

**超市关联规则实验及其分析报告**

学 院 信息工程学院

专 业 数据科学与大数据技术

学 号 419105060102

姓 名 姜欣微

指导教师 贾 楠

2021年10月22日

目录

[一、 数据信息 2](#_Toc85822538)

[1. 数据来源 2](#_Toc85822539)

[2. 数据属性 2](#_Toc85822540)

[3. 导入数据 3](#_Toc85822541)

[二、 数据探索 3](#_Toc85822542)

[1. 数据描述探索 3](#_Toc85822543)

[三、 数据预处理 4](#_Toc85822544)

[1. 去除唯一属性 4](#_Toc85822545)

[2. 处理缺失值 5](#_Toc85822546)

[3. 数据条目集生成 5](#_Toc85822547)

[a) 导入中间数据文件data1命名为data2 5](#_Toc85822548)

[b) 重编码所有条目 5](#_Toc85822549)

[c) 形成条目集列表 6](#_Toc85822550)

[四、 关联规则实现 6](#_Toc85822551)

[1. 关联规则实现准备 7](#_Toc85822552)

[2. 进行关联规则实现 7](#_Toc85822553)

[a) 选取数据集中非重复数据形成一个C1集合 7](#_Toc85822554)

[b) 选出频繁集 8](#_Toc85822555)

[c) 关联规则计算 10](#_Toc85822556)

[五、 结果与总结 11](#_Toc85822557)

[1. 结果展示 11](#_Toc85822558)

[2. 总结 11](#_Toc85822559)

[六、 mlxtend实现（导包） 11](#_Toc85822560)

[1. 下载mlxtend模块 11](#_Toc85822561)

[2. 准备数据集 12](#_Toc85822562)

[3. 找出频繁集 12](#_Toc85822563)

[4. 找出关联规则 14](#_Toc85822564)

[5. 结论 15](#_Toc85822565)

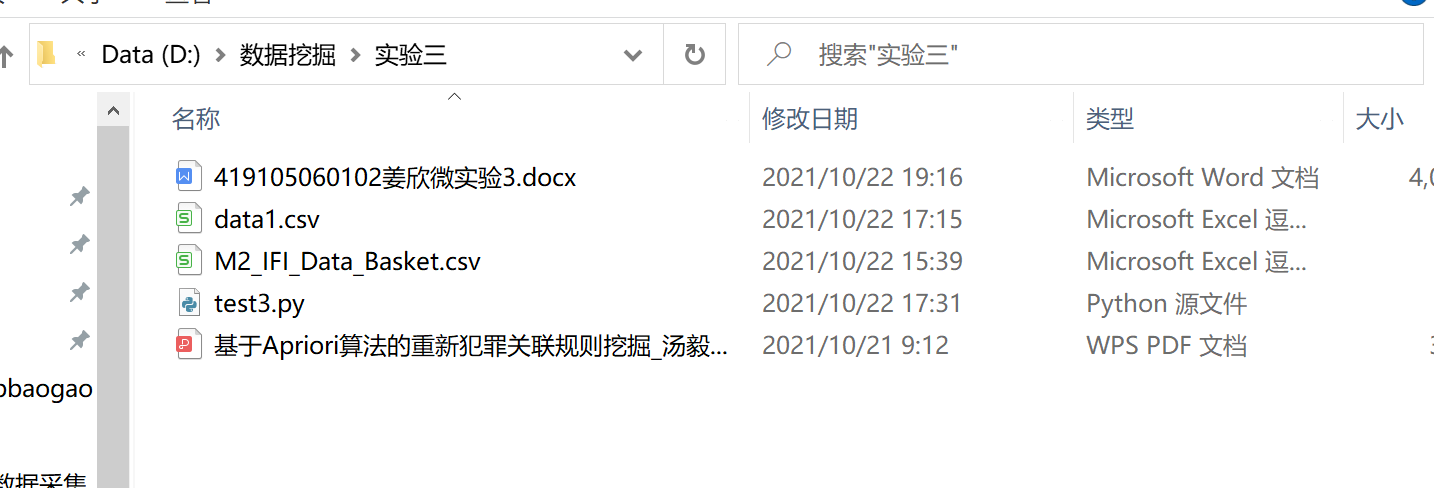
[七、 参考链接 15](#_Toc85822566)

[八、 附录（源码） 15](#_Toc85822567)

1. 数据信息
2. 数据来源



<https://zhuanlan.zhihu.com/p/32780493>

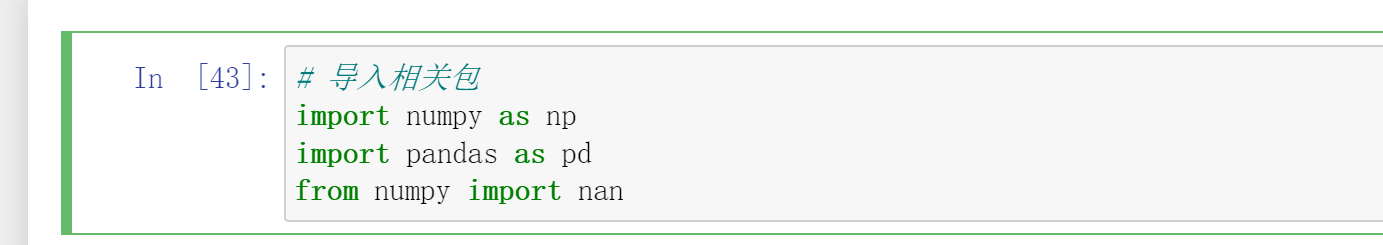


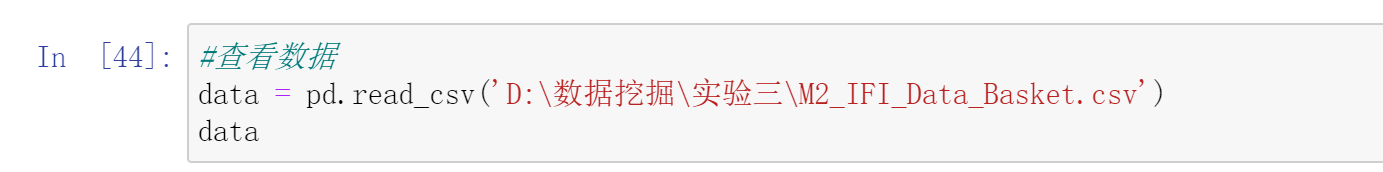
1. 数据属性

|  |  |
| --- | --- |
| M2\_IFI\_Data\_Basket | |
| 1：卡号 | Card num. |
| 2：数量 | Amount |
| 3：付款 | Payment |
| 4：性别 | Gender |
| 5：房客 | Tenant |
| 6：收入 | Income |
| 7：年龄 | Age |
| 8：水果和蔬菜 | Fruits & vegetables |
| 9：肉 | Meat |
| 10：乳制品 | Milk products |
| 11：罐装蔬菜 | Canned vegetables |
| 12：罐头肉 | Canned meat |
| 13：冷冻食物 | Frozen goods |
| 14：啤酒 | Beer |
| 15：葡萄酒 | Wine |
| 16：汽水 | Soda drinks |
| 17：鱼 | Fish |
| 18：纺织品 | Textile |

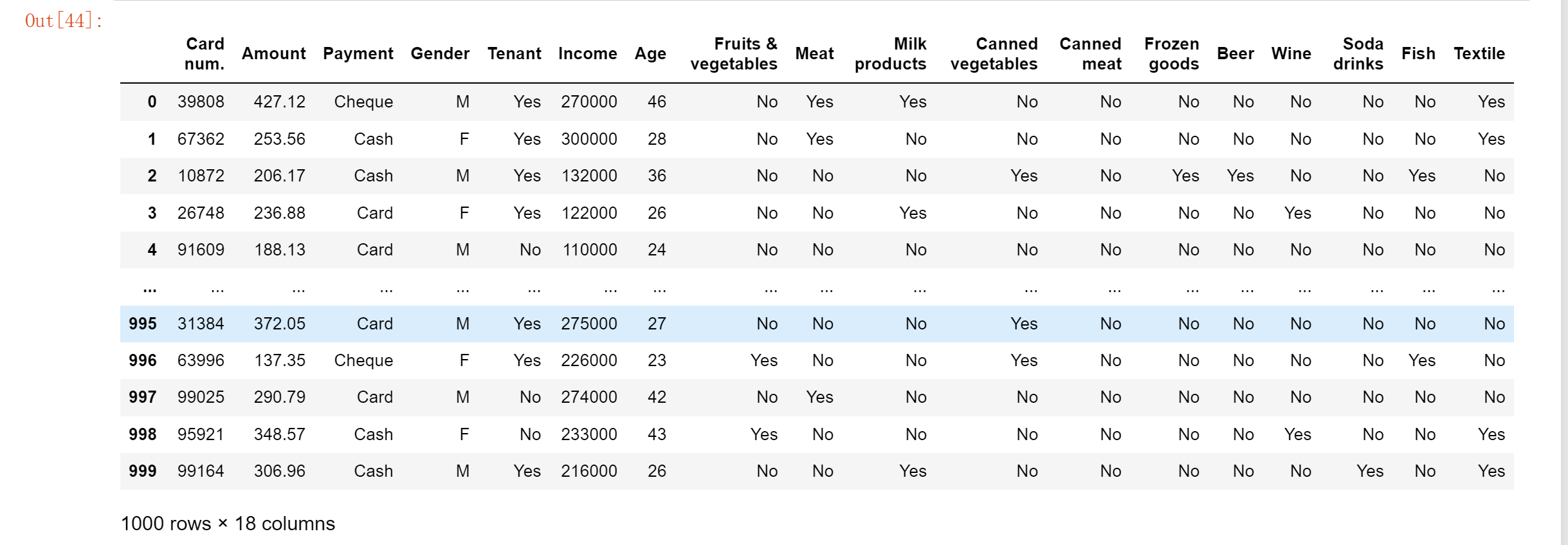
1. 导入数据

此次实验需要导的包



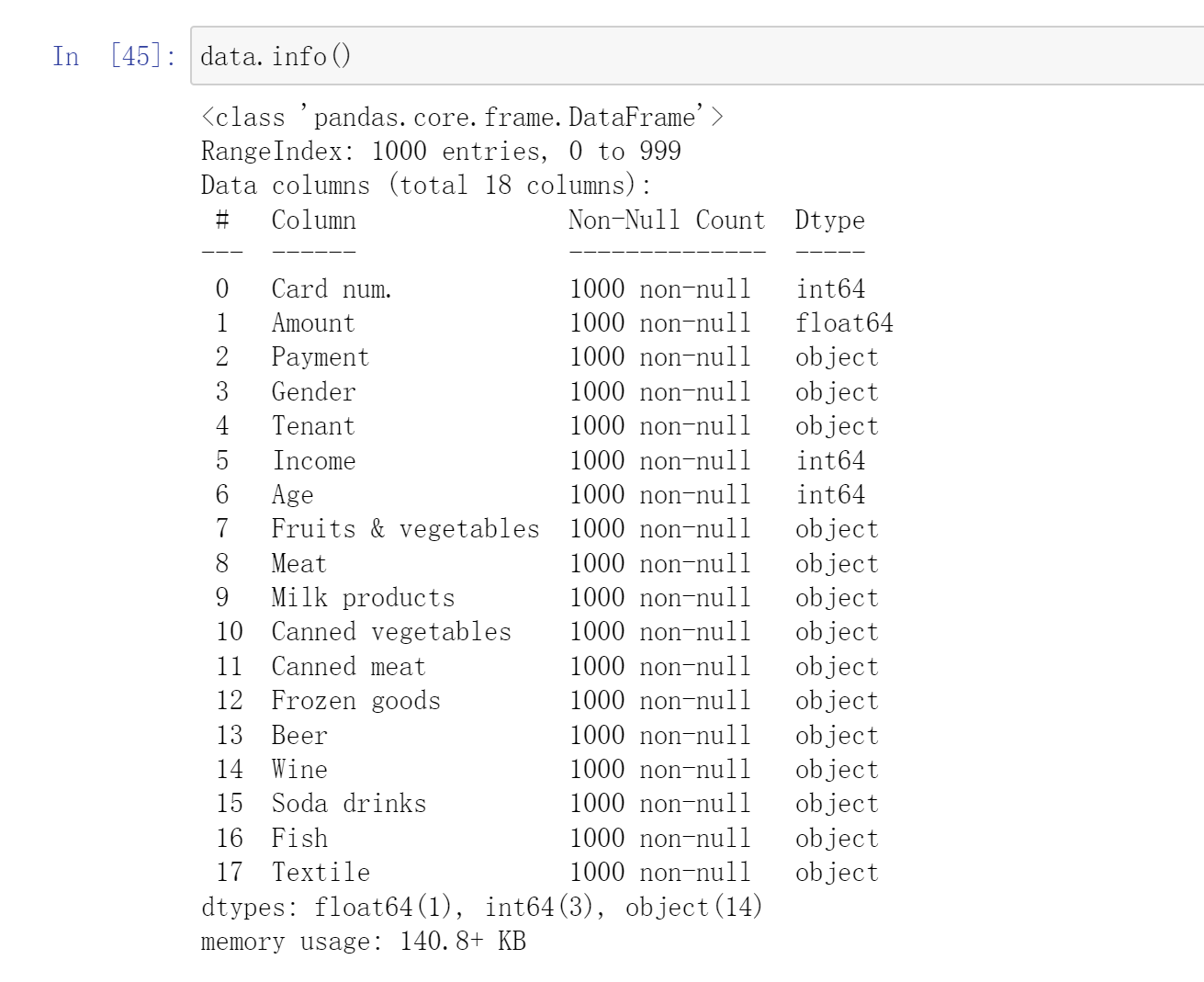


数据一览

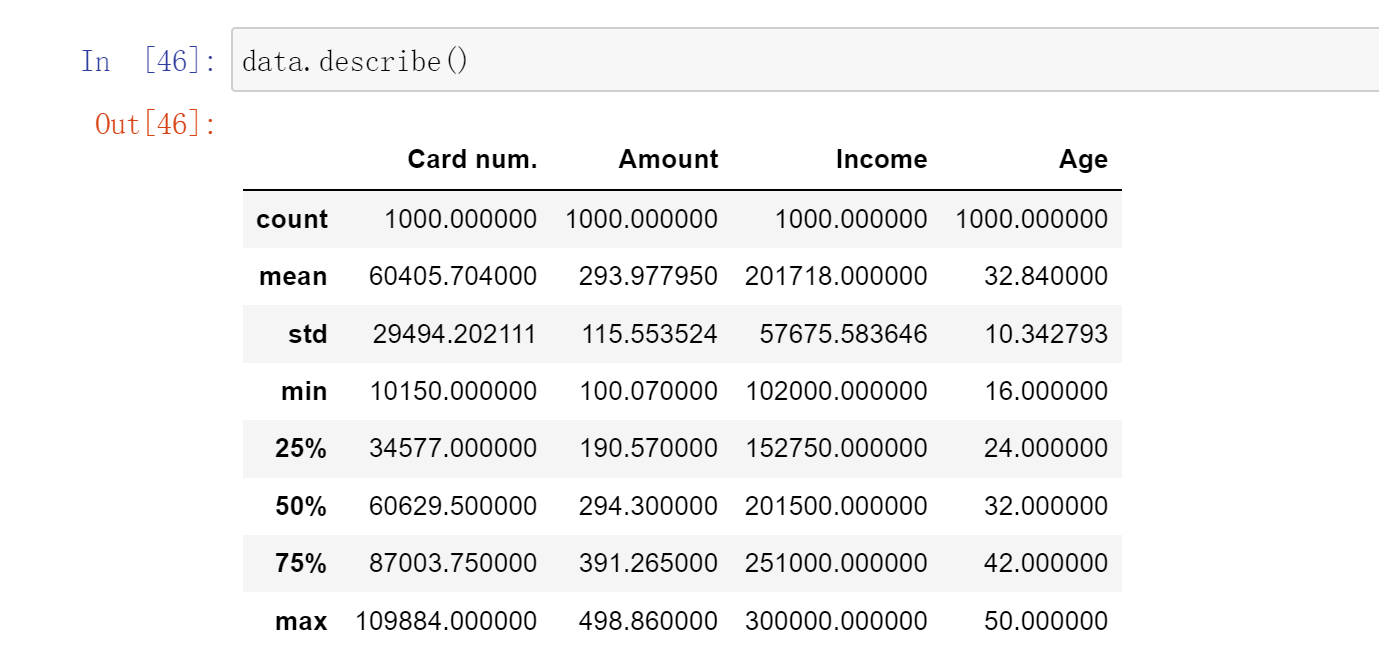


1. 数据探索
2. 数据描述探索

查看数据属性信息



查看数据基本信息



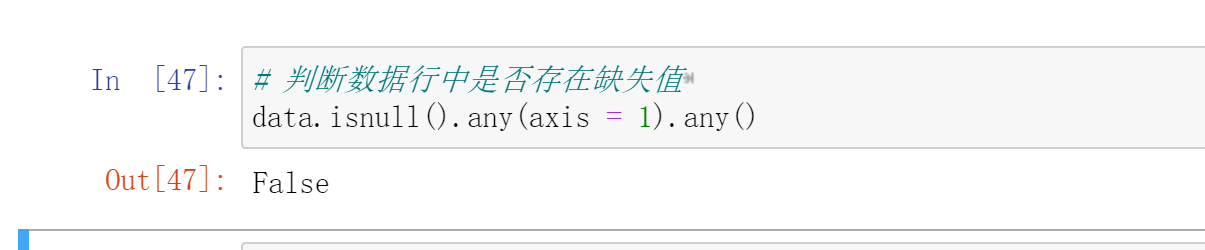
1. 数据预处理
2. 去除唯一属性

对于该数据集前七列特征值在此次关联规则中不需要，因此提取后7到18列属性值存入data1中写出



1. 处理缺失值

查看数据集中是否存在缺失值



不存在缺失值，并且由于此次分类数据的性质，零值的存在并不能代表缺失值，因此缺失值没有，不做缺失值处理。

1. 数据条目集生成
2. 导入中间数据文件data1命名为data2



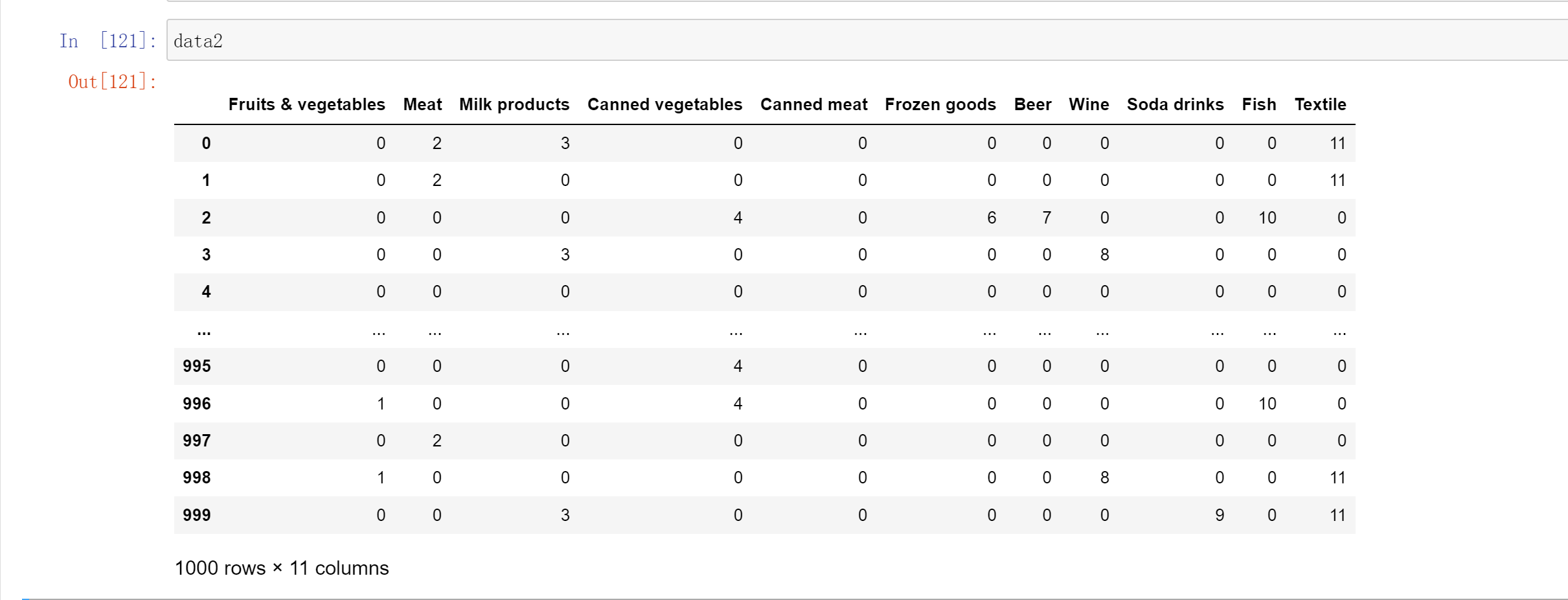
data2一览



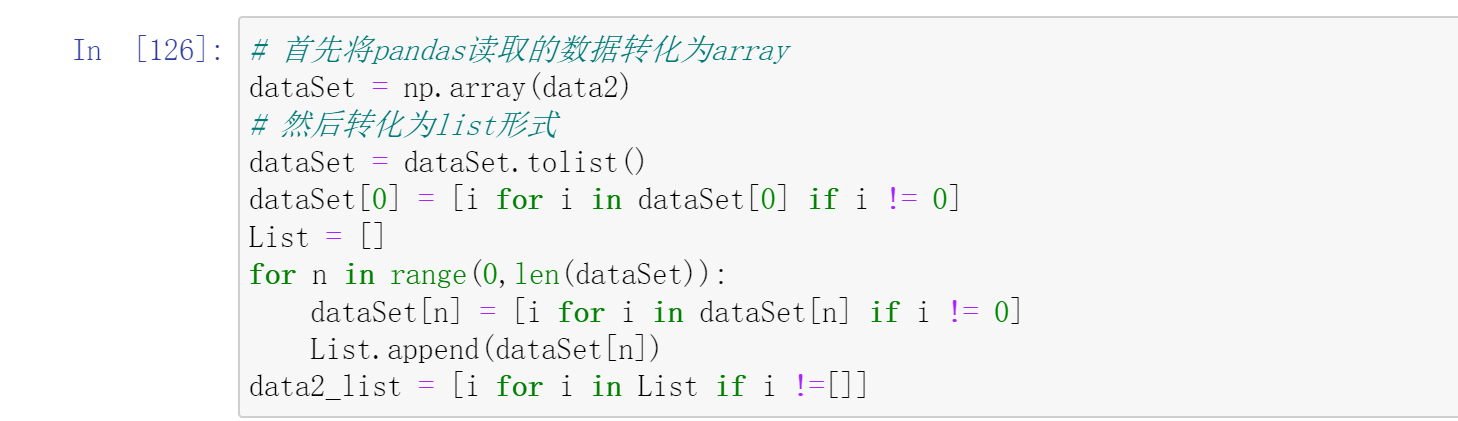
1. 重编码所有条目



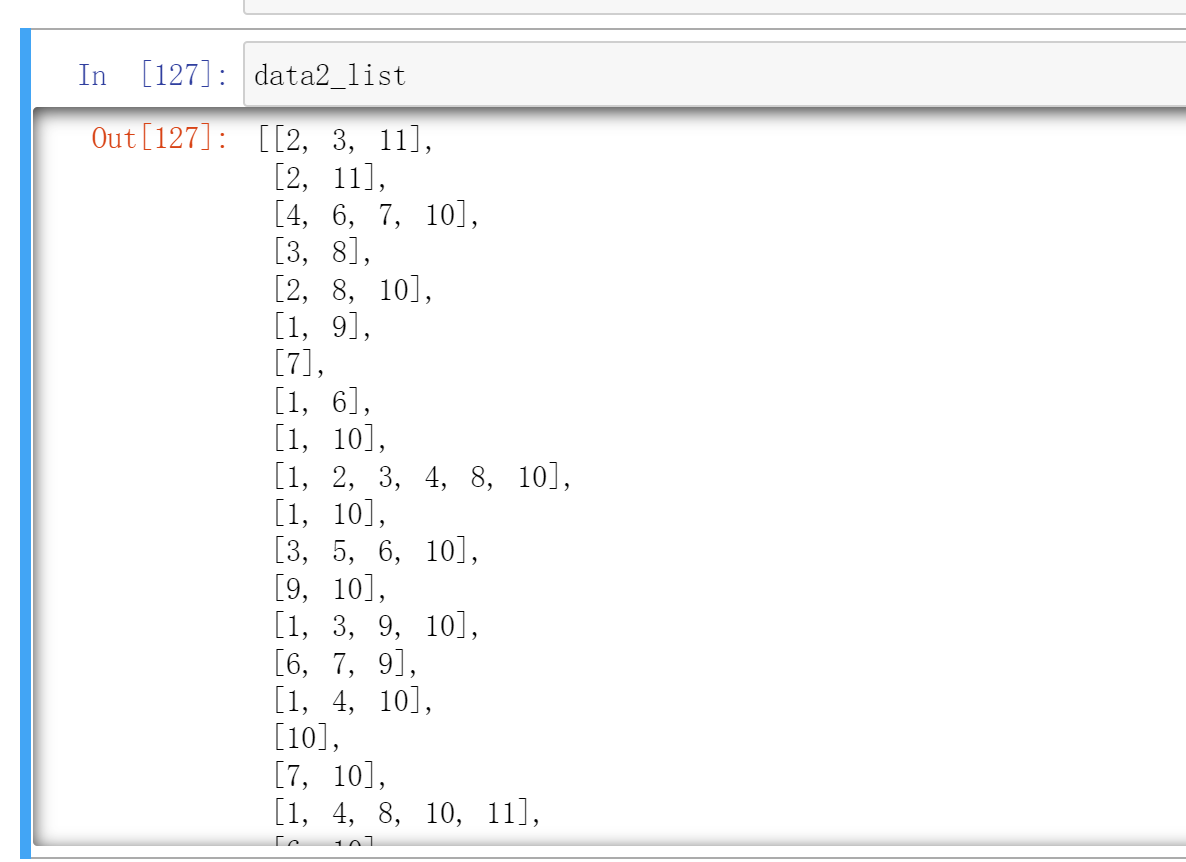
data2编码后结果



1. 形成条目集列表

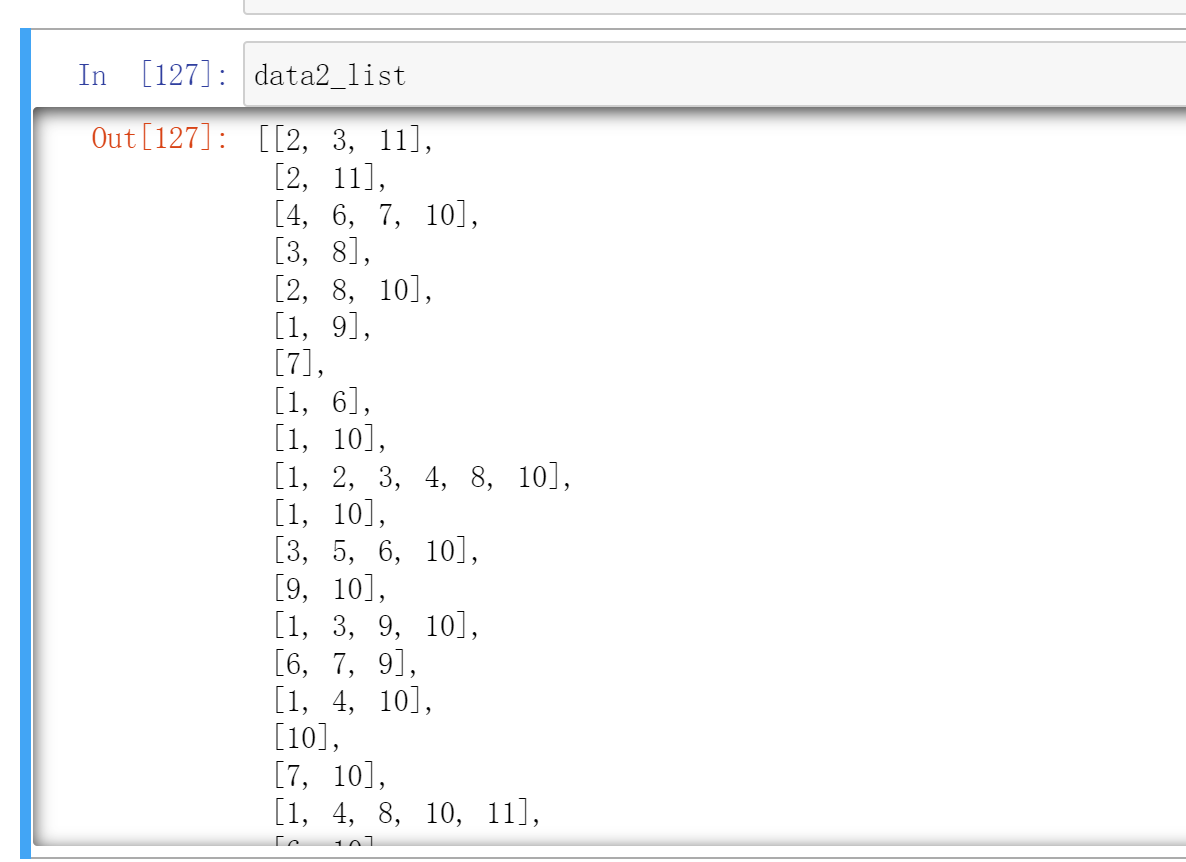


结果如下

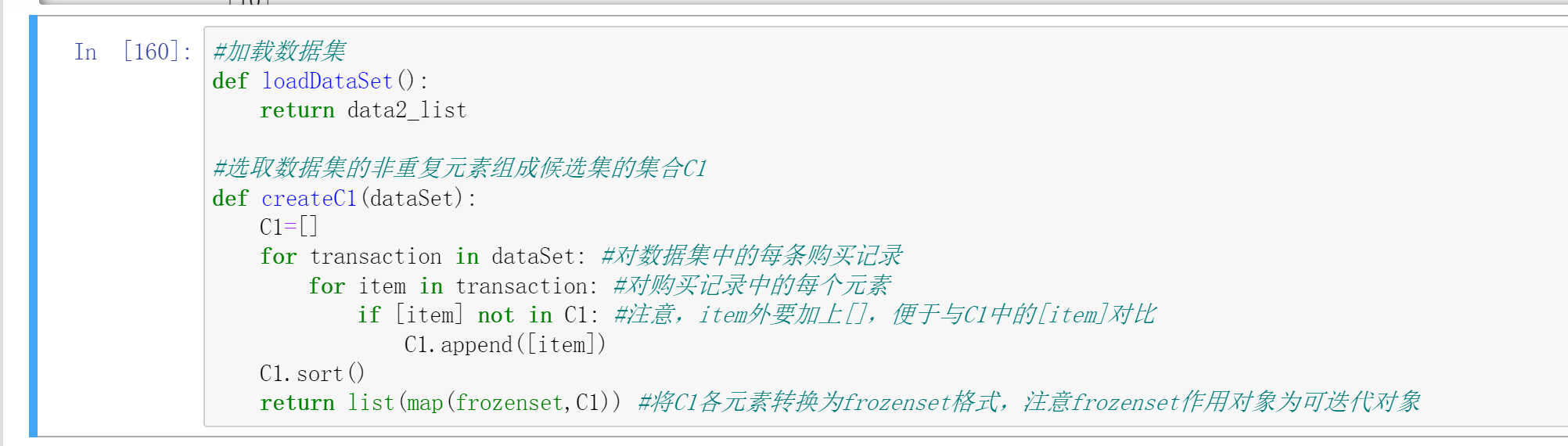


1. 关联规则实现
2. 关联规则实现准备

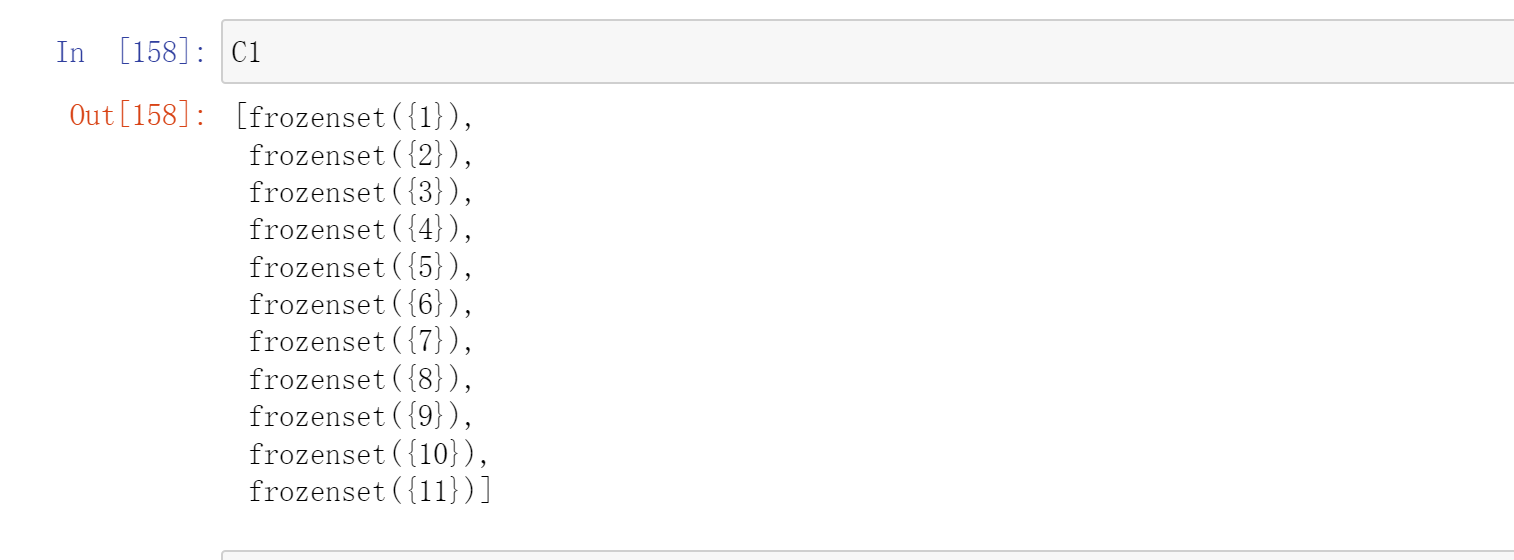
准备条目集列表data2\_list



1. 进行关联规则实现
2. 选取数据集中非重复数据形成一个C1集合



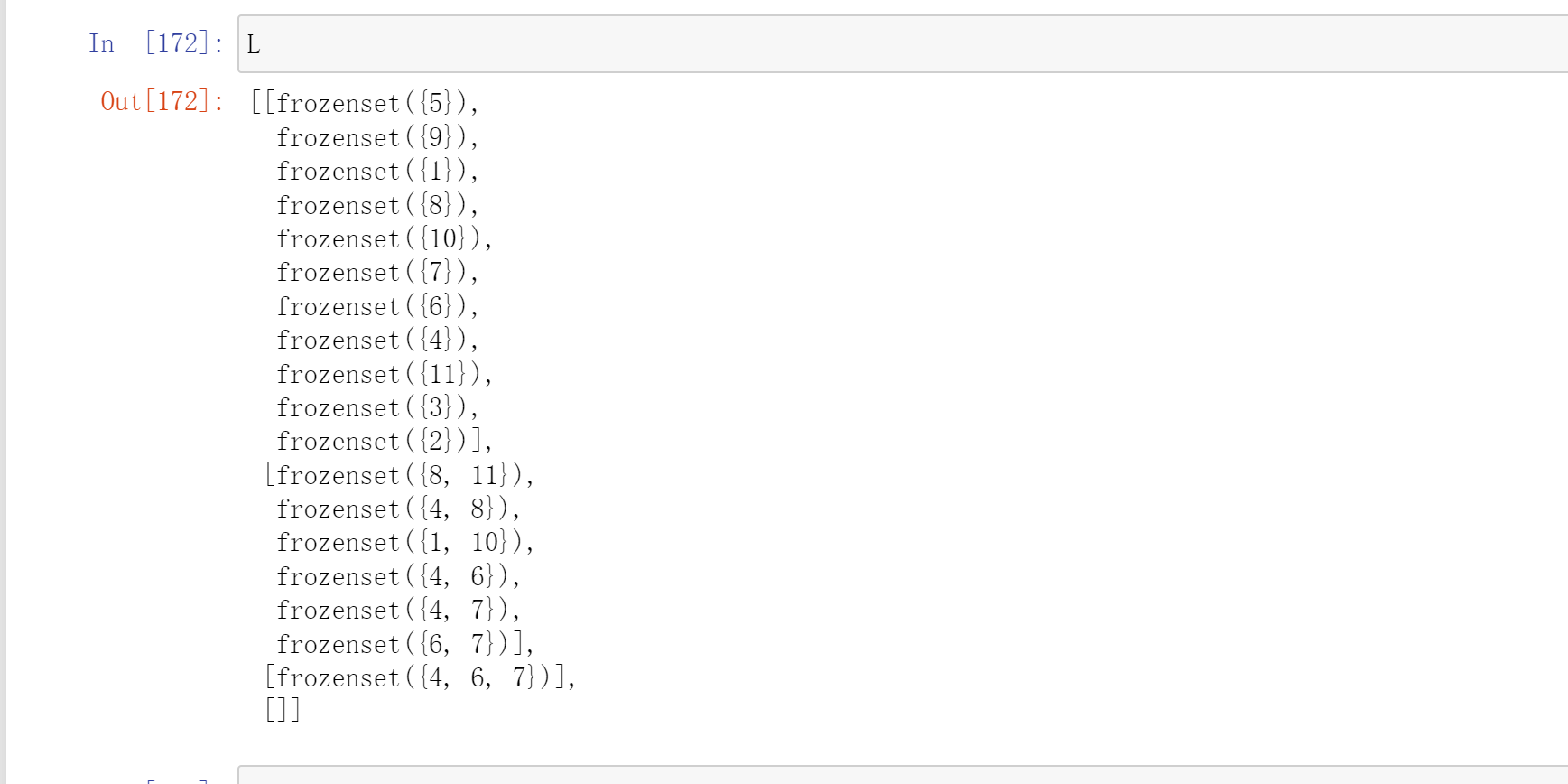
结果如下



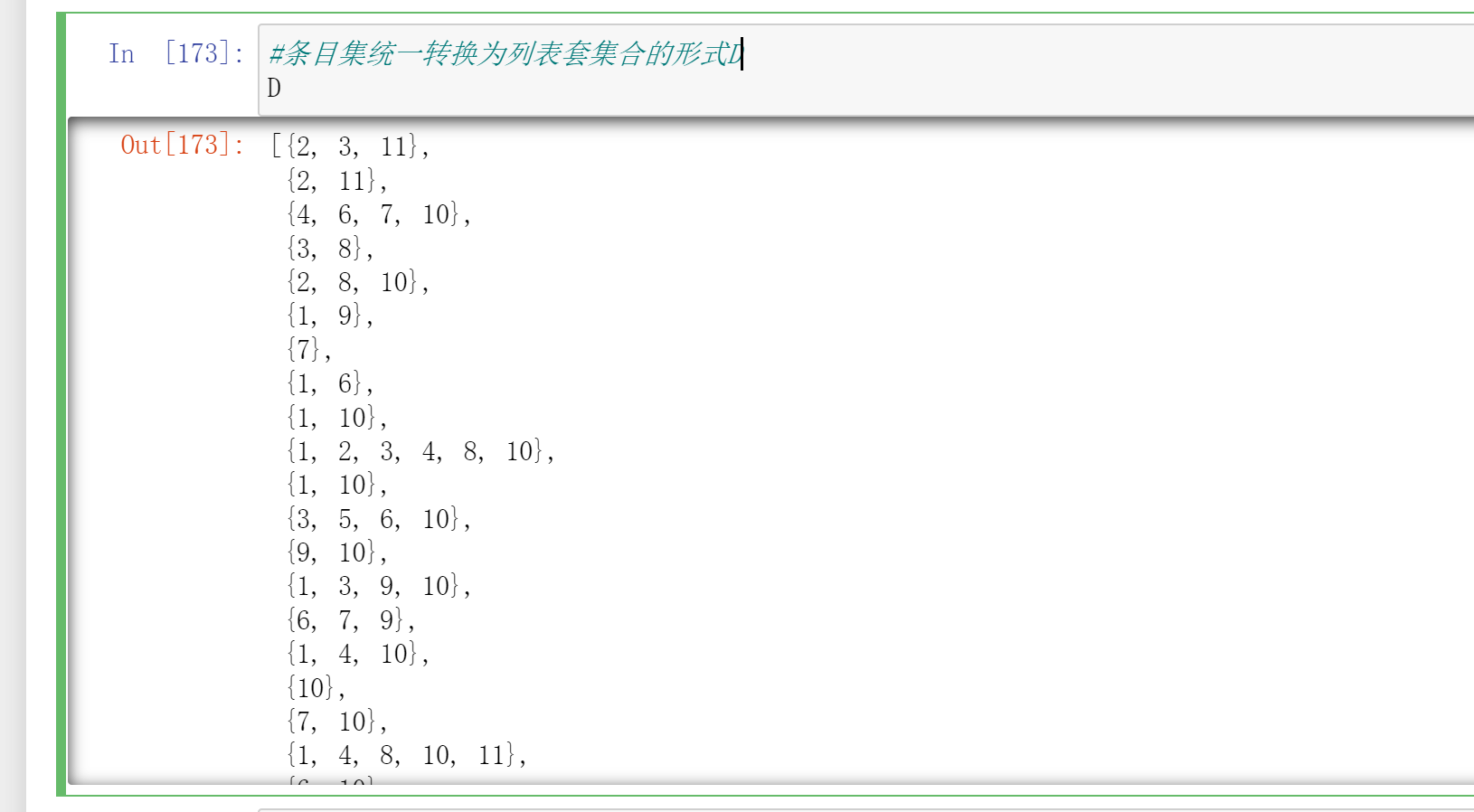
1. 选出频繁集



频繁集为L



条目集D



结果一览

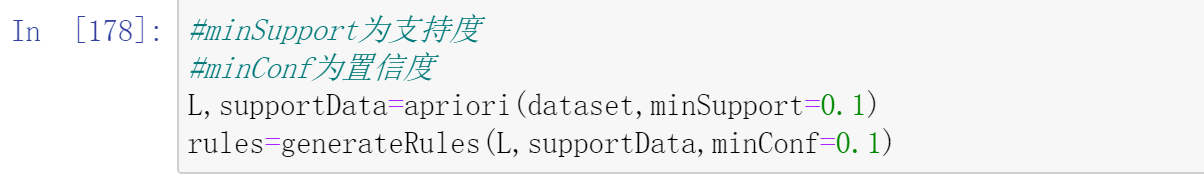


1. 关联规则计算

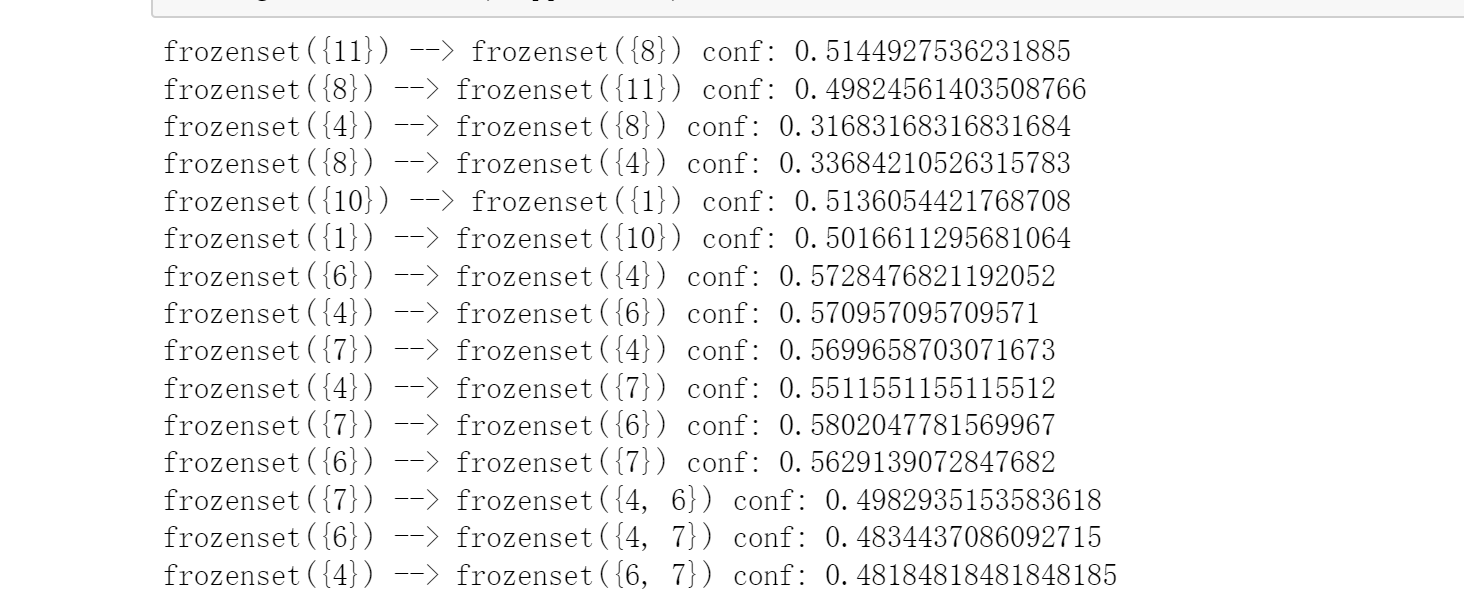
代码如下



带回原数据集data2\_list



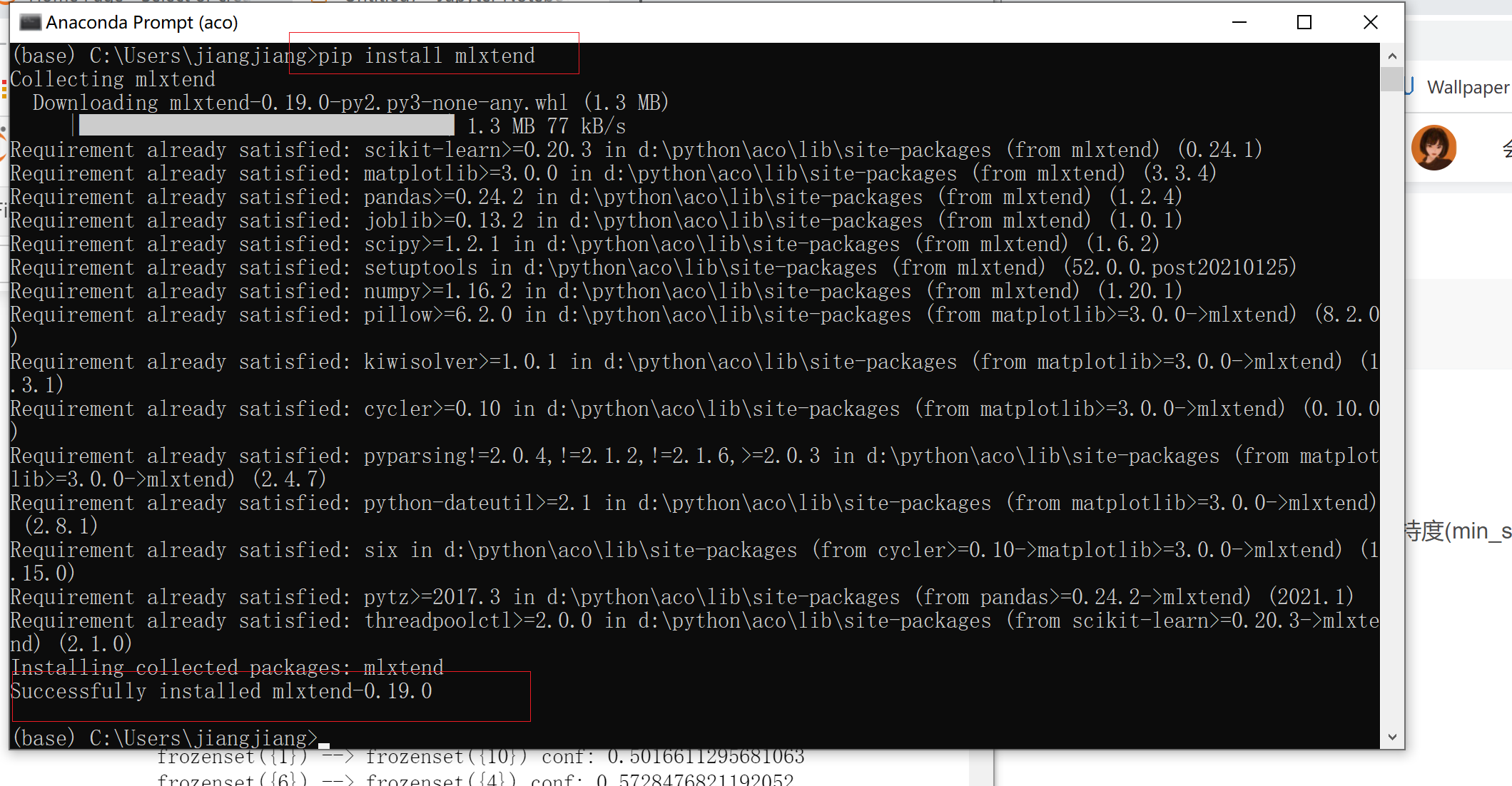
1. 结果与总结
2. 结果展示



1. 总结

从关联规则中可以看出7与6关联性已经是最大的了，6与4也是较大关联，但由于数据集只有一千条。关联规则置信度都偏低，回到数据本身，6为冷冻食物，4为罐装蔬菜，7为啤酒，都是一些速食，因此推断该数据来源地并不居家。

1. mlxtend实现（导包）
2. 下载mlxtend模块



需要导的包



1. 准备数据集

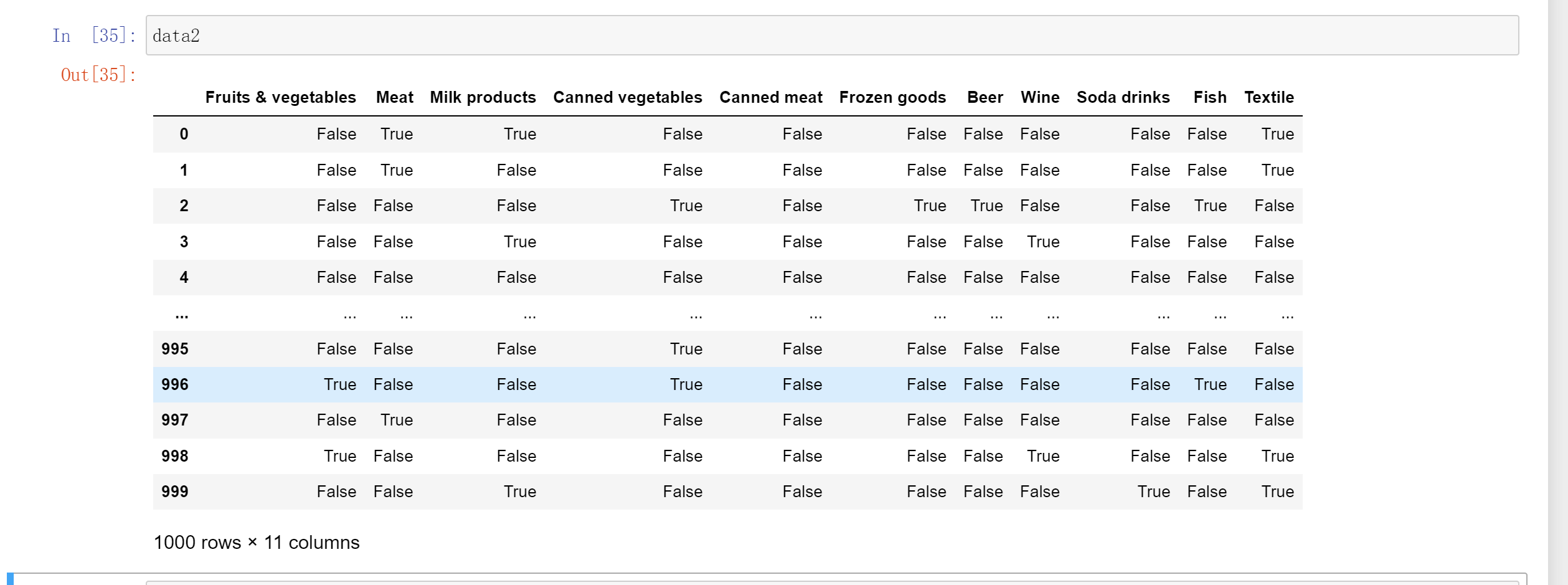
使用data1中间数据命名为data2



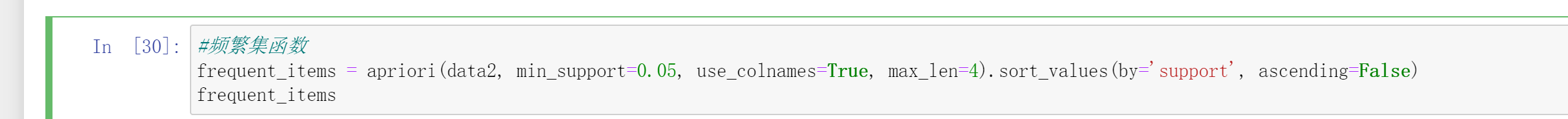
重编码



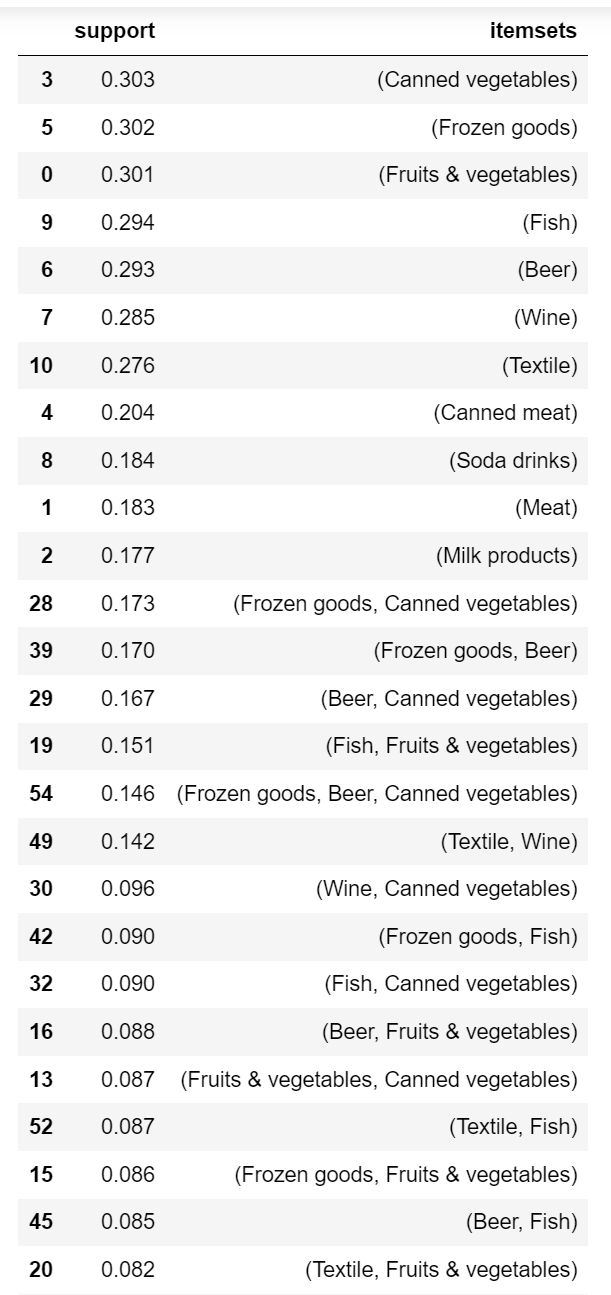
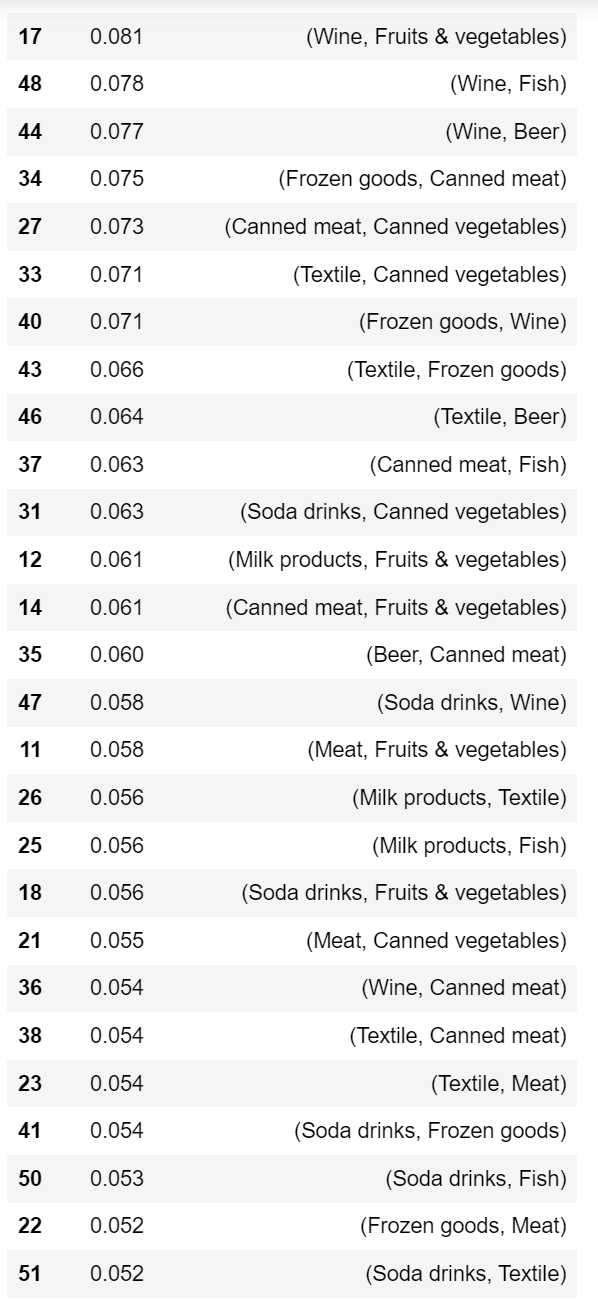
结果如下

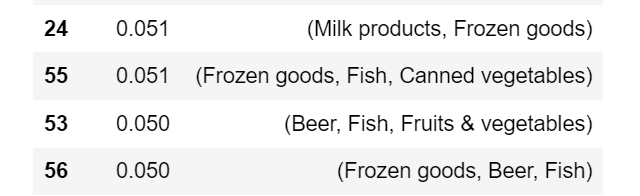


1. 找出频繁集

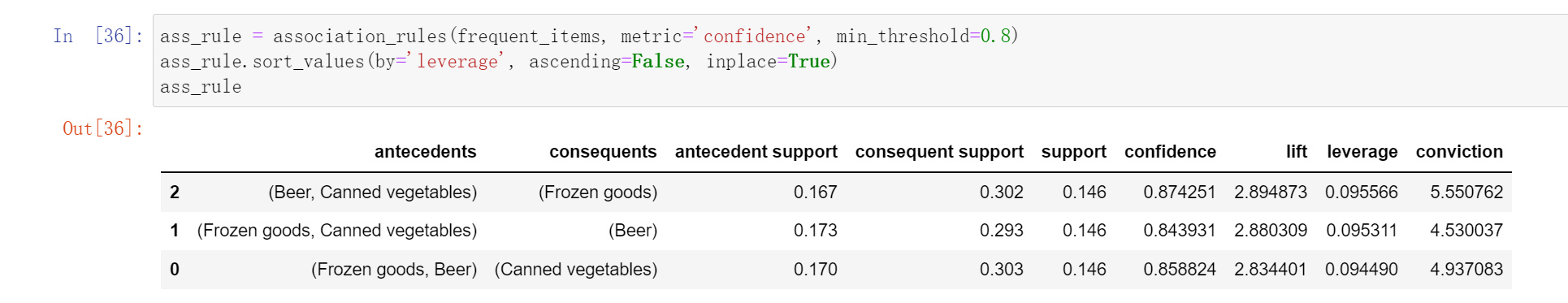


结果如下



1. 找出关联规则



1. 结论

通过两种方法实现关联规则算法，可以发现导包的方式更加快捷并且清晰明了的查看关联规则，比上述算法需要人为总结要优秀便利。结果上没有较大误差。

1. 参考链接

数据探索：

<https://blog.csdn.net/Orange_Spotty_Cat/article/details/80606980>

关联规则：

<https://zhuanlan.zhihu.com/p/52883806>

<https://www.jianshu.com/p/2ee0a247a8cc>

<https://blog.csdn.net/weixin_38008864/article/details/102578070>

1. 附录（源码）

# 导入相关包

import numpy as np

import pandas as pd

from numpy import nan

#%matplotlib inline

import matplotlib.pyplot as plt

plt.style.use('seaborn')

import seaborn as sns

sns.set\_style("whitegrid")

#查看数据

data = pd.read\_csv('D:\数据挖掘\实验三\M2\_IFI\_Data\_Basket.csv')

data

# 判断数据行中是否存在缺失值

data.isnull().any(axis = 1).any()

#提取需要的属性值

data1 = data.iloc[:,7:18]

data1.to\_csv('D:\数据挖掘\实验三\data1.csv')

#数据预处理重编码

data2 = pd.read\_csv('D:\数据挖掘\实验三\data1.csv',usecols=[1,2,3,4,5,6,7,8,9,10,11])

#数据处理，重编码，以列为单位对每一列进行重编码生成条目集

#Fruits & vegetables

data2['Fruits & vegetables'] = data2['Fruits & vegetables'].apply(lambda x:1 if x == "Yes" else 0)

#Meat

data2['Meat'] = data2['Meat'].apply(lambda x:2 if x == "Yes" else 0)

#Milk products

data2['Milk products'] = data2['Milk products'].apply(lambda x:3 if x == "Yes" else 0)

# Canned vegetables

data2['Canned vegetables'] = data2['Canned vegetables'].apply(lambda x:4 if x == "Yes" else 0)

# Canned meat

data2['Canned meat'] = data2['Canned meat'].apply(lambda x:5 if x == "Yes" else 0)

# Frozen goods

data2['Frozen goods'] = data2['Frozen goods'].apply(lambda x:6 if x == "Yes" else 0)

# Beer

data2['Beer'] = data2['Beer'].apply(lambda x:7 if x == "Yes" else 0)

# Wine

data2['Wine'] = data2['Wine'].apply(lambda x:8 if x == "Yes" else 0)

# Soda drinks

data2['Soda drinks'] = data2['Soda drinks'].apply(lambda x:9 if x == "Yes" else 0)

#Fish

data2['Fish'] = data2['Fish'].apply(lambda x:10 if x == "Yes" else 0)

# Textile

data2['Textile'] = data2['Textile'].apply(lambda x:11 if x == "Yes" else 0)

# 首先将pandas读取的数据转化为array

dataSet = np.array(data2)

# 然后转化为list形式

dataSet = dataSet.tolist()

dataSet[0] = [i for i in dataSet[0] if i != 0]

List = []

for n in range(0,len(dataSet)):

dataSet[n] = [i for i in dataSet[n] if i != 0]

List.append(dataSet[n])

data2\_list = [i for i in List if i !=[]]

#加载数据集

def loadDataSet():

return data2\_list

#选取数据集的非重复元素组成候选集的集合C1

def createC1(dataSet):

C1=[]

for transaction in dataSet: #对数据集中的每条购买记录

for item in transaction: #对购买记录中的每个元素

if [item] not in C1: #注意，item外要加上[]，便于与C1中的[item]对比

C1.append([item])

C1.sort()

return list(map(frozenset,C1)) #将C1各元素转换为frozenset格式，注意frozenset作用对象为可迭代对象

#由Ck产生Lk：扫描数据集D，计算候选集Ck各元素在D中的支持度，选取支持度大于设定值的元素进入Lk

def scanD(D,Ck,minSupport):

ssCnt={}

for tid in D: #对数据集中的每条购买记录

for can in Ck: #遍历Ck所有候选集

if can.issubset(tid): #如果候选集包含在购买记录中，计数+1

ssCnt[can]=ssCnt.get(can,0)+1

numItems=float(len(D)) #购买记录数

retList=[] #用于存放支持度大于设定值的项集

supportData={} #用于记录各项集对应的支持度

for key in ssCnt.keys():

support=ssCnt[key]/numItems

if support>=minSupport:

retList.insert(0,key)

supportData[key]=support

return retList,supportData

#由Lk产生Ck+1

def aprioriGen(Lk,k): #Lk的k和参数k不是同一个概念，Lk的k比参数k小1

retList=[]

lenLk=len(Lk)

for i in range(lenLk):

for j in range(i+1,lenLk): #比较Lk中的每一个元素与其他元素

L1=list(Lk[i])[:k-2];L2=list(Lk[j])[:k-2]

L1.sort();L2.sort()

if L1==L2: #若前k-2项相同，则合并这两项

retList.append(Lk[i]|Lk[j])

return retList

# 主函数，由频繁项集以及对应的支持度，得到各条规则的置信度，选择置信度满足要求的规则为关联规则

def apriori(dataSet,minSupport=0.5):

C1=createC1(dataSet)

D=list(map(set,dataSet))

L1,supportData=scanD(D,C1,minSupport)

L=[L1]

k=2

while len(L[k-2])>0: #当L[k]为空时，停止迭代

Ck=aprioriGen(L[k-2],k) #L[k-2]对应的值是Lk-1

Lk,supK=scanD(D,Ck,minSupport)

supportData.update(supK)

L.append(Lk)

k+=1

return L,supportData

# 为了避免将所有数据都对比一遍，采用与上述相同的逻辑减少计算量——一层一层计算筛选

def generateRules(L,supportData,minConf=0.7):

bigRuleList=[]

for i in range(1,len(L)):

for freqSet in L[i]:

H1=[frozenset([item]) for item in freqSet] # H1是频繁项集单元素列表，是关联规则中a->b的b项

if i>1:

rulesFromConseq(freqSet,H1,supportData,bigRuleList,minConf)

else:

calConf(freqSet,H1,supportData,bigRuleList,minConf)

return bigRuleList

# 置信度计算函数

def calConf(freqSet,H,supportData,brl,minConf=0.7):

prunedH=[] # 用于存放置信度满足要求的关联规则的b项，即“提纯后的H”

for conseq in H:

conf=supportData[freqSet]/supportData[freqSet-conseq]

if conf>=minConf:

print (freqSet-conseq,'-->',conseq,'conf:',conf)

brl.append([freqSet-conseq,conseq,conf])

prunedH.append(conseq)

return prunedH

# 关联规则合并函数

def rulesFromConseq(freqSet,H,supportData,brl,minConf=0.7):

m=len(H[0])

if len(freqSet)>(m+1): #查看频繁项集freqSet是否大到可以移除大小为m的子集

Hmp1=aprioriGen(H,m+1) # 从Hm合并Hm+1

Hmp1=calConf(freqSet,Hmp1,supportData,brl,minConf)

if len(Hmp1)>1: #若合并后的Hm+1的元素大于1个，则继续合并

rulesFromConseq(freqSet,Hmp1,supportData,brl,minConf)

#minSupport为支持度

#minConf为置信度

L,supportData=apriori(dataSet,minSupport=0.1)

rules=generateRules(L,supportData,minConf=0.1)