第5章

数据采集和预处理

在计算机广泛应用的今天，数据采集的重要性是十分显著的。它是计算机与外部物理世界连接的桥梁，各种类型信号采集的难易程度差别非常大。一般地，在进行实际数据采集时，所获得的数据极易受到噪声、缺失值和不一致数据的侵扰。数据的质量决定了数据挖掘的效果，因此在数据挖掘之前要对数据进行预处理以提高数据质量，从而提高数据挖掘的效果。

5.1 概述

数据采集是所有数据分析系统必不可少的，随着大数据越来越被重视，数据采集的挑战也变得尤为突出。同时，针对数据的种类多、数据量大、变化快等特点，利用各种预处理方法保证数据采集的可靠性、避免重复数据等，以保证数据的质量。

5.1.1 数据采集概述

大数据的来源非常广泛，例如信息管理系统、网络信息系统、物联网系统、科学实验系统等，其数据类型包括结构化数据、半结构化数据和非结构化数据。在大数据平台下，由于数据源具有更复杂的多样性，数据采集的形式也变得更加复杂、多样，当然业务处理也可能变得迥然不同。为了提升业务处理的性能，同时又希望保留历史数据以备进行数据挖掘与分析，对数据的采集要着重考虑以下几个方面：

(1) 根据业务处理的需求，往往从关系数据库中采集数据，这种情况的可伸缩性较小。一般需要满足查询与其他数据操作的实时性，这就需要定期将超过时间期限的历史数据清除。

(2) 有时需要将实时采集的源数据先写入某数据流存储平台。例如 Apache 软件基金会开发的开源流处理平台 Kafka，考虑到数据流处理业务的需求，数据采集会成为 Kafka 的消费者，就像一个水坝将上游源源不断的数据拦截住，然后根据具体业务做相应的处理(如去重、去噪、中间计算等)，之后再写入对应的数据存储中。

(3) 如果数据源为视频文件，需要提取特征数据。针对视频文件的大数据处理，加载图片后，根据某种识别算法识别并提取图片的特征信息，然后将其转换为具体业务需要的数据模型。在视频处理过程中，数据提取的耗时相对较长，也需要较多的内存资源。如果处理不当，可能会成为整个数据阶段的瓶颈。

在大数据采集阶段，一个棘手的问题是增量同步，尤其针对那些可变(即可删除、可修改)

的数据源。在人们无法掌控数据源的情况下通常会有 3 种选择：一是放弃同步，采用直连形式；二是放弃增量同步，选用全量同步；三是编写定期 Job，扫描数据源，以获得 delta 数据，然后针对 delta 数据进行增量同步。

为了更高效地完成数据采集，通常需要将整个流程切分成多个阶段，在细分的阶段中可以采用并行执行的方式。在这个过程中可能涉及 Job 的创建、提交与分发，采集流程的规划，数据格式的转换等。另外，在保证数据采集的高性能之外还要考虑数据丢失的容错。

5.1.2 数据采集的方法

传统的数据采集来源单一，且存储、管理和分析的数据量也相对较小，大多采用关系数据库和并行数据仓库即可处理。目前，在大数据采集方面增加了以下几种新的方法。

1. 网络数据采集方法

网络数据采集是指通过网络爬虫或网站公开 API 等方式从网站上获取数据信息。该方法可以将非结构化数据从网页中抽取出来，将其存储为统一的本地数据文件，并以结构化的方式存储。它支持图片、音频、视频等文件或附件的采集，附件与正文可以自动关联。除了网络中包含的内容之外，对于网络流量的采集可以使用 DPI 或 DFI 等流量解析技术进行处理。

2. 系统日志采集方法

很多互联网企业都有自己的海量数据采集工具，多用于系统日志采集，例如 Hadoop 的 Chukwa、Cloudera 的 Flume、Facebook 的 Scribe 等，这些工具均采用分布式架构，能满足每秒数百兆字节的日志数据采集和传输需求。

3. 其他数据采集方法

对于企业生产经营数据或学科研究数据等保密性要求较高的数据，可以通过与企业或研究机构合作，使用特定系统接口等相关方式采集数据。

5.1.3 数据预处理概述

数据预处理(Data Preprocessing)是指在数据挖掘之前对原始数据进行的一些处理。现实世界中的数据几乎都是“脏”数据，所采集的数据极易受到不一致数据、噪声、缺失值的侵扰。

(1) 不一致数据。原始数据是从各种实际应用系统中采集的，由于各应用系统的数据缺乏统一的标准和定义，数据结构也有较大的差异，所以各系统间的数据存在严重的不一致性。

(2) 噪声数据。在采集数据时很难得到精确的数据，如数据采集设备出现故障、数据传输过程中出现错误或存储介质出现损坏等，这些情况都会导致噪声数据的出现。

(3) 缺失值。由于系统在设计时可能存在的缺陷或者系统在使用过程中人为因素的影响，在数据记录中可能出现一些属性值丢失或不确定的情况，从而造成数据的不完整，例如数据采集传感器出现故障导致一部分数据无法采集等。

对于现实世界中不完整、不一致的“脏”数据，无法直接进行数据挖掘，或挖掘结果不尽如人意。为了提高数据挖掘的质量产生了数据预处理技术，这些技术在数据挖掘之前使用，大幅提高了数据的质量。那么高质量数据的标准是什么呢？

(1) 准确性。准确性是指数据记录的信息是否存在异常或错误。

(2) 完整性。完整性是指数据信息是否存在缺失的情况。数据缺失可能是整条数据记录的缺失，也可能是数据中某个属性值的缺失。

(3) 一致性。一致性是指数据是否遵循了统一的规范，数据集合是否保持了统一的格式。

(4) 时效性。时效性是指某些数据是否能及时更新，更新时间越短，时效性越高。

（5）可信性。可信性是指用户信赖数据的数量。用户信赖的数据越多，则可信性越强。

（6）可解释性。可解释性是指数据自身是否易于人们理解。数据自身越容易被人理解，可解释性就越强。

针对数据中存在的问题和数据质量要求，数据预处理过程主要包括数据清洗、数据集成、数据归约和数据变换等方法。

5.2　数据清洗

数据清洗（Data Cleaning）是指发现并纠正数据文件中可识别错误的最后一道程序。

5.2.1　缺失值清洗

在许多业务数据分析场景中数据不一定十分完整，总是存在部分缺失值，因此在数据清洗阶段对缺失值进行处理就显得尤为重要。

1. 缺失值处理方法

对于记录中缺失值的处理，最常用的方法有删除法、替换法和插补法。

（1）删除法。删除法是指将缺失值所在的观测记录删除（前提是缺失记录的比例非常低，例如5%以内），或者删除缺失值所对应的属性（前提是该属性中包含的缺失值比例非常高，例如70%左右）。

（2）替换法。替换法是指直接利用缺失变量的均值、中位数或众数替换该变量中的缺失值，其优点是缺失值的处理速度快，弊端是易产生有偏估计，导致缺失值替换的准确性下降。例如，假定电商信息表中顾客收入的数据分布是对称的，并且年平均收入为56000元，可以使用该值替换“收入”属性中的缺失值。

（3）插补法。插补法是利用有监督的机器学习方法（例如回归模型、树模型、网络模型等）对缺失值进行预测，其优势在于预测的准确性高，缺点是需要进行大量的计算，导致缺失值的处理速度大幅降低。例如，利用数据集中其他顾客的属性可以构造一棵决策树来预测缺失值。

需要注意的是，在某些情况下缺失值并不意味着数据有错误。例如申请信用卡时，要求申请人提供工作单位，但是刚毕业的大学生没有工作单位就自然不填写该属性了。

2. Pandas对缺失值的处理

Pandas对象的所有描述性统计默认都不包括缺失数据。对于数值数据，Pandas使用浮点值NaN表示缺失数据。

1）对缺失值的检测与统计

使用isnull()函数（或notnull()函数）可以直接判断列中的哪个数据为NaN，缺失值时为True（或False），非缺失值时为False（或True）。info()和sum()分别用于查看非缺失值的信息和统计各列缺失值的数量。

例5.1　缺失值检测和统计示例。

程序代码如下：

```
import numpy as np
import pandas as pd
df = pd.DataFrame([['S1','许文秀','女',20,'团员','计算机系','湖北'],
                   ['S2','刘世元','男',21,np.NaN,'电信系','贵州'],
                   ['S3','刘德峰','男',22,np.NaN,'统计系',np.NaN],
                   ['S4','于金凤','女',np.NaN,np.NaN,'计算机系',np.NaN],
                   ['S5','周新娥','女',23,'团员','电信系',np.NaN],
                   ['S6','王晓晴','女',22,np.NaN,np.NaN,np.NaN]],
```

```
                columns = ['学号','姓名','性别','年龄','政治面貌','系部','籍贯'])
print(df)
print(df.info())                    # 打印出各列数据的非缺失值信息
print(df.isnull())                  # 打印出缺失值信息,缺失值时为 True,非缺失值时为 False
print(df.isnull().sum())            # 打印出各列中缺失值的数量
```

2）删除缺失值

根据一定的规则将含有缺失值的行或列直接进行删除。dropna()为 Pandas 库中 DataFrame 的一个方法，用于删除缺失值。其常用形式如下：

```
dropna(axis = 0,how = 'any',thresh = None,subset = None,,inplace = False)
```

参数说明：

（1）axis 默认为 0，axis＝0，当某行出现缺失值时将该行删除；axis＝1，当某列出现缺失值时将该列删除。

（2）how 默认为'any'，用于确定缺失值的个数，表示只要某行有缺失值就将该行删除，how＝'all'表明某行全部为缺失值才将其删除。

（3）thresh 为阈值设定，表示当某行中非缺失值的数量少于给定的阈值时就将该行删除。

（4）subset 表示部分列中有缺失值时删除相应的行，例如 subset＝['a','d']，即删除列 a、d 中含有缺失值的行。

（5）inplace 默认为 False，表示将筛选后的数据存为副本，为 True 表示直接在原数据上更改。

例 5.2 删除缺失值示例。

程序代码如下：

```
import numpy as np
import pandas as pd
df = pd.DataFrame([['S1','许文秀','女',20,'团员','计算机系','湖北'],
                   ['S2','刘世元','男',21,np.NaN,'电信系','贵州'],
                   ['S3','刘德峰','男',22,np.NaN,'统计系',np.NaN],
                   ['S4','于金凤','女',np.NaN,np.NaN,'计算机系',np.NaN],
                   ['S5','周新娥','女',23,'团员','电信系',np.NaN],
                   ['S6','王晓晴','女',22,np.NaN,np.NaN,np.NaN]],
                  columns = ['学号','姓名','性别','年龄','政治面貌','系部','籍贯'])
print(df.dropna())                    # 删除含有缺失值的行
print(df.dropna(axis = 1))            # 删除含有缺失值的列
print(df.dropna(thresh = 5))          # 保留至少具有 5 个非 NaN 值的行
print(df.dropna(thresh = 3,axis = 1)) # 保留至少具有 3 个非 NaN 值的列
```

3）填充缺失值

直接删除缺失值的样本并不是一个很好的方法，可以用一个特定的值替换缺失值。当缺失值所在的属性为数值型时，通常利用均值、中位数和众数等描述其集中趋势的统计量来填充；当缺失值所在的属性为类别型时，可以选择用众数来填充。在 Pandas 库中提供了缺失值的替换方法 fillna()，其常用形式如下：

```
fillna(value = None,method = None,axis = None,inplace = False,limit = None)
```

参数说明：

（1）value 默认为 False，表示填充缺失值的标量值或字典对象。

（2）method 默认为 None，表示插值方式，'ffill'为向前填充或向下填充，'bfill'为向后填充或向上填充。

(3) axis 默认为 0,表示待填充的轴,axis=0 为 X 轴,axis=1 为 Y 轴。

(4) inplace 表示修改调用者对象而不产生副本。

(5) limit 表示(对于向前和向后填充)可以连续填充的最大数量。

常见的填充方法如下:

(1) 填充固定值。选取某个固定值/默认值填充缺失值。

(2) 填充均值。对每一列的缺失值,填充当前列的均值。

(3) 填充中位数。对每一列的缺失值,填充当前列的中位数。

(4) 填充众数。对每一列的缺失值,填充当前列的众数。如果存在某列缺失值过多、众数为 NaN 的情况,则填充每列删除掉 NaN 值后的众数。

(5) 填充上、下样本的数据。对每一数据样本的缺失值,填充其上面一个或下面一个样本的数据值。

(6) 填充插值得到的数据。用插值法拟合出缺失的数据,然后进行填充。interpolate()函数默认采用线性插值,即假设函数是直线形式,缺失值用前一个值和后一个值的平均数填充。

(7) 填充 KNN 数据。填充近邻的数据,先利用 KNN 计算邻近的 k 个数据,然后填充它们的均值。

(8) 填充模型预测的值。把缺失值作为新的 Label,建立模型得到预测值,然后进行填充。

例 5.3　填充缺失值示例。

程序代码如下:

```
import numpy as np
import pandas as pd
df = pd.DataFrame([['S1','许文秀','女',20,'团员','计算机系','湖北',387],
        ['S2','刘世元','男',21,np.NaN,'电信系','贵州',376],
        ['S3','刘德峰','男',22,np.NaN,'统计系',np.NaN,380],
        ['S4','于金凤','女',np.NaN,np.NaN,'计算机系',np.NaN,np.NaN],
        ['S5','周新娥','女',23,'团员','电信系',np.NaN,367],
        ['S6','王晓晴','女',22,np.NaN,np.NaN,np.NaN,np.NaN]],
        columns = ['学号','姓名','性别','年龄','政治面貌','系部','籍贯','总分'])
print(df.fillna( - 1))                          #填充缺失值为' - 1'
print(df.fillna(method = 'ffill'))              #向下填充缺失值
print(df['年龄'].fillna(df['年龄'].mean()))     #年龄列的缺失值用其均值填充
print(df.fillna(df.mode()))                     #利用众数填充缺失值
for n in df:
   df[n] = df[n].interpolate()                  #数值型属性利用线性插值
   df[n].dropna(inplace = True)
print(df)
```

5.2.2　异常值清洗

异常值是指在数据集中存在的不合理的值,这里所说的不合理的值是偏离正常范围的值,不是错误值。异常值的存在会严重干扰数据分析的结果。

1. 异常值检测

一般异常值的检测方法有基于统计的方法、基于聚类的方法,以及一些专门检测异常值的方法等,下面介绍一些简单的检测方法。

1) 简单统计分析

最常用的统计量是最大值和最小值,用来判断变量的取值是否超出合理的范围。例如电商信息表中的客户年龄 age=199,则该变量的取值存在异常。

例 5.4 计算成年人的身高、体重的公式为 Y=(X−100)×0.9,其中 X 为身高(cm),Y 为标准体重(kg)。

程序代码如下:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(100,230,5)                    #假设成年人(18 岁以上)的正常高度在 1～2.3 米
y = (x - 100) * 0.9
plt.rcParams['font.family'] = 'STSong'      #图形中显示汉字
plt.rcParams['font.size'] = 12
plt.title('身高和体重')
plt.plot(x,y,'.')
plt.plot(140,187,'r.')                      #异常值
plt.plot(156,212,'r.')                      #异常值
plt.plot(187,208,'r.')
plt.show()
```

程序运行结果如图 5.1 所示。

2) 散点图方法

通过数据分布的散点图可以检测异常数据。

例 5.5 探究房屋面积和房屋价格的关系示例。

程序代码如下:

```
import matplotlib.pyplot as plt
import numpy as np
x = [225.98,247.07,253.14,254.85,241.58,301.01,20.67,288.64, 163.56,120.06,207.83,342.75,
147.9,53.06,224.72,29.51,21.61,483.21, 245.25,299.25,343.35]   #房屋面积数据
y = [196.63,203.88,210.75,372.74,202.41,347.61,24.9,239.34,140.32,304.15,176.84,488.23,
128.79,49.64,191.74,33.1,30.74,400.02,205.35,330.64,283.45]    #房屋价格数据
plt.figure(figsize = (20,8),dpi = 100)                          #创建画布
plt.scatter(x,y)                                                #绘制散点图
plt.show()                                                      #显示图像
```

程序运行结果如图 5.2 所示。

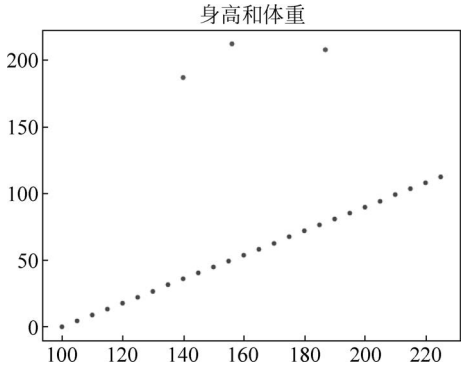


图 5.1 例 5.4 的运行结果

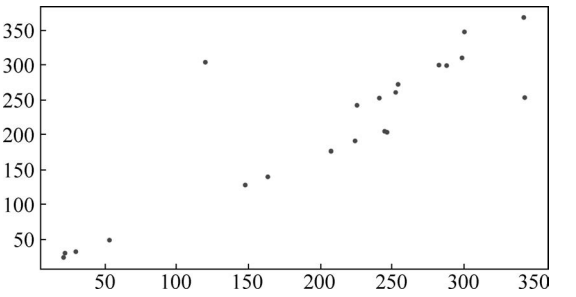


图 5.2 例 5.5 的运行结果

3) 3σ 原则

在正态分布中,σ 代表标准差,μ 代表均值,x=μ 即为图像的对称轴。

3σ 原则认为:数值分布在(μ−σ,μ+σ)中的概率为 0.6827;数值分布在(μ−2σ,μ+2σ)中的概率为 0.9544;数值分布在(μ−3σ,μ+3σ)中的概率为 0.9974。也就是说,Y 的取值几乎全部集中在(μ−3σ,μ+3σ)区间内,超出这个范围的可能性占不到 0.3%,属于极个别的小概率事件,因此超出(μ−3σ,μ+3σ)的值都可以认为是异常值,如图 5.3 所示。

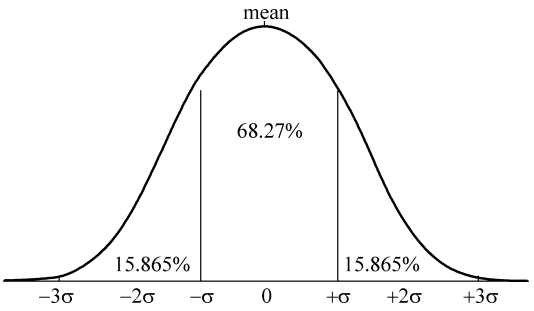


图 5.3　正太分布 3σ 示意图

3σ 原则要求数据服从正态或近似正态分布，且样本数量大于 10。

例 5.6　3σ 原则检测异常值示例。

程序代码如下：

```
import pandas as pd
data = [78,72, - 14,70,68,72,77,78,42,78,74,54,80,82,65,62]      #学生某门课程的成绩
s = pd.Series(data)
dmean = s.mean()
dstd = s.std()
print('检测出异常值: ')
yz1 = dmean - 3 * dstd
yz2 = dmean + 3 * dstd
for i in range(0,len(data)):
    if (data[i]< yz1)or(data[i]> yz2):
        print(data[i],end = ',')
```

结果检测出异常值－14。

4）箱线图

箱线图是通过数据集的四分位数形成的图形化描述，是一种非常简单而且有效的可视化异常值的检测方法。

例 5.7　箱线图检测异常值示例。

程序代码如下：

```
import pandas as pd
import matplotlib.pyplot as plt
data = [78,72,34,70,68,72,77,78,56,78,74,54,80,82,65,62]
s = pd.Series(data)
plt.boxplot(x = s.values,whis = 1.5)
plt.show()
```

程序运行结果如图 5.4 所示。

从图 5.4 可以看出，检测出的异常值为 34。

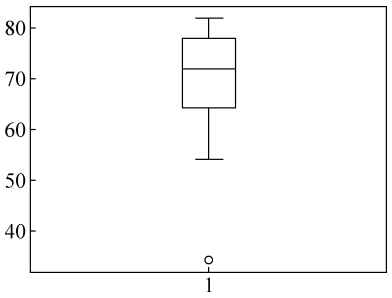


图 5.4　例 5.7 的运行结果

2. 异常值处理

异常值处理是数据预处理中的一个重要步骤，它是保证原始数据可靠性、平均值与标准差计算准确性的前提。

1）直接删除

直接删除是指直接将含有异常值的记录删除。这种方法简单、易行，但缺点也不容忽视，一是在观测值很少的情况下，这种删除操作会造成样本量不足；二是直接删

除可能会对变量的原有分布造成影响，从而导致统计模型不稳定。

2）视为缺失值

视为缺失值是指利用处理缺失值的方法来处理。这种方法能够利用现有变量的信息来填补异常值。注意，将该异常值作为缺失值处理，需要根据该异常值的特点来进行，此时需要考虑该异常值（缺失值）是完全随机缺失、随机缺失还是非随机缺失，针对不同情况进行不同处理。

3）平均值修正

如果数据的样本量很小，也可以用前、后两个观测值的平均值来修正该异常值。这其实是一种比较折中的方法，大部分的方法是针对均值来建模的，用平均值来修正，优点是能克服丢失样本的缺陷，缺点是丢失了样本的“特色”。

4）盖帽法

盖帽法是指将某连续变量均值上下三倍标准差范围外的记录替换为均值上下三倍标准差值，即进行盖帽处理，如图 5.5 所示。

5）分箱平滑法

分箱平滑法是指通过考察“邻居”（周围的值）来平滑存储数据的值。分箱的主要目的是消除异常值，将连续数据离散化，增加粒度。

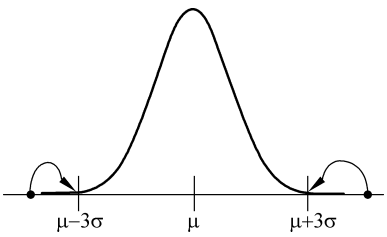


图 5.5　盖帽法示意图

（1）分箱。在分箱前，一定要先对数据排序，再将它们分到等深（或等宽）的箱子中。

① 等深分箱。等深分箱是指按记录数进行分箱，每箱具有相同的记录数，每箱的记录数称为箱子的权重，也称为箱子的深度。

② 等宽分箱。等宽分箱是指在整个属性值的区间上平均分布，即每箱的区间范围设定为一个常量，称为箱子的宽度。

例如，客户收入属性 income 排序后的值 2300，2500，2800，3000，3500，4000，4500，4800，5000，5300，5500，6000，6200，6700，7000，7200 的分箱结果如下（单位为元）：

采用等深分箱，如深度为 4，分箱结果为“箱 1：2300，2500，2800，3000；箱 2：3500，4000，4500，4800；箱 3：5000，5300，5500，6000；箱 4：6200，6700，7000，7200”。

采用等宽分箱，如宽度为 1200，分箱结果为“箱 1：2300，2500，2800，3000，3500；箱 2：4000，4500，4800，5000；箱 3：5300，5500，6000，6200；箱 4：6700，7000，7800”。

（2）数据平滑。在将数据划分到不同的箱子之后，可以运用以下 3 种策略对每个箱子中的数据进行平滑处理。

① 平均值平滑。箱子中的每一个值都被箱中数值的平均值替换。

② 中值平滑。箱子中的每一个值都被箱中数值的中值替换。

③ 边界平滑。箱子中的最大值和最小值称为箱子的边界，箱子中的每一个值都被最近的边界值替换。

6）回归插补

对于两个相关变量之间的变化模式，通过使数据适合一个函数来平滑数据。如果变量之间存在依赖关系，也就是 $y=f(x)$，那么就可以设法求出依赖关系 f，再根据 x 来预测 y，这也是回归问题的实质。在实际问题中更为常见的假设是 $p(y)=N(f(x))$，N 为正态分布。假设 y 是观测值并且存在异常值，求出 x 和 y 之间的依赖关系，再根据 x 来更新 y 的值，这样就能去

除其中的异常值，这也是回归消除异常值的原理。

7）多重插补

多重插补的处理要先删除 y 变量的缺失值，然后再进行插补，需要注意被解释变量有缺失值时不能插补，只能删除；另外只对放入模型的解释变量进行插补。

8）不处理

根据异常值的性质特点，使用更加稳健的模型来修饰，然后直接在数据集上进行数据挖掘。

5.2.3　格式内容清洗

在一般情况下，数据是由用户/访客产生的，也就有很大的可能存在格式和内容上不一致的情况，所以在进行模型构建之前需要先进行数据的格式内容清洗操作。格式内容清洗主要有以下几类：

（1）时间、日期、数值、半角/全角字符等显示格式不一致。直接将数据转换为一类格式即可，该问题一般出现在多个数据源整合的情况下。

（2）内容中有不该存在的字符。最典型的就是在头部、中间、尾部有空格等问题，在这种情况下，需要以半自动校验加半人工方式来寻找问题，并去除不需要的字符。

（3）内容与属性应有的内容不符。比如将姓名写成了性别、身份证号写成了手机号等问题。

5.2.4　逻辑错误清洗

一般通过简单的逻辑推理发现数据中的问题数据，防止分析结果走偏，主要包含以下几个方面的内容：

（1）数据去重。这里的去重不是简单地去除完全相同的数据，例如某系统中有"许文秀"和"许文　秀"等，系统认为是两个不同的字符串。

（2）去除/替换不合理的数据。例如某人填表时将年龄误填了 200 岁、年收入误填了 100000 万等。

（3）删除/重构不可靠的属性值（修改矛盾的内容）。例如某人的身份证号是 1101031980XXXXXXXX，而填表时将年龄填了 23 岁。在这种时候需要根据字段的数据来源判定哪个字段提供的信息更为可靠，删除或重构不可靠的字段。

5.2.5　非需求数据清洗

在一般情况下，人们会尽可能多地收集数据，那么是不是把所有的属性数据都可以应用到模型构建过程中呢？其实将所有的属性都放到构建模型中，最终模型的效果并不一定好。一般情况下，属性越多，模型的构建就会越复杂、越慢，所以有时候可以考虑将不相关的属性甚至是关联关系很弱的属性进行删除。

简而言之，这一步就是删除非需求的字段，但实际操作起来有很多问题，例如：

（1）把看上去不需要但实际上对业务很重要的字段删除了。

（2）觉得某个字段有用，但又没想好怎么用，不知道是否该删除。

（3）一时看走眼了，删错了字段。

对前两种情况的建议是，如果数据量没有大到不删除字段就没办法处理的程度，那么能不删除的字段尽量不删除，第三种情况需要经常备份原始数据。

5.2.6 关联性验证

如果数据有多个来源，那么有必要进行关联性验证，这经常应用到多数据源合并的过程中。通过验证数据之间的关联性来选择比较正确的特征属性，例如汽车的线下购买信息和电话客服问卷信息，两者之间可以通过姓名和手机号进行关联操作，匹配两者之间的车辆信息是否为同一辆，如果不是，那么就需要进行数据调整。

5.3 数据集成

数据集成是把多个不同来源、格式、特点及性质的数据在逻辑上或物理上有机地集中在一起，存放在一个一致的数据存储中，为用户提供全面的数据共享。

5.3.1 数据集成过程中的关键问题

数据集成解决方案越来越多地被企业采用，以支持日渐纷杂的业务项目和技术实施。数据集成过程中的关键问题是实体识别、数据冗余与相关性分析、数据值冲突的检测与处理等。

1. 实体识别

实体识别问题是数据集成中的首要问题，因为来自多个信息源的现实世界的等价实体才能匹配，同时还需要把两个本来不是同一个的实体区别开，主要涉及以下几个方面：

(1) 同名异义。例如“苹果”既可以指苹果手机也可以指苹果水果，又如“病毒”既可以指生物病毒又可以指计算机病毒，因此需要指明数据集成中涉及的到底是哪种实体。

(2) 异名同义。例如“电脑”和“计算机”指的是同一实体，又如诗人“李白”和“李太白”指的是同一个人，这就需要将这些称谓统一起来。

(3) 单位统一。用于描述同一个实体的属性有时候可能会出现单位不统一的情况，例如140cm 和 1.2m。大家要知道计算机在进行事务处理的时候没有量纲，要么统一量纲，要么将量纲标准化。

(4) ID-Mapping。这是一个互联网领域的术语，它是对每一条行为日志数据的唯一标识，目的是将不同数据库或者账号系统中的实体对应起来。例如刘世元同学办了一张中国移动的手机卡，移动公司存储了他的信息，而他在浏览今日头条时也会留下痕迹，如果现在中国移动要和今日头条合作，那么就需要对两边的数据进行集成，也就需要对在中国移动办卡的刘世元和浏览今日头条的刘世元进行匹配，这个过程可以根据设备的 IMSI(International Mobile Subscriber Identity，国际移动用户识别码)号码通过比照进行。简而言之，ID-Mapping 需要采用唯一识别号(例如学号、身份证号、学校-年级-班级-姓名、设备号等)进行账号的用户匹配，这在解决数据孤岛问题上有着重要的意义。

2. 数据冗余与相关性分析

如果一个属性能由另一个或另一组属性值推导出来，则这个属性可能是冗余的。冗余是数据集成的一个重要问题，有些冗余可以通过相关性分析检测出来。对于数值属性，可以使用协方差来评估一个属性值如何随另一个属性值变化。

1) 标称数据的 χ^2 相关检验

对于标称数据，两个属性 A 和 B 之间的相关关系可以通过 χ^2(卡方)检验发现。假设 A 有 c 个不同值 a_1、a_2、……、a_c，B 有 r 个不同值 b_1、b_2、……、b_r。用 A 和 B 描述的数据记录可以用一个相依表显示，其中 A 的 c 个值构成列，B 的 r 个值构成行。假设(A_i，B_j)表示属性 A 取值 a_i 且属性 B 取值 b_j 的联合事件，即($A=a_i$，$B=b_j$)。每个可能的(A_i，B_j)联合事件都在

表中有自己的单元。χ^2 值(又称 Pearsonχ^2 统计量)可以用下式计算：

$$\chi^2=\sum_{i=1}^{c}\sum_{j=1}^{r}\frac{(o_{ij}-e_{ij})^2}{e_{ij}} \tag{5-1}$$

o_{ij} 是联合事件(A_i,B_j)的观测频度(即实际计数)，而 e_{ij} 是(A_i,B_j)的期望频度，可以用下式计算：

$$e_{ij}=\frac{\text{count}(A=a_i)\times\text{count}(B=b_j)}{n} \tag{5-2}$$

n 是数据记录的个数，count$(A=a_i)$是 A 上具有值 a_i 的记录个数，而 count$(B=b_j)$是 B 上具有值 b_j 的记录个数。式(5-1)中的和在所有 r×c 个单元上计算。这里应注意，对 χ^2 值贡献最大的单元是其实际计数与期望计数很不相同的单元。

χ^2 统计检验假设 A 和 B 是独立的，检验基于显著水平，具有自由度(r−1)×(c−1)。

例 5.8　使用 χ^2 的标称属性的相关分析。假设调查了 1500 个人，每个人对他们喜爱阅读的材料类型是否为小说进行投票，有两个属性“性别”和“喜爱阅读”。每种可能的联合事件观测值(获计数)汇总成如表 5.1 所示的相依表。

表 5.1　属性“性别”和“喜爱阅读”的相依表

阅读材料	男	女	合　计
小说	250	200	450
非小说	50	1000	1050
合计	300	1200	1500

解：先计算(A_i,B_j)的期望频度 e_{ij}：

$$e_{11}=\frac{\text{count}('男')\times\text{count}('小说')}{n}=\frac{300\times450}{1500}=90$$

$$e_{12}=\frac{\text{count}('女')\times\text{count}('小说')}{n}=\frac{1200\times450}{1500}=360$$

$$e_{21}=\frac{\text{count}('男')\times\text{count}('非小说')}{n}=\frac{300\times1050}{1500}=210$$

$$e_{22}=\frac{\text{count}('女')\times\text{count}('非小说')}{n}=\frac{1200\times1050}{1500}=840$$

带入式(5-1)有：

$$\chi^2=\frac{(250-90)^2}{90}+\frac{(200-360)^2}{360}+\frac{(50-210)^2}{210}+\frac{(1000-840)^2}{840}$$

$$=284.44+71.11+121.90+30.48=507.93$$

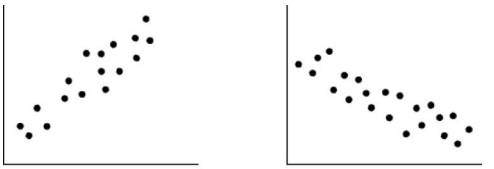
对于表 5.1，自由度为(2−1)(2−1)=1。对于自由度 1，在 0.001 的置信水平下，拒绝假设的值是 10.828(取自 χ^2 分布的百分点表，通常可以在任意统计学教科书中找到)。由于要计算的值大于该值，所以可以拒绝“性别”和“喜爱阅读”独立的假设，并断言对于给定的人群，这两个属性是(强)相关的。

2) 相关系数

对于数值数据，可以通过计算属性 A 和 B 的相关系数(又称皮尔逊积矩相关系数)来分析其相关性。相关系数 $r_{A,B}$ 定义为：

$$r_{A,B}=\frac{\sum_{i=1}^{n}(a_i-\overline{A})(b_i-\overline{B})}{n\sigma_A\sigma_B}=\frac{\sum_{i=1}^{n}(a_ib_i)-n\overline{A}\overline{B}}{n\sigma_A\sigma_B} \tag{5-3}$$

其中，n 是记录的个数，a_i 和 b_i 分别是记录 i 在 A 和 B 上的值，$\overline{A}$ 和 $\overline{B}$ 分别是 A 和 B 的均值，σ_A 和 σ_B 分别是 A 和 B 的标准差，$-1 \leqslant r_{A,B} \leqslant 1$。如果相关系数 $r_{A,B}$ 的取值为 0，则 A 和 B 是独立的，即它们之间不存在相关性；如果相关系数 $r_{A,B}$ 的取值小于 0，则 A 和 B 是负相关性的，一个值随另一个值减少而增加；如果相关系数 $r_{A,B}$ 的取值大于 0，则 A 和 B 是正相关的，意味着 A 值随着 B 值的增加而增加，值越大，相关性越强，如图 5.6 所示。



(a) 正相关　　(b) 负相关

图 5.6　属性间的相关关系

这里需要说明的是，相关性并不蕴涵着因果关系，也就是说如果 A 和 B 是相关的，这并不意味着 A 导致 B 或 B 导致 A。例如，在分析人口统计数据库时发现一个地区学校的数量与该地区所拥有的汽车数量是正相关的，但这并不意味着“学校数量的多少”会导致“汽车数量的多少”。实际上，二者必然地会关联到第三个属性——人口。

用 Python 求相关系数的方法有以下 3 种：

(1) 用 NumPy 模块中的 corrcoef()函数计算相关系数矩阵。

(2) 用 Pandas 模块中 DataFrame 对象自带的相关性计算方法 corr()可以求出所有列之间的相关系数。

(3) 自己编写 Python 程序计算相关系数。

例 5.9　求相关系数示例。

程序代码如下：

```
import seaborn as sna
import pandas as pd
data = sna.load_dataset('iris')                    #加载鸢尾花数据集
df = pd.DataFrame(data,columns = ['sepal_length','sepal_width','petal_length','petal_width'])
result = df.corr()
print(result)
```

程序运行结果如图 5.7 所示。

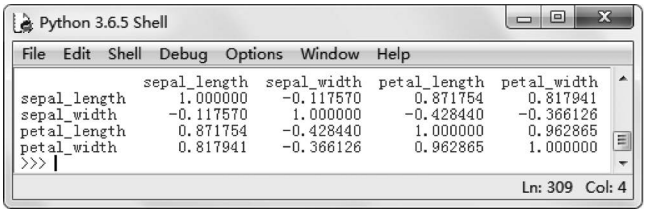


图 5.7　例 5.9 的运行结果

3) 协方差

在概率论和统计学中，协方差(Covariance)用于衡量两个变量的总体误差。方差是协方差的一种特殊情况，即两个变量相同时的协方差。它们都是用来评估两个属性如何一起变化的。考虑两个数值属性 A、B 和 n 次观测值集合$\{(a_1,b_1),(a_2,b_2),\cdots,(a_n,b_n)\}$。A 和 B 的均值分别称为 A 和 B 的期望值，即

$$E(A)=\overline{A}=\frac{\sum_{i=1}^{n}a_i}{n},\quad E(B)=\overline{B}=\frac{\sum_{i=1}^{n}b_i}{n}$$

A 和 B 的协方差定义为：

$$\mathrm{Cov}(A,B)=E((A-\overline{A})(B-\overline{B}))=\frac{\sum_{i=1}^{n}(a_i-\overline{A})(b_i-\overline{B})}{n} \tag{5-4}$$

如果把式(5-3)与式(5-4)相比较，则有：

$$r_{A,B}=\frac{\mathrm{Cov}(A,B)}{\sigma_A\sigma_B}=\frac{E((A-\overline{A})(B-\overline{B}))}{\sigma_A\sigma_B} \tag{5-5}$$

σ_A 和 σ_B 分别是 A 和 B 的标准差。另外还可以证明：

$$\mathrm{Cov}(A,B)=E(A\cdot B)-\overline{A}\overline{B} \tag{5-6}$$

对于两个趋向于一起改变的属性 A 和 B，如果 A 大于 $\overline{A}$(A 的期望值)，则 B 很可能大于 $\overline{B}$(B 的期望值)，此时 A 和 B 的协方差为正；另一方面，如果一个属性值小于期望值，另一个属性趋向于大于它的期望值，则 A 和 B 的协方差为负。

如果 A 和 B 是独立的(不具有相关性)，则 $E(A\cdot B)=E(A)\cdot E(B)$。因此协方差为 $\mathrm{Cov}(A,B)=E(A\cdot B)-\overline{A}\overline{B}=E(A)\cdot E(B)-\overline{A}\overline{B}=0$。其逆不成立。

例 5.10　表 5.2 给出了某电商和某高科技公司在 5 个时间点观测到的股票价格简表，请通过协方差分析股票的走势。

表 5.2　某电商和某高科技公司的股票价格简表

时　间　点	某电商的股票价格/元	某高科技公司的股票价格/元
t1	6	20
t2	5	10
t3	4	14
t4	3	5
t5	2	5

解：

$$E(\text{某电商的股票价格})=\frac{6+5+4+3+2}{5}=\frac{20}{5}=4$$

$$E(\text{某高科技公司的股票价格})=\frac{20+10+14+5+5}{5}=\frac{54}{5}=10.8$$

根据式(5-4)：

$$\begin{aligned}&\mathrm{Cov}(\text{某电商的股票价格},\text{某高科技公司的股票价格})\\&=\frac{6\times20+5\times10+4\times14+3\times5+2\times5}{5}-4\times10.8\\&=50.2-43.2=7\end{aligned}$$

由于协方差为正，因此可以说两个公司的股票同时上涨。

Python 可以利用 NumPy 模块中的 Cov()函数计算协方差。

例 5.11　Python 求协方差示例。

程序代码如下：

```
import numpy as np
from sklearn import datasets
iris = datasets.load_iris()                    #加载鸢尾花数据集
A = iris.data[:,0]
```

```
B = iris.data[:,1]
result = np.cov(A,B)
print(result)
```

3. 检测重复记录

除了检查属性的冗余之外，还要检测重复的记录。所谓重复记录，是指给定唯一的数据实体，存在两个或多个相同的记录。

使用 Python 的 NumPy 模块中的 unique()函数可以去掉一维数组或者列表中的重复元素；对于多维数组，如果指定 axis=0，可以把冗余的行去掉，如果指定 axis=1，可以把冗余的列去掉。

例 5.12 去掉多维数组的重复行。

程序代码如下：

```
import numpy as np
A = [['S1','许文秀','女',20],['S4','于金凤','女',20],['S1','许文秀','女',20], ['S2','刘德峰',
'男',22]]
result = np.unique(A,axis = 0)
print(result)
```

4. 数据值冲突的检测与处理

数据集成还涉及数据值冲突的检测与处理。

对于现实世界中的同一实体，来自不同数据源的属性值可能不同。例如，重量属性可能在一个系统中使用公制单位，而在另一个系统中使用英制单位；位于不同城市的连锁酒店，标间价格可能不同，服务也可能不同(如提供免费早餐等)。

如果要了解全国各个省份中每个高校的学生的成绩信息，需要同时访问每个学校的数据库，但是在数据库中存储学生成绩的方式是不一样的，而且成绩一般有两种类型，一种是基本课程成绩，通常为百分制，另一种是德育评估成绩，通常为等级制(包括 A、B、C、D 4 种，分别表示优秀、良好、合格、不合格)。例如，某大学的学生成绩数据表中每一行是成绩类型和相应的成绩，而另一所大学的学生成绩数据表中每一行是基本课程成绩和德育评估成绩，第三所大学的学生成绩数据库用两个表来存储学生的成绩，第一个表专门存储学生的基本课程成绩，第二个表专门存储学生的德育评估成绩。很明显，该问题是典型的模式层次上的数据冲突问题。再复杂一点，如果想全面地给学生一个综合成绩，那么需要在百分制成绩和等级制成绩之间进行转化，这里又涉及了语义层次上的数据冲突问题。

处理数据值冲突的方法是按照一定的规则建立起底层关系数据库模式的语义模型，然后利用建好的语义冲突本体来扩展关系数据库模式的语义，最后再给出基于本体和数据库语义模型解决冲突的具体方法。

5.3.2 Python 数据集成

在 Python 数据分析中所用到的数据集有可能来自于不同的数据源，因此经常需要对数据子集进行集成处理。使用 Pandas 模块中的 merge()、concat()方法可以完成数据的集成。

1. merge()方法

merge()方法主要是基于两个 DataFrame 对象的共同列进行连接。Python 中的 merge()方法与 SQL 中 join 的用法非常类似，merge()方法的常用形式如下：

```
merge(left,right,how = 'inner',on = None,left_on = None,right_on = None, sort = True)
```

参数说明：

(1) left 为连接的左侧 DataFrame 对象。

(2) right 为连接的右侧 DataFrame 对象。

(3) how 设置连接方式，取值{'inner' | 'outer'|'left'| 'right'}，默认为 inner。inner 是取交集，outer 是取并集，没有的属性取 NaN。left 是左边的取全部，右边的属性取 NaN；right 是右边的取全部，左边的属性取 NaN。这类似于 SQL 中 join 的内连接、外连接、左外连接、右外连接。

(4) on 设置用于连接的列名。

(5) left_on 设置左侧 DataFrame 对象中用于连接键的列。

(6) right_on 设置右侧 DataFrame 对象中用于连接键的列。

(7) sort 设置合并后是否会对数据进行排序，默认为 True，即排序。

例 5.13　merge()方法的数据集成示例。

程序代码如下：

```
import pandas as pd
S_info = pd.DataFrame({'学号':['S1','S2','S3','S4','S5'],
            '姓名':['许文秀','刘德峰','刘世元','于金凤','周新娥']})
course = pd.DataFrame({'学号':['S1','S2','S1','S4','S1'],
            '课程':['C2','C1','C3','C2','C4']})
df = pd.merge(S_info,course)
print(df)
```

程序运行结果如图 5.8 所示。

例 5.14　左、右数据子集中关键字不同的 merge()方法数据集成示例。

程序代码如下：

```
import pandas as pd
S_info = pd.DataFrame({'学号':['S1','S2','S4','S5'],
       '姓名':['许文秀','刘德峰','刘世元','于金凤']})
course = pd.DataFrame({'编号':['S1','S2','S1','S4','S1'],
       '课程':['C2','C1','C3','C2','C4']})
df = pd.merge(S_info,course,left_on = '学号',right_on = '编号')
print(df)
```

程序运行结果如图 5.9 所示。

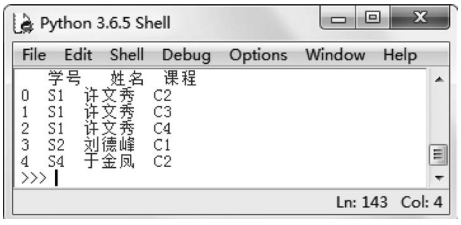


图 5.8　例 5.13 的运行结果

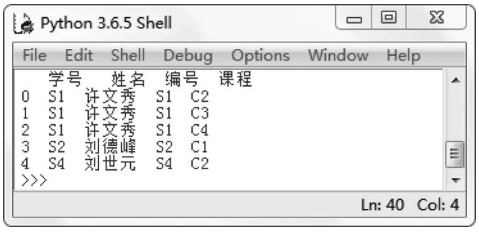


图 5.9　例 5.14 的运行结果

例 5.15　当 how＝'outer'时 merge()方法的数据集成示例。

程序代码如下：

```
import pandas as pd
grade1 = pd.DataFrame({'学号':['S1','S2','S3','S4','S5'],
     '姓名':['许文秀','刘德峰','刘世元','于金凤','周新娥'],
     '高数':[67,92,67,58,78],
     '英语':[82,88,96,90,87]})
```

```
grade2 = pd.DataFrame({'学号':['S1','S2','S4','S5','S6'],
      '数据库技术':[89,34,74,90,83]})
df = pd.merge(grade1,grade2,how = 'outer')
print(df)
```

程序运行结果如图 5.10 所示。

例 5.16 merge()方法通过多个键进行数据集成的示例。

程序代码如下：

```
import pandas as pd
info_s = pd.DataFrame({'学号':['S1','S2','S3','S4','S5'],'姓名':['许文秀','刘德峰','刘世元','于金凤','周新娥'],'性别':['女','男','男','女','女']})
course = pd.DataFrame({'学号':['S1','S2','S1','S3','S5','S2', 'S1'],'姓名':['许文秀','刘德峰','许文秀','刘世元','周新娥','刘德峰','许文秀'],'课程':['C1','C1','C3','C2','C2','C3','C4'],'成绩':[78,82,67,92,89,77,68]})
df = pd.merge(info_s,course,on = ['学号','姓名'])
print(df)
```

程序运行结果如图 5.11 所示。

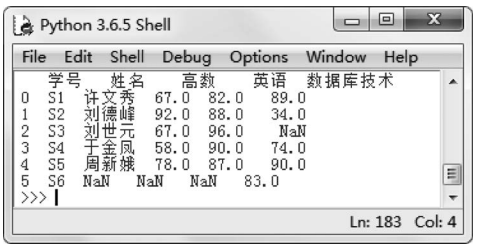


图 5.10 例 5.15 的运行结果

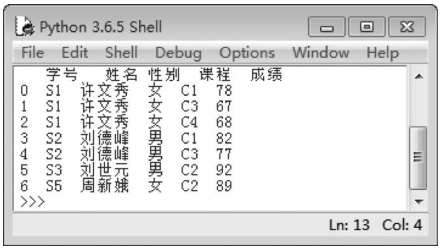


图 5.11 例 5.16 的运行结果

2. concat()方法

concat()方法用于对 Series 对象或 DataFrame 对象的数据集进行连接，可以指定按某个轴进行(行或列)连接，也可以指定连接方式 outer 或 inner。与 SQL 不同的是，concat()不会去重，要达到去重的效果，可以使用 drop_duplicates()方法。其常用形式如下：

```
concat(objs,axis = 0,join = 'outer')
```

参数说明：

(1) objs 为 Series 对象、DataFrame 对象或 list 对象。

(2) axis 设置需要连接的轴，axis=0 表示行连接，axis=1 表示列连接。

(3) join 设置连接的方式，取值为 inner 或 outer。

例 5.17 concat()方法连接示例。

程序代码如下：

```
import pandas as pd
data1 = [['S1','许文秀','女'],['S2','刘德峰','男'],
                        ['S3','刘世元','男'],['S4','于金凤','女'],
                        ['S5','周新娥','女']]
df1 = pd.DataFrame(data1,columns = ['学号','姓名','性别'])
data2 = [[78,89,80,61],[77,83,78,66],[90,54,68,78],[76,66,80,82]]
df2 = pd.DataFrame(data2,columns = ['高数','英语','数据库技术','数据挖掘'])
df = pd.concat([df1,df2],axis = 1,join = 'outer')
pd.set_option('display.unicode.east_asian_width', True)   #显示的中文列标题与数据对齐
print(df)
```

程序运行结果如图 5.12 所示。

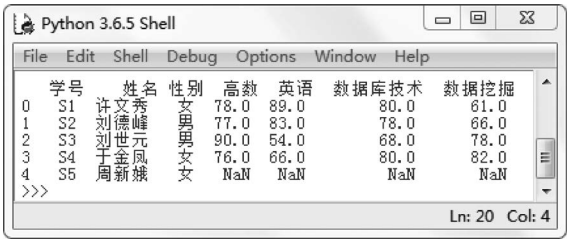


图 5.12　例 5.17 的运行结果

5.4　数据标准化

在进行数据分析之前，通常需要先将数据标准化(Standardization)，利用标准化后的数据进行数据分析，能够避免因属性之间不同度量和取值范围差异造成数据对分析结果的影响。

5.4.1　z-score 方法

z-score 方法是基于原始数据的均值和标准差来进行数据标准化的，处理后的数据均值为 0，方差为 1，符合标准正态分布，且无量纲。其主要目的是将不同量级的数据统一转换为同一个量级，用计算出的 z-score 值衡量，保证了数据之间具有可比性。其常用形式如下：

$$x_{normalization} = \frac{x - \mu}{\sigma} \tag{5-7}$$

其中，x 表示原始数据，μ 表示原始数据的平均值，σ 表示原始数据的标准差，$x_{normalization}$ 表示标准化后的数据。数据标准化的方法有自定义和 StandardScaler()等方法。

1. 自定义方法

用自定义方法进行数据标准化就是利用式(5-7)编程实现。

例 5.18　自定义数据标准化示例。

程序代码如下：

```
def my_scale(data):
    mean = sum(data)/len(data)                                    #求均值
    variance = (sum([(i - mean) ** 2 for i in data]))/len(data)   #求方差
    normal = [(i - mean)/(variance) ** 0.5 for i in data]         #按照公式标准化
    return normal
import numpy as np
X = np.array([[1., -1.,2.],[2.,0.,0.],[0.,1., -1.]])
scale = my_scale(X)
print(scale)
```

程序运行结果如图 5.13 所示。

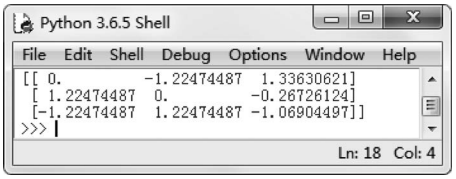


图 5.13　例 5.18 的运行结果

2. StandardScaler()

用户可以使用 sklearn 模块中的 StandardScaler()方法来实现数据标准化，但每次使用时需要调用 sklearn 包。

例 5.19　StandardScaler()方法数据标准化示例。

程序代码如下：

```
import numpy as np
from sklearn import preprocessing
X = np.array([[1., -1.,2.],[ 2.,0.,0.],[0.,1., -1.]])
scaler = preprocessing.StandardScaler().fit(X)
print(scaler.transform(X))      #在 fit 的基础上进行标准化、降维、归一化等操作
```

5.4.2 极差标准化方法

极差标准化也称为区间缩放法或0-1标准化，它是对原始数据所进行的一种线性变换，将原始数据映射到[0,1]区间。其常用形式如下：

$$y_{ij}=\frac{x_{ij}-\min\{x_{ij}\}}{\max\{x_{ij}\}-\min\{x_{ij}\}} \tag{5-8}$$

其中，$\min\{x_{ij}\}$和$\max\{x_{ij}\}$指的分别是和x_{ij}同一数据集的最小值和最大值。极差标准化的方法有自定义和MinMaxScaler()等方法。

1. 自定义方法

用自定义方法进行数据标准化就是利用式(5-8)编程实现。

例 5.20 极差标准化自定义方法示例。

程序代码如下：

```
def my_scale(data):
    data = (data - data.min())/(data.max() - data.min())
    return data
import numpy as np
X = np.array([[1., -1.,2.],[2.,0.,0.],[0.,1., -1.]])
scale = my_scale(X)
print(scale)
```

程序运行结果如图5.14所示。

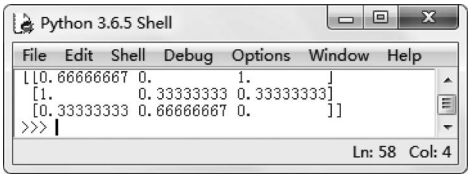


图5.14 例5.20的运行结果

2. MinMaxScaler()

用户可以使用sklearn中的MinMaxScaler()方法来实现数据标准化。MinMaxScaler()有一个重要参数——feature_range，控制着数据压缩到的范围，默认是[0,1]。

例 5.21 MinMaxScaler()方法数据标准化示例。

程序代码如下：

```
import numpy as np
from sklearn import preprocessing
X = np.array([[1., -1.,2.],[2.,0.,0.],[0.,1., -1.]])
scaler = preprocessing.MinMaxScaler()
scaler.fit(X)
print(scaler.transform(X))
```

5.4.3 最大绝对值标准化方法

最大绝对值标准化方法是x_{ij}除以其最大的绝对值，也就是将原始数据映射到[-1,1]区间内。其常用形式为：

$$y_{ij}=\frac{x_{ij}}{\max\{|x_{ij}|\}} \tag{5-9}$$

这种情况适合均值在0附近的数据集，或者稀疏矩阵。

用户可以使用 sklearn 中的 MaxAbsScaler()方法来实现最大绝对值标准化。

例 5.22　MaxAbsScaler()方法数据标准化示例。

程序代码如下：

```
import numpy as np
from sklearn import preprocessing
X = np.array([[1., -1.,2.],[2.,0.,0.],[0.,1., -1.]])
scaler = preprocessing.MaxAbsScaler()
scaler.fit(X)
print(scaler.transform(X))
```

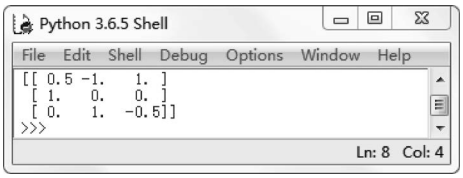


图 5.15　例 5.22 的运行结果

程序运行结果如图 5.15 所示。

5.5　数据归约

在进行数据分析时，所得到的数据集可能很大，在海量数据集上进行数据挖掘会需要很长的时间，因此要对数据进行归约。所谓数据归约(Data Reduction)，是指在对挖掘任务和数据本身内容理解的基础上寻找数据的有用特征，以缩减数据规模，从而在尽可能保持数据原貌的前提下最大限度地精简数据量。简而言之，在归约后的数据集上进行数据挖掘会更有效，而且仍会产生相同或相似的分析结果。数据归约包括维归约、数量归约和数据压缩。

5.5.1　维归约

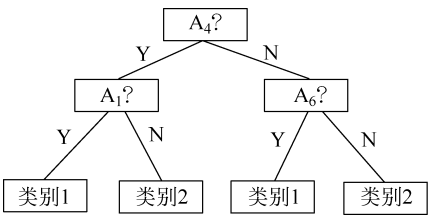
维归约(Dimensionality Reduction)的思路是减少所考虑的随机变量或属性的个数，所用方法有属性子集选择、小波变换和主成分分析。其目的就是把原始数据变换或投影到较小的数据空间上。

1. 属性子集选择

属性子集选择是一种维归约方法，对不相关、弱相关或冗余的属性或维进行检测和删除。其目标是找出最小属性集，使得数据集的概率分布尽可能地接近使用所有属性得到的原分布。它减少了数据模式上的属性数目，使得模式更易于理解。

对于 n 个属性，有 2^n 个可能的子集。对于属性子集选择，通常使用搜索子空间的启发式算法。这些方法是典型的贪心(启发式)算法，它们的策略是进行局部最优选择，期望由此得到全局最优解。在实践中，这种贪心方法是有效的，并可以逼近最优解。“最好的”(和“最差的”)属性通常使用统计显著性检验来确定。当然也可以使用一些其他属性评估度量，例如建立决策树使用的信息增益度量。基本启发式方法包括的主要技术如表 5.3 所示。

表 5.3　属性子集选择贪心(启发式)方法示意表

向 前 选 择	向 后 删 除	决策树归约
初始属性集： $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ 初始化归约集： $\{\}$ $=>\{A_1, A_4\}$ =>归约后的属性集： $\{A_1, A_4, A_6\}$	初始属性集： $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ $=>\{A_1, A_4, A_5, A_6\}$ =>归约后的属性集： $\{A_1, A_4, A_6\}$	初始属性集：$\{A_1, A_2, A_3, A_4, A_5, A_6\}$  归约后的属性集：$\{A_1, A_4, A_6\}$

(1) 逐步向前选择。逐步向前选择过程由空属性集作为归约集的起始,确定原属性集中最好的属性并添加到归约集中,迭代剩下的原属性集,并将最好的属性添加到该集合中。

(2) 逐步向后删除。逐步向后删除过程由整个属性集开始,在每次迭代中删除属性集中最差的属性。

(3) 逐步向前选择和逐步向后删除的组合。该方法将逐步向前选择和逐步向后删除相结合,每一步选择一个最好的属性并在属性集中删除一个最差的属性。

(4) 决策树归约。构造一个类似于流程图的决策树结构,每个内部结点表示一个属性上的测试,每个分支对应于测试的一个结果。在每个结点上选择"最好"的属性,将数据划分成类。在利用决策树进行子集选择时,由给定的数据构造决策树,不出现在树中的所有属性假定是不相关的,出现在树中的属性形成归约后的属性子集。

这些方法的结束条件可以不同,也可以使用一个度量阈值决定何时终止属性选择过程。

2. 小波变换

小波变换(Wavelet Transform,WT)是一种新的变换分析方法,它提供一个随频率改变的"时间-频率"窗口,是进行信号时频分析和处理的理想工具。小波变换的主要特点是通过变换能够充分突出问题某些方面的特征,能对时间(空间)频率进行局部化分析,通过伸缩平移运算对信号(函数)逐步进行多尺度细化,最终达到在高频处时间细分、低频处频率细分、小波变换能自动适应时频信号分析的要求,从而可聚焦到信号的任意细节。

小波变换有以下特点:

(1) 对于小频率值,频域分辨率高,时域分辨率低。

(2) 对于大频率值,频域分辨率低,时域分辨率高。

小波变换在频域分辨率和时域分辨率两者之间进行权衡:在与时间相关的特征上具有高分辨率,而在与频率相关的特征上也具有高分辨率。

例 **5.23** 小波变换示例。

程序代码如下:

```
import pywt
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.family'] = 'STSong'                          #图形中显示汉字
plt.rcParams['font.size'] = 12
img = cv.imread('D:/Data_Mining/tx5 - 23.jpg',0)                #读取图像
#对 img 进行 haar 小波变换,分量分别是低频、水平高频、垂直高频、对角线高频
cA,(cH,cV,cD) = pywt.dwt2(img,'haar')
#小波变换之后,低频分量对应的图像
p1 = plt.figure(figsize = (12,6),dpi = 80)                      #第一幅子图,并确定画布大小
ax1 = p1.add_subplot(2,2,1)                          #创建一个 2 行 2 列的子图,并开始绘制第一幅图
plt.axis('off')                                                 #不显示坐标轴
plt.title('低频分量图像')
AA1 = np.uint8(np.uint8(cA/np.max(cA) * 255))
plt.imshow(AA1,'gray')
ax1 = p1.add_subplot(2,2,2)
plt.axis('off')
plt.title('水平高频分量图像')
AA2 = np.uint8(np.uint8(cA/np.max(cH) * 255))
plt.imshow(AA2,'gray')
ax3 = p1.add_subplot(2,2,3)
plt.title('垂直高频分量图像')
plt.axis('off')
```

```
AA3 = np.uint8(np.uint8(cV/np.max(cH) * 255))
plt.imshow(AA3,'gray')
ax4 = p1.add_subplot(2,2,4)
plt.title('对角线高频分量图像')
plt.axis('off')
AA4 = np.uint8(np.uint8(cD/np.max(cH) * 255))
plt.imshow(AA4,'gray')
plt.show()
```

程序运行结果如图 5.16 所示。

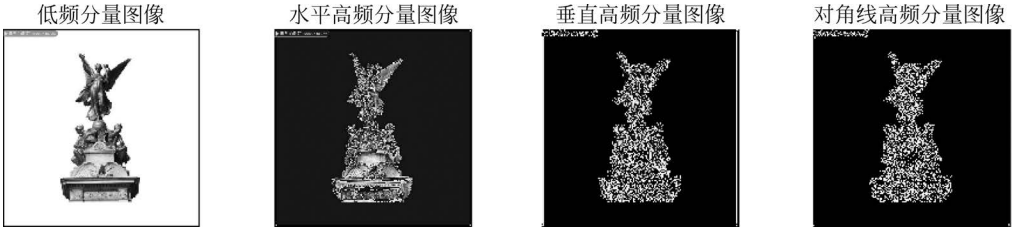


图 5.16　例 5.23 的运行结果

3. 主成分分析

主成分分析(Principal Component Analysis,PCA)是一种用于连续属性的数据降维方法。它需要找到一个合理的方法,在减少需要分析属性的同时尽量减少原指标包含信息的损失,以达到对所收集数据进行全面分析的目的。

1) PCA 算法

一般各属性间存在一定的相关关系,因此有可能用较少的综合属性来近似地表达整体综合信息。

例如,某班学生的语文(满分为 100 分)、数学(满分为 150 分)、物理(满分为 100 分)、化学(满分为 100 分)成绩如表 5.4 所示。

表 5.4　某班学生的成绩表

学　号	语　文	数　学	物　理	化　学
S1	90	140	99	100
S2	90	97	88	92
S3	90	110	79	83
…	…	…	…	…

首先假设这些科目成绩不相关,也就是说某一科目考多少分与其他科目没有关系。如果通过学生成绩进行一个简单排序,因为语文成绩相同,所以数学、物理、化学这 3 门课的成绩构成了这组数据的主成分(数学可以作为第一主成分,因为数学成绩最分散)。

主成分分析(PCA)又称 K-L(Karhunen-Loeve)方法,它是一种最常用的降维方法。PCA 通常用于高维数据集的探索与可视化,还可以用作数据压缩和预处理等,在数据压缩消除冗余和噪音消除等领域也有广泛的应用。

PCA 的主要目的是找出数据中最主要的特征代替原始数据。具体地,假如数据集是 n 维的,共有 m 个数据(x(1),x(2),…,x(m))。希望将这 m 个数据的维度从 n 维降到 n′维,这 m 个 n′维的数据集尽可能地代表原始数据集。

PCA 算法描述如下:

输入:n 维数据样本集 D=$\{x(1),x(2),\cdots,x(m)\}$,降维到的维数 n′。

输出:降维后的 n′维数据样本集 D′。

处理流程：

step1 对所有的样本进行中心化处理：$x(i)' = x(i) - \frac{1}{m}\sum_{j=1}^{m} x(j)$；

step2 计算样本的协方差矩阵 XX^T；

step3 对矩阵 XX^T 进行特征值分解；

step4 取出最大的 n'个特征值对应的特征向量$(w_1, w_2, \cdots, w_{n'})$，将所有的特征向量标准化后组成特征向量矩阵 W；

step5 对样本集中的每一个样本 x(i)，转化为新的样本 $z(i) = W^T x(i)$；

step6 得到输出样本集 $D' = (z(1), z(2), \cdots, z(m))$。

例 5.24 PCA 算法示例。

原始数据 $X(i) = (x(i), y(i))$，$x(i)' = x(i) - \bar{x}$，$y(i)' = y(i) - \bar{y}$，如表 5.5 所示。

表 5.5 原始数据 x(i)、y(i)与中心化数据 x(i)′、y(i)′

X(i)	x(i)	y(i)	x(i)′	y(i)′
X(1)	2.5	2.4	0.69	0.49
X(2)	0.5	0.7	−1.31	−1.21
X(3)	2.2	2.9	0.39	0.99
X(4)	1.9	2.2	0.09	0.29
X(5)	3.1	3.0	1.29	1.09
X(6)	2.3	2.7	0.49	0.79
X(7)	2	1.6	0.19	−0.31
X(8)	1	1.1	−0.81	−0.81
X(9)	1.5	1.6	−0.31	−0.31
X(10)	1.1	0.9	−0.71	−1.01
平均值	$\bar{x}=1.81$	$\bar{y}=1.91$		

(1) 对所有的样本进行中心化处理：x(i)′、y(i)′如表 5.5 所示。

(2) 求特征协方差矩阵：

$$\text{cov} = \begin{pmatrix} 0.616\,555\,556 & 0.615\,444\,444 \\ 0.615\,444\,444 & 0.716\,555\,556 \end{pmatrix}$$

(3) 求协方差矩阵的特征值和特征向量：

由 $|\text{cov} - \lambda I_2| = 0$，求得特征值：$\lambda_1 = 0.490\,833\,989$，$\lambda_2 = 1.28\,402\,771$。

由 $\text{cov}V = \lambda V$，求出特征向量矩阵并单位化：$U = \begin{pmatrix} -0.735\,178\,656 & -0.677\,873\,399 \\ 0.677\,873\,399 & -0.735\,178\,656 \end{pmatrix}$。

(4) 将特征值按照从大到小的顺序排序，这里选择其中最大的，$\lambda_2 = 1.28\,402\,771$，对应的特征向量为 $\mathbf{V}' = (0.677\,873\,399, -0.735\,178\,656)^T$。

(5) 在表 5.5 中，利用公式 $z(i) = (x(i)', y(i'))\mathbf{V}'(i=1,2,\cdots,10)$将原始样例的二维特征变成了一维，这就是原始特征在一维上的投影。

2) 主成分分析函数 PCA()

Python 的主成分分析利用 PCA()函数，其常用形式如下：

```
PCA(n_components = None,copy = True,whiten = False)
```

参数说明：

(1) n_components 用于设置想要的特征维度数目，可以是 int 型的数字，也可以是阈值百

分比，例如 95%。

（2）copy 为 bool 类型，取值为 True 或者 False，表示是否将原始数据复制一份，这样运行后原始数据值不会改变，默认为 True。

（3）whiten 为 bool 类型，设置是否对降维后的数据进行标准化，使方差为 1，默认为 False。

3）PCA 对象的常用属性和方法

PCA 对象的常用属性如下：

（1）explained_variance 表示降维后各主成分的方差值，方差值越大，表明越重要。

（2）explained_variance_ratio_表示各主成分的贡献率。

（3）components_表示特征空间中主特征方向的基向量，即公式推导中的特征向量组成的特征矩阵，这个矩阵的每一行都是一个特征向量。它是按照特征值由大到小的顺序进行排列的。

（4）mean_表示通过训练数据估计的每个特征上的均值。

PCA 对象的常用方法如下：

（1）fit(X)表示用数据 X 来训练 PCA 模型。

（2）fit_transform(X)表示用训练数据集 X 来训练 PCA 模型，同时返回降维后的数据。

（3）inverse_transform()表示将降维后的数据转换成原始数据。

（4）transform(X)表示将数据 X 转换成降维后的数据。

例 5.25　PCA()函数应用示例。

程序代码如下：

```
import numpy as np
from sklearn.decomposition import PCA
X = np.array([[ - 1,2,66, - 1],[ - 2,6,58, - 1],[ - 3,8,45, - 2],[1,9,36,1],[2,10,62,1],[3,5,83,
2]])
pca = PCA(n_components = 2)                    #降到二维
pca.fit(X)                                     #训练
newX = pca.fit_transform(X)                    #降维后的数据
print(pca.explained_variance_ratio_)           #输出贡献率
print(newX)                                    #输出降维后的数据
```

程序运行结果如图 5.17 所示。

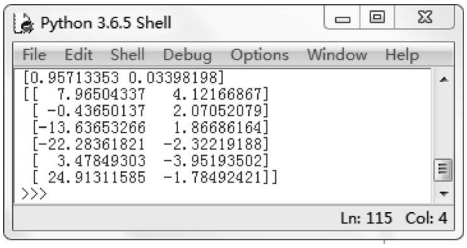


图 5.17　例 5.25 的运行结果

第一行为各主成分的贡献率，可以看出第一个特征占了很大比重，后面几行是降维后的数据。

5.5.2　数量归约

数量归约(Numerosity Reduction)是用替代的、较小的数据表示形式替换原始数据。

1. 特征归约

特征归约是从原有的特征中删除不重要或不相关的特征，或者通过对特征进行重组来减少特征的个数。其原则是在保留甚至提高原有判别能力的同时减少特征向量的维度。特征归约算法的输入是一组特征，输出是它的一个子集。在领域知识缺乏的情况下进行特征归约时一般包括 3 个步骤：

step1 搜索过程。在特征空间中搜索特征子集，由选中的特征构成的每个子集称为一个状态。

step2 评估过程。输入一个状态，通过评估函数对应预先设定的阈值输出一个评估值。

step3 分类过程。使用最终的特征集完成最后的算法。

特征归约处理的优点如下：

(1) 用最少的数据，提高挖掘效率。

(2) 具有更高的数据挖掘处理精度。

(3) 可以得到简单的数据挖掘处理结果。

(4) 使用更少的特征。

2. 样本归约

样本归约就是从数据集中选出一个具有代表性的样本子集，子集大小的确定要考虑计算成本、存储要求、估计量的精度以及其他一些与算法和数据特性有关的因素。数据挖掘处理的初始数据集描述了一个极大的总体，其中最大和最关键的就是样本的数目，也就是数据表中的记录数。一般对数据的分析只基于初始样本集的一个子集。在获得数据的子集后，用它来提供整个数据集的一些信息，这个子集通常叫作估计量，它的质量依赖于所选子集中的样本。事实上，取样过程总会造成取样误差，取样误差对所有的方法和策略来讲都是固有的、不可避免的，当子集的规模变大时，取样误差一般会降低。与针对整个数据集的数据挖掘相比，样本归约具有减少成本、速度更快、范围更广等优点，有时甚至能获得更高的精度。

3. 特征值归约

特征值归约是特征值离散化技术，它将连续型特征的值离散化，使之成为少量的区间，每个区间映射到一个离散符号。这种技术的优点在于简化了数据描述，并易于人们理解数据和最终的挖掘结果。特征值归约可以是有参的，也可以是无参的。有参方法使用一个模型来评估数据，只需存放参数，而不需要存放实际数据。

有参的特征值归约有以下两种：

(1) 回归。回归包括线性回归和多元回归。

(2) 对数线性模型。其近似离散多维概率分布。

无参的特征值归约有以下 3 种：

(1) 直方图。其采用分箱近似数据分布，其中 V-最优和 MaxDiff 直方图是最精确和最实用的。

(2) 聚类。这种归约将数据样本视为对象，将对象划分为群或聚类，使得在一个簇中的对象“相似”而与其他簇中的对象“不相似”，在数据归约时用数据的簇代替实际数据。

(3) 选样。这种归约用较少的随机样本表示大数据集，例如简单选择 n 个样本(类似样本归约)、聚类选样和分层选样等。

5.5.3 数据压缩

数据压缩(Data Compression)就是使用变换得到原始数据的归约或“压缩”表示。如果对

压缩后的数据进行重构时不损失信息，则该数据归约被称为无损的，否则称为有损的。现在比较流行和有效的有损数据压缩方法有小波变换和主成分分析，小波变换对于稀疏或倾斜数据以及具有有序属性的数据有很好的压缩效果。

例 5.26　数据压缩示例。

在电商评论文本中，最常见的就是数据质量参差不齐，通过简单的去重处理，可以删除一部分相同的评论，但是不能删除单条评论文本中重复出现的文字，而进行词语压缩的目的就是将单条文本中的重复文字删除。在本例中评论文本为“质量很好很好很好很好很好质量很好质量很好”“差差差差差差差差差差差差”“一般一般一般一般一般一般”等。

程序代码如下：

```
import numpy as np
dictB = ['质量', '质量很好','差', '一般']
maxDictB = max([len(word) for word in dictB])
sen1 = '质量很好很好很好很好很好质量很好质量很好'
sen2 = '差差差差差差差差差差差差'
sen3 = '一般一般一般一般一般一般'
def cutB(sentence):                          #基于字典的逆向最大匹配中文分词
    result = []
    sentenceLen = len(sentence)
    while sentenceLen > 0:
        word = ''
        for i in range(maxDictB,0, -1):
            piece = sentence[sentenceLen - i:sentenceLen]
            if piece in dictB:
                word = piece
                result.append(word)
                sentenceLen -= i
                break
        if word is '':
            sentenceLen -= 1
            result.append(sentence[sentenceLen])
    print(np.unique(result[::-1]))    #去掉重复词
cutB(sen1)
cutB(sen2)
cutB(sen3)
```

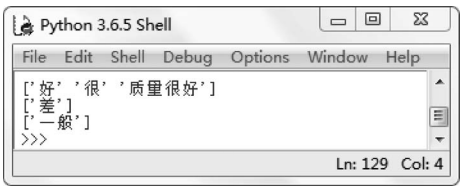


图 5.18　例 5.26 的运行结果

程序运行结果如图 5.18 所示。

5.6　数据变换与数据离散化

在数据预处理的过程中，不同的数据适合用不同的数据挖掘算法。数据变换是一种将原始数据变换成较好格式的方法。数据离散化是一种数据变换形式，它能减少算法的时间和空间开销，提高系统对数据样本的分类、聚类能力和抗噪声能力。

5.6.1　数据变换

数据变换常用于对数据进行规范化处理，以便将不同渠道的数据统一到一个目标数据集中。常见的数据变换包括特征二值化、特征标准化、连续特征变化、独热编码(One-Hot-coding)等。

1. 特征二值化

特征二值化(Binary Quantization)的核心在于设定一个阈值，将特征值与该阈值比较后转换为 0 或 1(有时只考虑某个特征出现与否，不考虑出现次数、程度)，它的目的是将连续数值细粒度的度量转化为粗粒度的度量。

sklearn.preprocessing.Binarizer()是一种属于预处理模块的方法，它在离散连续特征值中起关键作用。其常用形式如下：

```
Binarizer(threshold = 0.0)
```

其中，参数 threshold 是给定的阈值(float)，可选项，小于或等于 threshold 的值映射为 0，否则映射为 1。在默认情况下，阈值为 0.0。

例 5.27 特征二值化示例。

程序代码如下：

```
from sklearn.preprocessing import Binarizer
data = [[1,2,4],[1,2,6],[3,2,2],[4,3,8]]
binar = Binarizer(threshold = 3)          #将数值型数据转化为布尔型的二值数据
print(binar.fit_transform(data))          #对数据先进行拟合,然后标准化
```

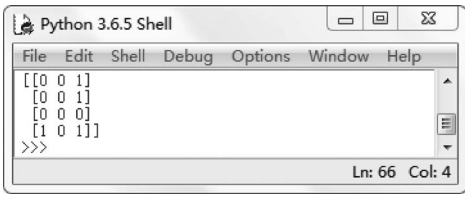


图 5.19 例 5.27 的运行结果

程序运行结果如图 5.19 所示。

2. 特征标准化

特征标准化(Characteristic Standardization)也称为数据无量纲化，主要包括总和标准化、标准差标准化、极差标准化(见 5.4.2 节)、最大绝对值标准化(见 5.4.3 节)。基于参数的模型或基于距离的模型都需要进行特征标准化。

1) 总和标准化

总和标准化处理后的数据在(0,1)区间内，并且它们的和为 1。总和标准化的步骤和公式非常简单，即分别求出各特征数据总和，用各特征的数据分别除以数据总和。其常用形式如下：

$$x'_{ij}=\frac{x_{ij}}{\sum_{i=1}^{m}x_{ij}},\quad \sum_{i=1}^{m}\sum_{j=1}^{n}x_{ij}=1 \tag{5-10}$$

2) 标准差标准化

标准差标准化的常用形式如下：

$$x'_{ij}=\frac{x_{ij}-\bar{x}_j}{s_j},\quad i=1,2,\cdots,m;j=1,2,\cdots,n \tag{5-11}$$

其中，$\bar{x}_j=\frac{1}{m}\sum_{i=1}^{m}x_{ij}$，$s_j=\sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_{ij}-\bar{x}_j)^2}$。

标准差标准化处理后所得到的新数据具有各特征(指标)的平均值为 0、标准差为 1 的特点，即：

$$\bar{x}'_j=\frac{1}{m}\sum_{i=1}^{m}x'_{ij}=0,\quad s'_j=\sqrt{\frac{1}{m}\sum_{i=1}^{m}(x'_{ij}-\bar{x}_j)^2}=1$$

3. 连续特征变换

连续特征变换(Continuous Feature Transformation)的常用方法有 3 种，即基于多项式的特征变换、基于指数函数的特征变换、基于对数函数的特征变换。连续特征变换能够增加数据的非线性特征并获取特征之间关系的可能性，有效地提高了模型的复杂度。

1) 多项式变换

一般来说，多项式变换(Polynomial Transformation)都是按照下面的方式进行的，一次函

数(degree=1)：f=kx+b；二次函数(degree=2)：$f=ax^2+bx+w$；三次函数(degree=3)：$f=ax^3+bx^2+cx+w$。多项式变换可以适当地提升模型的拟合能力，在线性回归模型上具有较广泛的应用。

如果对两个特征 u、v 进行多项式变换操作，那么就相当于多出来 3 个特征，即 u^2、u×v、v^2。一般在使用支持向量机的时候，由于数据在低维度上是不可分的，需要对数据做一个高维度的映射，使得数据能够在高维度上是可分的。

例如

$$\text{matrix}=\begin{pmatrix}0 & 1 & 2\\ 3 & 4 & 5\\ 6 & 7 & 8\end{pmatrix}$$

这里以第 2 行[3 4 5]为例进行多项式变换：

当 degree=2 时，变换后为[1　3　4　5　3×3　3×4　3×5　4×4　4×5　5×5]。

当 degree=3 时，变换后为[1　3　4　5　3×3　3×4　3×5　4×4　4×5　5×5　3×3×3　3×3×4　3×3×5　4×4×3　4×3×5　5×5×3　4×4×4　4×4×5　4×5×5　5×5×5]。

在 Python 中将数据变换为多项式特征的函数为 PolynomialFeatures()，其常用形式如下：

```
PolynomialFeatures(degree = 2)
```

degree=2 表示多项式的变化维度为 2。

例 5.28　多项式变换示例。

程序代码如下：

```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import FunctionTransformer
X = np.arange(9).reshape(3,3)          #生成多项式
print(X)
ploy = PolynomialFeatures(degree = 2)
print(ploy.fit_transform(X))
ploy = PolynomialFeatures(degree = 3)
print(ploy.fit_transform(X))
```

程序运行结果如图 5.20 所示。

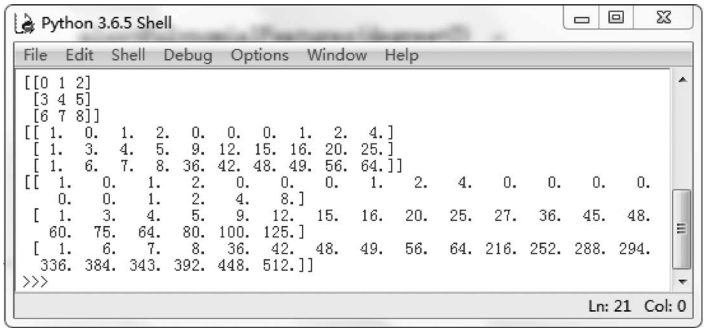


图 5.20　例 5.28 的运行结果

2) 指数变换

进行指数变换(Exponential Transformation)可以改变原先的数据分布，达到处理数据的目的。

在 NumPy 库中有以 e 为底的指数函数 exp()。

例 5.29 指数变换示例。

程序代码如下：

```
import numpy as np
from sklearn.preprocessing import FunctionTransformer
from sklearn.datasets import load_iris
iris = load_iris()
print('原数据：\n',iris.data[0])        #鸢尾花数据集的第[0]行数据
df = FunctionTransformer(np.exp)
df1 = df.fit_transform(iris.data[0])
print('指数变换后的数据：\n',df1)
```

程序运行结果如图 5.21 所示。

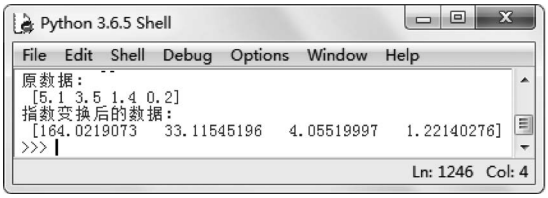


图 5.21 例 5.29 的运行结果

3）对数变换

进行对数变换(Logarithmic Transformation)可以改变原先的数据分布，其目的主要如下：

(1) 可以缩小数据的绝对数值，方便计算。

(2) 可以把乘法计算转换为加法计算。

(3) 分布的改变可以带来意想不到的效果。

NumPy 库中就有几类对数(log 表示以 e 为底的对数、log10 表示以 10 为底的对数、log2 表示以 2 为底的对数等)变换的方法，可以通过 from numpy import < log_name >导入使用。

例 5.30 对数变换示例。

程序代码如下：

```
from numpy import log
from sklearn.preprocessing import FunctionTransformer
from sklearn.datasets import load_iris
iris = load_iris()
print('原数据：\n',iris.data)
df = FunctionTransformer(log).fit_transform(iris.data)
print('对数变换后的数据：\n',df)
```

4. 独热编码

独热编码(One-Hot-Coding)又称为 One-Hot 编码，其方法是使用 N 位数值对 N 个状态进行编码，一位代表一种状态。该状态所在的位为 1，其他位都为 0。

例如，性别特征取值['男','女'](这里有两个特征，所以 N=2)，编码为男=> 10，女=> 01。又如，国家特征取值['中国','美国','法国'](这里有 3 个特征，所以 N=3)，编码为中国=> 100，美国=> 010，法国=> 001。再如，运动特征取值['足球','篮球','羽毛球','乒乓球'](这里有 4 个特征，所以 N=4)，编码为足球=> 1000，篮球=> 0100，羽毛球=> 0010，乒乓球=> 0001。

所以，当一个样本为 ['男','中国','乒乓球'] 的时候，完整的独热编码结果为[1,0,1,0,0,0,0,0,1]。

sklearn 中有封装好的独热编码函数 OneHotEncoder()。其常用形式如下：

```
OneHotEncoder(categories = 'auto',sparse = 'True',dtype = 'float')
```

参数说明：

(1) categories 默认为'auto'，用于根据训练数据自动确认类别，默认是数组的列表，categories[i]保存第 i 列中预期的类别。

(2) sparse 默认为 True。如果设置为 True，将返回稀疏矩阵，否则返回一个数组。

(3) dtype 默认为 float，用于设置所需的输出数据类型。

常用方法：

(1) fit(X)用于使 X 拟合 OneHotEncoder。

(2) fit_transform(X)用于使 X 拟合 OneHotEncoder，并且转换 X。

例 5.31　独热编码示例。

程序代码如下：

```
from sklearn.preprocessing import OneHotEncoder
data = [[0,0,3],[1,1,0],[0,2,1],[1,0,2]]
enc = OneHotEncoder(sparse = False)
enc.fit(data)
ans = enc.transform([[0,1,3]])   # 如果不指定 sparse = False,输出的是稀疏的存储格式,即索引加
                                 # 值的形式
print(ans)
```

程序运行结果如图 5.22 所示。

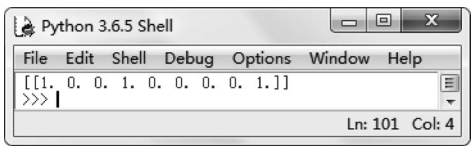


图 5.22　例 5.31 的运行结果

对于程序中给定的数组，把每一行当作一个样本，每一列当作一个特征。

先来看第一个特征，即第 1 列[0,1,0,1]，也就是说它有两个取值 0 和 1，那么 One-Hot 就会使用两位表示这个特征，[1,0]表示 0，[0,1]表示 1，在本例输出结果中的前两位[1,0,…]也就是表示该特征为 0；再看第二个特征，对应于第 2 列[0,1,2,0]，它有 3 种值，那么 One-Hot 就会使用 3 位来表示这个特征，[1,0,0]表示 0，[0,1,0]表示 1，[0,0,1]表示 2，在本例输出结果中的第 3 位到第 6 位[…,0,1,0,…]也就是表示该特征为 1；最后看第 3 个特征，对应于第 3 列[3,0,1,2]，它有 4 种值，那么 One-Hot 就会使用 4 位来表示这个特征，[1,0,0,0]表示 0，[0,1,0,0]表示 1，[0,0,1,0]表示 2，[0,0,0,1]表示 3，在本例输出结果中的最后 4 位[…,0,0,0,1]也就是表示该特征为 3。

5.6.2　数据离散化

数据离散化(Data Discretization)是指将连续的数据进行分段，使其变为一段段离散化的区间。因为在数据分析和统计的预处理阶段经常会碰到年龄、消费等连续型数值，而很多模型算法尤其是分类算法都要求数据是离散的，因此需要将数值进行离散化分段统计，提高数据区分度。例如年龄(1～150)是连续型特征，对于老、中、青的差异还需要通过数值层面才能理解，只有将年龄转换为离散型数据(例如老年人、中年人、青年人)才可以直观地表达出人们心中所想象的人群分类。连续数据的离散化方法如下：

1. 等宽法

等宽法(Equal-width Method)是将属性值分为具有相同宽度的区间，区间个数由数据本身的特点决定或由用户指定。比如属性值的区间为[0,60]，最小值为 0，最大值为 60，如果要将其分为 3 等份，则区间被划分为[0,20]、[21,40]、[41,60]，每个属性值对应属于它的那个区间。

使用 Pandas 的 cut()函数能够实现等宽法的离散化操作。其常用形式如下：

```
pd.cut(X,bins,right = True,labels = None,include_lowest = False,precision = 0)
```

参数说明：

(1) X 为一维数组，表示原始数据集。

(2) bins 为 int 类型，如果填入整数 n，则表示将 X 中的数据分成等宽的 n 份。

(3) right 为 boolean 类型，默认为 True，表示是否包含最右侧的数据。

(4) labels 接收 list、array，默认为 None，表示离散化后各个类别的名称。

(5) include_lowest 为 boolean 类型，默认为 False，表示不包含区间最左侧的数据。该参数表示是否包含最左侧的数据。

(6) precision 为 int 类型，表示精度，即表示区间值的小数位数，0 和 1 结果相同，默认值为 3。

例 5.32 利用等宽法进行数据离散化示例。

随机产生 200 人的年龄数据，然后通过等宽法离散化，并进行可视化。

程序代码如下：

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
def cluster_plot(d,k):
    plt.rcParams['font.sans - serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.figure(figsize = (12,4))
    for j in range(0,k):
        plt.plot(data[d == j],[j for i in d[d == j]],'o')
    plt.ylim( - 0.5,k - 0.5)
    return plt
data = np.random.randint(1,100,200)
k = 5                                    #分为 5 个等宽区间
d1 = pd.cut(data,k,labels = range(k))    #等宽离散
cluster_plot(d1,k).show()
```

程序运行结果如图 5.23 所示。

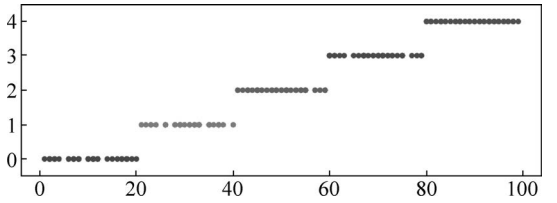


图 5.23 例 5.32 的运行结果

2. 等频法

等频法(Equal-frequency Method)是将相同数量的记录放在每个区间，保证每个区间的数量基本一致。它是将属性值分为具有相同宽度的区间，区间的个数 k 根据实际情况来决定。比如有 60 个样本，将其分为 3 部分，则每部分的长度为 20 个样本。这种方法的优点是数据变为均匀分布，缺点是会更改原有的数据结构。

用户也可以利用 Pandas 的 cut()函数进行等频离散化操作。

例 5.33 利用等频法进行数据离散化示例。

程序代码如下：

```
def cluster_plot(d,k):
    import matplotlib.pyplot as plt
    plt.figure(figsize = (12,4))
    for j in range(0,k):
        plt.plot(data[d == j],[j for i in d[d == j]],'o')
    plt.ylim( - 0.5,k - 0.5)
    return plt
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = np.random.randint(1,100,200)
data = pd.Series(data)
k = 6
w = [1.0 * i/k for i in range(k + 1)]
w = data.describe(percentiles = w)[4:4 + k + 1]      # 使用 describe()函数启动计算分位数
w[0] = w[0] * (1 - 1e - 10)
d4 = pd.cut(data,w,labels = range(k))                # 等频离散化
cluster_plot(d4,k).show()
```

程序运行结果如图 5.24 所示。

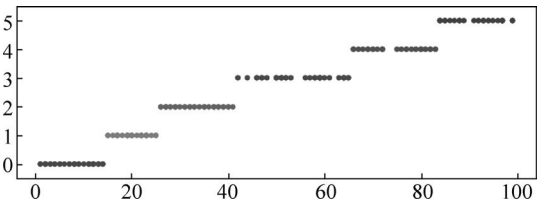


图 5.24　例 5.33 的运行结果

3. 聚类法

利用聚类法离散化包括两个过程：选取聚类算法(例如 k-means 算法)将连续属性值进行聚类；处理聚类之后得到的 k 个簇及每个簇对应的分类值(类似这个簇的标识)，将在同一个簇内的属性值作为统一标识。聚类分析的离散化需要用户指定簇的个数来确定产生的区间数。

4. 分位数法

固定宽度分箱非常容易计算，但如果计数值中有比较大的缺口，可能会产生很多没有任何数据的空箱子。根据数据的分布特点进行自适应的箱体定位，就可以解决这个问题。这种方法可以使用数据分布的分位数来实现数据的离散化(参见 4.3.2 节)。

5. ChiMerge 算法

ChiMerge 算法是利用卡方分箱的统计量对连续型变量进行离散化，该算法由 Kerber 提出。卡方分箱是典型的基于合并机制的自底向上离散化方法。它基于如下假设：如果两个相邻的区间具有非常类似的分布，则这两个区间可以合并，否则它们应当保持分开。此处衡量分布相似性的指标就是卡方值，卡方值越低，类分布的相似度越高。

卡方检验的主要思想是把所有的连续数据排序并分成多个区间，每次计算相邻两个区间的卡方值(Chi)，然后取卡方值最小的两个区间进行合并。

卡方值的计算公式：

$$\mathrm{chi}=\sum_{i=1}^{2}\sum_{j=1}^{k}\frac{(A_{ij}-E_{ij})^2}{E_{ij}} \tag{5-12}$$

其中，$E_{ij}=R_i\times\frac{C_j}{N}$，$R_i=\sum_{j=1}^{k}A_{ij}$(在 i 区间的 j 类别数)，$C_j=\sum_{i=1}^{2}A_{ij}$(j 类别样本个数)，$N=\sum_{i=1}^{2}R_i$(样本总数)。

下面通过一个简单的例子说明卡方值的计算。表 5.6 为特征 A_{ij} 观察值列表。

表 5.6　特征 A_{ij} 观察值列表

区　　间	类别 1	类别 2	类别 3	i 行的总数
[4.3,4.5]	1	0	0	1
[4.6,4.9]	0	1	2	3
j 列的总数	1	1	2	4

表 5.7 是根据表 5.6 计算出的期望值列表，计算方式为 i 行的数量乘以 j 列的数量除以总的数量。

表 5.7 期望值列表

区　间	类别 1	类别 2	类别 3
[4.3,4.5]	1×1/4=0.25	1×1/4=0.25	1×2/4=0.5
[4.6,4.9]	3×1/4=0.75	3×1/4=0.75	3×2/4=1.5

根据以上两个表计算这两个区间的卡方值：

$$\text{chi}=\frac{(1-0.25)^2}{0.25}+\frac{(0-0.25)^2}{0.25}+\frac{(0-0.5)^2}{0.5}+\frac{(1-0.75)^2}{0.75}+\frac{(1-0.75)^2}{0.75}+\frac{(1-1.5)^2}{1.5}$$
$$=2.25+0.25+0.5+0.08+0.08+0.17=3.33$$

ChiMerge 算法是一种基于卡方值的自下而上的离散化方法。

ChiMerge 算法的过程描述如下：

step1 初始化。根据要离散的属性对样本进行排序，每个样本属于一个区间。

step2 合并区间。合并区间又包括两个步骤：

step2-1 计算每一对相邻区间的卡方值。

step2-2 将卡方值最小的一对区间合并。

卡方值的阈值的确定：先选择显著性水平，再由公式得到对应的卡方值。得到卡方值需要指定自由度，自由度比类别数量小 1。例如有 3 类，自由度为 2，则 90%置信度(10%显著性水平)下卡方的值为 4.6(阈值)。阈值的意义在于，当类别和属性独立时有 90%的可能性计算得到的卡方值会小于 4.6，这样大于阈值的卡方值就说明属性和类不是相互独立的，不能合并。如果阈值选得较大，区间合并就会进行很多次，离散后的区间数量少、区间大。在一般情况下可以不考虑阈值，此时可以考虑最小区间数和最大区间数两个参数，只需指定区间数量的上限和下限，即最多几个区间、最少几个区间。

扫一扫

自测题

ChiMerge 算法推荐使用的置信度为 0.90、0.95、0.99，最大区间数取 10～15。

习题 5

5-1 选择题：

(1) 将原始数据进行集成、变换、维度归约、数值归约是在以下(　　)步骤的任务。

A. 频繁模式挖掘　　B. 分类和预测

C. 数据预处理　　D. 数据流挖掘

(2) 下面不属于数据预处理方法的是(　　)。

A. 变量代换　　B. 离散化　　C. 聚集　　D. 估计遗漏值

(3) 数据采集阶段最棘手的问题是增量同步，在无法掌控数据源的情况下通常有 3 种选择，不包括(　　)。

A. 放弃同步，采用直连形式

B. 放弃增量同步，选用全量同步

C. 具体业务需要进行增量同步

D. 扫描数据源以获得 delta 数据，然后针对 delta 数据进行增量同步

(4) (　　)反映数据的精细化程度，越细化的数据价值越高。

A. 规模　　B. 活性　　C. 关联度　　D. 颗粒度

(5) 数据清洗的方法不包括(　　)。

A. 缺失值处理　　B. 噪声数据清除　　C. 一致性检查　　D. 重复数据处理

(6) 在大数据时代使用数据的关键是(　　)。

A. 数据收集　　B. 数据存储　　C. 数据分析　　D. 数据再利用

(7) 假设有 12 个销售价格的记录组已经排序为 5,10,11,13,15,35,50,55,72,92,204,215,在使用等频(等深)划分方法将它们划分成 4 个箱子时,15 在第(　　)个箱子内。

A. 1　　B. 2　　C. 3　　D. 4

(8) 在上题中,等宽划分时(宽度为 50)15 在第(　　)个箱子内。

A. 1　　B. 2　　C. 3　　D. 4

(9) 采样分析的精确度随着采样随机性的增加而(　　),但与采样数量的增加关系不大。

A. 降低　　B. 不变　　C. 提高　　D. 无关

(10) 属性子集选择是一种维归约方法,它不需要检测和删除下列(　　)属性。

A. 不相关　　B. 相关　　C. 弱相关　　D. 冗余

5-2　填空题:

(1) 为了更高效地完成数据采集,通常需要将整个流程切分成多个阶段,在细分的阶段中可以采用(　　)执行的方式。

(2) 平均值修正法是指当数据的样本量很小时,可用前后两个观测值的(　　)来修正该异常值。

(3) 维归约的思路是减少所考虑的随机变量或属性的个数,使用的方法有属性子集选择、小波变换和(　　)等。

(4) 特征归约是从原有的特征中(　　)不重要或不相关的特征,或者通过对特征进行重组来减少特征的个数。

(5) 样本归约就是从数据集中选出一个有代表性的(　　)子集。

(6) 数据变换的目的是对数据进行(　　)处理,以便于后续的信息挖掘。

5-3　为什么要进行数据预处理?

5-4　为什么要对连续数据进行离散化处理?

5-5　假设两个数据集如表 5.8 和表 5.9 所示。

表 5.8　数据集 A

order	key	data1
0	b	0.3
1	b	1.5
2	a	1.2
3	c	2.3
4	a	0.4

表 5.9　数据集 B

order	key	data2
0	a	3.0
1	b	1.3
2	d	2.5

编程实现:

(1) 将两个数据集进行 inner(内)连接。

(2) 将两个数据集进行 left(左)连接。

(3) 将两个数据集进行 outer(外)连接。

5-6　利用随机函数 randint()生成 1～30 中整数的 10 个样本,5 个特征的数据集,利用 sklearn 模块中的 PCA()函数进行降维(降至二维)。

5-7　Python 编程实现在 3×2 数组[[0,1],[2,3],[4,5]]上进行多项式数据变换。