# 1 Apriori算法

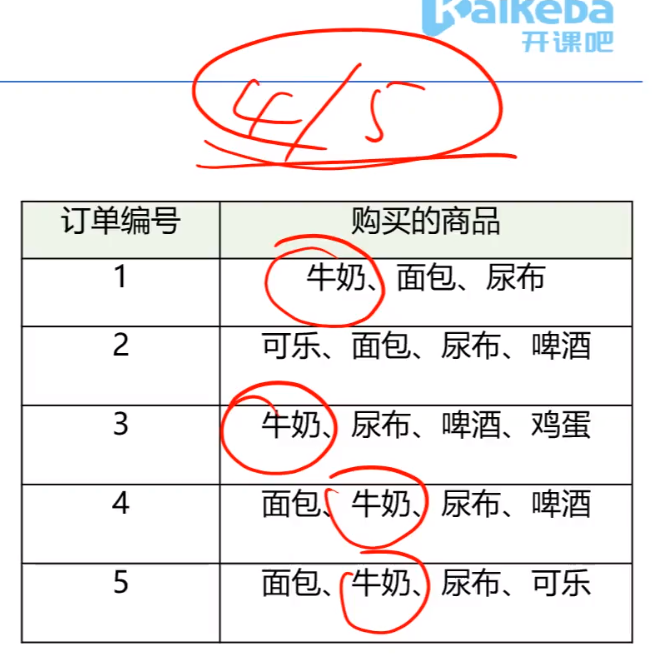
## 1.1关联分析的基本概念

关联规则：Association Rules或者是Basket Analysis（或则称之为购物篮分析）

例子：啤酒和尿布、高中女生怀孕的案例

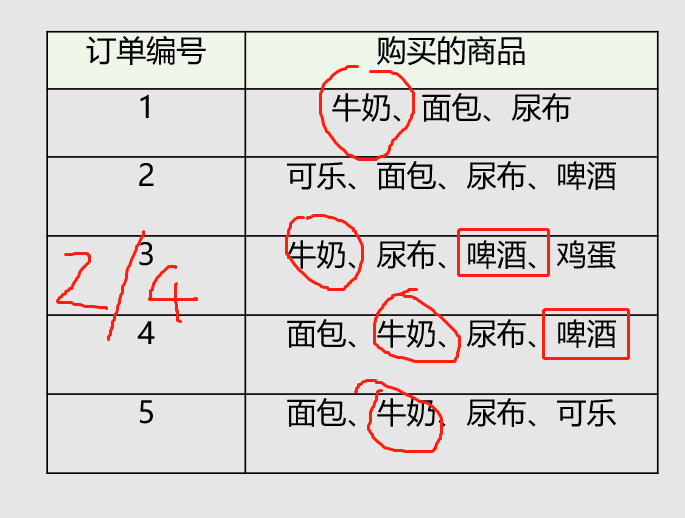
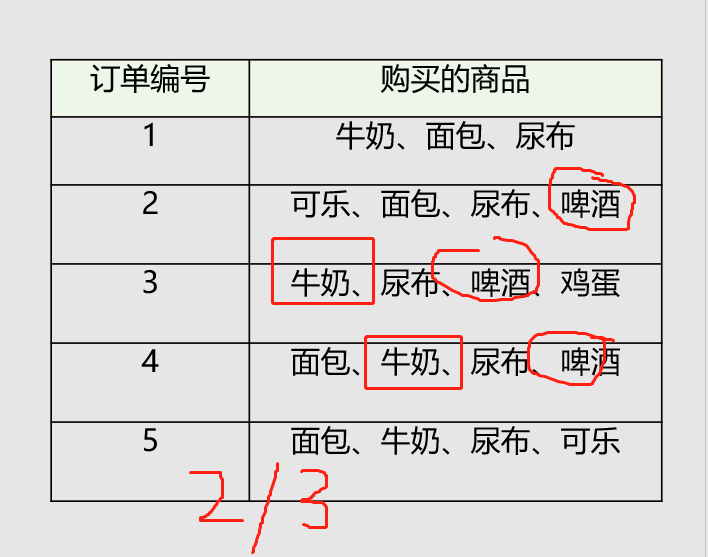
支持度：是个百分比，指的是**某个商品组合出现的次数与总次数之间**的比例。

P.S:支持度越高，代表这个组合出现的频率越大



“牛奶”的支持度=4/5=0.8

“牛奶+面包”的支持度=3/5=0.6。  
置信度：是个条件概念，指的是你购买了商品A，会有多大的概率购买商品B

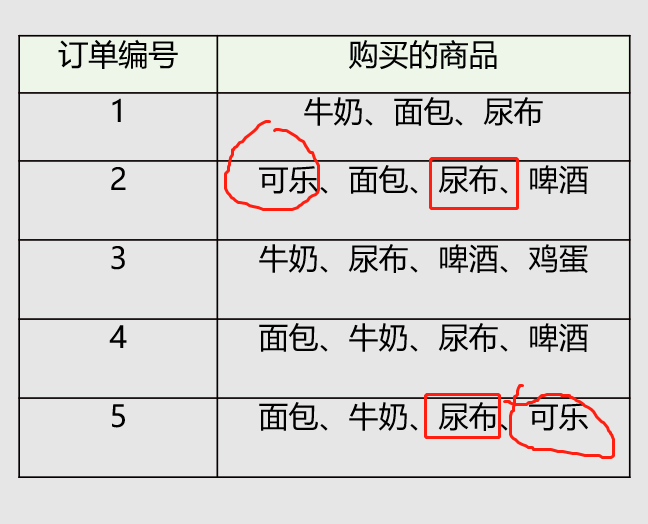


置信度（牛奶→啤酒）=2/4=0.5

置信度（啤酒→牛奶）=2/3=0.67

提升度：商品A的出现，对商品B的出现概率提升的程度

Q: 如果我们单纯看置信度(可乐→尿布)=1，也就是说可乐出现的时候，用户都会购买尿布，那么当用户购买可乐的时候，就需要推荐尿布么？



A: 答案显然是不会去这么做的，因为没有提升，所以不会去这么做，这个时候就引入了 lift 提升度这个指标

计算公式：提升度（A🡪B） = 置信度（A🡪B）/支持度（B）

提升度的三种可能：

* 提升度（A🡪B）> 1:代表有提升
* 提升度（A🡪B）= 1:代表没有提升，也没有下降
* 提升度（A🡪B）< 1:代表有下降

杠杆率：关联规则支持度 – 前件支持度 X 后件支持度

该值越大，表面 X和 Y的关联度越强

确信度：（1 – 后件支持度）/ (1 – 关联规则置信度)

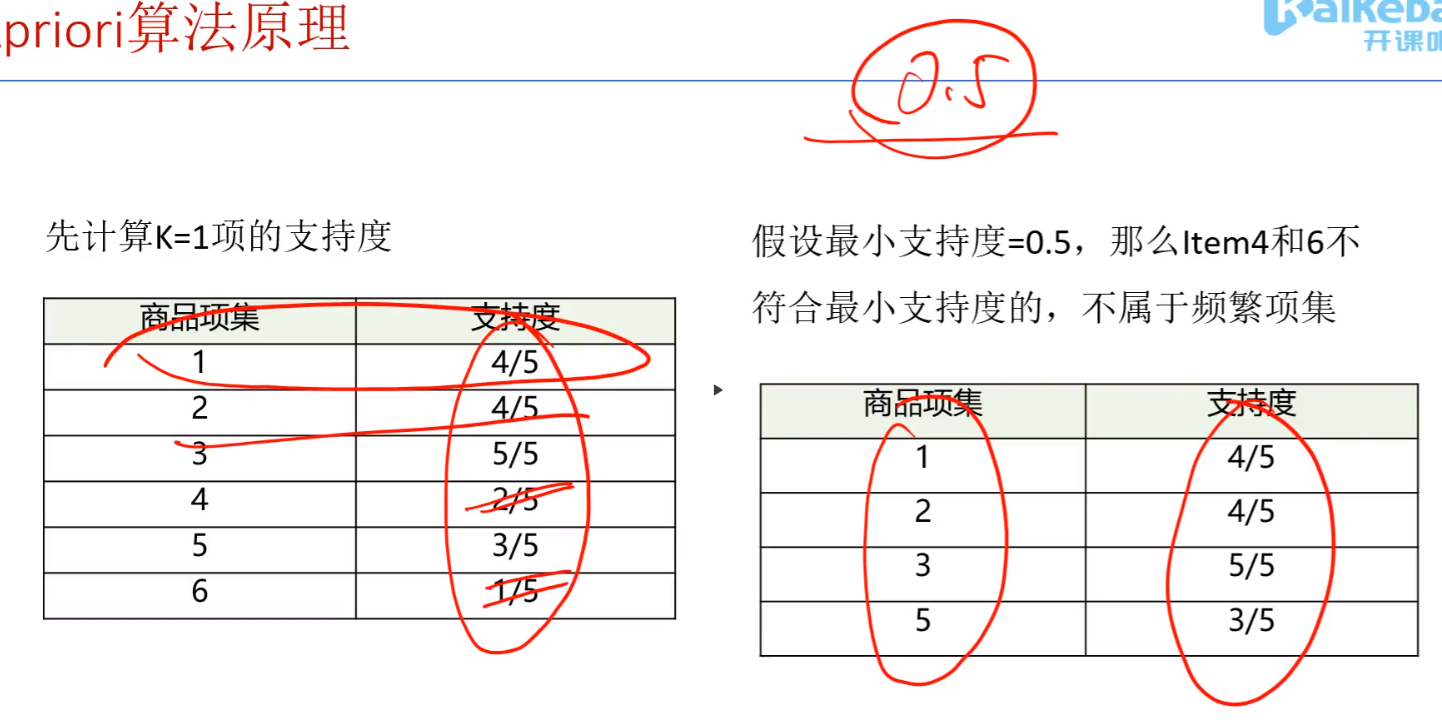
我们把上面案例中的商品用ID来代表，牛奶、面包、尿布、可乐、啤酒、鸡蛋的商品ID分别设置为1~ 6

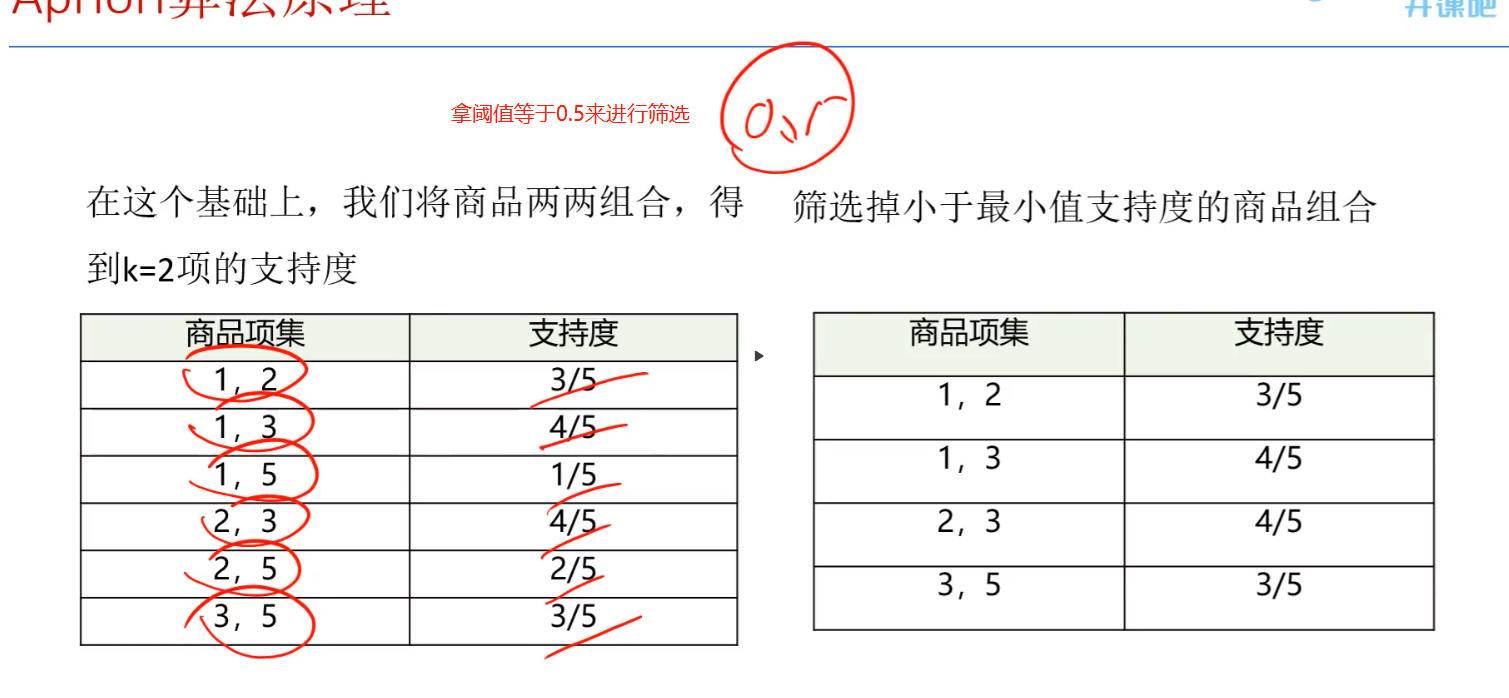


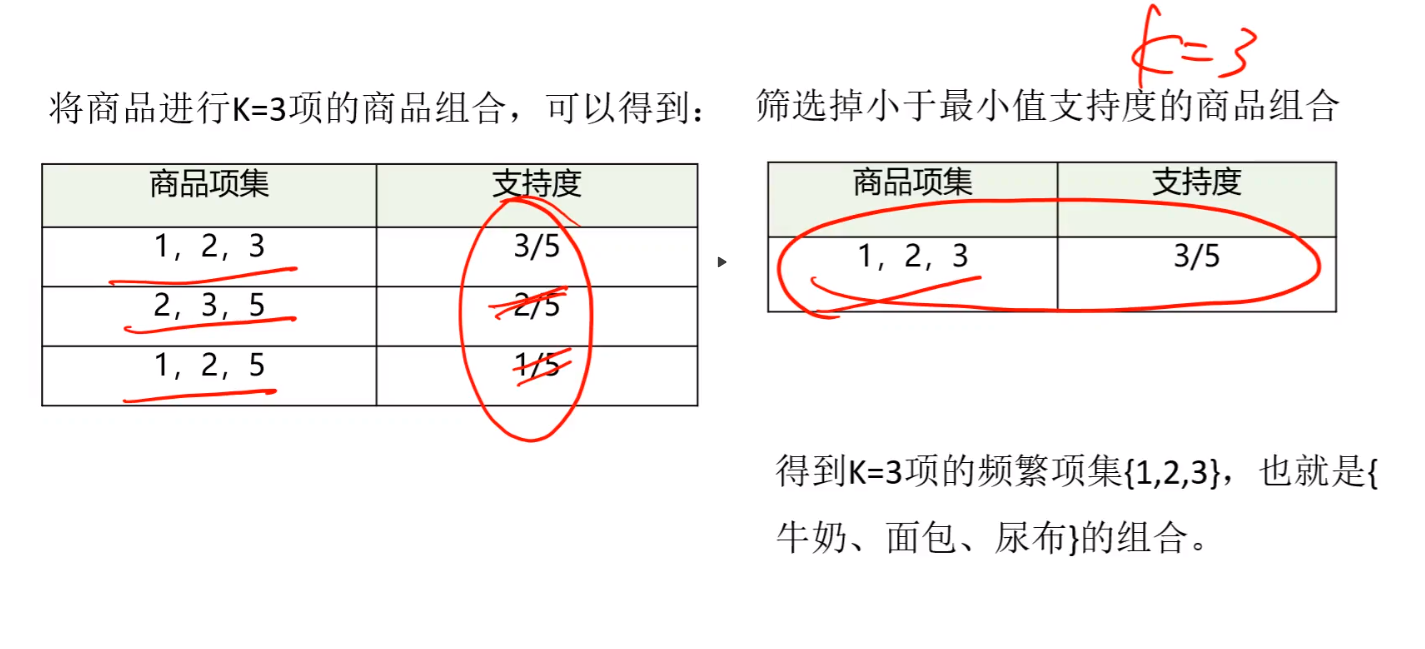
**Apriori算法就是查找频繁项集（frequent itemset）的过程**

频繁项集**：**支持度大于等于最小支持度（Min Support）阈值的项集

非频繁项集：支持度小于最小支持度的项集







这里面少个1、3、5的组合

## 1.2算法流程

Step1，K=1，计算K项集的支持度；

Step2，筛选掉小于最小支持度的项集；

Step3，如果项集为空，则对应K-1项集的结果为最终结果。

否则K=K+1，重复1-3步。

## 1.3工具包选用

* from efficient\_apriori import apriori

特点：运算速度快，但是结果较为简单

from mlxtend.frequent\_patterns import apriori

from mlxtend.frequent\_patterns import association\_rules

特点：运算速度相对较慢，但是结果·很详细

## 1.4关联分析实战

万物皆Transaction

* 超市购物小票的关联关系

每笔订单的商品（TransactionID => Item）

* 电影分类中的关联关系

每部电影的分类（MovieID => 分类）

* 电影演员中的关联关系

每部电影的演员列表（MovieID => 演员）

Antecedent 前因

Consequent 后果

超参数的设置经验：

面对的业务是人参与较多的业务，提升度可以设置的大些，如果面对的是全是物体的业务，提升度可以设置的相对小些

## 1.5关联规则与协同过滤的区别

* 关联规则是基于transaction,而协同过滤基于用户偏好（评分）
* 商品组合使用的是购物篮分析，也就是Apriori算法，协同过滤计算的是相似度
* 关联规则没有利用“用户偏好”，而是基于购物订单进行的频繁项集挖掘

推荐使用场景：

**当下的需求**：

推荐的基础是且只是当前一次的购买/点击

**长期偏好**：

基于用户历史的行为进行分析，建立一定时间内的偏好排序

两种推荐算法的思考维度不同，很多时候，我们需要把多种推荐方法的结果综合起来做一个混合的推荐

## 1.6关联规则的视角

不需要考虑用户一定时期内的偏好，而是基于Transaction

只要能将数据转换成Transaction，就可以做购物篮分析：

Step1、把数据整理成id=>item形式，转换成transaction

Step2、设定关联规则的参数（support、confident）挖掘关联规则

Step3、按某个指标（lift、support等）对以关联规则排序

## 1.7最小支持度、最小置信度确定方法

**最小支持度，最小置信度是实验出来的**

最小支持度：

不同的数据集，**最小支持度值**差别较大。可能是0.01到0.5之间

可以从高到低输出前20个项集的支持度作为参考

**最小置信度**：可能是0.5到1之间

**提升度**：表示使用关联规则可以提升的倍数，是置信度与期望置信度的比值

提升度至少要大于1

# 2.FPGrowth算法

## 2.1引入

Apriori在计算的过程中存在的不足：

可能产生大量的候选集。因为采用排列组合的方式，把可能的项集都组合出来了

每次计算都需要重新扫描数据集，计算每个项集的支持度

浪费了计算空间和时间,或者称之为时间和空间复杂度比较高

在Apriori算法基础上提出了FP-Growth算法：

* 创建了一棵FP树来存储频繁项集。在创建前对不满足最小支持度的项进行删除，减少了存储空间。
* 整个生成过程只遍历数据集2次，大大减少了计算量（这里的两次是一次从上到下，一次从下到上）

理解：Apriori存在的不足，有更快的存储和搜索方式进行频繁项集的挖掘

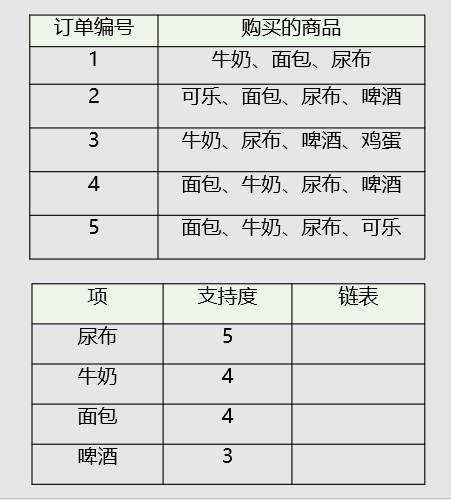
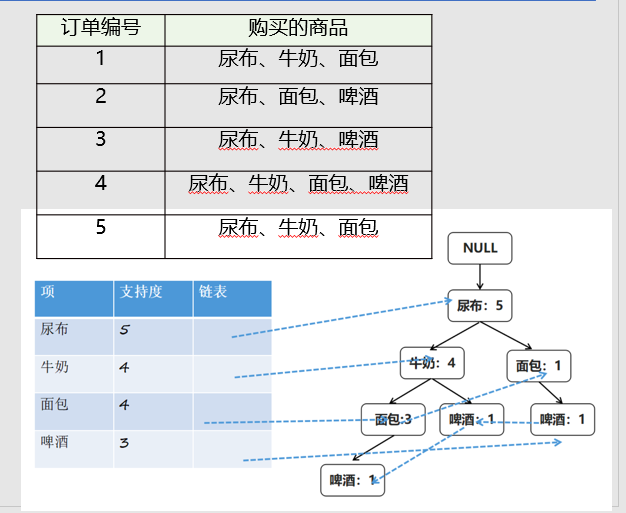
## 2.2算法流程

创建项头表（item header table）（**建树的时候是从上到下**）

* 作用是为FP构建及频繁项集挖掘提供索引。
* Step1、流程是先扫描一遍数据集，对于满足最小支持度的单个项（K=1项集）按照支持度从高到低进行排序，这个过程中删除了不满足最小支持度的项。

项头表包括了项目、支持度，以及该项在FP树中的链表。初始的时候链表为空

* Step2、对于每一条购买记录，按照项头表的顺序进行排序，并进行过滤。构造FP树，根节点记为NULL节点
* Step3、整个流程是需要再次扫描数据集，把Step2得到的记录逐条插入到FP树中。节点如果存在就将计数count+1，如果不存在就进行创建。同时在创建的过程中，需要更新项头表的链表。

按照左下图统计的支持度，重新排列左上的购买商品顺序，如右上图所示

* Step4、通过FP树挖掘频繁项集（挖掘频繁项集是从下往上找的）

现在已经得到了一个存储频繁项集的FP树，以及一个项头表。可以通过项头表来挖掘出每个频繁项集。

挖掘从项头表最后一项“啤酒”开始。

从FP树种找到所有“啤酒”节点，向上遍历祖先节点，得到3条路径。对于每条路径上的节点，其count都设置为“啤酒”的count

具体的操作会用到一个概念，叫“条件模式基”

因为每项最后一个都是“啤酒”，因此我们把“啤酒”去掉，得到条件模式基，此时后缀模式是（啤酒）



假设{啤酒}的条件频繁集为{S1,S2,S3}，则{啤酒}的频繁集为{S1+{啤酒},S2+{啤酒},S3+{啤酒}}，此时的条件频繁项集为{{}, {尿布}}，所以啤酒的频繁项集为{啤酒}，{尿布，啤酒}

继续找项头表倒数第2项面包，求得“面包”的条件模式基

根据条件模式基，可以求得面包的频繁项集：{面包}，{尿布，面包}，{牛奶，面包}，{尿布，牛奶，面包}



继续找项头表倒数第3项面包，求得“牛奶”的条件模式基

根据条件模式基，可以求得面包的频繁项集：{牛奶}，{尿布，牛奶}



继续找项头表倒数第4项面包，求得“尿布”的条件模式基

根据条件模式基，可以求得尿布的频繁项集：{尿布}

所以全部的频繁项集为：

{啤酒}，{尿布，啤酒}

{面包}，{尿布，面包}，{牛奶，面包}，{尿布，牛奶，面包}

{牛奶}，{尿布，牛奶}

{尿布}

