

实验一 python 实验环境与信息熵的计算实现

学号: 21030031009 姓名: 惠欣宇 电话: 18149045867

1. 实验目的

- 1) 掌握 python 的基础语法。
- 2) 学习使用 conda 命令进行环境搭建以及包管理。
- 3) 利用 Pycharm 或 VScode 和 Anaconda 熟悉 python 程序开发的流程。
- 4) 通过编写一个计算信源熵的小程序加深对信源熵的理解。

2. 实验预备知识及实验要求

2.1. 实验预备知识

- 1) Python 的基本语法
- 2) 信息熵的定义以及计算方法
- 3) 参考资料:[廖雪峰的官方网站](#)

2.2. 实验要求:

- 1) 实现信息熵的代码:

```
import math
#记录每个数据出现的次数
def StatDataInf(data):
    dataLen = len(data)
    diffDataNum = []
    diffData = []
    other = data
    for i in range(dataLen):
        cnt = 0
        j = i
        if (other[j] != '/'):
            temp = other[i]
            diffData.append(temp)
            while (j < dataLen):
                if (other[j] == temp):
                    cnt += 1
                    other[j] = '/'
                j = j + 1
            diffDataNum.append(cnt)
    return diffData, diffDataNum
```

```

#计算信息熵
def DataEntropy(data, diffData, diffDataNum):
    dataLen = len(data)
    diffDataLen = len(diffDataNum)
    entropyVal = 0
    for i in range(diffDataLen):
        proptyVal = diffDataNum[i] / dataLen
        entropyVal = entropyVal - proptyVal *
math.log2(proptyVal)
    return entropyVal

#测试样例
def main():
    data = [1, 2, 1, 2, 1, 2, 1, 2, 1, 2]
    [diffData, diffDataNum] = StatDataInf(data)
    entropyVal = DataEntropy(data, diffData, diffDataNum)
    print(entropyVal)
    data = [1, 2, 1, 2, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1]
    [diffData, diffDataNum] = StatDataInf(data)
    entropyVal = DataEntropy(data, diffData, diffDataNum)
    print(entropyVal)
    data = [1, 2, 3, 4, 2, 1, 2, 4, 3, 2, 3, 4, 1, 1, 1]
    [diffData, diffDataNum] = StatDataInf(data)
    entropyVal = DataEntropy(data, diffData, diffDataNum)
    print(entropyVal)
    data = [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3,
4, 1, 2, 3, 4, 1, 2, 3, 4]
    [diffData, diffDataNum] = StatDataInf(data)
    entropyVal = DataEntropy(data, diffData, diffDataNum)
    print(entropyVal)
    data = [1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,
1, 2, 3, 4, 5, 1, 2, 3, 4, 1, 2, 3, 4, 5]
    [diffData, diffDataNum] = StatDataInf(data)
    entropyVal = DataEntropy(data, diffData, diffDataNum)
    print(entropyVal)

if __name__ == '__main__':
    main()

```

2) 在 **vscode** 中安装 **python** 插件以及中文包，重启 **vscode**。接着创建项目，配置工作区域。

①首先创建一个本地文件夹，作为项目文件。

②在设置中配置工作区域，打开配置文件，配置 **flake8** 与 **yapf** 并关闭 **pylint** 工具。

③在工作区域内输入

```
"python.linting.flake8Enabled": true,  
"python.formatting.provider": "yapf",  
"python.linting.flake8Args": ["--max-line-length=248"],  
"python.linting.pylintEnabled": false
```

接下来就可以编写本次实验的代码了。

3) 实验结果。

根据一维离散信源的信息熵的计算公式：

$$H(x) = - \sum_{i=1}^n p_i \log p_i$$

分析其关键点在于：

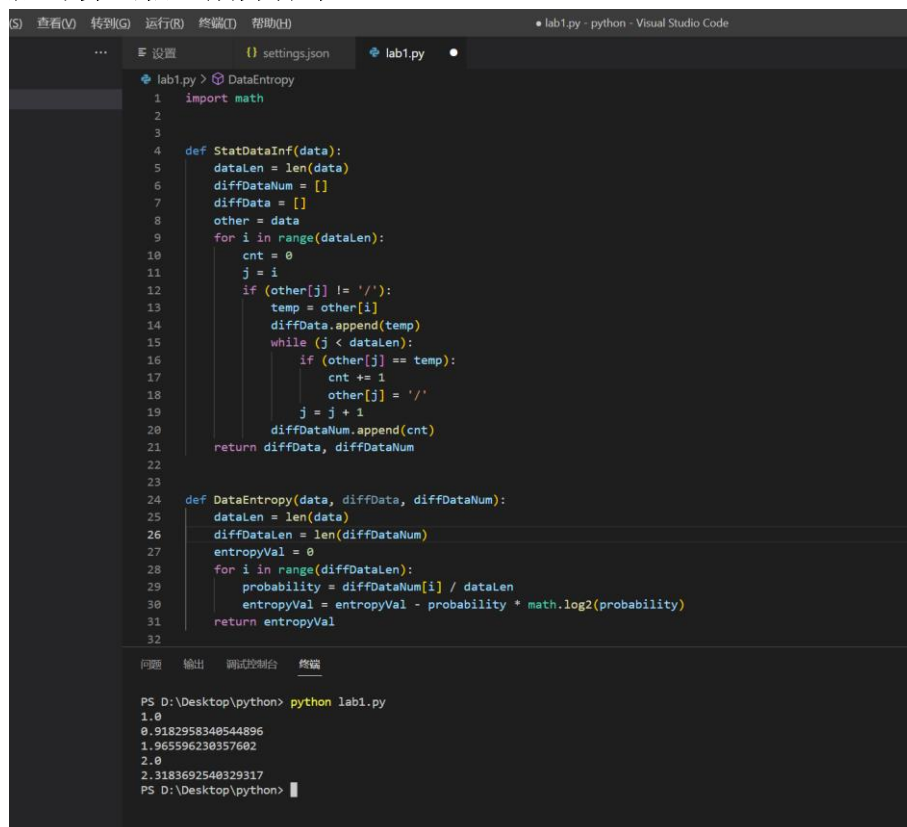
1. 统计离散信源中的不同信息出现的次数。

StatDataInf 函数通过对每一组数据的遍历计数来得到每一组数据中每个数字出现的次数，并储存在变量数组 **diffDataNum** 中，以完成对离散信源中不同信息出现次数的统计。

2. 计算出现不同信息的概率。

DataEntropy 函数操作每一组数据，通过遍历先得到变量数组 **diffDataNum**，并将每一个元素除以该组数据的总个数来计算得到每一个数据所占的比例并用变量 **probability** 中，再通过熵计算公式得到该组数据的熵，通过变量 **entropyVal** 返回。

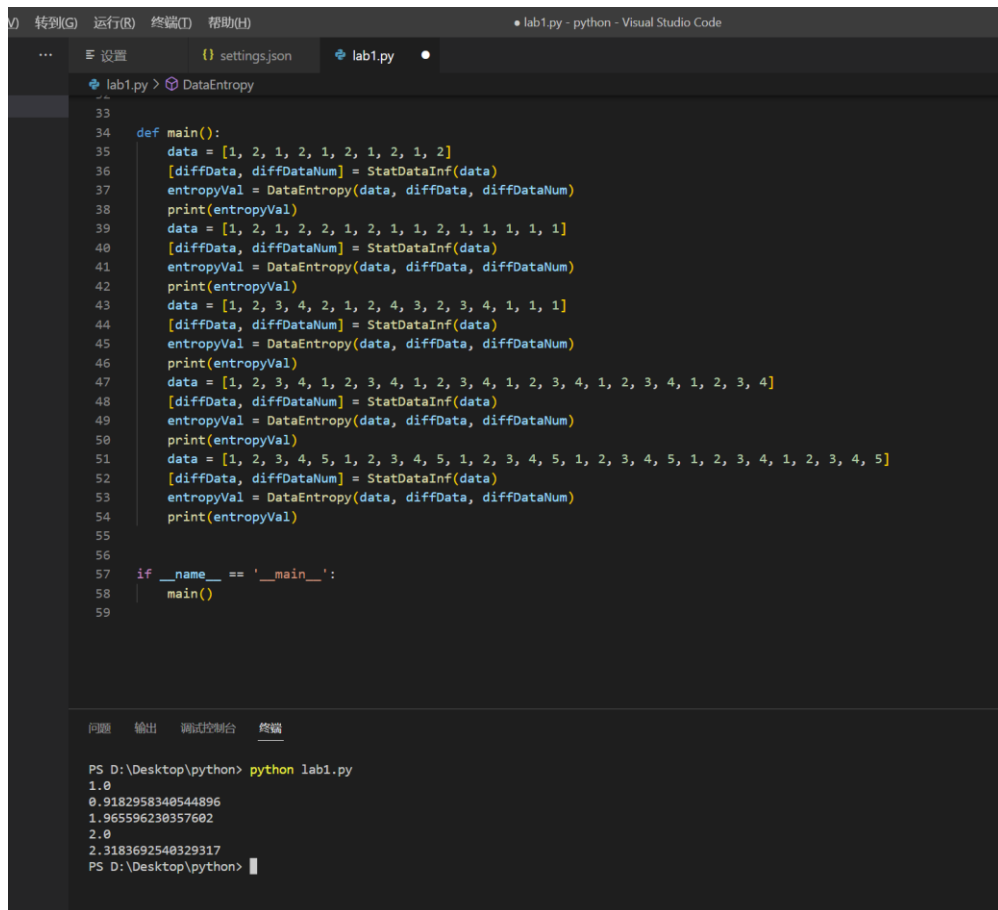
3. 最后计算出信息熵并打印。



```
lab1.py > DataEntropy  
1 import math  
2  
3  
4 def StatDataInf(data):  
5     dataLen = len(data)  
6     diffDataNum = []  
7     diffData = []  
8     other = data  
9     for i in range(dataLen):  
10        cnt = 0  
11        j = i  
12        if (other[j] != '/'):   
13            temp = other[i]  
14            diffData.append(temp)  
15            while (j < dataLen):  
16                if (other[j] == temp):  
17                    cnt += 1  
18                    other[j] = '/'  
19                    j = j + 1  
20            diffDataNum.append(cnt)  
21    return diffData, diffDataNum  
22  
23  
24 def DataEntropy(data, diffData, diffDataNum):  
25     dataLen = len(data)  
26     diffDataLen = len(diffDataNum)  
27     entropyVal = 0  
28     for i in range(diffDataLen):  
29         probability = diffDataNum[i] / dataLen  
30         entropyVal = entropyVal - probability * math.log2(probability)  
31    return entropyVal  
32
```

```
PS D:\Desktop\python> python lab1.py  
1.0  
0.9182958340544896  
1.965596230357602  
2.0  
2.3183692540329317  
PS D:\Desktop\python>
```

图 1 结果截图



```
33
34 def main():
35     data = [1, 2, 1, 2, 1, 2, 1, 2, 1, 2]
36     [diffData, diffDataNum] = StatDataInf(data)
37     entropyVal = DataEntropy(data, diffData, diffDataNum)
38     print(entropyVal)
39     data = [1, 2, 1, 2, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1]
40     [diffData, diffDataNum] = StatDataInf(data)
41     entropyVal = DataEntropy(data, diffData, diffDataNum)
42     print(entropyVal)
43     data = [1, 2, 3, 4, 2, 1, 2, 4, 3, 2, 3, 4, 1, 1, 1]
44     [diffData, diffDataNum] = StatDataInf(data)
45     entropyVal = DataEntropy(data, diffData, diffDataNum)
46     print(entropyVal)
47     data = [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
48     [diffData, diffDataNum] = StatDataInf(data)
49     entropyVal = DataEntropy(data, diffData, diffDataNum)
50     print(entropyVal)
51     data = [1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 1, 2, 3, 4, 5]
52     [diffData, diffDataNum] = StatDataInf(data)
53     entropyVal = DataEntropy(data, diffData, diffDataNum)
54     print(entropyVal)
55
56
57 if __name__ == '__main__':
58     main()
59
```

问题 输出 调试控制台 终端

```
PS D:\Desktop\python> python lab1.py
1.0
0.9182958340544896
1.965596230357602
2.0
2.3183692540329317
PS D:\Desktop\python>
```

图 2 结果截图

从本次实验的样例数据得出的结果可以看出，变量的不确定性越大，熵也就越大，所需要收集的信息量也就越大。

4) 总结分析。

①在计算时主要需要注意正向指标和负向指标的区分，对于负向指标可以在归一化时改变改变公式计算为正向，也可以数据预处理时提前对指标正向化，如可以把失败率正向化为 1-失败率。

②熵权法的熵的计算公式和信息熵的计算公式不同，计算概率时不是每个值出现的次数除以总次数，而是直接用值除以值的求和。