­­­­

数据挖掘实验报告

实验名称： 数据预处理

姓 名： 段欣然

学 号：202011081033

实验日期： 2023.4.2

# 一、实验目的及要求

1. 掌握数据预处理概念

2. 掌握基本的数据预处理方法

3. 将源代码及实验结果的截图，粘贴到实验步骤的每一小问中

## 二、实验所需Python库

|  |  |
| --- | --- |
| 名称 | 版本 |
| SKlearn | 0.22.1 |
| Numpy | 1.19.2 |
| Pandas | 0.25.0 |

## 数据集介绍

## Price-label数据集

此数据集由20条购物车记录组成，有2个属性，分为别“price”商品价格；“label”购物车标签，其中为“1”代表在购物车中，为“0”代表不在购物车中。

## Newsgroups20

此数据集是一个新闻组文档的集合。20个新闻组的已成为机器学习技术文本应用（如文本分类和文本聚类）实验的热门数据集。

**注**：本实验仅用到**二十**类中的**两**类，即：'alt.atheism','comp.graphics'。

## 实验内容

1. 读取price-label数据集，完成任务1，基于信息熵做数据离散化处理，并打印出处理结果。
2. 读取 Newsgroups20 文件中的数据，完成任务2中的习题。

## 实验步骤

**任务1：**

基于信息熵的数据离散化算法是由监督学习算法，在使用该方法对数据进行离散化时，需要数据有对应的标签。

本次实验使用的是一份用户最近点击的20个商品的价格与是否加入购物车对应关系。

数据文件为压缩包中“price-label.csv”文件。

**信息熵**

信息熵的定义:信息量度量的是一个具体事件发生了所带来的信息，而熵则是在结果出来之前对可能产生的信息量的期望——考虑该随机变量的所有可能取值，即所有可能发生事件所带来的信息量的期望。

信息熵就是信息量的数学期望。

公式：



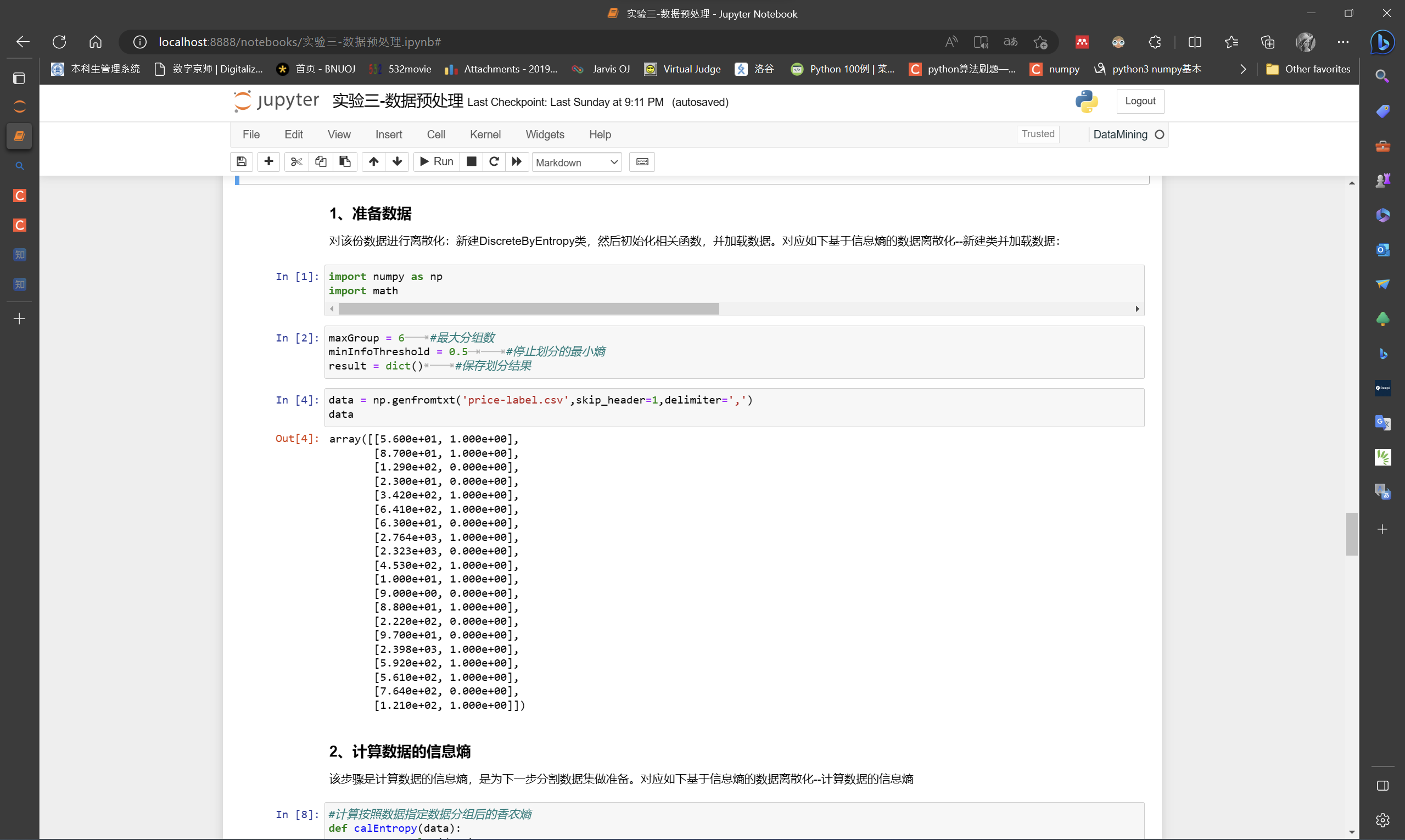
P(xi)表示随机事件X为 xi 的概率。

**步骤：**

1. 准备数据

初始化相关函数，并加载数据。对应如下基于信息熵的数据离散化--设置参数并加载数据。

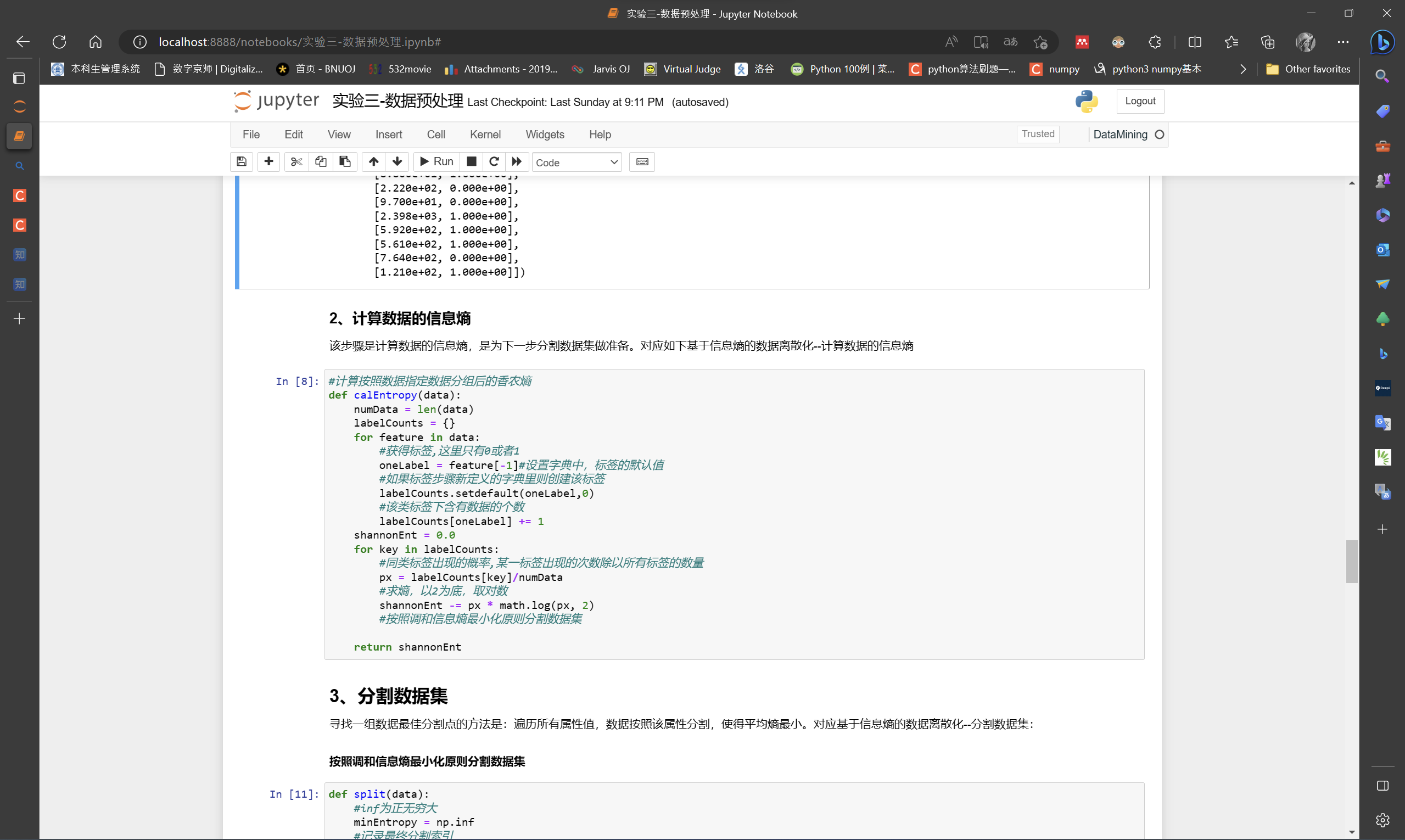
1. **import** numpy as np
2. **import** math
3. maxGroup = 6    #最大分组数
4. minInfoThreshold = 0.5      #停止划分的最小熵
5. result = dict()     #保存划分结果
7. data = np.genfromtxt('price-label.csv',skip\_header=1,delimiter=',')
8. data



（2）计算数据的信息熵

该步骤是计算数据的信息熵，是为下一步分割数据集做准备。

1. #计算按照数据指定数据分组后的香农熵
2. **def** calEntropy(data):
3. numData = len(data)
4. labelCounts = {}
5. **for** feature **in** data:
6. #获得标签,这里只有0或者1
7. oneLabel = feature[-1]#设置字典中，标签的默认值
8. #如果标签步骤新定义的字典里则创建该标签
9. labelCounts.setdefault(oneLabel,0)
10. #该类标签下含有数据的个数
11. labelCounts[oneLabel] += 1
12. shannonEnt = 0.0
13. **for** key **in** labelCounts:
14. #同类标签出现的概率,某一标签出现的次数除以所有标签的数量
15. px = labelCounts[key]/numData
16. #求熵，以2为底，取对数
17. shannonEnt -= px \* math.log(px, 2)
18. #按照调和信息熵最小化原则分割数据集
20. **return** shannonEnt



（3）分割数据集

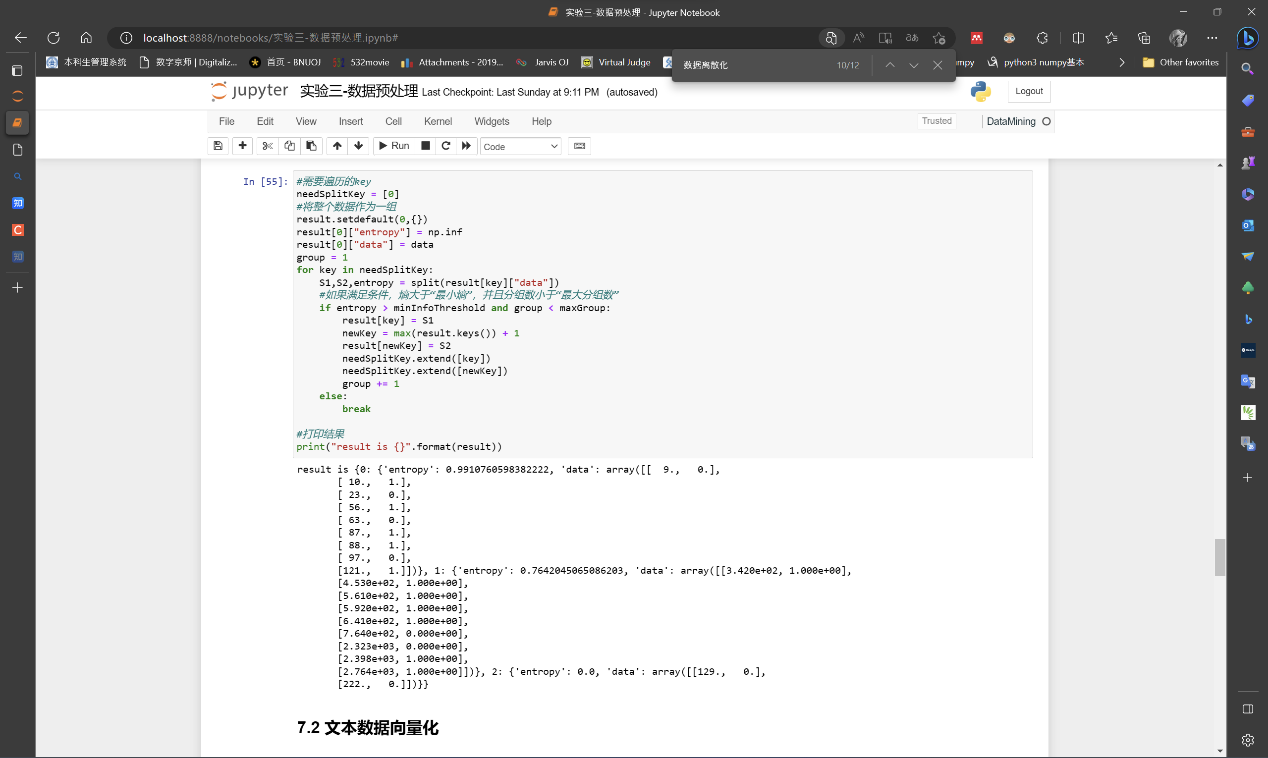
寻找一组数据最佳分割点的方法是：遍历所有属性值，数据按照该属性分割，使得平均熵最小。

1. **def** split(data):
2. #inf为正无穷大
3. minEntropy = np.inf
4. #记录最终分割索引
5. index = -1
6. #按照第一列对数据进行排序
7. sortData = data[np.argsort(data[:,0])]
8. #初始化最终分割数据后的熵
9. lastE1,lastE2 = -1,-1
10. #返回的数据结构，包含数据和对应的熵
11. S1 = dict()
12. S2 = dict()
13. **for** i **in** range(len(sortData)):
14. #分割数据集
15. splitData1,splitData2 = sortData[: i + 1],sortData[i + 1 :]#计算信息熵
16. entropy1,entropy2 = (
17. calEntropy(splitData1),
18. calEntropy(splitData2),
19. ) #计算调和平均熵
20. entropy = entropy1 \* len(splitData1) / len(sortData) + entropy2 \* len(splitData2) / len(sortData)
21. #如果调和平均熵小于最小值
22. **if** entropy < minEntropy:
23. minEntropy = entropy
24. index = i
25. lastE1 = entropy1
26. lastE2 = entropy2
27. S1["entropy"] = lastE1
28. S1["data"] = sortData[: index + 1]
29. S2["entropy"] = lastE2
30. S2["data"] = sortData[index + 1 :]
31. **return** S1,S2,minEntropy

（4）数据离散化

按照上面基于信息熵分组的内容，对数据做离散化处理并打印出处理结果。

1. #需要遍历的key
2. needSplitKey = [0]
3. #将整个数据作为一组
4. result.setdefault(0,{})
5. result[0]["entropy"] = np.inf
6. result[0]["data"] = data
7. group = 1
8. **for** key **in** needSplitKey:
9. S1,S2,entropy = split(result[key]["data"])
10. #如果满足条件，熵大于“最小熵”，并且分组数小于“最大分组数”
11. **if** entropy > minInfoThreshold **and** group < maxGroup:
12. result[key] = S1
13. newKey = max(result.keys()) + 1
14. result[newKey] = S2
15. needSplitKey.extend([key])
16. needSplitKey.extend([newKey])
17. group += 1
18. **else**:
19. **break**
21. #打印结果
22. **print**("result is {}".format(result))



**任务2：**

对Newsgroups20文本信息预处理。利用词袋模型，将文本数据向量化。

1. 对文本进行分词，得到词表，即一个包含不同单词，且无重复单词的集合，这里需要考虑：去除标点符号、单词均转化为小写。

例如：data=['I am a student.','I am a teacher.']，分词后得到集合('i', 'am', 'a', 'student', 'teacher')

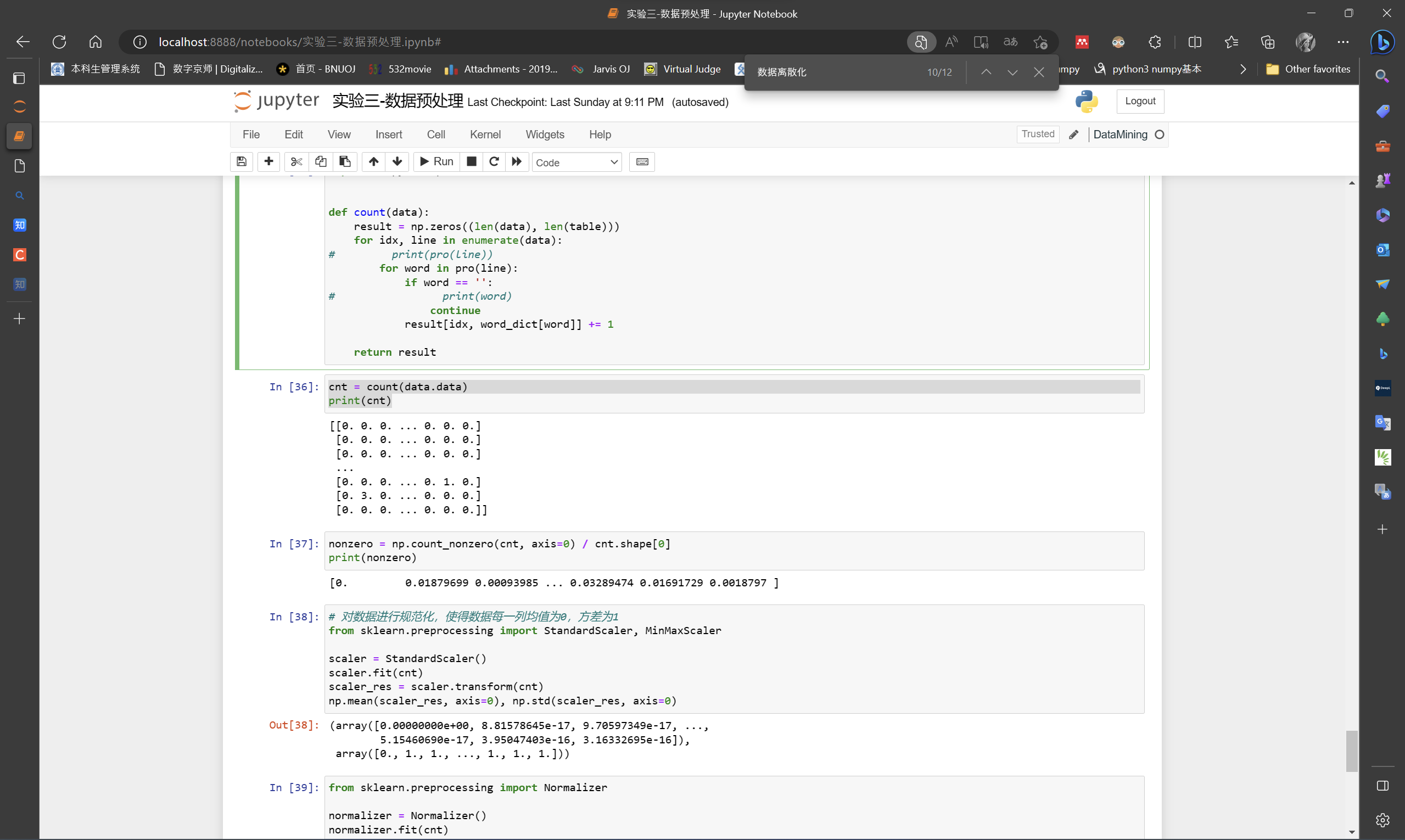
1. **def** pro(data):
2. **import** re
3. data = data.lower()
4. # 把所有非字母内容替换为空格
5. data = re.sub(r'[^a-z]', ' ', data)
7. result = data.split(' ')
8. **return** result
10. table = []
11. num = 0
12. **for** line **in** data.data:
13. num += 1
14. table += pro(line)
15. table = set(table)

2. 统计每篇文章出现的词的次数，得到二维数组

例如：上面的例子中得到二维数组[[1, 1, 1, 1, 0],[1, 1, 1, 0, 1]]

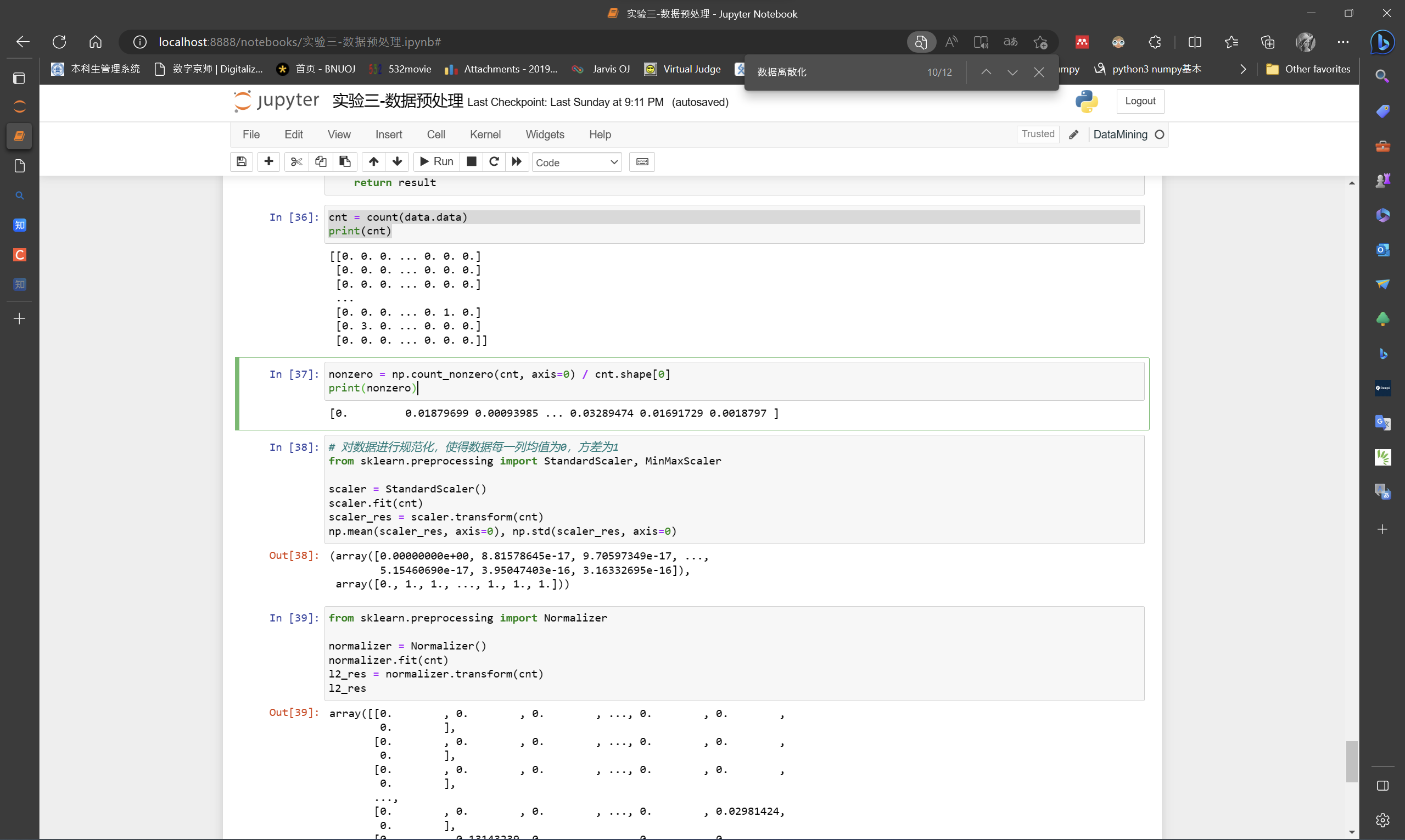
1. table = list(table)
2. word\_dict = {}
3. **for** idx, it **in** enumerate(table):
4. word\_dict[it] = idx
5. word\_dict
7. **import** numpy as np

10. **def** count(data):
11. result = np.zeros((len(data), len(table)))
12. **for** idx, line **in** enumerate(data):
13. #         print(pro(line))
14. **for** word **in** pro(line):
15. **if** word == '':
16. #                 print(word)
17. **continue**
18. result[idx, word\_dict[word]] += 1
20. **return** result
22. cnt = count(data.data)
23. **print**(cnt)



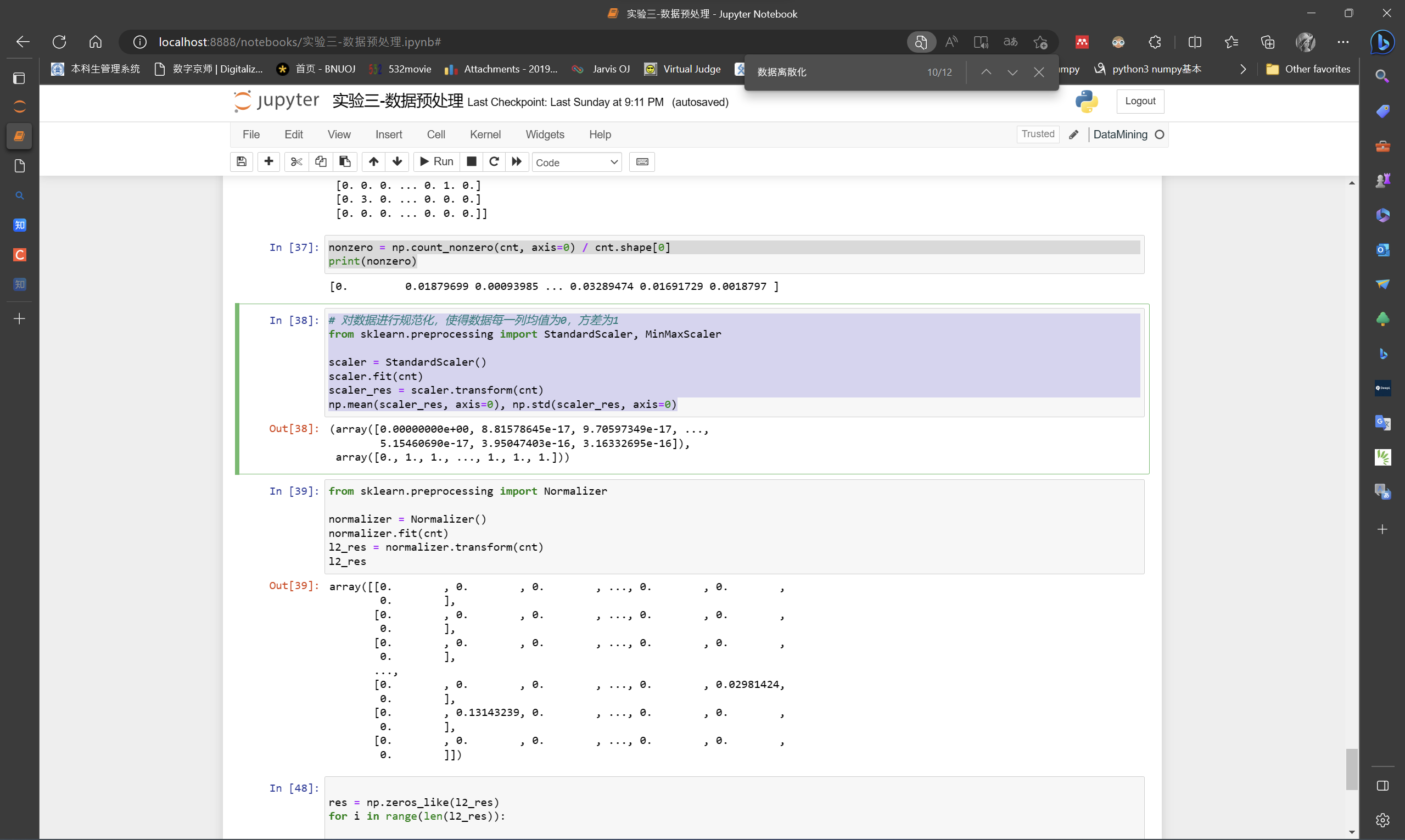
3. 计算`每列`非零元素的比例

1. nonzero = np.count\_nonzero(cnt, axis=0) / cnt.shape[0]
2. **print**(nonzero)



4. 对数据进行规范化，使得数据每一列均值为0，方差为1

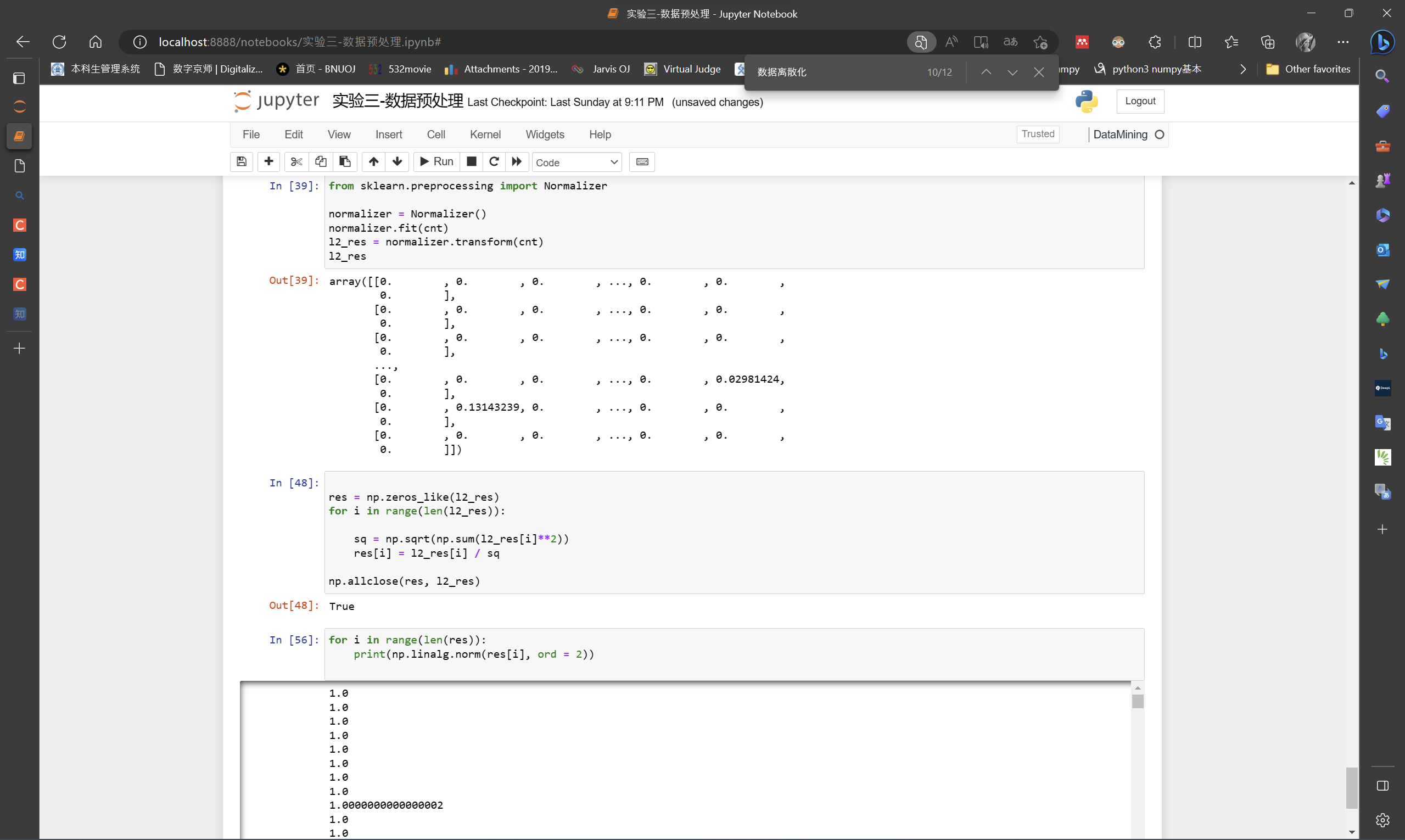
1. # 对数据进行规范化，使得数据每一列均值为0，方差为1
2. **from** sklearn.preprocessing **import** StandardScaler, MinMaxScaler
4. scaler = StandardScaler()
5. scaler.fit(cnt)
6. scaler\_res = scaler.transform(cnt)
7. np.mean(scaler\_res, axis=0), np.std(scaler\_res, axis=0)



5. 对每行数据进行范数规范化，

其中，为第行的向量，表示数组中位置为的元素。

1. **from** sklearn.preprocessing **import** Normalizer
3. normalizer = Normalizer()
4. normalizer.fit(cnt)
5. l2\_res = normalizer.transform(cnt)
6. l2\_res
8. res = np.zeros\_like(l2\_res)
9. **for** i **in** range(len(l2\_res)):
11. sq = np.sqrt(np.sum(l2\_res[i]\*\*2))
12. res[i] = l2\_res[i] / sq
14. np.allclose(res, l2\_res)
16. **for** i **in** range(len(res)):
17. **print**(np.linalg.norm(res[i], ord = 2))



## 六、个人感想

本次实验主题是数据预处理，旨在掌握数据预处理概念和基本的数据预处理方法。实验包含两个任务，分别是基于信息熵的数据离散化算法和对文本信息的预处理。

对于第一个任务，本次实验使用的是一份用户最近点击的20个商品的价格与是否加入购物车对应关系的数据文件。通过信息熵的计算，将数据进行离散化处理，得到一组新的数据集。

对于第二个任务，本次实验是对文本信息进行预处理。利用词袋模型，将文本数据向量化。具体步骤包括：对文本进行分词、统计每篇文章出现的词的次数、计算每列非零元素的比例、对数据进行规范化，使得数据每一列均值为0，方差为1，最后对每行数据进行 𝑙2−范数规范化。

总的来说，本次实验通过实践操作，让我了解了数据预处理的相关概念和方法，加深了对数据处理的理解和实践能力。