实验手册_关联规则挖掘

一.　关联规则挖掘算法原理

本节将详细介绍关联规则挖掘中两个常用的算法：Apriori 算法和 FP-growth 算法。这两种算法在挖掘频繁项集和关联规则时起着至关重要的作用，它们分别采用了不同的思想和技巧，以应对数据挖掘中的挑战和问题。

1　Apriori 算法

Apriori 算法是关联规则挖掘领域中最经典的算法之一，它于 1994 年由 Rakesh Agrawal 和 Ramakrishnan Srikant 提出。Apriori 算法是一种经典的关联规则挖掘算法，其名字来源于先验性原理，即如果一个项集是频繁的，那么它的所有子集也必然是频繁的。Apriori 算法通过候选项集的逐层搜索来发现频繁项集，然后基于频繁项集构建关联规则。

Apriori 算法是关联规则挖掘中最经典的算法之一，其核心思想基于 Apriori 性质和连接性质，通过迭代的方式逐步发现频繁项集和关联规则。Apriori 算法的原理基于两个重要的性质：

1.先验性原理（Apriori Principle）：如果一个项集是频繁的，那么它的所有子集也必然是频繁的。

2.连接性质（Join Operation）：如果两个项集的前 k−1 项相同，那么可以将它们合并为一个大小为 k 的候选项集。

基于以上两个性质，Apriori 算法通过迭代的方式生成候选项集并利用支持度剪枝策略来发现频繁项集。

Apriori 算法的流程

1.生成候选项集：首先生成所有单个项的候选项集（一项集），然后利用先验性原理和连接性质逐层生成大小逐渐增大的候选项集。

2.计算支持度：对于生成的候选项集，扫描数据集计算其支持度（即在数据集中出现的频率）。

3.筛选频繁项集：根据设定的最小支持度阈值，筛选出支持度不低于阈值的频繁项集。

4.生成关联规则：基于频繁项集，利用置信度公式生成关联规则，并根据设定的最小置信度阈值筛选出符合要求的关联规则。

Apriori 算法案例

假设我们有以下的交易数据集（设 min_sup=60%，min_conf=80%。）：

表 1 交易数据集

Transaction ID	Item Purchased
T01	F, H, A, D, B, C
T02	D, A, E, I
T03	C, D, B, E
T04	B, A, C, I, D
T05	A, G, H, C, E

现在我们要使用 Apriori 算法来找出频繁项集。

步骤 1：生成候选项集（C1）

首先，我们列出所有的单个物品（项）作为候选项集 C1：

L1 = {A，B，C，D，E，F，G，H，I}

步骤 2：扫描数据集，计算支持度（Support）

接下来，我们需要扫描数据集，计算每个候选项的支持度。

支持度（Support）是指包含某个项集的交易数量占总交易数量的比例。

计算每个项的支持度，假设我们设定最小支持度为 60%（5 次中的至少 3 次）。

Support（A）= 4/5 = 0.8；Support（B）= 3/5 = 0.6；Support（C）= 4/5 = 0.8；

Support（D）= 4/5 = 0.8；Support（E）= 3/5 = 0.6；Support（F）= 1/5 = 0.2；

Support（G）= 1/5 = 0.2；Support（H）= 2/5 = 0.4；Support（I）= 2/5 = 0.4。

步骤 3：筛选频繁项集（L1）

根据最小支持度，筛选出频繁项集 L1，即支持度不低于 50% 的项集：

L1 = {A，D，B，C，E}

步骤 4：生成候选项集（C2）

接下来，我们根据频繁项集 L1 生成候选项集 C2。

C2 是通过连接 L1 中的频繁项集来生成的。在这种情况下，我们得到以下候选项集：

C2 = {AD，AB，AC，AE，DB，DC，DE，BC，BE，CE}

步骤 5：计算支持度（Support）

然后，我们需要再次扫描数据集，计算每个候选项集的支持度。

Support（AD）= 3 / 5 = 0.6；Support（AB）= 1 / 5 = 0.2；

Support（AC）= 3 / 5 = 0.6；Support（AE）= 2 / 5 = 0.4；

Support（DB）= 3 / 5 = 0.6；Support（DC）= 3 / 5 = 0.6；

Support（DE）= 2 / 5 = 0.4；Support（BC）= 3 / 5 = 0.6；

Support（BE）= 1 / 5 = 0.2；Support（CE）= 2 / 5 = 0.4。

步骤 6：筛选频繁项集（L2）

根据最小支持度，筛选出频繁项集 L2，即支持度不低于 50% 的项集：

L2 = {AD，AC，DB，DC，BC}

步骤 7：生成候选项集（C3）和计算支持度（Support）

在接下来的步骤中，我们可以继续生成候选项集 C3，并计算其支持度。按照相似的步骤，我们将得到频繁项集 L3。

C3 = {ACD，BCD}

L3 = {BCD}

这就是 Apriori 算法的基本步骤。通过迭代地生成候选项集和筛选频繁项集，Apriori 算法能够找到数据集中的频繁项集，从而帮助我们发现数据中的关联规则。例如在该例中所有基于频繁 3 项集所得到的强关联规则有以下几项：

B∧C⇒D [60%，100%]

C∧D⇒B [60%，100%]

B∧D⇒C [60%，100%]

B⇒C∧D [60%，100%]

Apriori 算法适用场景

1.小规模数据集：Apriori 算法在处理小规模数据集时比较适用，因为它的实现比

较简单，易于理解和部署。

2.初始探索和理解：在初步探索数据集并理解数据之间关联关系时，Apriori 算法可以作为一个简单而有效的工具。

3.初学者学习和教学：由于其简单直观的特点，Apriori 算法常被用于教学和学术研究，帮助初学者理解关联规则挖掘的基本原理和方法。

Apriori 算法总结

Apriori 算法的优点之一是其简单易懂，易于理解和实现。同时，它能够有效地发现频繁项集和关联规则，为数据挖掘任务提供了重要的支持。然而，Apriori 算法也存在一些缺点。首先，算法效率较低，特别是在处理大规模数据集时，生成候选项集的开销较大。其次，需要多次扫描数据集，消耗大量的计算资源。

2 FP-growth 算法

FP-growth(频繁模式生长)算法是一种基于前缀树的频繁项集挖掘算法，由 Jiawei Han 和 Jian Pei 在 2000 年提出。FP-growth 算法是一种高效的关联规则挖掘算法，它通过构建 FP 树（频繁模式树）来实现对频繁项集的挖掘。相比于 Apriori 算法，FP-growth 算法不需要生成候选项集，因此在大规模数据集上具有更高的效率。

FP-growth 算法原理

FP-growth 算法的核心思想是利用 FP-tree 来存储数据集中的频繁项集信息，并通过递归方式挖掘频繁项集。FP-growth 算法具有以下特点：

1.构建 FP 树：遍历数据集两次，第一次统计每个项的频率并排序，第二次构建 FP 树，通过链接相似项来表示频繁项集的树形结构。

2.挖掘频繁项集：通过遍历 FP 树，利用递归方法挖掘频繁项集。

3.FP-growth 算法的优点在于不需要生成候选项集，避免了候选项集的指数级增长，因此在大规模数据集上具有较高的效率。

FP-growth 算法的优点之一是其高效的挖掘能力，特别适用于处理大规模数据集。通过压缩数据和利用树结构，减少了多次扫描数据集的开销。然而，FP-growth 算法也存在一些缺点。首先，实现相对复杂，需要构建 FP-tree 和条件模式基。其次，需

要一定的内存空间来存储 FP-tree。尽管 FP-growth 算法在实现上相对复杂，但其高效的挖掘能力使其成为处理大规模数据集时的首选算法之一。在实际应用中，根据具体的数据特点和需求，选择合适的关联规则挖掘算法是至关重要的。

FP-growth 算法案例

同 Apriori 算例一样，假设我们有以下的交易数据集（设 min_sup=60%，min_conf=80%）：

表 2 交易数据集

Transaction ID	Item Purchased
T01	F, H, A, D, B, C
T02	D, A, E, I
T03	C, D, B, E
T04	B, A, C, I, D
T05	A, G, H, C, E

我们将使用 FP-growth 算法来找出频繁项集。

步骤 1：构建 FP 树

首先，我们需要构建一个 FP 树（频繁模式树）。FP 树由项头表和项频繁项链表组成。

首先，扫描数据集，统计每个项的频率，并按照频率降序排序：

F-list = {{A:4},{C:4},{D:4},{B:3},{E:3}}

接下来，根据排序后的项集，构建 FP 树。从根节点开始，根据每个事务中的项依次添加节点。重复出现的项会增加节点的计数，而新的项则会创建新的分支。

构建的 FP 树如下所示：

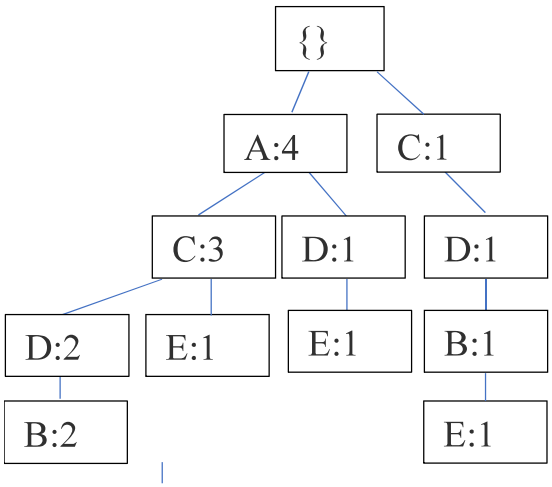

图 1 FP 树

步骤 2：挖掘频繁项集

接下来，我们从 FP 树中挖掘频繁项集。

对于每个频繁项（项头表中的每个项），从其相应的条件基（conditional pattern base）构建条件 FP 树。

对于条件 FP 树，递归地构建它的条件模式基（conditional pattern base）并创建对应的条件 FP 树。

继续这个过程，直到不能再构建出更多的条件 FP 树。

接下来，我们按照从底部到顶部的顺序，将每个项添加到频繁项集中，同时考虑条件模式基中的频繁项集。通过这个过程，我们可以获得所有的频繁项集。

在我们的示例中，假设我们的最小支持度阈值为 60%，即至少出现在 3 个交易中。

通过 FP-growth 算法，可以找出以下条件模式基以及频繁项集：

CPB(E) = {{AC:1}, {AD:1}, {CDB:1}}

CFBT(E)={ ∅}

以{E}为前缀得到的频繁项集为{∅}

CPB(B)={{ACD:2},{CD:1}}

CFBT(B)={CD:3}

以{B}为前缀得到的频繁项集为{{CB:3},{DB:3},{CDB:3}}

CPB(D)={{AC:2},{A:1},{C:1}}

CFBT(D)={{A:3},{C:3}}

以{D}为前缀得到的频繁项集为{{AD:3},{CD:3}}

CPB(C)={{A:3}}

CFBT(C)={A:3}

以{C}为前缀得到的频繁项集为{{AC:3}}

最终我们得到频繁项集{ A，C，D，B，E，CB，DB，CDB，AD，CD，AC}。这些频繁项集表示在数据集中频繁出现的项组合。通过 FP-growth 算法，我们可以高效地找到频繁项集，而不需要显式地生成所有可能的项集，从而提高了算法的效率。

FP-growth 算法适用场景

1.大规模数据集：FP-growth 算法在处理大规模数据集时表现更加优异，因为它通过构建 FP-tree 结构，减少了数据集的扫描次数和计算开销。

2.高效挖掘频繁项集：当数据集规模较大且频繁项集数量庞大时，FP-growth 算法的高效性能能够更好地应对挖掘任务。

3.实时数据分析：在需要快速分析和挖掘实时数据的场景下，FP-growth 算法可以提供更快速的计算和响应速度。

4.高性能要求：对于性能要求较高的应用场景，FP-growth 算法的高效性能使其成为首选算法之一。

算法比较与选择总结

Apriori 算法和 FP-growth 算法各有优缺点：

Apriori 算法：

优点：简单易懂，容易实现。

缺点：对大规模数据集的处理效率较低，需要多次扫描数据集。

FP-growth 算法：

优点：不需要生成候选项集，避免了候选项集的指数级增长，因此在大规模数据集上具有较高的效率。

缺点：构建 FP 树的过程可能会消耗较大的内存空间。

根据数据集的规模和具体需求，选择合适的算法进行关联规则挖掘，可以最大程度地提高挖掘效率和准确性。

二. 实验案例与实验任务

案例内容与步骤

关联规则挖掘最经典的使用场景即是市场篮子分析，本实验利用 AI Studio 内置数据（SamplesTemplates/Market Basket Analysis/Transactions）对实验步骤进行讲解。

1．数据理解

本数据除了行编码外，另有四个属性，分别为交易 ID（Invoice）；产品 ID（product 1）；订单数量（Orders）；销售额（Sales value）。

本实验的目的是通过确定经常一起购买的商品集并构建关联规则来得到商品销售建议，从而对产品之间的销售方案进行建模。

2．数据加载

我们直接从内置数据集中加载包含交易 ID、产品 ID、交易数据和销售额的交易数据。该数据能得到顾客购买了某种产品的次数。

我们的数据集非常干净，每个属性中都没有缺失的值，并且在范围和其他描述性统计信息中也没有明显的不一致数据。

Row No.	Invoice	product 1	Orders	Sales value
1	131506	Product 20	1	40
2	131506	Product 21	1	80
3	131507	Product 11	1	80
4	131508	Product 19	1	32
5	131509	Product 31	1	9
6	131510	Product 11	1	80
7	131510	Product 20	2	40
8	131510	Product 20	1	40
9	131519	Product 11	1	80
10	131541	Product 11	1	80

图 2 数据集

Name	Type	Missing	Statistics		Filter (4 / 4 attributes): Search for Attribute
Invoice	Polynominal	0	Least 647990 (1)	Most 646220 (7)	Values 646220 (7), 647096 (6), ...[491 more]
product 1	Polynominal	0	Least Product 30 (7)	Most Product 12 (90)	Values Product 12 (90), Product 11 (72), ...[21 more]
Orders	Integer	0	Min 1	Max 2	Average 1.006
Sales value	Integer	0	Min 5	Max 80	Average 41.223

图 3 数据统计信息

3．数据准备

对数据进行转换。利用 Aggregate 算子将交易数据转换成可被关联规则算法分析的聚合数据类型。

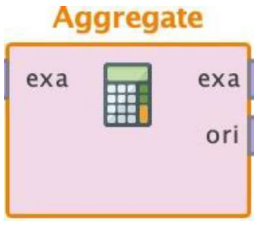

图 4 Aggregate 算子

图 5 Aggregate 算子参数设置

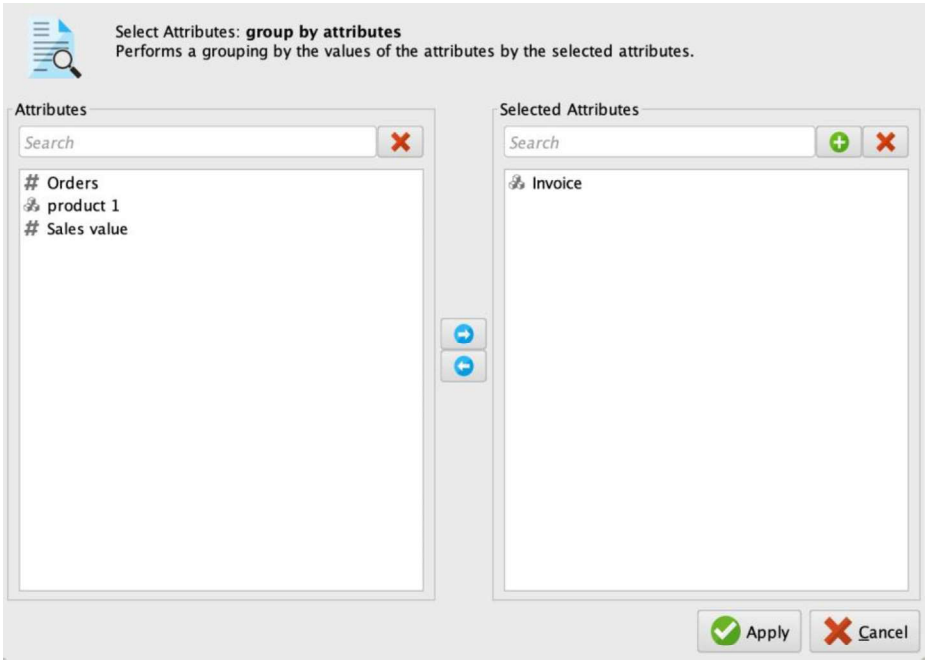

图 6 参数设置 2

将聚合后的数据中的交易 ID（Invoice）通过 Set Role 算子设置为 ID 属性

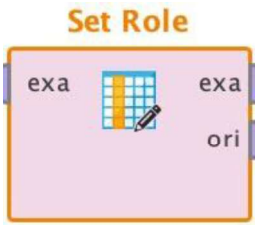

图 7 Set Role 算子

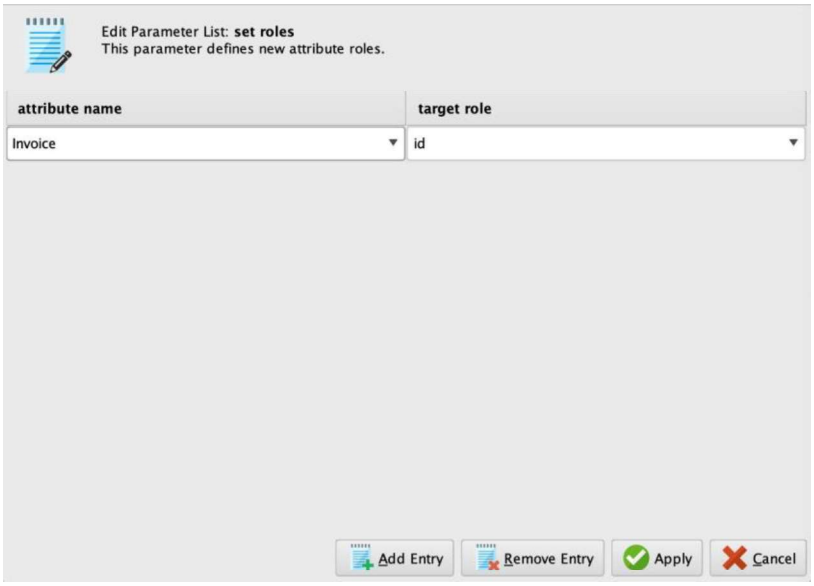

图 8 Set Role 算子参数设置

4. 模型构建

使用 FP-Growth，确定频繁的项目集。频繁的商品集表示该商品集中的物品（产品）经常一起购买，即在一定比例的交易中。该比率由项目集的支持率给出。

图 9 FP-Growth 算子

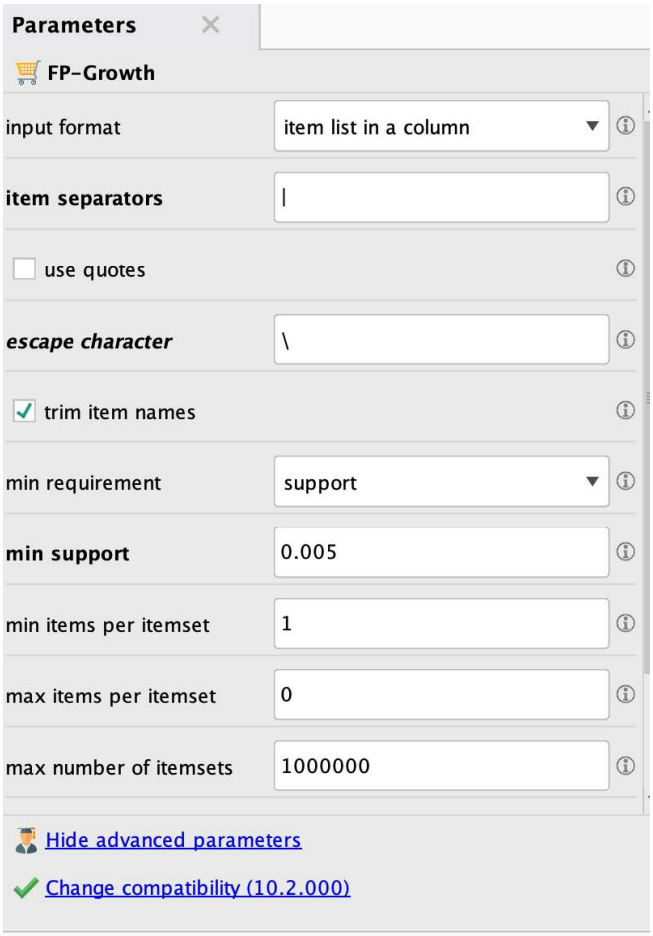

图 10 FP-Growth 算子参数设置

5．规则建立

使用 Create Association Rules 算子创建关联规则，这些规则可用于产品推荐，具体取决于这些规则的置信度。

图 11 Create Association Rules 算子

图 12 Create Association Rules 算子参数设置

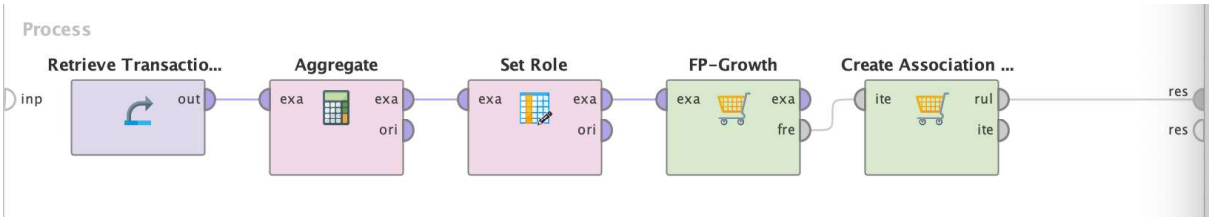

图 13 模型框架

6. 结果分析

Size	Support	Item 1	Item 2	Item 3
2	0.014	Product 12	product 1	
2	0.006	Product 12	Product 16	
2	0.006	Product 12	Product 29	
2	0.006	Product 12	Product 27	
2	0.010	Product 12	Product 31	
2	0.008	Product 12	Product 22	
2	0.014	Product 20	Product 18	
2	0.006	Product 20	Product 15	
2	0.008	Product 20	Product 26	
2	0.012	Product 20	Product 19	
2	0.008	Product 20	Product 29	
2	0.014	Product 20	Product 27	
2	0.006	product 1	Product 16	
3	0.006	Product 11	Product 12	Product 20
3	0.006	Product 11	Product 20	Product 19
3	0.006	Product 12	Product 20	Product 27

图 14 部分频繁项集

AssociationRules

```
Association Rules
[Product 18] --> [Product 12] (confidence: 0.103)
[Product 12] --> [product 1] (confidence: 0.104)
[Product 26] --> [Product 20] (confidence: 0.114)
[Product 20] --> [Product 19] (confidence: 0.118)
[Product 19] --> [Product 11] (confidence: 0.125)
[Product 19] --> [Product 12] (confidence: 0.125)
[product 1] --> [Product 16] (confidence: 0.125)
[Product 19] --> [Product 11, Product 20] (confidence: 0.125)
[Product 16] --> [Product 12] (confidence: 0.130)
[Product 16] --> [product 1] (confidence: 0.130)
[Product 20] --> [Product 18] (confidence: 0.137)
[Product 20] --> [Product 27] (confidence: 0.137)
[Product 21] --> [Product 12] (confidence: 0.138)
[Product 29] --> [Product 12] (confidence: 0.167)
[Product 11, Product 20] --> [Product 12] (confidence: 0.176)
[Product 11, Product 20] --> [Product 19] (confidence: 0.176)
[Product 18] --> [Product 20] (confidence: 0.179)
[Product 12] --> [Product 20] (confidence: 0.194)
[Product 27] --> [Product 12] (confidence: 0.214)
[Product 27] --> [Product 12, Product 20] (confidence: 0.214)
[Product 29] --> [Product 20] (confidence: 0.222)
[Product 12, Product 20] --> [Product 11] (confidence: 0.231)
[Product 12, Product 20] --> [Product 27] (confidence: 0.231)
[Product 11] --> [Product 20] (confidence: 0.250)
[Product 19] --> [Product 20] (confidence: 0.250)
[Product 20] --> [Product 12] (confidence: 0.255)
[product 1] --> [Product 12] (confidence: 0.292)
[Product 20] --> [Product 11] (confidence: 0.333)
[Product 12] --> [Product 15] (confidence: 0.343)
[Product 22] --> [Product 12] (confidence: 0.364)
[Product 31] --> [Product 12] (confidence: 0.417)
[Product 20, Product 27] --> [Product 12] (confidence: 0.429)
```

图 15 部分关联规则

实验任务

按照上述实验操作步骤将 AI Studio 内置数据（Samples/data/Transactions）进行关联规则挖掘。