# 实验三：关联挖掘之Apriori算法

## 一、目的与要求

Apriori算法是一种挖掘关联规则的频繁项集算法，其核心思想是通过候选集生成和情节的向下封闭检测两个阶段来挖掘频繁项集。而且算法已经被广泛的应用到商业、网络安全等各个领域。本实验将简单地利用Python实现Apriori算法，帮助同学们学习。

## 二、操作环境

实验采用头歌实践教学平台（[www.educoder.net](http://www.educoder.net)）进行自动测评，课程团队通过平台提供的公用云和公共镜像的方式提前在平台配置好软硬件环境及实验任务，操作系统为Ubuntu 16.04.3 LTS，并配置Python 3编程环境。

## 三、实验内容

### 第1关：判断项集是否满足先验性质

#### 任务描述

本关任务：编写完成判断项集是否满足先验性质的函数。

#### 相关知识

为了完成本关任务，你需要掌握：1.什么是项、项集；2.先验性质。

##### 项与项集

设itemset={item\_1, item\_2, …, item\_m}是所有项的集合，其中，item\_k(k=1,2,…,m)成为项。项的集合称为项集（itemset），包含k个项的项集称为k项集(k-itemset)。

##### 事务与事务集

一个事务T是一个项集，它是itemset的一个子集，每个事务均与一个唯一标识符Tid相联系。不同的事务一起组成了事务集D，它构成了关联规则发现的事务数据库。

##### 先验性质

任何非频繁的(k-1)项集都不是频繁k项集的子集。如果一个候选k项集Ck的(k-1)项子集不在Lk-1中，则该候选也不可能是频繁的，从而可以从Ck中删除，获得剪枝后的Ck。

#### 编程要求

根据提示，在右侧编辑器补充代码，完成判断项集是否满足先验性质的函数。

##### 提示：

由于Apriori算法假定项集中的项是按字典序排序的，而集合本身是无序的，所以我们在必要时需要进行set和list的转换。

#### 测试说明

测试程序会使用测试集对你编写的代码进行测试。

### 第2关 挖掘频繁项集

#### 任务描述

本关任务：通过Lk-1自身的连接操作；创建集合Ck：包括所有频繁出现的k个候选项集的集合；统计每个项集支持数计数，创建集合Lk:满足最小支持数的k个频繁项集的集合。

#### 相关知识

##### 连接步骤：频繁(k-1)项集Lk-1的自身连接产生候选k项集Ck

Apriori算法假定项集中的项按照字典序排序。如果Lk-1中某两个的元素（项集）itemset1和itemset2的前(k-2)个项是相同的，则称itemset1和itemset2是可连接的。所以itemset1与itemset2连接产生的结果项集是{itemset1[1], itemset1[2], …, itemset1[k-1], itemset2[k-1]}。

##### 剪枝策略

由于存在先验性质：任何非频繁的(k-1)项集都不是频繁k项集的子集。因此，如果一个候选k项集Ck的(k-1)项子集不在Lk-1中，则该候选也不可能是频繁的，从而可以从Ck中删除，获得压缩后的Ck。下文代码中的is\_apriori函数用于判断是否满足先验性质，create\_Ck函数中包含剪枝步骤，即若不满足先验性质，剪枝。

##### 删除策略

基于压缩后的Ck，扫描所有事务，对Ck中的每个项进行计数，然后删除不满足最小支持度的项，从而获得频繁k项集。删除策略包含在下文代码中的generate\_Lk\_by\_Ck函数中。

#### 具体步骤

1.每个项都是候选1项集的集合C1的成员。算法扫描所有的事务，获得每个项，生成C1（见下文代码中的create\_C1函数）。然后对每个项进行计数。然后根据最小支持度从C1中删除不满足的项，从而获得频繁1项集L1。

2.对L1的自身连接生成的集合执行剪枝策略产生候选2项集的集合C2，然后，扫描所有事务，对C2中每个项进行计数。同样的，根据最小支持度从C2中删除不满足的项，从而获得频繁2项集L2。

3.对L2的自身连接生成的集合执行剪枝策略产生候选3项集的集合C3，然后，扫描所有事务，对C3每个项进行计数。同样的，根据最小支持度从C3中删除不满足的项，从而获得频繁3项集L3。

4.以此类推，对Lk-1的自身连接生成的集合执行剪枝策略产生候选k项集Ck，然后，扫描所有事务，对Ck中的每个项进行计数。然后根据最小支持度从Ck中删除不满足的项，从而获得频繁k项集。

##### 创建Ck

len\_Lksub1用于记录Lksub1的长度，通过嵌套for循环，产生两个L1,L2两个列表，若两个列表前k-1项相同且符合函数is\_apriori，将该子项集添加到Ck。

##### 创建Lk

通过嵌套遍历data\_set与Ck，记录Ck中的项集item在data\_set出现的次数，item\_count用来记录，将候选项的支持度计数与min\_support比较，大于min\_support的加入Lk。

#### 编程要求

根据提示，在右侧编辑器补充代码，创建集合Ck与集合Lk。

#### 提示

由于要使用字典（support\_data）记录项集的支持度，需要用项集作为key，而可变集合无法作为字典的key，因此在合适时机应将项集转为固定集合frozenset。

#### 测试说明

测试程序会使用测试集对你编写的代码进行测试

### 第3关：由LK获得最后的频繁项集和关联规则

#### 任务描述

本关任务：实现得到最后的频繁项集，得到关联规则。

#### 相关知识

某超市的交易数据库:

交易号TID 顾客购买的商品  
T1 bread, cream, milk, tea  
T6 bread, tea  
T2 bread, cream, milk  
T7 beer, milk, tea  
T3 cake, milk  
T8 bread, tea  
T4 milk, tea  
T9 bread, cream, milk, tea  
T5 bread, cake, milk  
T10 bread, milk, tea

##### 项目与项集

定义一：设I={i1,i2,…,im}，是m个不同的项目的集合，每个ik称为一个项目。项目的集合I称为项集。其元素的个数称为项集的长度，长度为k的项集称为k-项集。引例中每个商品就是一个项目，项集为I={bread, beer, cake,cream, milk, tea}，I的长度为6。

##### 事务与事务数据库

定义二：每笔交易T是项集I的一个子集。对应每一个交易有一个唯一标识交易号，记作TID。交易全体构成了交易数据库D，|D|等于D中交易的个数。引例中包含10笔交易，因此|D|=10。

##### 支持度

定义三：对于项集X，设定count(X⊆T)为交易集D中包含X的交易的数量，则项集X的支持度为：support(X)=count(X⊆T)/|D|，引例中X={bread, milk}出现在T1，T2，T5，T9和T10中，所以支持度为0.5。

##### 最小支持度

定义四：最小支持度是项集的最小支持阀值，记为SUPmin，代表了用户关心的关联规则的最低重要性。支持度不小于SUPmin的项集称为频繁集，长度为k的频繁集称为k-频繁集。如果设定SUPmin为0.3，引例中{bread, milk}的支持度是0.5，所以是2-频繁集。

##### 关联规则

定义五：关联规则是一个蕴含式：R：X⇒Y，其中X⊂I，Y⊂I，并且X∩Y=⌀。表示项集X在某一交易中出现，则导致Y以某一概率也会出现。用户关心的关联规则，可以用两个标准来衡量：支持度和可信度。

##### 关联规则的支持度

定义六：关联规则R的支持度是交易集同时包含X和Y的交易数与|D|之比。即：  
support(X⇒Y)=count(X⋃Y)/|D|，支持度反映了X、Y同时出现的概率。关联规则的支持度等于频繁集的支持度。

##### 关联规则的可信度

定义七：对于关联规则R，可信度是指包含X和Y的交易数与包含X的交易数之比。即：confidence(X⇒Y)=support(X⇒Y)/support(X)，可信度反映了如果交易中包含X，则交易包含Y的概率。一般来说，只有支持度和可信度较高的关联规则才是用户感兴趣的。

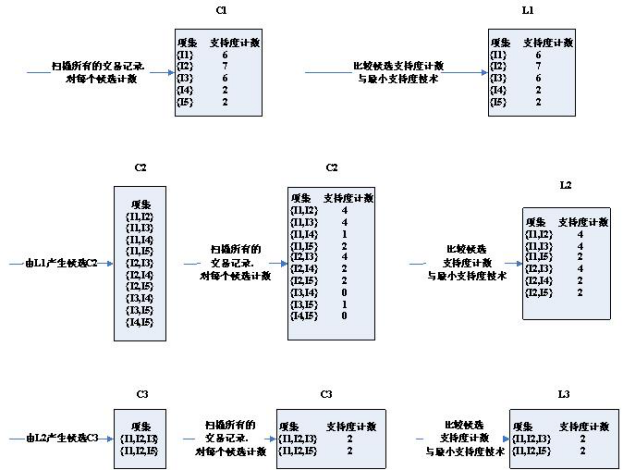
##### 强关联规则

定义八：设定关联规则的最小支持度和最小可信度为SUPmin和CONFmin。规则R的支持度和可信度均不小于SUPmin和CONFmin，则称为强关联规则。关联规则挖掘的目的就是找出强关联规则，从而指导商家的决策。

#### 测试数据

交易ID 商品ID列表  
T100 I1，I2，I5  
T200 I2，I4  
T300 I2，I3  
T400 I1，I2，I4  
T500 I1，I3  
T600 I2，I3  
T700 I1，I3  
T800 I1，I2，I3，I5  
T900 I1，I2，I3

#### 算法步骤图



可以看到，第三轮的候选集发生了明显的缩小，由于选取候选集的两个条件：  
1.两个K项集能够连接的两个条件是，它们有K-1项是相同的。所以，（I2，I4）和（I3，I5）这种是不能够进行连接的。缩小了候选集；

2.如果一个项集是频繁集，那么它不存在不是子集的频繁集。比如（I1，I2）和（I1，I4）得到（I1，I2，I4），而（I1，I2，I4）存在子集（I1，I4）不是频繁集。缩小了候选集。

第三轮得到的2个候选集，正好支持度等于最小支持度。所以，都算入频繁集。  
这时再看第四轮的候选集与频繁集结果为空，因为通过第三轮得到的频繁集自连接得到{I1，I2，I3，I5}，它拥有子集{I2,I3,I5}，而{I2,I3,I5}不是频繁集，不满足：频繁集的子集也是频繁集这一条件，所以被剪枝剪掉了。所以整个算法终止，取最后一次计算得到的频繁集作为最终的频繁集结果：  
也就是：['I1,I2,I3', 'I1,I2,I5']。

#### 编程要求

根据提示，在右侧编辑器补充代码，实现由LK获得最后的频繁项集和关联规则。

#### 测试说明

测试程序会使用测试集对你编写的代码进行测试。