得到的数组结果，在面临海量唯一值时会报错"ValueError:array is too big; `arr.size*arr.dtype.itemsize`is larger than the maximum possible size."，原因是：

·在32位Python版本下由于系统限制，即使内存再多，Python进程最多只能占用2G内存（如果使用IMAGE_FILE_LARGE_ADDRESS_AWARE方法将二值化结果做压缩，最多也只能占用4G内存，实际上大多数32位Python都没有做该设置）。

·即使在64位Python版本下，Numpy（sklearn基于numpy）的最大数据类型是complex128，其最大可占用的内存空间大约有5G左右，而ID对象的唯一值可能有无限大，例如将销售周期拉长，会发现销售SKU能达到几千万甚至上亿。这种有限的对象长度将无法承载无限增长的ID。

当然，可能有的读者会想到通过二值化转换形成压缩稀疏矩阵，默认的OneHot-Encoder方法也是提供基于csc（一种稀疏矩阵压缩方式）压缩稀疏矩阵，scipy.sparse也提供了相关方法可以基于压缩稀疏矩阵做数据合并等处理。这种想法没错，但是大多数的数据处理的输入都要求是数组或矩阵，转换为数组几乎是建模的前置条件，压缩矩阵目前则完全行不通。

（3）字符串分类转整数型分类

字符串分类转整数型分类以及后续的二值化标志问题，应用的前提是训练集中被转换的唯一值域必须是固定的，否则在预测集转换时遇到新数据值时就会报错。这点在之前提到过，在这里再次提出，希望读者注意。

（4）有关数据集中的NA值

在数据处理的一开始，我们就已经将NA值排除了。但读者是否想过，如果预测应用时，再次出现NA值该如何处理？

我们先分析输入变量在订单信息生成时，是否允许出现缺失值。attribution、cat、pro_id、pro_brand这几个字段都是根据数据库中商品信息自动匹配的，因此不应该出现缺失值；order_id、order_date、order_time、total_money、total_quantity、order_source、use_id、city是订单时生成的必填信息，也不应该有缺失；而pay_type要看具体业务部门如何定义：

·如果基于已经支付的订单做异常分类检测，那么该字段不应为空；

·如果基于全部订单做异常分类检测，那么该字段会经常出现为空的情况。

基于上述分析，我们会有如下对应策略：

·针对attribution、cat、pro_id、pro_brand字段，只要有pro_id（商品ID），就可以从商品库中匹配出这些信息来。

·针对order_id、order_date、order_time、total_money、total_quantity、order_source、use_id、city字段，只要有order_id，就可以从订单库中匹配出这些信息来。

如果商品库和订单库没有两个ID，或者即使匹配回来的数据仍然有NA值如何处理？由于这些数据理论上不应该为空，建议将其筛选出来单独存储（当然不作预测），然后跟IT部门沟通，分析到底为什么会出现缺失值并制定补足策略，该策略会应用到缺失值处理过程中。

如果缺失值无法预测、也无法避免，那么可以通过条件判断，如果数据记录中有缺失值则不做检测，毕竟缺失值只占1%不到；如果读者认为有必要，则可以将缺失值作为一种特殊值的分布形态，以具体值（例如0）做填充，用于后续数据处理和建模使用，这也是一种行之有效的方法。

通常，很多数据处理环节对NA是“无法容忍”的，例如OneHotEncoder就无法将NA值转换为二值化矩阵，因为NA不是整数型数据。此时将NA值以特定值填充转换是一种变通思路。

6.9 本章小结

内容小结：商品数据化运营几乎是每个公司做精细化运营、销售提升的重要支撑，本章几乎每个内容都能在实际运营中找到对应落地点，它们不仅可以用于商品运营分析，更可以用于其他商品自动化运营的场景，例如个性化推荐、智能促销等。

重点知识：本章需要读者重点掌握的知识点是商品数据化运营分析的模型、小技巧以及最后两个案例中的效果优化和提升方法，尤其是：

·使用lambda配合map实现特定功能

·使用SMOTE做样本均衡处理

·新产品市场定位模型

·商品规划的最优组合

·基于GridSearchCV的超参数的模型优化方法

·基于多种模型的投票组合模型的构建

外部参考：限于篇幅，本书未能言尽的内容包括：

·商品数据化运营涉及很多有关运营方面的话题，如何将运营与数据更好的结合，可以参考《数据化管理：洞悉零售及电子商务运营》，这是一本简明易懂的将数据与运营场景结合的书籍，其中的经验不可多得。

·市场分析不只是服务于商品运营，更能从整个企业宏观层面做企业级预测支持。在市场分析领域，麦肯锡处于企业咨询服务和市场服务的领先地位，有兴趣的读者可以阅读《麦肯锡问题分析与解决技巧》。这不是一本讲技术的书籍，但它能从思路、方法、步骤等方面帮你建立良好的思维习惯，从“术”和“法”走向更高的“道”。

·线性规划方法是非常有效的决策方法，与之对应的是非线性规划方法，有关这两种方法的更多内容，请读者参考《线性和非线性规划

（第3版）》，这是一本研究运筹学的经典教材。

·在6.8节中的案例的最后提到了将Python对象持久化和增量更新的问题。有关Python对象的持久化，请查阅资料了解Python的内置标准库pickle和cpickle，尤其是大型程序环境中后者用的比较多；对于Python数据挖掘和机器学习的增量更新，sklearn中已经提供了不少算法，如表6-13所示，有兴趣的读者可以做进一步了解和学习。

表6-13　sklearn支持增量学习的算法

算法类别	算　法
分类	sklearn.naive_bayes.MultinomialNB sklearn.naive_bayes.BernoulliNB sklearn.linear_model.Perceptron sklearn.linear_model.SGDClassifier sklearn.linear_model.PassiveAggressiveClassifier
回归	sklearn.linear_model.SGDRegressor sklearn.linear_model.PassiveAggressiveRegressor
聚类	sklearn.cluster.MiniBatchKMeans
数据预处理	sklearn.decomposition.MiniBatchDictionaryLearning sklearn.decomposition.IncrementalPCA sklearn.decomposition.LatentDirichletAllocation

应用实践：虽然本书尽量在各个知识的讲解中穿插步骤或示例做说明，希望能通过更多的案例帮助读者理解每个模型和方法如何应用，但“纸上得来终觉浅”，6.4节和6.5节中的每个模型、方法和技巧都需要读者多加练习才能融会贯通；尤其是本章最后两个案例演示了如何通过系统方法找到最优模型，不是所有的工作都交给程序完成，仍然需要读者具备一定的模型、算法、数据的理解能力，这样才能设置合适的参数列表并寻找合适的组合模型器。

第7章　流量数据化运营

流量是企业获得用户的第一步，对于大多数需要“自力更生”企业而言，流量几乎是企业运营的命脉之一，没有流量就没有一切。本章将围绕流量运营的相关话题，从流量采集处理工具、流量数据与企业数据的整合、流量运营指标、流量数据化运营分析模型、流量分析小技巧和“大实话”等方面展开，最后通过两个案例展示如何做流量建模分析。

7.1 流量数据化运营概述

本章的流量指从数字设备上访问企业的网站、APP应用、智能设备的用户行为，它主要包括用户从哪里来，在企业相关载体上有哪些行为、产生了哪些转化等。流量（Traffic）一词主要应用在互联网、广告、电子商务等相关行业和领域。企业经营的基础是客户，但是获得客户的第一步是通过一定广告媒介获得用户关注，然后才能产生访问、转化等行为。

在媒体信息碎片化，用户行为移动化、需求个性化的复杂背景下，企业想要获得用户关注愈发困难。并且随着营销成本的增加，企业希望每获取一个单位的流量就能将其转化为企业客户甚至订单客户，因此精准营销需求日益突出。

流量数据化运营正是在这种背景下产生的，它要解决的本质问题是如何通过数据获得更多流量以及有效流量，然后完成最大化的营销转化目标。

7.2 8大流量分析工具

关于流量数据的采集，大多数企业都选择使用第三方流量分析工具（或者称为网站分析工具，本书中两个词的含义相同），个别资源丰富的企业也会选择自建流量分析系统。从综合成本和产出价值来考虑，前者的应用性更加广泛；从流量的保密性、可控性、二次开发性以及跟企业内部数据做无缝整合的角度看，后者的应用更为有效。这里介绍一些市场上主流的流量分析工具，它们可以提供流量数据的跟踪、采集、配置、处理、分析和整合功能，其中有免费也有付费的。

1.Adobe Analytics

Adobe Analytics是世界范围内领先的付费网站分析解决方案，它是Adobe Marketing Cloud的一部分。除Adobe Analytics外，Adobe Marketing Cloud还包括Adobe Campaign、Adobe Social、Adobe Media Optimizer、Adobe Target和Adobe Experience Manager。

Adobe Analytics可以根据客户需求采用SAAS或Local的服务模式，数据监测范畴覆盖WEB/WAP/APP。它的功能范畴覆盖了数据跟踪采集、集成服务、本地网站分析解决方案、数据仓库、标签管理器等网站数据工作的上下游。同时，Adobe Analytics通过与Marketing Cloud中的其他产品整合，拓展了网站数据与业务结合的领域，包括测试优化、推荐、搜索、媒介管理、网站管理等业务模块。Adobe Analytics产品包括Analysis Workspace、报表生成器、报告与分析、临时分析、数据工作台、数据仓库、Activity Map/ClickMap。

Adobe Analytics进入中国市场较早，凭借其技术实力及在华跨国子公司占据了很大一部分国内市场，也是国内付费网站分析工具领域最主要的工具和服务供应商。

2.Webtrekk Suite

Webtrekk Suite是Webtrekk公司的数字智能解决方案，它可以实现跨营销广告、跨投放媒体、跨设备（WEB/WAP/APP）的数据跟踪及A/B测试功能。数据整合方面，Webtrekk Suite提供了通过自动端口、多种API协议以及数据源集成的整合方式。

Webtrekk Suite包含分析、数据管理平台、智能推荐、测试、标签集成和客户生命周期管理，通过内部多个模块的打通可以实现用户从到站到出站的所有行为检测与分析。另外，Webtrekk Suite还与多个广告平台、再营销平台、搜索引擎平台、社交网络平台打通集成，可以在网站和应用外部监测用户行为。

Webtrekk进入中国时间较晚，因此在市场时机上不占天时，但通过最近几年的发展以及本地化的快速响应，它已经广泛获得客户的认可并具有很高的市场知名度。

3.Webtrends

Webtrends作为网站分析工具的鼻祖（1993年成立），拥有比其他任何工具都早的品牌沉淀。Webtrends诞生于互联网早期，其产品也体现了当时的技术特点，即网站分析技术针对日志做分析，这也是Webtrends产品最突出的特征。随着互联网技术的不断发展和网站内容的日新月异，日志分析法已经开始出现不适应网站监测分析需求的情况，为了跟上时代的发展，Webtrends公司也开始提供在网页中加入“标签”的方式来实现数据监测，即通过SAAS模式为客户提供服务，但分析日志是其主要的产品特征。

从功能上来看，Webtrends基本不逊色于其他网站分析工具，从WEB/WAP/APP的设备跟踪覆盖，到普通的广告渠道、站内页面、用户和电子商务跟踪，再到更高级的应用，比如多渠道跟踪、A/B测试和优化，甚至与SAP、CRM和SharePoint打通做数据整合应用的功能都应有尽有。Webtrends的功能特征虽然与其他工具的实现方法不同，但结果差异不大。

Webtrends于1999年进入中国，由于其技术实力、行业经验及国外知名客户案例多等优势，再加上其本地化的部署方式，能最大化满足国内企业数据安全性的需求，因此初期客户规模庞大。

4.Google Analytics

Google Analytics最早基于谷歌收购的Urchin产生，之后几经演变形成现在的免费版本Universal Analytics以及付费版本Google Analytics360 Suite。Google Analytics以及之后的Universal Analytics是世界范围内应用

最为广泛的免费流量分析工具之一，由于它功能相对完善、易用性高、免费等特点，受到几乎所有行业的青睐，目前几乎已经成为免费网站分析工具的标杆。

由于各种原因，Google Analytics360 Suite仍未在中国大规模销售，这里我们仅讨论其免费版本Google Analytics。Google Analytics采用SAAS的服务模式，所有数据的跟踪采集和处理只能在Google云端服务器执行，它能广泛跟踪WEB/WAP/APP等不同平台。

虽然这是一个免费版本，但其数据跟踪、采集、定义、分析等方面的能力非常强大，它几乎已经满足大多数中小企业的日常需求。当然，对大型企业来说，受其流量处理规模的限制、有限的定制特性的限制，通常无法满足很多个性化、灵活性和大规模数据分析和整合的企业需求。这点已经成为大型企业应用的桎梏。

5.IBM Coremetrics

自从2010年Coremetrics被IBM收购之后，Coremetrics就作为IBM中EMM（Enterprise Marketing Management）的一部分而存在。IBM coremetrics主要包括两部分：网站分析套件（Web Analytics Suite）和数字营销优化套件（Digital Marketing Optimization Suite）。前者是针对网站流量统计、分析和数据挖掘的解决方案，后者是从网站分析套件中获取数据洞察和价值驱动点，然后整合到自身营销优化应用中，通过网络、社交和移动应用等针对性地进行业务活动。Coremetrics也是基于SAAS的服务模式。

6.百度统计

凭借百度在国内搜索引擎霸主的地位、广泛的SEM客户资源、雄厚的技术实力和国内较低层次的数据分析需求，百度统计已经成为世界上应用最多的免费网站统计工具之一。百度统计功能简单实用，页面体验较好，其功能侧重点是统计功能，覆盖WEB/WAP/APP跟踪统计，通过自定义还可实现跨屏分析、子目录、转化路径、事件跟踪和自定义变量分析。整体来看，百度统计满足了企业网站分析入门时只看数据统计结果的需求，尤其是做大规模百度投放的客户，结合这些数据后更利于SEM优化。

7.Flurry

Flurry是国外的一款基于SAAS的免费移动应用分析工具，它是移动应用统计分析领域的标杆，支持对iOS、Android、Blackberry、Windows平台以及使用Java开发的平台的跟踪，除了统计单个应用内的各类数据指标外，还可以提供跨应用之间的转化统计等针对企业级用户的功能。Flurry在移动应用分析领域的地位与Google Analytics在全球网站分析领域类似。

8.友盟

友盟是目前国内免费移动应用数据监测的领头羊，2013年被阿里巴巴收购。友盟提供针对iOS、Android和Windows Phone等多平台的服务，基于SAAS服务模式，用户只需要注册并下载友盟SDK并集成到APP开发中就可以实现针对APP的数据跟踪及其他服务。

7.3 如何选择第三方流量分析工具

选择第三方流量分析工具，需要综合其解决方案能力、产品易用性、功能丰富性、增值服务价值和费用，更重要的是需要结合企业自身需要做有效评估；否则，即使工具性能再强大，企业也无法发挥其真正价值。

1.整体解决方案能力

整体解决方案能力是指能完整的、与其他工具或解决方案融合提供更广泛支持的能力。整体解决方案能力包括两部分：整合数据（含内、外部数据跟踪）的能力，整合运营系统的能力。

（1）整合数据系统的能力

众所周知，网站数据只是企业数据的一部分。从网站数据开始，按照由个体到整体的数据范畴分别是：网站数据、运营数据、业务数据、企业数据四个层次。

·网站数据：以网站（WEB/WAP/APP）为数据生产环境，主要是站内流量相关数据。

·运营数据：围绕运营形成的数据环境，除站内流量数据外，还包括站外运营数据，如营销数据、线下数据等，不同的运营范畴定义具有不同的数据规模。如果运营范畴只为围绕网站端的所有业务动作，可能只包括营销、运营、用户体验和在线销售相关数据；如果运营有更大的业务范畴，可能还包括会员维系、活动策划、产品规划、市场规划等相关的数据。

·业务数据：围绕整个业务体系形成的数据环境，除运营数据外，还包括运营的上下游业务部门的数据，如IT、HR、管理、监察等体系数据，业务数据是企业所有业务类数据的总称。

·企业数据：企业数据包含所有企业产生的业务数据、财务数据、职能数据等，所有数据构成企业大数据集合。

网站数据只占企业整体数据的一小部分，无论是通过外部数据整合到网站分析系统之内，还是将网站数据整合到企业数据仓库之内，网站分析工具整合数据系统的能力越强，越容易实现数据集成以及基于整合数据的完整视角。

（2）整合运营系统的能力

网站数据发挥价值的方式之一是通过数据相关从业者提供数据解读，从而为业务策略和执行提供辅助建议；另外一种方式是通过相关系统对接，直接通过数据驱动的方式将数据价值输出到业务系统，实现数据的自我价值。

常见的与网站分析工具集成的运营系统包括：

·CRM：通过网站流量数据固定触发，针对性地完成CRM相关流程。如针对已经登录并将商品加入购物车但放弃购物的用户，网站分析工具将数据传输到CRM中，CRM根据预设条件进行判断并执行，如可针对性地发送优惠券、打折信息或其他刺激购买的方式提高转化率。

·销售系统：网站分析工具将网站流量数据传输到销售预测系统中，销售预测系统根据产品浏览趋势、用户属性和来源，以及转化率数据综合评估出未来N天的产品销售情况，并将该数据传输到相关库存系统，及时提醒相关采购补货。

·站外营销系统：网站分析工具对站内用户的关键行为进行采集并提取出特定特征，然后将用户喜好信息反馈到营销系统，营销系统根据此信息优化站外投放结构、素材和其他营销策略。

·站内推荐系统：网站分析工具将特定数据传输到站内推荐系统，站内推荐系统根据用户行为有针对性地推荐其喜好的内容或产品。

·网站运营系统：大多数网站运营系统都靠人工以手动方式调整素材和内容等，网站分析工具可以将固定广告位、内容位、资源位数据回传到网站运营系统中，网站运营系统根据数据自动调整页面展示内容，提高运营效率。

2.产品易用性

大多数企业在考虑产品易用性时只考虑了业务部门的应用需求，而忽视了IT部门的实施需求。业务部门与IT部门对网站分析工具易用性的关注点截然不同。

（1）IT部门关注产品易实施

大多数情况下，IT部门往往是企业评估网站分析工具时的重要参与者，对于IT部门来说，产品稳定性高、实施方便、技术开发难度小、数据安全性高是首要关注因素。对于SaaS模式而言，产品易实施是其最重要的关注点。整个网站分析项目，从计划到上线往往会耗费大量人力成本，IT部门的技术开发工作往往占据主要时间，容易实施的技术项目更容易得到IT部门的认可。

例如：实现页面热力图功能，不同的工具有不同的解决方案。有的工具只需要一段通用代码，有的工具需要单独标记热力图参考点，有的工具则需要将全页面元素ID指定给系统相关变量才能获得可靠的热力图数据。对IT部门来说，自然会选择第一种方案。

容易实施的技术方案不仅能减少技术开发工作量，从而保证网站分析项目快速落地，而且可以减少由于代码过多导致的版本更新、代码发布等网站数据采集的问题。

（2）业务部门关注产品易使用

网站分析工具的使用者是企业中的关键用户，包括网站分析师、业务人员、产品经理、交互设计师、技术工程师等。不同的用户具有不同的数据需求，同样一类用户由于职位不同也有不同的数据侧重点。

面对这种复杂的数据需求矩阵，如何让用户根据自身需要而快速、有效、深入地得出结论是评估一款网站分析工具的重要指标。

·以用户角色为出发点的产品定位。领导层和执行层看的数据不同，领导层关注结果、趋势或汇总数据，因此Dashboard、汇总报表必不可少，如果能有一个报表能涵盖领导日常关注的核心指标已经足够，通过多种发送、实时更新或数据对接产生更多的附加值则是锦上添花；执行层关注原因、细节和深入探究，因此多层级下钻、多维度交叉分析、用户群体细分、路径流及各种数据分析和挖掘模型必不可少。

·以用户应用为导向的功能设计。根据不同的业务模块划分数据报表，营销推广、站内运营、用户体验、在线销售等不同业务模块需要不同的报表，报表需要单独拆分呈现；并且要让用户以少的点击、最快的速度找到答案，尽量减少业务思考数据在哪的时间。

如果企业中只有网站分析师在使用工具，那么该企业的数据落地通常比较差——从不使用工具的人怎么会从数据中得到建议？

3.功能丰富性

对于网站分析工具，功能越丰富代表可通过工具获得更多数据视角的机会越多。网站分析工具的功能可以分为四类：基本功能、自定义功能、高级功能和特色功能。

·基本功能：基本维度、基本指标、APP跟踪、WAP跟踪、用户权限管理、热力图、Excel插件、标签管理器、下钻功能。

·自定义功能：自定义维度跟踪、自定义事件跟踪、自定义指标跟踪、自定义计算指标、自定义数据分类、自定义报表、自定义Dashboard等。

·高级功能：标签管理器、跨域追踪、跨设备跟踪、单归因功能、A/B测试、路径功能、漏斗功能、数据整合能力、实时数据、预警功能、自动发送服务等。

·特色功能：当前市场上的网站分析工具，尤其是商用付费工具的功能特性基本相同。但不同的工具具有一些其他工具没有的特征。例如Adobe Analytics的异常检测、无穷维度下钻、多序列模型的过滤器，Webtrekk的关联算法模型、描述性统计特征、预测性数据指标等。

4.增值服务价值

是否购买增值服务主要基于2个方面的因素考虑：

·在涉及部署、实施和二次开发等具体技术细节时，很可能需要厂家的技术支持。

·第三方服务商可以基于行业经验提供较好的应用建议，减少企业走弯路的风险，并可借鉴成功案例的实施思路，从而应用到企业自身。

增值服务的考察点分为3个方面：

（1）日常支持

日常支持会涉及部署、沟通、培训等各方面的问题，日常支持的方式（邮件、电话、进驻企业）、响应效率（2小时答复、7×24，还是其他）等都是重要参考因素。

（2）原厂服务团队

大多数国外网站分析工具都是通过代理商提供产品销售和服务支持，但由于各方面的问题，代理商团队很可能无法提供高质量的咨询服务。如果有原厂服务团队支持，无论是本地还是远程沟通，将更利于问题解决。

（3）Local办公和本地化作业

Local办公是指在中国有分公司或办公室，本地化作业是指根据企业需求入驻到企业内部共同推进该项目，这两方面是评价一个网站分析工具服务商本地化的重要参考标准。在开展项目工作尤其是网站部署和应用前期，通常需要以入驻的形式协助企业建立完整的网站分析工作流程。本地化作业最大程度上可以满足企业内部需求。

5.价格和费用

网站流量分析工具的价格通常包括三部分：流量费用、功能费用和服务费用。

·流量费用：大多数网站分析工具都是基于流量付费，流量规模决定了付费区间，流量越高，整体价格越高（单价其实更低）。

·功能费用：支付流量费用之后，网站分析工具大部分功能都可以使用，但某些功能模块可能需要额外付费。在选择网站分析工具时，需要确认是否所有的产品和功能特征都可用，以免后期使用时造成不必要

的麻烦。

·服务费用：服务费用通常与以上两种费用打包结算，国内目前很少按时间付费，这也是网站分析咨询服务价值低的一个体现。

网站分析工具的选择，一定要结合企业需求（包括短期需求、中期需求和长期需求）、预算、实现目标等自身情况，并综合考虑服务商的产品、服务、预期产出价值等因素综合评估。

7.4 流量采集分析系统的工作机制

完整的流量采集分析系统的工作机制包括数据采集、数据处理和数据应用三个部分，如图7-1所示。

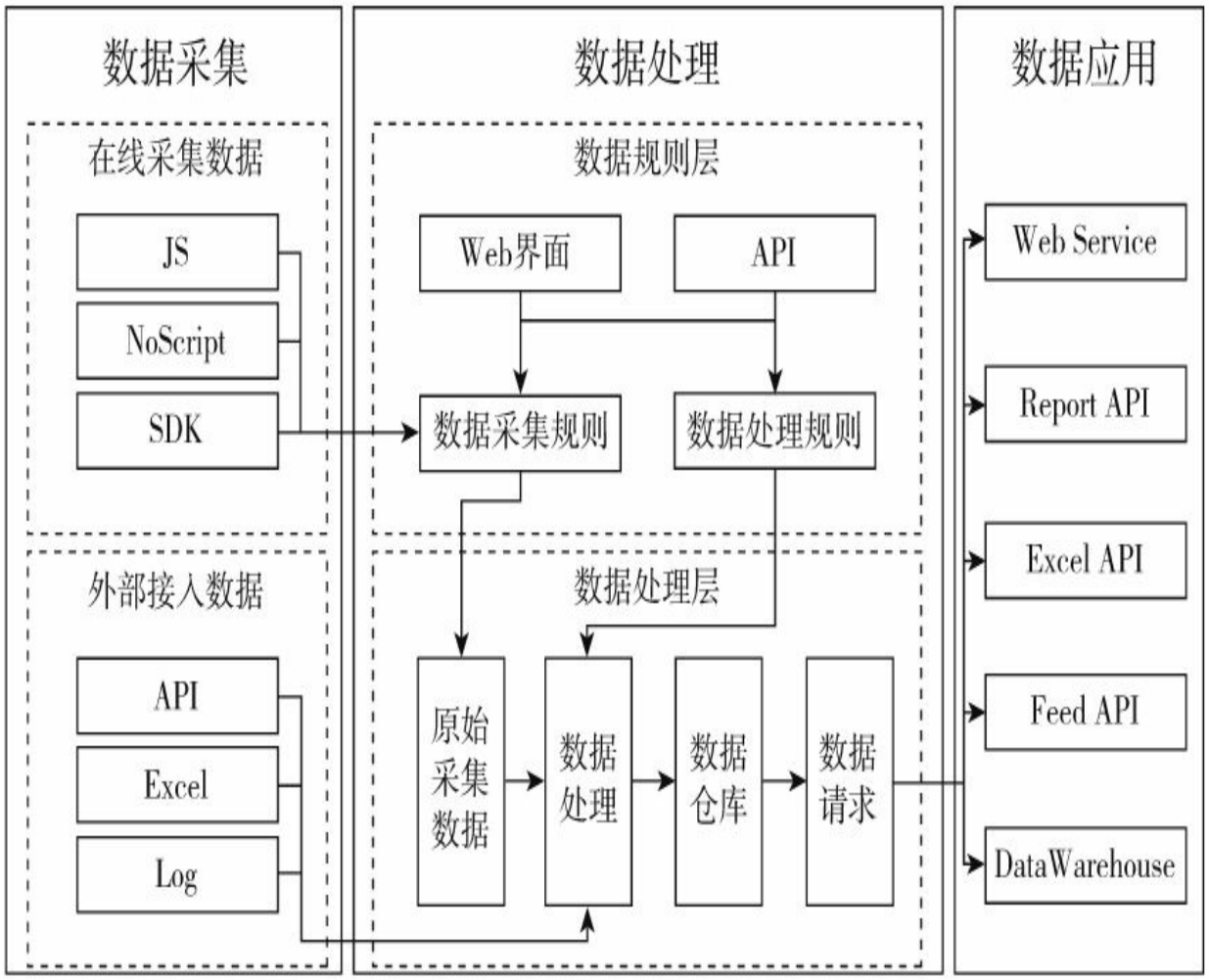


图7-1 流量采集分析系统的工作机制

·数据采集层：包括在线数据和外部数据的采集。

·数据处理层：在线数据在采集规则的约束下完成原始数据采集、处理和预运算，同时根据处理规则整合外部接入数据并做整合计算，最终可供外部调用的数据仓库数据或服务数据。

·数据应用层：根据外部特定请求以报告、数据源、数据服务、数

据API、数据仓库等形式返回结果。

7.4.1 流量数据采集

数据采集层分为两层，第一层是通过特定页面或Activity标记实现在线数据采集，在线数据是网站数据的核心组成；第二层是通过外部系统或手动形式导入的外部数据源，外部数据源是在线数据的拓展。

1.在线数据采集

在线数据采集根据平台可分为WEB站、WAP站和APP站。Web站以及由HTML5开发的WAP站都支持JS脚本采集；较早开发的不支持JS的WAP站则采用NoScript，即一个像素的硬图片实现数据跟踪；SDK是针对APP进行数据采集的特定方法和框架。这三种方法可以实现目前所有线上数据采集需求。数据采集阶段的工作流程如图7-2所示。

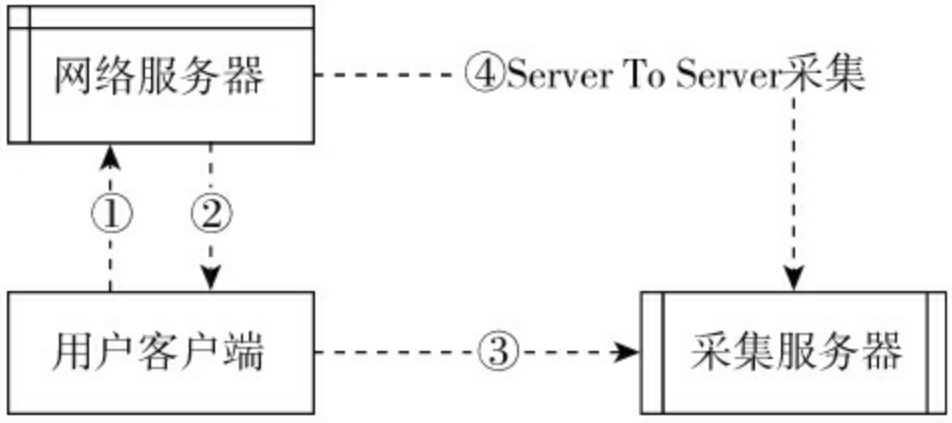


图7-2　在线数据采集流程

·当点击网站/APP时，用户客户端向网站服务器端发送请求，如①所示。

·网站服务器返回请求结果，如②所示。

·用户客户端开始加载页面，同时触发特定标记，特定标记将采集到的数据发送到网站分析系统的采集服务器处理，如③所示。

这种客户端-服务器的数据采集方法适用于大多数数据采集需求，但在这种采集方法的前期是页面标记需要在用户客户端触发才能实现，

如果数据不是通过客户客户端触发或触发时的数据在网站外部则无法收集。

比如：用户在使用在线支付的过程中，除了企业拥有结算工具意外，大多数网站都需要跳到特定网站如支付宝、网银等完成支付，而在这些支付由于存在于外部网站，无法通过页面标记形式收集支付成功数据，此时这种客户端-服务器端的采集方法失效。

另外，由于数据经历了从网站服务器→用户客户端→采集服务器三个节点，从网站服务器到用户客户端的过程可能会有数据丢失情况，尤其在于订单结算等核心信息上，这种客户端-服务器的采集方法可靠性较小。注意：不管采用何种采集方法，任何网站分析系统的数据都不可能与企业内部数据系统数据完全一致，数据不一致性存在于任何网站分析系统中，对网站分析系统数据准确性的要求是数据误差与企业数据系统误差比例较小（通常在5%以下）且数据误差稳定。

针对上述情况，某些网站分析系统如Webtrekk支持Server to Server即网站服务器对采集服务器的方法进行在线数据采集，这样就避免了数据在客户端的中转流失，如图7-2中④所示，S-S的数据采集过程如下：

1）当点击网站/APP时，用户客户端向网站服务器端发送请求，如①所示；

2）网站服务器将处理完的请求直接发送到采集服务器，而不必经过用户客户端，如④所示。

所有在线数据采集都会受到采集规则的制约，比如排除特定IP地址的流量、只采集某个域名下的数据等。数据采集规则是数据采集的重要控制节点，如果出现某些排除、隐藏或直接忽视数据的采集规则，将可能导致数据丢失。

所有SAAS网站分析系统都不能处理历史数据，这意味着，如果在数据采集阶段出现数据丢失将产生无法挽回的后果，建议原始数据采集阶段不设定任何排除规则；如果数据中可能含有大量的内部测试数据，测试环境与生产环境分账号采集。

2.外部接入数据

外部接入数据根据接入方式不同可分为API接入、Excel接入和Log接入等。

·API是主流的大批量数据集成方法，常见的数据源系统包括Baidu和Google的SEM数据、EDM数据等营销类数据，企业CRM数据等用户类数据、企业订单及销售数据等；

·Excel是临时性、小数据量的导入方式，人工通过前端界面上传实现；

·Log是原始服务器日志，部分网站分析系统日Webtrends支持混合页面标记数据和日志数据，共同作为网站分析系统数据源，支持Log的网站分析系统主要采用Local即本地服务器形式，数据直接在企业内部交换。

外部数据接入与在线数据采集是异步进行的。外部接入数据进入网站分析系统后，根据数据处理层的处理规则，在经过数据抽取、加载、转换之后，与在线采集数据整合形成完整数据源。外部接入数据必须具备一定的特征才能与在线采集数据整合，常见的特征是以某个字段作为关联主键，比如产品ID、渠道ID、用户ID、页面ID或订单ID等；也可以通过时间性特征进行数据整合处理，如按时间导入费用、站外投放数据等。

外部接入数据的工作流程大体是：原始外部数据（文档、服务器日志、在线其他系统数据、离线数据）通过自动或人工整理形成符合特定规范的数据文件，然后根据接入机制的不同完成数据整合工作，如图7-3所示。

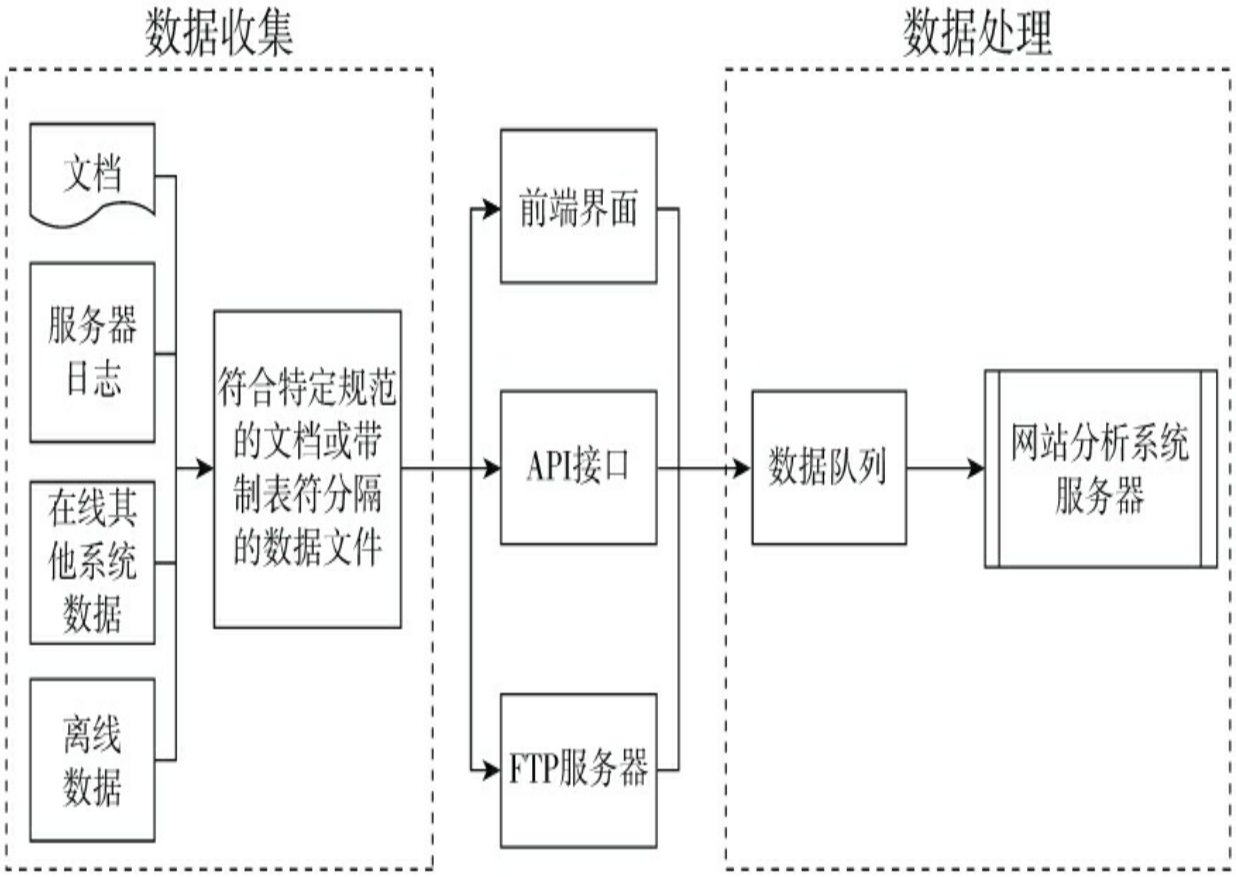


图7-3　外部接入数据工作流程

·文档类数据文件通常通过前端界面手动上传实现数据导入；

·在线其他系统数据以及离线数据通过API进入网站分析系统；

·服务器日志、在线其他系统数据以及离线数据也可以通过特定FTP服务器上传数据，具体流程为：企业内部通过程序生成特定数据文档，将稳定自动上传到网站分析系统指定的FTP服务器，网站分析系统从FTP服务器采集数据，通过验证后处理数据。

提示 理想情况下通过API接口导入外部数据是最优选择，但综合IT人力、物力和时间投入因素，通过FTP导入数据的方式却更易于实现。前期可以考虑使用FTP自动上传的机制，待数据需求稳定且业务实现思路无误后再通过技术开发API。

7.4.2 流量数据处理

在数据处理之前，由于原始在线采集数据和外部数据的都是细粒度的数据，无法无法提供支撑后期应用需求，因此需要经过特定规则处理。

1.数据规则层

不同网站分析系统的数据处理规则有所差异，网站分析系统的功能越强大，其处理规则越复杂。数据规则按照数据处理过程可分为代码部署规则、数据采集规则和数据处理规则。

（1）代码部署规则

代码部署规则是在数据采集阶段的语法规则，不同字段通过不同的语法实现。常见的收集规则包括用户类、事件类、页面浏览类、交互类、电子商务类等。

（2）数据采集规则

数据采集规则是在数据发送到服务器端时设置的只收集符合特定条件的数据，而对其他数据全部“忽略”，常见的数据采集规则是包含和排除，如只包含符合条件的数据，排除符合条件的数据；规则内容则有如下几种形式：

·特定网站内容的流量：如主机名、目录、请求URL、网页标题、着陆页地址信息；

·特定外部来源的流量：如推荐链接、社会化媒体来源、自定义来源跟踪标记（来源、媒介、位置、广告活动、内容、关键字等）；

·特定用户属性的流量：如浏览器、操作系统、设备信息、网络服务信息、操作设备（PC、WAP、APP应用）、国家、城市、地区、IP地址等；

·特定用户行为的流量：如搜索、购买、特殊事件标记、自定义用

户维度等；

（3）数据处理规则

数据处理规则是指对原始采集数据进行处理的规则要求，除满足日常系统功能需求而设定的处理逻辑以外，还有部分通过人工或API设定的特殊处理规则，这些规则综合影响最终的数据仓库数据。常见的数据处理规则可以包含上述所有的数据采集规则的内容，除此以外还包括某些特定用法，如数据提取、复制、转换、组合等。

2.数据处理层

数据处理层的处理对象分为两种，一是通用信息处理，二是特殊数据处理。

（1）通用信息处理

尽管不同网站分析系统功能有所差异，但有些功能是所有网站分析系统都具备的，这些信息在数据报告中可能涉及分析维度包括：全部来源渠道、引荐来源、搜索引擎和关键字、全部页面、进入网页、退出网页、访客地域、新老访客、时间等。涉及的指标包括UV、访问量、浏览量、停留时间、IP数、跳出数、跳出率等。

这些通过信息来源于客户端发出请求的HTTP中的标准内容，包括：发出请求的IP地址、时间戳、请求类型、请求主干、返回状态码、返回字节数、客户端信息等。如下是一段通用数据记录示例：

```
219.133.0.1 - - [17/Jan/2014:09:23:46 +0800] "GET /adobe-analytics-anomaly-
detection.html HTTP/1.1" 200 10935 "http://www.searchmarketingart.com/webtre
a-concern-commercial-web-site-analysis-
tools.html" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0;
kingsoft-ciba; .NET4.0E)"
```

（2）特殊数据处理

这部分数据是系统根据自身功能定义的数据规则信息，该信息受网站分析系统规则定义和页面代码部署双重影响。

特殊数据可能包括维度有：电子商务跟踪信息、产品信息、自定义渠道信息、站内搜索信息、用户路径信息、访问设备信息、目标转化信息、事件信息、漏斗信息、关联信息、用户细分和区段、归因模型信息、多渠道转化、异常检测信息、分组信息、媒体跟踪信息、A/B测试信息以及自定义维度信息等。可提供的指标可能包括：支持度、频次、首次转化价值、辅助转化价值、各级转化率、到达数、放弃率、完成率、交互度、访问价值、价格、数量、实例、位置值、登录注册数、排名、登入率、CTR、费用、周转率以及自定义指标等。

在经过数据处理之后，现在大多数的结果数据都会被放在“数据仓库”中，这里的数据仓库是泛指，其中包括MySQL等传统数据库，也包括Hive、HBase等大数据下常用存储。网站数据仓库是支撑高级分析需求的数据基础，初级的网站分析工具由于功能简单而无需网站数据仓库，所有数据报表都可以基于原始Log日志直接生成报表即可。

不同网站分析系统对于自身的数据仓库的数据结构定义不同。如Adobe Analytics的网站数据仓库是一个Data Feed集，拥有超过500个字段的巨型表；Webtrekk和Webtrends的网站数据仓库是一个结构化、雪花型数据仓库，含有超过20个关系表共同组成点击流数据仓库。

7.4.3 流量数据应用

数据应用层是网站数据输出的出口，常见用的请求主体有：Web Service、Report API、Excel API、Feed API、DataWarehouse。

·Web Service：SAAS模式的网站分析系统都是通过在线访问进入系统，所有在线访问产生的数据请求都可以归为Web Service，包括数据报告的下载、下钻、筛选、展现、上卷、更新、删除、新增等功能操作和分析操作。

·Report API：部分网站分析系统支持通过API调用数据报告，并集成到其他系统。

·Excel API：部分网站分析工具提供Excel插件，通过Excel实现数据查询、导出等操作。

·Feed API：Data Feed只在高端网站分析工具中才提供，DataFeed是结构化的原始网站数据的集合，也可以看成是结构化后的网站行为日志，Data Feed常用来与企业数据仓库（EDW）做数据整合使用。

·DataWarehouse：部分高端网站分析工具提供数据仓库导出接口，可直接通过数据仓库构成完整的点击流数据，这种方式更利于企业数据仓库的实现。

7.5 流量数据与企业数据的整合

网站数据作为企业数据的重要组成部分，记录了大量的客户和潜在客户的所有网站行为信息。网站数据的巨大价值是所有用户（即使还只是潜在用户）的行为都是可跟踪、可回溯、可量化、可分析的，并且分析结果可以直接应用到相关业务节点，这直接弥补了传统企业数据局限于已经完成特定转化如付款、交易之后的数据短板，使得企业的业务对象的所有行为形成数据闭环，可以建立基于完整闭环的业务认知。

7.5.1 流量数据整合的意义

（1）提高决策层的决策效率

基于整合后的统一数据源，很容易提供统一的数据可供决策；数据由于减少了不同系统、不同产品、不同报表甚至不同指标间的相互转换，节省了大量中间环节而提高了数据的实时性；基于整合数据，所有业务信息流前后贯穿，业务间的相互关系及对关键目标的作用一目了然，无论是基于目标的KPI考核还是基于过程的评估都可以做到有的放矢。

（2）深化业务层的商业洞察

企业数据整合前，业务部门在开展数据相关工作时工作效率低、反馈效果差，表现在所需报表及数据难以获取、报表制作过于烦琐、现有工具和认识难以深层次分析业务异常原因、不同部门间的数据结果难以有效共享等。因此，业务部门往往耗费大量时间在数据提取、整理、汇总和制作上，缺乏时间做更深入的数据洞察和价值提炼工作；即使得出结论又由于不同部门间缺乏统一的度量和定义而无法直接共享，仍然需要重新梳理数据。将数据整合后便可节省前期数据预处理阶段的大部分工作，而将重点放在数据探索和深入分析上。

（3）降低IT的数据维护成本

随着企业业务的不断发展及业务人员数据意识的提高，基于数据的采集、存储、应用需求不断调整，分散式的数据需求难以应付；数据报表的容量不断增加，对IT的系统性能（I/O、读表速度和效率、响应时间等）、维护成本（升级、备份、更新等）等要求不断提高，客观上也提高了IT运维成本。将多数据源整合后可有效利用企业资源的优势，将多个应用和需求做弹性管理，便于最大化资源的利用率，并降低重复占用资源的情况。

7.5.2 流量数据整合的范畴

流量数据整合的范畴指的是整合的数据范围，从数据在企业中不同的支持作用来看，数据整合范畴主要是将流量数据与业务数据和IT数据整合。

业务数据整合的目的是将所有围绕公司业务上下游的数据整合到一起，形成完整的业务流数据体系。这个过程中涉及网站营销数据、网站流量数据、线上支付数据、线下物流数据等；除此以外还可能包括网站运营数据、企业销售数据、线下会员数据、呼叫中心数据、仓储数据等。

IT数据整合的意义是利用IT数据拓展网站分析工具（尤其是SAAS模式网站分析工具）所缺乏的数据维度和指标。IT主要整合的数据是网站日志以及基于现有网站结构的拓展数据。例如页面结构、页面状态、服务器响应时间等。

7.5.3 流量数据整合的方法

流量数据整合可分为在线数据整合和本地数据整合，在线数据整合常用于部门或业务线数据应用，本地数据整合则是企业级数据应用。

（1）在线数据整合

在线数据整合是指借助现有的在线数据工具，整合其他所有的数据源到线上。网站分析工具由于已经具备网站流量相关数据，只需将外部数据整合到网站分析工具中即可，因此更多被应用为在线数据整合平台。

在线数据整合的最佳数据源包括营销数据、会员数据、运营数据和外部环境数据四类。

1）营销数据：网站分析工具通过插码标记来识别不同的推广渠道，不同的渠道通过代码区分。渠道代码就是营销数据关联的主键，通过整合的营销数据包括推广渠道分组、营销费用、营销媒介信息、投放时间等。

2）会员数据：大多数网站都有登录注册系统，当用户发生登录或注册行为后，可以记录该用户唯一识别标识（如用户ID），通过该标识可以把会员或CRM数据传到在线网站分析系统进行整合。

3）运营数据：网站运营数据整合涉及非常多的数据信息，包括以产品品类、品牌、参数、尺寸、颜色等，订单状态、订单来源、配送地域、配送用户信息、使用优惠券等），促销活动ID、时间、应用品类、限制金额、发放优惠券类型、促销费用等，站内资源位ID、页面、位置、排期、对应内容、轮播次数）等。

4）外部环境数据：外部环境数据是指通过一定关联特征（如时间）将外部客观环境的数据整合到网站分析工具中。这些外部数据是业务认定的对网站关键目标影响较大的因素，如外部搜索引擎收录数据、天气数据等。

（2）本地数据整合

本地数据整合是指将所有的数据整合到企业内部，形成供企业所有部门应用的企业数据仓库。本地数据整合相较于在线数据整合，它在原始业务节点、初始汇总业务流基础上形成全面的业务数据流，这是企业数据整合的最终阶段。

企业级数据集成大多基于本地实现，一方面企业中更多有价值、更巨量的数据产生于线下，例如交易、付款等；另一方面出于安全性的考虑，数据在企业内部通常更安全。

如果是自建流量分析系统，那么用户行为日志本身就在企业内部；如果是采用第三方免费或付费的流量分析系统，那么需要流量采集工具支持数据回传，并且最好是用户日志明细，数据粒度越细越能满足多样化的数据建模、挖掘和分析需求。

在第三方数据工具中，Adobe Analtyics和Webtrekk等除了能提供多种访问接口外，还支持导出原始用户行为日志到指定服务器，这为基于原始日志的解析和二次开发提供了便利条件。

7.6 流量数据化运营指标

流量数据化运营指标涵盖站外营销推广指标、网站流量数量指标和网站流量质量指标。

7.6.1 站外营销推广指标

（1）曝光量

广告曝光量是指广告在站外对用户展示的次数，广告曝光量又称广告展示量。从技术上来讲，广告曝光指的是特定广告跟踪代码被加载的次数，每加载一次就产生一次广告曝光。

广告曝光是衡量广告效果的初级指标，通常用来衡量展示类广告。广告曝光并不意味着广告一定会被用户看到，而是意味着广告被加载并展示出来，广告位置（首屏还是底部）、广告素材、广告形式及广告周边因素等因素都会影响到用户的注意力。

（2）点击量

广告点击量是指站外广告被用户点击的次数，每点击一次就记录一次。某些广告投放系统如蜂巢和Adwords等会对无效点击进行过滤，即当用户恶意点击广告时，恶意点击的部分会被“忽视”，而只保留系统认为的正常点击数据。这会导致站外广告系统出现少量有点击而无记录的情况，影响站内网站分析工具与站外广告监测监测数据的一致性。

站外广告监测系统与网站分析工具监测到的广告点击量通常不一致，原因除了上述无效点击过滤外，还包括用户点击后到达的遗漏监测、系统监测和判断逻辑、数据定义规则、数据发送丢失等因素。

（3）点击率

广告点击率也称广告点击通过率，常用CTR表示。点击率的计算公式为：点击率=点击量/曝光量

点击率是衡量站外广告效果的重要指标，它反映了用户对当前广告的喜好程度，也反映了所投放的媒介用户质量与投放广告的匹配度。通常，点击率通常越高越好，但过高的点击率可能也意味着点击作弊。

（4）CPM

CPM即Cost Per Mille，每千人成本。CPM是广告典型的付费方式之一，按照每千次展现付费。如一个广告展现了10000次，约定CPM为30元，那么对于该广告应该付费300元。

（5）CPD

CPD即Cost Per Day，按天展示成本。传统广告媒介尤其是门户广告普遍采用的费用结算方式，只根据展示的时间付费，对于期间任何广告效果（如展示、点击、目标转化等）不作任何承诺。与CPD类似的还有按周、月、季度付费购买，付费逻辑相同，都是根据固定周期付费。

（6）CPC

CPC即Cost Per Click，每次点击成本。CPC广告是部分展示类广告、SEM广告的主流投放形式，企业只需要按照点击的次数付费。CPC是衡量广告单位成本支出的重要指标。

（7）CPA

CPA即Cost Per Action，每次行动付费，通常会将行动定义为网站特定的转化目标，如下载、试用、填写表单、观看视频等，然后按照转化目标的数量付费。

（8）每UV成本

每UV成本指点击站外广告到达网站后，每个UV的成本。UV成本较为真实地反映了到底有多少“人”到达网站，因此是广告类部门的重点评估指标之一。计算公式为：每UV成本=广告费用/UV

（9）每访问成本

每访问成本指点击站外广告到达网站后，每个访问的成本。相对于每UV成本，每访问成本中增加了“频次”的考核，也是广告类部门的重点评估指标之一。计算公式为：每访问成本=广告费用/访问量

（10）ROI

ROI即投资回报率，指投入费用所能带来的收益比例，计算公式有

两种：ROI=利润/费用或ROI=成交金额/费用。ROI是广告部门评估投入产出效果的核心指标之一。

在大多数电商企业中，ROI计算都使用第二种公式，原因是电商企业的利润大多为负数，因此ROI更多评估的是每单位费用带来的销售额。

（11）每点击/UV/访问/目标转化收益

与成本相对的指标是收益，根据不同的收益单位可分为以下几种：

·每次点击收益。每次点击收益指每次站外广告点击能获得的转化收益，通常将转化定义为电子商务交易收入，即订单金额。该指标与CPC相对应。

·每UV收益指点击站外广告到达网站后，每个UV产生的转化收益。UV收益反映了每个“人”能带来多少订单收益，计算公式为：每UV收益=广告总收益/UV量。该指标与每UV成本相对应。

·每访问收益。每访问收益指点击站外广告到达网站后，每个访问产生的转化收益。相对于每UV收益，每访问收益中增加了“频次”的考核，反应的是每人次收益结果。计算公式为：每访问收益=广告总收益/访问量。该指标与每访问成本相对应。

·每次目标转化收益。对网站内的每个目标，通常会定义一个目标转化值。如根据业务经验，每一次下载会产生50元最终转化收入，那么可以将目标转化收益设定为50元。该指标与CPA相对应。

7.6.2 网站流量数量指标

（1）到达率

到达率指用户从站外广告点击后到达网站的情况，技术上的定义为用户从带有站外标记的链接点击进入网站后，触发站内跟踪代码的次数，因此到达数据仅发生在针对站外标记广告的落地页。

到达率用来衡量站外流量到达网站的比例，计算公式为：到达率=到达量/点击量

到达率指标越高，说明在广告点击与网站到达之间的流失越少。不同广告资源的到达率情况有所差异：通常广告类到达率较低，平均在50%～80%之间；SEM类到达率较高，在80%以上。

过低的到达率可能意味着用户质量较差或网站着陆页加载较慢，导致用户在页面完全打开前直接退出或数据无法正确统计。

（2）UV

UV即Unique Visitor，又称独立访客。UV根据定义时间的不同可分为每小时UV、每日UV、每周UV、每月UV等。每小时UV定义为用户在一小时内无论进入网站多少次或打开多少页面，都只计算为1，其他UV计算方法类似。

UV是衡量“人数”的重要指标，反映了来到网站的用户“数量”。UV定义只跟时间有关，与其他任何行为都没有关系。

（3）Visit

Visit又称访问量、访问次数或会话次数。Visit定义与UV类似，只不过大多数Visit的默认定义时间为30分钟，即用户在30分钟内重复打开网站，Visit只计为1；若超过30分钟，访问则记为一次新的访问。因此一个UV可以产生多个visit。

Visit是衡量次数的重要指标，反映了有多个“人次”来到网站，访问次数和独立访客一起可以评估网站来了多少“人”，同时黏性如何。如一个网站每天的UV是100万，但访问数是300万，反映了网站平均每个UV可以带来3次访问。

（4）PV

PV即Page View，又称页面浏览量、页面曝光量，PV与站外推广类指标中的曝光量定义相同，区别在于PV只用来衡量站内页面的曝光量。1个visit下可以产生多个PV。

（5）新访问占比

新访问指所有访问中新用户的占比，这里的新用户指的是之前没有任何访问记录的用户。没有访问记录的原因可能是用户没有来过网站，也可能是之前来访的Cookie信息被删除、更改。

新访问占比用来定义所有的访问中新访问的占比情况，反映了站外渠道或网站吸引新用户的能力，因此是站外广告投放效果评估的重要指标，尤其对于以吸引新用户关注为目的的渠道具有重要意义。计算公式为：新访问占比=新访问量/总访问量

如果用户在当天既产生第一次访问，又产生第二次访问，网站分析系统会认为该用户既属于新访问又属于老访问，并在计算网站新老访问量时分别加1。

（6）实例数

实例数（instance）是一个特殊的流量指标，用来衡量站内自定义对象的触发次数。实例数的计数原理是每次监测对象的代码触发一次，则实例数加1。

实例数通常用来统计站内自定义对象，如某个按钮、某个下拉菜单、某个功能区等；实例数在统计逻辑上类似于页面浏览量。理论上，页面级别的页面浏览量与页面实例数相等。

7.6.3 网站流量质量指标

（1）访问深度

访问深度又称人均页面浏览量、每次访问平均页面浏览量，用来评估平均每个访问内用户看了多少个页面。计算公式为：

访问深度=PV/访问量

访问深度是用户访问质量的重要指标，访问深度越大意味着用户对网站内容越感兴趣；但访问深度并不是越高越好，过高的访问深度可能意味着用户在网站中迷失方向而找不到目标内容。另外，在某些场景下，也会使用PV/UV来计算访问深度。

（2）停留时间

停留时间指用户在网站或页面的停留时间的长短。计算公式为：

·网站停留时间：最后一次请求时间戳-第一次请求时间戳。

·页面停留时间：下一个页面请求时间戳-当前页面时间戳。

停留时间的计算逻辑是2个时间戳的差值，在某些情况下这种计算方法将失效，如退出页面或跳出页面由于没有下一个时间戳而无法计算停留时间。针对这个问题，不同工具对于停留时间的处理有差异：

·估计法：有的工具提出“心跳监测”的方法，即每隔一段时间（通常是30秒）页面向服务器发送请求。如果用户在当前页面离开网站，在计算该页面停留时间时使用当前页面最后一次请求的时间作为最后时间戳来计算。

·填充法：有的工具则使用固定时间，例如30秒作为该页面的停留时间。

·舍弃法：有的工具将其作为退出或跳出页面的实例直接舍弃而不作统计。

通过上述算法可以看出，停留时间并不意味着用户真的“停留”在页面上浏览网页，用户可能打开网页后离开计算机、或者使用多TAB浏览器同时打开多个页面。

另外，对于停留时间的评估也不是越高越好。一个简单的页面，如果用户停留时间过长，可能意味着用户没有注意到页面关键信息或没有注意到引导按钮，从而降低用户体验或降低该页面的引导贡献。

（3）跳出/跳出率

跳出指用户在到达落地页之后没有点击第二个页面即离开网站的情况，跳出率指将落地页作为第一个进入页面的访问中直接跳出的访问比例。计算公式为：跳出率=跳出的访问/落地页访问。对于跳出的定义，很多工具的定义逻辑有差异，跳出的核心定义是用户在这个页面上没有其他动作，此时有几种情况：

·没有任何其他点击操作，仅仅是加载了1次页面或产生了1个PV而已。

·没有任何其他点击操作，但是却刷新了该页面，即产生了2个PV，但仍然只在落地页产生。

·仍然是在落地页上，但是却发生了特定的单页面行为，例如事件、点击等操作。

上述几类情况都可以作为跳出本身的认定因素，具体以网站分析系统的定义为准。

跳出是仅针对落地页发生的指标，用来评估用户进入网站后的第一反应情况。过高的跳出率意味着站外流量质量低或页面设计出现问题，导致用户不愿继续浏览网站。

（4）退出/退出率

退出指的是用户从网站上离开而没有进一步动作的行为。退出率指在某个页面退出的访问占该页面总访问的比例，计算公式为：

退出率=页面退出的访问/退出页面访问

退出与跳出的区别有两方面：一是跳出针对的是落地页，退出针对网站所有页面，因此只有落地页才有跳出率，但全站所有页面都有退出率（都存在成为离开网站出口的几率）；二是二者的分母不同，跳出率的分母是将落地页作为登录页（第一个进入页面）的访问量，退出率的分母则是该页面的总访问量（包含作为落地页和非落地页的访问量）。

在通常情况下，页面退出率越低越好，但某些特殊的页面出现高退出率也属于正常情况。如页面用来解决用户的某个问题，当用户需求得到满足而退出网站属于正常情况。

（5）产品页转化率

在大多数情况下，用户要完成订单需要先浏览产品页查看相关产品信息，确认信息之后才能继续购物车流程，因此浏览产品页会成为用户订单转化过程中的重要指标。产品页转化率计算公式为：

产品页转化率=产品页访问量/总访问量或产品页转化率=产品页UV/总UV

关于产品页转化率的计算既可以使用访问量，又可以使用UV，不同公司根据实际场景应用。

用户将商品加入购物车的来源途径通常包括活动页、促销页、产品列表页、超市页、会员中心等具有直接加入购物车功能的页面。在大多数情况下购物车商品来源集中在产品详情页，读者可根据企业自身情况加以区别。

（6）加入购物车转化率

加入购物车是用户进入购物车环节的第一步，用户在该步骤确认商品信息、数量等。加入购物车转化率计算公式为：

加入购物车转化率=加入购物车访问量/总访问量或加入购物车UV/总UV

加入购物车转化率比产品页转化率具有更高的参考意义，该动作意味着用户更加具有购物导向性，因此，该指标会用来衡量所有站外营销和站内运营的业务效果。加入购物车转化率高，意味着具有购物意向的用户比例高（作弊情况除外）。

（7）结算转化率

结算是用户购物车环节的第二步，用户在该步骤确认订单联系人、送货地址、送货时间、运费、优惠折扣等信息。结算转化率计算公式为：

结算转化率=结算访问量/总访问量或结算转化率=结算UV/总UV

结算转化率越高意味着用户完成订单的概率越大，因此也是网站相关业务部门的重要参考指标。

（8）下载转化率

很多网站都有资料可供用户下载，下载意味着用户具有更强的目标性或针对性。下载转化率计算公式为：

下载转化率=下载访问量/总访问量或下载转化率=下载UV/总UV

如果网站存在很多可供用户下载的业务目标，还可以综合评估每个用户的下载情况，通过每用户下载数量=下载量/访问量或每用户下载数量=下载量/UV来计算。

（9）注册转化率

注册转化率是以会员获取为目的的网站最常定义的目标，注册转化率计算公式为：

注册转化率=注册会员量/总UV数

由于注册转化率计算是以“人”为单位，因此这里分母使用UV而不是Visit。

（10）购物车内转化率

购物车内转化率与其他指标的定义维度都不同，该指标用来衡量加入购物车的用户中完成订单的比例情况。计算公式为：

购物车内转化率=提交订单的访问量/加入购物车的访问量或购物车内转化率

=提交订单的UV/加入购物车的UV

当用户将商品加入购物车时，意味着用户具有较强的购买意愿，如果用户中途放弃购物，则产生购物车放弃率，计算公式为：

购物车放弃率=1-购物车内转化率

购物车内转化率是所有销售类电子商务网站的重要监控指标。大多数电子商务网站的购物车内转化率在60%以上，如果低于这个数据说明流量可能存在作弊问题，或购物车流程设计问题（某些购买决策周期长的特殊商品除外，如保险类商品）。

除了以上目标外，企业还可能定义的转化目标包括产品收藏、商品评价、商品咨询、降价通知等用户行为，这些都是用户转化过程中的重要节点。

7.7 流量数据化运营应用场景

对于流量运营而言，流量采购和流量分发是运营的核心。

7.7.1 流量采购

流量采购指通过多种媒介和广告渠道采集或购买流量，从而实现流量目标。流量采购是各个公司非常重要的工作，这不仅是因为流量决定了后端所有运营的规模，更是因为流量采购的预算通常会占据公司整体预算中非常大的比例。所以企业都希望花出去的钱能产生最大价值，即以尽量低的单位成本获得目标规模的流量，并尽量保证流量质量。

围绕流量采购而产生的数据支持，有一个细分的数据工作领域几乎就是围绕该内容产生的——网站数据分析。网站数据分析的对象是流量，而流量的产生大多依赖于广告渠道或推广媒介，因此在很多公司内部网站数据分析都在广告中心或营销部门之下，目标就是更好地为流量采购服务。

除了采购流量外，企业当然也会包括自然流量，例如SEO、直接输入、社交媒体引荐、友情链接等，但在大多数企业来看，这些流量是相对稳定且没有大规模爆发能力的。在企业需要做大型促销时候或搞“大事件”时都需要依赖于外部采购。这种思路其实适用于95%以上的企业，包括京东这种体量的公司每年也花费巨额广告费在流量采买上。

在流量采购方面，数据主要支撑以下四方面内容：

（1）流量预测

流量预测是对未来的预估和推断，常被应用在业务执行前的计划和评估阶段。效果预测可以帮助业务建立合理的预期目标，并为实现目标建立资源需求图谱。常见的流量预测的应用示例：

·未来1个月SEO的流量预期能达到多少？

·如果有100万预算，SEM渠道能带来多少收入？

·在新浪首页投放一个banner，预计能带来多少访问量？

（2）效果评估

广告效果评估是已经发生的过去做出结果判断，以评估结果是否符合预期或存在异常情况。效果评估常应用的到业务状态进行时和业务状态完成后。业务状态进行时的结论定义可快速帮助业务建立实时数据反馈机制，通过即时的数据结果判断是否符合预期，并可通过措施优化当前业务状态；业务状态完成后的结论定义除了可以做业务效果评估外，还为原因解析和数据探究提供方向。常见的广告效果评估应用示例：

·今天刚上线的重定向媒体X效果如何？

·昨日广告渠道C的跳出率增加了45%，正常吗？

（3）效果分析

广告效果分析指对广告数据进行探索和研究以便发现进一步的数据观点和数据洞察，它是挖掘数据深层次原因和关系的关键动作，也是数据论证的主要过程，表现在数据结果中大多是数据论证过程。常见的广告效果分析应用示例：

·为什么公司投放了150万在暴风影音的弹窗，但实际上暴风的转化率只有0.01%？

·为什么品牌专区平日的ROI可以达到109%但在活动期间只有88%？

·为什么某大V的一篇转发文章能带来12万的访问量？

（4）作弊检测

流量作弊几乎伴随着广告行业一起“发展壮大”，流量作弊几乎已经成为广告行业的潜规则。对于广告主来讲，自然不希望花出去的钱买到的是一堆垃圾流量，因此对于广告流量的作弊检测是营销部门的重要工作。

流量作弊主要体现在以流量规模为主要衡量指标的媒体上，例如CPC、CPS、CPA等，这些媒体的普遍特点是按点击或转化收费，因此广告平台/广告代理/广告媒体自然希望能有更多的点击或转化产生，这样能产生更多的收益。

在反作弊的检测方面，数据能基于大多数普通用户的行为模式，对所有流量进行监控和预警，同时基于预警信息能为业务运营提供及时信息支持，帮助业务部门派出异常。

对于流量作弊检测结果，需要读者选择恰当的时间和方式给恰当的对象反馈。因为，大多数的广告部门的负责人对于广告投放直接责任，过多垃圾流量可能意味着工作失责；更重要的是，很多负责人其实了解其中实情，但限于某些原因而跟广告媒体达成“默契”。

7.7.2 流量分发

流量分发指如何对流量进行内部分配，通常这部分工作由网站运营中心完成。当流量通过广告或营销中心采购到达网站后，运营中心便通过四种方式进行流量分发：

（1）内部广告

内部广告是平台型网站流量分发和变现的主要途径之一。平台网站内部不同的店铺通过多种形式参与广告流量的竞价和购买，这种情况跟广告中心去外部广告媒体采购流量类似。只不过不同的是，已经到达网站的流量由于已经是具有一定兴趣的用户，因此购买转化率较高。

（2）活动引导

促销活动是各个网站的主要运营内容之一，要做促销就必然要有促销策略。促销策略的重要一环是如何设计流量引导的问题。流量引导涉及流量的分布、利润的倾斜、品类支持、标杆策划、走量品类转化等多个主题。

（3）自然引导

自然引导是网站设计时的自然链接，通常基于网站目标、产品策划和用户体验产生。自然引导的核心是能够将与当前页面主题最相关的其他信息连接起来，形成一张能够无限连接的网络，用户在网络上的每个节点都能快速找到兴趣信息。

（4）个性化推荐

严格意义上来讲，个性化推荐的初衷不是为了流量分发，而是通过分析用户的行为提高网站浏览体验和购物体验，并最终实现网站目标（例如浏览时间增加、访问深度增加、每次购买金额增加等）。但个性化推荐的实现，也需要依靠在特定页面展示推荐信息，每个推荐信息栏位都是一个流量入口，因此也是一种流量分发途径。

数据辅助于流量分发，可以支持的数据主题示例如下：

·首页焦点图上的每个banner，从第一个位置到最后一个位置的点击率递减规律是怎样的？

·为什么最后一屏的点击率要高于中间屏？

·品类热度对入口的位置依赖有哪些显著性关系？

·如何综合品类关注度、位置热度实现最大化点击量目标？

·内部广告与点击落地页的匹配度如何？

7.8 流量数据化运营分析模型

本章的分析模型围绕流量数据化运营展开，主要包括：流量波动检测、渠道特征聚类、广告整合传播模型、流量预测模型。

7.8.1 流量波动检测

在广告流量结构中，有几类流量是相对稳定并且效果较好的，例如导航类流量（例如360导航）、品牌专区流量（例如百度品牌区）、品牌关键字（例如百度品牌关键字）、SEO流量（例如百度SEO流量）。这些流量虽然会受到企业广告预算的影响，但一般情况下只要广告预算足够，企业是不会主动撤销对于这几类流量渠道的费用支持。

对这几类广告渠道可以通过广告流量波动模型进行监测，该模型可以对具有相对稳定或具有一定时间规律特征的数据做检测分析。在之前的异常检测类模型中，我们提到了监督式和非监督式两种，这里介绍一种相对传统的基于时间序列的异常检测方法。

基于时间序列的异常检测方法与其他异常检测方法最显著的特征是数据之间具有明显的时间先后次序，并且每个数据都有时间维度且按时间排列。与时间序列分析类似，在做检测应用时的整体流程都需要对时间周期数据做检验、差分并进行拟合，不同之处在于预测的结果数据中我们可以定义上下限的置信区间，如果真实值超出置信区间那么就意味着数据波动异常。

下面将结合4.6节中的步骤实现该方法：

步骤1　数据读取和预处理，主要是将字符串转换为时间格式。

步骤2　数据稳定性、白噪声检验和预处理。

步骤3　时间ARIMA或ARMA对时间序列数据拟合，找到最佳PDQ或QP参数值以及对应fit（训练时）的最佳模型结果对象。

以上三个步骤跟4.6节中代码的流程一致，下面开始定义和输出置信区间。

步骤4　基于最佳模型结果对象选择应用forecast方法做预测（而不是predict方法），并设置如下关键参数：

·steps：整数型，要预测的时间序列点之外的数据数，例如设置

step=6的效果与predict方法中设置predict（start='1991-07-28'，end='1991-08-02'）的周期是相同的。

·alpha：浮点型，设置具体置信区间范围，置信区间值设置为（1-alpha）%，例如设置alpha=0.05会计算在95%置信区间下的范围值。

例如使用forecast（steps=6，alpha=0.05）后返回的结果如下：

```
(array([ 183.03624893,  124.61319468,  134.67763687,  143.22815918,
         111.08688519,  113.70161409]),
 array([ 40.80850407,  43.94083939,  46.60465652,  50.11657005,
         50.13881589,  50.13929372]),
 array([[ 103.0530507 ,  263.01944716],
        [  38.49073202,  210.73565733],
        [  43.33418858,  226.02108516],
        [  45.00148685,  241.45483152],
        [  12.81661182,  209.35715855],
        [  15.43040419,  211.97282398]]))
```

结果包括三个数组：

·第一个数组是预测值，跟使用predict方法得到的结果相同。

·第二个数组是预测值的标准差。

·第三个数组是预测值的置信区间的上下限，是一个二维数组。

基于第三个数组可以定义出正常波动范围的上下限，如果超出该范围则可以认定为异常波动。

除了可以应用到广告流量的异常波动检测外，该模型还可以应用到流量运营中的网站重点内容的检测，例如首页、帮助中心、购物车流程页等，这些页面通常相对来讲从流量来源结构、用户访问特征等方面的特征相对稳定，也可以做流量波动性检测。

7.8.2 渠道特征聚类

当企业投放众多广告媒体时，第一次对如此众多的媒体多特征分析可能无从下手。此时可以考虑对广告渠道特征进行聚类，然后从几类具有比较显著的群体上再深入挖掘。

以几乎所有企业都会投放的SEM渠道为例，账户内的关键字拥有上千个长尾词是常态，大型企业过百万的关键字更是“家常便饭”，如何针对海量关键字效果做分析是一个难点。以聚类方法为例，首先可以使用聚类方法将所有的关键字的属性、操作和效果划分为多个群组。其中：

·属性：账户结构、质量度等。

·操作：预算、价格、黑名单、地域、匹配方式、时段、展示方式、匹配的创意、平台等。

·效果：SEM排名、点击价格等SEM指标，站外广告曝光、点击以及站内流量数量和转化类指标，具体参见7.4节的内容。

然后，基于划分的群组分析不同群组间的显著性特征，从中找到可以进一步分析和优化的方向。例如：

·某一类关键字的排名较差、质量度低、流量低、转化差，这些可能需要重新规划关键字投放策略；

·某一类关键字的排名好、质量度高，但是流量低，这些可能需要重点优化展示和创意的吸引度，以获得用户的关注和点击；

·某一类关键字的排名好、质量度高、流量高，但是转化差，这些关键字需要重点从着陆页开始做分析，将转化流程和步骤层层拆分，找到流失和转化的关键节点。

有关聚类分析的更多内容，请具体参见4.1节的内容。

7.8.3 广告整合传播模型

广告整合传播指所有企业的广告和传播活动都以统一的策略作为指导，通过一定方式的组合来实现传播效果的最大化目标。广告整合传播的概念很早就已经出现，跟这个概念类似的另一个概念是整合营销传播。但整合营销传播涵盖的内容几乎涉及企业经营的方方面面，范围太大，因此这里我们只讨论其中的广告整合传播的内容。

广告整合传播的出现主要基于两方面背景：

·当前的广告媒体以及用户接触信息的渠道非常多，导致没有一种广告渠道可以完全覆盖所有用户群体，因此媒体碎片化现象非常严重。企业要想覆盖尽量多的用户，只能选择更多的广告媒体一起投放。

·营销公司（尤其是4A公司）发现，不是所有的广告渠道都对于企业广告传播具有相同的作用和贡献，基于不同渠道的贡献情况需要在组合时使用一定的组合策略和方法。

如何选择广告媒体以及如何组织不同广告媒体的传播策略是广告整合传播关注的问题。实际上该问题在数据化运营时代之前已经开始研究，但当时受限于数据样本、技术等问题，只能通过抽样调查的方式开展，因此结果的参考性不大。

当前，通过数据量化的方式做广告整合传播模型分析，主要涉及三个数据分析方法的组合：

（1）广告来源路径

广告来源路径可以提供不同路径所产生的转化数量、转化价值、平均需要时间以及转化步长等。如图7-4所示的报告来自于Webtrekk（其他网站分析工具也提供类似的报告），报告中的每一条广媒体路径都是在转化前提下形成的路径。

序号	广告媒体路径	转化量	转化价值	平均转化天数*
1	Facebook	2326	11806.10	0.00
2	Facebook▶Facebook	454	2282.70	1.69
3	Direct	368	1831.40	0.00
4	Facebook▶Facebook▶Facebook	140	710.70	2.13
5	google	126	612.50	0.00
6	SEO	111	546.00	0.00
7	Facebook▶Facebook▶Facebook▶Facebook	40	205.80	0.65
8	SEO▶SEO	25	125.80	1.28
9	Direct▶Direct	23	134.60	12.22
10	Facebook▶Direct	21	103.40	0.76
	总计	3978	20112.50	0.54

页面 1 之于 14 行 10 显示136中从1 - 10

图7-4　用户广告来源路径

以序号10所代表的用户广告媒体路径为例，用户先通过Facebook后通过Direct进入网站并完成转化的数量是21次，价值是103.4，平均需要的转化时间是0.76天，总转化步长是2（两步）。

（2）目标转化归因

目标转化归因能够解决在不同的归因模式下，所有参与转化的广告媒体对于目标的贡献情况。我们在4.7节中已经详细介绍了在不同的归因模型下，不同渠道的贡献。很多网站分析工具可以提供多种可选归因模型。

如图7-5所示，是Webtrekk提供的多重归因模型，该模型可以对于根据位置综合归因，权重分配通常是为最初进入渠道和最末进入渠道订单贡献较大，其他渠道贡献较弱。如图中默认第一和末端渠道权重分别为30%和40%，其他渠道平均都是10%。

图7-5　Webtrekk的多重归因模型

在归因的价值度量上，可以选择使用多种度量指标，例如转化数量、转化价值等。具体以定义的转化目标为依据。通常情况下，电子商务转化中，对于订单类的贡献以订单量为衡量指标；对于非电子商务类转化，以目标完成次数为衡量指标（例如阅读数、提交次数、线索数量等）。

（3）广告渠道的关联访问

我们在4.4节中提到过，关联分析不仅可以用来做购物篮分析更可以扩展到用户访问行为、搜索行为等多种模式的分析。将关联分析应用到广告渠道的模式探索，是对用户广告来源路径的进一步深化。

在对广告来源路径的研究中，我们已经知道每一条用户转化路径以及包含的广告渠道，但是这种分析方法仍然有2个问题没有解决：

·该路径已转化为触发点，没有转化就没有路径，这会使得那些侧

重于曝光的信息无法产生路径信息，更无法测量其对网站的贡献意义，哪怕仅仅是流量贡献。

·在大多数情况下，转化路径都会产生长尾效应，即大多数的转化会集中在成百上千个各式各样的转化路径中，仅凭观察无法从所有路径中提取出关于渠道组合的有效规律。

将关联分析应用到广告渠道的关联访问，恰好可以解决上述2个问题。图7-6所示截图为Webtrekk的渠道关联访问报告。

在报表中，我们可以找到特定广告媒体之间的相互关联关系。这种关联关系跟转化无关，只与用户的先后访问行为和模式有关。例如，序号1代表的关联模式意味着用户先通过Facebook再通过Daily Banner访问网站的数量是21，支持度是0.76，提升度是1.03。其中数量对应关联分析结果中的实例数。

序号	Advertising Media in Lifecycle	数量	支持度	提升度
1	Facebook▶Daily Banner	21	0.76	1.03
2	Daily Banner▶Facebook	16	0.62	57.86
3	Anzeigengruppe Nr. 2▶Facebook	11	0.43	29.14
4	Direct▶Facebook	10	0.42	2.74
5	Direct▶Daily Banner	7	0.27	1.69
6	Daily Banner▶Direct	5	0.20	17.82
7	Daily Banner▶Anzeigengruppe Nr. 2	3	0.12	10.90
8	Anzeigengruppe Nr. 2▶Daily Banner	3	0.12	7.75
9	google▶Direct	2	0.08	2.01
10	SEO▶Daily Banner	2	0.08	2.12

图7-6　广告渠道的关联访问

综上，我们来总结一下如何通过这三种方法实现对广告整合传播模

型的更好解读：

·用户广告来源路径可以帮助我们了解带有转化的用户访问来源的所有先后序列以及转化步长和时间，这对于转化过程、时间和模式的理解非常重要。虽然每个转化的路径是一个全路径，但读者其实可以将其路径作为已经预处理好的关联分析的源数据，直接对其做关联分析可以从中找到有转化的用户的广告来源模式。

·目标转化归因可以帮助我们根据企业自身特点定义的归因模型，有效的对参与转化的广告渠道做贡献分配，从而辅助于价值评估和付费投入，尤其对于处于转化“前期”处于引流和辅助功能的渠道特别重要。

·广告渠道的关联访问可以帮助我们了解所有用户频繁的访问模式，尤其对于小范围的媒介组合尤其有效，它可以解决全过程（包含转化和非转化）的用户关联访问模式的问题。

虽然上述三种方法已经相对于以前的调研问卷方式有了进一步的量化提升，但仍然有以下几方面问题需要注意：

·上述方法的实现目前都是基于cookie的，而我们知道cookie的稳定性会随着时间、用户操作等因素改变，这会导致数据直接发生变化。

·用户应用平台的多样化以及多设备、多浏览器和多应用导致的同一个用户识别难度增加，如果用户没有有效的识别方式，那么数据会产生极大的分散性，也就无法产生关联效应。

·流量作弊的问题在广告领域比较频繁，在做整合分析之前的异常检测和数据排除工作也必不可少。

·受限于数据采集的限制，当用户仅仅浏览但是没有点击企业投放的广告并到达企业网站或应用时，由于企业无法获得广告曝光信息，因此无法对非点击或者点击非到达类的渠道做评估。常见的此类渠道以展示类广告为主，例如CPM类。

有关归因、路径以及其他网站分析的课题，具体参见4.7节的内容。

7.8.4 流量预测模型

广告流量预测几乎是每个营销部门在做广告策划时的必要步骤。通过广告流量预测模型可以基于现有的流量以及广告费用水平等因素，预测在一定条件下可以产生多少流量。在电子商务公司中，这种流量往往基于销售目标产生，通常思路是企业先确定销售任务，然后根据销售任务反推需要的流量支持。

流量预测根据不同的场景有不同的方法：

·如果是没有可控的自变量或无法找到自变量的，例如直接流量、引荐流量、自然社交媒体流量等，可以考虑使用时间序列分析方法，具体参见4.6节的内容。

·如果是费用控制类媒体，例如SEM、硬广、导航类广告等，可以使用回归类模型做流量预测。具体方法参见4.2节的内容。

流量预测应用跟其他数值型预测（例如销售预测）的方法类似，但存在一定的特殊性：

·广告费用的持续性。一般情况下，广告费用支出是持续的，但在某些情况下，可能由于费用到账不及时等因素导致广告无法投放，此时会出现有费用无流量的情况。这些通常是由于沟通机制和媒介自身因素导致。

·服务器并发的响应性。当企业做大型促销活动时，流量往往呈几倍甚至几十倍的增长，如果企业服务器无法支撑瞬时的高流量并发，那么会影响整个公司的数据工作，包括流量、销售、会员等。在数据方面的影响主要是没有流量数据、销售数据下跌等。

·广告媒体的相互影响。广告媒体的投放往往会产生交叉影响效应，这意味着即使某些媒体没有投放广告，也会受到其他媒体或活动的影响而产生数据变化。例如投放广告通常会增加SEM品牌关键字、品牌区、导航网站、直接输入渠道的流量。

·作弊流量。这里又一次提到作弊流量，原因是作弊这一因素很多

情况下不可控并且不一定能被检测出来。另外，不同类型的广告的作弊流量规模也不同。通常点击类（流量数量为主的广告渠道，例如硬广）作弊较为严重，SEM、导航、社交媒体等相对较好。

·广告效果的持续性。当广告停止投放之后，广告效果仍然会持续一段时间。这种现象比较常见，尤其是时效性较长的广告，例如电子邮件、社交媒体等。

·补量。补量的意思是广告媒介由于某些自身因素，没有达到预期承诺的广告投放标准，例如展示次数不足、点击量不足等，此时媒介会通过增加广告位置、延长广告时长等方式补足承诺效果。

7.9 流量数据化运营分析小技巧

本节我们将讨论一些常用的流量分析和应用时的小技巧。

7.9.1 给老板提供一页纸的流量dashboard

在广告效果报告中，我们通常会提供大量的数据、指标、图形来向领导层汇报公司的流量情况。但是，企业高层尤其是老板层面并不希望看到太多信息，他们只是关心影响全局性的重要内容。

如何快速地将关键信息传达给领导？这里建议使用一页纸的dashboard。dashboard也称为仪表盘，它是BI（商业智能）可视化模块的重要组成部分。在制作一页纸的dashboard过程中，最关键的因素是提取关键指标并配合适当的图形做展示。

（1）关键指标

关键指标在不同企业有不同的定位范畴，对于流量工作而言，关键指标通常包括流量数量、流量质量、成本结构和ROI四类指标。

·流量数量指标：代表流量数量的指标有访客（以及不同周期内的去重访客）、访问量、页面浏览量等指标，具体参照7.6.2节的内容。

·流量质量指标：流量质量涉及的基本行为、转化行为、交易行为等多种场景，具体可参照7.6.3节的内容。

·成本结构指标：在成本结构类指标上，高层领导往往更关注哪些媒体大类以及关键媒体花费了企业主要营销预算以及单位成本如何，具体参照7.6.1节的内容。

·ROI指标：ROI是投资回报率，虽然流量不直接跟收入挂钩，但在全公司的统一范畴内，都要衡量单位收益和成本的比例。

如何确定关键指标？通常跟三方面因素有关：

1）企业文化。企业通常情况下关注哪些指标，有哪些应用习惯，关键指标的选取一定在关注指标之内。例如公司对于流量数量的评估如果以访问量为基础，那么关键指标就不要选择访客。

2）活动目标。不同的营销活动可能对应的目标不同，例如促销类

广告活动可能以引流和销售为主要目标，而新品展示类广告活动可能以拉新为主要目标。这两种情况下，前者看重整体流量和销售，后者侧重新用户流量和销售。

3）企业所处的阶段。同一个企业在不同阶段的战略目标往往不同，企业初期会以扩大知名度为突破口，因此大量的覆盖媒体以及扩大宣传是侧重点，测试关于曝光和展示类的指标更重要；进入到快速上升阶段，需要扩大市场、引入大量新客户，因此对于广告流量、新用户的引入更为关注；在平稳发展阶段，对于用户的重复消费、市场结构的稳定和成本结构优化往往又是重点。

以电子商务网站为例，以下几个指标可以作为参考项：

·流量数量指标：访问量（访客以及其他去重逻辑的访客会增加对于人数理解的难度）。

·流量质量指标：产品页转化率、加入购物车转化率、购物车转化率（普通行为类指标的意义会小于具有交易转化的指标）。

·成本结构指标：不同广告类型的预算占比、不同细分广告渠道预算占比（既有总体类别又有细分渠道的预算占比，可以帮助领导快速了解费用支持的总项目和细分项目）、CPC（每次点击成本）、每订单成本。

·ROI指标：各个细分广告渠道ROI。

（2）适当的图形

有了合适的指标，下面需要做的就是通过适当的图形做展示。通常图形展示以简单、易懂的为主，例如饼图、柱形图、折线图、地图、漏斗图等能突出对比和趋势意义的图形；尽量少的使用玫瑰图、3D图、桑基图、双坐标轴图等复杂或需要先仔细观察才能理解的图形。

很多企业（尤其是第三方服务公司）会使用一些速度计、记分卡等具有单个信息元素的展示形式，对于此类展示是否适用要根据实际情况而定。如果企业领导层对于要展示的数据有比较清晰的概念，并且能快速一眼通过当前单个数据了解趋势、优劣等特征，那么可以使用该形

式；如果领导层对历史背景不甚了解或熟悉或者由于要展示的信息过多而导致无法一眼看出优、劣、好、坏的，那么就不应该使用该形式。

大多数情况下，即使无法通过记分卡、速度计等看出实际效果，企业领导一般也不会明确表达出来。所以，为了慎重起见，建议读者少用这些形式。

另外，对于一页纸内容长度的把控，建议读者根据要汇报对象的“高度”以及汇报内容的详尽程度具体定义，有以下规律可供遵循：

·领导级别越高，对于信息的汇总和宏观视角要求越高。一般不存在一份dashboard既能满足经理级别的需要又能满足CMO的需要。

·领导层级越高，其仔细阅读每个细分报告内容的时间和精力越少。因此，报告的内容长度应该随着汇报对象的级别增加而递减。

·如果没有特征要求，可以先从一份具有“中等”长度的报告做起，不断通过领导的反馈来做内容长度测试。

除此以外，还有下面有些小技巧可以用到做dashboard过程中：

·不要在一份dashboard中罗列太多指标，指标太多跟没有指标的效果差不多，都会淹没关键信息，并且也会失去dashboard的意义。

·不要期望一份dashboard能解释所有事情，如果能从中得到关键结论就可以，其他的可以通过专项工作深入研究。

·dashboard中尽量通过对比、趋势、加粗等多种方式突出主要信息点，并尽量减少汇报对象对于结果优、劣、好、坏的思考时间。

·对于dashboard中的异常变化一定要提前做功课，因为有功底的领导对于数据都是敏感的，当遇到数据异常或变化较大时，一定会追问为什么，作为分析师要提前做数据分析和探索找到原因，即使汇报对象不追问也可以主动解释。

·对大多数企业的内部应用而言，内容大于形式，在把数据研究透

彻之前，不要把过多精力放在版式和美化上。

7.9.2 关注趋势、重要事件和潜在因素是日常报告的核心

日常报告是流量数据化运营分析的基础形式，也是所有企业最常见的数据辅助支持形式。日常报告如果要在常规化的前提下做出特色，内容是最重要的一个方面，以下是针对日常报告中内容的4个建议。

·关注整体趋势。周期性报告一定要有关于整体趋势的定论，对比、环比、定基比都是比较好的趋势观察方法，关于整体趋势的变化结论除了描述涨落以外，还需要确定涨落异常；另外，确定标杆值也是日常数据描述的重要途径和参照点。

·关注重要事件。报告期内的重要事件是汇报对象普遍关注的模块，因此有必要将重要事件的数据及对整体的影响做简要分析。

·关注潜在因素。除了整体数据外，作为数据分析师一定能通过数据发现报告周期内的潜在因素，该因素可能是与整体趋势相近或相反的，但对整体可能产生重要影响的业务节点。

·关注成本对象。大多数情况下，企业内部对于成本的支出较为关注，因此应该在报告中将占有较高成本的对象的实际效果加以反馈，尤其对于其变化的原因加以了解。

某些数据从业者单纯发送数据表格的工作并不算报告，这些工作应该尽量通过系统自动实现；报告一定是有事实、有观点、有结论的主体，而不是一个数据陈列。

7.9.3　使用从细分到多层下钻数据分析

细分是网站分析的基本方法，也是数据分析的基本思路。细分分析的过程是对整体数据进行层层拆分，然后找到影响整体的局部因素。

举例：表7-1显示了全站昨日访问量环比增长1888，比例为39%，要针对该结论进行分析只需要将来源渠道做细分（下钻）即可。

步骤1　全站流量按来源模块可细分为广告、SEM、SEO和直接输入（假设只有4个模块）。细分发现广告是网站流量的主要来源（昨日访问量占比82%），访问量增长2194，比例为67%，说明了广告是网站访问量增长的主要驱动因素。

步骤2　对广告模块做进一步细分，发现其中主要增长模块为Sina，该模块昨日访问量占比79%，环比前日增长1990，比例为85%。如果该模块有不同的位置，还可以做进一步细分。

至此已经找到了昨日网站访问量增长的主要原因是Sina来源流量增长，此时可直接找到Sina模块的业务负责人进行沟通进一步原因。

表7-1　访问量细分数据

一级维度	二级维度	三级维度	昨日访问量	前日访问量	变化 %
全站			6721	4833	39%
	广告		5483	3289	67%
		Sina	4340	2350	85%
		Sohu	590	591	0%
		其他	553	348	59%
	SEM		457	520	-12%
	SEO		99	70	41%
	直接收入		550	580	-5%

传统的网站分析工具大多只提供了1层下钻或细分的功能，这往往难以发现具有深层原因的事物主体。某些工具则通过提供更多或无限层级的下钻能力，可以为分析师提供找到深入原因主体的可能性，例如Webtrekk提供了4层下钻，Adobe Analytics的临时分析则提供了无限下钻的功能。

在上述细分分析过程中，虽然整体访问量上升，但是我们发现部分模块流量下降且比例较大，如SEM下降比例为12%。这种权重较小的数据异常往往容易掩盖在整体数据之下，此时数据从业者可针对该问题提出质疑和分析。

7.9.4 通过跨屏追踪解决用户跨设备和浏览器的访问行为

用户行为信息的采集都是基于cookie实现的，而cookie是以浏览器作为依存环境，这意味着当同一个用户在同一个电脑的不同浏览器上或者不同的电脑上访问时，会被认为的是不同的用户。针对这种情况，我们会使用一种称为“跨屏追踪”的方法来实现用户行为的辨别和标志。

实现跨屏追踪必须依赖于可靠的唯一识别标志，该标志的产生有以下两个前提条件：

·登录。目前暂时还没有其他方案能取代登录，登录（包含注册）意味着用户有唯一的识别标志，该标志可以将真实的用户区分开，这是其中最为关键的一步。

·Cookie有效。这个环节可能被很多人忽略，Cookie有效意味着用户的电脑中必须存在之前跟登录ID关联的Cookie信息，这样才能将用户ID与其他浏览器下的CookieID关联起来，即实现用户跨屏追踪与关联。

是否具有跨屏追踪对于结果的影响是显著的，图7-7显示了两种状态的差异。

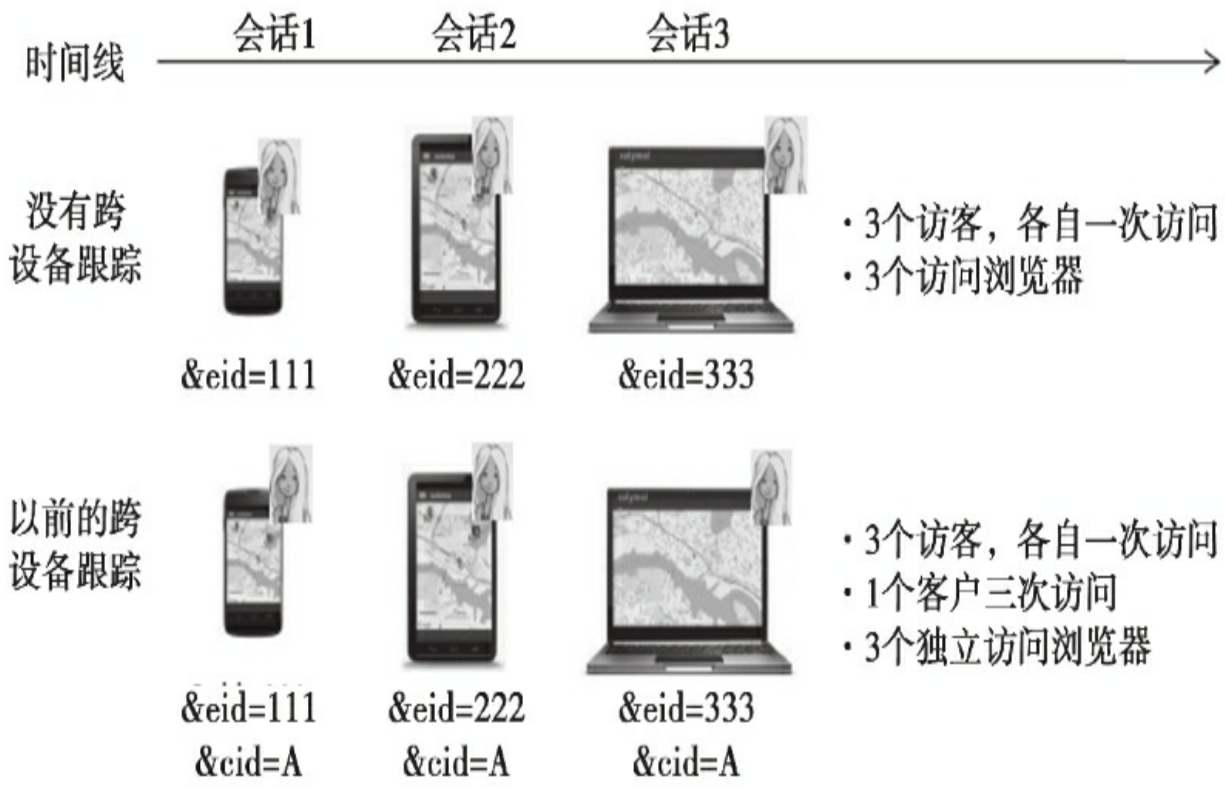


图7-7　跨屏追踪原理

其中：eid是唯一访客的标志，cid是唯一用户的标志：

·当没有跨屏追踪时，统计结果是3个访客、3个访问、3个独立浏览器；

·当有跨屏追踪时，统计结果是3个访客、3个访问、1个用户、3个独立浏览器。

两份不同结果显示了，当用户在不同的设备间存在登录行为时，我们可以将于其登录ID将多个设备的信息关联起来。但是相反的，如果没有用户ID（用户注册或登录）、或者没有CookieID（用户唯一识别标志）我们都无法准确识别用户的关联行为，也就无法跨屏追踪。

跨屏追踪能在有唯一登录ID的情况下对用户去重，这意味着某些指标或数据会发生变化。用户去重并不意味着数据丢失或者减少，而是由于系统将重复访问的用户合并。

不同的系统有不同去重应用方法：

在Adobe Analytics中，独特访客的计算依据将改变，原有的独特访客计算以Cookie为计算依据，根据不同的时间去重；启用该功能后，独特访客将首先依据用户ID进行去重，如果没有用户ID再根据Cookie去重。

举例：以图7-7所示的用户行为为例，在Adobe Analytics中的独特访客将变成1，而不是3。但实际上我们知道用户数为1，独特访客是为3，这种计算逻辑跟之前是有区别的。因此，建议大家在Adobe Analytics中慎重使用该功能。

在Webtrekk中，针对启用去重功能的账户会发生一些变化。

·变化1 针对用户的去重，跟Adobe Analytics类似，都在原有的UV指标基础上进行了过滤，所以情况与上述相同。

·变化2 针对原有的独特访客统计，新增了一个名为“Browsers，Unique”的指标，功能与原有的独特访客相同，都是基于Cookie统计的独特访客量，不想使用该功能的用户仍然可以针对登录情况，使用Browsers，Unique做独特访客统计。

虽然跨屏追踪受限于用户登录以及cookie的信息机制而无法对全部用户行为进行识别，但实际上正是那些有注册或登录的用户才是网站真正有价值的用户群体，因此对他们的研究和应用价值非常大。

应用1 跨设备交叉分析

统计不同设备之间的交叉访问情况，尤其结合推广、购买订单、浏览商品、预订服务等价值极大。比如要分析购买了A商品的客户特征，其中一个重要的维度是用户如何从手机、电脑和其他联网设备查看并购买该商品的，交叉比例怎么样。

应用2 跨设备的数据挖掘

既然用户的不同设备访问都有记录，我们可以基于跨设备的数据做进一步挖掘。例如：

·不同设备之间的访问间隔怎样？

·不同情况下（产品、广告、服务等），用户在不同设备间的关联特征如何，是手机到PC关联强还是PC到平板更具交叉性？

·不同设备间，如何开始一次新的行为？如针对一款商品的推广，用户通常先从PC开始还是先从手机开始了解？不同设备应该在商品推广周期中的前、中、后哪个环节出现？

·设备之间的广告投放是否存在重复推广给同一批用户导致过度曝光或浪费广告费用的情况？

·用户真的可以直接从手机端下端？如果没有下单或手机订单效果差，手机端是如何影响他们在PC下单的？

·服务公司一直在讲的“立体传播”“综合覆盖”是适合企业实际情况吗？

·公司多渠道访问路径又是如何与多设备立体交叉的？

7.9.5 基于时间序列的用户群体过滤

在做流量分析时，群体细分是针对多种研究对象进行筛选过滤的前提。细分（Segment）一词在不同的网站分析工具里面的翻译有差异，例如Adobe Analytics的区段、Google Analytics和Webtrekk的细分都是一个意思。

在大多数细分群体过滤条件应用时，都是以无时间序列的方式做条件组合，例如：

·看了首页又看了商品页的用户；

·搜索了A品牌词2次又看了帮助中心的用户；

·看了M活动页又将商品P加入购物车的用户。

但很多时候，我们可以将不同事件的时间序列模式加入到群体细分条件中，例如还是上面的3个示例，加入时间序列特征后条件如下：

·先看了首页后又看了商品页的用户；

·先搜索了A品牌词2次又看了帮助中心的用户；

·先看了M活动页又将商品P加入购物车的用户。

加入时间序列后的过滤条件要求目标过滤条件必须有明显的先后顺序特征，否则即使两个条件都满足也不符合条件。这种基于时间序列的用户群体过滤能应用到很多具有明显事件先后顺序的分析场景中，例如：

·分析促销活动效果，要求先浏览活动然后才购买活动中的商品；

·分析广告媒体效果，要求先看了A广告，然后再看了B广告；

·分析流量引导模式，要求先点击C1商品展位，后点击C2商品展位。

时间序列的有效范围的定义跟用户指标的范围一致，包括访客级别、访问级别、页面（事件级别），这些不同的级别分别指代多个发生的事件处于一个访客、访问、页面区间内，不同的区间可以混用；另外，再结合条件本身可以基于and、or等多种模式做多维度过滤，可以产生非常复杂的时间序列定义。下面仅以访问周期定义做示例：

1）简单的时间序列，指用户在点击查看特定页面之后又查看了另一个页面，两个页面之间只考虑先后关系，而不考虑访问次数或频率，两次点击发生在一个访问周期内。如图7-8所示的条件定义中，A→B→C→D和A→C→B的用户行为模式都能匹配该定义，而B→A无法匹配。

图7-8　简单时间序列

2）跨访问的时间序列，指在一次访问中完成一个点击，然后在另一次访问中完成一个点击，两个点击之间存在时间序列关系。与简单的时间序列相对，该访客需要匹配跨不同访问的点击。如图7-9所示的条件定义中，如果A和B处于不同的访问，且发生先后关系为先A后B，则匹配该区段；如果A和B发生在同一个访问内则无法匹配。

图7-9　跨访问的时间序列

3）混合访问的时间序列，指用户的前2个点击处于一个访问区间内并且具有先后序列关系，然后跟第三根点击处于不同的访问区间内，也具有序列关系。如图7-10所示的条件定义中，定义了一个用户在不确定数量的访问中查看了A和B两个页面，但在不同的访问中查看了C页面。此时，用户可以在相同或不同的访问中访问页面A和B，但C页面的访问必须与A和B的任何一方不在同一个访问区间内。

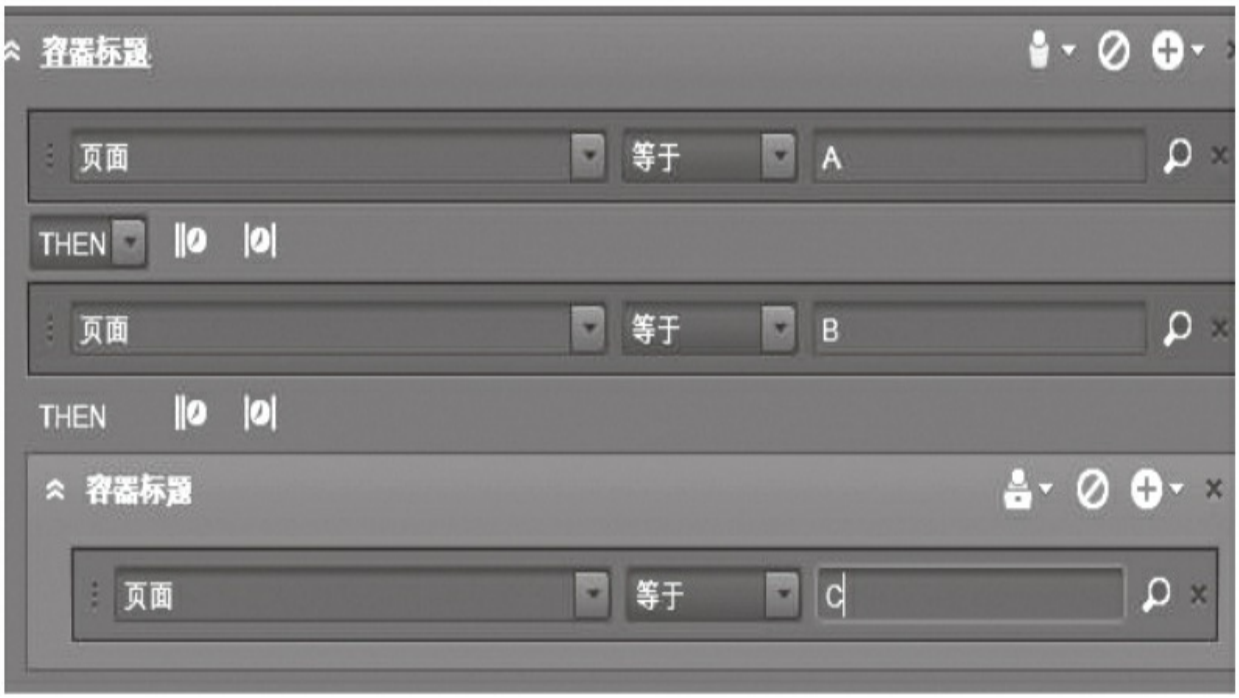


图7-10　混合访问的时间序列

4）混合访问的聚合时间序列，指用户的前2次不同访问内有不同的行为模式且具有先后时间序列关系，并且各个访问内部有聚合指标作为过滤条件。如图7-11所示，定义了一个在页面查看序列中，用户必须是第二次（含第二次）之后点击的页面是A，然后访问页面B或页面C，而与访问数无关。定义该条件后，由于访问次数维度的约束，页面A需要至少为第二个页面查看；然后，必须在相同或随后的访问中查看页面B，与访问数无关。

图7-11　混合访问的聚合时间序列

除了上述用法外，很多强大的网站分析工具还支持更多用法，例如之间模式（between and）、区间内模式（within）、不多于（less than）、不少于（more than）等，基于这些模式再配合not、and、or和多种过滤维度（事件），可以变换出无穷的过滤规则。

7.10　流量数据化运营分析的“大实话”

本节的大实话，将来探讨一些常见的流量工作的深层规律。

7.10.1 流量数据分析的价值其实没那么大

虽然流量是所有后续工作的先锋，看似拥有很高的企业价值，但作为网站数据分析从业者，是否问过自己这样一个问题：网站数据分析的价值真的那么大吗？假如公司没有网站数据分析，各项业务运作体系是否会受到影响？——如果你的回答是不肯定甚至是确定没有影响，那足以证明在整个公司流程中网站数据分析工作没有多少价值。

对于不同类型的公司，网站数据分析工作的价值大小有所差异。

在线营销类或服务类广告公司的业务核心是通过为广告主提供广告投放、评估和优化业务，进而获得费用差价、佣金返点、服务费等，由此形成公司核心利润业务。网站数据分析所处的角色是对这些业务体系提供数据评估和优化，此时由于从业者的工作与公司核心业务结合紧密，因此其职业价值会比较高。

线下苏宁、国美的大卖场为集团销售类业务贡献90%以上的利润，而线上的电商业务体系分割了不到10%甚至需要利润补贴进行运作，此时针对线上业务的网站分析体系的价值在整个苏宁和国美集团中显得微不足道。由于从业者的工作与公司核心业务距离较远，因此其职业价值会比较低。

上述两个例子阐述了网站数据分析在不同公司的不同存在意义，从本质上讲网站数据分析价值的外部环境取决于公司的基础基因，即公司的核心业务模式是否与网站分析相关以及相关性的强弱。

7.10.2 如何将流量的实时分析价值最大化

在做流量数据工作时都期望自己的工作内容能在实际运营中发挥重要作用。其中流量的实时分析是流量工作的重要内容之一，但针对流量的实时分析并不能在所有场景中都发挥作用，而是有特定的作用范围和要求：

·可监测的业务效果。实时分析发挥作用的前提之一是有数据支持，这要求数据既要可控于企业内部又要可测量。例如，企业在电视媒体上投放的广告由于不可测量而无法提供实时数据支持；企业在视频网站上投放品牌类广告，如果没有可供监测的播放时长、播放率等数据，也无法提供实时数据支持。

·可实时反馈的数据。实时分析的第二个关键点是数据可以实时更新，实时数据支持的基础频率是分钟和秒，在某些场景下采用按小时或天更新的频率无法满足实时分析需求。

·可优化的业务节点。可优化是实时分析的输出关键，这意味着产生实时分析结果后，业务方可针对性的改善和优化；如果实时监测的业务无法进行优化操作，那么实时分析的价值将大打折扣。

实时分析并不能针对所有渠道都发挥作用，某些渠道的固定性质决定了其不存在实时分析的落地价值。

（1）固定投放类渠道

固定投放类渠道包括包段投放的广告（例如包月、包天）、固定购买的网址导航（例如360导航的文字链）、商务合作的友情链接、SEM品牌区等。这些渠道的特点是购买或合作的媒介类型已经固定、无论是投放时间、投放内容还是投放频率，其可操作性较差。例如针对网址导航的媒介购买，通常是季度投放，即使通过实时数据发现一天中某个时间点存在可优化价值，由于其推广形式（通常是网站文字名称）和合作周期（固定一个季度）限制，实时分析无法落地。

（2）资源紧俏类渠道

虽然当前可供选择的广告媒介数量众多，但流量高、效果好、效果持续的媒介仍然是少数，如网址导航广告、百度关键字、门户首页和特定频道页、客户端弹窗等。在这种背景下，业务团队执行的第一要素仍然是抢夺投放资源和排期，因此即使出现可供实时支持的辅助决策，也会由于缺乏优化资源支持而无法实施。这类资源以流量巨大的门户、富媒体资源为主。

7.10.3 营销流量的质量评估是难点工作

针对营销部门的效果评估是流量工作的重点内容之一。流量效果评估可通过“7.4流量数据化运营指标”中的指标实现，其中涉及的流量质量的评估不仅是重点还是难点工作，主要表现在两方面：

一方面，“质量”并不是一个恒定且可以用数据直接定义的指标，不同目标、不用需求、不同时间下质量定义不同。

另一方面，“质量”结果的影响因素多种多样，很难准确区隔不同因素对“质量”的影响权重。例如，针对跳出率高的分析点至少包括三方面：

·站外渠道因素，即渠道本身的质量因素，包括群体喜好、需求、媒体质量等；

·站外广告因素，即广告素材对质量的影响，包括广告卖点、宣传商品、促销价格等；

·站内自身因素，包括着陆页设计、用户对网站品牌的认知度、熟悉程度等。

以上三方面因素综合影响跳出率，但很难只将站内渠道因素分离处理并得到渠道本身质量问题；尤其是关于信息匹配度、需求吻合度、价格敏感度、需求强烈度、品牌认可度等主观数据无法直接通过数据测量。这也是当前流量运营中的难点，因此大多数情况下流量效果评估仍然聚焦在流量规模上，而对于流量质量的评估上属于“辅助”评估因素。

作为数据从业者，仍然可以一定方法来对营销流量做评估和校验：

·对比分析是评估广告流量质量的实用方法，将广告流量与非广告流量进行对比即可了解到底是渠道质量问题还是网站自身质量问题。

·建立复合指标评估体系，将用户的复杂行为分解为可供站内评估的目标矩阵，如注册、试用、订单、产品页浏览、加入购物车等。

·A/B测试（双变量测试）是找到最佳方案的有效手段，可直接对比发现影响渠道质量的关键因素。

7.10.4 个性化的媒体投放仍然面临很多问题

个性化的媒体广告投放相对于大众广告投放具有以下3方面优势：

1）更好的用户体验。传统广告的单向传播模式在个性化媒体时代发生了变化，用户的每一个“声音”都被记录和分析；同时，媒体已经开始“猜测”用户需求，媒体的角色开始从主观推送需求向满足需求转变，整个过程的用户认知度、体验度和忠诚度提高。

2）营销效率的提升。传统广告的购买与投放执行主要依靠广告代理或企业自身，经过谈判之后购买固定时间、版面、人群、网站群等；由于整个过程主要靠人工操作，效率低且出错几率大，更重要的是要耗费大量人力、财力、物力和时间成本。个性化媒体投放通过自动化、智能化程序实现人群定位、素材管理与投放、效果评估与自我优化等关键流程，大大提高了营销效率。

3）营销效果的提升。个性化媒体投放从广告曝光、点击、到达这三个关键环节都是针对用户个性化需求而提供的内容，高度相关的内容提高了广告点击率、到达率，直接从广告源头提高流量，直接降低CPM、每次点击成本和每次访问成本的同时还能促进广告转化率的提升。

个性化媒体投放针对用户投放个性化内容，在广告整个生命周期内的各个环节效果指标都会有所提升，包括更好的点击率和点击次数、更好的二跳率和访问深度、更高的转化率与转化贡献价值等。

但是在真正应用过程中仍然面临很多问题：

（1）媒体认知问题

从大众传播开始媒体就一直处于强势地位，广告、消息等都是直接通过媒体向用户单向传播；媒体对于自身定位、服务价值、服务方式等定位仍然停留在强势媒体阶段，大多数媒体尤其是强势媒体还没有“客户服务”的概念，个性化媒体广告投放无从谈起。这直接导致了接入到个性化媒体平台的媒体数量，尤其是优质媒体数量的不足。

（2）优势资源问题

虽然现在可供选择的媒体平台众多，但真正的高价值流量仍然集中于少数媒体，这些媒体往往通过CPD（包段购买，如包天、包月）、CPM（每千次付费）等品牌传播类形式售卖优势资源，如新浪门户的首页焦点图、视频媒体TIPS弹窗等、导航文字链都属于此类售卖方式；而能够实现个性化媒体投放的资源往往是优势资源之外的“剩余资源”或低价值资源位，因此造成个性化媒体投放难以大规模应用到核心媒体资源或核心媒介中。覆盖面的不足直接导致了曝光的补足，个性化媒体从入口开始面临紧缩考验。

（3）技术实现问题

个性化投放的实现需要基于海量数据，但这些数据以及对应的实施技术和规则等仍然面临现实性难题：

·数据不全面。个性化媒体投放需要尽量多的关于平台、用户和投放企业的信息，而这些信息是分散在不同场景中的，媒体本身无法掌握全部信息。数据不全直接导致个性化投放效果不佳。例如我在A商城浏览了P商品，但之后2天我却在B商城购买该商品，此时A商城的个性化广告很可能会持续给我推送P商品的广告和促销活动。

·方案无法全部个性化。个性化的媒体广告投放，如果是涉及商品投放的可以直接调取商品属性和描述信息，但如果是文字、图片、视频以及交互类广告的则需要企业主提供多种交互方案，这本身就不是一个可以完全个性化的实施思路，因为广告主无法想到所有的个性化广告方案。

7.10.5 传统的网站分析方法到底缺少了什么

趋势、细分和转化是传统网站数据分析的三大方法，但这种方法随着数据工作的深入会发现其价值越来越不显著。这些分析方法到底缺少了什么？

这三种分析方法配合漏斗分析、路径分析、归因分析等可以为企业建立良好的整体性、流程性及趋势性结果，但是归根到底这都是针对群体或整体性的分析，而缺少针对个体性质以及个体间相互关系的分析。

当前市场上的大型第三方网站分析解决方案似乎也意识到了这一点，因此在保持传统网站分析方法的同时又增加了可以深入挖掘数据价值和规律的“数据挖掘”或“机器学习”方法。

（1）关联分析模型

我们在很多场景下都介绍过关联分析的应用，在网站分析工具中，关联分析模型的应用代表是Webtrekk。Webtrekk将关联模型封装到产品底层，其是基于Top-K关联规则产生（Top-K Association Rules）。这一算法是在传统基于最小支持度和置信度的前期下做的优化，它具有优异的性能以及可扩展性，尤其当用户想要控制输出规则的数量时非常有效。

该模型可以应用到页面关联分析、站内外搜索词关联分析、产品浏览关联分析、产品购买交叉分析、渠道访问交叉分析。在使用该报表和数据之前，需要先在Webtrekk中配置关联模型相关维度和参数，如图7-12所示。

Webtrekk的关联分析应用具有以下特点：

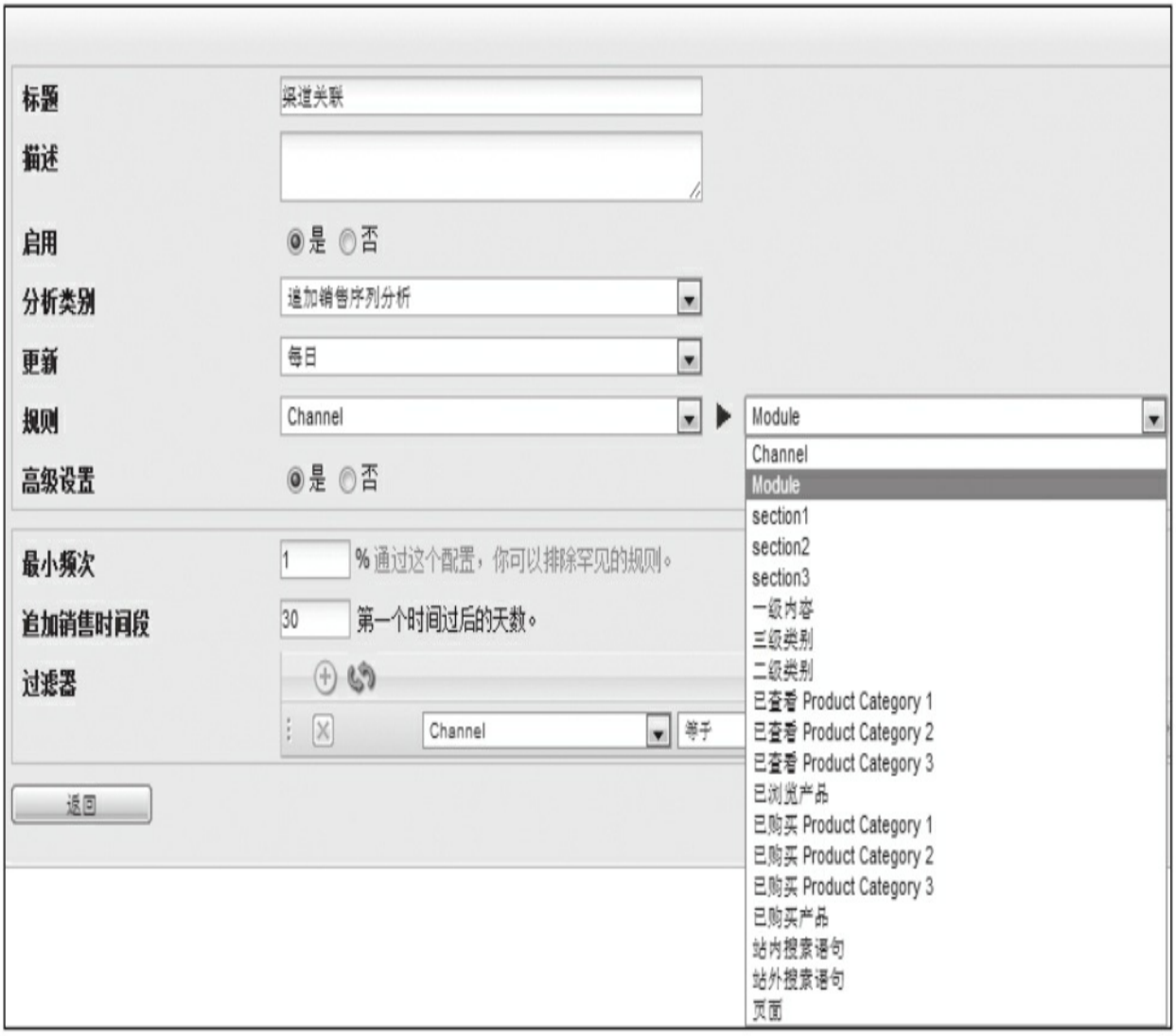


图7-12　Webtrekk关联模型配置

·关联分析的算法支持交叉销售算法和向上销售算法两种；

·数据集都是基于Raw Data（原始在线采集数据）；

·数据计算时间可设置为每天、每小时或一次；

·挖掘维度支持页面、渠道、产品、广告、站内外搜索词的浏览和购买关联；

·支持数据挖掘的高级配置。支持最小频繁度，分析类别选择追加销售序列分析时还可设置追加的数据集时间，支持基于细分群体的关

联，如只看某个页面的关联效果，只需要过滤该页面即可。

Webtrekk的关联模型应用范围广泛，它可以提供以下数据价值洞察：

·用户搜索了站内A关键词之后通常会优化搜索哪个关键词？

·用户在看了A页面之后，通常还会看哪个页面？

·用户买了A产品之后，还会一起买哪个产品？下次又会买哪个产品？

·用户从A渠道进入网站之后，通常还会从哪个渠道再次进入？

在所有的数据挖掘类算法中，规则提取类是最受业务应用关注的算法，原因是提取后的规则可直接帮助业务开展业务活动，实用价值最高（规则提取类算法包括关联、回归、决策树等以直接目标为分析导向，提取能实现目标规则的算法，如购买A的用户下一次通常会购买B）。

（2）RFM和RFE模型

我们在5.4.3节和5.4.2节分别提到过RFM和RFE两个模型。网站分析工具Webtrekk提供了这两种分析模型。

该模型在系统后台的配置如图7-13所示。

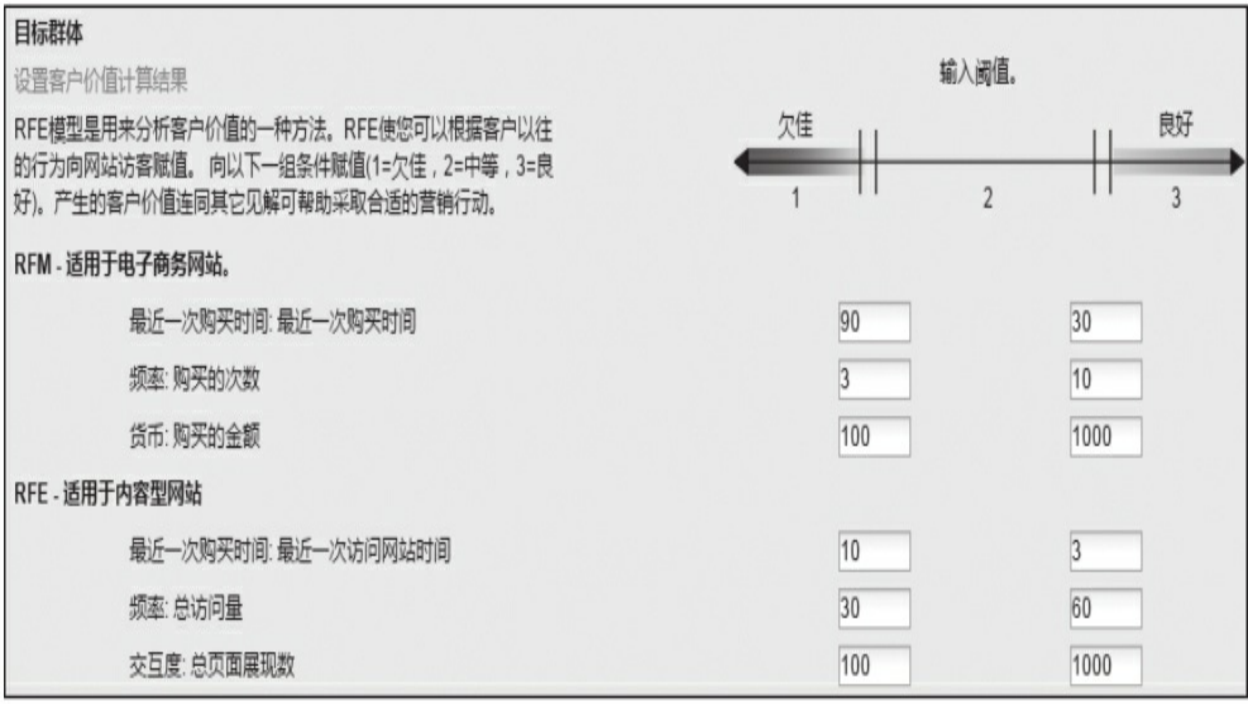


图7-13　Webtrekk RFM和RFE配置功能

这两种模型具有如下特点：

·每天运行一次模型，不同时间内得到的RFM和RFE得分有差异。

·基于阈值定义而非百分位（例如3分位）的区间划分。

·支持自定义阈值来控制三个区间的范围边界。

（3）异常检测模型

Adobe　Analytics提供了另外一种预测性检测方法——异常检测模型。Adobe Analytics使用的是时间序列的预测方法，由三种算法组成：

·Holt Winters Multiplicative（Triple Exponential Smoothing）——霍尔特温特斯乘法（三重指数平滑法）。

·Holt Winters Additive（Triple Exponential Smoothing）——霍尔特温特斯加法（三重指数平滑法）。

·Holts Trend Corrected（Double Exponential Smoothing）——Holts趋

势校正（双指数平滑法）。

这三种算法实际上共同组成了温特斯季节指数平滑模型，其基本思想是把具体线性趋势、季节变动和随机变动的时间序列进行分解研究，并与指数平滑法相结合，分别对长期趋势（U_t）、趋势的增量（B_t）和季节变动（F_t）作出估计，与指数平滑法结合起来，可以同时处理趋势和季节性变化，并能将随机波动的影响适当地过滤掉，然后建立预测模型，因此，特别适用于包含趋势和季节变化的时间序列的预测问题。

该模型配置比较简单，所有算法都已经封装到系统内部，用户只需要进入后台设置训练数据集、数据时间、数据维度即可：

·数据训练集时间：数据训练集，即要进行计算和处理的样本数据的时间范围，（默认截止到昨天）数据训练集的时间可选项为30天、60天、90天。

·数据查看集时间：与数据训练集时间选项相同，不同点在于数据直接用来验证训练集的结果。

·数据指标和维度：异常检测可以针对全站所有的维度和指标进行预测。默认情况下，系统后台是针对全站的异常检测配置，针对自定义维度设置预警需要额外配置。图7-14展示的是针对BD模块UV预警。

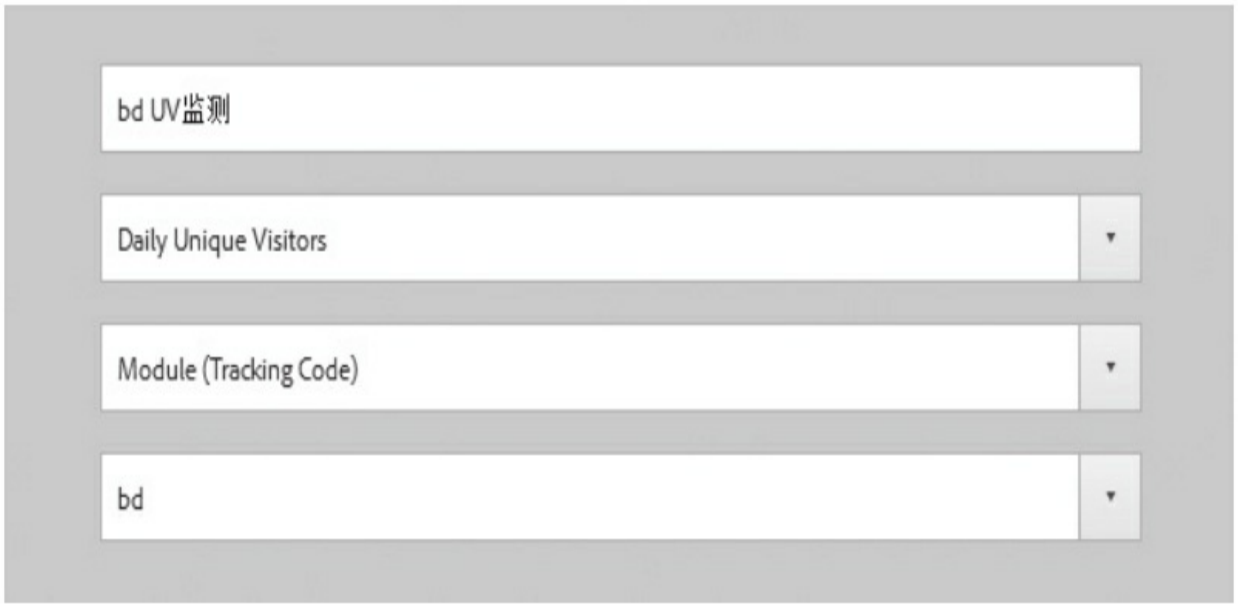


图7-14　针对BD模块UV预警设置

这种异常检测模型可以应用到以下场景：

·监测网站平均订单价值、订单量、订单转化率波动；

·注册或登录的异常变化；

·某个登录页面浏览量趋势；

·正在投入巨额广告费的渠道效果波动；

·网站跳出率情况是否正常波动。

除了上述算法外，Adobe的Data　workbench也提供了部分数据统计分析和挖掘模型。Data　workbench是Adobe　Analytics中数据挖掘能力最强的产品，它侧重于跨渠道（线上线下）数据整合分析。在Data　workbench集成了聚类模型、关联矩阵和评分模型三种算法，提供了更多统计学数据解读视角。

（4）聚类模型

通过访客聚类，可以利用客户特性对访客进行动态分类，并基于选定的数据输入生成聚类集，从而识别具有相似兴趣和行为的群组，以便进行客户分析和定位。Data　workbench的聚类使用的是K-Means算法，聚类分析结构如图7-15所示。

（5）关联矩阵

关联矩阵实际上不属于数据挖掘算法，它是一种统计学方法，用来分析不同变量间相关性关系。Data　Workbench中的统计关联基于皮尔逊关联模型，该模型的本质是验证变量间的线性依赖关系，关系强弱用R值表示。有关相关性分析的更多话题，请参见3.8节的内容。

（6）评分模型

通过倾向评分，针对每个访客计算的评分表示指定事件（由目标过滤器定义）可能发生的估计概率。因此，评分值的范围介于0%～100%

之间。该模型常用于在执行某个流程或发起某项促销活动之前，结合增益图和提升图预估可能产生的业务结果，图7-16为Data　workbench倾向评分模型。

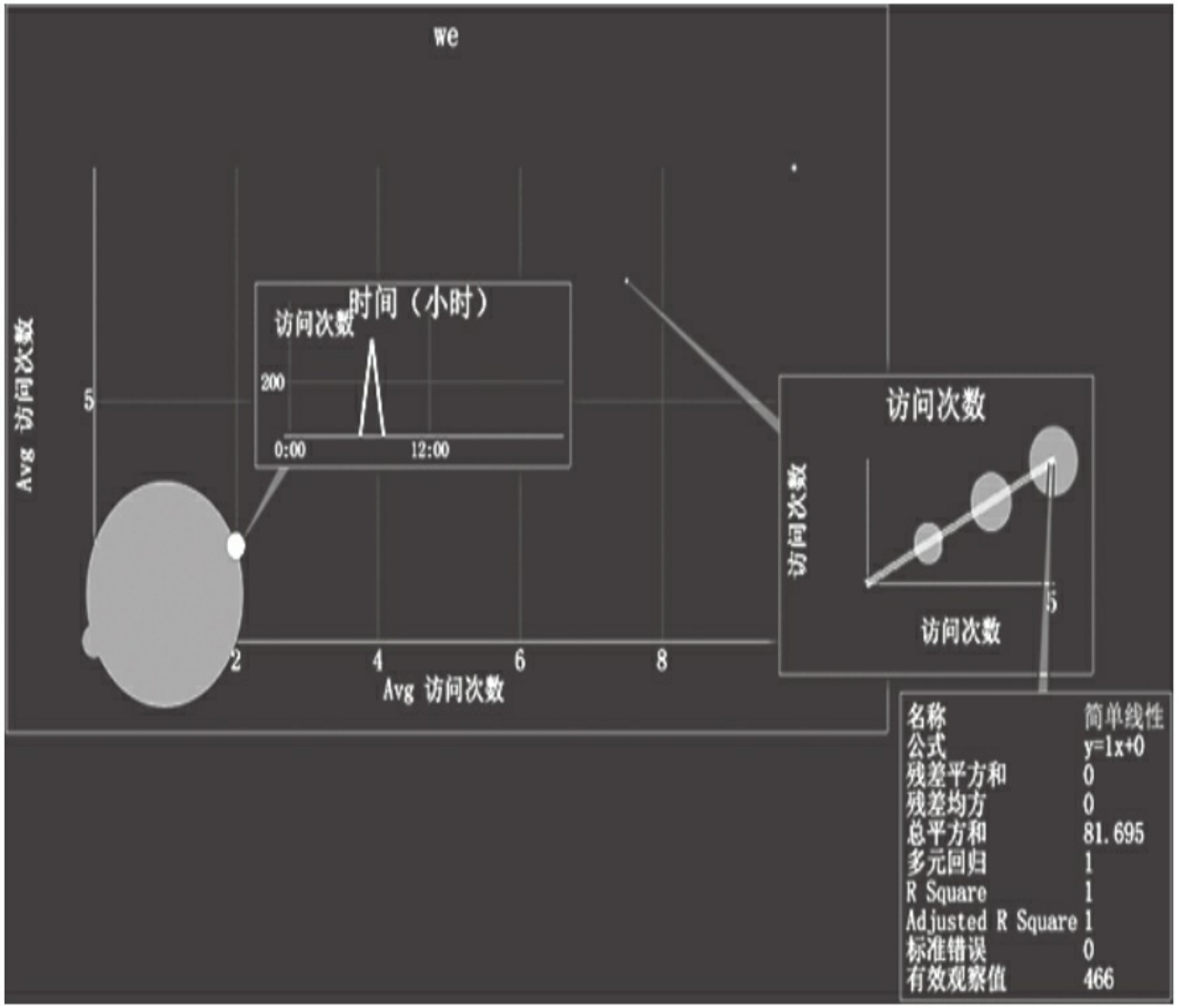


图7-15　Data workbench聚类结果

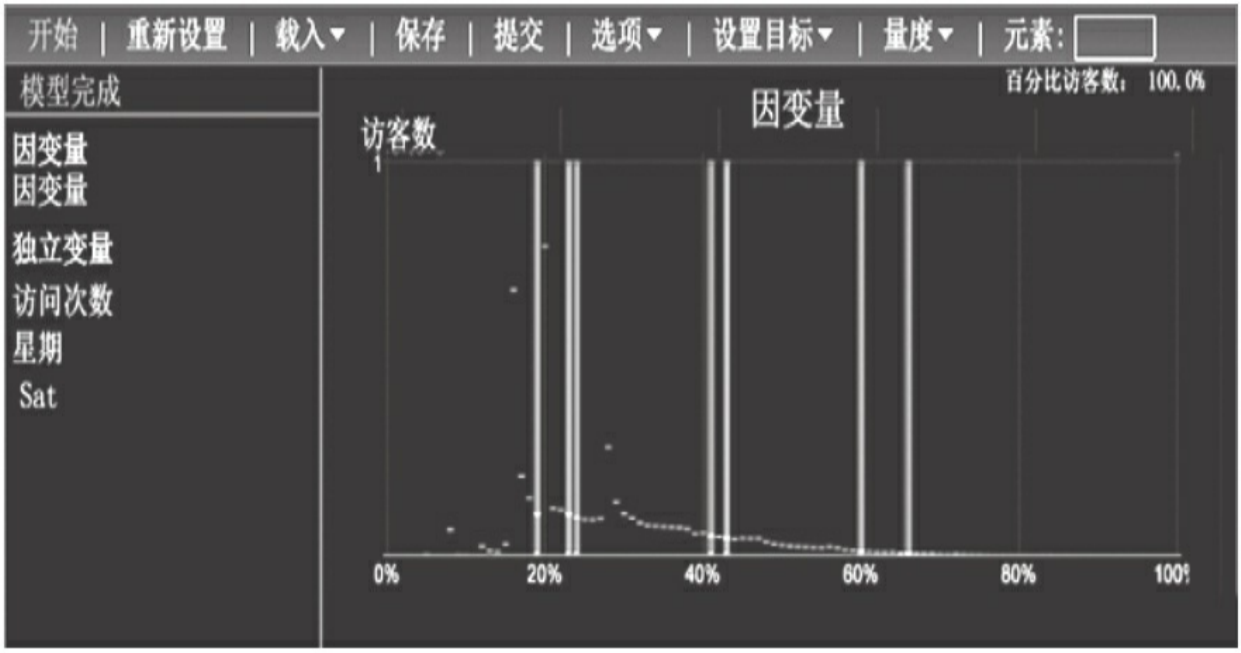


图7-16　Data workbench评分结果

但是，即使如此，针对流量数据的深度挖掘仍然还有很大的发展空间。因此，我们需要将更多的深度算法和模型应用到网站和流量分析工作中才能产生更大价值。

7.11　案例：基于自动节点树的数据异常原因下探分析

7.11.1　案例背景

日报、周报、月报等常规性报告是各个公司的基础数据支持形式。在日常报告中，经常会出现很多异常波动的指标，需要分析师找到异常波动的影响因素。但在寻找主要因素时由于需要下探的层级较多，实施起来会非常费时费力。以大型公司的广告投放渠道为例，可能包括以下层级：

·一级渠道包括SEM、AD、CPS、Social、导航等；

·二级渠道以SEM为基准包括百度、谷歌、360等；

·三级渠道以百度为基准包括关键字、网盟等；

·四级渠道以关键字为基准包括不同的广告计划；

·五级渠道以广告计划为基准可以细分到不同的广告组；

·六级渠道以广告组为基础可以细分到不同的关键字。

本案例介绍了一个自动化细分找到主要影响因素的方法——基于自动节点树的数据异常原因下探分析方法，该方法的实施思路是：找到每个层级上影响最大的因素并依次做下一因素的细分，直至最后一个因素。具体过程如下：

步骤1　统计全站在一定周期内、特定指标下的数据环比变化量和环比变化率。

步骤2　指定要分析日期并获得该日期及其前1天的数据。

步骤3　以全站数据为基准，下探第一层级维度并对指定日期和其前1天的数据做分类汇总。

步骤4　计算第一层级维度下分类汇总后的两天数据的差值并得到环比变化量和环比变化率。

步骤5　对第一层级维度下的变化量排序，并分别获得环比变化量最大和最小情况下的维度名称、变化量和变化率。

步骤6　计算下一层级变化量与上一层级变化量的比值，变化量最大值和最小值的比例将被定义为正向贡献率和负向贡献率。

步骤7　循环上述步骤，直至所有层级都计算完成。

步骤8　使用树形图展示所有层级下的变化量最大和最小的维度信息包括维度名称、环比变化量、环比变化率、贡献率等信息。

由于数据的波动可能包括正向波动和负向波动，因此在数据存储和展示时的正向和负向是相对的概念。如果全站波动为负值（数据下降），那么正向的波动为最小值（一般为负值）；如果全站波动为正值（数据增长），那么正向的波动为最大值（一般是正值）。

上述过程完成后会得到如图7-17所示的结果，有关该图的解释会在本节后面具体介绍。

本节案例的输入源数据advertising_data.csv和源代码chapter7_code1.py位于“附件-chapter7”中，默认工作目录为“附件-chapter7”（如果不是，请cd切换到该目录下，否则会报“IOError:File advertising_data.csv does not exist”）。程序的输出是直接打印图形并保存为图片。

7.11.2 案例主要应用技术

本案例用到的主要技术包括：

·数据处理：找到每个节点的极值并计算极值贡献，该过程通过Python“手动”实现。

·数据展示：基于GraphViz的节点树图展示。

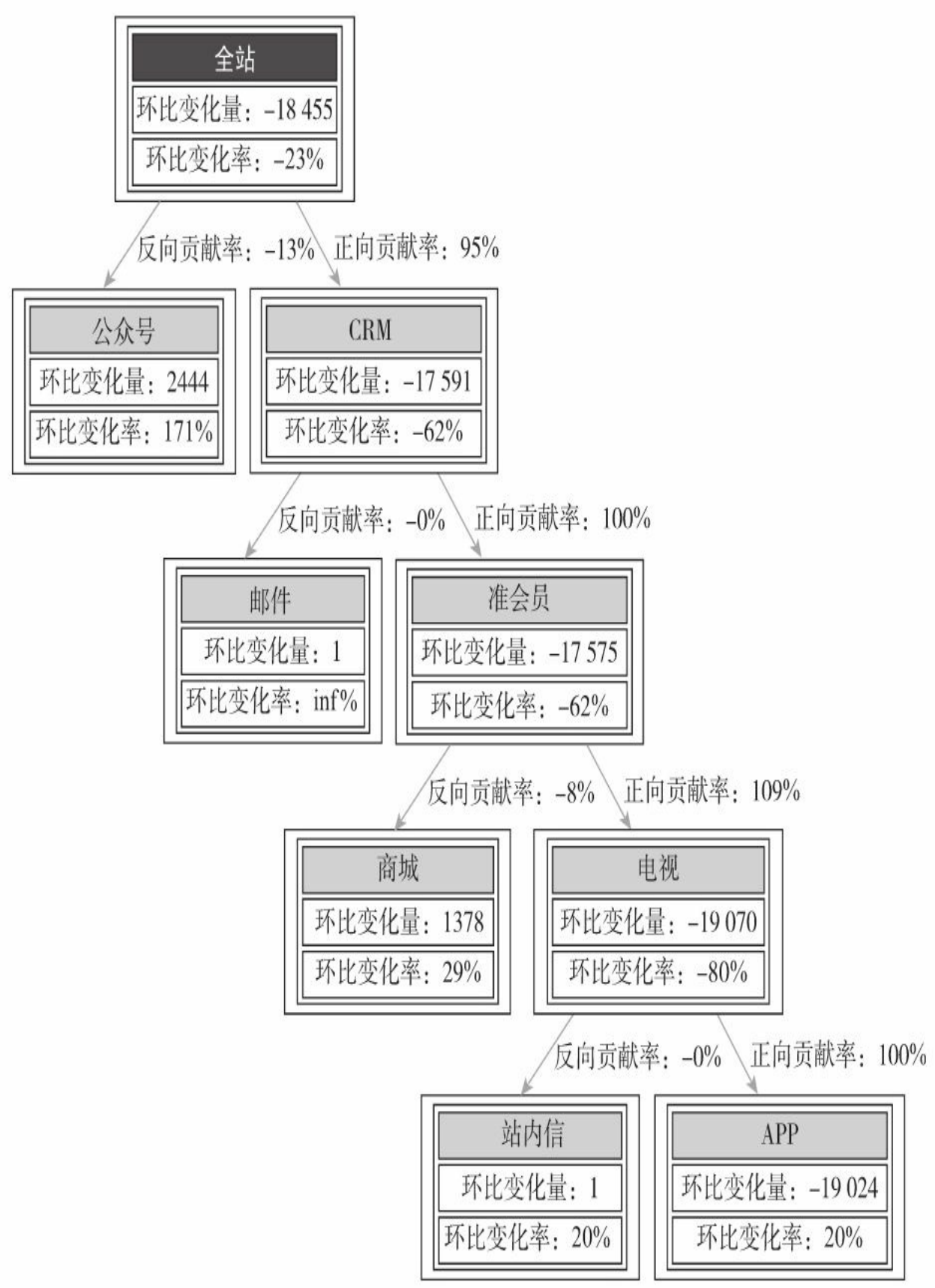

全站
环比变化量：-18 455
环比变化率：-23%
反向贡献率：-13%
正向贡献率：95%
公众号
环比变化量：2444
环比变化率：171%
CRM
环比变化量：-17 591
环比变化率：-62%
反向贡献率：-0%
正向贡献率：100%
邮件
环比变化量：1
环比变化率：inf%
准会员
环比变化量：-17 575
环比变化率：-62%
反向贡献率：-8%
正向贡献率：109%
商城
环比变化量：1378
环比变化率：29%
电视
环比变化量：-19 070
环比变化率：-80%
反向贡献率：-0%
正向贡献率：100%
站内信
环比变化量：1
环比变化率：20%
APP
环比变化量：-19 024
环比变化率：20%

图7-17　主要影响因素节点树形图

主要用到的库包括：sys、Numpy、Pandas、datetime、GraphViz，其中GraphViz是展示数据的核心。

从技术方法上讲，本案没有用到复杂的模型和算法，都是基于统计完成的。但是本案例的应用具有以下特点：

·自动按照指定的日期找到影响的最大因素，并可层层分解找到对应层级的上一层和下一层关联影响因素以及对应的贡献量。

·在节点树中除了关注正向影响，还增加了负向影响因素的信息，可以帮助分析师找到被整体波动埋藏的负向规律。

7.11.3 案例数据

案例数据是来自某企业的网站分析系统，其中的数据部分已经经过处理。以下是数据概况：

·维度数：6。

·数据记录数：10693。

·是否有NA值：无。

·是否有异常值：有。

以下是本数据集的6个维度的详细说明：

·date：日期，格式是YYYY/MM/DD。

·source：流量来源一级分类，来源于业务部门的定义。

·site：流量来源二级分类，来源于业务部门的定义。

·channel：流量来源三级分类，来源于业务部门的定义。

·media：流量来源四级分类，来源于业务部门的定义。

·visit：访问量。

上述的四个层级是企业对于流量来源类别的层次性划分，一级类别包含二级类别，二级类别包含三级类别，依次类推。

7.11.4 案例过程

步骤1 导入库。

```
import sys
reload(sys)
sys.setdefaultencoding('utf-8')
import datetime
import pandas as pd
import numpy as np
from graphviz import Digraph  # 画图用库
```

本案例主要用到了以下库：

·sys：系统库，用来将默认编码设置为UTF-8，目的是处理中文处理

·datetime：Python内置标准库，用来做时间处理

·Numpy：基本数据处理

·Pandas：数据读取、审查、异常值处理等

·GraphViz：画图用库，主要用来做树形图打印和输出

步骤2 读取数据，该步骤使用pandas的read_csv方法读取csv文件。

```
raw_data = pd.read_csv('advertising_data.csv')
```

步骤3 数据审查和校验，该步骤包含了数据概览、类型分布等。

```
print ('{:*^60}'.format('Data overview:'))
print (raw_data.head(2))  # 数据概览
print ('{:*^60}'.format('Data dtypes:'))
print (raw_data.dtypes)  # 数据类型
```

该部分内容已经多次讲过，因此这里不具体讲解用法。上述代码执行后输出如下结果：

数据概览：从结果中看到数据没有乱码，也没有格式问题，数据录入也正常。

```
**********Data overview:**********
        date       source   site     channel media  visit
0  2017/5/15     品牌营销_品牌词品牌词产品播放器播放标签     PC  17600
1  2017/5/15  手机_品牌营销_品牌词品牌词广告  15秒前贴片_app   app  15865
```

数据类型：输入的数据中，第一列是date但默认读取为字符串，原因是没有做特定转换；而visit应该是数值型格式，但是却被读取为字符串格式，因此这里存在问题。经过查看原始数据，发现在visit中，部分字段的数据为“-”，这类数据需要在预处理阶段处理掉。

```
**********Data dtypes:**********
date       object
source     object
site       object
channel    object
media      object
visit      object
dtype: object
```

步骤4　数据预处理，根据上述的数据审车校验结果，需要处理的内容包括：

·将date列转换为日期型，便于后期基于数据做筛选以及日期计算。

·将visit列中的“-”进行转换。由于从网站分析工具导出时，“-”代表的是没有数据，因此这里转换为0。

```
raw_data['visit'] = raw_data['visit'].replace('-', 0).astype(np.int64)  # 替换字符为0然后转换为整数型
raw_data['date'] = pd.to_datetime(raw_data['date'], format='%Y/%m/%d')  # 将字符串转换为日期格式
print ('{:*^60}'.format('Data overview:'))
print (raw_data.head(2))
print ('{:*^60}'.format('Data dtypes:'))
print (raw_data.dtypes)
```

以下是主要实现过程：

·通过数据框的字段选择方法，选取visit列并使用replace方法将“-”替

换为0，再使用astype方法将数据类型转换为int64。

·使用之前介绍过的pd.to_datetime方法将date列转换为datetime格式。

转换完成后，再次打印输出前2行数据以及数据类型如下：

```
**********Data overview:**********
        date       source   site     channel media  visit
0 2017-05-15     品牌营销_品牌词品牌词产品播放器播放标签     PC  17600
1 2017-05-15  手机_品牌营销_品牌词品牌词广告  15秒前贴片_app   app  15865
**********Data dtypes:**********
date       datetime64[ns]
source             object
site               object
channel            object
media              object
visit               int64
dtype: object
```

数据已经按照预期转换了格式，并且没有产生意外数据变更。

相关知识点：使用pandas的replace方法替换字符串

pandas中的replace方法是一个非常强大的字符串替换功能，它可以实现字符串、字典、列表、数字甚至正则表达式级别的转换。

语法：

```
replace(self, to_replace=None, value=None, inplace=False, limit=None, regex=
```

其中关键参数如下：

·to_replace：要替换的对象。它可以是str、regex、list、dict、Series、numeric、None。

·str：完全替换和匹配的字符串。

·regex：匹配符合正则表达式模式的对象。

·list：可以是字符串列表、正则表达式列表或数值列表。如果

to_replace（原始对象）和value（替换后的对象）都是列表格式，那么列表长度必须一致；如果regex参数设置为True，那么两个列表中的所有字符串将被应用正则表达式。

·dict：可以使用嵌套字典直接做匹配模式定义，例如嵌套字典（{'a'：{'b'：nan}}）将在列'a'中查找值'b'，并将其替换为nan，嵌套中也可以使用正则表达式。

·value：替换后的值。它也可以是scalar、dict、list、str、regex，其中dict对象可以将每个列指定为不同的替换值，例如{'a':1，'b':2}可以将列a替换为1，列b替换为2。

步骤5　计算整体波动量，这里先将每天的数据与其前1天的数据做环比变化统计。

```
day_summary = raw_data.ix[:, -1].groupby(raw_data.ix[:, 0]).sum()  # 按天求和汇总
day_change_value = day_summary.diff(1).rename('change')  # 通过差分求平移1天后的变化量
day_change_rate = (day_change_value / day_summary).round(3).rename('change_r相对昨天的环比变化率
day_summary_total = pd.concat((day_summary, day_change_value, day_change_rat合为完整数据框
print ('{:*^60}'.format('Data change summary:'))
print (day_summary_total)
```

具体实现过程如下：

·使用ix方法指定特定列，并使用groupby方法做分类汇总，目标是以第一列（date）为单位，对最后一列（visit）做求和（sum）汇总，得到每天的visit数据day_summary，这是一个series类型的数据。

·对day_summary使用diff方法做一次差分然后使用rename方法将其名称改为change，得到每1天相对于前1天的变化量day_change_value。

在4.6节我们提到的ARIMA方法中，需要针对数据做差分处理，在当时的案例中，我们使用的是log方法做处理。除此以外还可以使用diff方法做差分，差分用到的方法diff方法。diff能够对数据做一定量的偏移计算，参数为使用periods=n指定的偏移量，默认n=1。

·对day_change_value和day_summary求商得到环比变化率day_change_rate，对结果使用round（3）方法保留3位小数，并使用rename方法设置名称为change_rate。

由于在使用diff做差分时，默认的day_summary为float64，因此将day_change_value和day_summary直接做除法时可以得到浮点型值。如果day_change_value和day_summary的类型都为整数型，那么得到的结果是整数型。此时，需要将其中至少一个对象转换为float型才能得到浮点型结果，例如day_change_value/day_summary.astype（np.float64）。

·使用之前介绍过的pd.concat方法将上述3个Series沿列合并为一个数据框，打印结果如下：

```
********************Data change summary:********************
             visit   change  change_rate
date
2017-05-15  117260      NaN          NaN
2017-05-16  166124  48864.0        0.294
2017-05-17  157727  -8397.0       -0.053
2017-05-18  155805  -1922.0       -0.012
2017-05-19  115644 -40161.0       -0.347
2017-05-20  120833   5189.0        0.043
2017-05-21  123145   2312.0        0.019
2017-05-22  113624  -9521.0       -0.084
2017-05-23  131248  17624.0        0.134
2017-05-24  149783  18535.0        0.124
2017-05-25  112208 -37575.0       -0.335
2017-05-26   98556 -13652.0       -0.139
2017-05-27  125342  26786.0        0.214
2017-05-28  122626  -2716.0       -0.022
2017-05-29  134067  11441.0        0.085
2017-05-30  137391   3324.0        0.024
2017-05-31  150686  13295.0        0.088
2017-06-01   80334 -70352.0       -0.876
2017-06-02   90468  10134.0        0.112
2017-06-03   79892 -10576.0       -0.132
2017-06-04   91720  11828.0        0.129
2017-06-05   97115   5395.0        0.056
2017-06-06   97984    869.0        0.009
2017-06-07   79529 -18455.0       -0.232
2017-06-08   83676   4147.0        0.050
2017-06-09   74351  -9325.0       -0.125
2017-06-10   76256   1905.0        0.025
2017-06-11   78155   1899.0        0.024
2017-06-12  133994  55839.0        0.417
2017-06-13   77315 -56679.0       -0.733
2017-06-14   46273 -31042.0       -0.671
```

上述结果显示了每天的数据变化量以及变化率。由于数据是后一天相对于前一天做偏移计算，因此第一天（2017-05-15）的change和change_rate为空值。

基于上述数据，我们可以分析特定日期的变化原因。这里我们指定要分析的日期为2017-06-07，要找到到底哪些维度导致全站访问量的下降。

步骤6　指定日期自动下探分解。

获得指定日期、前1天以及各自对应的数据。

```
the_day = pd.datetime(2017, 6, 7)  # 指定要分析的日期
previous_day = the_day - datetime.timedelta(1)  # 自动获取前1天日期
the_data_tmp  =  raw_data[raw_data['date']  ==  the_day].rename(columns=
{'visit': the_day})  # 获得指定日期数据
previous_data_tmp = raw_data[raw_data['date'] == previous_day].rename(columr
得前1天日期数据
```

·使用pd.datetime指定分析日期，参数格式为年、月、日。

·使用datetime.timedelta方法做日期迁移，得到其前1天的日期值。

·使用数据框的字段选择方法，选取date列中值等于the_day等记录并将该列重命名为the_day，目的是后面合并起来的数据能有效区分不同日期。

·使用相同的方法获得前1天的数据。

定义要使用的变量。

```
dimension_list = ['source', 'site', 'channel', 'media']  # 指定要分析的维度：4
个层级
split_node_list = ['全站']  # 每层分裂节点名称列表
change_list = list()  # 每层分裂节点对应的总变化量
increase_node_list = []  # 每层最大增长贡献最大的1个维度
decrease_node_list = []  # 每层最小增长贡献最大的1个维度
```

·dimension_list：指定分析的维度，这里使用了广告媒体的全部类别特征。

·split_node_list：每个层级分裂节点的名称，其中全站为顶级节点，因此直接加入到列表。

·change_list：每层分裂节点对应的数据变化总量，该列表的每个值会用来做分裂方向判断。

·increase_node_list：每层最大增长贡献的维度，一般情况下列表中的值为正数。

·decrease_node_list：每层最小增长贡献的维度，一般情况下列表中的值是负数。

接下来是整个分裂过程的计算，由于代码比较长，我们拆分为6个part单独讲解。最外层的for循环过程针对每个维度做循环，下面开始6个部分的内容。

1）part1计算指定维度下的数据。

```
for dimension in dimension_list:  # 遍历每个维度
    # part1
     the_data_merge = the_data_tmp[[dimension, the_day]]  # 获得指定日期的特定
维度和访问量
    previous_data_merge = previous_data_tmp[[dimension, previous_day]]  # 获
得指定日期前1天的特定维度和访问量
    the_day_groupby = pd.DataFrame(the_data_merge.ix[:, -1].groupby(the_data
指定日期特定维度汇总求和
    previous_day_groupby = pd.DataFrame(
        previous_data_merge.ix[:, -1].groupby(previous_data_merge.ix[:, 0]).
指定日期前1天特定维度汇总求和
```

基于指定日期的数据，使用列名选择方法获得维度列（dimension）以及对应日期的访问量数据（the_day）。使用相同的方法获得指定日期前1天特定维度的数据。

对指定日期使用ix选择以最后一列为维度，对最后一列做汇总求和；使用相同的方法对前1天的数据做分类汇总。

经过上述步骤得到每个维度下各自特征的分类汇总数据。由于接下来我们需要使用数据框方法做合并，因此分类汇总后的数据使用pd.DataFram转换为数据框。

2）part2将两天的数据合并然后计算其变化量和变化率。

```
    # part2
    merge_data = pd.merge(the_day_groupby, previous_day_groupby, how='outer'
                  right_index=True)  # 合并2天的数据
    merge_data = merge_data.fillna(0)  # 将缺失值（没有匹配的值）替换为0
    merge_data['change'] = merge_data[the_day] - merge_data[previous_day]  #
算环比变化量
    merge_data['change_rate'] = merge_data['change'] / merge_data[previous_d
算环比变化率
    total_chage = merge_data['change'].sum()  # 获得分裂节点的总变化值
    change_list.append(total_chage)  # 将每个节点的变化值加入列表
```

使用pd.merge方法做汇总后的两天数据做合并，其中参数如下：

·the_day_groupby：第一个要合并的数据框。

·previous_day_groupby：第一个要合并的数据框。

·how='outer'：合并方式为outer，相当于SQL中的full outer join，两个数据框中的全部数据都会形成结果列表。匹配不上的数据会以NA值显示。

·left_index=True：指定以第一个数据框的index作为匹配对象。

·right_index=True：指定以第二个数据框的index作为匹配对象。

使用merge_data.fillna（0）方法将合并后没有匹配到的NA数据替换为0。

按照跟计算日环比变化量和环比变化率的相同方法计算该维度下的数据环比变化量和变化率。

使用merge_data['change'].sum（）对特定维度的变化量做求和。该值是分列节点变化量的总值，而非单独特定分裂节点的值。

使用change_list.append（total_chage）将分裂节点变化量的总值加入列表，该列表会在下面做节点图输出时用到。

3）part3计算当前维度下变化量最大值对应的各项信息。

```
    # part3
    merge_data = merge_data.sort_values(by='change')  # 按环比变化量正向排序
    max_increase_node = merge_data.ix[-1].name  # 获得增长变化量最大值节点名称
    max_value, max_rate = merge_data.ix[-1][2:4]  # 获得最大值节点变化量以及变化比例
    increase_node_list.append([max_increase_node, int(max_value), max_rate])最大值信息追加到列表
```

使用merge_data.sort_values（by='change'）对merge_data按照change列做排序，排序结果将以变化量change为维度做正向排序，第一行的值为最小值，最后一行的值为最大值。

使用merge_data.ix[-1].name获得增长变化量最大值节点名称。

使用merge_data.ix[-1][2:4]获得最大值节点变化量以及变化率。

将变化量最大的节点名称、变化量和变化率数据以append方法追加到列表中。

4）part4计算当前维度下变化量最小值对应的各项信息。

```
    # part4
    min_increase_node = merge_data.ix[0].name  # 获得增长变化量最小值节点名称
    min_value, min_rate = merge_data.ix[0][2:4]  # 获得最小值节点变化量以及变化比例
    decrease_node_list.append([min_increase_node, int(min_value), min_rate])最小值信息追加到列表
```

该部分的计算方法跟part3相同，区别仅在于part3在使用ix方法选择数据时选择的是最后一行（值为-1），而part4选择的是第一行（值为0）。

5）part5针对增长趋势的数据做逐层数据过滤。

从这里开始分别针对增长和下降两种趋势做数据逐层过滤。由于在针对异常值的自动下探过程中，会面临数据增长与数据下降两种情况。因此，针对数据增长的情况需要以左侧最大值作为分裂节点的条件来选择数据；而针对数据下降的情况需要以右侧最小值作为分裂节点的条件来选择数据。

```
    # part5
```

```
    if total_chage >= 0:  # 判断为增长方向
         split_node_list.append(max_increase_node)  # 将分裂节点定义为增长最大值
节点
        rules_len = len(split_node_list)  # 通过分裂节点的个数判断所处分裂层级
        if rules_len == 2:  # 第二层source，第一层为全站整体
            the_data_tmp = the_data_tmp[the_data_tmp['source'] == max_increa
source为维度过滤出指定日期符合最大节点条件的数据
            previous_data_tmp = previous_data_tmp[
                 previous_data_tmp['source'] == max_increase_node]  # 以
source为维度过滤出前1天符合最大节点条件的数据
        elif rules_len == 3:  # 第三层site
            the_data_tmp = the_data_tmp[the_data_tmp['site'] == max_increase
site为维度过滤出指定日期符合最大节点条件的数据
            previous_day_data_tmp = previous_data_tmp[
                 previous_data_tmp['site'] == max_increase_node]  # 以site为
维度过滤出前1天符合最大节点条件的数据
        elif rules_len == 4:  # 第四层channel
            the_data_tmp = the_data_tmp[the_data_tmp['channel'] == max_incre
channel为维度过滤出指定日期符合最大节点条件的数据
            previous_data_tmp = previous_data_tmp[
                 previous_data_tmp['channel'] == max_increase_node]  # 以
channel为维度过滤出前1天符合最大节点条件的数据
        elif rules_len == 5:  # 第五层media
            the_data_tmp = the_data_tmp[the_data_tmp['media'] == max_increas
media为维度过滤出指定日期符合最大节点条件的数据
            previous_data_tmp = previous_data_tmp[
                 previous_data_tmp['media'] == max_increase_node]  # 以media
为维度过滤出前1天符合最大节点条件的数据
```

当total_chage>=0时，数据变化量为正，因此判断为增长方向。此时需要将最大值节点max_increase_node追加到分裂节点列表split_node_list。

使用len（split_node_list）得到分裂节点的数量来判断当前循环所属的维度层级，由于split_node_list初始化时就已经具有一个顶层节点名称了，因此判断长度值从2开始：

rules_len为2时代表此时循环的是第二层，即第一个下探维度source。此时基于指定日期的数据将the_data_tmp中source列值为max_increase_node的记录过滤出来；同样，将指定日期前1天的数据previous_data_tmp中source列值为max_increase_node的记录过滤出来。

以此类推，当rules_len为3时代表此时循环的是第三层，即第二个下探维度site并将两天的数据过滤出来。由此可以得到每个循环下的过滤数据。


这里在使用过滤条件时，没有将上一层的条件加到下一层，原始是每一层过滤所使用的数据都是上一层过滤之后的数据，该数据已经继承了前个循环的条件。以rules_len==3为例，在rules_len==2时，已经从数据集中过滤出source等于对应层级的max_increase_node的数据；在第三层级时，自然已经基于上述过滤后的数据再针对第三层site等于对应层级的max_increase_node即可。

6）part6针对下降趋势的数据做逐层数据过滤。

```
    # part6
    else:  # 判断为下降方向
         split_node_list.append(min_increase_node)  # 将分裂节点定义为增长最大值
节点
        rules_len = len(split_node_list)  # 通过分裂节点的个数判断所处分裂层级
        if rules_len == 2:  # 第二层source
            the_data_tmp = the_data_tmp[the_data_tmp['source'] == min_increa
source为维度过滤出指定日期符合最小节点条件的数据
            previous_data_tmp = previous_data_tmp[
                 previous_data_tmp['source'] == min_increase_node]  # 以
source为维度过滤出前1天符合最小节点条件的数据
        elif rules_len == 3:  # 第三层site
            the_data_tmp = the_data_tmp[the_data_tmp['site'] == min_increase
site为维度过滤出指定日期符合最大节点条件的数据
            previous_day_data_tmp = previous_data_tmp[
                previous_data_tmp['site'] == min_increase_node]  # 以site为
维度过滤出前1天符合最大节点条件的数据
        elif rules_len == 4:  # 第四层channel
            the_data_tmp = the_data_tmp[the_data_tmp['channel'] == min_incre
channel为维度过滤出指定日期符合最大节点条件的数据
            previous_data_tmp = previous_data_tmp[
                 previous_data_tmp['channel'] == min_increase_node]  # 以
channel为维度过滤出前1天符合最大节点条件的数据
        elif rules_len == 5:  # 第五层media
            the_data_tmp = the_data_tmp[the_data_tmp['media'] == min_increas
media为维度过滤出指定日期符合最大节点条件的数据
            previous_data_tmp = previous_data_tmp[
                previous_data_tmp['media'] == min_increase_node]  # 以media
为维度过滤出前1天符合最大节点条件的数据
```

该部分的实现逻辑与part5相同，区别仅在于在每个层级应用过滤条件时，由于下降趋势中，需要针对变化量最小值的右侧节点做筛选，因此条件设置为min_increase_node（而非max_increase_node）。

至此，我们已经完成了整个下探工作并获得了4个关键列表：

·split_node_list：每层分裂节点名称列表。

·change_list：每层分裂节点对应的总变化量。

·increase_node_list：每层最大增长贡献维度及其对应的变化量和变化率。

·decrease_node_list：每层最小增长贡献维度及其对应的变化量和变化率。

步骤7　画图展示自动下探结果。以上4个列表对于数据结果的理解和展示效果并不直接，因此这里使用类似4.3.6节中决策树的图形展示结果。由于该过程也比较长，这里将其拆分为5个part。

1）part1定义节点树形图用到的属性和样式。

```
# patr1
node_style = {'fontname': "SimSun", 'shape': 'box'}  # 定义node节点样式
edge_style = {'fontname': "SimHei", 'fontsize': '11'}  # 定义edge节点样式
top_node_style  =  '<<table><tr><td  bgcolor="black"><font  color="white">{0}
</font>  </td></tr><tr><td>环比变化量:{1:d}</td></tr><tr><td>环比变化率:{2:.0%}
</td> </tr></table>>'  # 定义顶部node节点标签样式
left_node_style        =        '<<table><tr><td        bgcolor="chartreuse">
<font color="black">{0} </font></td></tr><tr><td>环比变化量:{1}</td></tr><tr>
<td>环比变化率:{2:.0%}</td> </tr></table>>'  # 定义左侧node节点标签样式
right_node_style        =        '<<table><tr><td        bgcolor="lightblue">
<font color="black">{0} </font></td></tr><tr><td>环比变化量:{1}</td></tr><tr>
<td>环比变化率:{2:.0%}</td> </tr></table>>'  # 定义右侧node节点标签样式
dot = Digraph(format='png', node_attr=node_style, edge_attr=edge_style)  # 创
建有向图
```

node_style定义的是node节点的样式，node节点指所有节点，包括分裂和非分裂节点。其中：

·fontname：定义字体名称为宋体，由于数据中包含中文，因此必须选择一个系统中存在的可以正常显示的字体样式。在windows中常用的中文字体名称如表7-2所示。

表7-2　windows常用中文字体

中文名称	英文名称
黑体	SimHei
宋体	SimSun
新宋体	NSimSun
仿宋	FangSong
楷体	KaiTi
楷体_GB2312	KaiTi_GB2312
仿宋_GB2312	FangSong_GB2312
微軟正黑體	Microsoft JhengHei
微软雅黑体	Microsoft YaHei

·shape：定义节点样式。graphviz支持的节点样式超过50种，常见的样式如图7-18所示。

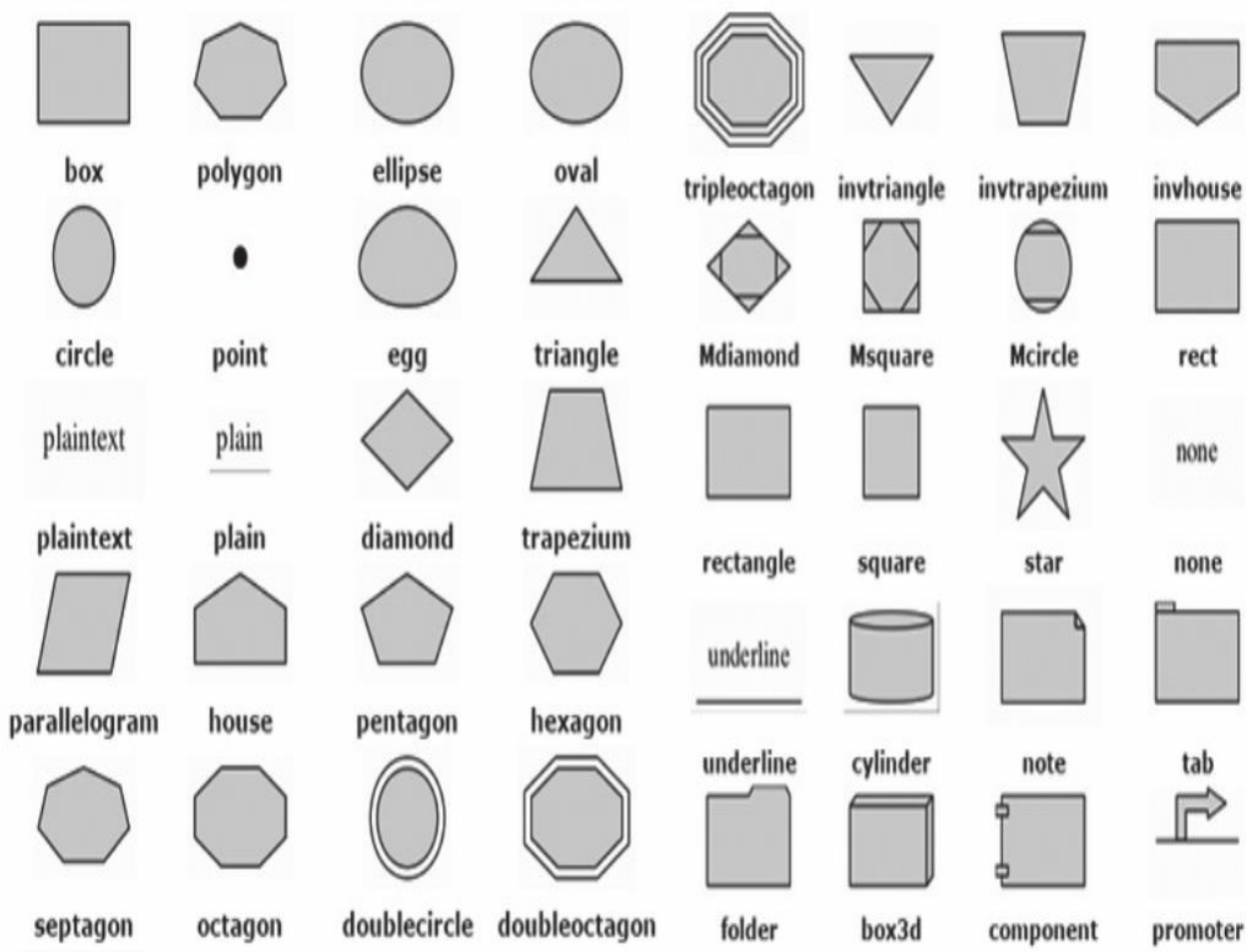


图7-18　常见的节点样式定义

edge_style：定义的是有向边的样式。其中：

·fontname：定义的是字体样式，跟node字体样式定义的用法相同。

·fontsize：定义的是字体大小，这里定义为11，单位为点（point）。

top_node_style：定义的是顶级节点的样式。由于下面我们会采用str.format的方法动态填充值，因此对其中的字符串变量使用format中的占位符表示方法。有关format方法的更多知识，请具体参照4.4.6节中的信息。

·{0}表示第一个对应变量的数据。

·{1:d}表示第二个对应变量的数据，其填充的数据类型为整数型。

·{2:.0%}表示第三个对应变量的数据，其填充的数据类型为百分比类型，保留0位小数。

在显示节点内容时，我们希望除了节点名称以外，还增加节点环比变化量以及环比变化率数据，并且以表格的形式展示出来。graphviz支持以类似HTML格式的语法显示对应效果。在top_node_style的字符串定义中，我们使用了类似于HTML的语法定义了一个3行1列的table，并且table的第一行底色为黑色，字为白色；其他两行的都是默认（底色为白色，字体为黑色）。

相关知识点：HTML中的表格

在HTML中定义表格通过<table>标签实现。一个表格中通常包含多个行或多个列，行是通过<tr>标签定义，列通过<td>或<th>标签定义。行和列的“交集”就是单元格，单元格中可以展示任何信息，比如文字、数字、图像、列表、直线以及嵌套表格等。

表格的定义一般是遵循先行后列的方法，如下代码定义了一个2行2列的表格：

```
<table border="1">
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
</table>
```

该表格的数据按照先行后列，从左到右依次是1/2/3/4，展示样式如下：

1	2
3	4

在表格定义过程中，我们会用到很多“样式”。常用样式属性如表7-3所示。

表7-3　table常用属性

属　性	值	描　述
align	left、right、center	表格元素的水平对齐方式
bgcolor	rgb值、十六进制值的颜色代码、特定颜色名称	表格的底色
border	像素值	边框宽度
cellpadding	像素值或百分比	单元格与其内容之间的空白距离

（续）

属　性	值	描　述
cellspacing	像素值或百分比	单元格之间的空白距离
width	像素值或百分比	表格宽度

例如<table width="600"border="0"align="center"cellpadding="0"cellspacing="0">定义了一个宽度为600像素，边框宽度为0（即不显示边框），对齐方式为水平居中，单元格边距为0，单元格间的边距为0的表格。

同样的，td标签也支持table中的属性，而tr则支持其中的部分属性。

left_node_style：定义的是左侧节点的样式。分别定义左侧两侧节点的样式是为了更好的区分。该部分的定义逻辑与top_node_style相同，仅仅在第一行的背景色和字体颜色上不同。这里的背景色定义为chartreuse（黄绿色），字体颜色为黑色。

right_node_style：定义的是右侧节点的样式。该部分与left_node_style定义类似，不同的是背景色定义为lightblue（浅蓝色）。

最后，使用Digraph（format='png'，node_attr=node_style，edge_attr=edge_style）创建一个有向图，然后应用上述指定的样式。其中format指的是图像文件保存为png格式图片。

2）part2获得每一层分裂节点相关信息。

这里的for循环用来循环读取每一层数据，但总量是读取4次，而非5次。这是由于每个树形图结构由两层信息构成：第一层的分裂节点和第二层左右一侧的分支节点；而下个层级又会以上一个层级的某个节点作为分裂节点在做分裂。当4层全部读取完毕之后，形成的图形便是5层层级了。

例如，当i=0时，得到的树形节点如图7-19所示。

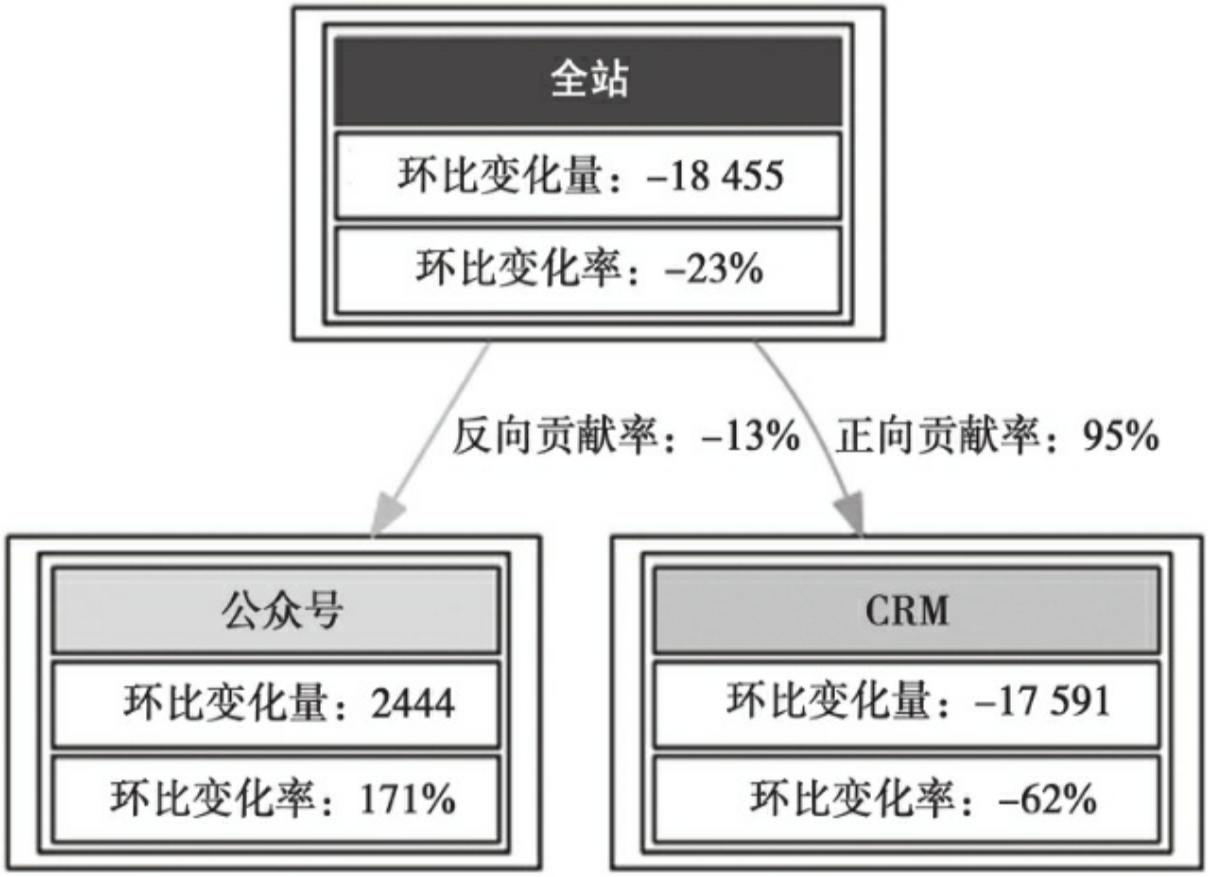


图7-19　第一层节点树

```
for i in range(4):  # 循环读取每一层
    # part2
    node_name = split_node_list[i]  # 获得分裂节点名称
    node_left, max_value, max_rate = increase_node_list[i]  # 获得增长最大值名
称、变化量和变化率
    node_right, min_value, min_rate = decrease_node_list[i]  # 获得增长最小值
名称、变化量和变化率
    node_change = change_list[i]  # 获得分裂节点的总变化量-非分裂节点变化量
    node_label_left = left_node_style.format(node_left, max_value, max_rate)
侧节点显示的信息
    node_label_right = right_node_style.format(node_right, min_value, min_ra
侧节点显示的信息
```

该部分共相对简单，主要通过列表读取对应循环次数下的各个信息：

·node_name：分裂节点名称。

·node_left、max_value、max_rate：分别是增长最大值名称、变化

量和变化率。

·node_right、min_value、min_rate：分别是增长最小值名称、变化量和变化率。

·node_change：分裂节点的总变化量。

·node_label_left：左侧节点显示的信息，这里基于part1定义的left_node_style使用str.format方法定义数据。该数据不是节点本身的数据，而是相当于左侧节点的显示数据，该数据会与node方法中的label参数结合使用。

·node_label_right：右侧节点显示的信息，该字段与node_label_left逻辑相同，也是右侧节点显示的数据。

3）part3定义顶级节点信息。由于顶级节点是全站数据，其变化量和变化率不在步骤5范围内，该数据需要从步骤4产生的每日汇总数据中获得。

```
    # part3
    if i == 0:  # 如果是顶部节点，则单独增加顶部节点信息
        day_data = day_summary_total[day_summary_total.index == the_day]  # 
得顶部节点的数据
        former_data = day_data.ix[0, 1]  # 获得全站总变化量
        node_lable = top_node_style.format(node_name, int(former_data), day_
别获取顶部节点名称、变化量和变化率
        dot.node(node_name, label=node_lable)  # 增加顶部节点
```

当i为0时，单独增加顶部节点信息。

·day_data为汇总数据的当日记录，包含visit、change、change_rate三列以及日期索引。

·former_data为全站总变化量，用途有2个，一是便于在顶级节点中显示总变化量信息，二是作为下面计算变化量贡献的初始值。

·node_lable为顶级节点显示的信息，基于part1定义的top_node_style使用str.format方法定义出顶级节点完整信息，包括节点名称、变化量和变化率。这里由于是全站级别的数据，因此可以直接使用

node_change；也可以使用day_data.ix[0，1]获取。

·通过dot.node（node_name，label=node_lable）方法单独增加顶级节点，重点是使用label方法将上面定义的顶级节点完整信息通过标签显示出来。

4）part4分别定义左侧和右侧边显示的贡献率信息。这里的贡献率定义为分裂下级节点的变化量与上级分裂节点的变化量的比值。正向的贡献率结果为正，负向的贡献率结果为负。

这里没有用到整个维度的总变化量（即node_change），而是计算关键节点直接的总占比。

```
    # part4
    contribution_rate_1 = float(max_value) / former_data  # 获得左侧变化量贡献
率
    contribution_rate_2 = float(min_value) / former_data  # 获得右侧变化量贡献
率
    if node_change >= 0:  # 如果为增长，则左侧为正向
                                        edge_lablel_left     =     '正向贡献率:
{0:.0%}'.format(contribution_rate_1)  # 左侧边的标签信息
                                       edge_lablel_right     =     '反向贡献率:
{0:.0%}'.format(contribution_rate_2)  # 右侧边的标签信息
        former_data = max_value  # 获得上一层级变化量最大值
    else:  # 如果为下降，则右侧为正向
                                        edge_lablel_left     =     '反向贡献率:
{0:.0%}'.format(contribution_rate_1)  # 左侧边的标签信息
                                       edge_lablel_right     =     '正向贡献率:
{0:.0%}'.format(contribution_rate_2)  # 右侧边的标签信息
        former_data = min_value  # 获得上一层级变化量最大值
```

contribution_rate_1：定义左侧变化量贡献率。这里使用float（max_value）/former_data而非两个变量直接除的原因是这两个变量都是整数型，如果直接除的结果将无法显示浮点型数据。

contribution_rate_2：定义右侧变化量贡献率，该变量定义方法与contribution_rate_1类似，差异点在于分子使用的是min_value。

接下来通过node_change来判断正负方向，目的是确认分裂节点沿左侧还是右侧分裂。

·当node_change>=0，则沿左侧分裂。此时通过str.format方法定义左

侧边标签edge_lablel_left，同理定义出右侧边标签edge_lablel_right。

·当node_chang<0，则沿右侧分裂。此时使用跟上述类似的方法定义左右两侧的边的标签信息。

至此，我们已经定义好了左侧节点、左侧边，右侧节点、右侧边的全部信息，接下来需要将节点、边都关联起来。

5）part5建立节点、边并输出图形。

```
    # part5
    dot.node(node_left, label=node_label_left)  # 增加左侧节点
    dot.node(node_right, label=node_label_right)  # 增加右侧节点
    dot.edge(node_name, node_left, label=edge_lablel_left, color='chartreuse
加左侧边
    dot.edge(node_name, node_right, label=edge_lablel_right, color='lightblu
加右侧边

dot.view('change summary')  # 展示图形结果
```

跟定义顶级节点的方法类似，使用dot.node定义左右两侧的节点，同时设置节点显示的label（标签）值。

使用dot.edge定义左侧边与右侧边，并设置label（标签）值和颜色值。为了对应，我们将边的颜色跟节点内使用HTML设置的颜色相同。

最后使用dot.view（）方法显示最终图形，该方法与之前用到过的dot.render（view=True）的结果是一样的，可以看成是其缩写版本。另外，该方法支持的参数如下：

```
view(self, filename=None, directory=None, cleanup=False)
```

·filename：展示文件的名称，本节中设置为change summary。

·directory：生成展示结果对应的数据源和图像的路径，默认为空意味着是当前路径。

·cleanup：是否清除生成展示结果对应的数据源文件。

上述代码完成后在当前Python工作路径下会产生2个文件：

·change summary：该文件没有扩展名，是上述生成展示结果对应的数据源文件。

·change summary.png：目标保存的图像文件，也是预览文件（7.10.1节图7-17所示图形），生成图像后系统会默认调用该对象的对应程序打开预览。

7.11.5　案例数据结论

本案例的核心在change summary.png中的信息，该图包含了5层分析维度，依次从全站、source、site、channel、media做层层细分。以第三层级到第四层级下探分支为例，如图7-20所示。

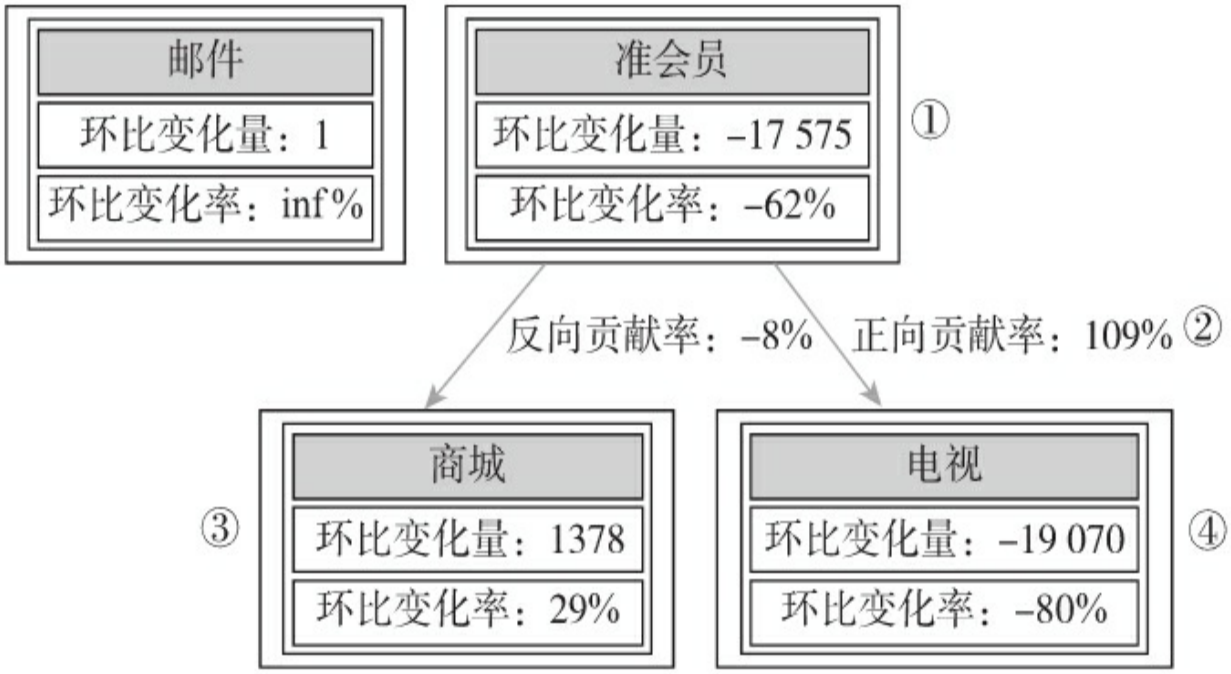


图7-20　第三层级向第四层级细分

图中包括四类信息体：

·第一类信息体：分裂节点，如图中的①，分裂节点是影响上一层变化的正向贡献最大的节点，数据包含环比变化量和环比变化率。

·第二类信息体：正向边和负向边信息，如图中的②，两个边都计算了下一层极变化量跟上一层次分裂节点变化量的比值。正向边为正值、负向边为负值，这两个值分别代表了下一个分支节点对于分裂节点的变化贡献率。

·第三类信息体：负向贡献因子节点，如图中的③，该因子是在跟整体相反趋势或贡献最小的因子，数据包括环比变化量和环比变化率。

·第四类信息体：正向贡献因子节点，如图中的④，该因子是导致分裂节点变化的主要贡献因素，数据包括环比变化量和环比变化率。

这里需要注意两个点：

·一是右侧贡献节点的环比变化量与上一层级的环比变化量数据不一致，并且其贡献率不一是100%。

·二是分裂节点左右两侧的变化量的总和不等于上层级的分裂节点的变化量的总和。

以上两个现象的产生原因是分裂节点下一层级的左右分支，仅仅是分裂节点信息的一部分，而非全部。以图7-20的第三层级分裂节点为例，在准会员的-17575的变化量中，电视是下降最主要的部分，除此以外还有其他的上升和下降量，例如商城、手机等维度。因此，我们会看到电视对准会员的正向贡献率会超过100%，原因是在反向节点（有增加值的节点）上会有正向值抵消一部分下降量；这部分抵消的信息因子，也是我们在做日常分析中需要从整体趋势中发掘的亮点或问题点。

参照7.10.1节图7-17的信息，我们得到如下结果：

全站访问量下降18455，下降比例达到23%，主要的source源是CRM，其下降量为17591，“贡献”了95%的主要因素；而导致CRM下降的主要site源是准会员，其下降量为17575，“贡献了”几乎100%的下降因素；再进一步细分，在影响准会员的channel中，电视源的下降量达到19090，“贡献”了109%的比例，而导致电视流量下降的主要media源是APP，其贡献了19024的下降流量，比例几乎是100%。与此同时，某些来源渠道的流量与全站的下降趋势相反，呈现良好的增长趋势，这在全站的下降主要因子中表现良好，包括source源中的公众号流量环比增长2444，增长率达到171%；s准会员中的商城部分的流量增长1378，增长率为29%。

在上述结论中，针对每个节点（含分裂节点和非分裂节点）我们都有两个方向的参照：

·横向因子，即查询哪些特征对于上一层级的变化有主要的正向和负向贡献，这是贡献率的来源；

·纵向因子，即查询每个节点本身相对前1天的环比变化率，这是其本身随着时间的变化特征，能有效了解其自身波动水平。

7.11.6 案例应用和部署

本案例是一个非常实用的应用，它已经部署到笔者之前所在公司的日常应用中，作为日常数据报告内容主要波动原因探查的主要途径和方法。

大多数情况下，我们会针对昨天的数据与前天数据做对比分析。因此，在应用部署时默认设置昨日的数据与前1日数据做对比，即在代码中的

```
        the_day       =       pd.datetime(2017,6,7)       #       指定要分析的日期替换为
        the_day = datetime.date.today()-datetime.timedelta(days=1)
```

这样系统就会默认指定昨天作为分析日期与前天的数据做对比。因此，每天早晨上班时只需跑一下程序即可。

7.11.7 案例注意点

本案例虽然思路简单，但在实现过程中仍然需要读者避开一些坑：

·分裂节点的左右分支的数量总和不等于分裂节点的变化量，这点跟决策树规则得到的结果逻辑不同。这里查询的仅仅是主要因子的贡献度。

·由于整体数据增长和整体数据下降需要不同的分裂因子以及沿不同边做下探，因此需要分开获取数据并展示。

·对于案例中使用上一节点的变化量作为正向贡献率的分母，如果读者有疑问，可以使用上一节点对应的波动总值作为分母，其值存储在change_list中，只需在计算contribution_rate_1和contribution_rate_2时使用node_change替换former_data即可。但这样做的坏处是边显示的贡献度跟节点的数据不是一个维度，导致信息判断混乱，因此不建议读者这样应用。

·在label标签定义时，我们使用了类似HTML的语法，之所以说是类似于，是因为graphviz不完全支持所有HTML格式的语法；并且定义时需要使用“<>”将HTML语法代码括起来，否则代码无法生效。

·node_name和node_left、node_right在分裂时其实是部分重叠的，这也是其能够形成连续节点树的原因。因此node本身就取自于node_left（当整体增长时）或node_right（当整体下降时）。在显示不同节点的信息时，注意区分开node数据与node label数据。

7.11.8 案例引申思考

案例使用的graphviz是一个非常强大的用于展示复杂关系库，我们在4.3.6节和4.4.6节都有用到。该库还有很多我们可以应用到的复杂场景，例如：

·基于网络转发的传播关系图；

·个人关系联络图；

·基于有时间序列的流程图；

·网络拓扑关系图；

·信息流和事件流图。

案例中分裂寻找的是变化量（也意味着变化率）最大的节点，如果有课题需要也可以将变化率或贡献度指定出来，在计算的时候按照指定的贡献率提取出一系列（而不是一个）因子。或者，可以依据业务手工分析的需求，沿着指定维度做层层下探，其实只是在维度循环时从预设维度变为根据系统选择传值而已。整个的实现思路，已经跟决策树的实现思想比较接近了，如果读者兴趣可以参照决策树的实现和优化思路。

7.12 案例：基于自动K值的KMeans广告效果聚类分析

7.12.1 案例背景

某企业由于投放的广告渠道比较多，需要对其做广告效果分析以实现有针对性的广告效果测量和优化工作。跟以应用为目的的案例不同的是，由于本案例是一个分析型案例，该过程的输出其实是不固定的，因此需要跟业务运营方具体沟通需求。

以下是在开展研究之前的基本预设条件：

·广告渠道的范畴是什么？具体包括哪些渠道？——所有站外标记的广告类渠道（以ad_开头）。

·数据集时间选择哪个时间段？——最近90天的数据。

·数据集选择哪些维度和指标？——渠道代号、日均UV、平均注册率、平均搜索量、访问深度、平均停留时间、订单转化率、投放总时间、素材类型、广告类型、合作方式、广告尺寸、广告卖点。

·专题分析要解决什么问题？——将广告分类并找出其重点特征，为接下来的业务讨论和数据分析提供支持。

明确了上述具体需求后，下面开始案例的主要工作部分。本节案例的输入源数据ad_performance.txt和源代码chapter7_code2.py位于“附件-chapter7”中，默认工作目录为“附件-chapter7”（如果不是，请cd切换到该目录下，否则会报“IOError:File ad_performance.txt does not exist”）。程序的输出为不同聚类类别的详细信息数据以及雷达图。

7.12.2 案例主要应用技术

本案例用到的主要技术包括：

·数据预处理：数据标准化、字符串分类转整数型分类。

·数据建模：KMeans聚类算法。

·数据展示：使用matplotlib输出雷达图。

主要用到的库包括：sys、numpy、pandas、sklearn、matplotlib，其中sklearn是展示数据的核心。

KMeans聚类在4.1节已经介绍过，本案例的重点有两个：

一是如何基于最优数据尺度确定最佳K值，该技术实现的思路是：最优佳的聚类类别划分从数据特征上看是类内距离最小化的同时类间的距离最大化，可以使用平均轮廓系数作为指标评估，通过枚举每个K计算平均轮廓系数并得到最优值。

二是通过极坐标系的设置方式输出雷达图。

7.12.3 案例数据

案例数据来自某企业的营销部门数据，该数据基于营销数据、网站分析系统数据和运营系统数据总结而来。以下是数据概况：

·维度数：除了渠道唯一标记之外，共12个维度。

·数据记录数：889。

·是否有NA值：有。

·是否有异常值：有。

以下是本数据集的13个字段的详细说明：

·渠道代号：业务方统一命名规划的唯一渠道标志。

·日均UV：每天的平均独立访客，从一个渠道中带来的一个访客即使一天中到达多次都统计为1次。

·平均注册率：日均注册的用户数量/平均每天的访问量。

·平均搜索量：平均每个访问的搜索次数。

·访问深度：总页面浏览量/平均每天的访问量。

·平均停留时间：总停留时间/平均每天的访问量。

·订单转化率：总订单数量/平均每天的访问量。

·投放总时间：每个广告媒介在站外投放的天数。

·素材类型：广告素材类型，包括jpg、gif、swf、sp。

·广告类型：广告投放类型，包括banner、tips、横幅、通栏、暂停以及不确定（不知道到底是何种形式）。

·合作方式：广告合作方式，包括roi、cpc、cpm和cpd。

·广告尺寸：每个广告投放的尺寸大小，包括140*40、308*388、450*300、600*90、480*360、960*126、900*120、390*270。

·广告卖点：广告素材上主要的卖点诉求信息，包括打折、满减、满赠、秒杀、直降、满返。

7.12.4 案例过程

步骤1 导入库。

```
import sys
reload(sys)
sys.setdefaultencoding('utf-8')
import pandas as pd
import numpy as np
from sklearn.feature_extraction import DictVectorizer  # 字符串分类转整数分类库
from sklearn.preprocessing import MinMaxScaler  # MinMaxScaler库
from sklearn.cluster import KMeans  # KMeans模块
from sklearn import metrics  # 导入sklearn效果评估模块
import matplotlib.pyplot as plt  # 图形库
```

本案例主要用到了以下库：

·sys：系统库，用来将默认编码设置为UTF-8，目的是处理中文处理。

·numpy：基本数据处理。

·pandas：数据读取、审查、异常值处理等。

·DictVectorizer：用来将字符串分类转换为整数型分类。

·MinMaxScaler：数据标准化库，用来将数据做标准化处理。

·KMeans：K均值聚类算法模块。

·metrics：sklearn效果评估模块。

·matplotlib.pyplot：用来做雷达图展示。

步骤2 读取数据，使用pandas的read_table方法读取txt文件，分隔符为TAB。

```
raw_data = pd.read_table('ad_performance.txt', delimiter='\t')
```

步骤3　数据审查和校验，该步骤包括多个环节：查看前2条数据、查看数据类型分布、查看缺失值、查看数据描述性统计信息、相关性分析等。

使用head（2）方法查看前2条数据：

```
print ('{:*^60}'.format('Data overview:'))
print (raw_data.head(2))  # 打印输出前2条数据
```

从返回结果看，原始数据能正常识别且没有异常信息，前2条数据如下：

```
***********************Data overview:***********************
渠道代号日均UV     平均注册率平均搜索量访问深度平均停留时间订单转化率投放总时间素材类型广告类型  \
0  A203    3.69  0.0071  0.0214  2.3071  419.77  0.0258   20.0  jpg  banner
1  A387  178.70  0.0040  0.0324  2.0489  157.94  0.0030   19.0  jpg  banner
合作方式广告尺寸广告卖点
0  roi  140*40   打折
1  cpc  140*40   满减
```

使用dtypes方法查看数据类型分布，由于行比较多，因此这里使用pd.DataFrame方法将其结果转换为数据框，然后通过.T做转置。

```
print ('{:*^60}'.format('Data dtypes:'))
print (pd.DataFrame(raw_data.dtypes).T)  # 打印数据类型分布
```

从数据类型结果来看，各个字段类型都能别正确识别，结果如下：

```
************************Data dtypes:************************
渠道代号日均UV     平均注册率平均搜索量访问深度平均停留时间订单转化率投放总时间  \
0  object  float64  float64  float64  float64  float64  float64  float64
素材类型广告类型合作方式广告尺寸广告卖点
0  object  object  object  object  object
```

使用isnull（）.sum（）查看所有字段中具有缺失值数据的统计量，同样使用pd.DataFrame将结果转换为数据框，然后通过.T做转置。

```
print ('{:*^60}'.format(' NA counts:'))
print (pd.DataFrame(raw_data.isnull().sum()).T)  # 查看缺失值情况
```

从结果中发现，在“平均停留时间”字段中有2个缺失值。结果如下：

```
************************ NA counts:************************
渠道代号日均UV     平均注册率平均搜索量访问深度平均停留时间订单转化率投放总时间素材类型广告类型合作方式 \
0      0      0       0       0      0        2       0       0      0      0      0
广告尺寸广告卖点
0      0      0
```

使用describe（）查看数据描述性统计信息，并使用round（2）保留2位小数，最后使用.T将其转置。

```
print ('{:*^60}'.format('Data DESC:'))
print (raw_data.describe().round(2).T)  # 打印原始数据基本描述性信息
```

如下的描述性统计结果中，反映了3个信息点：

·日均UV的数据波动非常大，说明了不同渠道间的特征差异非常明显。

·平均停留时间的有效数据（非空数据）只有887，比其他数据少2条，这也印证了上述缺失值统计结果。

·平均注册率、平均搜索量、订单转化率的多个统计量（例如最小值、25%分位数等）都为0，看似数据不太正常。

```
************************Data DESC:************************
          count    mean      std   min     25%     50%     75%       max
日均UV    889.0  540.85  1634.41  0.06    6.18  114.18  466.87  25294.77
平均注册率   889.0    0.00     0.00  0.00    0.00    0.00    0.00      0.04
平均搜索量   889.0    0.03     0.11  0.00    0.00    0.00    0.01      1.04
访问深度    889.0    2.17     3.80  1.00    1.39    1.79    2.22     98.98
平均停留时间  887.0  262.67   224.36  1.64  126.02  236.55  357.98   4450.83
订单转化率   889.0    0.00     0.01  0.00    0.00    0.00    0.00      0.22
投放总时间   889.0   16.05     8.51  1.00    9.00   16.00   24.00     30.00
```

以上三类异常点，经过跟业务方的沟通以及再次数据验证，其结果如下：

·日均UV的差异性问题：由于广告的流量型特征，很多广告的流量爆发明显，因此渠道间确实带有非常大的差异性，这些差异性应该保留，不能作为异常值处理。

·平均停留时间的缺失值，该字段由于统计缺失导致数据丢失，可以使用均值法做填充。

·平均注册率、平均搜索量、订单转化率等的多个字段为0的问题，这是由于在打印输出过程中仅保留了2位小数，而这几个统计量的数据本身就非常小，将其通过round（3）保留3位小数后就能正常显示。

使用corr（）方法做相关性分析：

```
print ('{:*^60}'.format('Correlation analysis:'))
print (raw_data.corr().round(2).T)  # 打印原始数据相关性信息
```

通过相关性结果分析，12个特征中平均停留时间和访问深度的相关系数为0.72，这两个指标具有较高的相关性，但特征也不是非常明显；其他特征之间的相关性关系都不突出。

步骤4　数据预处理，本步骤主要涉及缺失值替换、字符串分类转换为整数分类、数据标准化、数据合并等操作。

使用fillna方法将“平均停留时间”中的缺失值替换为均值：

```
data_fillna = raw_data.fillna(raw_data['平均停留时间'].mean())  # 用后面的值替换缺失值
```

字符串分类转整数分类

该内容在6.8节介绍过，主要区别在于这里由于没有“预测”环节，因此无需区分训练阶段和预测阶段。主要实现方法如下：

1）part1定义要转换的数据。

```
conver_cols = ['素材类型', '广告类型', '合作方式', '广告尺寸', '广告卖点']
convert_matrix = data_fillna[conver_cols]  # 获得要转换的数组
lines = data_fillna.shape[0]  # 获得总记录数
```

```
dict_list = []  # 总空列表，用于存放字符串与对应索引组成的字典
unique_list = []  # 总唯一值列表，用于存储每个列的唯一值列表
```

·通过convert_cols定义转换的列名，并在此基础上新建转换数据集convert_matrix。

·通过data_fillna.shape[0]获取总记录数，便于按行做循环。

·建立一个空列表dict_list用于存储字符串与对应索引的字典。由于我们要使用sklearn的DictVectorizer方法做转换，它要求转换对象是一个由字典组成的列表，字典的键值对是字符串及其对应的数字映射，我们的目的是从唯一字符集中取出其value和对应的index作为该键值对的组合。

·总空列表dict_list，用于存放字符串与对应索引组成的字典。

·总唯一值列表unique_list，用于存储每个列的唯一值列表。

2）part2使用for获得所有列的唯一值列表，然后存到dict_list中。

```
for col_name in conver_cols:  # 循环读取每个列名
    cols_unqiue_value = data_fillna[col_name].unique().tolist()  # 获取列的唯一值列表
    unique_list.append(cols_unqiue_value)  # 将唯一值列表追加到总列表
```

循环读取每个列名，单独获取该列数据并使用unique方法获取唯一值，由于该唯一值的结果是一个数组，因此需要使用tolist方法转换为列表，便于后期应用。最后将每个列的唯一值列表追加到总的唯一值列表中。

3）part3使用for循环将每条记录的具体值跟其在唯一值列表中的索引做映射。

例如，唯一值结果集是['1'，'2'，'3'，'4'，'5']，如果该列中有一个值为'4'，那么需要将该字符串转换为其对应的索引值3（注意索引从0开始）。

```
for line_index in range(lines):  # 读取每行索引
```

```
        each_record = convert_matrix.iloc[line_index]   # 获得每行数据，是一个Series
    for each_index, each_data in enumerate(each_record):  # 读取Series每行对应的索引值
            list_value = unique_list[each_index]  # 读取该行索引对应到总唯一值列表列索引下的数据(其实是相当于原来的列做了转置成了行，目的是查找唯一值在列表中的位置)
            each_record[each_index] = list_value.index(each_data)  # 获得每个值对应到总唯一值列表中的索引
    each_dict = dict(zip(conver_cols, each_record))  # 将每个值和对应的索引组合字典
    dict_list.append(each_dict)  # 将字典追加到总列表
```

由于转换是按行实现的，因此使用for循环读取每行索引，然后使用iloc方法获取对应行数据each_record，该数据是一个Series格式的列表；下面要做的是将每个列表中的value映射到唯一值列表中的index。

再使用一个for循环结合enumerate读取列表的每个值及对应索引，索引结合unique_list[each_index]用于从唯一值总列表中找到原始所处的列的唯一列表，值用于从unique_list[each_index]中匹配出值对应的索引，使用的是列表的index（value）方法，得到的索引再替换掉原始字符串数据。该子循环结束后，每条记录已经是转换为数值型分类的列表，使用dict结合zip方法将其与列名转换为字典。

上述循环完成后，我们已经在dict_list中存储了所有的数据记录，列表内的每个元素是一个以唯一值和对应索引值组成的字典。

4）part4使用DictVectorizer将字符串转换为整数。

```
model_dvtransform = DictVectorizer(sparse=False, dtype=np.int64)  # 建立转换模型对象
data_dictvec = model_dvtransform.fit_transform(dict_list)  # 应用分类转换训练
```

先建立DictVectorizer转换模型对象model_dvtransform，这里设置了2个参数，sparse=False指定转换后的数据集是一个数组，否则默认会是压缩后的稀疏矩阵，这样设置的原因是后续很多步骤和模型都不支持直接基于压缩后的稀疏矩阵做转换和建模；dtype=np.int64用于设置转换后的数据类型是整数型，否则默认是浮点型。

对model_dvtransform对象使用fit_tranform方法做转换应用。这里可以分开使用fit，然后再使用tranform方法。

数据标准化。由于不同字段间存在数值的量纲差异，例如日均UV有几万的量级，而转化率的范围却是0～1之间，因此需要做数据标准化，这里使用的是Min-Max标准化方法。

```
sacle_matrix = data_fillna.ix[:, 1:8]  # 获得要转换的矩阵
minmax_scaler = MinMaxScaler()  # 建立MinMaxScaler模型对象
data_scaled = minmax_scaler.fit_transform(sacle_matrix)  # MinMaxScaler标准化处理
```

标准化的实施是针对数值型字段进行的，因此这里通过data_fillna.ix[:，1:8]选择需要标准化的数据。下面的应用过程比较简单，是熟悉的sklearn的应用方法：使用MinMaxScaler（）建立标准化模型对象，然后对模型对象应用fit_transform方法做标准化转换。

合并所有输入维度。在上面的预处理阶段，我们分别针对数值型和字符串型两类输入维度做处理，这里需要将其处理后的结果合并起来。

```
X = np.hstack((data_scaled, data_dictvec))
```

使用numpy的hstack方法，将data_scaled、data_dictvec合并，形成最终输入X。

这里的X不包含渠道的唯一标志列，因为该列没有实际建模意义。

步骤5　通过平均轮廓系数检验得到最佳KMeans聚类模型。

K值的确定一直是KMeans算法的关键，而由于KMeans是一个非监督式学习，因此没有所谓的“最佳”K值。但是，从数据本身的特征来讲，最佳K值对应的类别下应该是类内距离最小化并且类间距离最大化。有多个指标可以用来评估这种特征，比如平均轮廓系数、类内距离/类间距离等都可以做此类评估。基于这种思路，我们可以通过枚举法计每个K下的平均轮廓系数值，然后选出平均轮廓系数最大下的K值。


即使在数据上聚类特征最明显，也并不意味着聚类结果就是有效的，因为这里的聚类结果用来分析使用，不同类别间需要具有明显的差异性特征并且类别间的样本量需要大体分布均衡。而确定最佳K值时却没有考虑到这些“业务性”因素。

```
score_list = list()  # 用来存储每个K下模型的平局轮廓系数
silhouette_int = -1  # 初始化的平均轮廓系数阈值
for n_clusters in range(2, 10):  # 遍历从2到10几个有限组
    model_kmeans = KMeans(n_clusters=n_clusters, random_state=0)  # 建立聚类模型对象
    cluster_labels_tmp = model_kmeans.fit_predict(X)  # 训练聚类模型
    silhouette_tmp = metrics.silhouette_score(X, cluster_labels_tmp)  # 得到每个K下的平均轮廓系数
    if silhouette_tmp > silhouette_int:  # 如果平均轮廓系数更高
        best_k = n_clusters  # 将最好的K存储下来
        silhouette_int = silhouette_tmp  # 将最好的平均轮廓得分存储下来
        best_kmeans = model_kmeans  # 将最好的模型存储下来
        cluster_labels_k = cluster_labels_tmp  # 将最好的聚类标签存储下来
    score_list.append([n_clusters, silhouette_tmp])  # 将每次K及其得分追加到列表
print ('{:*^60}'.format('K value and silhouette summary:'))
print (np.array(score_list))  # 打印输出所有K下的详细得分
print                    ('Best                    K                    is:
{0} with average silhouette of {1}'.format(best_k, silhouette_int.round(4)))
```

该步骤的主要实现过程如下：

定义初始变量score_list和silhouette_int。score_list用来存储每个K下模型的平局轮廓系数，方便在最终打印输出详细计算结果；silhouette_int的初始值设置为-1，每个K下计算得到的平均轮廓系数如果比该值大，则将其值赋值给silhouette_int。


对于平均轮廓系数而言，其值域分布式[-1，1]。因此silhouette_int的初始值可以设置为-1或比-1更小的值。

使用for循环遍历每个K值，这里的K的范围确定为从2～10。一般而言，用于聚类分析的K值不会太大。如果值太大，那么聚类效果可能不明显，因为大量信息的都会被分散到各个小类之中，会导致数据的碎片化。

通过KMeans（n_clusters=n_clusters，random_state=0）建立KMeans

模型对象model_kmeans，设置聚类数为循环中得到的K值，设置固定的初始状态。

对model_kmeans使用fit_predict得到其训练集的聚类标签。该步骤其实无需通过predict获得标签，可以先使用fit方法对模型做训练，然后使用模型对象model_kmeans的label_属性获得其训练集的标签分类。

使用metrics.silhouette_score方法对数据集做平均轮廓系数得分检验，将其得分赋值给silhouette_tmp，输入参数有两个：

·X：原始输入的数组或矩阵。

·cluster_labels：训练集对应的聚类标签。

接下来做判断，如果计算后的得分大于初始化变量的得分，那么：

·将最佳K值存储下来，便于后续输出展示。

·将最好的平均轮廓得分存储下来，便于跟其他后续得分做比较以及输出展示。

·将最好的模型存储下来，这样省去了后续再做最优模型下fit（训练）的工作。

·将最好的聚类标签存储下来，这样方便下面将原始训练集与最终标签合并。

每次循环结束后，将当次循环的K值以及对应的评论轮廓得分使用append方法追加到列表。

最后打印输出每个K值下详细信息以及最后K值和最优评论轮廓得分，返回数据如下：

```
**************K value and silhouette summary:***************
[[ 2.          0.46692821]
 [ 3.          0.54904646]
 [ 4.          0.56968547]
 [ 5.          0.48186604]
 [ 6.          0.45477667]
 [ 7.          0.48204261]
```

```
 [ 8.          0.50447223]
 [ 9.          0.52697493]]
Best K is:4 with average silhouette of 0.5697
```

上述结果显示了不同K下的平均轮廓得分。就经验看，如果平均轮廓得分值小于0，意味着聚类效果不佳；如果值大约0且小于0.5，那么说明聚类效果一般；如果值大于0.5，则说明聚类效果比较好。本案例在K=4时，得分为0.5697，说明效果较好。

步骤6　针对聚类结果的特征分析。

1）part1将原始数据与聚类标签整合。

```
cluster_labels = pd.DataFrame(cluster_labels_k, columns=['clusters'])  # 获得训练集下的标签信息
merge_data = pd.concat((data_fillna, cluster_labels), axis=1)  # 将原始处理过的数据跟聚类标签整合
```

该步骤中先将最佳K值下的cluster_labels_k转换为数据框，然后使用pandas.concat将原始数据与标签合并，axis=1设置为按列合并。

在合并数据集时，这里没有使用数据建模时的X与cluster_labels合并，而是使用了经过缺失值填充后的data_fillna数组做合并，目的是为了在分析时能够再现不同特征下原始值的特征。如果使用X则其标准化的数值则不符合业务的实际数据分布范围，会导致数据难以理解。

2）part2计算每个聚类类别下的样本量和样本占比。

```
clustering_count = pd.DataFrame(merge_data['渠道代号'].groupby(merge_data['clusters']). count()).T.rename(
    {'渠道代号': 'counts'})  # 计算每个聚类类别的样本量
clustering_ratio = (clustering_count / len(merge_data)).round(2).rename({'cc
算每个聚类类别的样本量占比
```

在计算各聚类类别下的样本量时，主要的函数用法如下：

·使用merge_data['渠道代号'].groupby（merge_data['clusters']）.count（）以clusters为维度对渠道代号做计数统计，然后使用pd.DataFrame方法将其转换为数据框。

·使用.T将分类汇总的数据框做转置操作。

·使用rename（{'渠道代号':'counts'}将分类结果数据框的名称改为由渠道代号改为counts，方便之后合并时的信息提示。

在计算各聚类类别样本量占比时，主要的函数用法如下：

·使用len（merge_data）获得样本总量。

·使用clustering_count/len（merge_data）获得各个类别与样本总量的百分比。

·使用round（2）将保留两位小数。

·使用rename（{'counts':'percentage'}将名称由'counts'改为'percentage'。

3）part3计算各个聚类类别内部最显著特征值。

```
cluster_features = []  # 空列表，用于存储最终合并后的所有特征信息
for line in range(best_k):  # 读取每个类索引
    label_data = merge_data[merge_data['clusters'] == line]  # 获得特定类的数据

    part1_data = label_data.ix[:, 1:8]  # 获得数值型数据特征
    part1_desc = part1_data.describe().round(3)  # 得到数值型特征的描述性统计信息
    merge_data1 = part1_desc.ix[2, :]  # 得到数值型特征的均值

    part2_data = label_data.ix[:, 8:-1]  # 获得字符串型数据特征
    part2_desc = part2_data.describe(include='all')  # 获得字符串型数据特征的描述性统计信息
    merge_data2 = part2_desc.ix[2, :]  # 获得字符串型数据特征的最频繁值

    merge_line = pd.concat((merge_data1, merge_data2), axis=0)  # 将数值型和字符串型典型特征沿行合并
    cluster_features.append(merge_line)  # 将每个类别下的数据特征追加到列表
```

建立一个空列表cluster_features，用于存储最终合并后的所有特征信息。

使用for循环读出最佳K下的每个聚类索引值。

使用merge_data[merge_data['clusters']==line]获得特定类的数据。

获得数值型特征值信息。使用label_data.ix[:，1:8]获得数值型数据特征；使用part1_data.describe（）.round（3）得到数值型特征的描述性统计信息，并保留3位小数；使用part1_desc.ix[2，:]得到得到描述性统计结果中的均值。该均值将作为各类的数值型数据的典型特征。

按照同样的方法，获得字符串型数据特征。在应用时有以下不同点：

·ix[:，8:-1]的意思是从第8列到最后一列，而无需指定最后一列的具体索引值。

·describe（include='all'）中通过include='all'指定对所有类型的数据做描述性统计，而非仅仅是数值型数据。

describe可针对数值型和字符串型数据做描述性统计。针对数值型数据的描述性统计字段包括计数、均值、标准差、最小值、25%分位数、50%分位数、75%分位数和最大值，当然也可以定义其他分位数，通过percentiles=[0.1，0.5，0.8]参数即可指定，但需要注意的是区间为0～1之间；针对字符串型的描述性统计由于没有数据可直接做计算，可以提供计数、唯一值数量、频数最多的字符串值及其频数这几个字段。

获得两种类型的典型数据特征后，使用pd.concat方法将数据合并，这里需要注意的是使用axis=0指定沿行合并而非列。使用cluster_features.append（merge_line）将每个类别下的数据特征追加到列表。

4）part4输出完整的类别特征信息。

```
cluster_pd = pd.DataFrame(cluster_features).T  # 将列表转化为矩阵
print ('{:*^60}'.format('Detailed features for all clusters:'))
all_cluster_set = pd.concat((clustering_count, clustering_ratio, cluster_pd)
每个聚类类别的所有信息合并
print (all_cluster_set)
```

该过程主要实现如下：

·使用pd.DataFrame（cluster_features）.T先将列表cluster_features转换为数据框然后做转置。

·使用pd.concat（（clustering_count，clustering_ratio，cluster_pd），axis=0）将类别样本量统计、样本量占比以及各字段典型特征合并起来。

打印输出如下结果：

```
************Detailed features for all clusters:*************
clusters          0        1        2        3
counts          411      297       27      154
percentage        0.46     0.33     0.03     0.17
日均UV          1369.81  1194.69  1263.03   2718.7
平均注册率          0.003    0.003    0.003    0.005
平均搜索量          0.082    0.144    0.151    0.051
访问深度           0.918     5.728      9.8    0.948
平均停留时间       165.094  285.992  374.689   104.14
订单转化率          0.009    0.016    0.017    0.007
投放总时间          8.462     8.57    7.996    8.569
素材类型             swf      jpg      swf      jpg
广告类型不确定不确定通栏    banner
合作方式             cpc      cpc      cpc      cpc
广告尺寸          600*90   600*90  900*120  308*388
广告卖点打折直降打折满减
```

步骤7　各类别显著数值特征对比。

从上面的各类别特征中可能很难直观发现不同类别的显著性特征，这里通过雷达图的形式对各个聚类类别的数值型特征做对比展示。

1）part1各类别数据预处理，由于不同特征的数据量级差异很大，因此需要先对其做标准化处理，这样才能使得不同数据间具有对比的可能性。

```
num_sets = cluster_pd.ix[:6, :].T.astype(np.float64)  # 获取要展示的数据
num_sets_max_min = minmax_scaler.fit_transform(num_sets)  # 获得标准化后的数据
```

先通过cluster_pd.ix[:6，:].T.astype（np.float64）获取要展示的数据，其中：

·ix[:6，:]：代表获取前6列数据；

·T：表示对其做转置；

·astype（np.float64）：表示将数据转换为数值型。

使用在数据预处理阶段建立的min-max对象minmax_scaler的fit_transform方法将数据做0-1标准化处理。

2）part2画布基本设置。

```
fig = plt.figure()  # 建立画布
ax = fig.add_subplot(111, polar=True)  # 增加子网格，注意polar参数
labels = np.array(merge_data1.index[:-1])  # 设置要展示的数据标签
cor_list = ['r', 'g', 'b', 'y']  # 定义不同类别的颜色
angles = np.linspace(0, 2 * np.pi, len(labels), endpoint=False)  # 计算各个区间的角度
angles = np.concatenate((angles, [angles[0]]))  # 建立相同首尾字段以便于闭合
```

先使用plt.figure（）创建一个画布对象fig，该画布对象会在下面用于增加子网格对象。

使用fig.add_subplot（111，polar=True）为fig对象增加一个子网格，其中参数polar=True用来设置该子网格对象显示极坐标系。

np.array（merge_data1.index[:-1]）设置要展示的数据标签，标签从merge_data1的索引中获取。

定义不同类别的颜色列表cor_list，分别代表红色、绿色、蓝色、黄色。

通过np.linspace（0，2*np.pi，len（labels），endpoint=False）计算各个区间的角度，由于要在雷达图中显示多个类别，这里需要按照类别将整个“圆”按照类别数平均划分。其中各个参数如下：

·0：创建间隔区间的起始。

·2*np.pi：创建间隔区间的末尾。

·len（labels）：间隔长度。

·endpoint：值设置False代表间隔的最后一个值不是间隔区间的末尾。

最后使用angles=np.concatenate（（angles，[angles[0]]））建立相同首尾字段以便于闭合。默认情况下，每个类别的多个特征在雷达图中首尾应该是相连的，这样才能形成闭合效果，因此这里将第一个只追加到列表末尾。

np.concatenate是实现数组合并的常用方法，之前我们经常使用的数组合并方法是hstack和vstack分别用于按列合并、按含合并，也可以使用dstack按第三个轴合并。concatenate方法则相当于是这些合并功能的“合体”，可以通过axis来指定合并的轴。

相关知识点：极坐标系

极坐标系是指在平面内由极点、极轴和极径组成的坐标系。以图7-21为例，图中在平面上取定一点O，称为极点；从O出发引一条射线OX，称为极轴；对于该空间中的任意一点M，其中ρ为M到O长度，称为极径。OM与OX的夹角称为θ。空间中任意一个点，都可以通过（ρ，θ）的有序对的形式表示，而这样建立起来的坐标系叫做极坐标系。

极坐标系是一个二维坐标系，与我们日常应用的另外一个坐标系——平面直角坐标系（也称为笛卡儿坐标系）可以相互转换。对于图7-21中的M（ρ，θ）而言，对应到平面直接坐标系中的x和y分别为：

$x=\rho\cos\theta$

$y=\rho\sin\theta$

当x不等于0时，上述公式可以转换为：

$\theta=\arctan(y/x)$

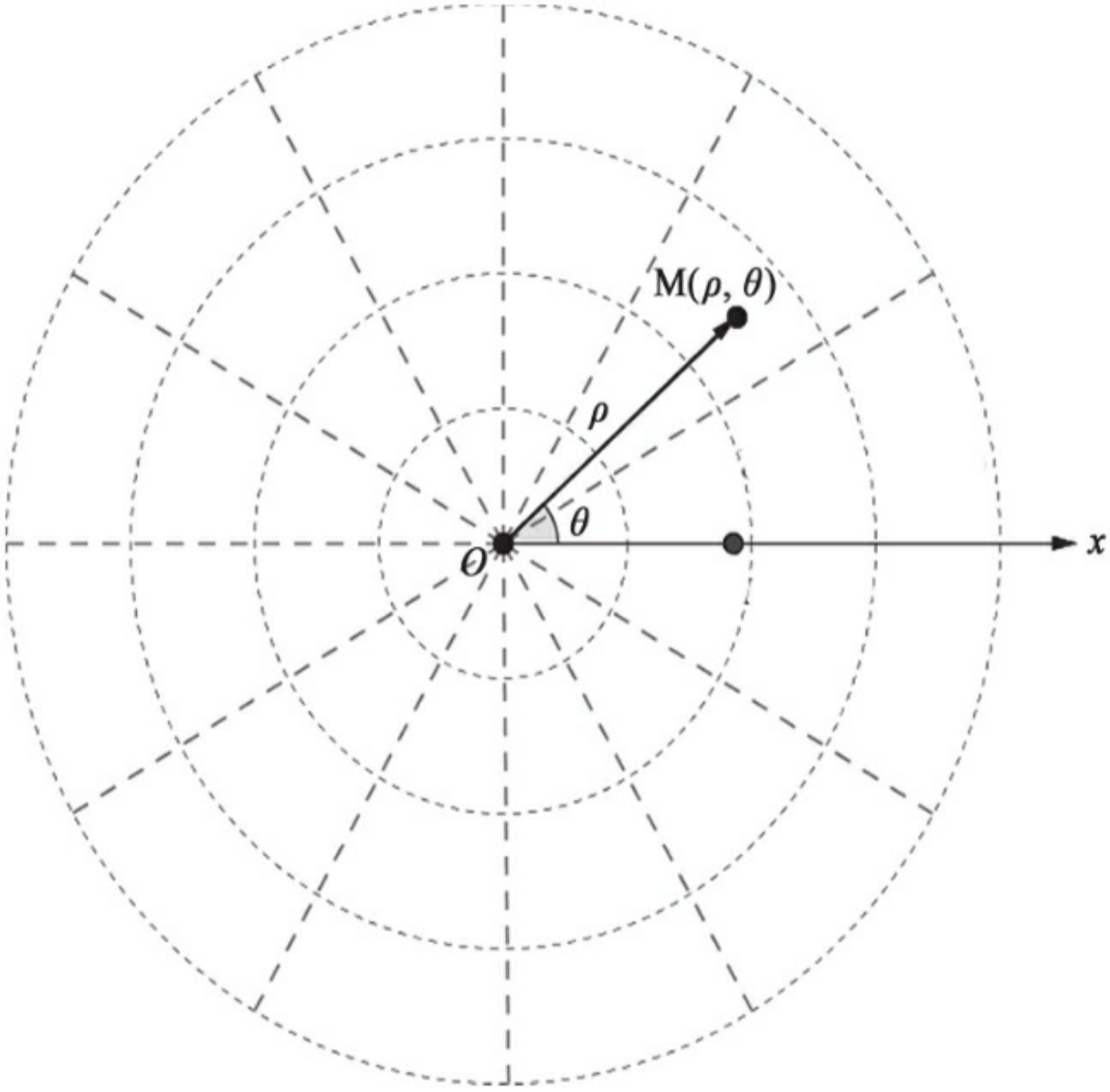


图7-21　极坐标系

在x=0的情况下：如果y为整数，那么θ=90埃蝗绻鹾为负数，那么θ=270啊？

提示 极坐标轴在很多场景下都会用到，例如使用opencv做图像处理时，就会涉及极坐标和平面直角坐标的转换问题。

3）part3画雷达图。

```
for i in range(len(num_sets)):  # 循环每个类别
    data_tmp = num_sets_max_min[i, :]  # 获得对应类数据
     data = np.concatenate((data_tmp, [data_tmp[0]]))  # 建立相同首尾字段以便于闭合
    ax.plot(angles, data, 'o-', c=cor_list[i], label=i)  # 画线
```

该过程通过for循环遍历每个类别索引，然后使用num_sets_max_min[i，:]获得对应类数据，再通过np.concatenate（（data_tmp，[data_tmp[0]]））建立相同首尾字段以便于闭合，最后通过ax.plot（angles，data，'o-'，c=cor_list[i]，label=i）画出雷达图，其中的参数：

·angles：“X”轴数据。

·data：“Y”轴数据。

·'o-'：线条样式，直线并在交点中显示圆。

·c=cor_list[i]：设置颜色。

·label=i：设置每个类别的标识索引。

4）part4设置图像显示格式。

```
ax.set_thetagrids(angles * 180 / np.pi, labels, fontproperties="SimHei")  # 置极坐标轴
ax.set_title("各聚类类别显着特征对比", fontproperties="SimHei")  # 设置标题放置
ax.set_rlim(-0.2, 1.2)  # 设置坐标轴尺度范围
plt.legend(loc=0)  # 设置图例位置
plt.show()  # 展示图像
```

使用ax.set_thetagrids（angles*180/np.pi，labels，fontproperties="SimHei"）设置极坐标系，其中：

·angles*180/np.pi：代表角度。

·labels：极坐标轴的标签。

·fontproperties：设置字体为黑体（SimHei）。

通过ax.set_title（）设置图像标题，通过ax.set_rlim（-0.2，1.2）设

置坐标轴尺度范围，方便显示在各类别极值下的显示效果，通过plt.legend（loc=0）设置图像位置为自动，最后显示图像如下：

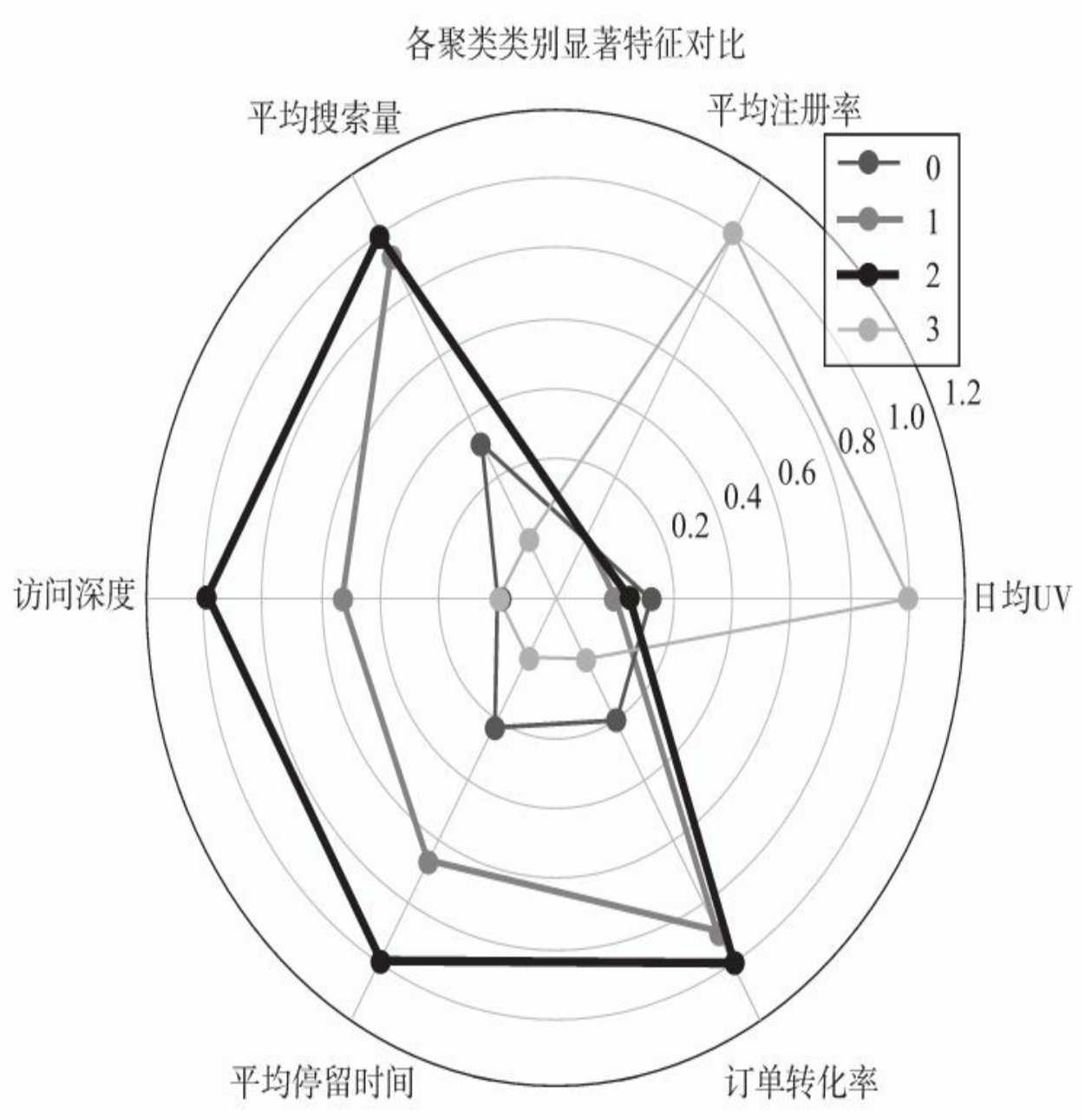


图7-22　各聚类类别显著特征对比

7.12.5 案例数据结论

（1）初步分析

从案例结果来看，所有的渠道被分为4个类别，每个类别的样本量分别为411、297、27、154，对应的占比分别为46%、33%、3%、17%。第三个类别样本量较少。

结合图7-22的雷达图，可以清晰地看到不同类别广告媒体的特征倾向：

聚类1（索引值为0）：各方面的特征都不明显，换句话说就是效果比较平庸，没有明显的优势或短板。但这些“中庸”的广告媒体却构成了整个广告的主体。

聚类2（索引值为1）：这类广告媒体在访问深度、平均停留时间、订单转化率以及平均搜索量等流量质量的特征上的表现较好，除了注册转化率较低外，该类渠道各方面比较均衡。更重要的是该类媒体的数量占据了33%的数量，因此是一类规模较大且综合效果较好的媒体。

聚类3（索引值为2）：这类广告媒体跟聚类2非常类似，并且相对聚类2的典型特征表现更好，但综合其只占3%的媒体数量，属于少量的“精英”类渠道。

聚类4（索引值为3）：这类渠道跟其他几类渠道有个明显的特征区隔，其日均UV和平均注册率非常突出，证明这是一类“引流”+“拉新”的渠道；而其他的流量质量方面的表现却比较差。

（2）深入分析

综合初步分析的结果，再结合输出各渠道详细数据：

聚类1的广告渠道各方面表现均比较一般，因此需要业务部门重点考虑其投放的实际价值。

聚类2的广告渠道的短板是日均UV和平均注册率，因此该类媒体无

法为企业带来大量的流量以及新用户。这类广告的特质适合用户转化，尤其是有关订单转化的提升。

聚类3的广告渠道跟聚类2的特质非常类似，也适合做订单转化的提升，并且由于其实际效果更好，因此效果会更佳明显。

聚类4的广告渠道更佳符合广告本身“广而告之”的基础诉求，因此适合在大规模的广告宣传和引流时使用，尤其对于新用户的注册转化上的效果非常明显，也适合“拉新”使用。

7.12.6 案例应用和部署

对于以上四类广告媒体，需要针对其不同的特征做针对性的运营应用：

对于聚类1的广告媒体，在资金不足或优化广告结构时，重点考虑其取舍问题。

对于聚类2和聚类3的广告媒体，优化应用的方向有两个：

·一是增加对于注册转化效果的优化效应，重点从人群匹配、注册引导、注册激励等方面加强运营。在该过程中，打折、直降等优惠宣传点可以重点突出，广告素材本身以较大中等广告为主，例如900*120。

·二是对于整体广告效果的支撑价值。考虑到其流量规模的局限性，这类渠道更加适合在广告结构中作为一类具有流量质量支撑价值的角色，可以用来提升全局广告的订单质量效果，因此是一类非常关键的质量渠道。尤其是聚类3的广告媒体其质量指标表现非常优异，应该优先考虑投放组合。

对于聚类4的广告媒体，其可以作为在每次大型促销活动时做引流的骨干，因为这更符合广告本身在全部推给媒体中的角色定位。对于这类媒体的单位流量成本需要重点关注，实际投放过程过程中可以满减为促销点、通过较大尺寸（如308*338）的广告类型做引导流量点击。

广告流量数量和质量是广告效果的“两条腿”，虽然大多数情况下广告更侧重于流量数量方面，但对于质量如果能够兼顾则可以锦上添花。因此对于广告结构中两类不同倾向的媒体，具体如何组织和结构调整，更多的要看营销中心自身的核心目标定位。对于广告效果不好的媒体（例如聚类1），也不一定就是效果不好就要切掉，这里面可能涉及多方的利益关联问题。

我们知道，聚类作为数据探索的初步阶段，还可以为后期的深入分析提供方向，相关的深入分析方向包括：

·通过分类方法基于不同的类别划分，找到各自类别内的显著特征，而非描述性统计特征，这些特征会成为进一步落地动作确认的基础。

·结合特定的广告运营目标并将其作为目标变量，通过分类方法找到各个广告渠道对于目标转化与否的重要性影响程度。例如平均注册率、订单转化率等。

·结合不同类别的订单详细信息，将其购买的商品数据提取出来，分析其平均订单金额、购买频率、商品购买数量，以实现对不同类别渠道购买力、消费潜力的评估。

·将广告成本与广告效果评估指标结合起来，通过回归方法预测不同成本下能够达成的广告效果，可以作为广告媒体策划的基础。

7.12.7 案例注意点

本案例有以下内容需要读者注意：

·异常值对于聚类效果的影响是显著的，但是广告类媒体的一个典型特征就是流量规模差异非常大，因此对于这类的异常值则不能轻易去除，因此异常值属于“正常现象”。

·在做描述性统计时，一般情况下会将数值型和字符串型的字段分开做统计。如果放在一起统计，虽然数据是合并到一起的，但由于两种类型的数据结果字段差异非常大，会导致很多字段为空，实际效果并不直观。

·在做聚类之前的数据标准化方法选择上，由于数据中涉及异常值的存在，因此使用的是可以兼顾异常值的处理方法，具体方法我们在3.1节的代码实操里面详细介绍过。

·Matplolib中如果要显示中文，需要设置对应的中文字体名称，否则无法正常显示中文。

7.12.8 案例引申思考

本案例中通过平均轮廓系数的方法得到的最佳K值不一定在业务上具有明显的解读和应用价值。如果最佳K值的解读无效怎么办？有两种思路：

·扩大K值范围，例如将K的范围调整为[2，12]，然后再次运算看更大范围内得到的K值是否更加有效并且能符合业务解读和应用需求。

·得到平均轮廓系数“次要好”（而不是最好）的K值，再对其结果做分析。

对于不同类别的典型特征的对比，除了使用雷达图直观地显示外，还可以使用多个柱形图的形式，将每个类别对应特征的值做柱形图统计，这样也是一个非常直观的对比方法。具体可参考图7-23。

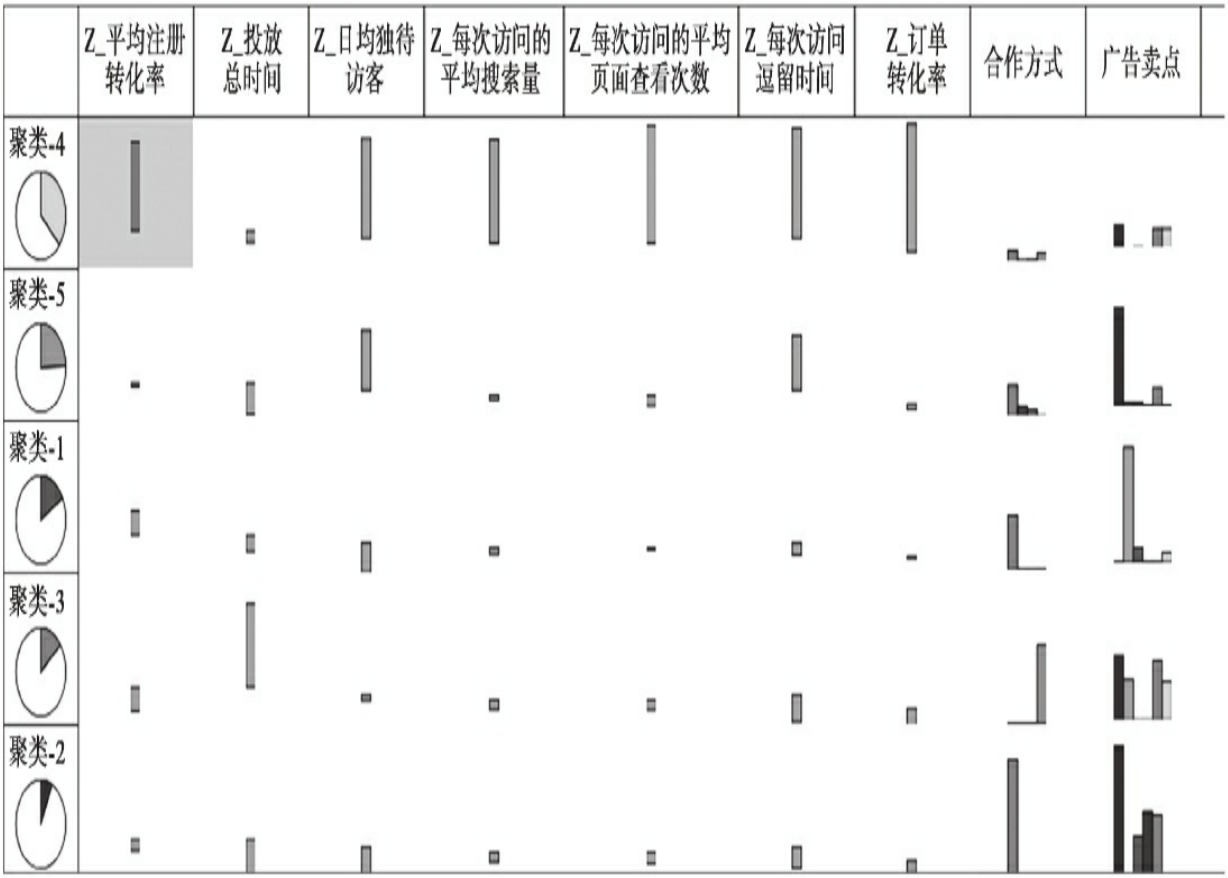


图7-23　不同类别多个指标对比显示

7.13 本章小结

内容小结：本章的内容除了沿袭第5章和第6章的思路外，新增了有关流量分析工具、流量采集分析系统的工作机制和流量数据与企业数据的整合等3个章节的内容介绍，原因是流量的数据采集和分析包括自主实施和使用第三方工具两部分，而更多时候主流的思路是直接使用第三方工具；而流量数据由于自身的特点，通常无法直接跟企业自身数据做完整集成，因此如何整合、整合哪些信息会成为整合的策划关键。本章的内容，希望能对传统的网站数据分析的思路和想法做更多拓展和应用。

另外，在企业在自主实施流量数据的跟踪、采集和分析时，往往不能取得良好效果，表现在工具本身的性能差，无法支持实时或准实时的查询，分析工具无法支持多维度的下钻和分析，工具无法提供多路径、归因等复杂模型的计算方法等。出现这种问题的根源在于流量数据工作是一个非常“费事”的工作，其需要耗费大量的人力、物力和服务器等软硬件资源，但在实施过程中需要专业的网站数据工作专家参与才能有较好的产品和数据产出，但这类的专家在中国还比较少；即使有专家一同参与完成这些工作，也由于流量数据工作无法产生较大的“显性”价值，而导致其价值认同度较低，因此也很难投入大量企业资源。

随着流量成本的提高，高质量的流量媒体越来越分散，相信以精细化运营为基础导向的流量运营工作，必然会增加对数据的依赖程度。与此同时，借助更多更深入的研究方法，流量数据分析也更能提升其自身对深入数据的挖掘和解读能力。这对于如何降低流量成本、增加流量规模、提高广告转化效果等方面具有重要意义。

重点知识：坦白讲，本章有关流量的知识包括7.2/7.3/7.4节都是非常重要的环节，这涉及数据的跟踪采集和生产机制，因此希望读者能够掌握。除此以外，流量数据化运营的分析指标、分析模型、小技巧以及最后两个案例也是重点知识，尤其是模型和案例中应用的思路和方法：

·基于自动节点树的数据下探的思路。

·KMeans最优K值的确定方法。

·使用GraphViz自己画出树形图。

·使用Matplotlib画雷达图。

外部参考：本章涉及的相关知识，需要读者通过外部资料做进一步学习和参考的内容如下：

·在HTML中颜色的应用非常丰富，有关更多的颜色名称及其对应的十六进制代码，请参照http://www.w3school.com.cn/tags/html_ref_colornames.asp。

·使用GraphViz做进一步的应用中，node是一个非常关键的对象。关于node节点还有更多可控的属性，除了本章介绍的内容外，请具体参考http://www.graphviz.org/content/node-shapes。

·使用Matplotlib库不仅可以画出常见的图形，更包括很多异形图，例如桑基图、玫瑰图、热力图、等高线图、世界地理位置图、三维图等，有关这些图形的具体示例，除了本书之前的章节外，其他请参照http://matplotlib.org/gallery.html和http://matplotlib.org/examples/index.html。

·虽然本书是有关流量数据化运营工作，但是对于流量的基础知识讲解甚少，有关流量的基础内容，读者可以通过《流量的秘密：Google Analytics网站分析与优化技巧（第3版）》了解更多，虽然这已经不是最新版本，但却是一个经典翻译本。

应用实践：网站数据分析与传统的数据分析和挖掘有其独立性，原因在于整个网站数据分析工作的各个过程大多通过网站分析工具“一条龙”完成。如果读者有兴趣，可以自己先通过免费的Google Analytics搭建一套网站流量分析系统，这里面会涉及很多数据采集、跟踪、预处理等规则问题，会加深对于流量数据工作各方面的认知。

流量数据工作的日常应用大多基于统计的，而对于流量数据的建模似乎很少用到算法、模型，本身提供的模型和案例思路，基本都使用了相关算法和模型，读者可应用到工作实践中。

第8章　内容数据化运营

内容运营是信息化媒体运营的核心，对于此类公司而言，内容即公司的核心价值。本章将围绕内容运营的相关话题展开，包括分析指标、应用场景、分析模型、分析小技巧、分析大实话，最后会通过两个案例来介绍如何通过Python做内容数据化运营支持。

8.1 内容数据化运营概述

内容运营是指基于内容的策划、编辑、发布、优化、营销等一系列工作，主要集中在互联网、媒体等以内容为主的行业领域。内容运营根据内容生产方式的不同可分为UGC、PGC和OGC三种。

1）UGC（User-generated Content），用户生产内容。这是论坛、贴吧、微博时代的主要内容生产方式，内容主要由参与内容载体的用户产生，运营方本身不产生任何实质性内容。这些用户一般都是非专业“写手”，通常基于兴趣、爱好等共同语言而自发形成内容。

2）PGC（Professionally-generated Content），专业生产内容。PGC相比UGC，都是由用户产生内容，但是这里的用户主要指的是有专业背景、资历的用户，包括行业领袖、知识专家、书籍作者等，这些人通常能产生非常高质量的专业内容。现在很多知识性网站都是此类形式，例如知乎、个人微信公众号等。

3）OGC（Occupationally-generated Content），职业生产内容。OGC相比PGC在内容专业度上相当，但是OGC的特点是将内容生产作为一门“职业”，因此相对应的从内容生产中获取收入是这一类型的显著性特征。OGC的普遍代表是各个新闻类网站和媒体，一般都以付费投稿、分成等方式吸引高质量的“写手”参与内容生产；当然，除了邀请外部专家参与内容生产，这类网站自身也拥有很多职业内容生产者。

8.2 内容数据化运营指标

内容数据化运营指标包括内容类指标、SEO类指标、内容流量指标、内容互动指标和目标转化指标。

1.内容质量指标

高质量的内容一般都以原创为主，在很多情况下内容生产也会参照其他已有资源。原创度用来评估内容有多大比例是原创产生的。原创度的评估，主要通过两个方面做对比参照：

·与网站本身内容做对比；

·与互联网已有内容做对比。

原创度指标的计算，可以先将不同的内容做分词，然后基于分词结果做词频统计，基于统计结果评估非重复关键字的占比。公式为：1-重复关键字数量/总关键字数量

2.SEO类指标

（1）收录数量/比例

被搜索引擎收录是能从搜索引擎中找到相关内容的基础，收录数量指的是所有内容中能被搜索引擎检索并加入到期内容索引中的数量，其数量占总内容数量的比例就是收录比例。网站内容的收录对应到搜索引擎中的指标是“索引”，该数据可以从搜索引擎站长工具中查询到。如下是笔者某个网站在百度站长工具中的文章索引数量。

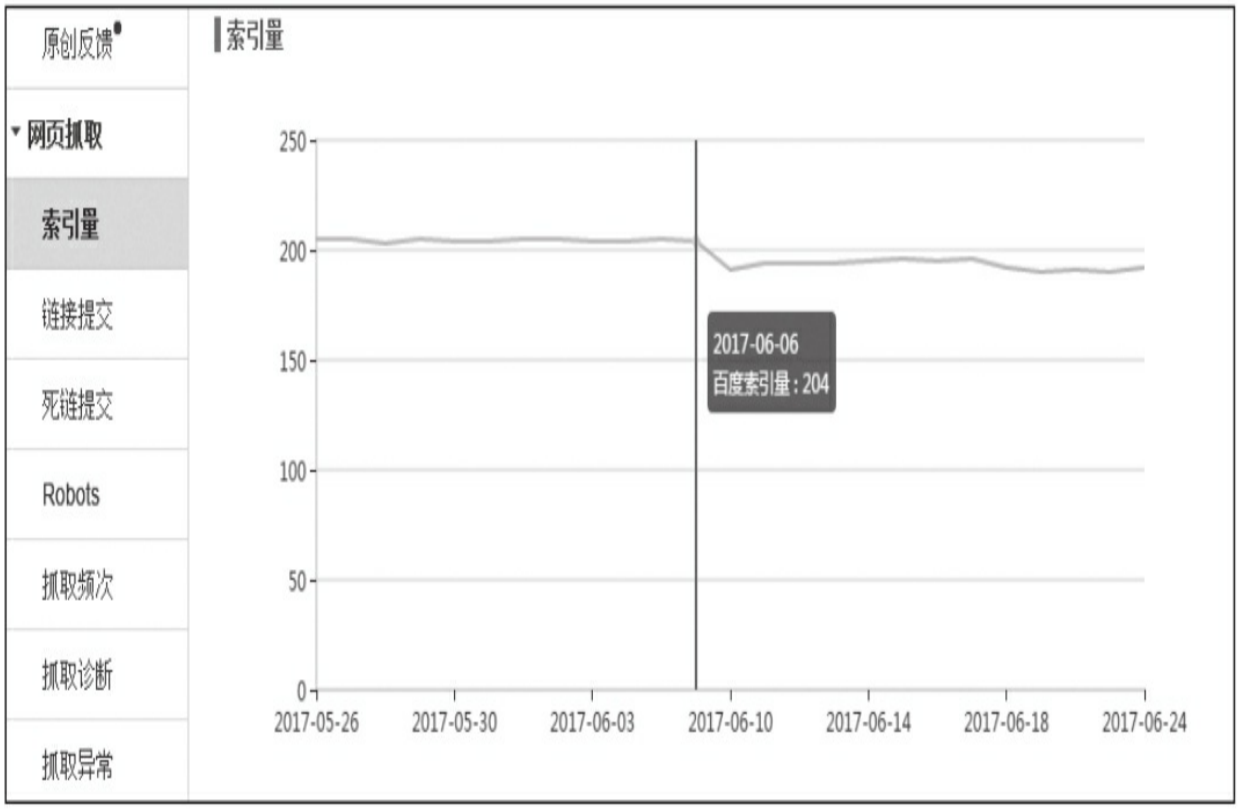


图8-1　索引数量

除了收录数量外，收录的速度也是衡量收录能力的重要指标。新内容被收录的越快，越能尽早的吸引具有相关兴趣的读者关注。

影响收录的主要因素包括：网站结构、robots.txt设置、服务器状态、站内互链设置、URL设置、网站内容的可理解性等。收录构成了可以在搜索引擎中找到的第一步。

（2）关键字排名

关键字排名指的是在搜索引擎中搜索某个文章相关的关键字时，自身网站在整个搜索引擎中的排名。一般情况下，关键字排名越高，被用户点击的机会越大。除了关键字的排名影响外，还包括内容关键字、内容形式等也会影响用户点击。

关键字排名是SEO优化的关键结果之一。影响关键字排名的因素也非常多，包括外链数量、关键字密度等可评估指标以及很多不具有固定

标准的参照指标，包括网站路径设置、子域名设置、合理的返回码、死链接处理、关键字在内容中的分布（如meta、title、alt等）、内容类型（文字比图像和视频更容易被搜索引擎理解）等。

（3）点击量和点击率率

被用户点击是获得用户流量的关键，前面所有的工作都是为了吸引用户流量，因此点击量和点击率是衡量用户点击程度和信息匹配程度的重要指标。点击率=点击量/总展示次数

由于内容流量的来源主要渠道是SEO（搜索引擎优化），因此做好SEO工作是产生稳定、免费流量的主要途径之一，对于不做广告宣传的网站来讲，这很可能是其主要的流量来源。因此在做内容运营时，SEO类指标是必须关注的指标。

3.内容流量指标

获得流量是内容运营的基础，有关流量的指标请参考7.5.2节和7.5.3节两部分的内容，这两部分内容对于内容运营基本适用。

4.内容互动指标

（1）收藏量

收藏是很多内容型网站都具有的功能，收藏越多意味着用户日后再次浏览或使用该内容的机会越多。因此，收藏量是衡量用户参与互动的重要指标。

跟收藏量相关的指标是收藏量、人均收藏量、每内容收藏率等。

（2）点赞量

点赞在不同的内容中有不同的形式，并且“点赞”不一定都是积极评价机制，因此点赞理解为评分更恰当。例如很多网站对于评价可分为很好、好、一般、差、很差，而另外有些网站可能只有点赞一种机制，不点赞可能意味着用户认为内容不佳或用户根本不想做出评价。

跟点赞量相关的指标是点赞率、平均评分、最高（最低）评分等。

（3）评论量

评论是对内容互动的基本形式，参与评论的用户越多证明该内容能吸引用户互动的价值越大。当然，评论多并不一定意味着内容本身的质量或价值度高，很多内容可能通过打擦边球甚至做有争议内容来增加用户的评论。

另外，对于评论本身的质量评估也往往是一个重点难点，这里面涉及很多非结构化文本信息的抓取、判断、识别、分析以及潜在语义的分析，情感分析、用户倾向分析等相关话题便针对此类分析展开。

（4）传播量/传播率

传播是可以产生更多曝光量、覆盖更多用户群体并产生更大价值的关键环节，传播通常基于一定的形式产生，例如转发、分享等。传播数量是评估内容质量高低的重要指标，传播率是衡量所有内容中产生传播行为的比例。

（5）二次传播率

在信息传播个性化、传播主题个人化的今天，所有人都可以生产和传播内容，因此内容的传播不再是由媒体到个人的单向传播，而是以每个人作为传播节点都能形成传播效应。整个传播路径如图8-2所示。

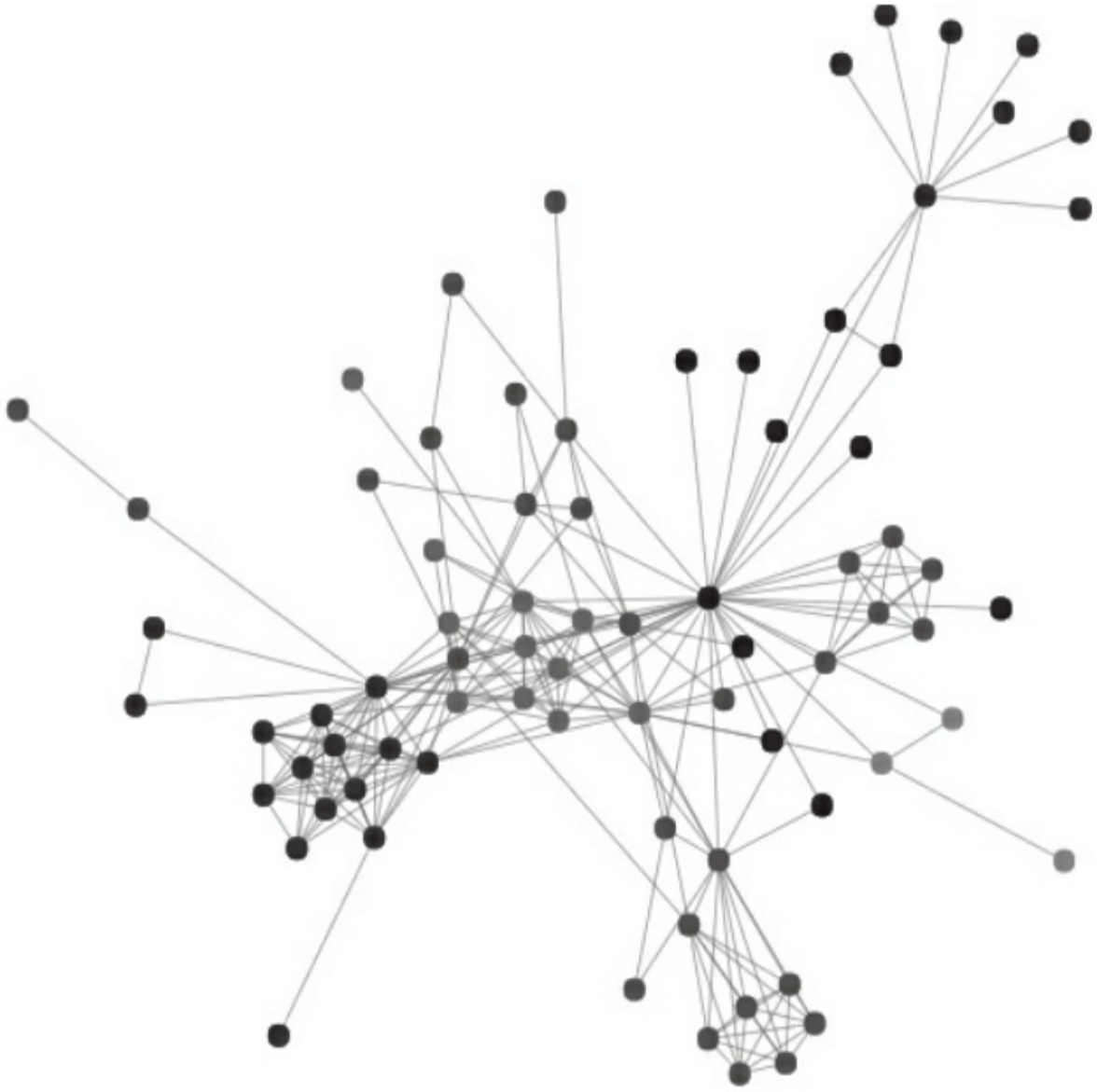


图8-2　内容传播路径示意图

二次传播意味着内容能够产生2次及2次以上的传播，因此是产生巨大传播效应的基础。以新浪微博的二次传播量定义：二次传播量=微博粉丝数×转发次数×30（其中30是平均每个粉丝自身拥有的粉丝数量）。假如有1000个粉丝，当发布1篇文章之后被100个粉丝转发，那么可以产生的传播量=1000×100×30=3000000。

从内容传播本身看，二次传播比一次传播能产生更大的价

值，但并不是所有内容都趋向于二次传播。基于内容和产品的定位来驱动运营，将内容长久的传播出去还是“阅后即焚”只是不同的运营方式而已，并没有标准的孰优孰劣。

5.目标转化指标

对内容型网站而言，虽然不像电子商务网站一样具有特定的商品作为转化参考，但每个网站既然存在都有自身的目标。常见的目标包括：

·第一类：希望用户越多尽量多的文章，停留更长的时间等。

·第二类：希望用户能够留下联系方式，方便日后用户运营。

·第三类：特定的目标事件，例如下载内容、点击广告等特定事件。

·第四类：能通过打赏、小额支持、内容付费等形式完成付费阅读。

对于这些目标而言，都是作为目标转化率的计算依据。具体方式如下：

·对于第一类目标，可以量化访问深度、平均停留时间等体验类指标，然后通过计算完成这些指标的用户数的比例来计算目标完成率。

·对于第二类指标，可以计算留下联系方式的用户比例来计算目标转化率。

·对于第三类指标，可以计算完成特定事件的发生次数来计算目标转化率。

·对于第四类指标，直接采用跟电子商务网站类似的方式，以打赏为标准，可以计算打赏量、打赏率、平均打赏金额、重复打赏率等指标。

8.3 内容数据化运营应用场景

内容数据化运营主要应用于内容采集、内容创作、内容分发和内容管理四个方面。

1.内容采集

内容采集是内容运营的起始流程，很多网站甚至自身不生产任何内容，专门以采集其他网站的内容为生。

对于自身不生产任何内容、也不以任何形式促使第三方产生内容的媒体来讲，有些是直接“复制”其他网站的内容然后自身做展示，这些媒体只是内容的搬运工；有些则通过个性化算法为用户提供精准内容，例如今日头条。

在内容采集过程中，数据主要可以应用的方向包括：

·不同来源的内容原创度和重复度如何？

·不同来源的信息主题分别是什么？

·如何从不同的采集文章中提取关键字标签？

·如何从不同网页中获取符合目标需求的数据内容？

2.内容创作

内容创作是自身生产内容的过程，主要涉及内容的主题、标题、排版、插图等内容本身，也包括基于SEO相关策略的内容优化。数据的主要应用场景包括：

·网站的用户群体主要关注哪些方面的内容主题？

·与K1关键字相关的其他关键字包括哪些？

·如何为每篇文章创建自动摘要和关键字？

·如何在内容审校时基于现有的内容规则做文本纠错？

·不同的内容间具有哪些潜在关系？

·所又内容的完整知识图谱是怎样的？

·最近有哪些新的热点？

3.内容分发

内容分发是将基于一定的分发策略将内容推送给特定目标对象的过程，它的核心是如何让用户更高效、精准的触达内容。数据的主要应用场景包括：

·哪些用户具有较高的相似度，可以推荐其他相似用户的关注内容？

·如何根据用户输入的句子，智能推荐最相关的搜索结果？

·如何识别不同用户的行为模式，然后针对性的提供其最匹配的内容？

·如何基于内容的相似度提供更多相关的文章给目标用户？

·如何将用户兴趣、时间周期变化以及内容结合起来形式最佳内容TOP榜单？

·如何根据不同的运营目标合理安排内容上线和下线时间？

·如何组织内容，以实现资源位对各个内容贡献的最大价值？

内容分发跟流量分发非常相似，有关该部分内容请参考“7.6.2流量分发”。

4.内容管理

内容管理是对内容相关信息的审核、校验、识别、分析等，它是管控和治理等日常性操作事务的统称。数据的主要应用场景包括：

·如何识别垃圾评论？

·如何将用户发布的违规“黄图”识别出来？

·用户评论的情绪是积极的还是消极的？

·哪些内容含有“不能出现”的关键字，或者相近关键字？

·用户新发布的内容是否直接“复制”产生的，其相似度有多少？

·如何对站内信中的垃圾信息做识别，以减少用户收到恶意广告的侵扰概率？

8.4 内容数据化运营分析模型

本节的数据化运营分析模型主要涉及情感分析模型、搜索优化模型、文章关键字模型、主题模型、垃圾信息检测模型。

8.4.1 情感分析模型

情感分析是对情感倾向的分析，用于分析特定对象对相关属性的观点、态度、情绪、立场以及其他主观感情的技术，分析结果通常属于正向、中性或负向的一种。

情感分析的应用场景：

·竞争情报：获取用户观点中关于竞争对手的特定信息。

·舆情监测：获得有关自身网站、内容、产品、服务、品牌、形象等相关信息的监控和预测，以获得有较强影响力、倾向性的言论和观点的现状及未来趋势。

·客户倾向分析：客户对于企业的倾向是积极还是消极的分析，利于建立全面的客户与企业形象认知。

·话题监督：监督特定话题下，所有用户的话题集中点、主要内容、话题演变等。

·口碑分析：用户对于企业各方面的感知和认识，尤其对于具有良好传播效应的意见领袖的口碑把控。

情感分析常用方法：除了非负矩阵分解、基于遗传算法的情感分析之外，使用的最多的还是监督学习算法，例如朴素贝叶斯、K近邻和支持向量机等。使用分类方法下做情感分析的基本思路是：

步骤1　文本预处理，包括去除无效标签、编码转换、文档切分、基本纠错、去除空白、大小写统一、去标点符号、去停用词、保留特殊字符等。

步骤2　文本分词，在中文环境下需要特定的分词模型。

步骤3　文本向量化，将文本特征转化为向量空间模型来标示。

步骤4　特征提取，对于海量稀疏特征做特征提取，包括特征选择和数据绛维等方法。

步骤5　分类建模和效果评估，选择特定的分类模型，建立模型并做效果评估和结论分析。

有关文本预处理和分类算法的更多内容，请见3.12.4节和4.3节。

8.4.2 搜索优化模型

用户在某些文本之间可能存在频繁的关联查阅关系，而这些关键字之间会蕴藏用户的潜在意图。例如，当用户在搜索引擎搜索“热度分析”一词时，相关的搜索词可能包括:空间热度分析、关键词热度分析、音频热度分析、热词分析、关键词热度分析十法、关键词热度分析、网络游戏热度排行榜等。

搜索优化模型可以帮助用户更快找到有兴趣的潜在内容，可用于搜索过程中的联想功能、相关的结果提示和二次搜索建议。

常用的搜索优化模型的方法是关联模型，例如Apriori、FP-growth等，有关关联模型的更多内容，请见4.4节。

8.4.3 文章关键字模型

关键字提取是从文本中提取跟内容最相关的词语，关键字抽取的结果常用于文档检索、文章标签编辑等，也经常用在文本聚类、文本分类、关键字摘要等方面。

关键字模型能生成简短的关于文档内容的指示性信息，将文档的主要内容或核心关键字呈现给用户，以决定是否要阅读文档的原文，这样能够节省大量的浏览时间并提高信息关键信息的展示能力。

文章关键字模型抽取应用场景：帖子、新闻、资讯、评论、问答等的标签、内容和meta信息的产生。

文章关键字模型抽取常用方法：通过词频统计、TF-IDF模型获得文本的主要关键字。

有关该模型的更多内容，请见4.9.3节。

8.4.4 主题模型

主题模型（Topic Model），是提炼出文字中隐含主题的一种建模方法。在统计学中，主题就是词汇表或特定词语的词语概率分布模型，它是文字（文章、话语、句子）所表达的中心思想或核心概念。例如，当提到IBM时，可能我们会想到ThinkPad；提到比尔盖茨，我们就会想到Windows。IBM和ThinkPad、比尔盖茨和Windows就是各自主题里面相关的概念。

如图8-3所示是有关数据工作的主题。

图8-3　数据工作主题

主题模型是一个能够挖掘语言背后隐含信息的利器，是语义挖掘、自然语言理解、文本解析和文本分析、信息检索的重要组成部分。

·它可以衡量文档之间的语义相似性，是文本聚类、分类、情感分析、文档相似度等应用的重要组成部分。

·它可以解决多义词的问题，实现准确的词性标注。

·它可以排除文本中噪音，从中准确的提炼出主题关键字。

主题模型克服了传统信息检索中文档相似度计算方法的缺点，能够在海量数据中自动寻找出文字间的语义主题。主题模型可以应用到围绕主题产生的应用场景中，例如搜索引擎领域、情感分析、舆情监控、个性化推荐、社交分析等。主题模型的得到的结果，可以在去停用词之后，配合标签云等形式做进一步的形象展示。

常用的主题模型包括：

·潜在狄利克雷分配模型（Latent Dirichlet Allocation，LDA）。

·概率潜在语义分析（ProbabilisticLatent Semantic Analysis，pLSA）。

·其他基于LDA的衍生模型，如Twitter LDA，TimeUserLDA，ATM，Labeled-LDA，MaxEnt-LDA等。

主题模型里面的潜在狄利克雷分配模型跟线性判别分析的英文缩写都是LDA，但二者毫无关联，仅仅是缩写相同而已。

8.4.5 垃圾信息检测模型

垃圾信息检测模型是一种分类应用，主要用于检测特定对象是否包含垃圾信息，是网站内容管理的重要方式和途径。

常见的垃圾信息检测应用包括：

·从电子邮件中过滤垃圾邮件。

·从站内信中过滤含有恶意信息的信息。

·从评论或留言中过滤过激言论。

·从用户发布的文章中识别负面题材。

垃圾信息检测模型可以将于分类模型来实现，常用方法：朴素贝叶斯、矩阵变换法、K近邻、支持向量机、神经网络等。有关分类算法的更多内容，请见4.3节。

除了基于有标签的训练集做监督式学习外，还可以使用非监督式的方法做垃圾信息监测，例如：

·基于内容相似度，分析新评论与已有的垃圾信息的内容相似度，如果相似度高于一定阈值，则认定为垃圾内容。当然，这样做的前提是有一份相对完整的垃圾信息的集合，并且需要不断维护。

·基于固定信息的过滤，例如固定IP、包含特定关键字、包含URL、来源于特定域等，这些就不属于算法类应用了。

除了针对文本垃圾信息检测外，还可包括更多类型的内容形式，例如视频、图片、语音等。

随着内容形式的多样化，针对其他类型的分类应用也非常常见，例如识别图片和视频内容中的“黄赌毒”信息、反动文字信息等并过滤。要完成这类的应用，难点还是在与海量训练数据本身，无论是视频还是图片，大多数企业本身通常是不具备海量资源的。

8.5 内容数据化运营分析小技巧

8.5.1 通过AB测试和多变量测试找到最佳内容版本

A/B测试是网站优化的基本方法，常见于高级网站分析系统。A/B测试包括双变量测试和多变量测试。Adobe Analytics、Webtrekk、Google Analytics等网站分析工具都自带A/B测试功能。

A/B测试的功能设置在不同系统中有差异，但流程基本一致：

步骤1 设置测试名称。

步骤2 设置测试的原始网页和优化网页。

步骤3 测试参数调整，包括测试参与测试的流量、版本的流量分配原则（是否平均分配）、数据测试时间、置信度阈值、设置转化目标（指定目标或事件）、结束后是否直接应用最优结果等。

步骤4 原始网页和测试页面部署测试代码。

步骤5 部署上线和测试应用及优化。根据上线测试结果做多个版本的迭代更新及测试，或将最佳版本上线应用。

经过以上五步，网站测试工作即可自动运行，待数据条件满足后，数据报告中会出现A/B测试结果；如果设置了自动应用，最优网页会自动上线而无需人工参与。图8-4显示了基本的A/B测试步骤。

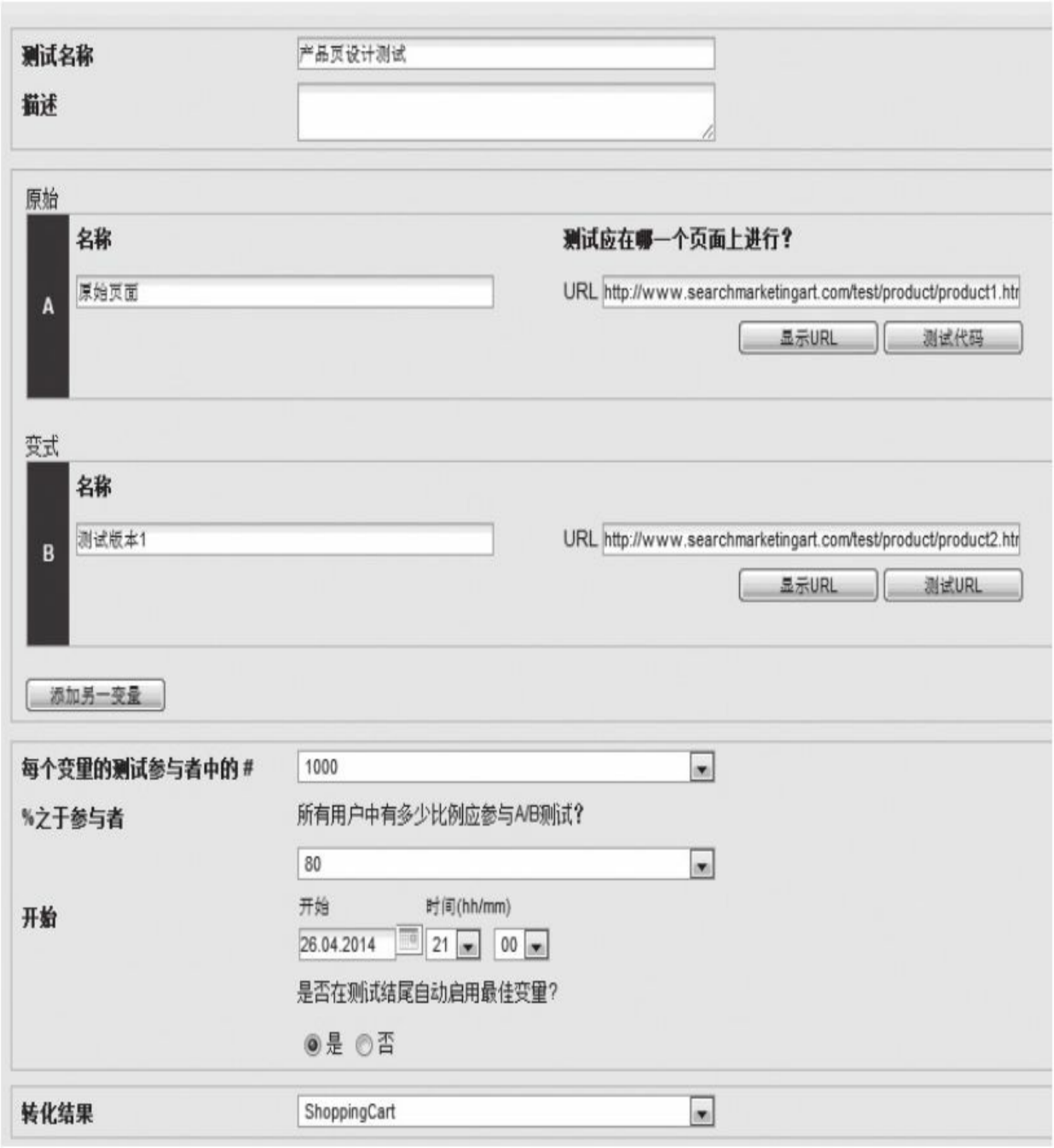


图8-4　Webtrekk A/B测试配置

在A/B测试过程中，很多公司会提供一种“循序渐进”的测试方法：

·内部测试：针对公司内部的人员做不同测试版本。

·超小样本用户测试：针对超小部分用户（例如0.1%）做测试，通

常这部分用户是公司的忠诚用户，对于测试的响应支持度较高。

·小样本用户测试：针对一小部分用户（例如1%）做测试，这部分用户往往代表了较为积极的用户行为和反馈模式。

·大样本用户测试：针对较高比例的用户（例如10%）做测试，这部分用户代表了与网站主流人群相似或相同的反馈模式。

这种方法能避免早期在选择测试对象时不谨慎导致的结果偏差。另外，以下几点关于A/B测试的应用需要读者额外注意：

·A/B测试是一种注重短期结果的行为，对于长期行为的测量往往无法发挥作用。例如以购买保险为转化目标的测试往往不能很好地发挥作用。

·A/B测试必须有明确的KPI目标，并且该目标一定是可测量的，如果没有目标则无法产生评估结果。

·A/B测试的执行条件必须与实际运营条件相似，这样产生的结果才具有可操作应用性。例如不能在有促销的条件下做A/B测试，然后把测试结果应用到非促销场景中。

A/B测试是页面设计和内容优化的最佳实践方法，多变量测试适用于页面更改元素较多或由于时间限制无法进行全面测试的情况。但是，在做测试时一定要注意测试样本的选择与全站用户样本成分和构成的一致性，否则可能会导致结果偏离预期。

8.5.2 通过屏幕浏览占比了解用户到底看了页面多少内容

在做用户行为分析时，我们经常会关注用户浏览了某个页面的内容，也知道用户的停留时间，但是却不知道到底用户在该页面上浏览了多少或哪些内容。借助于特定的页面浏览百分比方法，我们可以清楚地知道用户在每个页面上看了多少内容。

要实现这一数据的跟踪，通常需要额外的代码（一般称为Plugin）来实现。具体实现过程跟网站流量跟踪代码的部署有关，笼统而言，该过程可以分为三步：

步骤一：将实现特定功能的Plugins JS代码写入通用全局脚本；

步骤二：在通用全局脚本中，启用Plugins功能；

步骤三：在网站分析系统后台，指定该变量的赋值字段，即通过特定Plugins采集到的数据放到哪个字段或变量（注：如果已经在Plugins JS中指定变量则无需重复定义）。

完成以上步骤之后，在系统采集并处理完数据，在特定的报表中（具体报表取决于在步骤三中创建的字段或变量的位置）可以看到如图8-5所示类似数据。

No.	ScrollPosition	Qty ScrollPosition▼	% Qty ScrollPosition
1	100	375	24.61 %
2	70	92	6.04 %
3	20	87	5.71 %
4	75	84	5.51 %
5	80	83	5.45 %
6	85	81	5.31 %
7	60	79	5.18 %
8	65	69	4.53 %
9	45	66	4.33 %
10	55	62	4.07 %
	Total	1,524	100.00 %

图8-5　页面浏览百分比

图8-5的数据显示了用户浏览的每个百分比下的数量占比，该百分比以浏览器右侧的滚动条的位置比例作为计算依据。例如假设页面A每天有100万的PV产生，但是如果综合页面浏览百分比数据发现，其实只有25%的人完全看完该页面，大多数内容其实都没有机会曝光在用户面前。基于不同比例可以分析随着页面长度的增加用户能注意到的比率的下降趋势，该信息可以帮助内容运营者合理规划内容长度。

8.5.3 通过数据分析系统与CMS打通实现个性化内容运营

1.网站内容运营的三种形态

网站内容运营发展按照运营规则来看，依次经历了粗放型运营、群组运营和个性化运营三种形态。

1）基于全部用户群体的粗放型运营指所有网站运营工作的开展都是以企业自身为核心，将相同的规则应用到企业全部运营范围中。在该过程中，企业处于优势地位，用户只能被动接收网站信息而缺乏选择性和自主性。

2）基于细分规则的群组运营已经开始区分用户，并根据企业自身资源运营需求在不同类型的资源中运营不同的内容；或者根据用户的群组划分，对相同群组的用户展开相同的运营动作。在这个过程中，用户具有在一定范围内选择内容的权利，但内容仍然是基于群组规则而非个人规则。

3）基于个人推荐的个性化运营是在综合用户喜好、企业内部资源特点的基础上，将企业资源与用户需求完美结合，针对不同的用户进行精准运营的过程。个性化运营是企业与个人信息交互的过程，这个过程中不存在明显的主从关系，双发互为信息发送者和接受者。

三种形态的运营规则如图8-6所示。

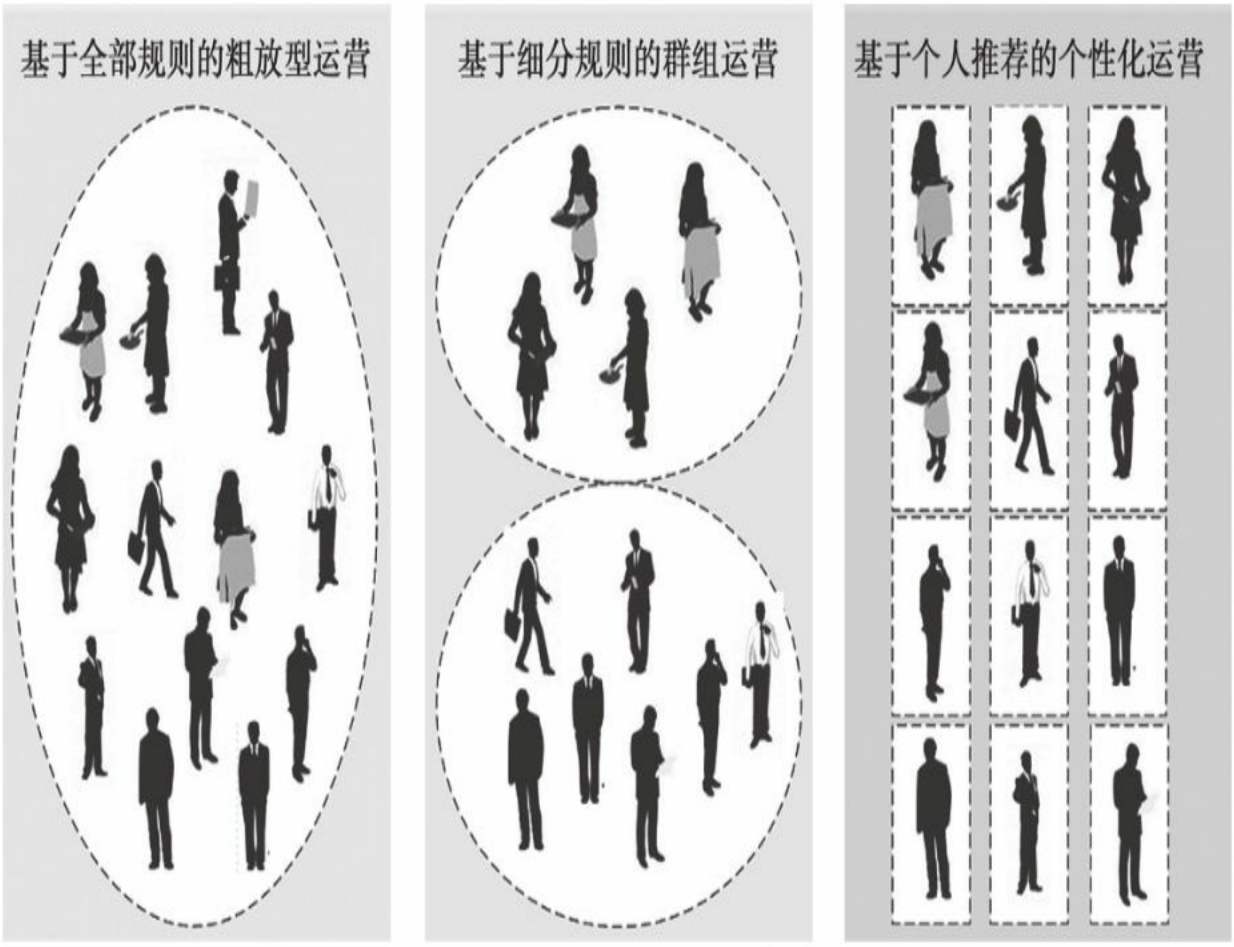


图8-6　三种形态的运营规则

2.个性化内容运营的价值

个性化网站运营与其他两种运营方式相比，其价值主要体现在以下3个方面。

（1）增加长尾内容曝光

很多网站由于内容过多，众多优质内容无法被用户看到，内容浏览呈现出长尾特点。站内个性化网站运营可以将长尾内容与用户个性化需求结合，通过长尾内容满足不同用户需求；个性化规则中还可以指定人工干预，通过固定展示规则达到针对特定内容曝光的目的。

（2）提高站内用户体验

个性化运营的本质是将运营的核心从企业转移到用户，即所有的内容都是根据用户需求和喜好而产生；在这一过程中，用户从进入落地页开始的整个体验度上升，同时企业也将从中受益，表现在数据上是访问深度和停留时间的增加、退出率和跳出率的降低以及最终转化效果的提升。

（3）提高网站转化率

对于内容型网站而言，个性化运营通过精准的内容匹配为用户推荐最适合或最喜欢的内容或服务，在相同的流量规模和流量结构下必然会提升网站转化率，这也是个性化运营对企业最重要的贡献之一；另外，个性化运营还能通过对已经标识的流失用户进行精准的信息推送，从而实现对流失用户的挽回。

3.个性化运营的信息推送方式

个性化内容运营需要将数据分析系统与CMS打通，打通后可以呈现以下个性化信息推送形式：

（1）根据浏览数据的个性化推送

·看了还看：根据用户当前浏览内容推荐下一个最可能感兴趣的内容，通常在内容详细页的两侧或底部出现，主要作用是引导用户浏览行为。

·其他用户在看：根据用户的浏览历史，推荐和该用户浏览行为类似的其他用户最可能浏览的内容，通常出现在页面底部，主要作用的引导用户浏览。

（2）根据搜索数据的个性化推送

根据搜索数据的个性化运营是所有个性化推荐中较为复杂的部分，原因是基于搜索的个性化推荐增加了自然语言处理的过程，这个过程相对复杂且准确率要求较高。根据搜索词的个性化推荐目前主要应用于两种形式：

·一是当用户搜索完成后，会在搜索页面侧边栏或底部出现“搜索该

词的用户还会搜索”，该部分是与上述推荐结果类似的信息展示。

·二是推荐系统会在用户搜索结果下面提示“相关搜索词”信息，用来确定搜索需求、扩大搜索范围、提高搜索质量等。

除了以上用户的行为类型外，其他可能出现的推荐场景包括：基于用户评论的推荐、基于收藏的推荐、基于关注的推荐等，其推荐方式与上述场景类似。

4.个性化运营的主要算法支撑

实现个性化运营的主要算法包括协同过滤、关联规则、基于内容的推荐、社会网络算法以及组合算法。

（1）协同过滤

协同过滤（Collaborative Filtering Recommendations，CF）是利用兴趣相同、拥有共同经验的群体喜好来预测用户喜好的方法。协同过滤通常分为基于项目的协同过滤（item-based CF）和基于用户的协同过滤（user-based CF），核心是根据不同用户对项目的评分来预测项目之间或用户之间的相似性，并基于这种相似性做出推荐。除此以外还有基于模型的协同过滤以及混合协同过滤机制。

举例：如图8-7所示是基于用户的协同过滤机制。假设用户A喜欢物品A、C，用户B喜欢物品B，用户C喜欢物品A、C、D；从用户的历史喜好信息中，我们可以发现用户A和用户C的口味和偏好是比较类似，因此可以物品D推荐给用户A。

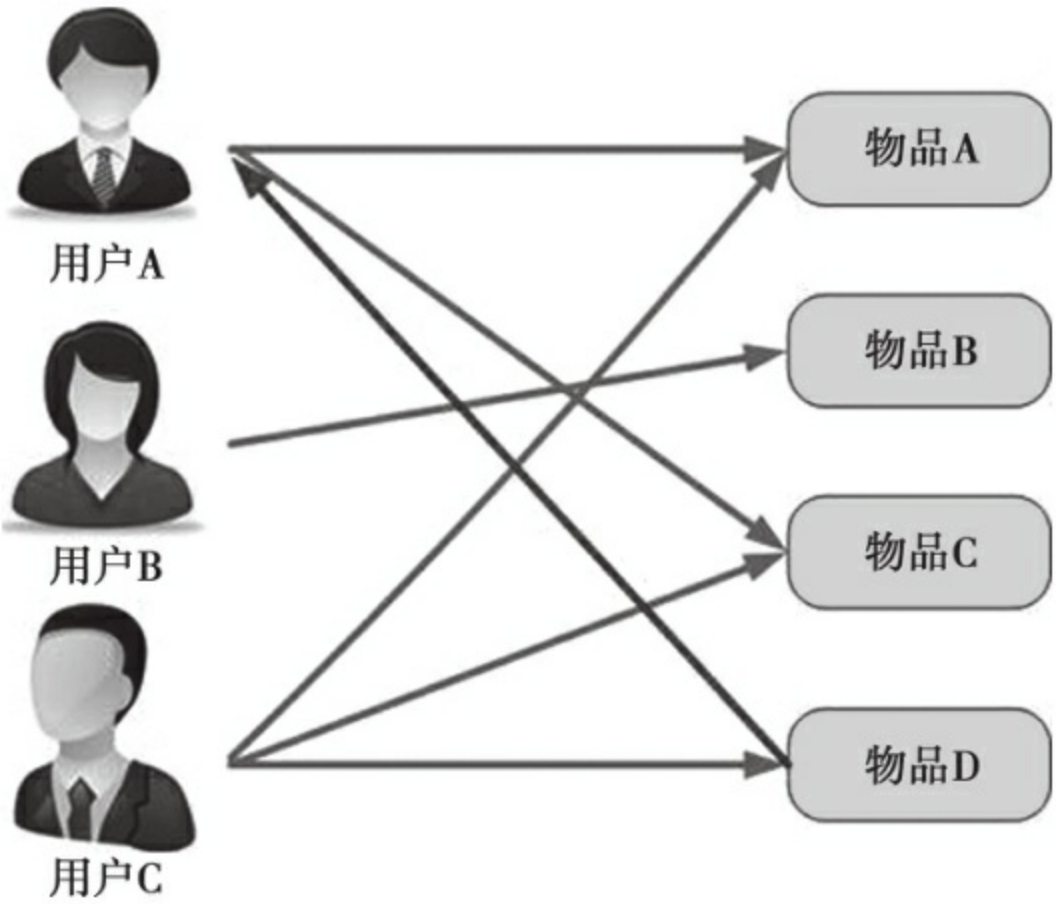


图8-7　基于用户的协同过滤机制

协同过滤推荐的自动化和个性化程度高，并且能处理复杂的内容和推荐对象，可以针对性的推荐用户尚未发掘的新兴趣点；但是，协同过滤在面对新客户由于无法与其他用户特征进行比对，因此无法产生有效推荐结果；另外，面对数据稀疏性的解决方案以及算法可扩展性较差上的问题，协同过滤仍然存在改进空间。

（2）关联规则

关联规则（Association　Rules）本书已经不止一次提到过，关联规则可以广泛应用到用户的浏览、搜索、购买、产品等推荐场景。

关联规则技术成熟且推荐结果较为直观，可以发现用户的长尾需求并进行推荐，尤其在销售领域应用广泛；但如果产品、内容或推荐项目存在同义性将无法产生准确结果，另外数据抽取规则复杂且耗时，无法

应用实时个性化推荐场景。更多有关关联规则的话题，请参考4.4节的内容。

（3）基于内容的推荐

基于内容的推荐（Content-based Recommendations）很大程度上是在进行文本、图像等内容的挖掘。基于内容的推荐通过分析内容提炼出特征，然后通过用户对特征的反馈来学习用户喜好特征，最后将具有其他类似特征的内容推荐给用户。

基于内容的推荐可以对用户兴趣很好地建模，并通过对物品属性维度的增加，获得更好的推荐精度。但是，当物品的属性有限时，将很难得到更多数据，而当物品属性过多时，对于如何分配属性间的权重并更快得到推荐结果该算法仍然存在问题；另外，算法本身只考虑到物品相似度的做法存在一定的片面性；对新用户冷启动时无任何喜好特征的场景仍然没有有效解决方法。

（4）社会网络算法

基于社会网络的推荐算法是通过收集用户在社交网络上的属性（人口社会属性）、标签（喜好、兴趣、类别）、地理位置、行为（原创、活动参与、转发、评论、分享、点赞、收藏）、社交关系（圈子、粉丝和关注、Follow、信任、拉黑、重点关注）等进行挖掘分析，找到用户兴趣点并个性化推荐。今日头条新闻推荐、新浪微博的关注推荐都是基于这种模型进行的。

这种推荐方法本质上是基于用户的推荐，社交网络数据具有动态性强、时间推移规律明显、算法过于复杂及耗时较长等特征，该算法应用的局限性较为明显。

（5）组合推荐算法

通过以上算法分析可以看出，每种算法都有独特优势和不足，通过组合不同算法可以避免或弥补各种推荐技术的弱点，常用的组合方法包括：

·加权处理。采用多种推荐技术运算并根据权重加权得出总评分，

并以此得出推荐结果。

·变换场景。不同的场景采用不同的推荐算法，如针对新用户基于内容推荐、针对老用户使用协同过滤推荐。

·混合展示。在得出推荐结果时，分别取出每种推荐结果的前几个项目组合到一起向用户展示。

·迭代计算。在使用一种推荐算法得出结果后，再采用其他算法在此技术上进行二次或多次运算。

除了以上常用算法外，还有部分算法不属于个性化推荐算法，但也会应用到个性化推荐场景中，如基于用户类别的推荐、物品类别的推荐、基于物品排名或时间热度等推荐，这些算法通常会作为特殊情况下的应用，如冷启动、数据稀疏性等。

8.5.4 将个性化推荐从网站应用到APP端

个性化运营可以应用到手机端，但和WEB端相比其应用具有特殊性：

·应用程序开发要求增加。在使用APP时，用户会更加关注程序对手机空间的占用情况、对网络流量的使用情况以及应用时的流畅度，如果在个性化运营的信息交互过程中出现如页面打开速度慢、流量使用过高等情况，会导致用户体验下降甚至直接卸载应用。

·展示空间有限。与WEB不同的是，APP的展示空间只限定在狭小的手机屏幕区域。由于手机端屏幕限制而无法提供过多的信息展示控制，这限制了个性化信息推荐的展示。

·应用场景更加具体。APP可以追踪到应用设备的时间、地理位置、联网环境、手机号等，这使得获得用户具体场景信息的可能性增加，同时也可以针对用户的不同场景推荐不同的内容。例如，当用户在大型商场时可能更需要时尚消费资讯，而当中午12点时可能更需要个性化餐饮消费信息。

·个性化运营针对性更强。每个应用程序都可以对应到个人，因此个性化的APP运营可直接映射到APP背后的使用者——终端用户。基于这一特性，只要用户使用APP，运营人员便可以精确区分和识别每个用户，并针对不同用户做基于群组、个性化的运营动作。

·跨设备和跨平台的推荐。通常情况下，当用户使用多个平台或设备时，其数据无法进行关联；但当用户登录之后，所有的数据便可通过登录ID关联起来，基于整合后的分析和推荐，运营人员可以随时随地、跨平台、跨设备的进行信息推荐。

·用户需求识别难度增加。APP上的数据除了网站端基本的点击、浏览、购买等行为外，还包括更多非结构化的数据，如语音、地理位置、拍照、视频等，这些数据的采集和分析挖掘将成为APP个性化运营的重点和难点。

目前主流的针对APP的个性化推荐和运营主要采用以下两种形式：

·形式一：通过调用网站端的挖掘结果来实现APP个性化运营。这种形式节省了APP数据回传后做分析挖掘的时间，从而提高了信息反馈效果并降低了数据交互量；但会导致运营内容不够个性化、匹配度不高的问题。

·形式二：只针对APP上的简单场景进行应用。这些场景主要是基于页面浏览的数据，如内容页、商品页等，目的是做更好的内容展示；但问题在于数据的应用场景过少，个性化运营效果不明显。

8.6 内容数据化运营分析的“大实话”

8.6.1 个性化内容运营不仅是整合CMS和数据系统

个性化内容运营的重要步骤是将数据系统与网站运营系统打通，这样才能直接通过系统之间的信息交互直接完成个性化应用。与个性化推荐系统结合紧密的网站运营系统包括：内容管理系统、客户管理系统、站外广告投放系统、活动营销系统、邮件系统、短信平台等，具体结合形式如下：

·内容管理系统：在内容管理系统中单独设置个性化推荐区域，并将个性化推荐系统的结果通过该区域展示，目的是提高用户体验及内容匹配度。

·客户管理系统：通过将客户管理系统与个性化推荐系统打通，将已经登录用户的个性化系统与客户管理系统信息整合，并通过特定接触点形成用户精准营销和个性化信息的站内、外推送，以此提高用户忠诚度、访问黏性并最终提高用户转化和订单效果。

·站外广告投放系统：通过将个性化推荐系统与站外广告投放系统结合，将用户喜好数据与广告投放数据进行关联，针对不同用户在不同平台的行为做个性化广告投放，提高广告点击率、网站访问深度和最终转化率，同时还能降低营销成本，提高ROI。

·活动营销系统：与内容管理系统相似，在活动营销系统中单独建立个性化信息模块，通过用户对不同活动的反馈来提高活动推送的精准度，最终提高活动促销及宣传效果。

·邮件系统：将个性化推荐系统与邮件系统打通后，在对用户的邮件推送过程中加入用户喜好、群体喜好内容，提高邮件打开率、点击率以及到达网站后的转化和重复购买效果。

·短信平台：通过打通个性化推荐系统与短信平台，在对用户信息推送时，通过个性化和针对性的信息标签进行用户提醒，进而提高用户关怀、信息推送的反馈效果。

8.6.2 用户在着陆页上不只有跳出和继续两种状态

着陆页设计是影响用户到达网站后体验的第一要素，也是站内漏斗的第一环节。着陆页设计的好坏会直接影响用户在着陆页的直接反应，当用户在着陆页上的行为远比我们想象的要丰富。

马上跳出、浏览后跳出、浏览其他页面还是浏览目标页面等，这分别代表了着陆页体验度的四个层次：

第一层：当用户到达着陆页之后，仅浏览了首屏（甚至什么都没看）就离开网站，这意味着用户的体验最为糟糕。

第二层：当用户到达着陆页之后，虽然没有浏览其他页面，但是浏览了着陆页的其他屏，并停留了较长时间。

第三层：用户虽然继续浏览网站，但其路径不是页面设计或页面主题预设的路径，即用户没有按照预想行动。

第四层：用户按照网站预想和设计方案行动，并到达指定后续页面。

这四个层次依次显示了不同情况下用户的行为结果。除了第四层外，其他结果都不符合着陆页设计预期。仅仅使用如跳出率、二跳率、停留时间等指标很难评估商业目标的完成程度，因此需要从更多维度进行分析。

·从着陆页到上一步（通常是站外点击链接）的相对转化率可用来评估到达率，到达率是站外流量入站效果的重要指标；

·从着陆页开始到达第二个页面的相对转化率为二跳率，传统定义的方式是1-跳出率，这里建议根据网站的实际着陆页目标进行定义，如发生某个事件、点击或点击到达某个页面才计算为二跳；

·着陆页的跳出无法准确评估页面停留时间，可通过页面点击热力图、页面浏览百分比、下一步访问路径、页面重复刷新率、特殊事件检测、页面加载时间等方法分析用户在页面上的行为；

·着陆页除了具有较强的目标性外，还具有一定的导流意义，因此分析从着陆页到其他页面的分流效果也是其重要环节；

·除了定量分析外，定性分析是必不可少的分析角度，如信息匹配度、页面设计体验、页面引流设计等都是重要的感性因素。

8.6.3 “人工组合”的内容运营价值最大化并非不能实现

在做运营分析时，当从公司整体利益出发时需要考虑如何通过优化配置资源来实现价值最大化的问题。整体价值最大化是通过对所有资源的投入和产出排列、组合，计算出最优化的运营整体方案，价值最大化的实现分为两种：基于数据驱动的个性化推荐以及基于运营编辑的人工组合。

对于个性化推荐的相关内容本章已经介绍很多，这里重点介绍基于运营编辑的人工组合。

基于运营编辑的人工组合指网站编辑通过人工方式进行资源调整和优化，以达到整体价值最大化的目的。

举例：现在有A、B、C三款商品，运营目的是通过在首页焦点图上进行曝光来提升内容整体点击量，即三种内容点击量总和最大化。首页焦点图的点击量按照从高到低依次是排序焦点图1、焦点图2、焦点图3。通过穷举法得到位置和内容组合的结果共6种，如图8-8所示。

图8-8　内容和焦点图组合结果

如何使用这六种方法解决上述问题？

方案一：直接通过测试做数据对照并从中选择最优方案。该方案的应用原则如下：

·当没有历史数据可供参考时，这是最优方法；

·在测试六种方案结果时，外部环境需要尽量保持一致，理想情况下应该同时上线六种方案并作随机测试，但对技术实现的要求比较高。

方案二：通过历史数据进行排列组合分析，找到最大化组合方案。该方案的应用原则如下：

·图中的每个位置以及每种内容的预期流量数据是可供分析和参考的；

·使用数据建模找到不同组合下的数据结果并确定最大值。

8.6.4 影响内容点击率的因素不仅有位置

位置对于内容获得的用户点击率的影响是显而易见的，其基本规律如下：

·首屏的点击率要好于其他屏；

·底部内容会好于中间楼层；

·左侧的广告好于右侧。

但是，以上点击分布规律也存在例外。除了位置之外，还有其他因素会影响用户点击：

·用户成分。如果网站以老用户为主，对网站结构非常熟悉，那么位置对内容点击的影响较小。

·内容标题。在内容标题中存在具有吸引力的文字，会更容易吸引用户点击，例如含有“千万别看”的逆向思维文字、含有暧昧词汇的文字等。

·内容配图。不得不承认，用户对于配图的关注度要远高于文字标题，尤其是配有擦边球的异性配图，效果更佳明显。

·位置接触成本。不同的位置所产生的“接触成本”的差异性大小是位置影响点击的重要因素。同样都是首屏的广告接触点的距离比底部广告的距离要长，此时位置的差异性影响是显著的。不同焦点图之间对用户来讲“接触成本”差异性较小，用户只需移动很短的鼠标距离即可查看所有广告。

接触成本指用户为了看到内容所要付出的成本要素，包括移动鼠标、滚动页面、查找时间、耐心等因素。位置对于点击的影响可总结为：当位置产生的接触成本较低时影响较小，当位置产生的接触成本较高时影响大。

8.7 案例：基于潜在狄利克雷分配（LDA）的内容主题挖掘

8.7.1 案例背景

本案例是从一堆新闻文件中建立相应的主题模型，然后得到不同模型的主题特点，并通过对新文本数据集的预测得到其可能的主题分类。

案例数据源文件有两部分，第一部分在“附件-chapter8”中的news_data.tar.gz压缩包中，该压缩包包含10个文件，这些是用来做主题模型的训练集；另外一个文件在相同目录下，名为article.txt，用来做主题预测。案例的程序文件chapter8_code1.py也在相同数据源目录之下。

本案例程序的默认工作目录为“附件-chapter8”（如果不是，请cd切换到该目录下，否则会报“IOError:File article.txt does not exist”）。

8.7.2 案例主要应用技术

本案例用到的主要技术包括：

·数据预处理：中文分词、TF-IDF向量空间模型转换、字符串全角转半角、XML文件内容解析。

·数据建模：潜在狄利克雷分配（LDA）模型。

主要用到的库包括：sys、tarfile、os、jieba、gensim、bs4，其中gensim是核心。

本案例的重点技术有3个：

·使用bs4的BeautifulSoup做XML文件内容解析。

·使用结巴分词工具做中文分词。

·使用gensim中的LDA（潜在狄利克雷分配模型）做主题模型，并可以基于训练好的模型做新文本的主题预测。

8.7.3 案例数据

案例数据来自搜狐新闻2012年6月到7月的部分新闻数据，涵盖了新闻、家居、体育等主题频道。压缩包中有10个新闻文件，每个文件中包含多条新闻，每条新闻的格式相同，包含新闻URL、页面ID、页面标题和页面内容4部分。以下是文件内容格式示例：

```
<doc>
<url>页面URL</url>
<docno>页面ID</docno>
<contenttitle>页面标题</contenttitle>
<content>页面内容</content>
</doc>
```

8.7.4 案例过程

步骤1 导入库。

```
import sys
reload(sys)
sys.setdefaultencoding('utf-8')
import tarfile  # tar压缩包库
import os  # 操作系统功能模块
import jieba.posseg as pseg  # 带词性标注的分词模块
from gensim import corpora, models  # gensim的词频统计和主题建模模块
from bs4 import BeautifulSoup  # 用于XML格式化处理
```

本案例主要用到了以下库：

·sys：系统库，用来将默认编码设置为UTF-8，目的是更好的处理中文。

·tarfile：用来解析原始数据压缩文件。

·os：用来判断数据是否存在以及遍历目录文件。

·jieba：中文分词，这里使用的是带词性标注的分析模块。

·gensim：主题模型库，这里使用的是其中的词频统计和主题建模模块。

·bs4：这里用到了其中的BeautifulSoup做XML文件内容解析。

步骤2 定义功能函数，包括中文分词、文本预处理、全角转半角、解析文件内容。

中文分词：

```
def jieba_cut(text):
    '''
    将输入的文本句子根据词性标注做分词
    :param text: 文本句子，字符串型
    :return: 符合规则的分词结果
    '''
    rule_words = ['z', 'vn', 'v', 't', 'nz', 'nr', 'ns', 'n', 'l', 'i', 'j',
```

```
保留状态词、名动词、动词、时间词、其他名词、人名、地名、名词、习用语、简称略语、成语、形容词、名形词
    words = pseg.cut(text)  # 分词
    seg_list = []  # 列表用于存储每个文件的分词结果
    for word in words:  # 循环得到每个分词
        if word.flag in rule_words:
            seg_list.append(word.word)  # 将分词追加到列表
    return seg_list
```

该函数主要用于将输入的文本句子根据词性标注做过滤，只保留符合条件的分词结果，由于分词过程中加入了词类判别，因此该过程用时较长。其输入参数如下：

·text：文本句子，字符串型。

函数返回：符合规则的分词结果。

该函数的具体定义如下：

·首先定义了一个要保留词的类别列表rule_words，只保留状态词、名动词、动词、时间词、其他名词、人名、地名、名词、习用语、简称略语、成语、形容词、名形词。完整分词类别，请见第4章表4-8常用结巴分词词性分类。

·使用jieba.posseg做分词得到一个迭代对象words。

·创建一个列表seg_list用于存储每个文件的分词结果。

·在for循环中，通过word.flag属性判断类别是否属于保留词类别，如果属于则将word.word分词结果追加到结果列表中，最后得到的是一个嵌套列表，列表中的每个元素都是对应文件的分词组成的列表。

·最后返回分词列表。

文本预处理：

```
def text_pro(words_list, tfidf_object=None, training=True):
    '''
    gensim主题建模预处理过程，包含分词类别转字典、生成语料库和TF-IDF转换
    :param words_list: 分词列表，列表型
    :param tfidf_object: TF-IDF模型对象，该对象在训练阶段生成
    :param training: 是否训练阶段，用来针对训练和预测两个阶段做预处理
```

```
      :return:  如果是训练阶段，返回词典、TF-IDF对象和TF-IDF向量空间数据；如果是预测阶段，返回TF-IDF向量空间数据
    '''
    # 分词列表转字典
    dic = corpora.Dictionary(words_list)  # 将分词列表转换为字典形式
    print ('{:*^60}'.format('token & word mapping review:'))
    for i, w in dic.items()[:5]:  # 循环读出字典前5条的每个key和value，对应的是索引值和分词
        print ('token:%s -- word:%s' % (i, w))
    # 生成语料库
    corpus = []  # 建立一个用于存储语料库的列表
    for words in words_list:  # 读取每个分词列表
        corpus.append(dic.doc2bow(words))   #  将每个分词列表转换为语料库词袋（bag of words）形式的列表
    print ('{:*^60}'.format('bag of words review:'))
    print (corpus[0])  # 打印输出第一条语料库
    # TF-IDF转换
    if training == True:
        tfidf = models.TfidfModel(corpus)  # 建立TF-IDF模型对象
        corpus_tfidf = tfidf[corpus]  # 得到TF-IDF向量稀疏矩阵
        print ('{:*^60}'.format('TF-IDF model review:'))
        for doc in corpus_tfidf:  # 循环读出每个向量
            print doc  # 打印第一条向量
            break  # 跳出循环
        return dic, corpus_tfidf, tfidf
    else:
        return tfidf_object[corpus]
```

该函数是gensim主题建模预处理过程，包含分词类别转字典、生成语料库和TF-IDF转换。由于在训练主题以及应用新的文本集做主题预测时都会用到该部分，因此单独拿出来用一个函数封装调用。其输入参数如下：

·words_list：分词列表，列表型。

·tfidf_object：TF-IDF模型对象，该对象在训练阶段生成。

·training：是否训练阶段，用来针对训练和预测两个阶段分别做预处理。

函数返回：如果是训练阶段，返回词典、TF-IDF对象和TF-IDF向量空间数据；如果是预测阶段，返回TF-IDF向量空间数据。

该函数的具体定义如下：

·分词列表转字典。使用corpora.Dictionary方法将列表转换为字典形式，转换后的字典是每个分词和对应的token列表。通过for循环，打印

输出字典数据的前5条数据。

·生成语料库。先建立一个用于存储语料库的空列表corpus，然后用for循环读取分词列表中的每个元素，使用字典对象的dic.doc2bow方法将每个分词列表转换为语料库词袋形式的列表，最后将转换后的元素追加到语料库列表中。语料库词袋的每个元素都是由token（分词对应的ID）和token计数组成，用于记录每个分词出现的次数。

·TF-IDF转换。该功能需要根据training=True或training=False来区分训练集或预测集的调用，原因是在训练集中建立的TF-IDF模型在预测应用时不必重新建立，只需直接调用即可，因此可以通过if条件做判断。当训练集调用时（training=True），先使用model.TfidfModel建立TF-IDF模型对象tfidf，使用tfidf[corpus]转换为TF-IDF向量稀疏矩阵，然后打印该向量矩阵的第一条数据，使用break方法终止循环。最后返回的dic和corpus_tfidf在主题建模时使用、tfidf在预测新的文本数据集时使用。当预测集调用时，直接调用上述训练集的TF-IDF模型对象，应用到新的数据集做转换并返回转换结果。

这里在打印输出TF-IDF向量时，使用了break方法终止循环，实际上，在更多情况下会配合if条件做判断，当符合一定条件后，程序跳出循环体不再继续执行。

全角转半角：

```
def str_convert(content):
    '''
    将内容中的全角字符，包含英文字母、数字键、符号等转换为半角字符
    :param content: 要转换的字符串内容
    :return: 转换后的半角字符串
    '''
    new_str = ''
    for each_char in content:  # 循环读取每个字符
        code_num = ord(each_char)  # 读取字符的ASCII值或Unicode值
        if code_num == 12288:  # 全角空格直接转换
            code_num = 32
         elif (code_num >= 65281 and code_num <= 65374):  # 全角字符（除空格）
根据关系转化
            code_num -= 65248
        new_str += unichr(code_num)
    return new_str
```

该函数用来将文本内容中的全角字符串转换为半角字符串，这样会避免在自然语言处理过程中由于字符不统一而导致的信息混乱问题。其输入参数如下：

·content：要转换的字符串内容。

函数返回：转换后的半角字符串。

由于该功能是针对每一个字符做识别转换的，因此其过程比较慢。

该函数的具体定义如下：

·定义一个空字符串对象new_str，用于存储转换后的结果。

·使用for循环读取每个字符，然后使用ord方法读取字符的ASCII值或Unicode值赋值给code_num，接下来根据值做判断：当code_num值等于12288时，这是一个全角空格，此时直接将其值设置为32来转换成半角空格；当code_num值在[65281，65374]时，将其直接减去65248来转换为半角字符值。

·数值转换完成后，使用unichr方法将其转换为字符，并通过直接相加的方法重新组合字符串。

·最后返回转换后的新字符串。

相关知识点：全角字符串转半角字符串

全角指的是一个字符占2个标准字符的位置（例如中国汉字），半角指的是占1个标准字符的位置（例如普通的字符a）。在文本相关的处理过程中，半角和全角字符通常是数据预处理的必要过程。全角字符包含两类字符：

·一类是特殊字符：空格，它的全角值十进制整数为12288，表示为十六进制结果是0x3000；而其半角十进制整数数值为32，十六进制结果是0x20。

·一类是有规律的字符，这类字符的全角十进制整数的范围是[65281，65374]，用十六进制表示的结果是[0xFF01，0xFF5E]；其对应的半角十进制整数值为[33，126]，用十六进制表示的结果是[0x21，0x7E]。

除了空格外，有规律的字符在半角和全角之间的差值是65248，因此我们可以直接在全角数值上减去65248即可得到半角数值。

例如，全角字符串“00527825CBD”转换为半角字符串的结果是“00527825CBD”。

在转换过程中，我们用到了2个新的函数：ord和unichr，这两个是配对使用的函数，前者用来从字符串读取对应的数值，后者用于将数值转换为unicode字符串。

并不是所有的全角字符都能被转换为半角字符，例如汉字是全角字符，占2个字符的位置，但无法被转换；只有英文字母、数字键、符号键等才能可以做全角和半角之间的转换。

解析文件内容：

```
def data_parse(data):
    '''
    从原始文件中解析出文本内容数据
    :param data: 包含代码的原始内容
    :return: 文本中的所有内容，列表型
    '''
    raw_code = BeautifulSoup(data, "lxml")  # 建立BeautifulSoup对象
      content_code = raw_code.find_all('content')   # 从包含文本的代码块中找到
content标签
    content_list = []  # 建立空列表，用来存储每个content标签的内容
    for each_content in content_code:  # 循环读出每个content标签
        if len(each_content) > 0:  # 如果content标签的内容不为空
            raw_content = each_content.text  # 获取原始内容字符串
            convert_content = str_convert(raw_content)  # 将全角转换为半角
            content_list.append(convert_content)  # 将content文本内容加入列表
    return content_list
```

该函数用来从原始文件中解析出文本内容数据。由于原始每个数据文件中都包含了类似XML格式的数据，因此需要将其中特定对象（content标签内的内容）解析出来。在2.2.5节我们介绍了使用xml库获

取并解析xml数据的方法，在3.12.1节中介绍了使用bs4解析标准的HTML格式的数据，这里应用bs4来解析XML格式的数据。

函数输入参数如下：

·data：包含代码的原始内容。

函数返回：文本中的所有内容，列表型。每个新闻为一个列表元素。

该函数的具体定义如下：

·使用BeautifulSoup库建立处理对象raw_code，这里指定的解析库是lxml。

默认情况下，直接使用BeautifulSoup（data）会调用Python默认HTML解析器html.parser，这是Python内置标准库，该库执行速度适用。这里我们指定使用lxml库可以提供更快的解析速度。如果读者之前没有安装过该库，需要通过pip install lxml安装。

·对raw_code使用find_all方法查找所有的content标签，将返回的列表结果赋值给content_code。

·新建一个空列表对象content_list，用来存储每个content标签的内容。

·使用for循环读出每个content标签，先判断标签是否为空，如果不为空，则将标签的text属性（标签内的内容）赋值给raw_content，然后调用str_convert函数将其中的全角转换为半角，最后将结果追加到列表content_list。

·最后返回包含每条内容的列表。

步骤3　解压缩文件。

定义完各种功能函数后，从该步骤开始进入到应用过程。解压缩文件步骤是将tar.gz中的压缩文件提取出来。

```
if not os.path.exists('./news_data'):  # 如果不存在数据目录，则先解压数据文件
    print ('extract data from news_data.tar.gz...')
    tar = tarfile.open('news_data.tar.gz')  # 打开tar.gz压缩包对象
    names = tar.getnames()  # 获得压缩包内的每个文件对象的名称
    for name in names:  # 循环读出每个文件
        tar.extract(name, path='./')  # 将文件解压到指定目录
    tar.close()  # 关闭压缩包对象
```

该过程主要定义如下：

·if not os.path.exists（'./news_data'）：通过os.path.exists来判断指定目录是否存在，如果不存在则执行该之后的功能。

os.path.exists是对文件做操作的常用方法，基于该方法可以对文件做增、删、改等操作。而os.path则是更加实用的功能库，该库可以用来对文件或目录做查找、判断、合并、分裂等，通过更多的会配合目录或文件操作命令实现目录操作。

·使用tarfile.open方法打开压缩包并建立操作对象tar。

·使用tar.getnames（）获得压缩包内的每个文件对象的名称。

·通过for循环配合extract方法将每个文件提取到指定目录下面（path定义的目录）。

·关闭压缩包对象。

步骤4　汇总所有内容。

该步骤可以将所有文件的content中的内容合并起来。

```
print ('walk files and get content...')
all_content = []  # 总列表，用于存储所有文件的文本内容
for root, dirs, files in os.walk('./news_data'):  # 分别读取遍历目录下的根目录、子目录和文件列表
    for file in files:  # 读取每个文件
        file_name = os.path.join(root, file)  # 将目录路径与文件名合并为带有完整路径的文件名
        with open(file_name) as f:  # 以只读方式打开文件
            data = f.read()  # 读取文件内容
        all_content.extend(data_parse(data))  # 从文件内容中获取文本并将结果追加到总列表
```

该步骤功能实现如下：

·建立空列表all_content，用于存储所有文件的文本内容。

·然后两层for循环做目录下的所有文件遍历，第一层for循环使用os.walk方法读取指定目录下的路径和文件信息，第二层for循环读出文件列表中的每个文件。

·在循环体内部，使用os.path.join方法将文件路径和文件名合并为带有完整路径的文件名，用于下面的文件内容读取。

·使用with　open（）方法打开每次合并后的文件对象，然后基于该对象使用read方法读取其中内容。

·调用data_parse功能对提取文本内容并将结果追加到列表中。

这里在读取文件时，没有使用标准的文件读取方法，而是使用了with open（）方法，这种写法相对于之前的标准方法的实现结果是相同的，但是却更加简洁，因该方法无需使用close（）方法关闭文件对象。因此，推荐读者更多使用这种方法。

相关知识点：利用os.walk遍历目录

在针对多文件做数据读取和应用操作时，通常需要程序自动遍历找到所有的文件，然后自动执行批量操作。此时通常会用到os.walk方法，该方法可以返回指定目录下所有的文件目录和文件的迭代器，该迭代器可以循环读取。os.walk参数如下：

```
walk(top, topdown=True, onerror=None, followlinks=False)
```

·top：要遍历的根目录。

·topdown：指定目录树是否自上而下产生。

·onerror：默认情况下在os.walk方法执行时遇到错误会被忽略，也

可以通过一个函数指定遇到错误时的方法，例如放弃、忽略并继续还是提示错误。

·followlinks：设置为true，则通过软链接访问目录。

os.walk的迭代器是一个由文件夹路径dirpath、文件夹名称dirnames、文件名称filenames组成的三元组，其中文件夹路径dirpath是一个字符串，代表遍历的目录；文件夹名称dirnames是一个列表，代表文件夹路径下dirpath所有的文件夹名称，这些文件夹构成了可以做下次迭代的目录（dirpath）；文件名称filenames是一个列表，代表的是一个个文件本身。

步骤5　获取每条内容的分词结果

该步骤用来从内容列表中读取每条内容，然后对其做中文分词。

```
print ('get word list...')
words_list = []  # 分词列表，用于存储所有文件的分词结果
for each_content in all_content:  # 循环读出每个文本内容
    words_list.append(list(jieba_cut(each_content)))  # 将文件内容的分词结果以
列表的形式追加到列表
```

该步骤的实现比较简单，主要包括以下定义：

新建words_list用于存储所有文件的分词结果。

通过for循环读取all_content中的每条文本内容，然后调用jieba_cut做中文分词，分词结果直接使用list方法转换为列表，再使用append方法追加到总分词列表中。

步骤6　建立主题模型。

该步骤是本案例的核心应用过程。

```
print ('train topic model...')
dic, corpus_tfidf, tfidf = text_pro(words_list, tfidf_object=None, training=
练集的文本预处理
num_topics = 3  # 设置主题个数
lda = models.LdaModel(corpus_tfidf, id2word=dic, num_topics=num_topics)  # 通
过LDA进行主题建模
print ('{:*^60}'.format('topic model review:'))
```

```
for i in range(num_topics):  # 输出每一类主题的结果
    print (lda.print_topic(i))  # 输出对应主题
```

本步骤的主要定义如下：

调用定义的文本预处理函数text_pro，得到处理后的字典dic、稀疏矩阵corpus_tfidf和TF-IDF模型对象tfidf。除了返回的对象外，打印输出的内容如下：

分词列表转字典后的数据——token和word键值对：

```
****************token & word mapping review:****************
token:58875 -- word:铅锌
token:25710 -- word:楼兰
token:97812 -- word:平定
token:50462 -- word:供则
token:13113 -- word:考察队员
```

语料库词袋（bag of words）形式的列表——token和token计数列表：

```
********************bag of words review:********************
[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9,
```

TF-IDF向量稀疏矩阵——token和对应向量值：

```
********************TF-IDF model review:********************
[(0, 0.10154448591145282), (1, 0.16383481532598135), (2, 0.14602961468426073
```

通过num_topics设置主题个数为3，该变量会在建立以及打印输出中用到。使用models.LdaModel方法做主题建模，并设置如下参数：

·corpus_tfidf：主题建模的数据集，来自预处理阶段的TF-IDF结果集。

·id2word：分词字典，来自预处理阶段。

·num_topics：要挖掘的主题个数，来自num_topics的定义。

并打印输出每个建模主题如下：

```
********************topic model review:*********************
0.003*"比赛"     +     0.002*"搜狐"     +     0.002*"登录"     +     0.001*"欧洲
杯"  +  0.001*"是"  +  0.001*"体育"  +  0.001*"图"  +  0.001*"球队"  +  0.001*"球
员" + 0.001*"北京"
0.002*"散布"  +  0.002*"民族"  +  0.002*"稳定"  +  0.002*"标题"  +  0.001*"犯
罪" + 0.001*"社会" + 0.001*"谣言" + 0.001*"赌博" + 0.001*"教唆" + 0.001*"封建迷
信"
0.003*"小区"  +  0.002*"编号"  +  0.002*"户型"  +  0.002*"面积"  +  0.002*"装
修"   +   0.002*"建筑面积"   +   0.001*"楼层"   +   0.001*"室"   +   0.001*"有效
期" + 0.001*"厅"
```

步骤7　主题预测。

主题模型除了可以针对特定数据集做主题分析外，还可以对新的文本做主题预测，常用于非监督式的主题归类（这点跟KMeans非常类似）。这里通过给定一个新的文本集article.txt，对其所属的主题做预测。

```
print ('topic forecast...')
with open('article.txt') as f:  # 打开新的文本
    text_new = f.read()  # 读取文本数据
text_content = data_parse(data)  # 解析新的文本
words_list_new = jieba_cut(text_new)  # 将文本转换为分词列表
corpus_tfidf_new = text_pro([words_list_new], tfidf_object=tfidf, training=F
文本数据集的预处理
corpus_lda_new = lda[corpus_tfidf_new]  # 获取新的分词列表（文档）的主题概率分布
print ('{:*^60}'.format('topic forecast:'))
print (list(corpus_lda_new))
```

本步骤的定义过程如下：

使用with　open方法打开文件article.txt并使用文件对象的read方法读取内容。

通过data_parse（data）解析出文件内容中的文本。

使用jieba_cut（text_new）将文本转换为分词列表。

调用text_pro对新文本做预处理，在函数参数传值过程中，需要注意以下几点：

·[words_list_new]：数据集对象是列表形态，即使列表中只有一个列表元素。

·tfidf_object：使用从训练阶段返回的TF-IDF对象tfidf。

·training：设置为False使用预测阶段的功能。

通过lda[corpus_tfidf_new]获取新内容的主题预测概率分布，将其转换为列表后打印输出。整个输出内容如下：

```
topic forecast...
****************token & word mapping review:***************
token:0 -- word:未能
token:1 -- word:身着
token:2 -- word:内
token:3 -- word:来自
token:4 -- word:成为
********************bag of words review:********************
[(0, 1), (1, 1), (2, 1), (3, 1), (4, 2), (5, 4), (6, 2), (7, 1)... (185, 1),
**********************topic forecast:***********************
[[(0, 0.6573075077659087), (1, 0.30406664373195513), (2, 0.0386258485021361
```

由于“bag of words review”中的内容较长，这里将其中的部分内容省略，只保留首尾数据示例。

8.7.5 案例数据结论

从主题建模得到的结果看：

（1）第一类主题

0.003*"比赛"+0.002*"搜狐"+0.002*"登录"+0.001*"欧洲杯"+0.001*"是"+0.001*"体育"+0.001*"图"+0.001*"球队"+0.001*"球员"+0.001*"北京"

主题中权重比较高的词语是比赛、搜狐、登录等，权重值分别为0.003、0.002和0.002；其次是欧洲杯、体育、球队等，权重值均为0.001。因此大体可以判断该主题是与比赛、体育相关。

（2）第二类主题

0.002*"散布"+0.002*"民族"+0.002*"稳定"+0.002*"标题"+0.001*"犯罪"+0.001*"社会"+0.001*"谣言"+0.001*"赌博"+0.001*"教唆"+0.001*"封建迷信"

主题中权重比较高的词语是散布、民族、稳定、标题等，权重值均是0.002；其次是犯罪、社会、谣言等，权重值均为0.001。因此大体可以判断该主体与政治、社会、新闻等主题相关。

（3）第三类主题

0.003*"小区"+0.002*"编号"+0.002*"户型"+0.002*"面积"+0.002*"装修"+0.002*"建筑面积"+0.001*"楼层"+0.001*"室"+0.001*"有效期"+0.001*"厅"

主题中权重比较高的词语是小区、编号、户型、面积、装修、建筑面积等，权重值均为0.002；其次是楼层、室育、有效等，权重值均为0.001。因此大体可以判断该主题是与生活、社区、民生主题相关。

对新主题的预测上，通过结果可以发现，该数据集属于第一、二、三主题的概率是65.7%、30.4%、3.9%。因此新数据集更偏向于第一主

题，即比赛、体育类话题。读者可查看源文件，该文件确实属于该内容方向。

8.7.6 案例应用和部署

对于此类以文本挖掘为主的分析类案例，后期的主要应用方向包括：

·从各个类别的主题中提取关键字，并将关键字作为各个主题的SEO关键字优化主题；

·不断增加新的文本数据，然后将每次的主题关键字做对比，查找分析新出现的主题关键字，建立对于特定主题的分析和后续内容运营机制；

·基于历史主题建模结果做对比，发现各个主题中的新词、新趋势和新话题点；

·基于主题关键字，建立或优化主题页的自动关键字和自动摘要信息；

·基于具有显著性的关键字（较高权重），将所有文章进行重新类别划分或优化，使主题间的关联性和话题紧密型更强。

8.7.7 案例注意点

本案例实施过程中，需要注意以下几点：

·在做预测集应用时，所有的文本预处理过程都需要重新做一遍，但TF-IDF模型不需要重新做，只需要应用训练好的模型即可。

·主题类别的划分，严重依赖于中文分词的结果。本案例中，只保留具有特定意义的词语，而“有意义”的认知依赖于具体应用环境；通常情况下，名词、动词等类型的词语在主题提炼中更具有显著性意义。

·新的预测集在做文本预处理时，需要保持跟训练集相同的格式（此案例中都是嵌套列表）。

·如果有大量的文本数据集，可以将训练好的LDA（以及其他模型）持久性保存到硬盘，使用save和load方法可以实现。这样可以避免所有数据对象都存储在内存中，由于数据计算容量的增加导致内存溢出等问题。

8.7.8 案例引申思考

（1）如何提高结巴分词效率

案例中的结巴分词模块，对于大文本下的分词效率比较低。结巴分词支持并行分词，它可以将目标文本按行分隔后，把各行文本分配到多个Python进程并行分词然后归并结果，从而获得分词速度的提升。并行分词基于Python自带的multiprocessing模块，但是目前暂不支持Windows。

（2）关于有效主题数量的确定

主题模型中关于主题个数的确定，跟KMeans算法中的K非常类似，如果是针对新的文本集，在没有任何先验经验的前提下要获得有意义的主题是比较困难的，这需要读者通过多次实验获的。

8.8 案例：基于多项式贝叶斯的增量学习的文本分类

8.8.1 案例背景

文本分类是对内容做类别划分的常用场景。本案例是从一堆新闻文件中通过对文本内容建立分类模型，通过增量学习的方式实现对未知数据的预测。

案例数据源文件有两部分，第一部分在“附件-chapter8”中的news_data.tar.gz压缩包中，该压缩包包含10个文件，这些是用来做主题模型的训练集；另外一个文件在相同目录下，名为test_sets.txt，该文件是用来做每次增量学习后的测试和检验；最后一个是article.txt，用来做分类预测。案例的程序文件chapter8_code2.py也在数据源目录之下。

本案例程序的默认工作目录为“附件-chapter8”（如果不是，请cd切换到该目录下，否则会报“IOError:File article.txt does not exist”）。

8.8.2 案例主要应用技术

本案例用到的主要技术包括：

·数据预处理：字符串全角转半角、XML文件内容解析、文本转稀疏矩阵。

·数据建模：基于增量学习的分类器。

主要用到的库包括：re、tarfile、os、Numpy、bs4、Sklearn，其中Sklearn是核心。

本案例的重点技术有3个：

·使用bs4的BeautifulSoup做XML文件内容解析，该内容已经在8.7节中介绍过。

·文本转稀疏矩阵，基于Sklearn的HashingVectorizer库实现，而非之前介绍过的分词技术。

·使用sklearn中的MultinomialNB（多项式朴素贝叶斯）做分类学习，并基于增量学习的策略实现文本分类训练和预测。

8.8.3 案例数据

本案例的数据与上个案例的数据格式相同，具体参照8.7.3节。

8.8.4 案例过程

步骤1 导入库。

```
import re
import tarfile  # tar压缩包出库
import os  # 操作系统功能模块
import numpy as np
from bs4 import BeautifulSoup  # 用于XML格式化处理
from sklearn.feature_extraction.text import HashingVectorizer  # 文本转稀疏矩
阵
from sklearn.naive_bayes import MultinomialNB  # 贝叶斯分类器
from sklearn.metrics import accuracy_score  # 分类评估指标
```

本案例主要用到了以下库：

·re：正则表达式库，用来在文本解析时提取标签。

·tarfile：用来解析原始数据压缩文件。

·os：用来判断数据是否存在以及遍历目录文件。

·Numpy：基础数据处理模块。

·bs4：这里用到了其中的BeautifulSoup做XML文件内容解析。

·HashingVectorizer：文本转稀疏矩阵。

·MultinomialNB：贝叶斯分类器。

·accuracy_score：分类评估指标。

步骤2 定义功能函数，包括全角转半角、解析文件内容、交叉检验、word to vector、label to vecotr。

全角转半角：

```
def str_convert(content):
    '''
    将内容中的全角字符，包含英文字母、数字键、符号等转换为半角字符
    :param content: 要转换的字符串内容
```

```
    :return: 转换后的半角字符串
    '''
    new_str = ''  # 新字符串
    for each_char in content:  # 循环读取每个字符
        code_num = ord(each_char)  # 读取字符的ASCII值或Unicode值
        if code_num == 12288:  # 全角空格直接转换
            code_num = 32
         elif (code_num >= 65281 and code_num <= 65374):  # 全角字符（除空格）
根据关系转化
            code_num -= 65248
        new_str += unichr(code_num)
    return new_str
```

该步骤与8.7.4节“步骤2”中的“全角转半角”内容一致，具体解析请查阅上个案例。

解析文件内容：

```
def data_parse(data):
    '''
    从原始文件中解析出文本内容和标签数据
    :param data: 包含代码的原始内容
    :return: 以列表形式返回文本中的所有内容和对应标签
    '''
    raw_code = BeautifulSoup(data, "lxml")  # 建立BeautifulSoup对象
    doc_code = raw_code.find_all('doc')  # 从包含文本的代码块中找到doc标签
    content_list = []  # 建立空列表，用来存储每个content标签的内容
    label_list = []  # 建立空列表，用来存储每个content对应的label的内容
    for each_doc in doc_code:  # 循环读出每个doc标签
        if len(each_doc) > 0:  # 如果dco标签的内容不为空
             content_code = each_doc.find('content')  # 从包含文本的代码块中找到
doc标签
            raw_content = content_code.text  # 获取原始内容字符串
            convert_content = str_convert(raw_content)  # 将全角转换为半角
            content_list.append(convert_content)  # 将content文本内容加入列表

            label_code = each_doc.find('url')  # 从包含文本的代码块中找到url标签
            label_content = label_code.text  # 获取url信息
            label = re.split('[/|.]', label_content)[2]  # 将URL做分割并提取子
域名
            label_list.append(label)  # 将子域名加入列表
    return content_list, label_list
```

该函数主要用来从原始文件中解析出文本内容和标签数据。函数输入参数：

·data：包含代码的原始内容。

函数返回：以列表形式返回文本中的所有内容和对应标签。

该函数的具体定义如下：

·使用BeautifulSoup库建立处理对象raw_code，这里指定的解析库是lxml。

·对raw_code使用find_all方法查找所有的doc标签，将返回的列表结果赋值给doc_code。

·分别新建空列表对象content_list和label_list，用来存储每个content标签的内容及其对应标签。

·使用for循环读出每个doc标签对象each_doc，先判断其是否为空，如果不为空，则从each_doc中使用find方法查找content标签，然后使用text属性获得标签内的内容，再调用str_convert函数将其中的全角转换为半角，最后将结果追加到列表content_list。使用相同的思路从each_doc中提取URL信息，然后使用re正则表达式库根据/和.做字符串分割，从得到的结果中提取第3个字符（子域名）作为标签加入到标签列表。

·最后返回内容列表和标签列表。

由于在原始文本中，url的子域名代表该文本所属的主题分类，因此直接从URL中提取子域名可以做该内容的分类标签。

相关知识点：使用re.split对内容做分割操作

在Python中，对字符串做分割是常用操作。默认的字符串支持split方法做分割，并且可指定分割符号。例如通过如下操作可以将字符串s以逗号为分隔符分割为6个元素对象：

```
s = 'a,2,3,4,f,5'
print (s.split(','))
```

但是如果要分割的对象中包含多个分割字符规则，那么可以使用正则表达式库的split方法做分割。re.split的主要参数如下：

```
re.split(pattern, string, maxsplit=0, flags=0)
```

·pattern：分割规则。

·string：要分割的字符串。

pattern是正则表达式灵活处理字符的核心，在本案例中，我们使用了最简单的并列字符的方法来表示，基于/或.都分割。除此以外，正则表达式还支持多种规则模式。

1）关于对象的表示方法：

首先是正则表达式对于不同对象的表示方法，例如数字、字母等，如表8-1所示。

表8-1　常用正则表达式的对象表示方法

正则表达式	表示的对象
[0-9]	0123456789 任意数字
[a-z]	从 a 到 z 的任意小写字母
[A-Z]	从 A 到 Z 的任意大写字母
\d	等同于 [0-9]
\D	非数字，等同于 [^0-9] 匹配非数字
\w	等同于 [a-z0-9A-Z_] 匹配大、小写字母，数字和下划线
\W	等同于 [^a-z0-9A-Z_] 等同于上一条取非
.	代表任意字符

2）关于对象的位置：

在对某个对象做匹配时，对象可能具有特定的位置属性，例如字符的开头、结尾等，此时可以利用其位置属性做规则匹配，如表8-2所

示。

表8-2　常用正则表达式的位置表示方法

正则表达式	表示的位置
^	匹配字符串的开头，在多行模式下匹配每行开头
$	匹配字符串结束，在多行模式下匹配每行末尾
\A	仅匹配字符串开头
\Z	仅匹配字符串末尾

3）关于对象的次数：

在匹配过程中，可能我们希望对其出现的次数做限制，例如出现5个数字，3个字母等，此时我们需要次数控制规则，如表8-3所示。

表8-3　常用正则表达式的此时表示方法

正则表达式	表示的次数
*	匹配前面的字符或者子表达式 0 次或多次
+	匹配前一个字符或子表达式一次或多次
?	匹配前一个字符或子表达式 0 次或 1 次重复
{m,n}	匹配前一个字符或子表达式至少 *m* 次至多 *n* 次
{m,}	匹配前一个字符或者子表达式至少 *m* 次
{n}	匹配前一个字符或子表达式 *n* 次

4）关于特殊匹配模式：

·很多时候，我们会有一些特殊的匹配模式，例如：

·匹配多个并列条件，此时使用|来表示。

·匹配字符集，使用[]表示，[]中可以使用表8-1中的规则来定义字符对象。

·对字符集的总体取非操作，使用[^]。

·定义一个字符集区间，例如使用[1-5]定义一个区间。

交叉检验：

```
def cross_val(model_object, data, label):
    '''
    通过交叉检验计算每次增量学习后的模型得分
    :param model_object: 每次增量学习后的模型对象
    :param data: 训练数据集
    :param label: 训练数据集对应的标签
    :return: 交叉检验得分
    '''
    predict_label = model_object.predict(data)  # 预测测试集标签
     score_tmp = round(accuracy_score(label, predict_label), 4)  # 计算预测准确率
    return score_tmp
```

本函数通过交叉检验计算每次增量学习后的模型得分。输入参数：

·model_object：每次增量学习后的模型对象。

·data：训练数据集。

·label：训练数据集对应的标签。

函数返回：交叉检验得分。

函数功能具体实现如下：

直接调用增量学习对象model_object的predict方法对测试数据集data做测试，并对照其真实label做数据检验，最后返回每次得分。其中：

·使用指定的accuracy_score方法获得预测标签与真实标签的预测准确率。

·使用round（）方法保留结果为4位小数。


这里没有使用普通的cross_val_score方法做多折交叉检验，原因是cross_val_score方法中默认使用函数对象的fit方法做训练，而不支持增量学习的方法，因此无法对增量学习的效果做验证。所以，这里通过“手动”的方法做结果测试。

```
#word to vector
def word_to_vector(data):
    '''
    将训练集文本数据转换为稀疏矩阵
    :param data: 输入的文本列表
    :return: 稀疏矩阵
    '''
          model_vector   =   HashingVectorizer(non_negative=True)     #   建立
HashingVectorizer对象
    vector_data = model_vector.fit_transform(data)  # 将输入文本转化为稀疏矩阵
    return vector_data
```

本函数用来将训练集文本数据转换为稀疏矩阵。在之前的章节中，我们介绍过使用结巴分词、Sklearn中的TfidfVectorizer、gensim中的TfidfModel建立文本向量模型。这里我们介绍另外一种方法HashingVectorizer。

HashingVectorizer用来将文本文档集合转化为token发生次数的集合（这点跟上个案例中gensim的doc2bow思路很像），其结果是用token发生次数（或者二进制信息）的scipy.sparse稀疏矩阵来表示文本文档信息。这种方法的优势在于：

·无须在内存中存储任何字典信息，因此非常适合大数据集的计算。

·没有任何对象的“状态”信息，因此可以很快的执行pickle或un-pickle操作。

·非常适合在增量学习或者管道方法中应用，因为其在fit期间没有任何状态变更。

如何理解HashingVectorizer的状态？在之前的章节中，我们要做预测应用时，一般都需要针对训练集和预测集分阶段执行，原因是很多对象都有固定的转换或计算模式，该模式在fit期间产生，在训练或预测时都是用相同的fit对象的计算模式。例如使用PCA做降维，先对其做fit操作形成降维对象，然后分别对训练集和预测集做降维。而HashingVectorizer则在fit方法下其对象不产生任何状态的变更，这点决定了我们不需将其对象存储下来，也不需要分阶段针对不同数据集做区分应用。

该函数的实现过程如下：

·通过HashingVectorizer（non_negative=True）建立HashingVectorizer对象，指定结果非负。

·model_vector.fit_transform（data）将输入文本转化为稀疏矩阵，并返回其结果。

```
#label to vecotr
def label_to_vector(label, unique_list):
    '''
    将文本标签转换为向量标签
    :param label: 文本列表
    :unique_list: 唯一值列表
    :return: 向量标签列表
    '''
     for each_index, each_data in enumerate(label):  # 循环读取每个标签的索引及对应值
         label[each_index] = unique_list.index(each_data)  # 将值替换为其索引
    return label
```

本函数用来将文本标签转换为向量标签，例如将['sports'，'house'，'news']转换为[0，1，2]。输入参数：

·label：原始文本标签列表。

·unique_list：标签唯一值列表。

函数返回：向量标签列表。

函数的实现过程如下：

·先通过for循环结合enumerate（）方法，从label中读取每个索引及其对应文本标签值。

·然后使用列表赋值方法，逐个将其文本值替换为索引值。

·最后返回新的向量标签列表。

步骤3　解压缩文件，该步骤与8.7.4节的“步骤3”完全一致，具体解释请参照其内容。

```
if not os.path.exists('./news_data'):  # 如果不存在数据目录，则先解压数据文件
    print ('extract data from news_data.tar.gz...')
    tar = tarfile.open('news_data.tar.gz')  # 打开tar.gz压缩包对象
    names = tar.getnames()  # 获得压缩包内的每个文件对象的名称
    for name in names:  # 循环读出每个文件
        tar.extract(name, path='./')  # 将文件解压到指定目录
    tar.close()  # 关闭压缩包对象
```

步骤4　定义对象，该步骤用来定义在接下来的训练过程中用到的全局变量和常量。其中：

```
all_content = []  # 列表，用于存储所有训练集的文本内容
all_label = []  # 列表，用于存储所有训练集的标签
score_list = list()  # 列表，用于存储每次交叉检验得分
pre_list = list()  # 列表，用于存储每次增量计算后的预测标签
unique_list = ['sports', 'house', 'news']  # 标签唯一值列表
print ('unique label:', unique_list)
model_nb = MultinomialNB()  # 建立MultinomialNB模型对象
```

由于上述定义非常简单，在注释中都有解释，在此不再赘述，仅说明几个可能有疑问的定义：

·pre_list：由于每次增量学习时，都会调用增量学习后的对象做预测，因此这里可以分析每次预测与实际值是否相符。

·unique_list：由于我们在做训练之前已经获知内容分类，因此这里直接定义；如果没有该先验经验信息，那么需要单独从列表中获取。需要注意的是，该列表的运算只能全局性运算一次，否则每个文件中由于训练集出现的顺序不同，可能导致不同文件下唯一值列表的顺序不同。例如文件1的唯一值列表是['sports'，'house'，'news']，而文件2的唯一值列表可能是['house'，'sports'，'news']，虽然对应的索引都是[01，2]，但

对应的类别已经不同。

上述过程返回唯一值列表信息如下：（'unique label:'，['sports'，'house'，'news']），该列表将作为新数据集预测的索引结果参照。

步骤5　交叉检验和预测数据集预处理，该步骤用来实现在增量学习过程中涉及的交叉检验和预测数据集的预处理工作。由于在训练过程中会不断调用该信息，因此这里统一处理之后再做后续调用，避免重复计算浪费时间和资源。

```
# 交叉检验集
with open('test_sets.txt') as f:
    test_data = f.read()
test_content, test_label = data_parse(test_data)  # 解析文本内容和标签
test_data_vector = word_to_vector(test_content)  # 将文本内容向量化
test_label_vecotr = label_to_vector(test_label, unique_list)  # 将标签内容向量化
# 预测集
with open('article.txt') as f:
    new_data = f.read()
new_content, new_label = data_parse(new_data)  # 解析文本内容和标签
new_data_vector = word_to_vector(new_content)  # 将文本内容向量化
```

上述实现过程比较简单，基本思路是：

·使用with　open（）方法打开文件并读取文件中的数据，形成原始数据对象，交叉检验和预测数据集都是独立于训练集的数据。

·调用data_parse（）解析出文本内容和对应标签。

·调用word_to_vector（）将文本内容向量化。

·调用label_to_vector（）将标签内容向量化（仅针对交叉检验集）。

步骤6　增量学习，该步骤是本节内容的主要环节。

```
print ('{:*^60}'.format('incremental learning...'))
for root, dirs, files in os.walk('./news_data'):  # 分别读取遍历目录下的根目录、子目录和文件列表
    for file in files:  # 读取每个文件
        file_name = os.path.join(root, file)  # 将目录路径与文件名合并为带有完整
```

```
路径的文件名
        print ('training file: %s' % file)
        # 增量训练
        with open(file_name) as f:  # 以只读方式打开文件
            data = f.read()  # 读取文件内容
        content, label = data_parse(data)  # 解析文本内容和标签
        data_vector = word_to_vector(content)  # 将文本内容向量化
        label_vecotr = label_to_vector(label, unique_list)  # 将标签内容向量化
        model_nb.partial_fit(data_vector, label_vecotr, classes=np.array([0,
量学习
        # 交叉检验
        score_list.append(cross_val(model_nb, test_data_vector, test_label_v
交叉检验结果存入列表
        # 增量预测
        predict_y = model_nb.predict(new_data_vector)  # 预测内容标签
        pre_list.append(predict_y.tolist())

print ('{:*^60}'.format('cross validation score:'))
print (score_list)  # 打印输出每次交叉检验得分
print ('{:*^60}'.format('predicted labels:'))
print (pre_list)  # 打印输出每次预测标签索引值
print ('{:*^60}'.format('true labels:'))
print (new_label)  # 打印输出正确的标签值
```

先通过两层for循环获取目录news_data下的每个文件名。使用os.path.join将目录和文件名连接起来，方便后续按文件名读取内容。然后针对每个文件做3部分应用：增量训练、交叉检验、增量预测。

·增量训练。使用with　open（）方法读取每个文件的数据，然后依次调用data_parse、word_to_vector、label_to_vector函数实现解析文本内容和标签并将内容和标签做向量化转换。使用贝叶斯分类器的partial_fit方法而非fit方法做增量训练。

在第一次增量训练时，必须通过classes来指定分类的类别，后续过程的类别指定是可选的。

·交叉检验。调用cross_val函数，并将在步骤4中处理好的数据以及增量学习的模型对象做检验检查，将结果追加到score_list列表中。

·增量预测。调用贝叶斯增量学习对象的predict方法做预测，将预测结果转换为列表后追加到pre_list。

最后依次输出每次交叉检验得分、预测标签索引值、正确的标签值。

上述代码完成后返回如下信息：

```
*****************incremental learning...******************
training file: news.sohunews.010806.txt
training file: news.sohunews.020806.txt
training file: news.sohunews.030806.txt
training file: news.sohunews.040806.txt
training file: news.sohunews.050806.txt
training file: news.sohunews.060806.txt
training file: news.sohunews.070806.txt
training file: news.sohunews.080806.txt
training file: news.sohunews.110806.txt
training file: news.sohunews.120806.txt
*****************cross validation score:******************
[0.8707, 0.9013, 0.9067, 0.9088, 0.9099, 0.912, 0.9147, 0.9142, 0.9147, 0.91
*********************predicted labels:**********************
[[0], [0], [0], [0], [0], [0], [0], [0], [0], [0]]
************************true labels:************************
[u'sports']
```

8.8.5 案例数据结论

从cross validation score得到的结果看，随着每次数据量的增加，交叉检验的得分趋势不断提高，这也证实了增量学习本身对于准确率的提升贡献，但在第8次训练时，总体得分从0.9147下降到0.9142，其中可能包含以下原因：

·第8次的数据集本身是有误的（或者不准确的），导致检验结果下降。

·之前的数据中可能存在有误信息，而第8次本身的信息是准确的，导致第8次的结果略有下降。

从10次的检验结果来看，整体趋势的增长是良好的。

对新数据集的预测时，无论哪个阶段都能准确的预测出其类别归属（'sports'对应的索引值为0）。

8.8.6 案例应用和部署

在此类应用的部署中，有以下几点需要经过改造才能部署到应用环境：

·类别标签的定义，应该从常量定义改为从固定标签获取。但考虑到增量学习时，一般情况下标签值都是固定的，即不能出现第2次训练时的预测标签类别是[1，2，3]，到后期变成[2，3，4]。因此这种一次性操作只需在首次执行时设置即可。同样的，还有在第一次增量学习时，指定的classes值也需要做对应设置。

·数据来源，需要改造为实际读取数据的环境。一般情况下，这种增量学习适合实时计算需求比较高的场景，例如个性化推荐，因此数据来源一般会基于实时产生数据的系统或机制，例如网站分析系统等。

8.8.7 案例注意点

在针对文本做分类时，需要读者注意以下几个问题：

·针对文本的全角和半角转换必不可少，实际上除此以外还包括大小写转换等操作，这些更多的用于英文处理。本案例中没有任何对于大小写转换的操作，原因是在HashingVectorizer中默认通过lowercase=True来实现该功能。

·在针对每个文档（doc）做解析时，由于每个doc下面只含有一个url和content，因此使用的是find而不是find_all方法（当然find_all方法也能实现，但返回的是一个列表）。

·不要直接使用不支持增量学习的交叉检验方法做模型做检验，否则将得到错误的得分信息。

·增量学习的一般趋势是良好的，但如果遇到随着增量学习其准确率不断下降或不稳定的情况，需要检查检验方法是否准确或者分析是否存在数据集标签标定或解析错误的问题。

8.8.8 案例引申思考

（1）关于增量学习的价值

增量学习的优点并不是通过算法或模型本身来提供较高的准确率，而是通过不断有新数据的加入来提高模型的准确率，因此在一定意义上，模型本身的选择以及调参等动作都变得“不那么重要”，因为只要数据足够大，即使再差的模型也会由于掌握了足够的多的数据规律而更加精准的预测新样本，这是增量学习的关键所在。

当然，增量学习还能实现在物理硬件限制（尤其是内存）及其他软硬件不作任何优化的条件下，对于海量数据的训练的支持，是一种非常好的解决大数据量计算问题的有效方法。

（2）关于本案例中涉及到的方法

训练集的文本跟预测集的文本不一致，会导致训练时的中间过程或分类模型无法适用于预测过程，这点在文本分类时非常常见。案例中使用的HashingVectorizer能将词语出现的频率映射到固定维度空间，即使出现新的词语也不会影响固定维度空间的模式，因此非常适合预测应用时新词较多的场景。

HashingVectorizer本身能提供压缩后的稀疏矩阵，其本身就能大量降低对于系统内存的占用，非常适合大数据集下的计算和处理。

贝叶斯分类器广泛应用于文本分类领域，其效果较好。除了本文提到的MultinomialNB外，还包括BernoulliNB和GaussianNB两种方法，他们各自有其适用场景。

8.9 本章小结

内容小结：本章针对文本内容的监督式应用（重点是文本分类）比较少，原因是在此类的应用中最难的是词性标注的部分，即哪些词是积极的，哪些词是消极的，以及对于积极和消极的评分定义各自是怎样的。这些内容大多以“字典”的形式存在，在定义过程中会继承对语言相关知识的理解，而国内目前还没有一个非常准的词典开源出来（包括SentiWordNet、《知网》中文版、NTUSD等在内都不太准确），这直接影响到对标签的标注问题，从而影响到所有后续的处理。因此做一个案例不难，难的是在大规模商业应用时具备较高的准确率，而词典正是影响准确率本身的最大因素。

在自然语言方面，语言本身的相关资源的专业性已经具备稀缺性，因此拥有较高质量的语料库和高质量的词典会成为内容分析和建模的关键，在这方面，百度和腾讯其实更具优势，尤其是后者。

在图片和视频分类等领域，基本上识别和分类效果核心是图片资源的丰富程度以及是否具有满足海量计算的庞大计算资源（相对而言算法本身的重要性是其次），而在这方面具有优势资源的百度、谷歌等更具有潜力。

聚焦于内容本身，其类型包括语言、图片、视频、语音等类型，这些内容在其涉及的所有领域通用性极高；而在一些具有特定专业的领域，也可以通过增加专业知识（例如词典、内容材料等）的方法来提升其内容识别和应用效果。因此，有关内容的应用基本上是市场领头羊赢者通吃的局面。因此，当一些巨头把相关内容服务开放出来时，更多的我们会选择合作和应用，而非完全自研。

重点知识：本章所围绕的是内容分析和运营知识，其中内容数据化运营指标、内容数据化运营分析模型、内容数据化运营分析小技巧以及最后的两个案例是重点，尤其是其中：

·使用BeautifulSoup做不同格式的XML文件内容解析。

·使用正则表达做字符串分割以及匹配。

·使用HashingVectorizer完成文本转稀疏矩阵。

·使用MultinomialNB的增量学习机制来实现大数据量的训练、检验和预测。

外部参考：由于内容涉及的内涵和外延非常多，以下是本书有所涉及但并未深入的部分，读者可通过外部相关资源了解更多：

·SEO是内容运营非常关键的一环，有关这个话题的内容非常庞大。由于目前国内的搜索引擎核心是百度，因此关于SEO的主要对象是围绕百度产生的，更多知识请参考http://baiduseoguide.com/。书籍可以参考《SEO实战密码：60天网站流量提高20倍（第3版）》。

·百度站长平台不仅提供了有关SEO的优化工具和资源入口，更提供了很多在内容建设方面的有效建议及相关工具，具体参考http://zhanzhang.baidu.com/。

·由于当前以自然语言为主的内容仍然占主要因素，对于这部分内容的深入分析和挖掘有两个专门的知识领域：文本挖掘和自然语言理解。有关这两方面的话题，还请读者自行查找相关资料。

·很多行业巨头都提供了开放的内容识别和分析服务，例如百度。有关调用百度API做情感倾向分析以及更多自然语言理解的应用的更多信息，请查阅https://cloud.baidu.com/doc/NLP/NLP-Python-SDK.html#.E5.AE.89.E8.A3.85Python.20SDK。

·Python及相关库在应对大规模数据处理时，往往会有不同的实现方法。除了在本书中我们提到的特征工程、sklearn中设置n_jobs参数、增量学习外，还包括64位Python版本、并行处理、多线程计算等，有关这些内容，需要读者书外了解更多。

·在有关知识、关系等非结构化领域的主题挖掘中，还有一个专门的领域叫做“知识图谱”，通过知识图谱能够将具有不同深度和平行关系的主体对象及其关系通过“图模型”的形式表达出来，有关知识图谱的更多内容，请读者通过书外更多资源加以了解。

应用实践：

·尝试调取一些开放的服务来满足企业内部的内容识别尝试与初步应用，例如文字识别、智能推荐、语音识别等。

·尝试开发一些针对企业内部的特定应用，例如垃圾内容识别、内容协同过滤等，这些内容相对外部可参看资源较多且应用成熟。

第9章　数据化运营分析的终极秘籍

数据化运营分析不止于“分析”，它可以包含更多内容，例如数据化产品、自动模型、智能应用等，这些丰富的数据支持方式使得数据不只是一份报告、文字和说明性结论，而成为企业运营闭环中的一个环节来发挥更大作用。本章将针对撰写出彩的数据分析报告提出5个建议，介绍数据化运营支持的另外4种方式，并在最后介绍提升数据化运营价值度的5种途径。

9.1 撰写出彩的数据分析报告的5个建议

数据分析报告几乎是每个数据分析师的必备技能，撰写一份数据报告看似简单，但是如何出彩却非常难。本节将介绍一些常用技巧来帮助读者撰写出彩报告。

9.1.1 完整的报告结构

完整的报告结构是一份规范报告的基础。它能帮助读者在有限时间内迅速掌握报告要表达的内容，并正确理解报告结论。

标准的数据分析报告的结构通常包括以下几个部分：

·封皮和封底。每个公司都有自己的封皮和封底模板，尤其是在大公司中使用公司统一封皮和封底页非常重要。

·摘要页。摘要页是对报告中内容的概述，方便领导层通过阅读摘要直接了解报告内容而无需阅读整个报告。事实上，大多数领导由于时间和精力限制都只看摘要信息，如果有感兴趣的内容则查看对应报告主体或直接询问分析师。

·目录页。如果报告内容过多，则需要通过目录告诉阅读者包括哪些内容。由于一般的报告内容不会太多，在3～20页之内的报告是比较好的。内容过多会使读者丧失耐心、内容过少又往往无法完整表达内容。对于目录页来讲，需要包含标题和页面两个信息，标题一般出现1级标题即可。

·说明页。说明页是关于报告中数据时间、数据粒度、数据维度、数据定义、数据计算方法和相关模型等内容的特殊说明，目的是增强报告的可理解性，尤其是读者关注的采用什么数据、使用什么方法这些关键信息要讲清楚。

·正文页。正文页是报告的核心，通常使用总-分-总的思路撰写报告，在撰写报告之前一般需要先使用MindManager等思维导图工具画出撰写思路。作为日常报告，除了数据陈列外，一定要有数据结论；而对于数据分析和挖掘类报告，必须要仅仅围绕主题撰写，不可面面俱到。

·附录。如果报告存在外部数据引用、原始数据、数据模型解释等，建议作为附录增放在报告最后，而不要都加入到报告正文内容中，因为大多数人对于数据细节不太关注。

9.1.2 精致的页面版式

要想做出出彩的报告，只使用默认的Excel、Word或PPT模板可不行。精致的页面版式包括以下元素：

（1）配色

配色是一门学问，这关乎到审美，而审美是因人而异的。配色的意义主要体现在以下几个方面：

·协调：让版面看起来更赏心悦目。

·对比：更加明显的突出关键信息。

·划分：区隔不同的内容主体，使内容块更容易被理解。

归属：不同的行业和领域有专门用色，使用特定的颜色更能彰显报告与对象之间的归属关系。

在配色中，尤其需要注意以下几种用法：

·主色：主色是整个报告中的主要色调，一般主色的选择跟报告内容和行业归属有关。

·相近色：相近色是跟主色“相似”的颜色，但是作用是辅助主色做信息表达，相近色不能“抢”主色的焦点。

·互补色：互补色是跟主色“互补”的颜色，它能起到强调、突出以及拉开距离的作用，例如红与绿、橙与蓝、黄与紫就是互为补色的关系。但互补色不宜使用过多。

·无色色系：所谓无色色系指的是没有特定颜色的颜色，这类颜色在色环中是没有的。无色色系包括黑、白、灰三类，这类颜色通常起到辅助设计、颜色修饰或者页面背景色的作用。

（2）留白

留白是中国艺术作品创作中常用的一种手法，极具中国美学特征。留白一词原指书画艺术创作中为使整个作品画面、章法更为协调精美而有意留下相应的空白，留有想象的空间。在数据分析报告中，留白的意义是使得报告更加美观、特定内容更容易获得焦点。

留白应用到数据分析报告中时，一般在页面的四周以及不同内容块之间产生，使用留白时需要注意以下几点：

·不要大面积的留白，那样会更感觉报告缺乏内容。

·留白是相当的，例如行间、段落间自然就有间隙留白，真正的留白要与这些自然间隙相区别。

·留白不能在所有内容上都使用，要能突出重点信息。

（3）对称

对称是设计的基本法则，该法则体现在所有页面元素的设计当中。基本的对称报告垂直和水平两种，对称的使用有以下重要信息点：

·如果元素过多，那么相同对称格式一般用于具有相同属性特征的对象上，这样能起到区分的作用。

·水平对称是大家最常关注的部分，但不要忽视垂直对称。

·对称设计不要靠“眼力”，使用工具中的对称样式设计，因为人眼对于距离的把握是不准确的，例如依靠人眼无法对1像素的差异做准确调整。

·对称可以隔元素体现的，例如四角对称、斜角对称等，这些对称对象之间往往有其他元素。

（4）字体

一般情况下报告默认字体是宋体，但在报告设计过程中要善于利用多种字体表达不同的信息。但汉字（也包含其他语言）都有非常多的字体可供使用，不同的字体具有以下特点，如表9-1所示。

表9-1　常用字体适用场景

字　体	适用场景
微软雅黑	笔画简洁而舒展，易于阅读，适合完美的中英文搭配
楷体	在做具有文化感和传统味的场景中常用，也可以作为注释用文字
宋体	适合大多数场景下的正文内容，尤其是权威、正统的场合下更为适用，大粗体宋体也可以作为标题使用
仿宋体	适合正文内容，可以用于中小号标题，也适合诗词、古典文献和仿古版面
黑体	一般用于排各级大、小标题字，封面字及正文中要突出的部分

除了选择合适的字体外，还有很多内容需要读者注意：

·一份报告内尽量不要使用超过3种以上的字体。

·字体的大小要根据报告的应用场景而定，一般情况下以投影PPT为主的字号要在18以上，普通阅读的正文建议在14～16左右。

·行间距的设计跟排版有关，一般情况下笔者会使用1.3～1.5倍行间距。

·不要使用过于花哨的字体，虽然看起来很有意思，但是会增加阅读成本。

·截图与正文的字体一般都是内容的一部分，不要忽视截图中的字体。

·如果有英文，不要全部使用大写。

·不同字号的文字做对比更能有突出和对比作用，尤其是字号差异明显时。

·文字的信息是否醒目也受背景色影响，注意文字颜色和背景色要

使用互补色。

9.1.3 漂亮的可视化图形

可视化图形是每份报告的必备元素，就笔者的倾向而言，更喜欢简单、平面的图形。

在可视化图像应用中，首先是多种可视化的图形方式，除了传统饼图、线图、柱形图、雷达图、散点图、地图外，还有很多可以表示复杂关系的图形，例如：

·树形图：本书多次用到，用来表示分割关系或递进关系的不同主体，如图9-1中的①。

·桑基图：用来表示不同对象在不同流程或环节之间的相互变化和演变趋势，如图9-1中的②。

·关系图：尤其是对多个元素具有相互关系的表示，例如关联关系，如图9-1中的③。

·象形图：根据不同的数据表达需求，使用图形的方式将其数量表示出来，以达到贴近业务形态的目的和直观效果，如图9-1中的④。

·动态图：此类图形一般以实时数据展示为主，动态展示数据变化；某些数据虽然不是实时计算，但是展示过程中会以动态的形式突出数据的发展过程。如图9-1中的⑤。

·子弹图：通过多个条形图或柱形图的叠加以及颜色对比，突出颜色块的差异性，主要用于完成率、进度相关的表示场景，如图9-1中的⑥。

·热力图：通过在特定图像上叠加不同颜色的半透明区块，实现基于位置的信息标注和可视化的颜色差异对比，常用于页面点击分布、地理位置集中度等，如图9-1中的⑦。

·字符云：通过字符的大小、颜色、位置、偏移等方式，突出不同字符的数量和对比，主要用于关键字或短文字相关的信息展示，如图9-1中的⑧。

·日历图：通过将特定数据与日历相结合，可基于时间做数据对比和分部差异分析，如图9-1中的⑨。

这里不建议读者使用一些看起来很漂亮，但需要询问看法等理解困难的图形，例如：

·玫瑰图：极坐标图围绕极坐标中心点展开，但是信息的展示成圆形，坐标却是垂直或水平形式，会增加读者的阅读成本。如图9-2中的①。

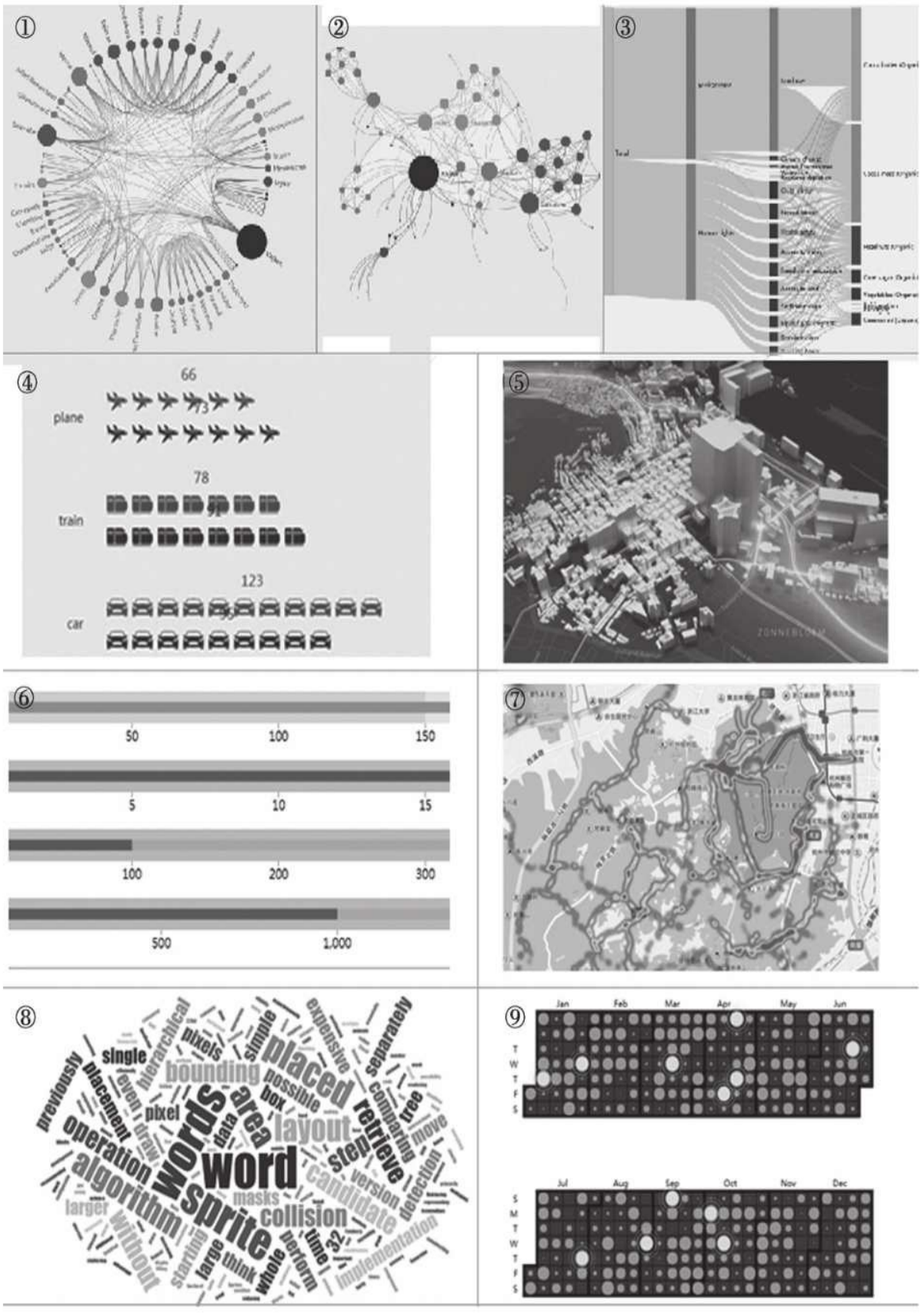

①
②
③
④
66
plane
78
train
123
car
⑤
⑥
50
100
150
5
10
15
100
200
300
500
1,000
⑦
⑧
words
word
sprite
placed
algorithm
operation
collision
layout
retrieve
bounding
area
pixel
pixels
hierarchical
simple
expensive
separately
single
previously
placement
possible
box
tree
compare
move
step
version
detection
candidate
implementation
masks
larger
without
starting
large
think
whole
perform
time
32
⑨
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
S
M
T
W
T
F
S

图9-1　建议的可视化图形

·极坐标图：极坐标图具有特定的应用范围，不适合大多数场景下跟圆周没有关系的信息展示。如图9-2中的②。

有了可视化图形后，很多读者会使用“特效”来增加强调图形，例如增加阴影、立体、透视、边框等。对于此类应用，笔者的建议是根据场合而定：

·如果公司内的工作文化就是需要这些图形形式（例如某些传统行业和单位），那么应该“入乡随俗”。

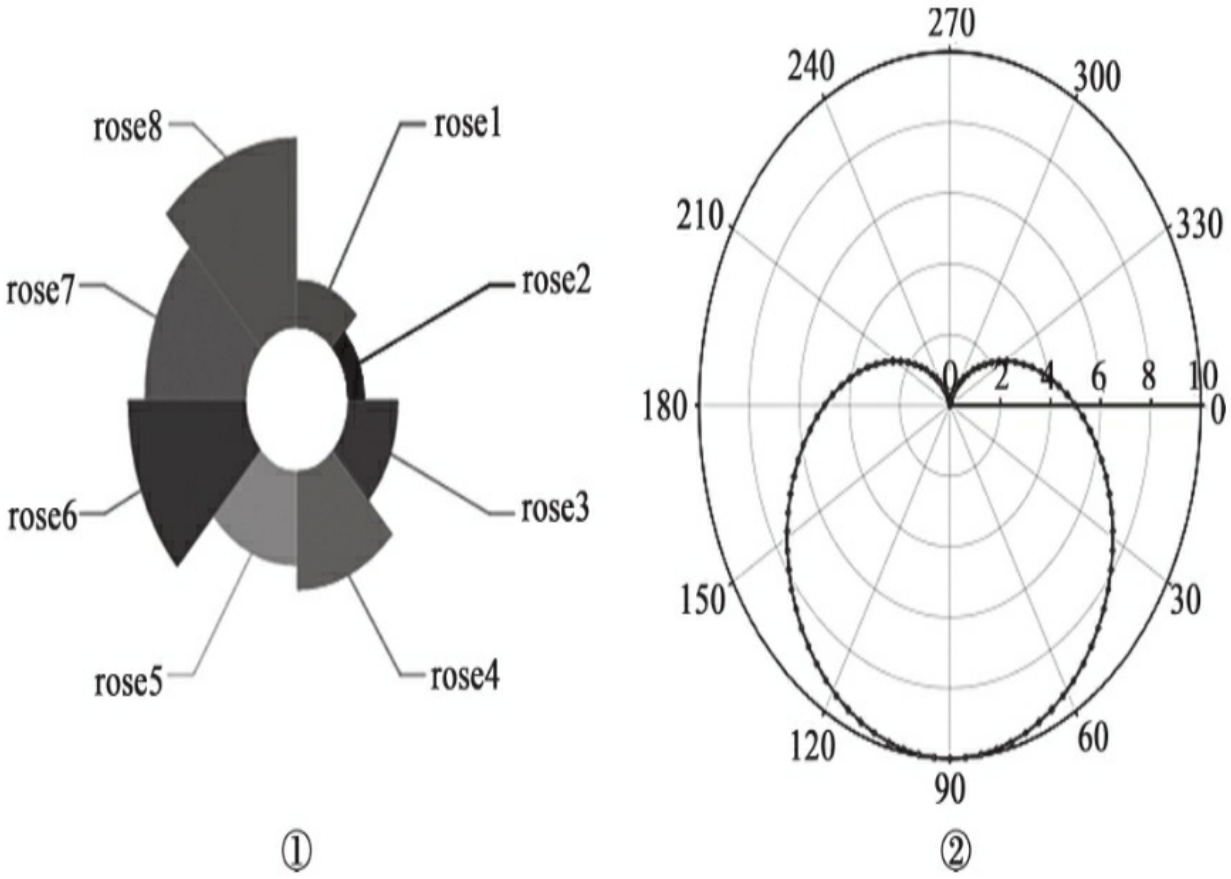


图9-2　不建议的可视化图形

·如果工作环境中没有此类特殊要求，一般情况下不建议使用。图形样式会增加图形信息的表达量，过多使用会增加报告阅读者对图形的关注度。但图形的作用一般是辅助于信息表达，过多吸引注意力是不恰

当的。

9.1.4 突出报告的关键信息

很多人在做报告时，恨不能把所有与报告有关的内容都放到报告里，试图“事无巨细”而不让任何一个信息点被遗忘，但实际上这种做法并不恰当。每份报告都有自己的主题，主题就是这份报告有解决的问题，一份报告通常只解决一个问题就好，不要试图解决所有问题。

常见的报告关键信息不明确的表现是：

·通篇报告只有陈述，没有总结。

·报告内容过多，让报告对象无法知晓到底哪些内容更重要。

·报告总结的内容过多，每个都有“划重点”，让报告对象找不到重点。

关键信息过多与报告没有关键信息是一样的，都会让报告对象找不到重点。

（1）哪些是报告中的关键信息？

是否关键取决于具体场景，以下信息在大多数情况下都是关键信息：

成本支出最多的元素，例如部门、广告渠道、促销活动等。

·收入（尤其是利润）贡献信息，所有对此构成贡献的信息都内关键信息范围之内。

·异常波动信息，在报告中呈现异常的状态、趋势、成分变化的信息。

·跟总体趋势相反的信息，这是潜藏在总体数据规律之下不容易被发现的信息，但往往可能会成为未来发展的起点。

·新业务或尝试性业务信息，很多公司都有探索性业务，对于这些

业务的信息反馈一般要重点表达，因为新业务往往是公司探索和试错的过程，是大方向性的战略活动。

（2）通过哪些方式突出报告关键信息？

在本节之前我们提到了很多关于页面板式和可视化图形可用于突出报告关键信息：

·使用不一样的字体对比突出关键信息。

·使用强烈的字号对比（通常是更大号字）突出关键信息。

·使用与正文不同的颜色（包括背景色、字体颜色等）突出关键信息。

·关键信息周围留白。

·使用特殊的图形表示关键信息的结论。

·关键信息的排版“故意”跟整体排版格格不入，往往能强化这种认知。

一般而言，每页报告内容都有自己的关键信息结论，这称之为小结论，每份报告综合各个小结论形成大结论。

9.1.5 用报告对象习惯的方式撰写报告

在撰写报告时，应该从报告对象的角度出发，而非分析师（撰写报告的人）的角度出发来撰写报告。

（1）报告对象是谁？

报告对象是要看数据分析报告的人或人群。确定报告对象是策划分析报告的基础，不同的对象有不同的数据工作方式：

·老板或较高级别的领导侧重于结论，而忽视推导过程。因此言简意赅地说明问题非常重要，不要期望这些领导们能完整地听完报告，很多时候他们会不断发散思维提出很多问题甚至直接终止报告汇报。

·经理等中层管理级别的对象，往往具有丰富的业务工作经验，同时对于数据有一定的理解，因此对于报告的期望侧重于思维清晰、报告完整、可落地应用。

·基层报告对象由于刚进入公司不久或工作经验较少，对于报告内容的需求侧重于可理解、可解释和可应用，而对于其中复杂的推导、完整的结构等方面不太在意。

（2）哪些是报告对象习惯的方式？

报告对象习惯的方式和内容包括：

·报告对象关注哪些内容？——这往往影响报告的书写侧重点。

·报告对象的数据意识如何？——这决定了组织语言时如何表达特定数据关系、结论及来龙去脉，尤其是对于数据规律的解答。

·报告对象有哪些看或读报告的习惯？——这方面的影响包括文字、图形的使用技巧，特殊语言的规范，报告结构，组织顺序等。

·报告对象最多能一次性接受多少信息？——这决定了报告的内容长度及深度。

·报告对象对于数据的信任度如何？——虽然数据工作是相对公正的，但很可能报告对象本身对数据的认知态度不同，这会导致报告汇报期间产生各种问题。例如，如何说明数据取样、如何解释算法逻辑、如何解释数据的正确性、如何将数据结论落地等。

（3）有哪些通用技巧能使数据分析报告更好的匹配报告对象的工作方式？

由于报告对象的差异新很大，在此提出一些通用技巧供读者参考：

·说行话。行话是一个行业和领域内沟通的“共同语言”，行话要求分析师对于行业的认知较深，需要时间和经历积累。

·使用习惯用语。资历越深的报告对象对特定事务的认知越“固化”，要尽量使用他们已经固化认知的语言来表达。例如对于页面浏览量有的人称之为PV、页面曝光量等。

·尽量少使用专有名词。在数据工作中，有很多专有名词，尤其涉及算法时更多。对报告对象而言，大多数人都不太了解算法及其他数据行业专属名词，很多时候也无暇仔细听取解释。

·举例子打比方。对于数据工作方法和工作结论的说明，使用举例子和打比方的方式非常有效，尤其是以企业内的事物举例更加恰当。因为大家对于企业的认知有共同基础。

·往报告对象的以往工作经历上靠。假如分析师了解一些报告对象的过往工作经历，那么可以基于其过往经历做陈述和表达，这往往更能产生共鸣效应。很多报告对象可能有90%的内容听不懂，但就是那10%跟自己的过往经历相关，所以更容易理解。

·使用浅显易懂的数据工作方法和模型。有关数据方法和模型的问题本书已经多次介绍过，尽量使用大家都容易理解的方法来工作，涉及算法解释时，决策树、线性回归、相关分析、聚类等要比神经网络、主成分分析、SVM等更容易被理解。

9.2 数据化运营支持的4种扩展方式

对数据化运营而言，其产出成果可能更多地集中在分析报告上。但是在很多场景下，数据有更丰富的表示方式以及服务于企业运营的方法。

9.2.1 数据API

数据API是给数据应用方提供的轻型数据服务，该服务一般用于第三方主体的数据请求。基于数据API的方式获取数据有以下几种优势：

·便于控制请求数量，建立良好的流量管理策略，防止服务过载。

·利于授权和访问控制，尤其是涉及安全方面的数据监控的实现。

·利于数据商业化变现，只需按使用量付费，例如请求次数、请求数据量等。

·降低数据交互的开发难度，不同的数据应用方都能采用统一的数据接口开发方式，从而提高开发效率。

·降低数据基础设施，例如宽带、服务器、机房等投入，使得数据应用更容易“轻装上阵”。

数据API一般以验证、识别、授权等服务为主，很多数据API也能提供较粗粒度的数据概要，而对于细粒度的数据通常不会以API的方式提供出来。

数据API的商业价值在于它能帮助企业实现数据资产的变现，数据将不再局限于企业内部流转，更能将其作为“产品”或“服务”打包出来，无论在企业价值的评估上，还是资产负债表上都能将数据价值直接量化出来。

9.2.2 数据模型

数据模型是将常用的具有固定思路和工作方法的应用固定下来，然后通过程序封装为一个小应用，这种应用一般具有特定的运行机制。在本书的内容中，我们介绍过多次将数据应用模型固定化的思路和方法。

能被固定下来的应用一般具有以下特点：

·数据源以及数据从产生规则相同，没有新增或变更。

·数据处理流程可以固化，并能支持一定程度的容错性。

·能适应较大数据量，不会随着数据量的增加而出现重大运行问题。

·应用的运行就有特定触发机制，无需依靠人力“手动”触发，例如基于时间周期、基于特定事件等。

数据模型的固化一般集中在具有“应用性”的算法基础上，例如分类、回归、关联、异常检测等，其输出结果具有固定模式；而对于侧重于“分析性”的算法和应用上则不适合。

9.2.3 数据产品

数据产品大多数以辅助决策支持产品为呈现方式，它的侧重点是通过产品化的形式提供辅助决策支持，是数据驱动业务的重要方式。

数据产品聚焦于数据，通过特定方式提供数据结论，但本身并没有任何运营类功能，只能与运营系统对接或依赖运营人员的理解来发挥作用。

数据产品是当前数据化运营支持的重要载体，更是企业内部实现数据化运营的主要支撑方式，任何企业要实现成熟的数据应用工作必须有对应的数据产品做支持。

常见的数据产品包括：

·数据报表系统：数据报表系统是数据产品的最初阶段，这类产品只提供统计报表功能，而不支持对报表结果做分析。例如指数类、统计类都属于这种应用。

·数据分析系统：数据分析系统在报表系统基础上，增加了分析功能。不同分析系统提供的分析能力不同，大体分析能力包括下钻、临时分析、即席分析、可视化分析、多维分析等。这种分析功能的加入使得报表不再只有结果而没有过程，能满足初级基于简单分析方法的分析应用。

·流量智能系统：数据智能系统包括数据挖掘类型、商业智能类等多种产品形态，这类产品在功能上增加了很多可以实现智能化的机制，例如告警控制、算法模型、预测能力等，它们可以辅助于人类在海量数据和潜在规律上的挖掘和探索。

从数据产品的进化层次上，三个层级的产品依次递进，一般情况下各个企业内是循序渐进实现的。因为这里面不同层次不同的技术挑战外，还涉及业务应用的问题，业务方的数据应用以及数据工作文化的形成往往也是一个渐进过程。

9.2.4 运营产品

运营产品往往披着一层运营的外衣，但内在其实是数据主导的应用。这类应用的核心是数据工作流程，落地点是业务产品。

运营产品相对于数据产品的进化之处在于，运营产品无需依赖于业务方的理解和执行，而是依赖于自动化IT工作机制自动完成业务流程和运转。它能有效解决以下问题：

·业务方由于自身因素对数据价值落地的干扰，例如数据理解能力、工作经验、实施能力等。

·提高数据落地的效率，所有环节自动化完成。

·将决策、推导、执行、验证结合起来，形式一体化数据工作机制，避免各个环节分散到不同部门或体系中导致的执行难的问题。

常见的由数据主导的运营产品例如：

·个性化推荐：个性化推荐系统是互联网和电子商务发展的产物，它是建立在海量数据挖掘基础上的一种高级商务智能平台，直接在网站以及其他提供展示信息的载体上直接为客户提供个性化信息。

·精准营销系统：精准营销系统是运营产品的一种，它往往能细分出多种产品，例如重定向和再营销系统、精准广告系统等。这些精准营销系统的背后是基于对用户行为的挖掘而找到最佳广告投放对象，再将广告投放功能“嫁接”上便具有了投放功能。

运营产品很多时候往往无法直接区分出来，原因就是其运营的属性是显性的，而数据的属性是隐性的，对于数据在运营产品背后发挥的作用，往往只有产品设计人员了解，而运营使用人员可能不清楚。但正是这种模糊了边界的应用，才将数据与运营无缝结合起来，使得二者密不可分，数据不再是可有可无的角色。

9.3　提升数据化运营价值度的5种途径

9.3.1　数据源：不只有结构化的数据，还有文本、图片、视频、语音

虽然在每个章节都尽量介绍一些非结构化数据的工作方法，但本书的主体内容仍然围绕结构化数据展开。实际上，非结构化数据现在正在成为海量数据的核心，包括文本、图片、视频、语音等。原因在于两方面：

一是非结构化数据的容量会越来越大，并成为大数据的核心。我们知道结构化数据的存储一般比较小，但是非结构化的数据容量却非常大。使用iPhone拍摄一张普通的非全景照片，其容量大概有3M～5M，相当于3145728～5242880个字节（Byte），如果是汉字的话大概有157万～262万个（每个汉字占2个字节）。

二是非结构化数据能提供更丰富的信息要素，这点是结构化数据所不具备的。例如：以从视频中做人脸分析为例：从视频中可以分析目标对象所处的环境因素、面部表情、年龄特征等，这些信息都是客观存在并可以被分析的；结合视频本身的序列性和动态性，又能基于时间和空间的演变规律提取出来。

基于非结构化数据的这些特性，再加上当前相对廉价和低成本的数据获取、处理和计算成本，各个企业针对这些内容的研究方兴未艾。

9.3.2 自动化：建立自动任务，解除重复劳动

重复劳动是对数据工作者价值产出的重大阻碍，人类的优势在于创造性的工作，重复劳动正是电脑的优势。幸运的是，我们使用Python可以实现自动化的工作机制。

常见的可以自动化的工作内容包括：

·数据抽取：数据抽取也称“取数”，有固定规律的取数工作一般都可以基于固定机制自动抽取。比如每次月报、周报、日报的数据，每月促销活动的数据提取工作。

·数据清洗、数据汇总、数据统计：这类基础的数据预处理工作大多数都可以固定下来。

·分析模型：常用的分析模型，尤其是具有“应用性”的是可以被固定下来的。

·数据结果发送：当数据报表或报告有结果时，经常会需要发送给特定目标对象，这个过程也是可以通过自动化的方式发送的，尤其是日常性报告。

·多系统的数据交互：当通过数据工作产生结果时，如果基于该结果有二次应用，那么应该通过自动化的方式将数据存储到目标数据库或指定路径。

9.3.3 未卜先知：建立智能预警模型，不要让运营先找你

预测性工作是极具落地价值的应用，配合特定的算法、阈值或波动区间的定义，我们可以建立良好的预警机制。

（1）预警机制的作用

预警应用具有以下作用：

·提前提防可能出现的问题，避免问题严重化和扩大化。

·在问题的发展过程中就提出问题，并建立预防性解决方案，防止事出突然而导致的手足无措。

·建立正确的数据认知，避免“温水煮青蛙”导致的业务持续下滑而无法感知到危机的问题；同时也避免由于正常波动导致业务方“沾沾自喜”，认为自身运营取得良好成果。

（2）预警线的建立

预警线的建立一般包含预警上限和下限两种，如图9-2所示。

图中在正常波动上限之上的值域是异常增长区间，属于趋好的异常值，这种值的结果如图9-2中的A点，数据位于正常波动上限的上方。在实际业务中，数据会因为各种原因而导致结果高于或低于预期，定义引起关注的正向值区间，可以帮助业务过滤信息噪点而真正定位和分析异常增长情况。

图中在正常波动区间之下的值域是异常下降区间，属于异常低的情况。该结果已经排除了正常下降波动导致的数据减少，意味着已经出现严重的低于预期的情况。对于该情况需要立即通报相关部门或负责人引起重视。如图中的D。

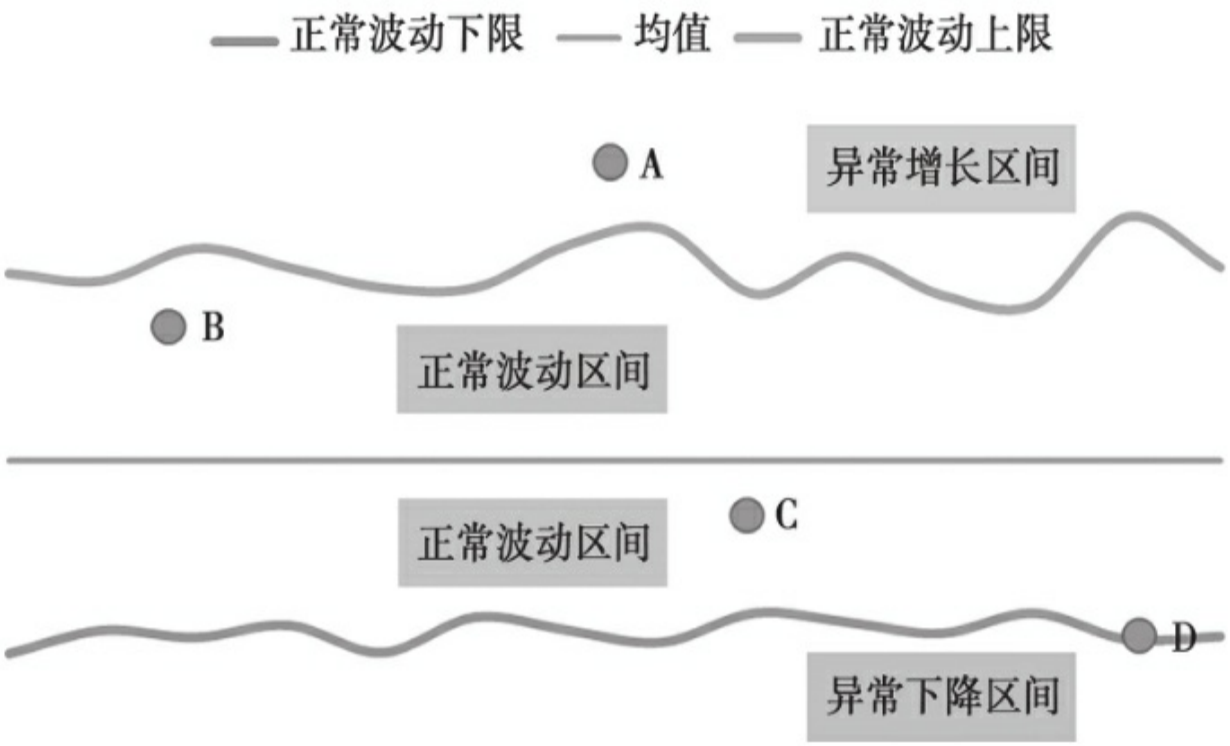


图9-2　数据波动上下限定义

而图中的B和C则由于处于正常波动范围，则无须引起预警。

（3）预警机制的触发

预警机制的检测和触发一般是实时应用，可通过自动触发或手动触发进行业务提醒和优化。

自动触发。这种信息是在建立自定义预警条件后，当触发预警时自动执行响应程序，例如：

·通过邮件、短信等形式告知关联运营部门。

·通过程序将特定问题对象的权限、范围、功能、数据做锁定、限制等，防止问题恶化。

·调用应急预案机制，直接对异常问题做应急处理，这要求提前已经定义好预警机制并且该问题经常发生。

手动触发。这是对自动触发信息的补充，用于检测机器无法识别的

预警情况，尤其适合路径类、关联访问类的信息排除。

9.3.4 智能化：向BI-AI的方向走

BI商业智能（Business Intelligence）和AI人工智能（Artificial Intelligence）是在数据工作领域的两个常见智能化方向。前者已经提出很多年并且应用更加成熟，而后者虽然也产生较早，但直到最近随着大数据的突飞猛进才发展起来。

BI的侧重点在于基于结构化的数据再加上商业理解而产生对过去发生了什么、正在发生什么、为什么会发生、以后会发生什么的解释。出发点是从企业经营思考如何利用数据智能提高数据的驱动能力。

AI的侧重点则更多的在于基于全量数据（包括结构化和非结构化数据）像人类一样思考问题，这里面会涵盖BI的工作内容，但是在数据覆盖面上、工作方式上、计算方法上、实现能力上会有更大突破，主要表现在：

·AI具有更接近于人脑的解决问题的方式，例如神经网络、深度学习等，这些方法是以往BI所不具备的。

·AI能够直接面对视频、图片、语音等非结构化数据并从中提取有效信息，传统BI则不具备这种能力。

·AI的应用更加聚焦于解决问题的场景化应用，例如人脸识别、火焰检测、黄图识别、语音识别等，而不像BI一样覆盖整个商业运营的方方面面。

·AI对于大数据的支持性要远远高于BI，尤其是海量、非结构化特征的数据。

·AI能应用到各个领域，尤其是与人类生活息息相关的领域，而BI则侧重于商业（企业）应用。

因此，我们会看到现在没有一个AI能解决任何问题，实际上每个AI应用一般都只解决一个问题，但在AI应用的领域他们的工作能力已经可以超越人类，这在之前是不可想象的。例如AlphaGo以3比0的总比分战胜排名世界第一的世界围棋冠军柯洁。

当然，无论是AI、BI、CI（客户智能Customer Intelligence）、DI（数据智能Digital Intelligence），虽然口号不同，但最终我们的目的是一致的，那就是如何使用更多的数据更好地为人类服务。

9.3.5 场景化：将数据嵌入运营环节之中

“数据驱动”一词是数据工作者为之奋斗的目标，但很多情况下下我们会发现数据“驱不动”业务。导致这一问题的因素有很多，例如：业务方工作能力、企业数据工作文化、数据工作机制、分析师的推动能力等。但是这些因素中，有一个潜在的因素未被提及，那就是数据根本不在运营环节之中。

数据怎样才算是在运营环节之中？如果没有数据，那么业务工作将开展不下去，这就是数据在运营环节中作为必要一环的体现。但可惜的是当前大多数企业都不具备这一特征。所有公司的商业模式和企业基因决定了其未来的趋向，如果数据不是随着其商业模式共同建立的，那么必然只能作为辅助类角色。

将数据嵌入运营环节，有以下几个建议点：

1）选择核心商业模式所涉及的环节，而非边缘环节。例如企业以财务经营为主，那么就需要将数据工作嵌入到财务运作过程中；如果嵌入营销、运营等环节，虽然也能发挥数据价值，但其价值将不会是最大化的。

2）数据在运营环节中的工作方式应该是不以人力参与为主的。这句话的意思是数据在整个工作环节中需要自动化、智能化参与其中，而不应该靠流程、制度等外部条件的约束来保证实施。数据的工作必须借助于IT系统，而不是人力参与。

3）将数据作为运营的一方主体，而不要仅运营方的辅助对象。在几乎所有的内部运营工作中，数据都是作为客体来测量、优化、改进运营主体的工作，这个过程中的数据的工作都围绕运营工作展开，没有运营就没有数据。但实际上，数据本身就有机会作为运营主体方，基于自身做策划、评估、优化和提升将使数据成为运营方的一种主体存在，而非依赖客体。

9.4 本章小结

内容小结：本章介绍的有关分析报告的撰写技巧、数据化运营支持方式以及提升数据价值度的途径都是笔者多年经验总结，虽然随着时间的发展会有不同的数据主题及热点，但我们发现其实那些“精彩纷呈”的口号下面所蕴含数据工作机制是类似的，只是由于新数据源、新方法和新模式的产生会改变、提升或优化原有数据工作。

重点知识：本章的内容只有3部分，从“立竿见影”的价值产出来看，9.1节的短期价值更大，因此更值得分析师去关注；从长期的视角来分析，9.2节和9.3节的内容更值得管理层去审视。

外部参考：作为一本能够帮助分析师快速提升的书籍，下面列出一些让分析报告更出彩的建议。

·Excel图表展示：使用Excel进行图表信息展示本是一个非常小的知识，而且似乎不值得一提。但当你阅读一本书名为《Excel图表之道》之后，你会发现原来Excel还可以这样。这是一本讲如何通过Excel做出专业级展示效果的书，这本书将让你了解什么是真正的数据生产力。

·制作精彩的PPT。PPT几乎是每个分析师必用工具，但是不是所有的分析师都有很好的设计稿和美感，推荐读者去了解下“秋叶老师ppt”，他做的PPT（及其模板）将让分析师的报告提升N个台阶。

应用实践：本章的内容更篇宏观，如果要实践，先从撰写出一份出彩的PPT开始吧。读者可以在下次写PPT时注意应用本章提到的技巧，同时结合外部参考中提到的两位老师的相关知识，相信你一定会有专业产出。

附录

本附录提供了两部分内容：附录A提供了大量的公开数据集，主要用于机器学习和数据挖掘等大数据场景下的测试和学习；附录B提供了Python数据工作工具箱清单，几乎涵盖了所有Python数据工作中用到的库、函数和方法，可在读者在有特定和应用需求时查询、索引。

附录A　公开数据集

公开数据集指的是不同的公司、组织公开的用于机器学习、深度学习、人工智能等方向大规模数据集合。按照数据工作的不同应用主题方向，主要分为音频数据集、图像和视频数据集、自然语音数据集以及综合数据集。

1.综合数据集

（1）UCI数据集

UCI数据集中包括了众多用于监督式和非监督式学习的数据集，数量大概400多个，其中很多数据集在其他众多数据工具中被反复引用，例如Iris、Wine、Adult、Car Evaluation、Forest Fires等。

每个数据集中都有关于数据实例数、数据产生领域、值域分布、特征数量、数据产生时间、模型方向、是否有缺失值等详细数据介绍，可用于分类、回归、聚类、时间序列、推荐系统等。

推荐度：★★★，推荐应用方向：监督式、非监督式机器学习，数据挖掘。

介绍和下载地址：http://archive.ics.uci.edu/ml/。

（2）UCI KDD数据集

UCI KDD（知识发现）是数据挖掘和可视化的研究项目，专注于大型数据收集中的实体事件关系。它是涉及几所大学的更广泛的KDD项目的一部分，UCI始于2002年10月。

推荐度：★★，推荐应用方向：监督式、非监督式机器学习。

介绍地址：http://kdd.ics.uci.edu/

下载地址：http://kdd.ics.uci.edu/databases/

（3）雅虎Webscope

雅虎Webscope用于为学者和其他科学家在非商业用途中使用。所有数据集已经过审查，以符合雅虎的数据保护标准，包括严格的隐私控制。数据集中包含了多个主题数据集：广告和市场营销、自然语言数据、科学数据、图形和社会化数据、图像数据等7个主题。需要注意的是：数据集只适用于同意数据共享协议的教师和大学研究人员的在学术上使用。

推荐度：★★★，推荐应用方向：监督式、非监督式机器学习、深度学习、自然语言理解等。

介绍和下载地址：https://webscope.sandbox.yahoo.com/

（4）AWS公开数据集

亚马逊提供的数据集涵盖气候、红外图像、卫星遥感、人类微生物、日本人口普查、公共电子邮件档案、歌曲、材料安全、谷歌图书语料库、石油等非常多的主题数据，并且这些数据可直接集成到AWS进行数据挖掘和学习。

推荐度：★★★，推荐应用方向：监督式、非监督式机器学习、深度学习、神经网络、自然语言理解等。

介绍和下载地址：https://aws.amazon.com/cn/datasets/

（5）斯坦福网络数据集

斯坦福网络分析平台（SNAP）是一种用于分析和操纵大型网络的通用高性能系统，其本身使用的网络相关数据也对外开放，包括设计、社区、通信、网络图、互联网、道路、维基百度网络、在线社区和评论等不同主题，可用于分析大型社会和信息网络方面的研究成果。

推荐度：★★★，推荐应用方向：神经网络。

介绍和下载地址：http://snap.stanford.edu/data/index.html

（6）KONECT网络数据集

KONECT数据集是一个大型网络数据集的项目，在科布伦茨-兰道大学网络科学与技术研究所的网络科学和相关领域进行研究。KONECT包含数百种各种类型的网络数据集，包括有向、无向、二分、加权、未加权、签名和评级的网络。KONECT的网络覆盖了许多不同领域，如社交网络，超链接网络、作者网络、物理网络、交互网络和通信网络等。

推荐度：★★★，推荐应用方向：神经网络。

介绍和下载地址：http://konect.uni-koblenz.de/

2.图像和视频数据集

（1）MNIST数据集

机器学习领域内用于手写字识别的数据集，数据集中包含6个万训练集、10000个示例测试集。，每个样本图像的宽高为28×28。这些数据集的大小已经归一化，并且形成固定大小，因此预处理工作基本已经完成。在机器学习中，主流的机器学习工具（包括sklearn）很多都使用该数据集作为入门级别的介绍和应用。

推荐度：★★★，推荐应用方向：机器学习入门。

介绍和下载地址：http://yann.lecun.com/exdb/mnist/

（2）CIFAR 10&CIFAR 100数据集

CIFAR-10数据集由10个类别的60000张32×32彩色图像组成，每个类别有6000张图像。有50000个训练图像和10000个测试图像。数据集的类别涵盖航空、车辆、鸟类、猫类、狗类、狐狸类、马类、船类、卡车等日常生活类别，可用于计算机视觉相关方向。

推荐度：★★★，推荐应用方向：图像处理和图像识别。

介绍和下载地址：http://www.cs.toronto.edu/~kriz/cifar.html/

（3）谷歌Open Images Dataset图像数据集

其中包括大约9百万标注图片、横跨6000个类别标签，平均每个图像拥有8个标签。该数据集的标签涵盖比拥有1000个类别标签的ImageNet具体更多的现实实体，可用于计算机视觉方向的训练。

推荐度：★★★，推荐应用方向：图像处理和图像识别。

介绍地址：https://research.googleblog.com/2016/09/introducing-open-images-dataset.html

下载地址：https://github.com/openimages/dataset

（4）ImageNet数据集

ImageNet数据集是目前深度学习图像领域应用得非常多的一个领域，该数据集有1000多个图像，涵盖图像分类、定位、检测等应用方向。Imagenet数据集文档详细，有专门的团队维护，在计算机视觉领域研究论文中应用非常广，几乎成为了目前深度学习图像领域算法性能检验的“标准”数据集。很多大型科技公司都会参加ImageNet图像识别大赛，包括百度、谷歌、微软等。

推荐度：★★★，推荐应用方向：图像识别。

介绍和下载地址：http://www.image-net.org/

（5）Tiny Images Dataset

该数据集由79302017张图像组成，每张图像为32×32彩色图像。该数据以二进制文件的形式存储，大约有400Gb图像。

推荐度：★★，推荐应用方向：图像识别。

介绍和下载地址：http://horatio.cs.nyu.edu/mit/tiny/data/index.html

（6）CoPhIR

CoPhIR是从Flickr中采集的大概1.06亿个图像数据集，图像中不仅包含了图表本身的数据，例如位置、标题、GPS、标签、评论等，还可提取出颜色模式、颜色布局、边缘直方图、均匀纹理等数据。

推荐度：★★，推荐应用方向：图像识别。

介绍和下载地址：http://cophir.isti.cnr.it/whatis.html

（7）LSUN数据集

国外的PASCAL VOC和ImageNet ILSVRC比赛使用的数据集，数据领域包括卧室、冰箱、教师、厨房、起居室、酒店等多个主题。

推荐度：★★，推荐应用方向：图像识别。

介绍和下载地址：http://lsun.cs.princeton.edu

（8）Labeled Faces in the Wild数据集

该数据集是用于研究无约束面部识别问题的面部照片数据库。数据集包含从网络收集的13000多张图像。每张脸都贴上了所画的人的名字，图片中的1680人在数据集中有两个或更多不同的照片。

推荐度：★★，推荐应用方向：人脸识别。

介绍和下载地址：http://vis-www.cs.umass.edu/lfw/

（9）SVHN

SVHN数据来源于Google街景视图中房屋信息，它是一个真实世界的图像数据集，用于开发机器学习和对象识别算法，对数据预处理和格式化的要求最低。它跟MNIST相似，但是包含更多数量级的标签数据（超过60万个数字图像），并且来源更加多样，用来识别自然场景图像中的数字。

推荐度：★★，推荐应用方向：机器学习、图像识别。

介绍和下载地址：http://ufldl.stanford.edu/housenumbers/

（10）COCO

COCO（Common Objects in Context）是一个新的图像识别、分割和图像语义数据集，由微软赞助，图像中不仅有标注类别、位置信息，还有对图像的语义文本描述。COCO数据集的开源使得近两、三年来图像分割语义理解取得了巨大的进展，也几乎成为了图像语义理解算法性能评价的“标准”数据集。

推荐度：★★★，推荐应用方向：图像识别、图像语义理解。

介绍和下载地址：http://mscoco.org/

（11）谷歌YouTube-8M

YouTube-8M一个大型的多样性标注的视频数据集，目前拥有700万的YouTube视频链接、45万小时视频时长、3.2亿视频/音频特征、4716个分类、平均每个视频拥有3个标签。

推荐度：★★★，推荐应用方向：视频理解、表示学习（representation learning）、嘈杂数据建模、转移学习（transfer learning）和视频域适配方法（domain adaptation approaches）。

数据集介绍和下载地址：https://research.google.com/youtube8m/

（12）Udacity开源的车辆行使视频数据集

数据集大概有223G，主要是有关车辆驾驶的数据，其中除了车辆拍摄的图像以外，还包括车辆本身的属性和参数信息，例如经纬度、制动器、油门、转向度、转速等。这些数据可用于车辆自动驾驶方向的模型训练和学习。

推荐度：★★★，推荐应用方向：自动驾驶。

介绍和下载地址：chttps://github.com/udacity/self-driving-car

（13）牛津RobotCar视频数据集

RobotCar数据集包含时间范围超过1年，测试超过100次的相同路线的驾驶数据。数据集采集了天气、交通、行人、建筑和道路施工等不同组合的数据。

推荐度：★★★，推荐应用方向：自动驾驶。

介绍和下载地址：http://robotcar-dataset.robots.ox.ac.uk/

（14）Udacity开源的自然场景短视频数据集

数据集大概为9T，由3500万个视频剪辑组成，每个视频为短视频（32帧），大约1秒左右的时长。

推荐度：★★★，推荐应用方向：目标跟踪、视频目标识别。

介绍和下载地址：http://web.mit.edu/vondrick/tinyvideo/#data

3.自然语言数据集

（1）MS MARCO

MS MARCO是一种新的大规模阅读理解和问答数据集。在MS MARCO中，所有问题都是从真正的匿名用户查询中抽取的。使用先进的Bing搜索引擎版本，从实际的Web文档中提取数据集中的答案的上下文段落。

推荐度：★★★，推荐应用方向：自然语言理解、智能问答。

介绍和下载地址：http://www.msmarco.org/

（2）Question Pairs

第一个来源于Quora的包含重复/语义相似性标签的数据集。数据集由超过40万行的潜在问题的问答组成。每行数据包含问题ID、问题全文以及指示该行是否真正包含重复对的二进制值。

推荐度：★★★，推荐应用方向：自然语言理解、智能问答。

介绍和下载地址：https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

（3）SQuAD

斯坦福问答回答数据集（SQuAD）是一个新的阅读理解数据集，从维基百科中提炼出的问题组成，每个问题的答案都是相应段落的一段文本。在500多篇文章中有超过10万个问答对。

推荐度：★★★，推荐应用方向：文本挖掘、自然语言理解、智能问答。

介绍和下载地址：https://rajpurkar.github.io/SQuAD-explorer/

（4）Maluuba NewsQA

Maluuba的NewsQA数据集的目的是帮助研究团队建立能够回答需要人为理解和推理技能的问题的算法。它包含了从DeepMind问答数据集中的CNN文章中抽取了120K个常见问题。

推荐度：★★，推荐应用方向：文本挖掘、自然语言理解、智能问答。

介绍地址：https://datasets.maluuba.com/NewsQA

下载地址：https://github.com/Maluuba/newsqa

（5）1 Billion Word Language Model Benchmark

这是一个大型、通用的语言建模数据集，该项目的目的是提供语言建模实验的标准培训和测试，常用于如word2vec或Glove的分布式词语表征。

推荐度：★★，推荐应用方向：文本挖掘、自然语言理解。

介绍和下载地址：http://www.statmt.org/lm-benchmark/

（6）Maluuba Datasets

这是一个用于自然语言理解研究的复杂的人工数据集，主要包括NewsQA和Frames。它主要用于机器阅读理解、面向对象的对话系统、对话界面和加强学习。

推荐度：★★，推荐应用方向：自然语言理解、智能问答。

介绍和下载地址：https://datasets.maluuba.com/

（7）Common Crawl

Common Crawl包含了超过7年的网络爬虫数据集，拥有PB级规模，常用于学习词嵌入。

推荐度：★★，推荐应用方向：文本挖掘、自然语言理解。

介绍和下载地址：http://commoncrawl.org/the-data/

（8）20 Newsgroups

该数据集包含大约20000个新闻组文档，在20个不同的新闻组中平均分配，是一个文本分类的经典数据集，它是机器学习技术的文本应用中的实验的流行数据集，如文本分类和文本聚类。

推荐度：★★，推荐应用方向：文本挖掘。

介绍和下载地址：http://qwone.com/~jason/20Newsgroups/

4.音频数据集

（1）大型音乐分析数据集FMA

该数据集是免费音乐存档（FMA）的转储，这是一个高质量的合法音频下载的互动库。这些数据集中包含歌曲名称、音乐类型、曲目计数

等信息，共计689种歌曲和68种类型。该数据集可用于音乐分析。

推荐度：★★★，推荐应用方向：音乐分析挖掘。

介绍和下载地址：https://lts2.epfl.ch/datasets/fma/

（2）音频数据集AudioSet

谷歌发布的大规模一品数据集，AudioSet包括632个音频事件类的扩展类目和从YouTube视频绘制的2084320个人类标记的10秒声音剪辑的集合。类目被指定为事件类别的分层图，覆盖广泛的人类和动物声音，乐器和风格以及常见的日常环境声音。

推荐度：★★★，推荐应用方向：音乐、人声、车辆、乐器、室内等自然和人物声音分析挖掘。

介绍和下载地址：https://github.com/audioset/ontology

（3）2000 HUB5 English Evaluation Transcripts

该数据集由NIST（国家标准与技术研究院）2000年发起的HUB5评估中使用的40个英语电话对话的成绩单组成，其仅包含英语的语音数据集，百度最近的论文《深度语音：扩展端对端语音识别》使用的是这个数据集。

推荐度：★★★，推荐应用方向：音乐、人声、车辆、乐器、室内等自然和人物声音识别。

介绍和下载地址：https://catalog.ldc.upenn.edu/LDC2002T43

（4）LibriSpeech

该数据集为包含文本和语音的有声读物数据集，由Vassil Panayotov编写的大约1000小时的16kHz读取英语演讲的语料库。数据来源于LibriVox项目的阅读有声读物，并经过细致的细分和一致。

推荐度：★★★，推荐应用方向：自然语音理解和分析挖掘。

介绍和下载地址：http://www.openslr.org/12/

（5）VoxForge

该数据集是带口音的语音清洁数据集，对测试模型在不同重音或语调下的鲁棒性非常有用。

推荐度：★★，推荐应用方向：语音识别。

介绍和下载地址：http://www.voxforge.org/

（6）TIMIT

这是一份英文语音识别数据集，包含630个扬声器的宽带录音，八个主要方言的美式英语，每个阅读十个语音丰富的句子。TIMIT语料库包括时间对齐的正字法，语音和单词转录以及每个话语的16位，16kHz语音波形文件。

推荐度：★★，推荐应用方向：语音识别。

介绍和下载地址：https://catalog.ldc.upenn.edu/LDC93S1

（7）CHIME

这份语音一份包含环境噪音的用于语音识别挑战赛（CHiME Speech Separation and Recognition Challenge）的数据集。该数据集包含了训练集、开发机、测试集三部分，每份里面包括了多个扬声器在不同噪音环境下的数据。

推荐度：★★★，推荐应用方向：语音识别。

介绍和下载地址：http://spandh.dcs.shef.ac.uk/chime_challenge/index.html

（8）TED-LIUM

TED Talk的音频数据集，包含1495个录音和音频会议、159848条发音词典和部分WMT12公开的语料库。

推荐度：★★★，推荐应用方向：语音识别。

介绍和下载地址：http://www-lium.univ-lemans.fr/en/content/ted-lium-corpus

除了上述公开数据集外，不要忘记大多数机器学习和数据挖掘工具本身也附带有datasets资源，甚至sklearn还提供了生成模拟数据的功能，请见sklearn中的datasets方法。

附录B　Python数据工具箱

Python数据工具箱涵盖从数据源到数据可视化的完整流程中涉及的常用库、函数和外部工具。其中既有Python内置函数和标准库，又有第三方库和工具。这些库可用于文件读写、网络抓取和解析、数据连接、数清洗转换、数据计算和统计分析、图像和视频处理、音频处理、数据挖掘/机器学习/深度学习、数据可视化、交互学习和集成开发以及其他Python协同数据工作工具。

为了区分不同对象的来源和类型，本节将在描述中通过以下方法进行标识：

·[Python内置函数]：Python自带的内置函数。函数无需导入，直接使用。例如要计算-3.2的绝对值，直接使用abs函数，方法是abs（-3.2）。

·[Python标准库]：Python自带的标准库。Python标准库无需安装，只需要先通过import方法导入便可使用其中的方法。例如导入string模块，然后使用其中的find方法：

```
import string
string.find('abcde','b')
```

·[第三方库]：Python的第三方库。这些库需要先进行安装（部分可能需要配置），具体方法参考1.2.3节，然后通过import方法导入便可使用其中的方法。

·[外部工具]：非Python写成的库或包，用于Python数据工作的相关工具。

1.文件读写

文件的读写包括常见的txt、Excel、xml、二进制文件以及其他格式的数据文本，主要用于本地数据的读写。

库/函数	描　述	推荐度
open(name[, mode[, buffering]])	[Python 内置函数]Python 默认的文件读写方法	★★★
numpy.loadtxt、numpy.load 和 numpy.fromfile	[第三方库]Numpy 自带的读写函数，包括 loadtxt、load 和 fromfile，用于文本、二进制文件读写	★★★
pandas.read_*	[第三方库]Pandas 自带的 read 文件方法，例如 ead_csv、read_fwf、read_table 等，用于文本、Excel、二进制文件、HDF5、表格、SAS 文件、SQL 数据库、Stata 文件等的读写	★★★
xlrd	[第三方库] 用于 Excel 文件读取	★★
xlwt	[第三方库] 用于 Excel 文件写入	★★
pyexcel-xl	[第三方库] 用于 Excel 文件读写	★★
xluntils	[第三方库] 用于 Excel 文件读写	★★
pyExcelerator	[第三方库] 用于 Excel 文件读写	★
openpyxl	[第三方库] 用于 Excel 文件读写	★
lxml	[第三方库]xml 和 HTML 读取和解析	★★★
xml	[Python 标准库]xml 对象解析和格式化处理	★★★
libxml2	[第三方库]xml 对象解析和格式化处理	★
xpath	[第三方库]xml 对象解析和格式化处理	★★
win32com	[第三方库] 有关 Windows 系统操作、Office（Word、Excel 等）文件读写等的综合应用库	★

2.网络抓取和解析

网络抓取和解析用于从互联网中抓取信息，并对HTML对象进行处理，有关xml对象的解析和处理的库在“1.文件读写”中找到。

库/函数	描　述	推荐度
requests	[第三方库]网络请求库，提供多种网络请求方法并可定义复杂的发送信息	★★★
urllib	[Python 标准库]Python 自带的库，简单的读取特定 URL 并获得返回的信息	★★
urllib2	[Python 标准库]Python 自带的库，读取特定 URL 并获得返回的信息，相对于 urllib 可处理更多 HTTP 信息，例如 cookie、身份验证、重定向等	★★
urlparse	[Python 标准库]Python 自带的 URL 解析库，可自动解析 URL 不同的域、参数、路径等	★★★
HTMLParser	[Python 标准库]Python 自带的 HTML 解析模块，能够很容易的实现 HTML 文件的分析	★★★
Scapy	[第三方库]分布式爬虫框架，可用于模拟用户发送、侦听和解析并伪装网络报文，常用于大型网络数据爬取	★★★
Beautiful Soup	[第三方库]Beautiful Soup 是网页数据解析和格式化处理工具，通常配合 Python 的 urllib、urllib2 等库一起使用	★★★

3.数据库连接

数据库连接可用于连接众多数据库以及访问通用数据库接口，可用于数据库维护、管理和增、删、改、查等日常操作。

库 / 函数	描　　述	推 荐 度
mysql-connector-python	[第三方库]MySQL 官方驱动连接程序	★★★
MySQL-python	[第三方库]MySQL 连接库	★★
cx_Oracle	[第三方库]Oracle 连接库	★★★
psycopg2	[第三方库]Python 编程语言中非常受欢迎的 PostgreSQL 适配器	★★★
redis	[Python 标准库]Redis 连接库	★★★
pymongo	[第三方库]MongoDB 官方驱动连接程序	★★★
HappyBase	[第三方库]HBase 连接库	★★★
py2neo	[第三方库]Neo4j 连接库	★★★
cassandra-driver	[第三方库] Cassandra（1.2+）和 DataStax Enterprise（3.1+）连接库	★★★
sqlite3	[Python 标准库] Python 自带的模块，用于操作 SQLite 数据库	★★★
pysqlite2	[第三方库] SQLite 3.x 连接库	★★
bsddb3	[第三方库]Berkeley DB 连接库	★★
bsddb	[Python 标准库] Python 自带的模块，提供了一个到 Berkeley DB 库的接口	★★
dbhash	[Python 标准库] Python 自带的模块，dbhash 模块提供了使用 BSD 数据库打开数据库的功能。该模块镜像了提供对 DBM 样式数据库访问的其他 Python 数据库模块的接口。 bsddb 模块需要使用 dbhash	★★
adodb	[第三方库] ADOdb 是一个数据库抽象库，支持常见的数据和数据库接口并可自行进行数据库扩展，该库可以对不同数据库中的语法进行解析和差异化处理，具有很高的通用性	★★★
SQLObject	[第三方库] SQLObject 是一种流行的对象关系管理器，用于向数据库提供对象接口，其中表为类、行为实例、列为属性	★★
SQLAlchemy	[第三方库] SQLAlchemy 是 Python SQL 工具包和对象关系映射器，为应用程序开发人员提供了 SQL 的全部功能和灵活性控制	★★
ctypes	[第三方库] ctypes 是 Python 的一个外部库，提供和 C 语言兼容的数据类型，可以很方便地调用 C DLL 中的函数	★★★
pyodbc	[第三方库] Python 通过 ODBC 访问数据库的接口库	★★★
Jython	[第三方库] Python 通过 JDBC 访问数据库的接口库	★★★

4.数据清洗转换

数据清洗转换主用于数据正式应用之前的预处理工作。

库 / 函数	描　　述	推荐度
frozenset([iterable])	[Python 内置函数] 返回一个新的 frozenset 对象，可选择从 iterable 取得的元素	★★★
int(x)	[Python 内置函数] 返回 x 的整数部分	★★★
isinstance(object, classinfo)	[Python 内置函数] 返回 object 是否是指定的 classinfo 实例信息	★★★
len(s)	[Python 内置函数] 返回对象的长度或项目数量	★★★
long(x)	[Python 内置函数] 返回由字符串或数字 x 构造的长整型对象	★★★
max(iterable[, key])	[Python 内置函数] 返回一个可迭代或最大的两个或多个参数中的最大项	★★★
min(iterable[, key])	[Python 内置函数] 返回一个可迭代或最大的两个或多个参数中的最小项	★★★
range(start, stop[, step])	[Python 内置函数] 用于与 for 循环一起创建循环列表，通过指定 start（开始）、stop（结束）和 step（步长）控制迭代次数并获取循环值	★★★
raw_input(prompt)	[Python 内置函数] 捕获用户输入并作为字符串返回（不推荐使用 input 作为用户输入的捕获函数）	★★★
round(number[, ndigits])	[Python 内置函数] 返回 number 小数点后 ndigits 位的四舍五入的浮点数	★★★
set([iterable])	[Python 内置函数] 返回一个新的集合对象，可选择从 iterable 获取的元素	★★★
slice(start, stop[, step])	[Python 内置函数] 返回表示由范围（start、stop、step）指定的索引集的切片对象	★★
sorted(iterable[, cmp[, key[, reverse]]])	[Python 内置函数] 从 iterable 的项中返回一个新的排序列表	★★★
xrange(start, stop[, step])	[Python 内置函数] 此函数与 range() 非常相似，但返回一个 xrange 对象而不是列表	★★★
string	[Python 标准库] 字符串处理库，可实现字符串查找、分割、组合、替换、去重、大小写转换及其他格式化处理	★★★
re	[Python 标准库] 正则表达式模块，在文本和字符串处理中经常使用	★★★
random	[Python 标准库] 该模块为各种分布实现伪随机数生成器，支持数据均匀分布、正态（高斯）分布、对数正态分布、负指数分布、伽马和 β 分布等	★★★
os	[Python 标准库] 用于新建、删除、权限修改、切换路径等目录操作，以及调用执行系统命令	★★★
os.path	[Python 标准库] 针对目录的遍历、组合、分割、判断等操作，常用于数据文件的判断、查找、合并	★★★
prettytable	[Python 标准库] 格式化表格输出模块	★★
json	[Python 标准库]Python 对象与 json 对象的转换	★★★
base64	[Python 标准库] 将任意二进制字符串编码和解码为文本字符串的 Base16，Base32 和 Base64	★★★

5.数据计算和统计分析

数据计算和统计分析主要用于数据探查、计算和初步数据分析等工作。

库 / 函数	描　述	推荐度
numpy	[第三方库]NumPy 是 Python 科学计算的基础工具包，很多 Python 数据计算工作库都依赖它	★★★
scipy	[第三方库]Scipy 是一组专门解决科学和工程计算不同场景的主题工具包	★★★
pandas	[第三方库]Pandas 是一个用于 Python 数据分析的库，它的主要作用是进行数据分析。Pandas 提供用于进行结构化数据分析的二维的表格型数据结构 DataFrame，类似于 R 中的数据框，能提供类似于数据库中的切片、切块、聚合、选择子集等精细化操作，为数据分析提供了便捷	★★★
statsmodels	[第三方库]Statsmodels 是 Python 的统计建模和计量经济学工具包，包括一些描述性统计、统计模型估计和统计测试，集成了多种线性回归模型、广义线性回归模型、离散数据分布模型、时间序列分析模型、非参数估计、生存分析、主成分分析、核密度估计以及广泛的统计测试和绘图等功能	★★★
abs(x)	[Python 内置函数] 返回 x 的绝对值	★★★
cmp(x, y)	[Python 内置函数] 比较两个对象 x 和 y，并根据结果返回一个整数。如果 x<y，则返回值为负数，如果 x == y，则为零，如果 x>y，则返回值为正	★★
float(x)	[Python 内置函数] 返回从数字或字符串 x 构造的浮点数	★★★
pow(x, y[, z])	[Python 内置函数] 返回 x 的 y 次幂。如果 z 存在，则返回 x 的 y 次幂，模 z	★★★
sum(iterable[, start])	[Python 内置函数] 从左到右依次迭代，返回总和	★★★
math	[Python 标准库] 数学函数库，包括正弦、余弦、正切、余切、弧度转换、对数运算、圆周率、绝对值、取整等数学计算方法	★★★
cmath	[Python 标准库] 与 math 基本一致，区别是 cmath 运算的是复数	★★
decimal	[Python 标准库]10 进制浮点运算	★★
fractions	[Python 标准库] 分数模块提供对有理数算术的支持	★★

6.自然语言处理和文本挖掘

自然语言处理和文本挖掘库主要用于以自然语言文本为对象的数据处理和建模。

库/函数	描　述	推荐度
nltk	[第三方库]NLTK是一个Python自然语言处理工具，它用于对自然语言进行分类、解析和语义理解。目前已经有超过50种语料库和词汇资源	★★★
pattern	[第三方库]Pattern是一个网络数据挖掘Python工具包，提供了用于网络挖掘（如网络服务、网络爬虫等）、自然语言处理（如词性标注、情感分析等）、机器学习（如向量空间模型、分类模型等）、图形化的网络分析模型	★★★
gensim	[第三方库]Gensim是一个专业的主题模型（发掘文字中隐含主题的一种统计建模方法）Python工具包，用来提供可扩展统计语义、分析纯文本语义结构以及检索语义上相似的文档	★★★

（续）

库/函数	描　述	推荐度
结巴分词	[第三方库]结巴分词是国内流行的Python文本处理工具包，分词模式分为三种模式：精确模式、全模式和搜索引擎模式，支持繁体分词、自定义词典等，是非常好的Python中文分词解决方案，可以实现分词、词典管理、关键字抽取、词性标注等	★★★
SnowNLP	[第三方库]SnowNLP是一个Python写的类库，可以方便的处理中文文本内容。该库是受到了TextBlob的启发而针对中文处理写的类库，和TextBlob不同的是这里没有用NLTK，所有的算法都是自己实现的，并且自带了一些训练好的字典	★★
smallseg	[第三方库] Smallseg是一个开源的、基于DFA的轻量级的中文分词工具包。可自定义词典、切割后返回登录词列表和未登录词列表、有一定的新词识别能力	★★
spaCy	[第三方库] spaCy是一个Python自然语言处理工具包，它结合Python和Cython使得自然语言处理能力达到了工业强度	★★★
TextBlob	[第三方库]TextBlob是一个处理文本数据的Python库，可用来做词性标注、情感分析、文本翻译、名词短语抽取、文本分类等	★★
PyNLPI	[第三方库] PyNLPI是一个适合各种自然语言处理任务的集合库，可用于中文文本分词、关键字分析等，尤其重要的是其支持中英文映射，支持UTF-8和GBK编码的字符串等	★★★

7.图像和视频处理

图像处理和视频处理主要适用于基于图像的操作、处理、分析和挖掘，如人脸识别、图像识别、目标跟踪、图像理解等。

库 / 函数	描　述	推荐度
PIL	[第三方库]PIL 是一个常用的图像读取、处理和分析的库，提供了多种数据处理、变换的操作方法和属性	★★
OpenCV	[第三方库]OpenCV 是一个强大的图像和视频工作库。它提供了多种程序接口，支持跨平台 (包括移动端) 应用。OpenCV 的设计效率很高，它以优化的 C / C ++ 编写，库可以利用多核处理。除了对图像进行基本处理外，还支持图像数据建模，并预制了多种图像识别引擎，如人脸识别	★★★
scikit-image	[第三方库] scikit-image (也称 skimage) 是一个图像处理库，支持颜色模式转换、滤镜、绘图、图像处理、特征检测等多种功能	★★
imageop	[Python 标准库] Python 自带的函数，对图像基本操作，包括裁剪、缩放、模式转换	★
colorsys	[Python 标准库] Python 自带的函数，实现不同图像色彩模式的转换	★
imghdr	[Python 标准库] Python 自带的函数，返回图像文件的类型	★

8.音频处理

音频处理主要适用于基于声音的处理、分析和建模，主要应用于语音识别、语音合成、语义理解等。

库 / 函数	描　述	推荐度
TimeSide	[第三方库] TimeSide 是一个能够进行音频分析、成像、转码、流媒体和标签处理的 Python 框架，可以对任何音频或视频内容非常大的数据集进行复杂的处理	★★★
audiolazy	[第三方库]audiolazy 是一个用于实时声音数据流处理的库，支持实时数据应用处理、无限数据序列表示、数据流表示等	★★
pydub	[第三方库] pydub 支持多种格式声音文件，可进行多种信号处理（例如压缩、均衡、归一化）、信号生成（例如正弦、方波、锯齿等）、音效注册、静音处理等	★★★
audioop	[Python 标准库] Python 自带的函数，可实现对声音片段的一些常用操作	★★
tinytag	[第三方库] tinytag 用于读取多种声音文件的元数据，涵盖 MP3、OGG、OPUS、MP4、M4A、FLAC、WMA、Wave 等格式	★★
aifc	[Python 标准库] Python 自带的函数，读写 AIFF 和 AIFC 文件	★
sunau	[Python 标准库] Python 自带的函数，读写 Sun AU 文件	★
wave	[Python 标准库] Python 自带的函数，读写 WAV 文件	★★
chunk	[Python 标准库] Python 自带的函数，读取 EA IFF 85 块格式的文件	★
sndhdr	[Python 标准库] Python 自带的函数，返回声音文件的类型	★
ossaudiodev	[Python 标准库] 该模块支持访问 OSS（开放声音系统）音频接口	★★★

9.数据挖掘/机器学习/深度学习

数据挖掘、机器学习和深度学习等是Python进行数据建模和挖掘学习的核心模块。

库 / 函数	描　述	推荐度
Scikit-Learn	[第三方库]scikit-learn（也称 SKlearn）是一个基于 Python 的机器学习综合库，内置监督式学习和非监督式学习机器学习方法，包括各种回归、聚类、分类、流式学习、异常检测、神经网络、集成方法等主流算法类别，同时支持预置数据集、数据预处理、模型选择和评估等方法，是一个非常完整、流行的机器学习工具库	★★★
TensorFlow	[第三方库]TensorFlow 是谷歌的第二代机器学习系统，内建深度学习的扩展支持，任何能够用计算流图形来表达的计算，都可以使用 TensorFlow	★★★
NuPIC	[第三方库] NuPIC 是一个以 HTM（分层时间记忆）学习算法为工具的机器智能平台。NuPIC 适合于各种各样的问题，尤其适用于检测异常和预测应用	★★★
Orange	[第三方库] Orange 通过图形化操作界面，提供交互式数据分析功能，尤其适用于分类、聚类、回归、特征选择和交叉验证工作	★★★
theano	[第三方库] Theano 是非常成熟的深度学习库。它与 Numpy 紧密集成，支持 GPU 计算、单元测试和自我验证	★★★
mlxtend	[第三方库] Mlxtend 是一个 Python 工具和扩展包，里面包含了多种数据计算、统计分析以及数据挖掘功能，包括数据预处理、关联、分类、聚类、回归、效果评估、特征工程等	★★
keras	[第三方库] Keras 是一个用 Python 编写的高级神经网络 API，能够运行在 TensorFlow 或者 Theano 之上，它的开发重点是实现快速实验	★★

（续）

库 / 函数	描　述	推荐度
neurolab	[第三方库] Neurolab 是具有灵活网络配置和 Python 学习算法的基本神经网络算法库。它包含通过递归神经网络（RNN）实现的不同变体，该库是同类 RNN API 中最好的选择之一	★★
PyLearn2	[第三方库] PyLearn2 是基于 Theano 的深度学习库，它旨在提供极大的灵活性，并使研究人员可以进行自由可控制，参数和属性的灵活、开放配置是亮点	★★★
OverFeat	[第三方库] OverFeat 是一个深度学习库，主要用于图片分类、定位物体检测	★★
Pyevolve	[第三方库] Pyevolve 是一个完整的遗传算法框架，也支持遗传编程	★★
Caffe	[第三方库] Cafffe 是一个深度学习框架，主要用于计算机视觉，它对图像识别的分类具有很好的应用效果	★★★
PyTorch	[第三方库] Facebook 开源的基于 Python 的工具包，专门针对 GPU 加速的深度神经网络（DNN）编程	★★★

10.数据可视化

数据可视化主要用于做数据结果展示、数据模型验证、图形交互和探查等方面。

库 / 函数	描　述	推荐度
Matplotlib	[第三方库] Matplotlib 是 Python 的 2D 绘图库，它以各种硬拷贝格式和跨平台的交互式环境生成出版质量级别的图形，开发者可以仅需要几行代码，便可以生成多种高质量图形	★★★
seaborn	[第三方库] Seaborn 是在 Matplotlib 的基础上进行了更高级的 API 封装，它可以作为 Matplotlib 的补充	★★
bokeh	[第三方库] Bokeh 是一种交互式可视化库，可以在 WEB 浏览器中实现美观的视觉效果	★★★
Plotly	[第三方库] Plotly 提供的图形库可以进行在线 WEB 交互，并提供具有出版品质的图形，支持线图、散点图、区域图、条形图、误差条、框图、直方图、热图、子图、多轴、极坐标图、气泡图、玫瑰图、热力图、漏斗图等众多图形	★★★
VisPy	[第三方库] VisPy 是用于交互式科学可视化的 Python 库，旨在实现快速，可扩展和易于使用	★★
PyQtGraph	[第三方库] PyQtGraph 是一个建立在 PyQt4 / PySide 和 numpy 之上的纯 Python 图形和 GUI 库，主要用于数学 / 科学 / 工程应用	★★
ggplot	[第三方库] ggplot 是用 Python 实现的图形输出库，类似于 R 中的图形展示版本	★★★
pyecharts	[第三方库] 做过前端或产品的读者一定知道，Echarts 是百度开源的一个数据可视化 JS 库，pyecharts 则是一个用于生成 Echarts 图表的类库	★★★

11.交互学习和集成开发

交互学习和集成开发主要用来做Python开发、调试和集成之用，包括Python集成开发环境和IDE。

库 / 函数	描　述	推荐度
IPython	[第三方库]IPython 是一个基于 Python 的交互式 shell，比默认的 Python shell 好用得多，支持变量自动补全、自动缩进、交互式帮助、魔法命令、系统命令等，内置了许多很有用的功能和函数	★★★
Elpy	[第三方库] Elpy 是 Emacs 用于 Python 的开发环境，它结合并配置了许多其他软件包，它们都是用 Emacs Lisp 和 Python 编写的	★★
PTVS	[第三方库] Visual Studio 的 Python 工具	★★
PyCharm	[外部工具]PyCharm 带有一整套可以帮助用户在使用 Python 语言开发时提高其效率的工具，比如调试、语法高亮、项目管理、代码跳转、智能提示、自动完成、单元测试、版本控制并可集成 IPython、系统终端命令行等，在 PyCharm 里几乎就可以实现所有有关 Python 工作的全部过程	★★★
LiClipse	[外部工具] LiClipse 是基于 Eclipse 的免费多语言 IDE，通过其中的 PyDev 可支持 Python 开发应用	★★
Spyder	[外部工具] Spyder 是一个开源的 Python IDE，由 IPython 和众多流行的 Python 库的支持，是一个具备高级编辑、交互式测试、调试以及数字计算环境的交互式开发环境	★★

12.其他Python协同数据工作工具

其他Python协同数据工作工具指除了上述主题以外，其他在数据工作中常用的工具或库。

库 / 函数	描　述	推荐度
tesseract-ocr	[外部工具] 这是一个 Google 支持的开源 OCR 图文识别项目，支持超过 200 种语言（包括中文），并支持自定义训练字符集，支持跨 Windows、Linux、Mac OSX 多平台使用	★★★
RPython	[第三方库]R 集成库	★★★
matpython	[第三方库]MATLAB 集成库	★★★
Lunatic Python	[第三方库]Lua 集成库	★★
PyCall.jl	[第三方库]Julia 集成库	★★
PySpark	[第三方库] Spark 提供的 Python API	★★★
dumbo	[第三方库] 这个模块可以让 Pythoner 轻松的编写和运行 Hadoop 程序，程序版本比较早，可以作为参考	★★
dpark	[第三方库]Python 对 Spark 的克隆版本，类 MapReduce 框架	★★
streamparse	[第三方库] Streamparse 允许通过 Storm 对实时数据流运行 Python 代码	★★★

Table of Contents