**Apriori关联规则挖掘**

# 1简介

在某个商场，销售经理希望通过重新布置商品的摆放情况来提高销售额，那么该如何摆放既有利于方便顾客购买，也刺激顾客的购买欲呢？商品的摆放并不是随意的，首先是将同类商品放在一起外；其次，可以考虑将客户会经常同时购买的物品放在相邻的区域，可以有效地促进消费方便客户。在这种策略的指引下，于是这位销售经理开始统计购物小票，希望能从中有所发现。

# 2环境及数据准备

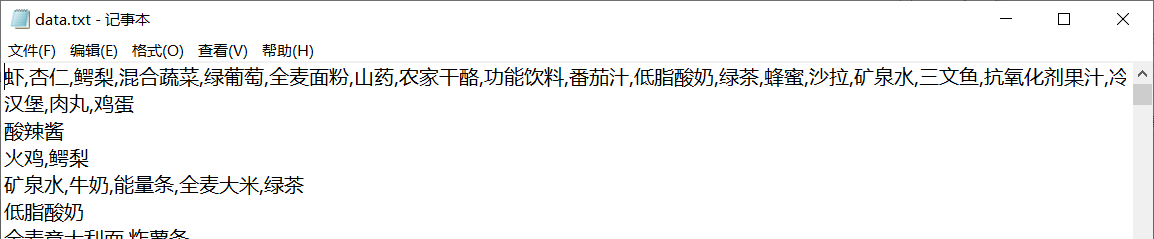
## 2.1环境及依赖

本实验需要安装第三方mlxtend依赖包，请大家使用pip进行安装，所用到的指令为：pip install mlxtend

import pandas as pd  
from mlxtend.frequent\_patterns import apriori  
from mlxtend.frequent\_patterns import association\_rules

## 2.2数据说明

请大家打开文件目录下的“data.txt”文档。



可以看到，我们的小票数据集每一行为一个样本，购买的商品之间用逗号隔开。

## 2.3数据读入

因为我们的原始数据集不再是标准的csv或excel表格，而是一个不规则文本文件。它的读取不宜使用pandas进行，而适合使用python原生方法。

首先，我们要打开文件，使用open函数实现，需要指明数据集的路径、进行的操作和文件的编码。

*# 打开文件*origin\_data = open(**"D:/input/data.txt"**, **"r"**, encoding=**'utf-8'**)

我们可以查看一下数据集的第一行，因为不是通过pandas读入文件，我们此处不能使用head函数。

*# 读取一行数据*line = origin\_data.readline()  
line //可以看到，我们的文件读取进行成功。

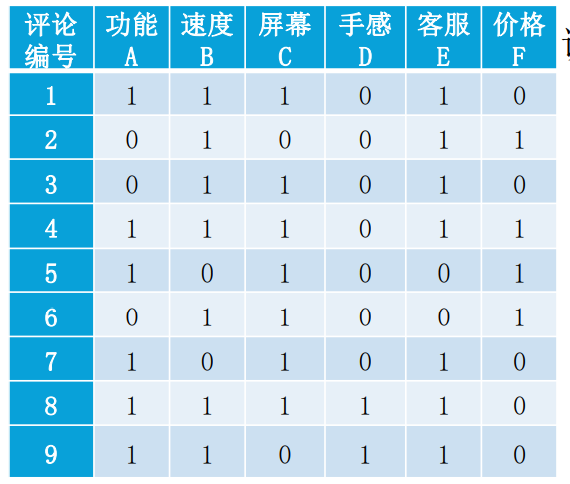
# 3数据处理

我们要进行关联规则挖掘，首先要把现在的购物小票数据转换为标准的csv或excel数据表。

这里我们选择大家更熟悉的excel数据表进行转换。

## 3.1获取列名集

我们所需的数据表是如下图所示的标准数据集。



在目的数据表中，每个商品名作为一列，每个样本作为一行。每个样本购买的商品对应取值为1，未购买的商品对应取值为0。

首先我们需要获取小票数据中出现的所有商品名，第一步建立一个空列表，用于商品名存放。

*# 创建空列表*title\_list = []

接下来逐行读取数据集，对每个样本根据逗号分割成单个商品名，将这些商品名添加到列表中。

*# 迭代所有行，直到把文件所有行读完*while line:  
 *# 去掉换行符* line = line.strip()  
 *# 把一行数据按照“，”隔开为一个个商品名* product\_items = line.split(**','**)  
 *# 将这些商品名都加入到列表中* for item in product\_items:  
 title\_list.append(item)  
 *# 读取下一行，继续迭代* line = origin\_data.readline()

数据集中存在很多相同的值，这也导致我们的title\_list中有很多重复值，而目的数据集列名是不能有重复的。故而我们使用数据结构set进行去重。

*# 使用set数据结构去重*title\_list = set(title\_list)

可以将去重后的列名数组打印出来。

*# 打印商品列表*title\_list//查看列表长度，即为数据集的列数。

*# 打印商品列表长度*print(**"共有"**, len(title\_list) ,**"种商品"**)

可以看到，我们的小票数据中共出现了116种商品，而对应地，目的数据集也会有116列。

完成文件的一次读取后，我们需要关闭文件。

*# 关闭文件*origin\_data.close()

## 3.2数据转换

上一步，我们已经得到了所有商品名的列表，接下来我们将根据原数据集完成目的数据集的创建。

首先，创建一个dataframe用于存储目的数据集。它的列即为此前得到的商品名列表项。

*# 创建一个空的dataframe用来存放处理后的数据集*changed\_data = pd.DataFrame(columns=title\_list)

再次打开原数据文件。

*# 打开文件*origin\_data = open(**"D:/input/data.txt"**, **"r"**, encoding=**'utf-8'**)

接下来我们根据原数据集样本，生成目的数据集样本。对每个样本，检测其是否含有每个商品，若有则此项为1，否则此项为0。

*# 读取一行数据*line = origin\_data.readline()  
*# 迭代所有行，直到把文件所有行读完*while line:  
 *# 去掉换行符* line = line.strip()  
 *# 存储这条记录中，各个商品的购买情况* customer = {}  
 *# 把一行数据根据“，”隔开为一个个商品名* product\_items = line.split(**','**)  
 *# 逐个查看是否购买商品* for item in title\_list:  
 if(product\_items.\_\_contains\_\_(item)) :  
 customer[item] = 1  
 else:  
 customer[item] = 0  
 *# 将这一行添加到数据集* changed\_data = changed\_data.append(customer, ignore\_index=True)  
 *# 读取下一行，继续迭代* line = origin\_data.readline()

因为数据集样本较多，转换操作需要一些时间。转换完成后，我们输出前几行数据进行查看。

*# 打印前五行数据*changed\_data.head(5)

可以看到，数据集已经被转化为我们所需要的标准形式。

我们将处理后的数据保存为一个excel表格，可以本地查看数据集或供以后使用。

*# 将处理后的数据集保存*changed\_data.to\_excel(**"D:/output/changed\_data.xlsx"**, index=False)

大家可以本地打开文件查看。

# 4关联规则挖掘

准备好了数据集，我们终于可以进入到关联规则挖掘步骤了。

这一步，我们是调用mlxtend工具包实现的，故而代码非常简单。

Apriori函数有多个参数：

* min\_support：给定的最小支持度。
* use\_colnames：默认False，则返回的物品组合用编号显示，为True的话直接显示物品名称。
* max\_len：最大物品组合数，默认是None，不做限制。如果只需要计算两个物品组合的话，便将这个值设置为2。

frequent\_itemsets = apriori(changed\_data, min\_support=0.05, use\_colnames=True)

随后输出运算得到的频繁项集。

*# 输出频繁项集*frequent\_itemsets

大家可以看到符合设置条件的频繁项集和它对应的支持度。

我们可以用频繁项集生成关联规则，同样使用mlxtend提供的方法实现。

association\_rules方法同样可以设置多个参数：

* frequent\_itemsets：这个不用说，就是 Apriori 计算后的频繁项集。
* metric：可选值['support','confidence','lift','leverage','conviction']。
* 里面比较常用的就是置信度和支持度。这个参数和下面的min\_threshold参数配合使用。
* min\_threshold：参数类型是浮点型，根据 metric 不同可选值有不同的范围，
* metric = 'support' => 取值范围 [0,1]
* metric = 'confidence' => 取值范围 [0,1]
* metric = 'lift' => 取值范围 [0, inf]
* support\_only：默认是 False。仅计算有支持度的项集，若缺失支持度则用 NaNs 填充。

rules = association\_rules(frequent\_itemsets, metric=**"lift"**, min\_threshold=1)

输出得到的关联规则：

rules

输出结果中，各个列对应的含义如下：

* antecedents前项集
* consequents后项集
* support支持度
* confidence置信度
* lift提升度

本案例难点在于对初始小票数据的处理，请大家尽力理解数据处理代码，争取能做到自己完成数据转换。

同时，大家也可以调整参数，获取其他标准下的关联规则。