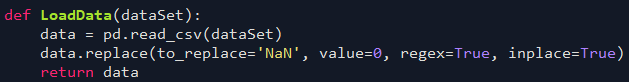
**数据分析与处理部分实践作业 1#**

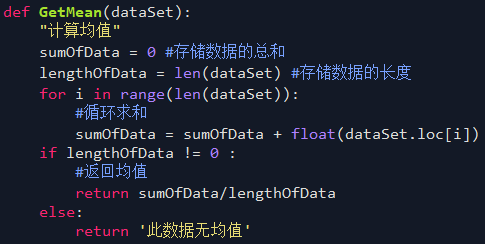
# 实现数值型数列均值、方差、分位数的计算

## 1.1 均值计算

1、传入数据：

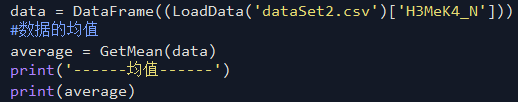


2、然后是构造计算均值的函数：



3、步骤为：（1）传入要计算列的参数；（2）设置变量存储该列数据的总和，以及设置变量存储数据的长度；（3）长度调用len()函数；（4）数据总和循环整个长度（也可以设置循环当有值存在时，长度加1，同时累加数据的和）；（5）数据总和除以数据长度得到均值（需满足长度不为0）。

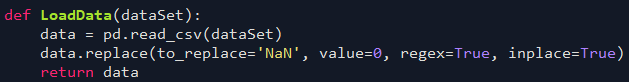
4、调用函数，得到计算结果：



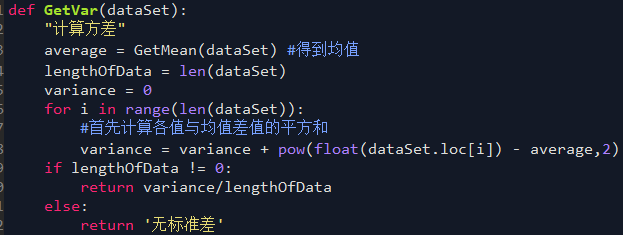


## 1.2 方差计算

1、传入数据：

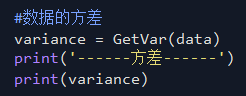


2、构造计算方差函数：



3、步骤为：（1）传入要计算的数据列；（2）设置变量存储该列的均值，调用之前构造的均值计算函数；（3）设置变量存储数据列中每个值与均值差值的平方和，通过循环整个数据列的长度，其中平方和计算调用了pow(x,2)函数，也可以使用x\*\*2计算；（4）若数据长度不为0返回方差结果。

4、调用函数，得到计算结果：

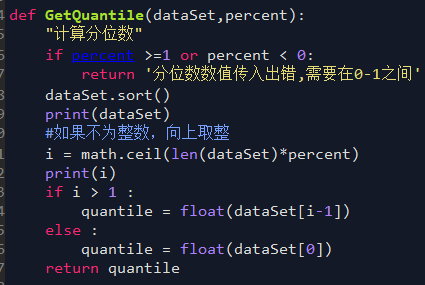




## 1.3 分位数计算

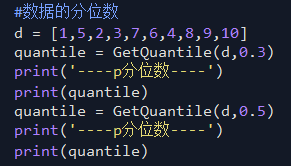
1、构造数据：随机输入乱序的列表

2、构造分位数计算函数：

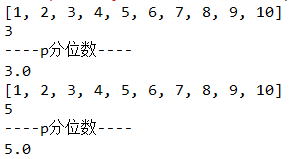


3、步骤为：（1）传入参数为数据列和要计算的分位数的数值；（2）判断分位数是否输入正确，错误则直接返回；（3）对传入的数据列进行升序排序；（4）确定该分位数的位置，即长度与分位数的乘积，若位置坐标不为整数时，向上取整；（5）根据位置坐标来选取分位数的数值结果，位置大于1时在列表中的坐标为实际位置减1，位置小于1时，即第一个点的位置。

4、调用函数，得到计算结果：



如图计算了两个结果：

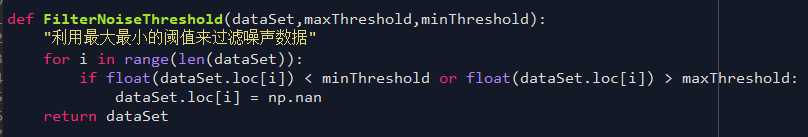


# 实现2种噪声数据过滤和缺失值补全方法

## 2.1 噪声数据过滤

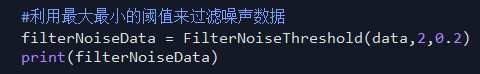
### 2.1.1 最大最小阈值过滤噪声数据

1、传入数据列dataframe类型或者其他的如列表格式也可以，只是对每个数据值的操作不同，构造函数：

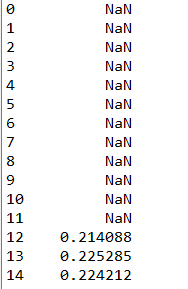


2、步骤为：（1）传入参数数据列、最大最小的阈值；（2）通过循环整个数据列的长度，对每个数据进行数值大小判断，当该数据小于最小阈值或者大于最大阈值时，数据置空，最后返回数据列。

3、函数调用，以及输出结果：



即我的目的是过滤掉大于2且小于0.2的数据：



### 2.1.2 等频装箱均值填充过滤噪声数据

1、等频装箱：将数据排序后，分为不同箱子，在每个箱子中数据的个数一样，使用每个箱子的均值来填充每个箱子中的原本的值，进行噪声平滑处理。

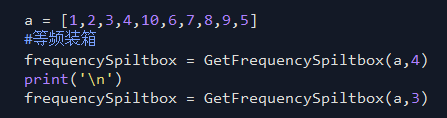
2、传入数据列：

3、构造等频装箱函数：



3、步骤为：（1）传入数据列，以及需要装箱的箱子数目；（2）对数据列进行排序，计算长度，然后取余每个箱子的个数，无余数则说明正好分箱，有余数说明不能正好分箱；（3）字典存储分箱的结果；（4）若箱子不能正好分完，说明盒子个数为len(data)+1，且对于第len(data)+1个箱子需要特殊处理：对前面的每个箱子依次间隔箱子中的数值个数来截取数值，然后变量j每次增加box的值，对于最后一个箱子则赋值当前位置到最后的值，均值填充每个箱子的值；（5）若箱子刚好分完，仅进行（4）中的第一部分的操作。

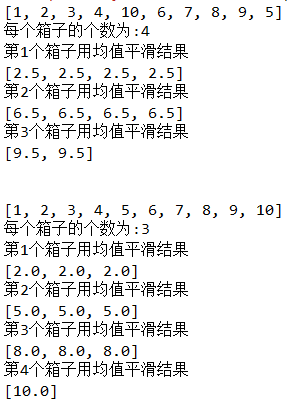
4、调用函数，打印结果：



箱子数为4时，装箱结果为：[1,2,3,4],[5,6,7,8],[9,10]

箱子数为3时，装箱结果为:[1,2,3],[4,5,6],[7,8,9],[10]

再利用均值平滑结果为：

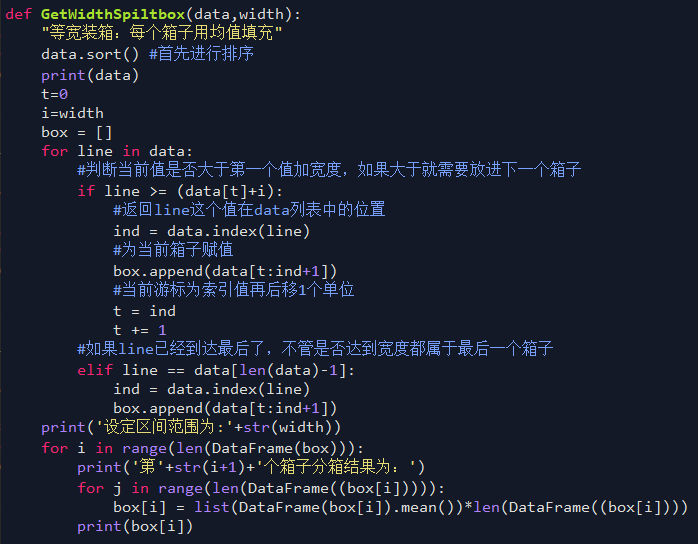


### 2.2.3 等宽装箱均值填充过滤噪声数据

1、等宽装箱：将数据排序后，分为不同箱子，在每个箱子中数据最大最小的差值是一样的，即箱子的宽度相同，使用每个箱子的均值来填充每个箱子中的原本的值，进行噪声平滑处理。

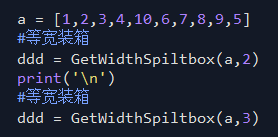
2、传入数据列：

3、构造等宽装箱函数：



4、步骤为：（1）首先传入数据列和箱子的宽度；（2）用列表来存储装箱结果；（3）对整个数据列进行循环操作，判断当前的数值是否满足小于宽度加上第一个值；（4）如果当前值已经超过了第一个位置的值加上宽度了，说明属于下一个箱子了，因此当前箱子就是第一个值到当前位置之前的列表切片，将此切片赋值给box即可，然后游标后移当前索引值加1的位置；（5）循环结束的判断：如果此时循环已经到达了数据的最后一个值了，不管它是否满足第一个值加宽度，都属于最后一个箱子；（6）对于每个箱子使用箱子中均值来替换其中的数值。

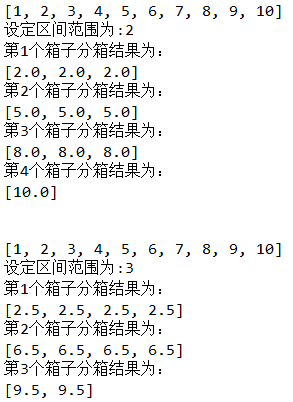
5、调用函数，打印装箱结果：



Width = 2时，装箱结果为[1,2,3],[4,5,6],[7,8,9],[10]:

Width = 3 时，装箱结果为[1,2,3,4],[5,6,7,8],[9,10]:

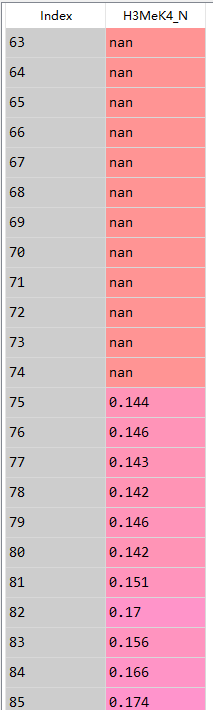
均值平滑噪声处理结果为：



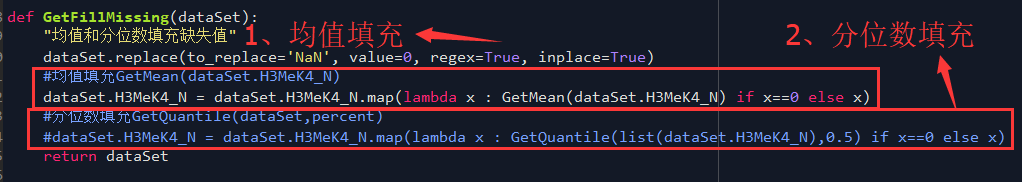
## 2.2缺失值的填充

### 2.2.1 均值/分位数填充缺失值

1、传入数据，由图所示传入含有确实的一列数据：



2、构造函数填充缺失值：

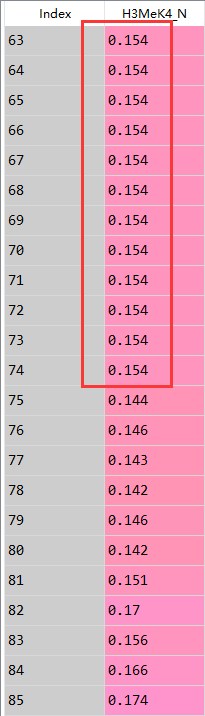


3、步骤为：（1）传入需要填充的数据列参数；（2）首选将数据中的缺失值替换为0；（3）均值填充：如果当前值为0，则调用均值函数替换当前值；（4）分位数填充：如果当前值为0，则调用分位数函数替换当前值；（5）返回缺失值补全的数据列。

4、调用函数，显示结果：

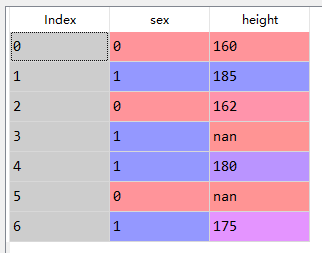


如图所示，之前的缺失值替换为了数据列的均值：



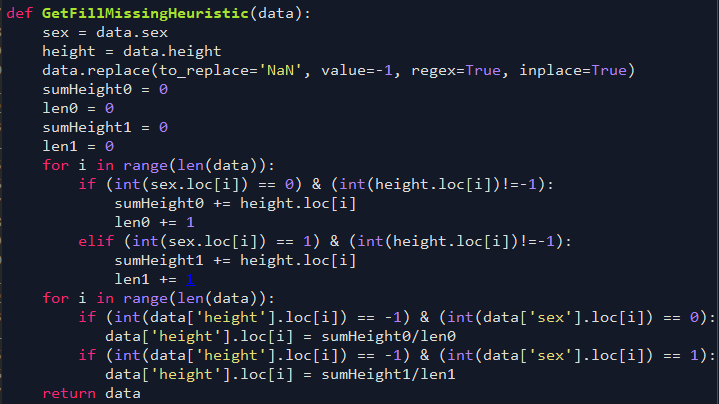
### 2.2.2 启发式补全

1、传入dataframe类型的数据，0代表女性，1代表男性：



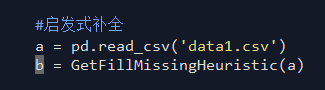
2、启发式补全：对于其中3号男性和5号女性使用启发式补全缺失的身高，想要用女性的平均身高填充5号身高，用男性的平均身高填充3号的身高。

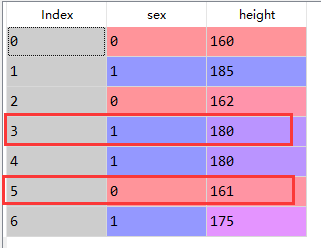
3、构造函数：



4、步骤为：（1）首先填充缺失值为0；（2）循环所有的数据，对于性别为女的身高求一次平均值，对于身高为男性的求一次平均值；（3）再循环所有的数据，给对应性别的身高缺失的数据填充对应的性别的身高均值。

5、调用函数，结果为：





# 实现2种数据离散化、数据数值化、数据归一化方法

## 3.1 数据离散化

### 3.1.1 等频装箱

1、等频装箱：将数据排序后，分为不同箱子，在每个箱子中数据的个数一样。

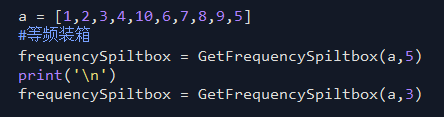
2、传入数据列：

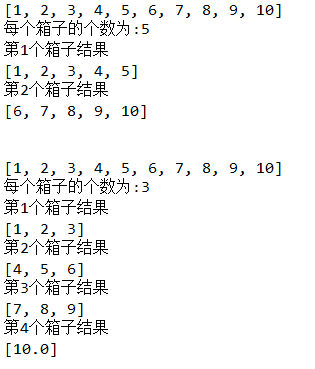
3、构造函数：



4、步骤为：步骤为：（1）传入数据列，以及需要装箱的箱子数目；（2）对数据列进行排序，计算长度，然后取余每个箱子的个数，无余数则说明正好分箱，有余数说明不能正好分箱；（3）字典存储分箱的结果；（4）若箱子不能正好分完，说明盒子个数为len(data)+1，且对于第len(data)+1个箱子需要特殊处理：对前面的每个箱子依次间隔箱子中的数值个数来截取数值，然后变量j每次增加box的值，对于最后一个箱子则赋值当前位置到最后的值；（5）若箱子刚好分完，仅进行（4）中的第一部分的操作。

5、调用函数，结果打印：



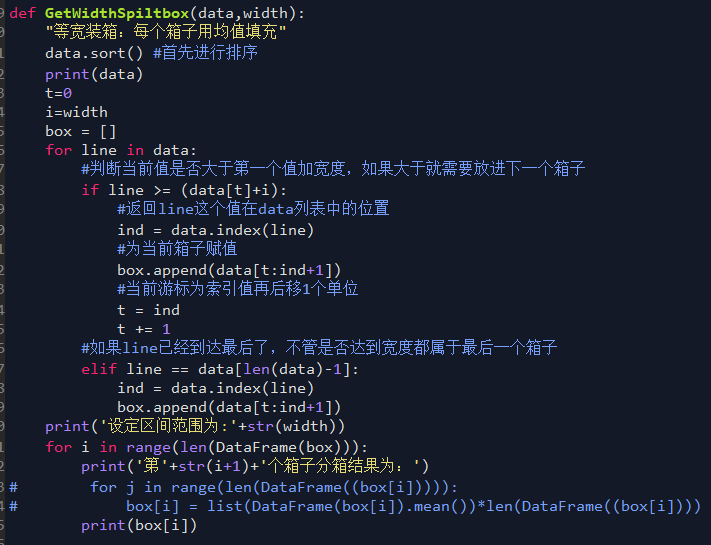


### 3.1.2 等宽装箱

1、等宽装箱：将数据排序后，分为不同箱子，在每个箱子中数据最大最小的差值是一样的，即箱子的宽度相同。

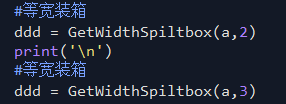
2、传入数据列：

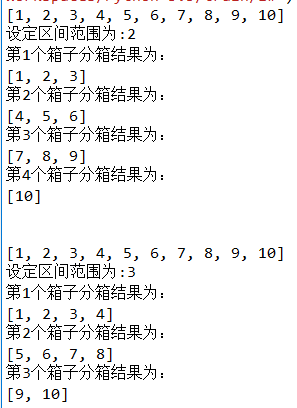
3、构造函数：



4、步骤为：（1）首先传入数据列和箱子的宽度；（2）用列表来存储装箱结果；（3）对整个数据列进行循环操作，判断当前的数值是否满足小于宽度加上第一个值；（4）如果当前值已经超过了第一个位置的值加上宽度了，说明属于下一个箱子了，因此当前箱子就是第一个值到当前位置之前的列表切片，将此切片赋值给box即可，然后游标后移当前索引值加1的位置；（5）循环结束的判断：如果此时循环已经到达了数据的最后一个值了，不管它是否满足第一个值加宽度，都属于最后一个箱子。

5、调用函数，打印结果：



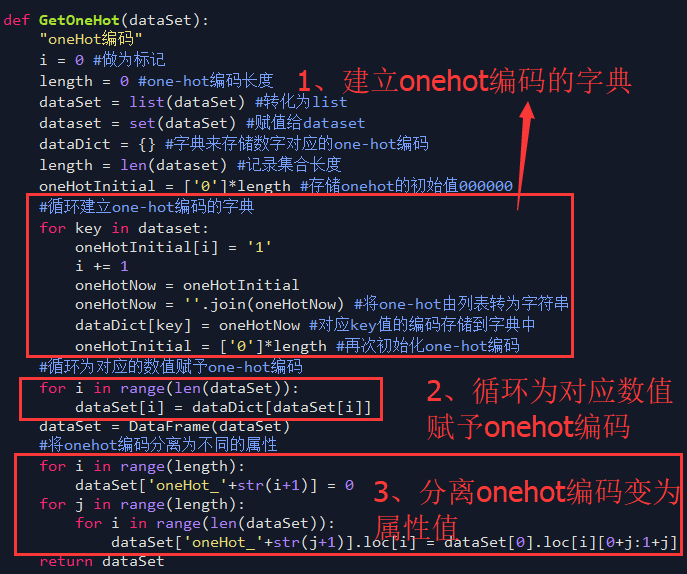


## 3.2 数据数值化

### 3.2.1 one-hot编码

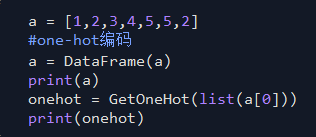
1、传入数据列：

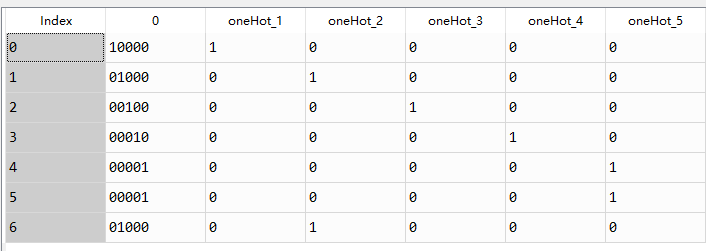
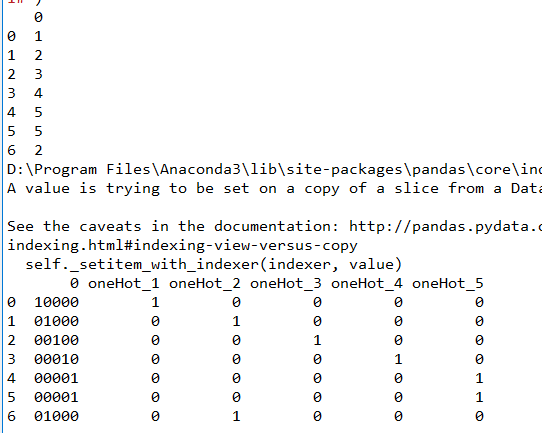
2、构造函数：



3、步骤为：（1）首先将数据列转为一个集合，集合的长度就是onehot编码的长度，初始为集合长度个‘0’；（2）构建一个字典存储onehot编码；（3）对于数据列对应的集合中的数值：1,2,3,4……，就是在对应的位置上，该处的值由‘0’变为‘1’，因此循环所有数据集合，给集合中的每个数值进行onehot编码，并记录对应的键值；（4）字典写入完成后，循环整个数据列表，为对应数值赋予onehot编码；（5）操作已经完成onehot编码，然后分离所有的编码，变成相应的属性。

4、调用函数，打印结果：

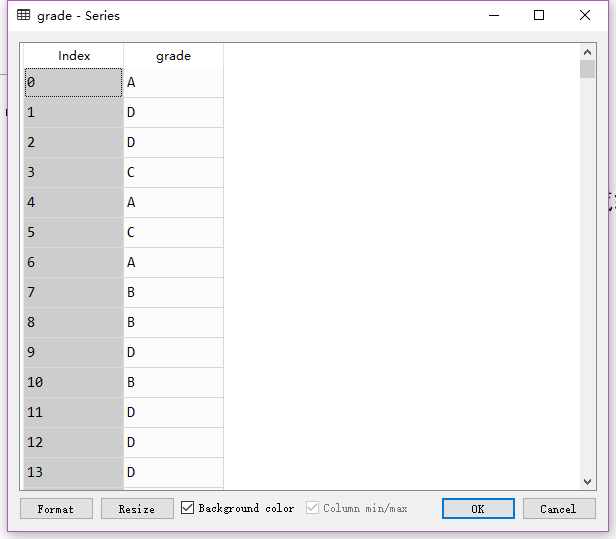




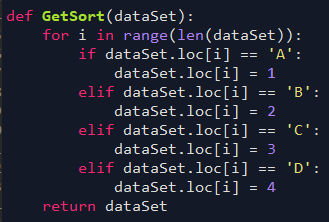
### 3.2.2 排序编码

1、排序编码通常是一些等级进行数值化，比如成绩的优良中差、ABCD转为排序：1234或者4321。

2、数据的传入，调用Python随机函数生成了一列ABCD的数据列，代表成绩的等级：

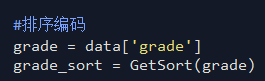


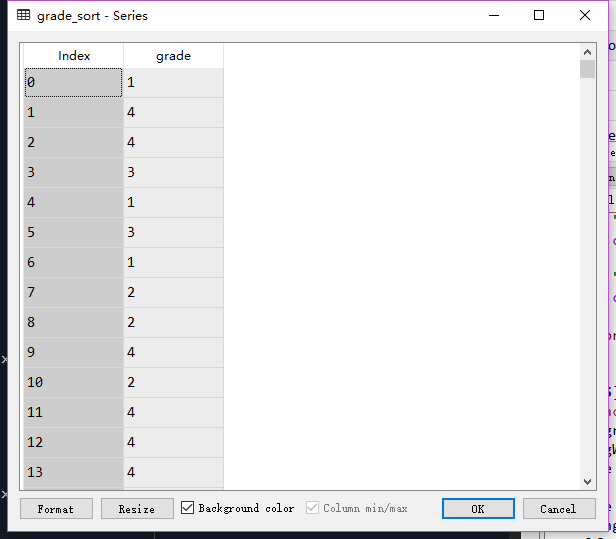
3、函数构造：



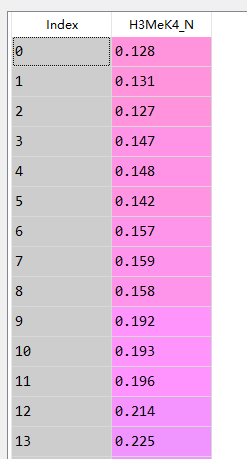
4、步骤为：（1）传入数据列；（2）循环整个数据长度，分别对 ABCD赋值为1234.

5、调用函数，结果为：



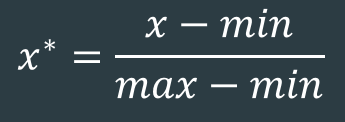


## 3.3 数据规范化

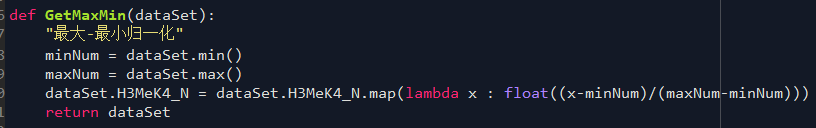
传入数据列：

### 3.3.1 最大最小归一化

1、公式：𝑥



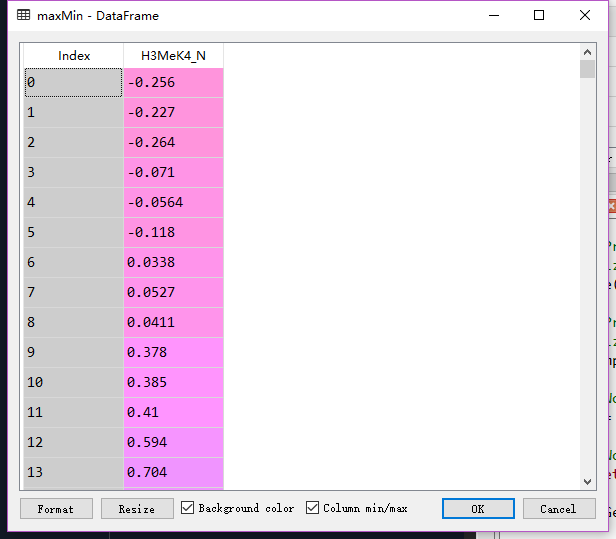
2、函数构造：



3、步骤为：（1）传入数据列参数；（2）分别计算最大最小值（循环整个数据列长度，通过比较大小找最值也可以）；（3）利用公式对该列数据全部map上最大最小归一化结果。

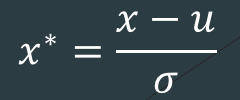
4、调用函数，结果为：



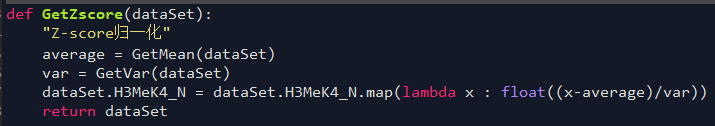


### 3.3.2 Z-Score标准化

1、公式：



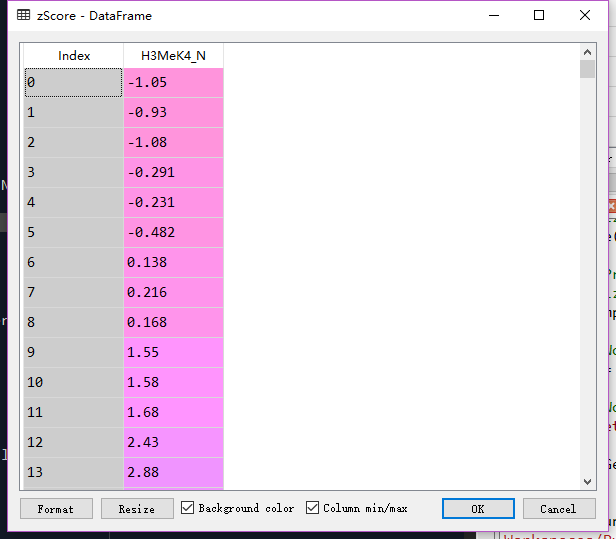
2、函数构造：



3、步骤为：（1）传入数据列参数；（2）获取均值和方差；（3）根据公式对每个值都map上对应的z-score规范化的值。

4、调用函数，结果为：

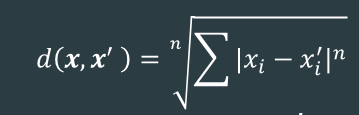




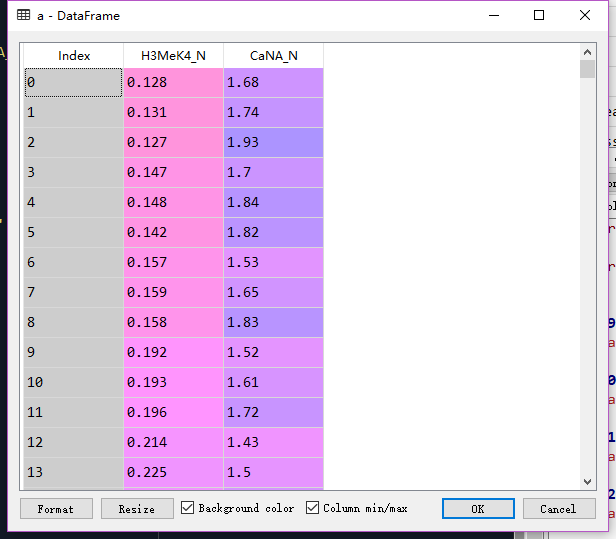
# 实现两种相似度计算方法

## 相似度--距离度量

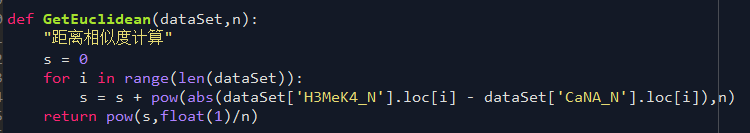
1. 公式：



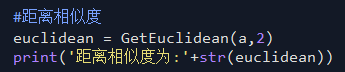
2、传入数据格式：



1. 函数构造：



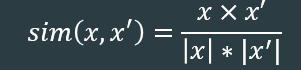
1. 步骤为：（1）传入数据列参数，以及属性有几个；（2）循环所有的数据列长度，对两列的值依次计算差的绝对值，再求该值的n次方，设置变量来求和；（3）对计算的和开n次根，即1/n次幂。
2. 调用函数，打印结果：



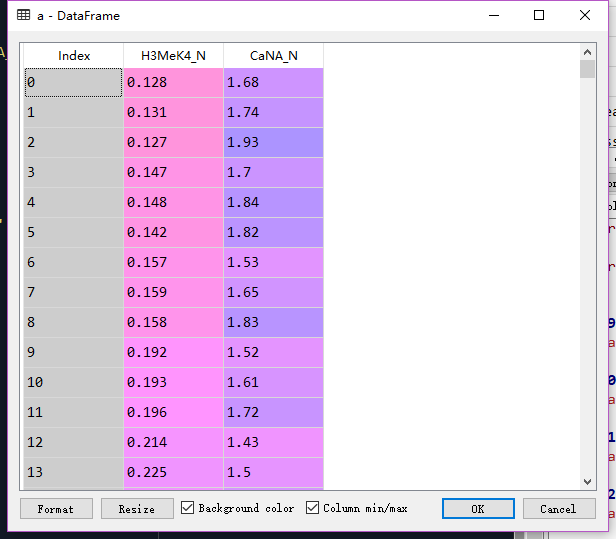


## 相似度--余弦相似度

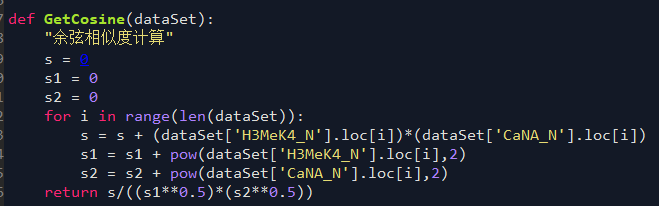
1、 公式：



2、传入数据格式：

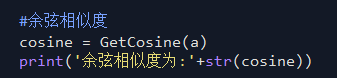


3、函数构造：



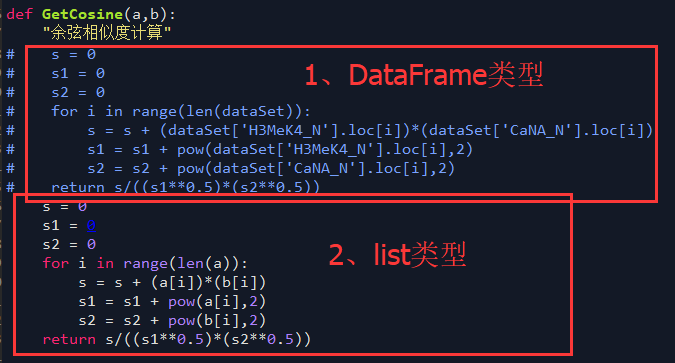
4、 步骤为：（1）传入数据列参数；（2）循环所有的数据列长度，对两列的值依次计算平方和再开根号就是距离；（3）对两列值计算乘积的平方和；（4）代入余弦相似度公式，返回结果。

1. 调用函数，打印结果：

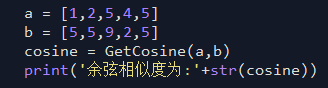


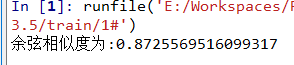


6、列表格式：文中有些使用了dataframe的结构，但是列表list也是可以的，以余弦相似度为例：



注释掉上面的代码，重写余弦相似度计算函数，原理一样，只是传入的数据结构不同：





# 心得体会

本次实验的作业是进行数据的预处理的常用方法的实现，在等频和等宽装箱上花的间比较多，原理最简单，但是实现起来却最麻烦，其原因是装箱的过程对数据的处理是一个动态的过程，每一次操作都会影响后续的装箱操作。

而对于其他比如均值、方差等等，数据是摆在那里的，需要做的就是直接写函数来计算当前数值，只要搞定了算法原理，实现起来就很容易了。

本次实现过程使用的数据结构有**list**、**dict**、**dataframe**，在实际应用中，在数据量大的时候，利用dataframe数据结构处理数据非常实用，也会很方便。