第5章

数据预处理

【学习目标】

学完本章之后，读者将掌握以下内容。

- 数据预处理的必要性和主要步骤。
- 基于 Python 的重复值、缺失值和噪声的检测与处理。
- 基于 Python 的数据列冗余与数据值冲突的判断与处理。
- 基于 Python 的属性子集选择和抽样方法。
- 基于 Python 的数据合并、抽取和计算的数据变换方法。

5.1 数据预处理的必要性

数据分析依赖于数据质量。低质量的数据将导致低质量的分析结果。然而，真实世界中的数据通常会存在大量脏数据，即数据通常不完整（如属性值空缺等）、不一致（如相同数据类型或值等不一致等）、受噪声（如存在偏离期望的孤立点）侵扰。数据预处理是数据分析、知识发现的重要过程。数据预处理是指在主要的处理以前对数据进行的一些处理，将原始数据转换为可以理解或者适合分析的样式，改进数据质量，提高分析的精度，为数据分析做铺垫。

数据质量涉及许多因素，如准确性、完整性和一致性是数据质量的三个基本要素。

导致数据不正确或不准确的原因有多种。收集数据的设备可能出故障；在数据输入时出现人或计算机的错误；当用户不希望提交个人信息时，可能故意向强制输入字段输入不正确的值。错误也可能在数据传输中出现。不正确的数据可能是由命名约定或所用的数据代码不一致，或输入字段的格式不一致而导致的。例如表 5-1，同样是 2021 年 1 月 5 日，可以有很多种时间格式。重复元组也需要数据清理。

表 5-1 不同时间表示格式

5th January,2021	2021-01-05	Jan 5, 2021	01/05/2021	2021.01.05	2021/01/05

导致数据不完整的原因也有多种。有些感兴趣的属性并非总能得到,如学生原生态家庭的精神情况或病史;输入时认为信息不重要而未收集;由于理解错误,或者因为设备故障而导致的相关数据没有记录;由于与其他记录不一致,而造成的数据删除等。此外,历史或修改的数据可能被忽略。缺失的数据,特别是某些属性列上具有缺失值的记录,可能需要将其缺失值推导出来并补齐。

数据不一致性,是指各类数据的矛盾性、不相容性。数据不一致性的原因也有多种。数据冗余,重复存放的数据未能进行一致性地更新。例如,学生入伍造成的学习状态调整,即学生处的学生状态已经改为休学或退学,而教务处未做相应更改,产生矛盾的就读状态。此外,如果软硬件出现故障或者操作错误导致数据丢失或数据损坏,也将引起数据不一致。

除了上述三个基本要素外,数据时效性(Timeliness)、可信性(Believability)、可解释性(Interpretability)也同样影响数据质量。

一般来说,数据预处理的主要步骤包括数据清洗、数据集成、数据规约和数据变换,如图 5-1 所示。

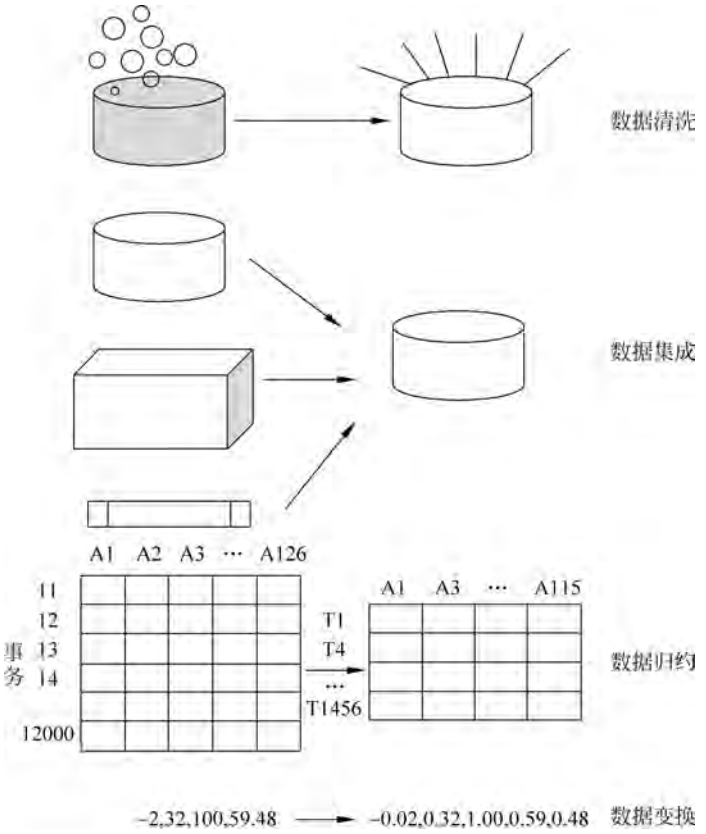

图 5-1 数据预处理步骤

- 数据清洗主要填写缺失值，光滑噪声，识别和处理离群点，解决不一致性以“清理数据”。
- 数据集成是把不同来源、格式、特点性质的数据在逻辑上或物理上有机地集中，从而更容易进行数据分析。
- 数据归约是指在尽可能保持数据原貌的前提下，最大限度地精简数据量，用替代的、较小的数据表示形式替换原数据，得到信息内容的损失最小化。
- 数据变换是对数据进行规范化处理，将数据转换成“适当的”形式，以适用分析任务及算法的需要。

这四个大步骤在做数据预处理时未必都要执行，也不是互斥的。例如，冗余数据的删除既是一种数据清洗形式，也是一种数据归约。

5.2 数据清洗

在数据分析时，海量的原始数据中存在大量不一致、不完整、有噪声的数据，严重影响数据分析的结果。在脏数据之上即使使用最好的分析方法，也将产生错误结果，并误导业务本身。因此在数据分析过程中，数据清洗尤为重要并占据了很大的工作量。数据清洗又叫数据清理或数据净化，是数据分析的第一步。本节介绍的主要内容包括重复值、缺失值和噪声检测与处理。

5.2.1 重复值检测与处理

在实际的数据采集、处理和分析中，经常会遇到重复数据。重复数据的产生可能是由于记录时的错误操作，也可能是真实存在的重复记录。重复数据在进行数据分析的过程中，对输出结果有重要的影响。例如，在逻辑回归分析中，重复数据会影响模型的拟合优度。需要说明的是，重复数据的处理也是选择性的，并不是所有情况下都要做。

以下主要介绍基于 pandas 库中的函数进行重复值的识别与处理。

1. 重复值检测

pandas 库中的 duplicated()函数可以实现查找并显示数据表中的重复值。此函数返回一个布尔型的 Series，显示是否有重复行，没有重复的行显示为 FALSE。其语法格式为：

```
duplicated(subset = None, keep = 'first')
```

其参数描述如表 5-2 所示。

表 5-2 duplicated()函数主要参数描述

参数	描 述
subset	列标签或标签序列，可选。仅对某些列进行重复项标识，默认情况下使用所有列
keep	查找重复值的模式。 有三个不同的值，默认值为“first”： keep='first'：除了第一次出现外，其余相同的数据被标记为重复，默认值。 keep='last'：除了最后一次出现外，其余相同的数据被标记为重复。 keep=False：所有相同的数据都被标记为重复

指定 subset 参数可控制检测重复行的粒度。当 subset 不指定时,检测数据表中记录行是否重复,即当两条记录中所有列数据都相等时才判断为重复行;当 subset 指定了列标签或列标签序列时,则只在指定列或列的组合上的所有数据重复才被判断为重复行,其余未指定列不检测。

2. 删除重复值

使用 drop_duplicates()函数可实现重复值删除。此函数返回一个移除了重复行的数据框对象 DataFrame。其语法格式为:

```
drop_duplicates (subset = None, keep = 'first', inplace = False, ignore_index = False)
```

其中参数描述如表 5-3 所示。

表 5-3 drop_duplicates()函数的主要参数描述

参数	描 述
subset	列标签,默认为 None,去除重复项时要考虑的标签。当 subset=None 时所有列都相同才认为是重复项
keep	表示是否保留。默认为"first"。 keep='first':去重时每组重复数据保留第一条数据,其余数据丢弃。 keep='last':去重时每组重复数据保留最后一条数据,其余数据丢弃。 keep=False:去重时每组重复数据全部丢弃,不保留
inplace	布尔值,表示直接在原来数据上修改还是保留一个副本,默认为 False。 inplace=False:去重之后不覆盖原表格数据。 inplace =True:去重之后原表格数据被覆盖

例 5.1 学生信息数据"学生数据.xlsx"中有四列信息,即姓名,性别,出生日期,学号,其数据显示如图 5-2 所示。检查是否存在学生信息重复录入的情况。若存在,则将重复信息删除。

	A	B	C	D
1	name	gender	birth	number
2	张三	女	1993.4.12	20161601
3	李四	男	1992.2.15	20161602
4	李明	男	1994.3.21	20161603
5	王梅	女	1994.5.24	20161604
6	张强	男	1996.3.23	20161605
7	周星星	男	1998.3.24	20161606
8	张三	女	1993.4.12	20161601
9	张强	男	1996.3.23	20161605
10				

图 5-2 学生数据显示

```
1:  from pandas import read_excel
2:  df = read_excel(r'C:/case/学生数据.xlsx')
3:  print('数据集是否存在重复观测: ', df.duplicated())
```

```
4:  newdf = df.drop_duplicates()
5:  print(newdf)
```

【例题解析】

上述代码是对重复值进行识别与删除。

第 1 行从 pandas 库中导入 read_excel()函数。第 2 行通过 read_excel()函数将学生信息数据读入数据框 df。第 3 行使用 duplicated()函数检测数据中是否存在重复值，默认 keep='First'。因此，数据在第一次出现时(即第 0～5 行)显示为 False，但是在第 6 行和第 7 行再次出现时，被标记为了 True，即重复行。第 4 行使用 drop_duplicates()函数将出现重复值的行删除并赋值给新的数据框对象 newdf。第 5 行打印并显示 newdf 中数据。

【运行结果】

第 3 行的输出结果：

```
数据集是否存在重复观测：0    False
                      1    False
                      2    False
                      3    False
                      4    False
                      5    False
                      6    True
                      7    True
                      dtype：bool
```

第 5 行的输出结果：

	name	gender	birth	number
0	张三	女	1993.4.12	20161601
1	李四	男	1992.2.15	20161602
2	李明	男	1994.3.21	20161603
3	王梅	女	1994.5.24	20161604
4	张强	男	1996.3.23	20161605
5	周星星	男	1998.3.24	20161606

5.2.2 缺失值检测与处理

除了重复值之外，真实世界中的数据也存在普遍的数据缺失现象。数据具有缺失值，并不意味着数据有错误。数据缺失的原因有很多，例如，由于工作人员的疏忽，造成无意的数据缺失；或者由于数据采集器故障等原因造成的缺失；本身数据不存在造成的数据缺失，比如一个未婚者的配偶名字、孩子的收入状况等。明确了缺失值来源，才能对症下药。

缺失值的检测可以使用 isnull()判定。isnull()函数无参，返回一个布尔值，若该处值缺失，返回 True，否则返回 False。

常见的缺失值处理方法有直接删除,数据填补以及不进行任何处理。

1. 删除含有缺失值的行或列

dropna()函数可去除数据中值为空的数据行或列,其语法格式为:

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

其中主要参数描述如表 5-4 所示。

表 5-4 dropna()函数主要参数描述

参 数	描 述
axis	axis=0(默认值):删除包含缺失值(NaN)的行。 axis=1:删除包含缺失值(NaN)的列
how	how='any'(默认值):有缺失值(NaN)即删除。 how='all':所有的值都缺失(NaN)才删除
thresh	如果非缺失值(NaN)的数量大于 thresh 则保留
subset	定义要在哪些列中查找缺失值
inplace	是否直接在原 DataFrame 上修改

例 5.2 如图 5-3 的学生成绩记录数据包含九列,分别为:学年,学期,考试科目,考试性质,学分,成绩,班号,学号,备注。其中"备注"是为记录学生异常状态的预留列,如"休学"、"退学"等。请检查数据集中是否存在缺失,若列中缺失值大于 100,直接将此列删除;若存在空行也进行相应删除。

	学年	学期	考试科目	考试性质	学分	成绩	班级	学号	备注
2	2015	秋	体育 I	正常考试	1.00000	85.00000	B15	20151803	
3	2015	秋	体育 I	正常考试	1.00000	70.00000	B15	20151795	
4	2015	秋	体育 I	正常考试	1.00000	80.00000	B15	20151819	
5	2015	秋	体育 I	正常考试	1.00000	70.00000	B15	20151809	

图 5-3 学生成绩记录.xlsx 部分文件显示

```
1:  from openpyxl import load_workbook
2:  import pandas as pd
3:  records = load_workbook(r'C:/case/学生成绩记录.xlsx')
4:  ws = records.active
5:  all_value = []
6:  for row in ws.values:
7:      all_value.append(row)
8:  records_1 = pd.DataFrame(all_value)
9:  records_2 = records_1.dropna(axis = 1,thresh = 100)
10: print(records_2)
11: records_3 = records_1.dropna(how = "all")
12: print(records_3)
13: records_3.to_excel(r'C:/case/学生成绩记录(去缺失值).xlsx',header = False, index =
False)
```

【例题解析】

第1～7行使用例4.12中的load_workbook()函数读取数据。第8行将数据存入DataFrame，并赋值给records_1。第9行利用dropna()函数检测在列的方向上若出现100个以上NaN则将此列删除，并将删除后的数据赋值给records_2。其中，axis=1表示检查列值，thresh=100定义将缺失值超过100则删除。第10行输出records_2。因此，在运行结果中，第10行输出结果删除了“备注”列。

第11行dropna()函数删除行全空的值，即在行的方向上所有值均缺失(NaN)则将此行删除。删除出现空值的行并将其结果赋值给records_3。第12行输出此操作结果。因此，运行结果中在records_2的基础上删除空行剩余7923行。第13行将修改后的数据存入学生成绩记录(去缺失值).xlsx中。

【运行结果】

第10行输出结果(截取一部分显示)：

	0	1	2	3	4	5	6	7
0	学年	学期	考试科目	考试性质	学分	成绩	班级	学号
1	2015	秋	体育Ⅰ	正常考试	1	85	B15	20151803
2	2015	秋	体育Ⅰ	正常考试	1	70	B15	20151795
3	2015	秋	体育Ⅰ	正常考试	1	80	B15	20151819
4	2015	秋	体育Ⅰ	正常考试	1	70	B15	20151809
	…	…	…	…	…	…	…	…
7921	2023	春	毕业实习	正常考试	6	100	B19	20192365
7922	2023	春	毕业设计	正常考试	11	79.9	B19	20192365
7923	None	None	None	None	None	None	None	None
7924	None	None	None	None	None	None	None	None
7925	None	None	None	None	None	None	None	None

[7926 rows x 8 columns]

第12行输出结果(截取一部分显示)：

	0	1	2	3	4	5	6	7
0	学年	学期	考试科目	考试性质	学分	成绩	班级	学号
1	2015	秋	体育Ⅰ	正常考试	1	85	B15	20151803
2	2015	秋	体育Ⅰ	正常考试	1	70	B15	20151795
3	2015	秋	体育Ⅰ	正常考试	1	80	B15	20151819
4	2015	秋	体育Ⅰ	正常考试	1	70	B15	20151809
	…	…	…	…	…	…	…	…
7918	2023	春	毕业实习	正常考试	6	89.9	B19	20192341
7919	2023	春	毕业设计	正常考试	11	79.9	B19	20192371
7920	2023	春	毕业实习	正常考试	6	79.9	B19	20192371
7921	2023	春	毕业实习	正常考试	6	100	B19	20192365
7922	2023	春	毕业设计	正常考试	11	79.9	B19	20192365

[7923 rows x 8 columns]

将包含缺失值的记录直接删除的方法简单,在数据量非常大且缺失值不多的情况下有效。然而,这种通过减少历史数据换取完整信息的方式,可能造成很多隐藏的重要信息丢失;当缺失数据比例较大,特别是缺失数据非随机分布时,直接删除可能会导致数据分布特征的偏离。特别地,当样本量本身不大且缺失很多时,不建议使用直接删除。

2. 数据填补

常用的缺失值填补方法包括四种。

第一种,人工填写。一般来说,这种方法非常费时。当数据集很大、缺少值很多时,该方法可能行不通。

第二种,使用一个全局常量填充。将空缺的属性值用一个常数替换,尽管该方法简单,但是容易让分析过程误以为形成了一个有趣的概念和模式,因此并不推荐使用。

第三种,使用数据列的中心度量(如均值、中位数或众数)填充。对于非数值数据,使用众数(mode)或中位数填补;对于数值型数据,使用平均数(mean)或中位数(median)填补。一般地,如果特征分布为正态分布时,使用平均值效果比较好,而当分布由于异常值存在而不是正态分布的情况下,使用中位数效果比较好。

第四种,使用最优可能的值填充。可以用回归、贝叶斯形式化方法等基于推理的工具或决策树归纳确定。

pandas 库提供了 fillna()函数实现数据的填充。其语法格式为:

```
df.fillna(value = None,method = None,axis = None,inplace = False,limit = None, **kwargs)
```

其中的常用参数描述如表 5-5 所示。

表 5-5 fillna()函数参数描述

参数	描 述
value	用于填充缺失值,或者指定为每个索引或列使用 Serise/DataFrame 的值
inplace	inplace=True:直接修改原对象。 inplace=False:创建一个副本,修改副本,原对象不变(默认)
method	method=pad/ffill:用前一个非缺失值去填充该缺失值。 method=backfill/bfill:用后一个非缺失值填充该缺失值。 Method=None:指定一个值去替换缺失值(默认为这种方式)
limit	限制填充个数
axis	修改填充方向,0 代表行,1 代表列

例 5.3 接例 5.2 中名为“学生成绩记录(去缺失值).xlsx”的数据。提取考试科目为“体育Ⅱ”的数据,将其成绩使用此类课程成绩的平均数填充缺失值。

```
1:  import pandas as pd
2:  records = pd.read_excel(r'C:/case/学生成绩记录(去缺失值).xlsx')
3:  records_2 = records[records['考试科目'] == "体育Ⅱ"]
```

```
4:  records_2['成绩'] = records_2['成绩'].astype(float).fillna(records_2['成绩'].mean())
5:  print(records_2)
6:  records_2.to_excel(r'C:/case/体育成绩记录(填缺失值).xlsx',index = False)
```

【例题解析】

第 1 行引入 pandas 库。第 2 行使用 read_excel()函数读取学生成绩记录数据,括号内为文件路径。第 3 行提取考试科目为“体育Ⅱ”的数据,命名为 records_2。第 4 行先将数据成绩一列的数据类型转换为 float 型,然后将成绩列中空值使用此列的平均数进行填充。第 5 行打印填充后的数据值。第 6 行将修改后的数据存入体育成绩记录(填缺失值).xlsx 中。

【运行结果】

	学年	学期	考试科目	考试性质	学分	成绩	班级	学号
467	2016	春	体育Ⅱ	正常考试	1.0	96.0	B15	20151819
469	2016	春	体育Ⅱ	正常考试	1.0	91.0	B15	20151825
470	2016	春	体育Ⅱ	正常考试	1.0	93.0	B15	20151823
473	2016	春	体育Ⅱ	正常考试	1.0	80.0	B15	20151837
476	2016	春	体育Ⅱ	正常考试	1.0	90.0	B15	20151845
	…	…	…	…	…	…	…	…
6356	2020	春	体育Ⅱ	正常考试	1.0	69.0	B19	20192371
6366	2020	春	体育Ⅱ	正常考试	1.0	71.0	B19	20192391
6376	2020	春	体育Ⅱ	正常考试	1.0	67.0	B19	20192389
6391	2020	春	体育Ⅱ	正常考试	1.0	83.0	B19	20192341
6426	2020	春	体育Ⅱ	正常考试	1.0	73.0	B19	20192365

[111 rows x 8 columns]

3. 不处理

空值填补是用估计值填补未知值,不一定完全符合客观事实。在对不完备信息进行补齐处理的同时,或多或少地将改变原始信息。对空值不正确的填充也可能引入新的噪声,为分析带来错误的结果。因此,在某些情况下,希望在保持原始信息不发生变化的前提下对数据进行处理。

5.2.3 噪声检测与处理

噪声(Noise)是数据集中的干扰数据(对场景描述不准确的数据),即被测量变量的随机误差或方差。噪声数据中存在着错误或异常,这将对数据分析造成干扰。一般而言,观测值是数据真实值与噪声的叠加,因此噪声在数据集中很常见。噪声在数据分析(包括离群点分析)中不是令人感兴趣的,需要在数据预处理中剔除,减少对后续模型预估的影响。

常用的数据平滑去噪的技术有分箱(Binning)、回归(Regression)和离群点分析(Outlier analysis)。

1. 分箱

分箱方法通过考察数据的近邻(即周围的值)来光滑有序数据值。这些有序的值被分布到一些桶或箱中。由于分箱方法考察近邻的值,因此是数据的局部光滑。常用方法有3种,即按箱均值平滑、按中值平滑和按边界值平滑,如图5-4表示。

按箱均值平滑是用箱中的均值替换箱中每一个值。在图5-4的例子中,有9个成绩值,首先将成绩按大小排序,然后被划分到大小为3的等频箱中(即每个箱包含3个)。箱中的值被此箱中的均值替代。类似地,按中值平滑即箱中的每一个值都被替换为该箱的中位数。对于按边界值平滑,给定箱中的最大值和最小值同样被视为箱边界,而箱中的每一个值都被替换为最近的边界值。例如,在图5-4中的箱边界光滑,以箱1为例,边界值为60、77,而61距离60更近,则使用60代替。一般而言,宽度越大,光滑效果越明显。分箱也可以作为一种离散化技术使用。

成绩排序后的结果:60, 61, 77, 80, 85, 93, 95, 96, 100

划分为等频的箱:
箱1:60, 61, 77
箱2:80, 85, 93
箱3:95, 96, 100

用箱均值光滑:
箱1:66, 66, 66
箱2:86, 86, 86
箱3:97, 97, 97

用箱边界光滑:
箱1:60, 60, 77
箱2:80, 80, 93
箱3:95, 95, 100

图5-4 数据光滑的分箱方法

例5.4 接例5.2中的“体育成绩记录(填缺失值).xlsx”的数据。将成绩分为10个箱,并对成绩进行箱均值光滑和箱边界光滑处理。

```
import pandas as pd
import numpy as np
def binning(filename,box_num):
    my_list1 = []
    noise_data = pd.read_excel(filename)
    my_list1 = sorted(noise_data['成绩'])
    box_list = []
   len_box = int(np.ceil(len(my_list1)/float(box_num)))
   for i in range(0,10):
        each_box = my_list1[i * len_box:(i + 1) * len_box]
        box_list.append(each_box)
```

```
    return box_list
def box_mean_smooth(box_list):
    for i in range(0,len(box_list)):
        box_avg = int(np.average(box_list[i]))
        for j in range(0,len(box_list[i])):
            box_list[i][j] = box_avg
   return box_list
def box_boundary_smooth(box_list):
    for i in range(0,len(box_list)):
        left_bdy = box_list[i][0]
        right_bdy = box_list[i][ - 1]
        for j in range(0,len(box_list[i])):
            if abs(box_list[i][j] - left_bdy)< abs(box_list[i][j] - right_bdy):
                box_list[i][j] = left_bdy
            else:
                box_list[i][j] = right_bdy
    return box_list
filename = ''r'C:/case/体育成绩记录(填缺失值).xlsx'''
box_list = binning(filename,10)
print (box_list)
print (box_mean_smooth(box_list))
print (box_boundary_smooth(box_list))
```

【例题解析】

第 1 行表示引入 pandas 库；第 2 行引入 numpy 库。

第 3～12 行定义了等频分箱函数，其中参数 filename 表示读取文件名，参数 box_num 是分箱个数。等频分箱函数的基本思路是：先读取数据，并根据成绩列进行排序(第 5～6 行)；然后分箱，每箱的长度为总体数据长度÷箱数，根据每箱的长度对排序后的数据切分(第 8～12 行)。排序后的分箱成绩放入 box_list 中返回。

第 13～18 行定义箱均值光滑函数，参数 box_list 列表存放分箱数值。定义箱均值光滑函数的基本思路是：对于 box_list 中每分箱数据 box_list[i]利用 np.average 函数求均值(第 14～15 行)；然后，将每分箱 box_list[i]中数据用该箱均值替代。针对分箱中的每一个分箱数据 box_list[i]，采用 np.average 函数求出该分箱均值并赋值给 box_avg(第 16～17 行)。

第 19～28 行定义箱边界光滑函数，参数 box_list 列表存放分箱数值。定义箱边界光滑函数是基本思路是：首先，找到每个分箱数据 box_list[i]的边界值(第 20～22 行)，即最大值 right_bdy 和最小值 left_bdy；然后，计算箱中的每个数值距离哪个边界值较近，则使用较近的边界值替代(第 23～27 行)。

第 30～33 行调用分箱函数对数据进行分箱处理，打印结果。

【运行结果】

[[0.0，49.0，50.0，50.0，50.0，50.0，52.0，60.0，60.0，60.0，60.0，60.0]，…，[95.0，96.0，96.0]]

[[50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50], … , [95, 95, 95]]
[[50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50], … , [95, 95, 95]]

2. 回归

也可以用一个函数拟合数据来光滑数据,这种技术称为回归。线性回归涉及找出拟合两个属性(或变量)的最佳直线,使得一个属性可以用来预测另一个。多元线性回归是线性回归的扩充,其中涉及的属性多于两个,并且数据拟合到一个多维曲面。回归将在第6章进一步讨论。

3. 离群点分析

与噪声容易混淆的概念是离群点。离群点也称为异常值,是那些远离绝大多数样本点的特殊群体。离群点跟噪声数据不一样。离群点本身属于观测数据,通常这样的数据点在数据集中表现出不合理的特性,离群点相对噪声来讲比较罕见。

离群点有可能影响数据分析的结果和结论。因此,在数据的探索过程中,常常需要进行离群点分析,或者需要在报告结论中对离群点进行特别讨论。常用的离群点检测主要有 3σ 原则和画图分析(例如,箱线图或散点图),也可以通过如聚类来检测离群点。

这里主要介绍 3σ 原则。3σ 原则,又叫拉依达原则,是指假设一组检测数据中只含有随机误差,需要对其进行计算得到标准偏差,按一定概率确定一个区间,对于超过这个区间的误差,就不属于随机误差,需要将含有该误差的数据进行剔除。3σ 原则下数据的数值分布几乎全部集中在区间$(\mu-3\sigma,\mu+3\sigma)$内,超出这个范围的数据仅占不到0.3%。故根据小概率原理,可以认为超出 3σ 的部分数据为异常数据。

例 5.5 利用 3σ 原则,检查"学生成绩记录.xlsx"数据中是否存在异常值并打印。

```
 1:  import pandas as pd
 2:  import numpy as np
 3:  from pandas import read_excel
 4:  import matplotlib.pyplot as plt
 5:  df = read_excel(r'C:/case/学生成绩记录.xlsx')
 6:  ymean = np.mean(df['成绩'])
 7:  ystd = np.std(df['成绩'])
 8:  threshold1 = ymean - 3 * ystd
 9:  threshold2 = ymean + 3 * ystd
10:  outlier = []
11:  for i in range(0, len(df['成绩'])):
12:      if (df.成绩[i] < threshold1)|(df.成绩[i] > threshold2):
13:          outlier.append(df.成绩[i])
14:      else:
15:          continue
16:  print(outlier)
```

【例题解析】

第1～4行引入 pandas 库、numpy 库和 read_excel 函数。

第5～9行计算获得($\mu-3\sigma$,$\mu+3\sigma$)的区间。其中,第5行读入数据。第6行求出成绩的平均值。第7行求出成绩数据值的标准差,第8行和第9行分别计算 $\mu-3\sigma$ 和 $\mu+3\sigma$。

第10～16行是将异常值保存在 outlier 中,其中,第10行定义列表意在将异常值放入。第11～15行检测成绩列中数值在($\mu-3\sigma$,$\mu+3\sigma$)之外的成绩值,放入 outlier。第16行将异常值打印。

从结果发现,异常值包括两种。一种是如1000这样的超高值,成绩最高为100,显然1000超过了成绩的范围,另一种是偏低的数值,即29以下的成绩,此类学生偏离平均水平较远,建议采取取消补考资格直接重修。

【运行结果】

[22.0,1108.0,19.0,13.0,20.0,0.0,0.0,0.0,0.0,11.0,12.0,11.0,0.0,0.0,0.0,6.0,…,9.0,12.0,21.0,22.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]

5.3 数据集成

在很多应用场合,分析需要合并来自多个不同来源的数据。由于不同的数据源定义表名和列名时命名规则不同,存入的数据格式、取值方式、单位都会有不同。因此,小心集成有助于减少结果数据集的冗余和不一致,提高分析准确性。数据集成的本质是整合数据源,因此多个数据源中字段的语义差异、结构差异、字段间的关联关系,以及数据的冗余重复,都是数据集成面临的问题。

5.3.1 实体识别问题

数据集成将多个数据源中的数据合并,存放在一个一致的数据存储中。这些数据源可能包括多个数据库或一般文件。在数据集成时,有许多问题需要考虑,如模式集成和对象匹配。来自多个信息源的现实世界的等价实体如何才能“匹配”?这涉及实体识别问题。例如,数据分析者要确定一个数据库中的 student_id 与另一数据库中的 stud_number 是同一个属性、指同一个实体。

集成时需要注意每个数据列的元数据包括名字、含义、数据类型和属性的允许取值范围,以及处理空白、零或空值规则等。在整合数据源的过程中,可能出现:

- 同列名但不同语义的情况,如两个数据源中都有一个列名字叫“成绩”,但其实一个数据源中记录的是未加平时成绩的考试成绩,另一个数据源中是加平时成绩、课堂表现等后的综合成绩。
- 不同列名但同语义的情况,如两个数据源都有数据列记录加平均成绩后的成绩,但是一个数据源中列名为 score,另一个数据源中列名为 grade。
- 同列名同语义但不同字段结构的情况,同样存储学生成绩字段,一个数据源存为 int,另一个数据源中存为 char。
- 字段取值范围不同,如学生成绩字段,一个数据源中是百分制,另一个数据源中是十分制等。

为了解决上述问题,需要在数据集成前,进行业务调研,确认每个字段的实际意义,不

被误导。另外,在集成期间,当一个数据集的数据列与另一个数据集的数据列匹配时,必须特别注意源系统中的函数依赖和参照约束与目标系统中的匹配。

5.3.2 数据列冗余问题

冗余是数据集成的另一个重要问题。一个数据列(例如,年收入)如果能由另一个或另一组数据列"导出",则这个数据列可能是冗余的。有些冗余可以被相关分析检测到。

给定两个数据列,根据可用数据度量一个列能在多大程度上蕴含另一个。对于类别数据,可以使用 χ^2(卡方)检验。对于数值型数据列,使用相关系数(Correlation Goefficient)和协方差(Govariance)。

1. 类别数据的 χ^2(卡方)检验

对于类别数据,两个数据列 A 和 B 之间的相关联系可以通过卡方检验发现。假设 A 有 c 个不同值 $a_1,a_2,\cdots,a_c$,B 有 r 个不同值 $b_1,b_2,\cdots,b_r$。用 A 和 B 描述的数据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。另 (A_i, B_j) 表示列 A 取值 a_i、列 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 Pearson χ^2 统计量),可以用公式(5-1)计算:

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

其中,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,可以用公式(5-2)计算:

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

其中,n 是数据集大小,$\text{count}(A=a_i)$ 是 A 上具有值 a_i 的个数,而 $\text{count}(B=b_j)$ 是 B 上具有值 b_j 的个数。式(5-1)中的和在所有 $r\times c$ 个单元上计算。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 $(r-1)\times(c-1)$。如果可以拒绝该假设,则说明 A 和 B 是统计相关的。

Python 的 Scipy 库中包含众多进行科学计算、统计分析的函数。Scipy 是世界上著名的 Python 开源科学计算库,建立在 NumPy 之上。可通过 Scipy 库中的 chi2_contingency()函数进行卡方检验,其语法格式为:

```
chi2_contingency(observed, correction = True, lambda_ = None)
```

其中常用参数描述如表 5-6 所示,返回值如表 5-7 所示。

表 5-6 chi2_contingency()函数常用参数描述

参数	描 述
observed	列联表,表包含每个类别中观察到的频率(即发生次数)。在二维情况下,表通常被描述为"R×C 表"

续表

参数	描　　述
correction	如果为 True，并且自由度为 1，则应用 Yates 校正以保持连续性。校正的效果是将每个观察值向相应的期望值调整 0.5
lambda_	float 或 str，可选。默认情况下，此测试中计算的统计量是 Pearson 的卡方统计量

表 5-7　chi2_contingency()函数返回值描述

参数	描　　述
chi2	float，卡方值
p	float，p 值
dof	int，自由度
expected	ndarray，预期频率，基于表的边际总和

例 5.6　假设存在如表 5-8 所示男女所读书目的类型统计。检验性别与阅读类别是否有关；设 H_0：性别与阅读类别无关，H_1：性别与阅读类别有关。

表 5-8　男女阅读种类数据

	男	女		男	女
小说	250	200	非小说	50	1000

```
1:  from scipy.stats import chi2_contingency
2:  import numpy as np
3:  kf_data = np.array([[250,200],[50,1000]])
4:  kf = chi2_contingency(kf_data)
5:  print('chisq-statistic=%.4f, p-value=%.4f, df=%i expected_frep=%s'%kf)
```

【例题解析】

第 1 行引入 scipy.stats 库的 chi2_contingency()函数。第 2 行引入 NumPy 库。

第 3 行以数组的形式写入数据，其中，表 5-8 中的数据以列(或行，无影响)为单位存入两个列表中。第 4 行使用 chi2_contingency()函数对数组数据进行卡方检验。第 5 行输出卡方值，p 值，自由度以及上述数组顺序的期望值。

因为其 p-value 近似于 0，因此拒绝两者独立的假设，即性别与阅读类别显著相关。

【运行结果】

chisq-statistic=504.7669，p-value=0.0000，df=1 expected_frep=[[90. 360.]
[210. 840.]]

2. 数值数据的相关系数

对于数值数据，可以通过计算数据列 A 和 B 的相关系数(又称 Pearson 积矩系数，Pearson's Product Moment Coefficient)，估计这两个数据列的相关度 $r_{A,B}$，

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

其中，n 是数据集大小，a_i 和 b_i 分别是第 i 行数据在列 A 和列 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，$\sum_{i=1}^{n}(a_ib_i)$ 是 AB 叉积和(即列 A 每一个值乘以列 B 对应位置的值)。严格地说，Pearson 的相关性要求每个数据集正态分布。与其他相关系数一样，此系数在 -1 和 $+1$ 之间变化($-1\leqslant r_{A,B}\leqslant+1$)，0 表示没有相关性。

如果 $r_{A,B}$ 大于 0，则 A 和 B 正相关，这意味着 A 值随着 B 值的增加而增加。该值越大，相关性越强(即每个属性蕴含另一个可能性越大)。因此，一个较高的 $r_{A,B}$ 值表明 A(或 B)可以作为冗余列。如果该结果值等于 0，则 A 和 B 是独立的，并且它们之间不存在相关性。如果该结果值小于 0，则 A 和 B 是负相关的，一个值随着另一个减少而增加。这意味着每一个属性列都阻止另一个出现。散点图也可以用来观察属性之间的相关性。

Stats 模块是 Scipy 的统计模块，其中包含很多用于统计检验的函数。使用 Stats 模块的 pearsonr()函数可计算皮尔逊相关系数和测试非相关性的 p 值，其语法格式为：

```
scipy.stats.pearsonr(x,y)
```

其中，x、y 为输入变量的数组，返回皮尔逊相关系数和测试非相关性的 p 值。

例 5.7 图 5-5 中记录了两个公司(ALLElect 和 Hightech)不同时刻的每支股票信息的单价信息，判断两支股票的相关性。

	A	B	C
1	时间点	ALLElect	Hightech
2	t1	6	20
3	t2	5	10
4	t3	4	14
5	t4	3	5
6	t5	2	5

图 5-5 股票.xlsx 部分数据显示

```
1:  import pandas as pd
2:  import scipy.stats as stats
3:  df = pd.read_excel(r'C:/case/股票.xlsx')
4:  print(stats.pearsonr(df["ALLElect"],df["Hightech"]))
```

【例题解析】

第 1～2 行引入 pandas 库与 Scipy 库的 Stats 模块。第 3 行读取数据。第 4 行打印数据中两列的相关系数以及 p 值。

因为结果中 ALLElect 与 Hightech 两列的相关系数为 0.867，$p=0.057$。因此，在 90%的置信水平上拒绝原假设，即两个公司股票呈现显著性相关。

【运行结果】

(0.8674427949190671, 0.05676876648986295)

3. 数值数据的协方差

在概率论和统计学中,协方差和方差是两个类似的度量,评估两个数据列如何一起变化。考虑两个数值列 A、B 和 n 次观测的集合$\{(a_1,b_1),\cdots,(a_n,b_n)\}$。$A$ 和 B 的均值又分别称为 A 和 B 的期望,即

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

且

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

A 和 B 的协方差(covariance)定义为

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

如果把公式(5-3)和公式(5-4)相比较,则可以看到

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

其中,σ_A 和 σ_B 分别是 A 和 B 的标准差。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列 A 和 B,如果 A 大于 $\overline{A}$,则 B 很可能大于 $\overline{B}$。因此,A 和 B 的协方差为正。另外,如果当一个属性小于它的期望值时,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(即它们不具有相关性),则 $E(A\cdot B)=E(A)\cdot E(B)$。因此,协方差 $\mathrm{cov}(A,B)=E(A\cdot B)-\overline{A}\overline{B}=E(A)\cdot E(B)-\overline{A}\overline{B}=0$。然而,其逆不成立。某些随机变量对(属性对)可能具有协方差 0,但不是独立的。仅在某种附加的假设下(如数据遵守多元正态分布),协方差 0 蕴含独立性。

Python 中使用 DataFrame.cov()函数计算协方差,其语法格式为

```
DataFrame.cov([min_periods])
```

该函数计算列的成对协方差,不包括 NA/null 值,其中,参数 min_periods 表示样本最少的数据量,返回值为表示协方差的 DataFrame 对象。

5.3.3 数据值冲突问题

数据集成还涉及数据值冲突的检测与处理。例如,对于现实世界的同一对象,来自不

同数据源的值可能不同。这可能是因为表示、尺度或编码不同。例如,同样是学生缴纳的学费列,数据类型均为数值型,但是一个数据源中使用逗号分隔,另一个数据源中用科学记数法。重量属性可能在一个系统中以公制单位存放,而在另一个系统中以英制单位存放。不同学校交换信息时,每个学校可能都有自己的课程计划和评分方案。一所大学可能采取学季制,开设3门数据库系统课程,用A+～F评分;而另一所大学可能采取学期制,开设两门数据库课程,用1～10评分。很难在这两所大学之间指定准确的课程成绩变化规则,这使得信息交换非常困难。

列也可能在不同的抽象层,其中列在一个系统中记录的抽象层可能比另一个系统中"相同的"属性低。例如,"籍贯"在一个数据库中可能填写的是城市,而另一个数据库中相同名字的列可能表示的是县或者省份等。

对待这种问题,需要对实际业务知识有一定的理解。同时,对数据进行调研,尽量明确造成冲突的原因,如果数据的冲突实在无法避免,就要考虑冲突数据是否都要保留,是否要进行取舍,如何取舍等问题。

5.4 数据规约

5.4.1 策略概述

用于数据分析的数据集可能非常大。但是,在海量数据集上进行复杂的数据分析可能需要很长的时间。数据规约产生更小但保持原数据完整性的数据集。在规约后的数据集上进行分析和挖掘将更有效率。

数据规约策略包括维规约、数量规约和数据压缩。

维规约(Dimensionality Reduction)减少所考虑的随机变量或属性的个数。维规约方法包括小波变换和主成分分析法,它们把原数据变换或投影到较小的空间。属性子集选择是一种维规约方法,其中不相关、弱相关或冗余的属性或维被检测和删除。

数量规约(Numerosity Reduction)用替代的、较小的数据表示形式替换原数据。这些技术可以是参数的或非参数的。对于参数方法而言,使用模型估计数据,使得一般只需要存放模型参数,而不是实际数据(离群点可能也要存放)。回归和对数-线性模型就是例子。存放数据规约表示的非参数方法包括直方图、聚类、抽样和数据立方体聚集。

数据压缩(Data Compression)使用变换,以便得到原数据的规约或"压缩"表示。如果原数据能够从压缩后的数据重构,而不损失信息,则该数据规约称为无损的。如果只能近似重构原数据,则该数据规约称为有损的。对于串压缩,有一些无损压缩算法。然而,它们一般只允许有限的数据操作。维规约和数量规约也可以视为某种形式的数据压缩。

5.4.2 属性子集选择

属性子集选择属于维规约方法中的一种。用于分析的数据集可能包含数以百计的属性,其中大部分属性可能与分析任务不相关,或者是冗余的。例如,如果分析任务是"学生选择'Python数据分析'这门课程的影响因素分析",与"专业"和"选修课"不同,诸如学生的电话号码等属性多半是不相关的。尽管领域专家可以挑选出有用的属性,但这可能是一项困难而费时的任务,特别是当数据的行为不是十分清楚的时候更是如此。遗漏相关属

性或留下不相关属性都可能是有害的，会导致所用的分析和挖掘方法无所适从。这可能导致发现质量很差的模式。此外，不相关或冗余的属性增加了数据量，可能会减慢分析进程。

属性子集选择通过删除不相关或冗余的属性（或维）减少数据量。属性子集选择的目标是找出属性最小属性集，使得数据内的概率分布尽可能地接近使用所有属性得到的原分布。在缩小的属性集上分析和挖掘还有其他的优点：它减少了出现在发现模式上的属性数目，使得模式更容易理解。

如何找出原属性的一个“好的”子集？对于 n 个属性，有 2^n 个可能的子集。穷举搜索找出属性的最佳子集可能是不现实的，特别是当 n 和数据集的数目增加时。因此，属性子集选择通常使用压缩搜索空间的启发式算法。通常，这些方法是典型的贪心算法，在搜索属性空间时，总是做看上去最佳的选择。他们的策略是做局部最优选择，期望由此导致全局最优解。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。这种检验假定属性是相互独立的。也可以使用一些其他属性评估度量，如建立分类决策树使用的信息增益度量。属性子集选择的基本启发式方法包括以下技术，如图 5-6 所示。

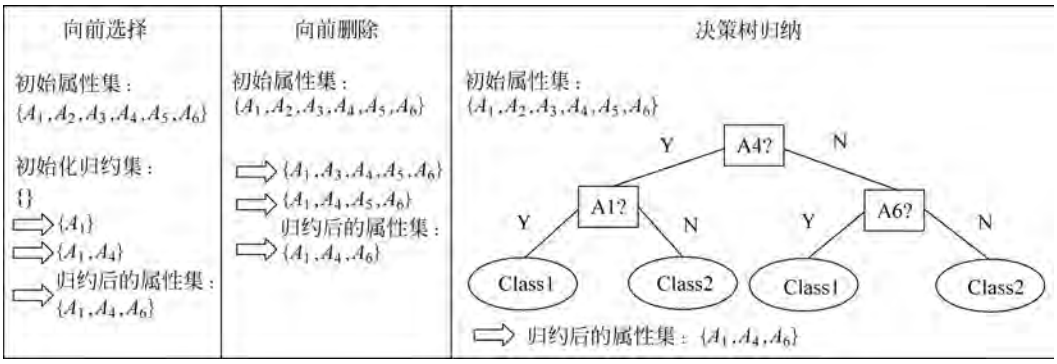

图 5-6 属性子集选择的贪心（启发式）方法

（1）逐步向前选择：该过程由空属性集作为归约集开始，确定原属性集中最好的属性，并将它添加到归约集中。在其后的每一次迭代，将剩下的原属性集中的最好属性添加到该集合中。

（2）逐步向后删除：该过程由整个属性集开始。在每一步中，删除尚在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：可以将逐步向前选择和逐步向后删除方法结合在一起，每一步选择一个最好的属性，并在剩余属性中删除一个最差的属性。

（4）决策树归纳：决策树算法（例如 ID3、C4.5 和 CART）最初是用于分类的。决策树归纳构造一个类似于流程图的结构，其中每个内部（非树叶）结点表示一个属性上的测试，每个分枝对应于测试的一个结果；每个外部（树叶）结点表示一个类预测。在每个结点上，算法选择“最好”的属性，将数据划分成类。

当决策树归纳用于属性子集选择时，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。该过程可以使用一个度量阈值来决定何时停止属性选择过程。

5.4.3 抽样

抽样可以作为一种数据归约技术使用,因为它允许用比数据小得多的随机样本(子集)表示大型数据集。假定大型数据集 D 包含 N 个元组。可以用于数据归约的、最常用的对 D 的抽样方法包括:

s 个样本的无放回简单随机抽样:从 D 中抽取 s 个样本,而且每次抽取一个样本,不放回数据集 D 中。

s 个样本的有放回简单随机抽样:该方法类似于无放回简单随机抽样,不同之处在于当一个样本从 D 中抽取后,记录它,然后放回原处。也就是说,一个样本被抽取后,它又被放回 D,以便它可以被再次抽取。

簇抽样:如果 D 中的样本被分组,放入 M 个互不相交的"簇",则可以得到 s 个簇的简单随机抽样(SRS),其中 $s<M$。例如,在空间数据库中,可以基于不同区域位置上的邻近程度定义簇。

分层抽样:如果 D 被划分成互不相交的部分,称作"层",则通过对每一层的 SRS 就可以得到 D 的分层抽样。特别是当数据倾斜时,这可以帮助确保样本的代表性。例如,可以得到关于顾客数据的一个分层抽样,其中,分层对顾客的每个年龄组创建。这样,具有的顾客人数最少的年龄组肯定能够被代表。

簇抽样与分层抽样的区别是:分层是为了保证得到的样本可以代表总体中的不同群体,并且每层中的样本都是随机抽取的,这样就降低了样本之间的变异性。层内的个体是同质的,但层与层之间是互不相同的。而簇抽样中的簇总是或多或少地相似,但每个簇都是异质的。被选择的簇"全部"提取。

采用抽样进行数据规约的优点是,得到的样本花费正比例于样本集的大小 s,而不是数据集的大小 N。用于数据规约时,抽样最常用来估计聚集查询的回答。在指定的误差范围内,可以确定(使用中心极限定理)估计一个给定的函数所需的样本大小。样本的大小 s 相对于 N 可能非常小。对于规约数据的逐步求精,抽样是一种自然选择。通过简单地增加样本大小,这样的集合可以进一步求精。

在 Python 中可以使用 DataFrame.sample()随机抽样,用于从 DataFrame 中随机选择行和列。参数 replace 控制是有放回抽样,还是无放回抽样。其语法格式为:

```
DataFrame.sample(n = None, frac = None, replace = False, weights = None, random_state = None,
axis = None)
```

该函数实现从对象轴返回一个随机样本。参数描述如表 5-9 所示。

表 5-9 DataFrame.sample()的参数描述

参数	描 述
n	int,可选参数,表示抽取的行数
frac	float,可选参数,表示抽取的比例,需为小数值。例如,随机抽取 30%的数据,则设置 frac=0.3。不能与 n 一起使用

续表

参数	描　述
replace	bool,默认为 False。表示允许或不允许多次采样同一行。False 表示不允许多次采样同一行。即 False 表示无放回取样,True 表示有放回取样
weights	str 或 ndarray-like,可选参数。默认为 None 即等概率加权
random_state	int, array-like, BitGenerator, np. random. RandomState,可选参数,随机种子,本质是一个控制器,设置此值为任意实数,则每次随机的结果是一样的
axis	0 或 'index', 1 或 'columns', None, 默认为 None。表示抽取数据的行还是列,axis=0 时是抽取行,axis=1 时是抽取列

例 5.8　从"学生成绩记录(去缺失值). xlsx"的数据随机无放回抽取 10 条数据。

```
1:   import pandas as pd
2:  df =  pd.read_excel(r'C:/case/学生成绩记录(去缺失值).xlsx')
3:  print(pd.DataFrame.sample(df,n = 10,replace = False,axis = 0))
```

【例题解析】

此例题显示了无放回随机抽样的使用。

第 1 行导入 pandas 库。第 2 行读取学生成绩记录文件。第 3 行打印使用 DataFrame. sample()函数随机抽取的数据。其中,DataFrame. sample()函数内 df 表示数据集,n=10 代表抽取 10 条数据,replace=False 表示无放回抽样,因为默认即为 False 故此参数可以不写,axis=0 表示在行上进行抽样。

【运行结果】

	学年	学期	考试科目	考试性质	学分	成绩	班级	学号
5153	2020	春	Web 应用系统开发	正常考试	3.0	92.0	B17	20171993
6993	2021	春	数据库技术及应用	正常考试	4.0	76.0	B19	20192337
1944	2018	秋	Web 应用系统开发	正常考试	3.0	74.0	B15	20151837
6836	2021	秋	决策支持系统	正常考试	2.0	79.0	B18	20182099
2787	2018	秋	C++面向对象程序设计	正常考试	4.0	98.0	B17	20171993
4167	2019	春	中国近现代史纲要	正常考试	2.0	80.0	B18	20182149
7172	2021	秋	西方经济学(宏观)(B)	正常考试	2.0	93.0	B19	20192379
3045	2018	秋	大学英语Ⅰ	正常考试	4.0	84.0	B18	20182123
6783	2021	秋	IT 项目管理	正常考试	3.0	87.0	B18	20182085
6473	2021	春	毕业设计	正常考试	11.0	89.9	B17	20172047

例 5.9　从不同课程的学生成绩记录中随机抽取两个学生。

```
1:  import random
2:  import pandas as pd
3:  def get_sample(df, k = 1, stratified_col = None):
4:      grouped  =  df.groupby(by = stratified_col)[stratified_col[0]].count()
```

```
 5:         group_k = grouped.map(lambda x:k)
 6:         res_df = pd.DataFrame(columns = df.columns)
 7:         for df_idx in group_k.index:
 8:             df1 = df
 9:             df1 = df1[df1[stratified_col[0]] == df_idx]
10:             idx = random.sample(range(len(df1)), group_k[df_idx])
11:             group_df = df1.iloc[idx,:].copy()
12:             res_df = res_df.append(group_df)
13:         return res_df
14: if __name__ == '__main__':
15:     df = pd.read_excel(r'C:/case/学生成绩记录(去缺失值).xlsx')
16:     a = get_sample(df = df, k = 2,stratified_col = ['课程名称'])
17:     print(a)
```

【例题解析】

从不同的课程进行抽样,即分层抽样。该例旨在演示如何进行分层抽样。

第 1～2 行引入 random、pandas 库。

第 3～13 行定义分层抽样函数。参数 df 是抽样对象,k 为每一个层抽样的个数,stratified_col 表示分层依据。这里的基本思路是,先对整体数据进行分层处理,然后随机从每层中抽取指定数量的样本。

具体来讲,第 4 行使用 groupby 根据分层依据 stratified_col 进行分组,确定每层的数据量。第 5 行确定每层中抽样的个数。第 6 行创建新的数据框,其列名与原数据列名一致,意欲存放样本数据。第 7～12 行,通过循环语句对数据进行分层抽样,抽取的数据存进 res_df。第 13 行返回保存的 res_df 抽样结果。

第 14～17 行为程序主体。其中,第 15 行导入数据,括号内为文件路径。第 16 行使用自定义的分层抽样函数对 df 数据依据课程名称进行抽样。每层抽样数目为 2。第 17 行打印抽样结果。

【运行结果】

	学年	学期	考试科目	考试性质	学分	成绩	班级	学号
1020	2017	春	数据库开发工具	正常考试	4.0	71.0	B15	20151837
313	2016	春	VisualBasic 程序设计	正常考试	3.0	80.0	B15	20151823
3236	2019	春	信息系统安全与保密	正常考试	2.0	84.0	B16	20162119
2328	2018	夏	社会实践	正常考试	1.0	89.9	B16	20162089
2959	2018	春	C 语言程序设计	补考	3.5	31.0	B17	20172003
2922	2018	春	马克思主义政治经济学原理	正常考试	2.0	70.0	B17	20172043
6521	2021	春	英语听说Ⅱ	正常考试	1.5	83.0	B18	20182099
4115	2019	秋	中国文化概论	正常考试	2.0	NaN	B18	20182147
7109	2021	春	信息存储与检索	正常考试	2.0	65.0	B19	20192351
7094	2021	春	人力资源管理	正常考试	2.0	80.0	B19	20192351

5.5 数据变换

数据变换即对数据进行规范化处理，将数据转换成“适当的”形式，以便于后续的分析和挖掘。本节将从数据合并、数据抽取和数据计算三方面介绍。

5.5.1 数据合并

记录合并指把两个数据表合并成一个数据表。例如，有两张成绩表的数据类型、格式等均一致，需要把两张表合到一起，以便后续分析使用。在 Python 中表现为把两个数据框合并成一个数据框。

1. concat()函数

concat()函数的语法格式为：

```
concat(objs,axis = 0,join = 'outer',join_axes = None,keys = None,verify_integrity = False,
copy = True,ignore_index = False)
```

返回值为 DataFrame。其功能是实现将数据根据指定轴进行拼接。concat()函数中的常用参数及其描述如表 5-10 所示。

表 5-10 concat()函数参数描述

参 数	描 述
objs	参与连接的 pandas 对象的列表或字典
axis	指明连接的轴向，默认为 0 即横轴
join	接收 inner 或 outer，默认 outer
join_axes	指定根据哪个轴来对齐数据
keys	与连接对象有关的值，用于形成连接轴向上的层次化索引。可以是任意值的列表或数组、元组数组、数组列表(如果将 levels 设置成多级数组)
verify_integrity	检查结果对象新轴上的重复情况，如果发现则引发异常。默认(False)允许重复
copy	是否复制数据。默认为 True
ignore_index	不保留连接轴上的索引，产生一组新索引 range(total_length)

2. merge()函数

merge()函数连接两个数据框对象 DataFrame 并返回连接之后的数据框对象 DataFrame，与 SQL 中的 join 用法类似。其语法格式为：

```
merge(left, right, how = 'inner', on = None, left_on = None, right_on = None,left_index =
False, right_index = False, sort = True,suffixes = ('_x', '_y'), copy = True)
```

其中参数描述如表 5-11 所示。

表 5-11 merge()函数参数描述

参 数	描 述
left	参与合并的左侧 DataFrame
right	参与合并的右侧 DataFrame
how	连接方式：'inner'(默认)、'outer'、'left'、'right'
on	用于连接的列名，必须同时存在于左右两个 DataFrame 对象中，如果未指定，则以 left 和 right 列名的交集作为连接键
left_on	左侧 DataFarme 中用作连接键的列
right_on	右侧 DataFarme 中用作连接键的列
left_index	将左侧的行索引用作其连接键
right_index	将右侧的行索引用作其连接键
sort	根据连接键对合并后的数据进行排序，默认为 True。有时在处理大数据集时，禁用该选项可获得更好的性能
suffixes	字符串值元组，用于追加到重叠列名的末尾，默认为('_x','_y')。例如，左右两个 DataFrame 对象都有'data'，则结果中就会出现'data_x','data_y'
copy	设置为 False，可以在某些特殊情况下避免将数据复制到结果数据结构中。默认总是复制

例 5.10 设有贷款状态表 loan_stats 和用户等级表 member_grade，如图 5-7 所示。查询不同等级会员的贷款状态包括数额、年限等信息。

	A	B	C	D	E
1	id	member_id	loan_amnt	term	int_rate
2	1077501	1296599	5000	36 months	10.65%
3	1077175	1313524	2400	36 months	15.96%
4	1075358	1311748	3000	60 months	12.69%
5	1075269	1311441	5000	36 months	7.90%
6	1072053	1288686	3000	36 months	18.64%
7	1071795	1306957	5600	60 months	21.28%

(a) 贷款状态表loan_stats

	A	B
1	member_id	grade
2	1296599	B
3	1313524	C
4	1277178	C
5	1311441	A
6	1304742	C
7	1306957	F

(b) 用户等级表member_grade

图 5-7 例 5.10 数据表

```
1:  import pandas as pd
2:  loanstats = pd.DataFrame(pd.read_excel(r'C:/case/loan_stats.xlsx'))
```

```
3:  member_grade = pd.DataFrame(pd.read_excel(r'C:/case/member_grade.xlsx'))
4:  loan_inner = pd.merge(loanstats,member_grade,how = 'inner')
5:  print(loan_inner)
```

【例题解析】

贷款状态 loan_stats 表中有贷款状态的各种信息，但缺少用户等级；而 member_grade 表中有用户等级，但缺少贷款的各种状态。将两张表通过 member_id 连接起来，即可获得不同等级会员的贷款状态信息。

因此，第 1 行导入 pandas 库。第 2～3 行使用 read_excel 函数分别读取 loan_stats 与 member_grade 中数据。第 4 行使用 merge()函数中的 inner 方式连接两张表。inner 方式是通过相同列(即 member_id)，将具有相同 member_id 的行拼接起来，并仅留一个 member_id 列。

【运行结果】

运行结果如图 5-8 所示。

	id	member_id	loan_amnt	term	int_rate	grade
0	1077501	1296599	5000	36 months	0.1065	B
1	1077175	1313524	2400	36 months	0.1596	C
2	1075269	1311441	5000	36 months	0.0790	A
3	1071795	1306957	5600	60 months	0.2128	F

图 5-8 inner 连接表结果

3. join()函数

join()函数是将两个不同列索引的数据框 DataFrame 组合成单一 DataFrame 的方法，默认为左外连接，其语法格式为：

```
join(other,on = None,how = 'left',sort = False)
```

其参数描述如表 5-12 所示。

表 5-12 join()函数参数说明

参 数	说 明
other	DataFrame、series 或者 DataFrame 组成的 list
on	列名，包含列名的 list 或 tuple，或矩阵样子的列，跟上面的几种函数一样，用来指明依据哪一列进行合并。如果没有赋值，则依据两个数据框的 index 合并
how	连接方式：inner、outer、left(默认)、right
sort	sort：布尔型，默认为 False。如果为 True，将链接键(on 的那列)按字母排序

4. append()函数

append()函数用于向DataFrame对象中添加新的行,如果添加的列名不在DataFrame对象中,将会被当作新的列进行添加。其语法为:

```
DataFrame.append(other,ignore_index = False,verify_integrity = False,sort = None)
```

其参数的意义如表5-13所示。

表5-13 append()函数参数描述

参 数	描 述
other	DataFrame、series、dict或list,表示要追加的数据
ignore_index	bool,默认为False。如果为True,则不要使用索引标签
verify_integrity	bool,默认为False。如果为True,在创建带有重复项的索引时引发ValueError
sort	bool,默认为None。如果self和other的列没有对齐,则对列进行排序

以上这四种函数对比如表5-14所示。

表5-14 concat()、merge()、join()和append()的适用情形

函数	使用场景	调用方法	备 注
concat()	用于两个或多个df间行方向(增加行)或列方向(增加列)进行内连或外连拼接操作,默认行拼接,取并集	result = pd.concat([df1,df2],axis=1)	提供了参数axis设置行/列拼接的方向
merge()	可用于两个df间行方向(一般用join代替)或列方向的拼接操作,默认列拼接,取交集(即存在相同主键的df1和df2的列拼接)	result = pd.merge(df1,df2,how='left')	提供了类似于SQL数据库连接操作的功能,支持左连、右连、内连和外连等全部四种SQL连接操作类型
join()	可用于dfi间列方向的拼接操作,默认左列拼接,how='left'	df1.join(df2)	支持左连、右连、内连和外连四种操作类型
append()	可用于df间行方向的拼接操作,默认		

例5.11 2017学年学生选课的记录如图5-9所示。图5-10中显示的是2018学年学生的选课记录和成绩。2017学年学生的成绩、得奖记录和双学位成绩如图5-11所示。

	A	B	C	D	E	F	G
1	学年	学期	考试科目	考试性质	学分	班级	学号
2	2017	春	证券投资学	正常考试	2.00000	B15	20151809
3	2017	春	证券投资学	正常考试	2.00000	B15	20151845
4	2017	春	企业管理	正常考试	1.50000	B15	20151801
5	2017	春	企业管理	正常考试	1.50000	B15	20151851
6	2017	春	证券投资学	正常考试	2.00000	B15	20151803
7	2017	春	管理学原理	正常考试	1.50000	B15	20151863
8	2017	春	货币银行学(B)	正常考试	2.00000	B15	20151831
9	2017	春	企业管理	正常考试	1.50000	B15	20151835

图5-9 2017学年学生选课表

	A	B	C	D	E	F	G	H
1	学年	学期	考试科目	考试性质	学分	成绩	班级	学号
2	2018	秋	体育舞蹈	正常考试	1.00000	80.00000	B15	20151861
3	2018	秋	多媒体技术应用基础	正常考试	3.00000	82.00000	B15	20151801
4	2018	秋	音乐知识与作品鉴赏	正常考试	1.50000	83.00000	B15	20151801
5	2018	秋	形体与健美	正常考试	1.00000	0.00000	B15	20151845
6	2018	春	VisualFoxPro程序设计	正常考试	3.50000	62.00000	B15	20151819
7	2018	春	Java语言程序设计	正常考试	3.00000	87.00000	B15	20151809
8	2018	春	UNIX/LINUX基础	正常考试	2.00000	83.00000	B15	20151809
9	2018	春	竞技体育	正常考试	1.00000	82.00000	B15	20151821
10	2018	春	竞技体育	正常考试	1.00000	84.00000	B15	20151829
11	2018	春	女性学	正常考试	1.50000	55.00000	B15	20151857
12	2018	春	知识产权法	正常考试	2.00000	81.00000	B15	20151825
13	2018	春	女性学	正常考试	1.50000	67.00000	B15	20151863
14	2018	春	面向对象程序设计	正常考试	4.00000	67.00000	B15	20151801

图 5-10　2018 学年学生选课成绩表

	A
1	成绩
2	60.10000
3	60.10000
4	76.00000
5	80.00000
6	60.10000
7	85.00000
8	60.10000
9	67.00000

(a) 成绩表

	A	B
1	学号	得奖信息
2	20151809	一等奖
3	20151845	二等奖
4	20151801	一等奖
5	20151851	三等奖
6	20151803	三等奖
7	20151863	三好学生

(b) 学生得奖记录表

	A	B	C
1	学号	双学位课程	双学位成绩
2	20151809	英语	79
3	20151845	英语	82
4	20151801	马克思	80

(c) 学生双学位表

图 5-11　2017 学年学生考试成绩、获奖记录和双学位课程信息表

```
 1:  import pandas as pd
 2:  df1 = pd.read_excel(r'C:/case/2017 学生选课.xlsx')
 3:  df2 = pd.read_excel(r'C:/case/2017 学生成绩.xlsx')
 4:  df3 = pd.read_excel(r'C:/case/2018 学生成绩.xlsx')
 5:  df4 = pd.read_excel(r'C:/case/2017 学生得奖记录.xlsx')
 6:  df5 = pd.read_excel(r'C:/case/2017 学生双学位.xlsx')
 7:  dfa = df1.join(df2)
 8:  print(dfa)
 9:  dfb = dfa.append(df3)
10:  print(dfb)
11:  dfc = pd.merge(dfb,df4)
12:  print(dfc)
13:  dfd = pd.concat([dfc,df5],axis = 1,join = 'inner')
14:  print(dfd)
```

【例题解析】

此例意在进一步了解四种函数的区别。

第 1 行导入 pandas 库。第 2～6 行分别导入相应数据；

第 7～8 行使用 join()函数中连接 df1 与 df2，默认根据行标签进行连接，获得了选修课程成绩，命名为 dfa 并打印。第 9～10 行使用 append()函数拼接 dfa 与 df3，命名为 dfb 并打印。如此获得了 2017 学年和 2018 学年学生的选课记录和成绩。

第 11～12 行使用 merge()函数连接 dfb 与 df4，以相同的列“学号”作为连接列名，取

两个数据框中同时包含的对象,命名为 dfc 并打印。实现了显示获奖同学的体育课成绩。第 13～14 行使用 concat()函数连接 dfc 与 df5,axis=1 表示在列的方向上进行连接,inner 表示取交集。在第 12 行运行结果的基础上,进一步显示获得了奖励并同时修读双学位的同学信息,命名为 dfd 并打印。

【运行结果】

第 8 行运行结果:

	学年	学期	考试科目	考试性质	学分	班级	学号	成绩
0	2017	春	证券投资学	正常考试	2.0	B15	20151809	60.1
1	2017	春	证券投资学	正常考试	2.0	B15	20151845	60.1
2	2017	春	企业管理	正常考试	1.5	B15	20151801	76.0
3	2017	春	企业管理	正常考试	1.5	B15	20151851	80.0
4	2017	春	证券投资学	正常考试	2.0	B15	20151803	60.1
5	2017	春	管理学原理	正常考试	1.5	B15	20151863	85.0
6	2017	春	货币银行学(B)	正常考试	2.0	B15	20151831	60.1
7	2017	春	企业管理	正常考试	1.5	B15	20151835	67.0

第 10 行运行结果:

	学年	学期	考试科目	考试性质	学分	班级	学号	成绩
0	2017	春	证券投资学	正常考试	2.0	B15	20151809	60.1
1	2017	春	证券投资学	正常考试	2.0	B15	20151845	60.1
2	2017	春	企业管理	正常考试	1.5	B15	20151801	76.0
3	2017	春	企业管理	正常考试	1.5	B15	20151851	80.0
4	2017	春	证券投资学	正常考试	2.0	B15	20151803	60.1
5	2017	春	管理学原理	正常考试	1.5	B15	20151863	85.0
6	2017	春	货币银行学(B)	正常考试	2.0	B15	20151831	60.1
7	2017	春	企业管理	正常考试	1.5	B15	20151835	67.0
0	2018	秋	体育舞蹈	正常考试	1.0	B15	20151861	80.0
1	2018	秋	多媒体技术应用基础	正常考试	3.0	B15	20151801	82.0
2	2018	秋	音乐知识与作品鉴赏	正常考试	1.5	B15	20151801	83.0
3	2018	秋	形体与健美	正常考试	1.0	B15	20151845	0.0
4	2018	春	VisualFoxPro 程序设计	正常考试	3.5	B15	20151819	62.0
5	2018	春	Java 语言程序设计	正常考试	3.0	B15	20151809	87.0
6	2018	春	UNIX/LINUX 基础	正常考试	2.0	B15	20151809	83.0
7	2018	春	竞技体育	正常考试	1.0	B15	20151821	82.0
8	2018	春	竞技体育	正常考试	1.0	B15	20151829	84.0
9	2018	春	女性学	正常考试	1.5	B15	20151857	55.0
10	2018	春	知识产权法	正常考试	2.0	B15	20151825	81.0
11	2018	春	女性学	正常考试	1.5	B15	20151863	67.0
12	2018	春	面向对象程序设计	正常考试	4.0	B15	20151801	67.0

第 12 行运行结果：

	学年	学期	考试科目	考试性质	学分	班级	学号	成绩	得奖信息
0	2017	春	证券投资学	正常考试	2.0	B15	20151809	60.1	一等奖
1	2018	春	Java 语言程序设计	正常考试	3.0	B15	20151809	87.0	一等奖
2	2018	春	UNIX/LINUX 基础	正常考试	2.0	B15	20151809	83.0	一等奖
3	2017	春	证券投资学	正常考试	2.0	B15	20151845	60.1	二等奖
4	2018	秋	形体与健美	正常考试	1.0	B15	20151845	0.0	二等奖
5	2017	春	企业管理	正常考试	1.5	B15	20151801	76.0	二等奖
6	2018	秋	多媒体技术应用基础	正常考试	3.0	B15	20151801	82.0	二等奖
7	2018	秋	音乐知识与作品鉴赏	正常考试	1.5	B15	20151801	83.0	二等奖
8	2018	春	面向对象程序设计	正常考试	4.0	B15	20151801	67.0	二等奖
9	2017	春	企业管理	正常考试	1.5	B15	20151851	80.0	三等奖
10	2017	春	证券投资学	正常考试	2.0	B15	20151803	60.1	三等奖
11	2017	春	管理学原理	正常考试	1.5	B15	20151863	85.0	三好学生
12	2018	春	女性学	正常考试	1.5	B15	20151863	67.0	三好学生

第 14 行运行结果：

	学年	学期	考试科目	考试性质	学分	...	成绩	得奖信息	学号	双学位课程	双学位成绩
0	2017	春	证券投资学	正常考试	2.0	...	60.1	一等奖	20151809	英语	79
1	2018	春	Java 语言程序设计	正常考试	3.0	...	87.0	一等奖	20151845	英语	82
2	2018	春	UNIX/LINUX 基础	正常考试	2.0	...	83.0	一等奖	20151801	马克思	80

[3 rows x 12 columns]

5.5.2 数据抽取

数据抽取是分析师日常工作中经常遇到的需求。将数据从源表中原封不动地抽取出来，并转换成数据分析需要的格式。从数据抽取的方向，可以分为纵向数据抽取和横向数据筛选。

1. 纵向数据抽取

纵向数据抽取主要是两个操作，即将一个字段拆分成多个字段和从一个字段中抽取特定位置的数据形成新的字段。前者使用的是 split()函数，后者采用 slice()函数。

1)split()函数

该函数通常用于将字符串切片并转换为列表,返回分隔后的字符串列表。其语法格式为:

```
split (sep,n,expand = False)
```

其中参数描述如表 5-15 所示。

表 5-15 split()函数参数描述

参 数	描 述
sep	用于分隔字符串的分隔符
n	分隔后新增的列数(不分隔 n=0,分隔为两列 n=1,以此类推)
expand	是否展开为数据框,默认为 False,一般都设置为 True。 如果 expand 为 True,则返回 DataFrame; 如果 expand 为 False,则返回 Series

例 5.12 "上网记录.xlsx"数据如图 5-12 所示,数据列为:学号、手机号、IP。将 IP 地址以"."为分隔符分为四列,并存于数据框 df1。

学号	手机号	IP
2308024	18603518513	221.205.98.55
2308025	15623462341	183.184.226.205
2308026	13876129892	221.205.98.55
2308027	15172923748	222.31.51.200

图 5-12 上网记录.xlsx

```
1:  import pandas as pd
2:  df = pd.read_excel(r'C:/case/上网记录.xlsx')
3:  df1 = df["IP"].str.split(".",3,True)
4:  print(df1)
```

【例题解析】

第 1 行导入 pandas 库。第 2 行读取上网记录文件,括号中为数据路径。第 3 行先将 IP 列转换为 str 类型,然后使用 split()函数将其以"."为分隔符分开,分开后新增三列,即现在有四列,返回数据框。第 4 行打印结果。

【运行结果】

	0	1	2	3
0	221	205	98	55
1	183	184	226	205
2	221	205	98	55
3	222	31	51	200

2) slice()函数

从数据表中抽出特定位置的数据列,做成新列,采用 slice()函数进行字段截取。slice()

函数的主要作用是获取对象(常用于列表、字符串、元组等)的切片对象。其语法格式为:

```
slice(start,stop,step)
```

其中参数描述如表 5-16 所示。

表 5-16 slice()函数参数描述

参 数	描 述	参 数	描 述
start	起始位置	step	间距
stop	结束位置		

例 5.13 从例 5.12 中的"上网记录.xlsx"数据的电话列中分别取出前三位、中间四位和后四位分别表示品牌、地区和手机号码。例如,18603518513,前三位 186 代表中国联通,中间 0351 表示太原,后四位 8513 是手机号码。

```
1:  import pandas as pd
2:  df = pd.read_excel(r'C:/case/上网记录.xlsx')
3:  df['手机号'] = df['手机号'].astype(str)
4:  df2 = pd.DataFrame()
5:  df2['品牌'] = df["手机号"].str.slice(0,3)
6:  df2['地区'] = df["手机号"].str.slice(3,7)
7:  df2['手机号码'] = df["手机号"].str.slice(7,11)
8:  print(df2)
```

【例题解析】

第 1 行导入 pandas 库。第 2 行读取上网记录文件。第 3 行将手机号列强制类型转换为 str 数据类型。第 4 行定义一个空的数据框存放品牌、地区和手机号码数据。第 5～7 行分别对手机号列转换为 str 类型并使用 slice()函数进行分隔。第 8 行打印结果。

【运行结果】

	品牌	地区	手机号码
0	186	0351	8513
1	156	2346	2341
2	138	7612	9892
3	151	7292	3748

2. 横向数据抽取

横向数据抽取是指从行的角度从数据集中筛选满足一定条件的集合。可以采用数据框的布尔索引实现,即 dataframe[condition],其中,condition 表示过滤条件,返回值为 DataFrame 对象。

例 5.14 在"学生成绩记录(去缺失值).xlsx"数据中,分别筛选出成绩为 60,成绩为 90～100 的同学的详细信息。

```
1:  import pandas as pd
2:  df = pd.read_excel(r'C:/case/学生成绩记录(去缺失值).xlsx')
3:  print(df[df.成绩 == 60])
4:  print(df[df.成绩.between(90,100)])
```

【例题解析】

第 1 行导入 pandas 库。第 2 行读取学生成绩记录文件,括号中为数据存放路径。第 3 行打印成绩列数值为 60 的数据信息,第 4 行筛选出成绩为 90～100 的数据。

【运行结果】

	学年	学期	考试科目	考试性质	学分	成绩	班级	学号
4	2015	秋	体育Ⅰ	正常考试	1.0	60.0	B15	20151813
9	2015	秋	体育Ⅰ	正常考试	1.0	60.0	B15	20151823
58	2015	秋	大学英语Ⅰ	正常考试	4.0	60.0	B15	20151813
73	2015	秋	大学英语Ⅰ	正常考试	4.0	60.0	B15	20151809
76	2015	秋	体育Ⅰ	正常考试	1.0	60.0	B15	20151845
	…	…	…	…	…	…	…	…
7424	2021	春	计算机网络技术	正常考试	3.0	60.0	B19	20192371
7534	2022	春	女性学	正常考试	1.5	60.0	B19	20192351
7544	2022	春	生活美语与北美文化	正常考试	1.5	60.0	B19	20192391
7546	2022	春	日语专题	正常考试	2.0	60.0	B19	20192389
7621	2022	春	信息系统安全与保密	正常考试	2.0	60.0	B19	20192405

[251 rows x 8 columns]

	学年	学期	考试科目	考试性质	学分	成绩	班级	学号
8	2015	秋	马克思主义哲学原理	正常考试	2.5	90.0	B15	20151853
16	2015	秋	高等数学(A)Ⅰ	正常考试	4.0	90.0	B15	20151795
17	2015	秋	高等数学(A)Ⅰ	正常考试	4.0	90.0	B15	20151803
21	2015	秋	高等数学(A)Ⅰ	正常考试	4.0	98.0	B15	20151823
29	2015	秋	高等数学(A)Ⅰ	正常考试	4.0	90.0	B15	20151853
	…	…	…	…	…	…	…	…
7901	2023	春	毕业实习	正常考试	6.0	100.0	B19	20192367
7904	2023	春	毕业实习	正常考试	6.0	100.0	B19	20192399
7907	2023	春	毕业实习	正常考试	6.0	100.0	B19	20192369
7909	2023	春	毕业实习	正常考试	6.0	100.0	B19	20192393
7920	2023	春	毕业实习	正常考试	6.0	100.0	B19	20192365

[809 rows x 8 columns]

5.5.3 数据计算

本节将讨论数据的简单计算、数据规范化和离散化方法。

1. 简单计算

一个数据表中的各字段(或列)可以进行加、减、乘、除四则算术运算,结果作为新的数据列。两个不同数据表中的数据,可以使用 pandas 中数据框(DataFrame)对象,在运算时通过对象索引自动对齐进行相应列的计算;如果参加运算的数据框中存在不同索引,则结果索引是所有索引的并集,对应不同索引的值标记为 NaN。也可以使用 add(加)、sub(减)、div(除)和 mul(乘)等方法,将其他 DataFrame 对象的值传入指定 DataFrame 对象,实现两个不同数据表中数据的简单计算。

例 5.15 在"学生成绩记录(去缺失值).xlsx"数据中,根据学分与成绩计算标准成绩,即学分×成绩。

```
1:  import pandas as pd
2:  df = pd.read_excel(r'C:/case/学生成绩记录(去缺失值).xlsx')
3:  df['标准分数'] = df['学分'] * df['成绩']
4:  print(df)
```

【例题解析】

第 1 行引入 pandas 库。第 2 行读取学生成绩记录数据。第 3 行增加一列"标准分数",值为学生成绩记录中的成绩一列乘以对应学分列。第 4 行打印结果。

【运行结果】

	学年	学期	考试科目	考试性质	学分	成绩	班级	学号	标准分数
0	2015	秋	体育Ⅰ	正常考试	1.0	85.0	B15	20151803	85.0
1	2015	秋	体育Ⅰ	正常考试	1.0	70.0	B15	20151795	70.0
2	2015	秋	体育Ⅰ	正常考试	1.0	80.0	B15	20151819	80.0
3	2015	秋	体育Ⅰ	正常考试	1.0	70.0	B15	20151809	70.0
4	2015	秋	体育Ⅰ	正常考试	1.0	60.0	B15	20151813	60.0
	…	…	…	…	…	…	…	…	…
7917	2023	春	毕业实习	正常考试	6.0	89.9	B19	20192341	539.4
7918	2023	春	毕业设计	正常考试	11.0	79.9	B19	20192371	878.9
7919	2023	春	毕业实习	正常考试	6.0	79.9	B19	20192371	479.4
7920	2023	春	毕业实习	正常考试	6.0	100.0	B19	20192365	600.0
7921	2023	春	毕业设计	正常考试	11.0	79.9	B19	20192365	878.9

[7922 rows x 9 columns]

2. 数据规范化

在数据文件中所用的度量单位可能影响数据分析。例如,把身高的度量单位从米变成英寸,把体重的度量单位从千克改成市斤,可能导致完全不同的结果。为了避免对度量单位选择的依赖性,数据应该规范化或标准化。这涉及变换数据,使之落入较小的共同区间,如[−1,1]或[0.0,1.0]。在数据预处理中,术语"规范化"和"标准化"可以互换

使用。

规范化数据试图赋予所有数据列相等的权重。有许多数据规范化的方法,本节主要介绍最小-最大规范化、z分数规范化和按小数定标规范化。在以下的讨论中,令 A 是数值属性列,具有 n 个观测值 $v_1,v_2,\cdots,v_n$。

1) 最小-最大规范化

对原始数据进行线性变换。假设 $\min_A$ 和 $\max_A$ 分别为数据列 A 的最小值和最大值。最小-最大规范化通过计算

$$v_i'=\frac{v_i-\min_A}{\max_A-\min_A}(\text{new\_max}_A-\text{new\_min}_A)+\text{new\_min}_A \tag{5-7}$$

把 A 的值 v 映射到区间[new_min_A,new_max_A]中的 v_i'。最小-最大规范化保持原始数据值之间的联系。如果今后的输入实例落在 A 的原数据值域之外,则该方法将面临“越界”错误。

例 5.16 使用最小-最大规范化方法,对“学生成绩记录(去缺失值).xlsx”数据中的成绩规范到区间[0,1]。

```
1:  import pandas as pd
2:  df = pd.read_excel(r'C:/case/学生成绩记录(去缺失值).xlsx')
3:  print((df["成绩"] - df["成绩"].min())/(df["成绩"].max() - df["成绩"].min()))
```

【例题解析】

第 1 行引入 pandas 库。第 2 行导入文件,括号中为文件路径。第 3 行打印对成绩最小-最大规范化结果。

【运行结果】

```
0       0.076715
1       0.063177
2       0.072202
3       0.063177
4       0.054152

7917    0.081137
7918    0.072112
7919    0.072112
7920    0.090253
7921    0.072112
Name: 成绩, Length: 7922, dtype: float64
```

2) 小数定标规范化

通过移动数据列 A 值的小数点位置进行规范化。小数点的移动位数依赖于 A 的最大绝对值。A 的值 v_i 被规范化为 v_i',由式(5-8)计算:

$$v'_i=\frac{v_i}{10^j} \tag{5-8}$$

其中，j 是使得 max（$|v'_i|$）<1 的最小整数。

例 5.17 对“学生成绩记录(去缺失值).xlsx”数据使用小数定标规范化对成绩进行规范。

```
1:  import pandas as pd
2:  import numpy as np
3:  df = pd.read_excel(r'C:/case/学生成绩记录(去缺失值).xlsx')
4:  df = df['成绩']/10 ** np.ceil(np.log10(df['成绩'].abs().max()))
5:  print(df)
```

【例题解析】

第 1 行引入 pandas 库。第 2 行引入 NumPy 库。

第 3 行导入文件，括号中为文件路径。第 4 行使用小数定标的计算公式对成绩列进行计算。第 5 行打印拟合后的结果。

【运行结果】

```
0       0.00850
1       0.00700
2       0.00800
3       0.00700
4       0.00600

7917    0.00899
7918    0.00799
7919    0.00799
7920    0.01000
7921    0.00799
Name：成绩，Length：7922，dtype：float64
```

3. 数据离散化

离散化是指将连续数据、特征或变量转换或划分为离散或标称属性/间隔的过程。下面介绍基于 Python 的标签化方法和分箱离散化方法。

1）标签化

在进行数据分析时，有时需要对数据列做标签化处理，便于理解或后续处理。例如，在进行数据记录时，为了方便将考试性质以“0，1，2”存储，但是这样不容易理解。此时，可对这些值进行标签化处理，将其恢复为易于理解的“正常考试、补考、重修”形式。

首先使用 DataFrame 的 astype()函数将原始数据转换为 category 类型。category 类型在 pandas 中是和 string、int 等类型并列的一种数据类型，中文翻译可以理解为分类。

category 类型可以有效地对数据进行分组进行汇总统计工作。然后,利用 cat.categories()为数据值挂标签时,赋值的时候是按照顺序进行对应的。

例 5.18 “2017 年学生成绩记录节选.xlsx”数据如图 5-15 所示。数据列分别为“学年”“学期”“考试科目”“考试性质”“学分”“成绩”“班级”“学号”。请为考试性质设置标签“0”(正常考试)、“1”(补考)、“2”(重修)。

学年	学期	考试科目	考试性质	学分	成绩	班级	学号
2017	春	证券投资学	正常考试	2.00000	60.10000	B15	20151809
2017	春	证券投资学	正常考试	2.00000	60.10000	B15	20151845
2017	春	企业管理	正常考试	1.50000	76.00000	B15	20151801
2017	春	企业管理	正常考试	1.50000	80.00000	B15	20151851
2017	春	证券投资学	正常考试	2.00000	60.10000	B15	20151803
2017	春	管理学原理	正常考试	1.50000	85.00000	B15	20151863

图 5-15 2017 年学生成绩记录节选

```
1:  import pandas as pd
2:  df = pd.read_excel(r'C:/case/2017 年学生成绩记录节选.xlsx')
3:  df['考试性质_标签'] = df['考试性质'].astype('category')
4:  df['考试性质_标签'].cat.categories = ['0','1','2']
5:  print(df)
```

【例题解析】

第 1 行引入 pandas 库。第 2 行读取数据。第 3 行将考试性质列变为 category 类型并命名列为考试性质_标签。第 4 行将考试性质_标签列通过 cat.categories()函数进行标签“0”“1”“2”处理。第 5 行输出挂上标签以后的数据。

【运行结果】

	学年	学期	考试科目	考试性质	学分	成绩	班级	学号	考试性质_标签
0	2017	春	证券投资学	正常考试	2.0	60.1	B15	20151809	0
1	2017	春	证券投资学	正常考试	2.0	60.1	B15	20151845	0
2	2017	春	企业管理	正常考试	1.5	76.0	B15	20151801	0
3	2017	春	企业管理	正常考试	1.5	80.0	B15	20151851	0
4	2017	春	证券投资学	正常考试	2.0	60.1	B15	20151803	0
…	…	…	…	…	…	…	…	…	…
1044	2017	秋	高等数学(A)Ⅰ	正常考试	4.0	82.0	B17	20172043	0
1045	2017	秋	高等数学(A)Ⅰ	正常考试	4.0	55.0	B17	20171997	0
1046	2017	秋	高等数学(A)Ⅰ	正常考试	4.0	100.0	B17	20172013	0
1047	2017	秋	高等数学(A)Ⅰ	正常考试	4.0	55.0	B17	20172037	0
1048	2017	秋	高等数学(A)Ⅰ	正常考试	4.0	61.0	B17	20172007	0

[1049 rows x 9 columns]

2) 分箱离散化

使用 pandas 库下的 cut()函数实现对数值型数据分段标签,其语法格式为

```
pandas.cut(x,bins ,right = True,labels = None,retbins  = False,precision =  3 ,include_lowest
 = False)
```

其中,参数解释如表 5-17 所示。

表 5-17 pandas.cut()函数参数描述

参 数	描 述
x	将要操作的一维数组对象
bins	将要对 x 数组中的每个元素归到 bins 个组中(间距不一定相等),有两种形式:整数、序列。如果 bins 是一个整数,它定义了 x 宽度范围内的等宽面元数量,但是在这种情况下,x 的范围在每个边上被延长 1%,以保证包括 x 的最小值或最大值。如果 bins 是序列,它定义了允许非均匀 bin 宽度的 bin 边缘,在这种情况下没有 x 的范围的扩展
right	分组时是否包含右边的值(即区间右边是否是闭合),形式:right = True
labels	是否用标签来代替分组的结果。必须与结果箱相同长度,比如 bins=[1,2,3],划分后有两个区间(1,2],(2,3],则 labels 的长度必须为 2
retbins	为布尔值,默认为 False。表示是否将分割后的 bins 返回,当 bins 为一个 int 型的标量时比较有用,这样得到划分后的区间
precision	精度,默认为 3
include_lowest	是布尔值,默认为 False。表示区间的左边是开还是闭,当为 False 时,不包含区间左部

例 5.19 将上述“学生成绩记录(去缺失值).xlsx”数据,根据成绩列的数值,进行分段标签处理,并将此列命名为“成绩_区间”。

```
1:  import pandas as pd
2:  df =  pd.read_excel(r'C:/case/学生成绩记录(去缺失值).xlsx')
3:  listBins = [0,59,69,79,89,100]
4:  listlabels = ['0 - 59', '60 - 69', '70 - 79', '80 - 89','90 - 100']
5:  df['成绩_区间'] = pd.cut(df['成绩'], bins = listBins, labels = listlabels,right =
True)
6:  print(df)
```

【例题解析】

第 1 行引入 pandas 库。第 2 行读取学生成绩记录数据,引号内为文件位置。第 3 行定义成绩的分组的列表。第 4 行根据成绩分组创建标签的列表。第 5 行使用 cut()函数将学生成绩记录数据的成绩列按 listBins 分组,并使用 listlabels 的标签来代替分组的结果,并赋值给 df['成绩_区间']。第 6 行输出结果。

【运行结果】

	学年	学期	考试科目	考试性质	学分	成绩	班级	学号	成绩_区间
0	2015	秋	体育Ⅰ	正常考试	1.0	85.0	B15	20151803	80-89
1	2015	秋	体育Ⅰ	正常考试	1.0	70.0	B15	20151795	70-79
2	2015	秋	体育Ⅰ	正常考试	1.0	80.0	B15	20151819	80-89

3	2015	秋	体育Ⅰ	正常考试	1.0	70.0	B15	20151809	70-79
4	2015	秋	体育Ⅰ	正常考试	1.0	60.0	B15	20151813	60-69
	…	…	…	…	…	…	…	…	…
7917	2023	春	毕业实习	正常考试	6.0	89.9	B19	20192341	90-100
7918	2023	春	毕业设计	正常考试	11.0	79.9	B19	20192371	80-89
7919	2023	春	毕业实习	正常考试	6.0	79.9	B19	20192371	80-89
7920	2023	春	毕业实习	正常考试	6.0	100.0	B19	20192365	90-100
7921	2023	春	毕业设计	正常考试	11.0	79.9	B19	20192365	80-89

[7922 rows x 9 columns]

小结

数据分析依赖于数据质量,高质量的数据才可能得出高质量的结果。因此本章主要介绍数据预处理的主要步骤和相关方法,包括数据清洗、数据集成、数据规约和数据变换等。

在数据清洗方面,主要介绍了重复值、缺失值和噪声的检测与处理。其中,重复值的检测一般使用 Python 的 duplicated()函数,重复值的删除一般使用 drop_duplicates()函数;缺失值的检测一般使用 isnull()函数,删除使用 dropna()函数,填充使用 fillna()函数;常用的数据平滑去噪的技术有分箱(Binning)、回归(Regression)和离群点分析(Outlier Analysis)。

在数据集成方面,主要介绍了实体识别问题,使用假设检验方法进行数据冗余问题的识别以及数据值冲突问题;在数据归约方面,重点介绍了从列的角度选择对数据分析结果较为重要作用的列的方法,即属性子集选择;从行的角度出发,对数据进行随机抽样,分层抽样以及簇抽样的方法。

在数据变换方面,主要介绍了数据的合并、抽取和计算方法。其中,数据的合并可以通过 Python 的 concat()函数、merge()函数、join()函数和 append()函数实现;数据的抽取包括纵向数据抽取和横向数据抽取,关于纵向数据抽取介绍了 split()函数和 slice()函数,横向数据抽取可以采用 Python 的 dataframe[condition];关于数据的计算,主要介绍了简单计算、数据的规范化以及离散化。

习题

请从以下各题中选出正确答案(正确答案可能不止一个)。

1. 数据清洗通常包括以下哪种?(　　)

A. 重复值处理　　B. 缺失值处理　　C. 异常值处理　　D. 以上均不是

2. 给定一组数据,如图 5-14 所示,分析每门课的平均成绩,需要进行什么样的数据清洗?(　　)

A. 去重　　B. 缺失值填充

姓名	语文	英语	数学
张飞	66	65	-
关羽	95	85	88
赵云	95	92	96
黄忠	90	88	77
典韦	80	90	90
典韦	80	90	90

图 5-14　平均成绩

C. 噪声平滑　　D. 以上均不对

3. 以下关于 drop_duplicates 函数的说法中错误的是(　　)。

A. 仅对 Dataframe 和 Series 类型的数据有效

B. 仅支持单一特征的数据去重

C. 数据重复时默认保留第一个数据

D. 该函数不会改变原始数据排列

4. (　　)是一个观测值,它与其他观测值的差别如此之大,以至于怀疑它是由不同的机制产生的。

A. 边界点　　B. 质心　　C. 离群点　　D. 核心点

5. 将原始数据进行集成、变换、维度规约、数值规约是在以下哪个步骤的任务?(　　)

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

6. 假设 12 个销售价格记录组已经排序如下:5,10,11,13,15,35,50,55,72,92,204,215,使用如下每种方法将它们划分成四个箱。等频(等深)划分时,15 在第几个箱子内)?(　　)

A. 第一个　　B. 第二个　　C. 第三个　　D. 第四个

7. 假设属性 income 的最大最小值分别是 12 000 元和 98 000 元。利用最大最小规范化的方法将属性的值映射到 0～1 的范围内。对属性 income 的 73 600 元将被转换为(　　)。

A. 0.821　　B. 1.224　　C. 1.458　　D. 0.716

8. 关于下列数据合并函数的说法错误的是(　　)。

A. join(),主要用于基于索引的纵向合并拼接

B. merge(),主要用于基于指定列的横向合并拼接

C. concat(),可用于横向和纵向合并拼接

D. append(),主要用于纵向追加

9.
```
u = "www.doiido.com.cn"
print (u.split('.',1))
```

下列输出结果正确的是(　　)。

A. ['www', 'doiido', 'com',' cn']

B. ['www', 'doiido', 'com.cn']

C. ['www', 'doiido.com', 'cn']

D. ['www', 'doiido.com.cn']

10. 若知道数据的 Max(最大值)、Min(最小值)、mu(均值)、sigma(标准差)。下列几种数据标准化函数正确的是(　　)。

A. (0,1)标准化:

```
def  MaxMinNormalization(x,Max,Min):
       x = (x - Min) / (Max - Min)
       return x
```

B. Z-score 标准化:

```
def  Z_ScoreNormalization(x,mu,sigma):
      x = (x - mu) / sigma
      return x
```

C. Sigmoid 函数:

```
defsigmoid(X,useStatus):
    if useStatus:
    return 1.0 / (1 + np.exp(-float(X)))
else:
    return float(X)
```

D. 以上均正确