

1 pandas进行数据预处理



pandas进行数据预处理

1.1.1 合并数据

1.1.2 清洗数据

1.1.3 标准化数据

1.1.4 转换数据

1.1.5 数据规约

- 1.1.1 合并数据
 - 堆叠合并数据
 - 主键合并数据
 - 重叠合并数据

●1.1.1 合并数据- 堆叠合并数据

• 1. 横向表堆叠

- 横向堆叠，即将两个表在X轴向拼接在一起，可以使用concat函数完成，concat函数的基本语法如下。

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

- 常用参数如下所示。

参数名称	说明
objs	接收多个Series, DataFrame, Panel的组合。表示参与链接的pandas对象的列表的组合。无默认。
axis	接收0或1。表示连接的轴向，默认为0。
join	接收inner或outer。表示其他轴向上的索引是按交集（inner）还是并集（outer）进行合并。默认为outer。
join_axes	接收Index对象。表示用于其他n-1条轴的索引，不执行并集 / 交集运算。

●1.1.1 合并数据- 堆叠合并数据

• 1. 横向表堆叠

参数名称	说明
ignore_index	接收boolean。表示是否不保留连接轴上的索引，产生一组新索引range(total_length)。默认为False。
keys	接收sequence。表示与连接对象有关的值，用于形成连接轴向上的层次化索引。默认为None。
levels	接收包含多个sequence的list。表示在指定keys参数后，指定用作层次化索引各级别上的索引。默认为None。
names	接收list。表示在设置了keys和levels参数后，用于创建分层级别的名称。默认为None。
verify_integrity	接收boolean。表示是否检查结果对象新轴上的重复情况，如果发现则引发异常。默认为False。

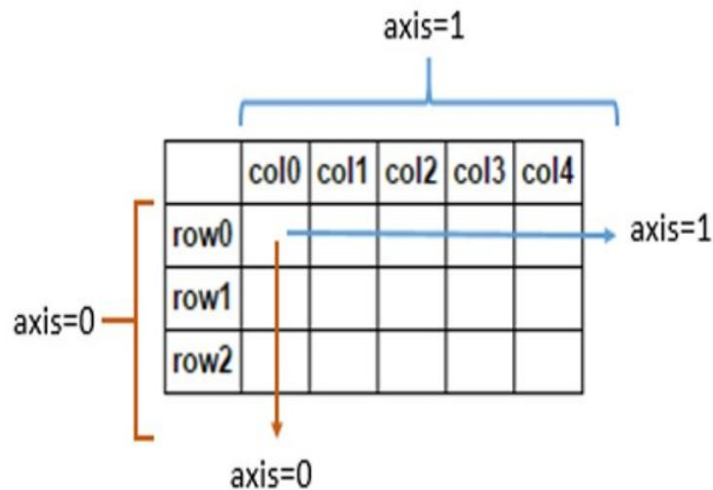
1.1 pandas进行数据预处理

●1.1.1 合并数据- 堆叠合并数据

• 1. 横向表堆叠

- 当axis=1的时候，concat做横向合并，列为主，然后将不同列名称的两张或多张表合并。当两个表索引不完全一样时，可以使用join参数选择是内连接还是外连接。在内连接的情况下，仅仅返回索引重叠部分。在外连接的情况下，则显示索引的并集部分数据，不足的地方则使用空值填补。
- 当两张表完全一样时，不论join参数取值是inner或者outer，结果都是将两个表完全按照X轴拼接起来。

表1					表2				合并后表3							
	A	B	C	D		B	D	F		A	B	C	D	B	D	F
1	A1	B1	C1	D1	2	B2	D2	F2	1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	4	B4	D4	F4	2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	6	B6	D6	F6	3	A3	B3	C3	D3	NaN	NaN	NaN
4	A4	B4	C4	D4	8	B8	D8	F8	4	A4	B4	C4	D4	B4	D4	F4
									6	NaN	NaN	NaN	NaN	B6	D6	F6
									8	NaN	NaN	NaN	NaN	B8	D8	F8



1.1 pandas进行数据预处理

●1.1.1 合并数据- 堆叠合并数据

• 2. 纵向堆叠——concat函数

- 行为主，使用concat函数时，在默认情况下，即axis=0时，concat做列对齐，将不同行索引的两张或多张表纵向合并。在两张表的列名并不完全相同的情况下，可join参数取值为inner时，返回的仅仅是列名交集所代表的列，取值为outer时，返回的是两者列名的并集所代表的列，其原理示意如图。
- 不论join参数取值是inner或者outer，结果都是将两个表完全按照Y轴拼接起来

表1				
	A	B	C	D
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4

表2				
	B	D	F	
2	B2	D2	F2	
4	B4	D4	F4	
6	B6	D6	F6	
8	B8	D8	F8	

合并后表3					
	A	B	C	D	F
1	A1	B1	C1	D1	NaN
2	A2	B2	C2	D2	NaN
3	A3	B3	C3	D3	NaN
4	A4	B4	C4	D4	NaN
2	NaN	B2	NaN	D2	F2
4	NaN	B4	NaN	D4	F4
6	NaN	B6	NaN	D6	F6
8	NaN	B8	NaN	D8	F8

完成以下练习：

数据如下图所示 打开concat_append.ipynb文件练习

一：文件中已给定数据集，请执行以下操作

1. concat进行纵向合并 注意axis合并方向
2. 并将ignore_index设定为True：重置index

二：定义资料集，进行join参数的练习

join (合并方式) 如果为'inner'得到的是两表的交集，如果是outer，得到的是两表的并集

1. 纵向"外"合并df1与df2
2. 纵向"内"合并df1与df2
3. 重置index并打印结果

完成以下练习：

打开concat_append.ipynb文件练习

join_axes (依照 axes 合并) 可以指定根据那个轴来对齐数据

#已有定义好的资料集

#要求依照`df1.index`进行横向合并

●1.1.1 合并数据- 堆叠合并数据

2. 纵向堆叠——append方法

- append方法也可以用于纵向合并两张表。但是append方法实现纵向表堆叠有一个前提条件，那就是两张表的列名需要完全一致。append方法的基本语法如下

pandas.DataFrame.append(self, other, ignore_index=False, verify_integrity=False)。

- 常用参数如下所示。

参数名称	说明
other	接收DataFrame或Series。表示要添加的新数据。无默认。
ignore_index	接收boolean。如果输入True，会对新生成的DataFrame使用新的索引（自动产生）而忽略原来数据的索引。默认为False。
verify_integrity	接收boolean。如果输入True，那么当ignore_index为False时，会检查添加的数据索引是否冲突，如果冲突，则会添加失败。默认为False。

完成以下练习：

打开concat_append.ipynb文件练习

#已有定义好的资料集 df1, df2

#将df2合并到df1的下面，以及重置index，并打印出结果

●1.1.1 合并数据- 主键合并数据

• 主键合并

- 主键合并，即通过一个或多个键将两个数据集的行连接起来，类似于SQL中的JOIN。针对同一个主键存在两张包含不同字段的表，将其根据某几个字段——一对应拼接起来，结果集列数为两个元数据的列数和减去连接键的数量。

左表1				右表2				合并后表3					
	A	B	Key		C	D	Key		A	B	Key	C	D
1	A1	B1	k1	1	C1	D1	k1	1	A1	B1	k1	C1	D1
2	A2	B2	k2	2	C2	D2	k2	2	A2	B2	k2	C2	D2
3	A3	B3	k3	3	C3	D3	k3	3	A3	B3	k3	C3	D3
4	A4	B4	k4	4	C4	D4	k4	4	A4	B4	k4	C4	D4

●1.1.1 合并数据- 主键合并数据

- 主键合并——merge函数

- 和数据库的join一样，merge函数也有左连接（left）、右连接（right）、内连接（inner）和外连接（outer），但比起数据库SQL语言中的join和merge函数还有其自身独到之处，例如可以在合并过程中对数据集中的数据进行排序等。

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

- 可根据merge函数中的参数说明，并按照需求修改相关参数，就可以多种方法实现主键合并。

●1.1.1 合并数据- 常用参数及其说明

参数名称	说明
left	接收DataFrame或Series。表示要添加的新数据。无默认。
right	接收DataFrame或Series。表示要添加的新数据。无默认。。
how	接收inner, outer, left, right。表示数据的连接方式。默认为inner。
on	接收string或sequence。表示两个数据合并的主键（必须一致）。默认为None。
left_on	接收string或sequence。表示left参数接收数据用于合并的主键。默认为None。
right_on	接收string或sequence。表示right参数接收数据用于合并的主键。默认为None。
left_index	接收boolean。表示是否将left参数接收数据的index作为连接主键。默认为False。
right_index	接收boolean。表示是否将right参数接收数据的index作为连接主键。默认为False。
sort	接收boolean。表示是否根据连接键对合并后的数据进行排序。默认为False。
suffixes	接收接收tuple。表示用于追加到left和right参数接收数据重叠列名的尾缀默认为('_x', '_y')。

完成以下练习：

打开merge.ipynb文件练习

#依据key column合并，并打印出结果

●1.1.1 合并数据- 主键合并数据

- 主键合并——join方法

- join方法也可以实现部分主键合并的功能，但是join方法使用时，两个主键的名字必须相同。

`pandas.DataFrame.join(self, other, on=None, how='left', lsuffix="", rsuffix="", sort=False)`

- 常用参数说明如下。

参数名称	说明
other	接收DataFrame、Series或者包含了多个DataFrame的list。表示参与连接的其他DataFrame。无默认。
on	接收列名或者包含列名的list或tuple。表示用于连接的列名。默认为None。
how	接收特定string。inner代表内连接；outer代表外连接；left和right分别代表左连接和右连接。默认为inner。
lsuffix	接收string。表示用于追加到左侧重叠列名的末尾。无默认。
rsuffix	接收string。表示用于追加到右侧重叠列名的末尾。无默认。
sort	根据连接键对合并后的数据进行排序，默认为True。

1.1 pandas进行数据预处理

●1.1.1 合并数据- 重叠合并数据

- combine_first方法

- 数据分析和处理过程中若出现两份数据的内容几乎一致的情况，但是某些特征在其中一张表上是完整的，而在另外一张表上的数据则是缺失的时候，可以用combine_first方法进行重叠数据合并，其原理如下。

表8				表9				合并后表10			
	0	1	2		0	1	2		0	1	2
0	NaN	3.0	5.0	1	42	NaN	8.2	0	NaN	3.0	5.0
1	NaN	4.6	NaN	2	10	7.0	4.0	1	42	4.6	8.2
2	NaN	7.0	NaN					2	10	7.0	4.0

●1.1.1 合并数据- 重叠合并数据

combine_first方法

- combine_first的具体用法如下。

pandas.DataFrame.combine_first(other)

- 参数及其说明如下。

参数名称	说明
other	接收DataFrame。表示参与重叠合并的另一个DataFrame。无默认。

完成以下练习：

打开combine_first.ipynb文件练习

用a的数据填补b的缺失值

用b的数据填补a的缺失值



pandas清洗数据

- (1) 检测与处理重复值
- (2) 检测与处理缺失值
- (3) 检测与处理异常值

●1.2.1 清洗数据-检测与处理重复值

记录重复，即一个或者多个特征某几个记录的值完全相同

➤ 方法一是利用列表（list）去重，自定义去重函数：

```
def delRep(list1):  
    list2=[]  
    for i in list1:  
        if i not in list2:  
            list2.append(i)  
    return list2
```

➤ 方法二是利用集合（set）的元素是唯一的特性去重，如dish_set = set(dishes)。

●1.2.1 清洗数据-检测与处理重复值

记录重复

- 比较上述两种方法可以发现，方法一代码冗长。方法二代码简单了许多，但会导致数据的排列发生改变。
- pandas提供了一个名为drop_duplicates的去重方法。该方法只对DataFrame或者Series类型有效。这种方法不会改变数据原始排列，并且兼具代码简洁和运行稳定的特点。该方法不仅支持单一特征的数据去重，还能够依据DataFrame的其中一个或者几个特征进行去重操作。

pandas.DataFrame(Series).drop_duplicates(self, subset=None, keep='first', inplace=False)

参数名称	说明
subset	接收string或sequence。表示进行去重的列。默认为None，表示全部列。
keep	接收特定string。表示重复时保留第几个数据。First：保留第一个。Last：保留最后一个。False：只要有重复都不保留。默认为first。
inplace	接收boolean。表示是否在原表上进行操作。默认为False。

完成以下练习：

打开`drop_duplicates.ipynb`文件练习

使用`drop_duplicates`方法对进行去重的列进行去重

●1.2.1 清洗数据-检测与处理缺失值

- 利用isnull或notnull找到缺失值
 - 数据中的某个或某些特征的值是不完整的，这些值称为缺失值。
 - pandas提供了识别缺失值的方法isnull以及识别非缺失值的方法notnull，这两种方法在使用时返回的都是布尔值True和False。
 - 结合sum函数和isnull、notnull函数，可以检测数据中缺失值的分布以及数据中一共含有多少缺失值。
 - isnull和notnull之间结果正好相反，因此使用其中任意一个都可以判断出数据中缺失值的位置。

●1.2.1 清洗数据-检测与处理缺失值

1. 删除法

- 删除法分为删除观测记录和删除特征两种，它属于利用减少样本量来换取信息完整度的一种方法，是一种最简单的缺失值处理方法。
- pandas中提供了简便的删除缺失值的方法dropna，该方法既可以删除观测记录，亦可以删除特征。
pandas.DataFrame.dropna(self, axis=0, how='any', thresh=None, subset=None, inplace=False)
- 常用参数及其说明如下。

参数名称	说明
axis	接收0或1。表示轴向，0为删除观测记录（行），1为删除特征（列）。默认为0。
how	接收特定string。表示删除的形式。any表示只要有缺失值存在就执行删除操作。all表示当且仅当全部为缺失值时执行删除操作。默认为any。
subset	接收类array数据。表示进行去重的列行。默认为None，表示所有列/行。
inplace	接收boolean。表示是否在原表上进行操作。默认为False。

●1.2.1 清洗数据-检测与处理缺失值

• 2. 替换法

- 替换法是指用一个特定的值替换缺失值。
- 特征可分为数值型和类别型，两者出现缺失值时的处理方法也是不同的。
 - 缺失值所在特征为**数值型**时，通常利用其均值、中位数和众数等描述其集中趋势的统计量来代替缺失值。
 - 缺失值所在特征为**类别型**时，则选择使用众数来替换缺失值。

●1.2.1 清洗数据-检测与处理缺失值

2. 替换法

- pandas库中提供了缺失值替换的方法名为fillna，其基本语法如下。

pandas.DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None)

- 常用参数及其说明如下。

参数名称	说明
value	接收scalar, dict, Series或者DataFrame。表示用来替换缺失值的值。无默认。
method	接收特定string。backfill或bfill表示使用下一个非缺失值填补缺失值。pad或ffill表示使用上一个非缺失值填补缺失值。默认为None。
axis	接收0或1。表示轴向。默认为1。
inplace	接收boolean。表示是否在原表上进行操作。默认为False。
limit	接收int。表示填补缺失值个数上限，超过则不进行填补。默认为None。

完成以下练习：

打开dropna.ipynb文件练习

使用dropna删除缺失值

●1.2.1 清洗数据-检测与处理缺失值

• 3. 插值法

- 删除法简单易行，但是会引起数据结构变动，样本减少；替换法使用难度较低，但是会影响数据的标准差，导致信息量变动。在面对数据缺失问题时，除了这两种方法之外，还有一种常用的方法—插值法。
- 常用的插值法有线性插值、多项式插值和样条插值等：
 - 线性插值是一种较为简单的插值方法，它针对已知的值求出线性方程，通过求解线性方程得到缺失值。
 - 多项式插值是利用已知的值拟合一个多项式，使得现有的数据满足这个多项式，再利用这个多项式求解缺失值，常见的多项式插值法有拉格朗日插值和牛顿插值等。
 - 样条插值是以可变样条来作出一条经过一系列点的光滑曲线的插值方法，插值样条由一些多项式组成，每一个多项式都是由相邻两个数据点决定，这样可以保证两个相邻多项式及其导数在连接处连续。

●1.2.1 清洗数据-检测与处理缺失值

• 3. 插值法

- 从拟合结果可以看出多项式插值和样条插值在两种情况下拟合都非常出色，线性插值法只在自变量和因变量为线性关系的情况下拟合才较为出色。
- 而在实际分析过程中，自变量与因变量的关系是线性的情况非常少见，所以在大多数情况下，多项式插值和样条插值是较为合适的选择。
- SciPy库中的interpolate模块除了提供常规的插值法外，还提供了例如在图形学领域具有重要作用的重心坐标插值（BarycentricInterpolator）等。在实际应用中，需要根据不同的场景，选择合适的插值方法。

●1.2.1 清洗数据-检测与处理异常值

- 异常值

- 异常值是指数据中个别值的数值明显偏离其余的数值，有时也称为离群点，检测异常值就是检验数据中是否有录入错误以及是否含有不合理的数据。
- 异常值的存在对数据分析十分危险，如果计算分析过程的数据有异常值，那么会对结果会产生不良影响，从而导致分析结果产生偏差乃至错误。
- 常用的异常值检测主要为**3 σ 原则**和**箱线图分析**两种方法。

1.2 pandas进行数据预处理

●1.2.1 清洗数据-检测与处理异常值

• 1. 3σ原则

- 3σ原则又称为拉依达法则。该法则就是先假设一组检测数据只含有随机误差，对原始数据进行计算处理得到标准差，然后按一定的概率确定一个区间，认为误差超过这个区间的就属于异常值。
- 这种判别处理方法仅适用于对正态或近似正态分布的样本数据进行处理，如下表所示，其中σ代表标准差，μ代表均值， $x=\mu$ 为图形的对称轴。
- 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。

数值分布		在数据中的占比
数值分布	在数据中的占比	0.6827
$(\mu - \sigma, \mu + \sigma)$	0.6827	
$(\mu - 2\sigma, \mu + 2\sigma)$	0.9545	
$(\mu - 3\sigma, \mu + 3\sigma)$	0.9973	0.9545
数值分布	在数据中的占比	
$(\mu - \sigma, \mu + \sigma)$	0.6827	
$(\mu - 2\sigma, \mu + 2\sigma)$	0.9545	0.9973
$(\mu - 3\sigma, \mu + 3\sigma)$	0.9973	
数值分布	在数据中的占比	
$(\mu - \sigma, \mu + \sigma)$	0.6827	0.9973
$(\mu - 2\sigma, \mu + 2\sigma)$	0.9545	
$(\mu - 3\sigma, \mu + 3\sigma)$	0.9973	

●1.2.1 清洗数据-检测与处理异常值

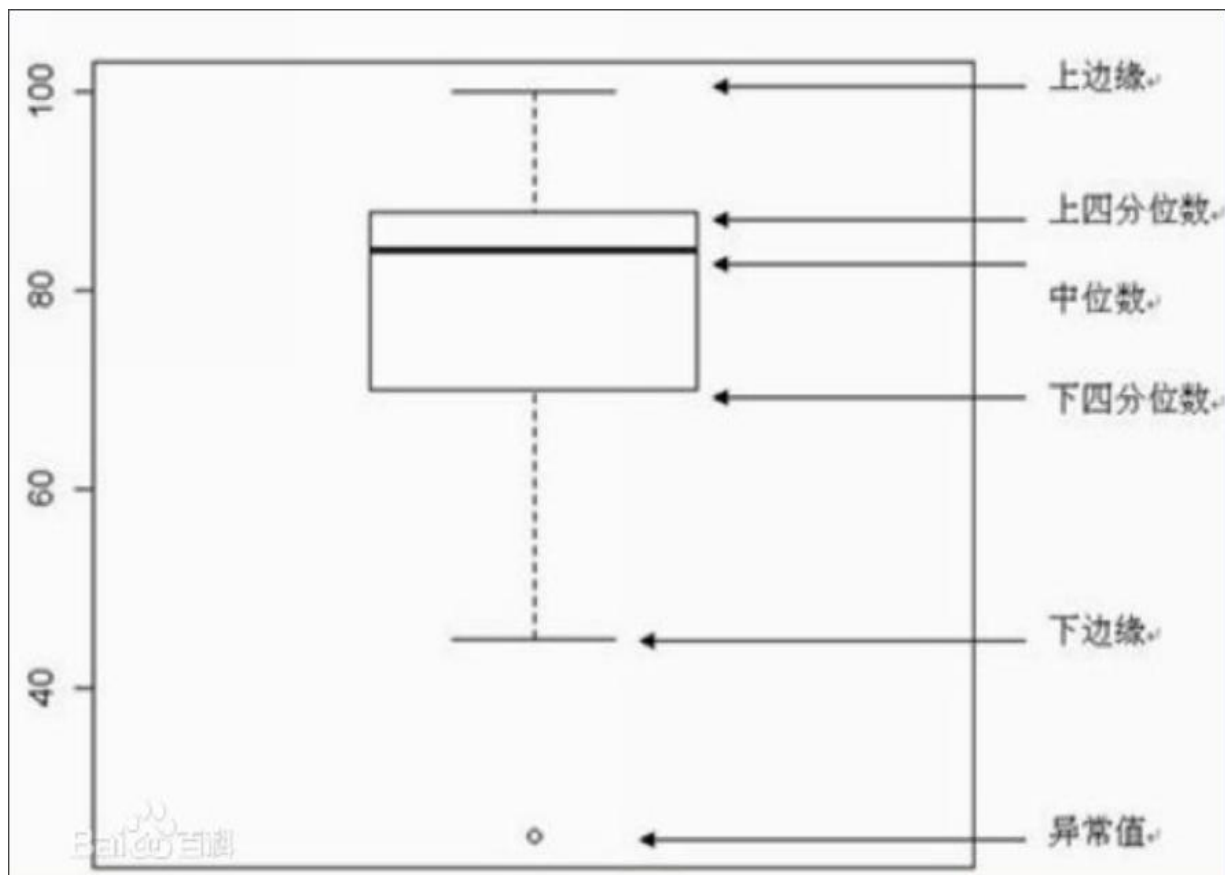
• 2.箱线图分析

- 箱型图提供了识别异常值的一个标准，即异常值通常被定义为小于 $QL - 1.5IQR$ 或大于 $QU + 1.5IQR$ 的值。
 - QL 称为下四分位数，表示全部观察值中有四分之一的数据取值比它小。
 - QU 称为上四分位数，表示全部观察值中有四分之一的数据取值比它大。
 - IQR 称为四分位数间距，是上四分位数 QU 与下四分位数 QL 之差，其间包含了全部观察值的一半。
- 箱线图依据实际数据绘制，真实、直观地表现出了数据分布的本来面貌，且没有对数据做任何限制性要求，其判断异常值的标准以四分位数和四分位数间距为基础。
- 四分位数给出了数据分布的中心、散布和形状的某种指示，箱形图判断异常值的标准以四分位数和四分位距为基础，箱线图识别异常值的结果比较客观，因此在识别异常值方面具有一定的优越性。

1.2pandas进行数据预处理

●1.2.1 清洗数据-检测与处理异常值

• 2.箱线图分析



●1.2.1 清洗数据-检测与处理异常值

• 异常值

在数据预处理时，异常值是否剔除，需视具体情况而定，因为有些异常值可能蕴含着有用的信息。异常值处理常用方法见表 4-3。

表4-3 异常值处理常用方法

异常值处理方法	方法描述
删除含有异常值的记录	直接将含有异常值的记录删除
视为缺失值	将异常值视为缺失值，利用缺失值处理的方法进行处理
平均值修正	可用前后两个观测值的平均值修正该异常值
不处理	直接在具有异常值的数据集上进行挖掘建模

●1.2.1 标准化数据-离差标准化数据

- 离差标准化公式

离差标准化是对原始数据的一种线性变换，结果是将原始数据的数值映射到[0,1]区间之间，转换公式为

$$X^* = \frac{X - \min}{\max - \min}$$

其中max为样本数据的最大值，min为样本数据的最小值，max-min为极差。离差标准化保留了原始数据值之间的联系，是消除量纲和数据取值范围影响最简单的方法。

●1.2.1 标准化数据-离差标准化数据

- 离差标准化的特点

- 数据的整体分布情况并不会随离差标准化而发生改变，原先取值较大的数据，在做完离差标准化后的值依旧较大。
- 当数据和最小值相等的时候，通过离差标准化可以发现数据变为0。
- 若数据极差过大就会出现数据在离差标准化后数据之间的差值非常小的情况。
- 同时，还可以看出离差标准化的缺点：若数据集中某个数值很大，则离差标准化的值就会接近于0，并且相互之间差别不大。若将来遇到超过目前属性[min,max]取值范围的时候，会引起系统出错，这时便需要重新确定min和max。

●1.2.1 标准化数据-标准差标准化数据

- 标准差标准化的公式及特点

标准差标准化也叫零均值标准化或分数标准化，是当前使用最广泛的数据标准化方法。经过该方法处理的数据均值为0，标准差为1，转化公式如下。

$$X^* = \frac{X - \bar{X}}{\delta}$$

其中 \bar{X} 为原始数据的均值， δ 为原始数据的标准差。标准差标准化后的值区间不局限于[0,1]，并且存在负值。同时也不难发现，标准差标准化和离差标准化一样不会改变数据的分布情况。

1.2 pandas进行数据预处理

●1.2.1 转换数据-哑变量处理类别数据

• 哑变量处理

- 数据分析模型中有相当一部分的算法模型都要求输入的特征为数值型，但实际数据中特征的类型不一定只有数值型，还会存在相当一部分的类别型，这部分特征需要经过哑变量处理才可以放入模型之中。哑变量处理的原理示例如图。

哑变量处理前		哑变量处理后					
	城市		城市_广州	城市_上海	城市_杭州	城市_北京	城市_深圳
1	广州	1	1	0	0	0	0
2	上海	2	0	1	0	0	0
3	杭州	3	0	0	1	0	0
4	北京	4	0	0	0	1	0
5	深圳	5	0	0	0	0	1
6	北京	6	0	0	0	1	0
7	上海	7	0	1	0	0	0
8	杭州	8	0	0	1	0	0
9	广州	9	1	0	0	0	0
10	深圳	10	0	0	0	0	1

1.2 pandas进行数据预处理

●1.2.1 转换数据-哑变量处理类别数据

- get_dummies函数

➤ Python中可以利用pandas库中的get_dummies函数对类别型特征进行哑变量处理。

pandas.get_dummies(data, prefix=None, prefix_sep='_', dummy_na=False, columns=None, sparse=False, drop_first=False)

参数名称	说明
data	接收array、DataFrame或者Series。表示需要哑变量处理的数据。无默认。
prefix	接收string、string的列表或者string的dict。表示哑变量化后列名的前缀。默认为None。
prefix_sep	接收string。表示前缀的连接符。默认为 '_' 。
dummy_na	接收boolean。表示是否为Nan值添加一列。默认为False。
columns	接收类似list的数据。表示DataFrame中需要编码的列名。默认为None，表示对所有object和category类型进行编码。
sparse	接收boolean。表示虚拟列是否是稀疏的。默认为False。
drop_first	接收boolean。表示是否通过从k个分类级别中删除第一级来获得k-1个分类级别。默认为False。

●1.2.1 转换数据-哑变量处理类别数据

- 哑变量处理的特点

- 虚拟变量，也叫哑变量和离散特征编码，可用来表示分类变量、非数量因素可能产生的影响。
- 离散特征的编码分为两种情况：
 - 1、离散特征的取值之间没有大小的意义，比如color: [red,green],那么就使用one-hot编码
 - 2、离散特征的取值有大小的意义，比如size:[X,XL,XXL],那么就使用数值的映射{X:1,XL:2,XXL:3}
- 对于一个类别型特征，若其取值有m个，则经过哑变量处理后就变成了m个二元特征，并且这些特征互斥，每次只有一个激活，这使得数据变得稀疏。
- 对类别型特征进行哑变量处理主要解决了部分算法模型无法处理类别型数据的问题，这在一定程度上起到了扩充特征的作用。由于数据变成了稀疏矩阵的形式，因此也加速了算法模型的运算速度。

●1.2.1 转换数据-哑变量处理类别数据

1.什么是One-Hot编码以及优点？

One-Hot编码，又称为一位有效编码，主要是采用N位状态寄存器来对N个状态进行编码，每个状态都由他独立的寄存器位，并且在任意时候只有一位有效。

One-Hot编码是分类变量作为二进制向量的表示。这首先要求将分类值映射到整数值。然后，每个整数值被表示为二进制向量，除了整数的索引之外，它都是零值，它被标记为1。

解决了分类器不好处理属性数据的问题，让特征之间的距离计算更加合理

在一定程度上也起到了扩充特征的作用，比如性别本身是一个特征，经过one hot编码以后，就变成了男或女两个特征。

将离散特征的取值扩展到了欧式空间，离散特征的某个取值就对应欧式空间的某个点。

●1.2.1 转换数据-哑变量处理类别数据

1.什么是One-Hot编码？

假设我们有一个带有' red' 和' green' 值的标签序列。我们可以将' red' 的整数值分配为0, ' green' 的整数值为1。只要我们总是将这些数字分配给这些标签, 这称为整数编码。一致性是重要的, 所以我们可以稍后反转编码, 并从整数值获取标签。

接下来, 我们可以创建一个二进制向量来表示每个整数值。对于2个可能的整数值, 向量的长度为2。

编码为0的“红色”标签将用二进制向量[1,0]表示, 其中第0个索引被标记为值1。然后, 编码为1的“绿色”标签将用一个二进制向量[0, 1], 其中第一个索引被标记为1。

如果我们有序列:

```
'red', 'red', 'green'.
```

我们可以用整数编码来表示它:

```
0, 0, 1
```

而One-Hot编码就为:

```
1      2      3  
[1, 0] [1, 0] [0, 1]
```

六. 什么情况下(不)需要归一化?

- 需要: 基于参数的模型或基于距离的模型, 都是要进行特征的归一化。
- 不需要: 基于树的方法是不需要进行特征的归一化, 例如随机森林, bagging 和 boosting等。

1.2pandas进行数据预处理

●1.2.1 转换数据-哑变量处理类别数据

独热编码实现方法比较

1、pandas自带的get_dummies()【适合含有字符串类型的数据】

好处： * 1.本身就是 pandas 的模块，所以对 DataFrame 类型兼容很好。

* 2.无论你的列是字符型还是数字型都可以进行二值编码。

* 3.能根据用户指定，自动生成二值编码后的变量名。

在scikit-learn中有个LabelEncoder类，可方便实现以上功能：

```
1 from sklearn.preprocessing import LabelEncoder
2 class_le = LabelEncoder()
3 y = class_le.fit_transform(df['classlabel'].values)
4 y
5
6 -----
7 array([0, 1, 0])
```

●1.2.1 转换数据-离散化连续型数据 离散化

- 某些模型算法，特别是某些分类算法如ID3决策树算法和Apriori算法等，要求数据是离散的，此时就需要将连续型特征（数值型）变换成离散型特征（类别型）。
- 连续特征的离散化就是在数据的取值范围内设定若干个离散的划分点，将取值范围划分为一些离散化的区间，最后用不同的符号或整数值代表落在每个子区间中的数据值。
- 因此离散化涉及两个子任务，即确定分类数以及如何将连续型数据映射到这些类别型数据上。其原理如图。

离散化处理前		离散化处理后的	
	年龄		年龄
1	18	1	(17.955, 27]
2	23	2	(17.955, 27]
3	35	3	(27, 36]
4	54	4	(45, 54]
5	42	5	(36, 45]
6	21	6	(17.955, 27]
7	60	7	(54, 63]
8	63	8	(54, 63]
9	41	9	(36, 45]
10	38	10	(36, 45]

●1.2.1 转换数据-离散化连续型数据

• 1. 等宽法

- 将数据的值域分成具有相同宽度的区间，区间的个数由数据本身的特点决定或者用户指定，与制作频率分布表类似。pandas提供了cut函数，可以进行连续型数据的等宽离散化，其基础语法格式如下。

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

参数名称	说明
x	接收数组或Series。代表需要进行离散化处理的数据。无默认。
bins	接收int, list, array, tuple。若为int，代表离散化后的类别数目；若为序列类型的数据，则表示进行切分的区间，每两个数间隔为一个区间。无默认。
right	接收boolean。代表右侧是否为闭区间。默认为True。
labels	接收list, array。代表离散化后各个类别的名称。默认为空。
retbins	接收boolean。代表是否返回区间标签。默认为False。
precision	接收int。显示的标签的精度。默认为3。

●1.2.1 转换数据-离散化连续型数据

• 1. 等宽法

- 使用等宽法离散化的缺陷为：等宽法离散化对数据分布具有较高要求，若数据分布不均匀，那么各个类的数目也会变得非常不均匀，有些区间包含许多数据，而另外一些区间的数据极少，这会严重损坏所建立的模型。

●1.2.1 转换数据-离散化连续型数据

• 2. 等频法

- cut函数虽然不能够直接实现等频离散化，但是可以通过定义将相同数量的记录放进每个区间。
- 等频法离散化的方法相比较于等宽法离散化而言，避免了分类分布不均匀的问题，但同时却也有可能将数值非常接近的两个值分到不同的区间以满足每个区间中固定的数据个数。采用qcut方法

●1.2.1 转换数据-离散化连续型数据

3. 基于聚类分析的方法

- 一维聚类的方法包括两个步骤：
 - 将连续型数据用聚类算法（如K-Means算法等）进行聚类。
 - 处理聚类得到的簇，将合并到一个簇的连续型数据做同一标记。
- 聚类分析的离散化方法需要用户指定簇的个数，用来决定产生的区间数。
- k-Means聚类分析的离散化方法可以很好地根据现有特征的数据分布状况进行聚类，但是由于k-Means算法本身的缺陷，用该方法进行离散化时依旧需要指定离散化后类别的数目。此时需要配合聚类算法评价方法，找出最优的聚类簇数目。

●1.2.1 数据规约

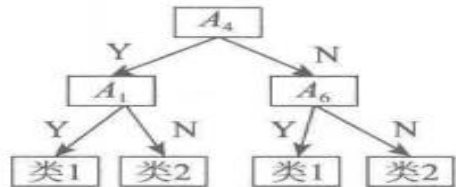
- 分为属性规约和数值规约
- 在大数据集上进行复杂的数据分析和挖掘需要很长的时间，数据规约产生更小但保持数据完整性的新数据集，在规约后的数据集上进行分析 and 挖掘将更有效率
- 意义在于：
 - 降低无效，错误数据对建模的影响，提高建模的准确性
 - 少量且具有代表性的数据将大幅缩减数据挖掘所用的时间
 - 减低储存数据成本

1.2 pandas进行数据预处理

●属性规约

- 通过属性合并来创建新属性维数，或者直接删除不相关的属性来减少数据维数，从而提高数据分析与挖掘的速度，目的是寻找最小的属性子集并确保新数据子集的概率分布尽可能的接近原来的数据集概率分布，常用方法见下表

表4-6 属性规约常用方法

属性规约方法	方法描述	方法解析
合并属性	将一些旧属性合为新属性	初始属性集: $\{A_1, A_2, A_3, A_4, B_1, B_2, B_3, C\}$ $\{A_1, A_2, A_3, A_4\} \rightarrow A$ $\{B_1, B_2, B_3\} \rightarrow B$ \Rightarrow 规约后属性集: $\{A, B, C\}$
逐步向前选择	从一个空属性集开始，每次从原来属性集合中选择一个当前最优的属性添加到当前属性子集中。直到无法选择出最优属性或满足一定阈值约束为止	初始属性集: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ $\{\} \Rightarrow \{A_1\} \Rightarrow \{A_1, A_4\}$ \Rightarrow 规约后属性集: $\{A_1, A_4, A_6\}$
逐步向后删除	从一个全属性集开始，每次从当前属性子集中选择一个当前最差的属性并将其从当前属性子集中消去。直到无法选择出最差属性为止或满足一定阈值约束为止	初始属性集: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ $\Rightarrow \{A_1, A_3, A_4, A_5, A_6\} \Rightarrow \{A_1, A_4, A_5, A_6\}$ \Rightarrow 规约后属性集: $\{A_1, A_4, A_6\}$
决策树归纳	利用决策树的归纳方法对初始数据进行分类归纳学习，获得一个初始决策树，所有没有出现在这个决策树上的属性均可认为是无关属性，因此将这些属性从初始集合中删除，就可以获得一个较优的属性子集	初始属性集: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$  \Rightarrow 规约后属性集: $\{A_1, A_4, A_6\}$
主成分分析	用较少的变量去解释原始数据中的大部分变量，即将许多相关性很高的变量转化成彼此相互独立或不相关的变量	详见下面计算步骤

1.2 pandas进行数据预处理

- 数值规约
- 指的是通过选择替代的，较小的数据来减少数据量，包括有参数，无参数方法
- 有参数：使用模型来评估数据，只存放参数，不存放实际数据 eg：线性回归/多元回归
- 无参数：存放实际数据，直方图/聚类/抽样

1.2pandas进行数据预处理

● 总结

