

## DATA MINING 第 03 次作业

### 习题 3.11

使用习题 3.3 给出的 age 数据：

(a) 画一个宽度为 10 的等宽的直方图

(b) 要描述如下每种抽样技术的例子：SRSWOR, SRSWR, 簇抽样, 分层抽样, 使用大小为 5 的样本以及层 “young”, “middle\_aged”, “senior”。

实验目的：熟练掌握一种绘制统计图的工具，熟悉常见的抽样方式

实验原理：jupyter notebook, python pandas.hist() 函数，抽样原理的定义  
实验过程及结果：

一、利用 python pandas.hist() 函数绘制数据直方图

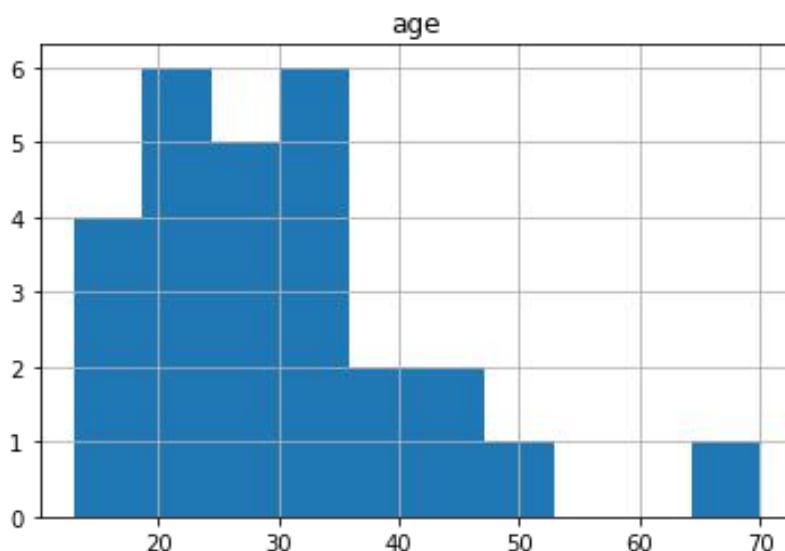


图 1：3.11 第一问直方图

二、利用各种抽样方式的定义做出在 age 数据集上抽样的例子

1、SRSWOR: 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。

例子：13 (s=1)

2、SRSWR: S 个样本的有放回简单随机抽样：从 D 中抽取 s 个样本，而且每次抽取一个样本，记录后放回数据集 D 中。

例子：15 (s=1)

3、簇抽样：如果 D 中的元组被分组，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  $s < M$ 。

例子：13、15、16、16、19、20 (s=1、M=6、对样本以 10 为距离划分簇)

4、分层抽样：如果 D 被划分为互不相交的部分，称做层，则可通过对每一层的 SRS 就可以得到 D 的分层抽样。

例子：13 25 70

### 习题 3.12

Chimerge[Ker92]是监督的、自底向上的（即基于合并的）数据离散化。它依赖于卡方分析：具有最小的卡方值的相邻区间合并在一起，直到满足确定的停止标准。

（a）简略描述 Chimerge 如何工作；

（b）取 iris 数据集作为离散的数据集合，使用 Chimerge 方法，对四个数值属性分别进行离散化（停止条件为：max-interval=6）。要求写一个小程序，避免麻烦的计算。提交你的简要分析和检查结果：分裂点，最终的区间，以及源程序文档。

实验目的：了解并掌握 Chimerge 离散化方法，熟练应用程序实现 Chimerg 方法

实验原理：Chimerge 方法,Pycharm

实验过程及结果：

#### 一、Chimerge 工作原理<sup>[1]</sup>：

Chimerge[Ker92]是监督的、自底向上的（即基于合并的）数据离散化。它依赖于卡方分析：具有最小的卡方值的相邻区间合并在一起，直到满足确定的停止标准。

其基本思想是：对于精确的离散化，相对类频率在一个区间内应当完全一致。因此如果两个邻近区间具有非常类似的分布，则这个区间可以合并，否则，它们应当保持分开。其过程是初始时，把数值属性 A 的每个不同值看作一个区间。对于每对相邻区间进行卡方检验，具有最小卡方的相邻区间合并在一起，因为低卡方值表明它们具有相似的类分布。该合并过程递归地进行，直到满足预定义的终止条件。

具体步骤如下：

#### 1、初始化：

根据要离散的属性对实例进行排序；每个实例属于一个区间。

#### 2、合并区间，又包括两步骤：

A、计算每一对相邻区间的卡方值。

B、将卡方值最小的一对区间合并。

简化为：

将离散属性值进行升序排序；

将每个实例设置成单独区间；

While（截止条件）

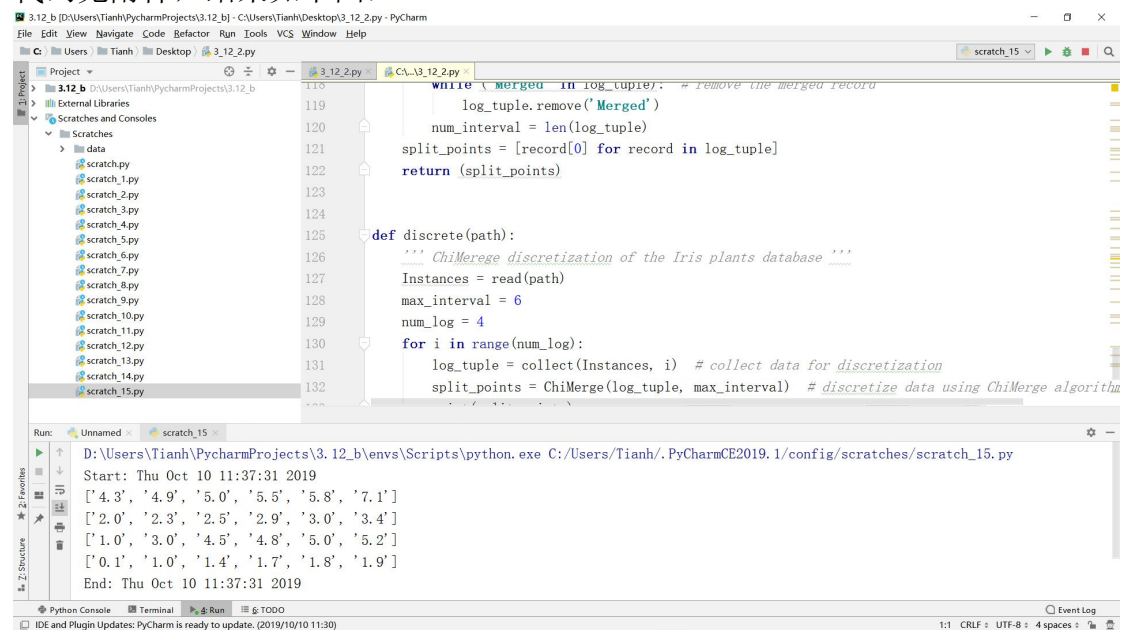
```
{
    循环对每对相邻区间进行卡方计算，找出最小卡方值的相邻区间；
    对相邻区间进行合并；
}
```

注：

卡方检验就是统计样本的实际观测值与理论推断值之间的偏离程度，实际观测值与理论推断值之间的偏离程度就决定卡方值的大小，如果卡方值越大，二者偏差程度越大；反之，二者偏差越小；若两个值完全相等时，卡方值就为 0，表明理论值完全符合<sup>[2]</sup>。

## 二、编写函数利用 python 实现 Chimerge 算法<sup>[2]</sup>

代码见附件，结果如下图：



```
118 while (merged in log_tuple): # remove the merged record
119     log_tuple.remove('Merged')
120     num_interval = len(log_tuple)
121     split_points = [record[0] for record in log_tuple]
122     return (split_points)
123
124
125 def discrete(path):
126     """ ChiMerge discretization of the Iris plants database """
127     Instances = read(path)
128     max_interval = 6
129     num_log = 4
130     for i in range(num_log):
131         log_tuple = collect(Instances, i) # collect data for discretization
132         split_points = ChiMerge(log_tuple, max_interval) # discretize data using ChiMerge algorithm
```

Run: Unnamed x scratch\_15

D:\Users\Tianh\PycharmProjects\3.12\_b\envs\Scripts\python.exe C:/Users/Tianh/.PyCharmCE2019.1/config/scratches/scratch\_15.py

Start: Thu Oct 10 11:37:31 2019

['4.3', '4.9', '5.0', '5.5', '5.8', '7.1']

['2.0', '2.3', '2.5', '2.9', '3.0', '3.4']

['1.0', '3.0', '4.5', '4.8', '5.0', '5.2']

['0.1', '1.0', '1.4', '1.7', '1.8', '1.9']

End: Thu Oct 10 11:37:31 2019

图 2： 3.12 第二问程序运行结果

## 习题 3.13

对如下问题，使用伪代码或你喜欢用的程序设计语言，给出一个算法：

- (a) 对于标称数据，基于给定模式中属性的不同值的个数，自动产生概念分层；
- (b) 对于数值数据，基于等宽划分规则，自动产生概念分层；
- (c) 对于数值数据，基于等频划分规则，自动产生概念分层；

实验目的：熟练掌握离散化的方法，提高将思想转化为代码的能力

实验原理：jupyter notebook，标称数据离散化规则，等宽划分规则，等频划分规则

实验过程及结果：

### 一、用伪代码表示如下

导入数据

对数据去重

对数据排序

对每一个数据分配 int 型变量，保存至 C++ 容器 map 中

输出 map

二、利用 `pd.cut()` 实现等宽离散化<sup>[3]</sup>，并利用 pandas 包源码<sup>[4]</sup>实现 `pd.cut()` 函数

以 iris 数据集为例，划分结果如下：

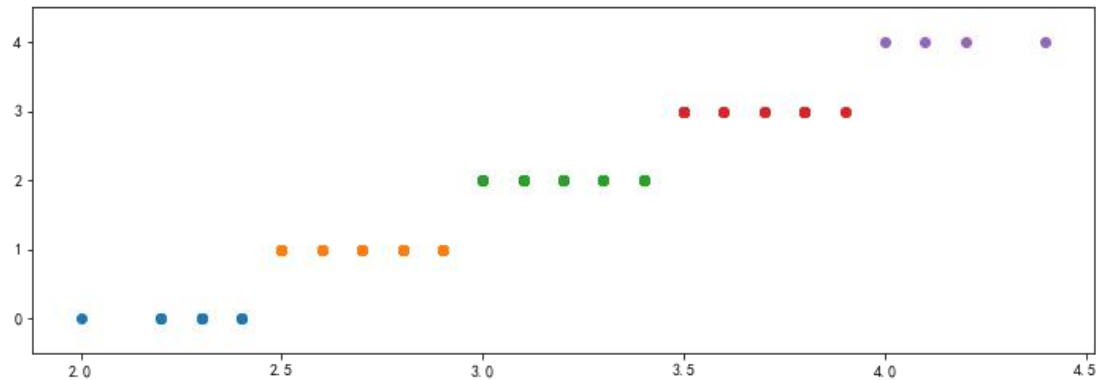


图 3：3.13 第二问对 iris 数据集进行等宽离散化

三、利用 `pd.cut()` 实现等频离散化，并利用 pandas 包源码实现 `pd.cut()` 函数  
以 iris 数据集为例，划分结果如下：

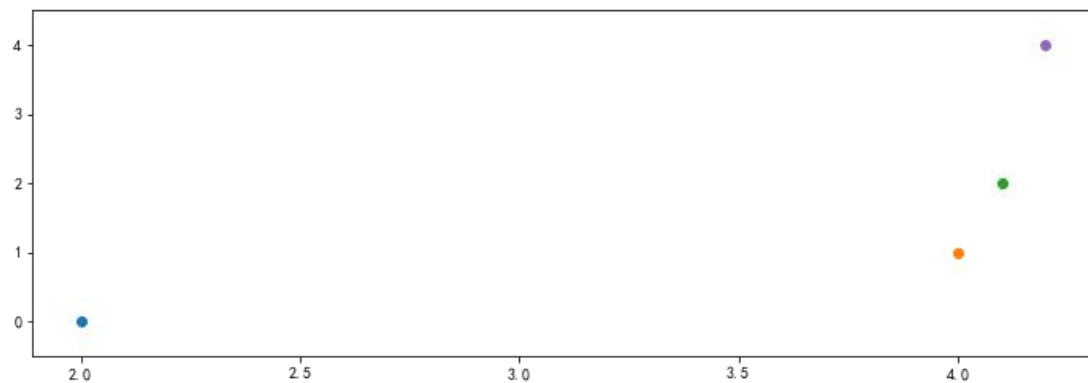


图 4：3.13 第三问对 iris 数据集进行等频离散化

## 思考题：

试绘制 从  $n$  个样本  $K$  个簇里抽出的数都来自不同的簇的概率图像

实验目的：深入体会簇抽样

实验原理：设簇的个数  $k = 8$ ，jupyter notebook, matplotlib

实验过程及结果：

利用 jupyter notebook 绘制简单的图像如下

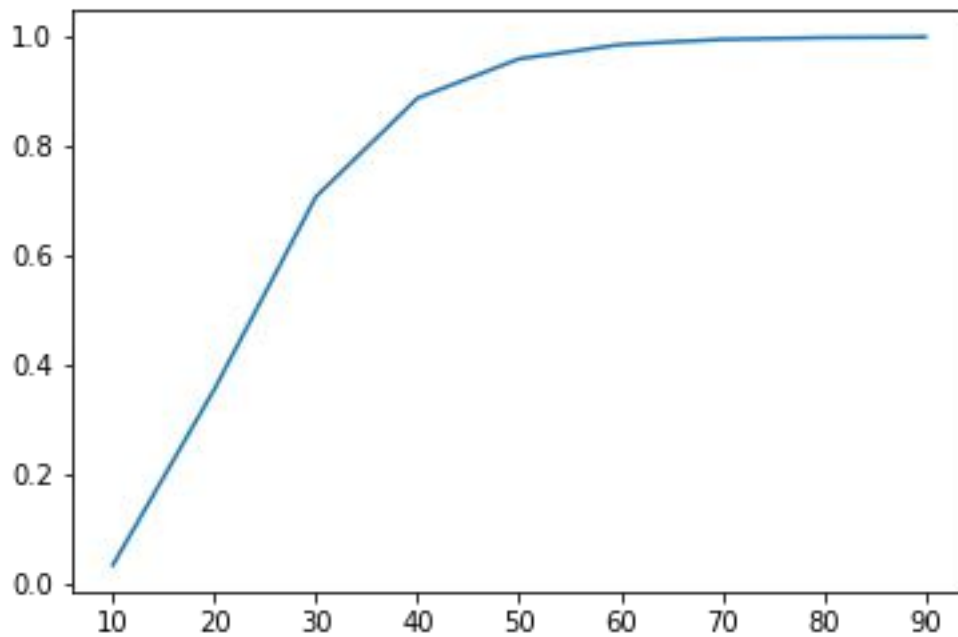


图 5: 从  $n$  个样本  $K$  个簇里抽出的数都来自不同的簇的概率图像

#### 参考文献

- [1] <https://yq.aliyun.com/articles/4008>
- [2] <https://blog.csdn.net/zhaoyl03/article/details/8689440>
- [3] [https://blog.csdn.net/weixin\\_40683253/article/details/81780910](https://blog.csdn.net/weixin_40683253/article/details/81780910)
- [4] <https://github.com/pandas-dev/pandas/blob/v0.25.1/pandas/core/reshape/tile.py>

附录:

3\_11\_1

```
import pandas as pd
import numpy as np
from pandas.core.frame import DataFrame
```

```
l=[13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36,
40, 45, 46, 52, 70]
```

```
l1 = {'age':l}
data = DataFrame(l1)
```

```
data.hist()
```

3\_12

```
from time import ctime
```

```
def read(file):
```

```
    '''read raw data from a file'''
```

```
    Instances = []
```

```
    fp = open(file, 'r')
```

```
    for line in fp:
```

```
        line = line.strip('\n')  # discard '\n'
```

```
        if line != "":
```

```
            Instances.append(line.split(','))
```

```
    fp.close()
```

```
    return (Instances)
```

```
def split(Instances, i):
```

```
    ''' Split the 4 attributes, collect the data of the ith attributes, i=0,1,2,3
```

```
    Return a list like [['0.2', 'Iris-setosa'], ['0.2', 'Iris-setosa'],...]''''
```

```
    log = []
```

```
    for r in Instances:
```

```
        log.append([r[i], r[4]])
```

```
    return (log)
```

```
def count(log):
```

```
    '''Count the number of the same record
```

```
    Return a list like [['4.3', 'Iris-setosa', 1], ['4.4', 'Iris-setosa', 3],...]''''
```

```
    log_cnt = []
```

```
    log.sort(key=lambda log: log[0])
```

```

i = 0
while (i < len(log)):
    cnt = log.count(log[i])  # count the number of the same record
    record = log[i][:]
    record.append(cnt)  # the return value of append is None
    log_cnt.append(record)
    i += cnt  # count the next diferent item
return (log_cnt)

```

```

def build(log_cnt):
    """Build a structure (a list of truples) that ChiMerge algorithm works properly on
    it """

```

```

    log_dic = {}
    for record in log_cnt:
        if record[0] not in log_dic.keys():
            log_dic[record[0]] = [0, 0, 0]
        if record[1] == 'Iris-setosa':
            log_dic[record[0]][0] = record[2]
        elif record[1] == 'Iris-versicolor':
            log_dic[record[0]][1] = record[2]
        elif record[1] == 'Iris-virginica':
            log_dic[record[0]][2] = record[2]
        else:
            raise TypeError("Data Exception")
    log_tuple = sorted(log_dic.items())
    return (log_tuple)

```

```

def collect(Instances, i):
    """ collect data for discretization """
    log = split(Instances, i)
    log_cnt = count(log)
    log_tuple = build(log_cnt)
    return (log_tuple)

```

```

def combine(a, b):
    """ a=('4.4', [3, 1, 0]), b=('4.5', [1, 0, 2])
        combine(a,b)=('4.4', [4, 1, 2]) """
    c = a[:]  # c[0]=a[0]
    for i in range(len(a[1])):
        c[1][i] += b[1][i]
    return (c)

```

```

def chi2(A):
    ''' Compute the Chi-Square value '''
    m = len(A);
    k = len(A[0])
    R = []
    for i in range(m):
        sum = 0
        for j in range(k):
            sum += A[i][j]
        R.append(sum)
    C = []
    for j in range(k):
        sum = 0
        for i in range(m):
            sum += A[i][j]
        C.append(sum)
    N = 0
    for ele in C:
        N += ele
    res = 0
    for i in range(m):
        for j in range(k):
            Eij = R[i] * C[j] / N
            if Eij != 0:
                res = res + (A[i][j] - Eij) ** 2 / Eij
    return res

```

```

def ChiMerge(log_tuple, max_interval):
    ''' ChiMerge algorithm '''
    ''' Return split points '''
    num_interval = len(log_tuple)
    while (num_interval > max_interval):
        num_pair = num_interval - 1
        chi_values = []
        for i in range(num_pair):
            arr = [log_tuple[i][1], log_tuple[i + 1][1]]
            chi_values.append(chi2(arr))
        min_chi = min(chi_values) # get the minimum chi value
        for i in range(num_pair - 1, -1, -1): # treat from the last one
            if chi_values[i] == min_chi:
                log_tuple[i] = combine(log_tuple[i], log_tuple[i + 1]) # combine

```



*the two adjacent intervals*

```
        log_tuple[i + 1] = 'Merged'
    while ('Merged' in log_tuple): #remove the merged record
        log_tuple.remove('Merged')
    num_interval = len(log_tuple)
    split_points = [record[0] for record in log_tuple]
    return (split_points)
```

```
def discrete(path):
    ''' ChiMerge discretization of the Iris plants database '''
    Instances = read(path)
    max_interval = 6
    num_log = 4
    for i in range(num_log):
        log_tuple = collect(Instances, i) # collect data for discretization
        split_points = ChiMerge(log_tuple, max_interval) # discretize data using
ChiMerge algorithm
        print(split_points)
```

```
if __name__ == '__main__':
    print('Start: ' + ctime())
    discrete('c:\\Users\\Tianh\\Desktop\\iris_data.csv')
    print('End: ' + ctime())
```

3\_13\_2 + 3\_12\_3

```
import pandas as pd
import numpy as np
```

```
datafile = r'C:\Users\Tianh\Desktop\iris_data.csv'
#                                     df                                     =
pd.read_csv(r'C:\Users\Tianh\Desktop\iris_data.csv', header=None)
data = pd.read_csv(datafile, header=None)
data = data.iloc[:, 1]
k = 5
```

```
data.head()
```

#等宽离散

```
d1 = pd.cut(data, k, labels = range(k))
```

```

def cluster_plot(d,k):
    import matplotlib.pyplot as plt
    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

```

```

cluster_plot(d1, k).show()

```

#等频离散

```

data = data.drop_duplicates(keep=False)
w = [1.0*i/k for i in range(k+1)]
w = data.describe(percentiles = w)[4:4+k+1]
w[0] = w[0]*(1-1e-10)
d2 = pd.cut(data, w, labels = range(k))

```

```

cluster_plot(d2, k).show()

```

思考题:

```

import numpy as np
import matplotlib.pyplot as plt

```

```

x=np.arange(10,100,10)
y=(1-0.9**x)**8

```

```

plt.plot(x, y)

```