# 数据分析第一次作业

## 1. 实现均值，方差，分位数的计算

a,均值计算,函数参数为可变参数。数据源为：nums = range(1, 21)

```
def cal_mean(*nums):
    sum = 0
    for i in nums:
        sum += i
    mean = sum / len(nums)
    return mean
```

执行结果为：10.5

b,方差计算 函数为可变参数 数据源不变

```
def cal_var(*nums):
    mean = cal_mean(*nums)
    sumvar = 0
    for j in nums:
        sumvar += (j - mean) ** 2
    var = sumvar / len(nums)
    return var
```

执行结果为：33.25

c,分位数计算

```python
def cal_quantile(n,*nums):
    #nums = range(1,21)
    #分位数的个数
    #n = 3

    #从列表上里边随机抽取ran_count个数
    ran_count = 10
    if ran_count < len(nums):
        samlist = rd.sample(list(nums), ran_count)
    else:
        return
    #对随机数进行排序
    sortlist = np.sort(samlist)
    lens = len(sortlist)
    #每一个区间的长度
    part = (sortlist[lens - 1] - sortlist[0]) / n
    i = 1
    x = 0
    list_quantile = []
    x += sortlist[0] + part
    list_quantile.append(x)
    for i in range(1,n-1):
        x += part
        list_quantile.append(x)
    return sortlist,part,list_quantile
```

其中 part 为每一个区间的长度，list_quantile 为保存分位点的数组，
由于数据源是随机抽取的，所以每次的执行结果都不尽一样

程序结果为：[7.0 13.0]
程序测试函数如下：

```python
if __name__ == "__main__":
    nums = range(1, 21)
    # print(cal_mean(*nums))
    # print(cal_var(*nums))
    print(cal_quantile(3,*nums))
```

## 2. 实现两种噪声数据过滤，和缺失值补全方法

1.等箱装箱平滑(中位数光滑)，每个箱子的宽度都是一样的

```python
#等宽平滑
def equal_bins():
    nums = [7,9,11,5,6,12,3,2,13,14,18,20,22]
    sorted_nums = sorted(nums)
    lens = len(sorted_nums)
    # n 表示每个箱子的宽度
    n = 3
    # 箱子的数量
    count_bin = (sorted_nums[lens - 1] - sorted_nums[0]) / n
    real_countbin = int(count_bin + 1)
    i = 0
    var = 0
    m = 0
    list = []
    # [2, 3, 5, 6, 7, 9, 11, 12]

    for x in range(real_countbin):
        if var <= lens - 1:
            m = sorted_nums[var] + n
            while var <= lens - 1 and sorted_nums[var] <= m:
                list.append(sorted_nums[var])
                var += 1
            print(list,"中位数光滑之后为:%d" % np.median(list))
            list = []
```

程序执行结果为（其中每个箱子的宽度是 n = 3）：

```
[2, 3, 5] 中位数光滑之后为:3
[6, 7, 9] 中位数光滑之后为:7
[11, 12, 13, 14] 中位数光滑之后为:12
[18, 20] 中位数光滑之后为:19
[22] 中位数光滑之后为:22
```

## 2.等频装箱平滑(均值光滑)

```python
#等频平滑
def equal_frequency():
    nums = [47,81,21,74,78,98,94,56,56,78,94,65,1,23,29,78,26]
    lens = len(nums)
    #假定每个箱子里边的数量是4个
    n = 5
    sortnums = sorted(nums)
    #箱子数量  判断余数是否大于零点五，以便于决定加一后四舍五入或者直接四舍五入
    bin_num = lens / n
    remain = lens % n
    if remain > 0:
        bin_num += 1
    i = 0
    for x in range(int(bin_num)):
        print(sortnums[i:i+n])
        print("第%d个箱子的均值光滑后的数值为:%.2f" % (x+1,np.mean(sortnums[i:i+4])))
        i += n
```

执行结果为（用均值代替光滑后的数值）：

```
[1, 21, 23, 26, 29]
第1个箱子的均值光滑后的数值为:17.75
[47, 56, 56, 65, 74]
第2个箱子的均值光滑后的数值为:56.00
[78, 78, 78, 81, 94]
第3个箱子的均值光滑后的数值为:78.75
[94, 98]
第4个箱子的均值光滑后的数值为:96.00
```

1.均值补全缺失值

```
#均值补全
def missvalue_process1():
    #0表示缺失值,用均值代替缺失数值
    nums = [84,45,12,37,89,45,13,21,0,65,48,7]
    mean = cal_mean(*nums)
    nums[8] = round(mean,2)
    return nums
```

2.启发式补全

```python
#启发式的补全
def missvalue_process2():
    #1代表男生 0代表女生  最后两个数值缺失
    sex_height = [[0,1,0,1,0,1,0,1],[161,175,155,174,163,180,0,0]]
    #首先求出有数值的男生身高
    sexs = sex_height[0]
    heights = sex_height[1]
    location_man = []
    location_women = []
    for i in range(len(sexs) - 2):
        if sexs[i] == 1:
            location_man.append(i)
        if sexs[i] == 0:
            location_women.append(i)
    #print("男的为%s,女的为:%s" % (location_man,location_women))
    #统计男生身高
    hei_man = 0
    hei_women = 0
    for j in location_man:
        hei_man += heights[j]
    for m in location_women:
        hei_women += heights[m]
    #男生平均身高，保留2位小数
    man_mean = round(hei_man / len(location_man), 2)
    #女生平均身高
    women_mean = round(hei_women / len(location_women),2)
    heights[7] = man_mean
    heights[6] = women_mean
    return heights
```

执行结果为：

```
[161, 175, 155, 174, 163, 180, 159.67, 176.33]
```

最后两个为补全之后的数值

## 3. 实现两种数据离散化，数据数值化，数组归一化方法

```
"""
  数据离散化的方法(连续型数据离散化)

    1.等箱法  每个箱子的宽度一样

    2.等频法  每个箱子里边的数量是一样的
"""
```

1，　等箱法

```python
def equal_bins():
    nums = [14,9,11,5,6,12,13,2,13,19,18,20,22]
    sorted_nums = sorted(nums)
    lens = len(sorted_nums)
    # n 表示每个箱子的宽度
    n = 3
    # 箱子的数量
    count_bin = (sorted_nums[lens - 1] - sorted_nums[0]) / n
    real_countbin = int(count_bin + 1)
    i = 0
    var = 0
    m = 0
    list = []
    for x in range(real_countbin):
        if var <= lens - 1:
            m = sorted_nums[var] + n
            while var <= lens - 1 and sorted_nums[var] <= m:
                list.append(sorted_nums[var])
                var += 1
            print(list)
            list = []
```

执行结果：

```
[2, 5]
[6, 9]
[11, 12, 13, 13, 14]
[18, 19, 20]
[22]
```

2.等频法

```python
def equal_frequency():
    nums = [17,81,21,44,78,98,94,56,56,78,94,45,1,23,29,78,26]
    lens = len(nums)
    #假定每个箱子里边的数量是4个
    n = 4
    sortnums = sorted(nums)
    #箱子数量  判断余数是否大于零点五，以便于决定加一后四舍五入或者直接四舍五入
    bin_num = lens / n
    remain = lens % n
    if remain > 0:
        bin_num += 1
    i = 0
    for x in range(int(bin_num)):
        print(sortnums[i:i+n])
        print("第%d个箱子的类标记为:%.2f" % (x+1,np.mean(sortnums[i:i+4])))
        i += n
```

执行结果：

```
[26, 29, 44, 45]
第2个箱子的类标记为:36.00
[56, 56, 78, 78]
第3个箱子的类标记为:67.00
[78, 81, 94, 94]
第4个箱子的类标记为:86.75
[98]
第5个箱子的类标记为:98.00
```

```
"""
  数据标准化
一、min-max 标准化（Min-Max Normalization）

也称为离差标准化，是对原始数据的线性变换，使结果值映射到[0 - 1]之间。转换函数如下：

    x_new = (x - min)/(max - min)
二、Z-score 标准化方法

这种方法给予原始数据的均值（mean）和标准差（standard deviation）进行数据的标准化。
   经过处理的数据符合标准正态分布，即均值为 0，标准差为 1，转化函数为：

   x_new = (x - mean) / variance
"""
```

1，z-score 标准方法

```
#z-score标准化方法
def data_noraml1():
    nums = range(1, 21)
    mean = cal_mean(*nums)
    variance = cal_var(*nums)
    new_data = []
    for i in nums:
        value = (i - mean) / variance
        new_data.append(round(value, 2))
    return new_data
```

执行结果

```
[-0.29, -0.26, -0.23, -0.2, -0.17, -0.14, -0.11, -0.08, -0.05, -0.02, 0.02, 0.05, 0.08, 0.11, 0.14, 0.17, 0.2, 0.23, 0.26, 0.29]
```

2.min-max 标准化

```
#min-max标准化方法
def data_noraml2():
    nums = range(1, 21)
    mean = cal_mean(*nums)
    variance = cal_var(*nums)
    min = np.max(nums)
    max = np.min(nums)
    max_min = max - min
    new_data = []
    for i in nums:
        value = (i - min) / max_min
        new_data.append(round(value, 2))
    return new_data
```

执行结果：

【1, 0.94，0.89，0,83, 0,79 0.73 0.68 0.63 0.57 0.53 0.46 0.41 0.37 0.32 0.26 0.20 0.16 0.11】

```
#数据数值化
"""
 数据数值化
  1.one-hot 编码

  2.排序编码
"""
```

one-hot 编码

```python
#one-hot编码
def data2numerical1():
    nums = []
    i = 0
    m = 0
    weather = ["rainy","sunny","snowy","windy","cold"]
    for value in weather:  #控制行
        while m < len(weather):  #控制列,
            if value == weather[i]:
                nums.append(1)
            else:
                nums.append(0)
            m += 1
            i += 1
        m = 0
        i = 0
        print(value+"的one-hot编码如下:%s" % nums)
        nums = []
```

执行结果为：

```
rainy的one-hot编码如下:[1, 0, 0, 0, 0]
sunny的one-hot编码如下:[0, 1, 0, 0, 0]
snowy的one-hot编码如下:[0, 0, 1, 0, 0]
windy的one-hot编码如下:[0, 0, 0, 1, 0]
cold的one-hot编码如下:[0, 0, 0, 0, 1]
```

排序编码：

```python
#排序编码
def data2numerical2():
    #基本的类型
    ball_basic = ["small","median","big","large"]
    #数字标记集合
    ball_type = []
    balls = ["small","median","big","large","small","median","large","big"]
    i = 1
    loc = 0
    nlist = []
    for value in ball_basic:
        if i <= len(ball_basic):
            ball_type.append(i)
            i += 1
    for x in balls:
        for m in range(len(ball_basic)):
            if x == ball_basic[m]:
                nlist.append(ball_type[m])
    return nlist
```

执行结果：

```
[1, 2, 3, 4, 1, 2, 4, 3]
```

## 4. 实现两种数据离散化，数据数值化，数组归一化方法

实现两种相似度计算方法

1.数值型(欧氏距离)

```python
#.数值型相似度实现(欧式距离)
def numercial_type():
    nums_one = [7,87,41,32,17,8,97,45,12,31,48,4]
    nums_two = [13,6,9,7,54,8,41,23,1,64,54,5]
    #对数据做一个归一化处理
    normalData_one = data_noraml2(*nums_one)
    normalData_two = data_noraml2(*nums_two)
    lens = len(nums_one)
    dis = [normalData_one[i]-normalData_two[i] for i in range(lens)]
    sum = 0
    for x in dis:
        sum += x**2
    return round(m.sqrt(sum),2)
```

## 2.标称型 dp

```
#.标称型相似度实现 d(x,x')=(p-m)/p
def nominal_type():
    type_one = [8,4,6,51,23,12,31]
    type_two = [31,8,41,23,12,3,4]
    lens = len(type_one)
    sim_counts = 0
    for i in type_one:
        for j in type_two:
            if i == j:
                sim_counts += 1
    dis = (lens - sim_counts) / lens
    return round(dis,2)
```

执行结果为：

```
0.29
```