SỞ GIÁO DỤC VÀ ĐÀO TẠO THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC FPT

-----OO🕮OO-----

FPT UNIVERSITY

ASSIGNMENT 02

DATA STRUCTURE AND ALGORITHM

Người thực hiện: **ĐẶNG HOÀNG NGUYÊN – SE171946**

Lớp : **IA1708**

Khoá : **17**

Người thực hiện: **NGUYỄN MINH TRÍ – SE172250**

Lớp : **IA1708**

Khoá : **17**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Trong bối cảnh của thời đại thông tin, có thể hình dung được tầm quan trọng của tài nguyên dữ liệu và tầm quan trọng của việc sử dụng tài nguyên dữ liệu có nghĩa là - khai thác dữ liệu cũng đã xuất hiện. Trong tình hình hiện tại, tất cả đều đang ở giai đoạn tương đối bình đẳng, nên tận dụng tốt nguồn dữ liệu, sử dụng thuật toán Apriori để khai thác các quy tắc liên kết, xây dựng chiến lược tiếp thị, thúc đẩy tăng trưởng doanh số và làm chậm lại sự mất mát của GDP quốc gia. Các quốc gia cũng có thể sử dụng khai thác dữ liệu để đưa ra dự đoán và hỗ trợ các quyết định. Mô tả ngắn gọn các khái niệm cơ bản về khai phá dữ liệu và các quy tắc kết hợp. Chỉ với sự hiểu biết cơ bản về khai thác dữ liệu và các quy tắc kết hợp, chúng ta mới có thể hiểu rõ hơn về thuật toán Apriori, tại sao nó là một trong những thuật toán cổ điển và có ảnh hưởng nhất. Hiểu bản chất và chức năng của nó, đồng thời đưa ra các đề xuất cải tiến bằng cách phân tích các ý tưởng của thuật toán Apriori, hiểu quy trình, ưu điểm và nhược điểm của nó và hiểu ứng dụng của thuật toán trong lĩnh vực thương mại điện tử và các trường đại học, và hiệu quả mà nó sẽ đạt được.

MỤC LỤC

GIẢI THUẬT APRIORI

1. Giới thiệu về bài toán

Apriori là thuật toán khả sinh được đề xuất bởi R. Agrawal và R. Srikant vào năm 1993 để khai thác các tập item đối với các khai phá luật kết hợp kiểu boolean. Tên của thuật toán dựa trên việc thuật toán sử dụng tri thức trước (prior knowledge) của các thuộc tính tập item phổ biến. Apriori dùng cách tiếp cận lặp được biết đến như tìm kiếm level-wise, với các tập k item được dùng để thăm dò các tập (k+1) item. Đầu tiên, tập các tập 1 item phổ biến được tìm thấy bằng cách quét cơ sở dữ liệu để đếm số lượng từng item, và thu thập những item thỏa mãn độ hỗ trợ tối thiểu (minsup). Tập kết quả đặt là L_1. Tiếp theo, L_1được dùng để tìm L_2, tập các tập 2 item phổ biến, nó được dùng để tìm L_3, và cứ thế tiếp tục, cho tới khi tập k item phổ biến không thể tìm thấy. Việc tìm kiếm cho mỗi L_k đòi hỏi một lần quét toàn bộ cơ sở dữ liệu.

Thuật toán Apriori có độ phức tạp về thời gian là O(k * (k^2 + t * n)) với k là kích thước tập mục phổ biến, t là kích thước cơ sở dữ liệu và n là số tập mục của t. Vậy, độ phức tạp về thời gian sẽ là O(k^3 +k* t * n). Hay là, O(k* t * n) khi t >> k, n >> k.

2. Giải quyết bài toán

2.1 Mô tả cấu trúc dữ liệu

2.1.1 Thuật ngữ trong giải thuật Apriori

Giải thuật Apriori sử dụng khai phá luật kết hợp Boolean (Association Rule Boolean). Quy tắc đơn giản được định nghĩa theo dạng A → B với điều kiện A ∩ B ≠Ø. Trong đó A là điều kiện (antecedent) còn B được coi là kết quả (consequent).

Itemset - Itemset là tập hợp của những item (không trùng) trong cơ sở dữ liệu mà nó được xác định bởi I = {i_1, i_2,…, i_n}, trong đó n là số lượng item.

Transaction (giao dịch)- Transaction là một thành phần mà nó bao gồm tập hợp các item. Transaction được ký hiệu là T và T ⊆ I. Một Transaction chứa tập hợp các item T = {i_1, i_2, …, i_n}.

Minimum support – Minimum support là điều kiện cần được đáp ứng bởi các item đề ra để có thể tiến hành xử lý item kế tiếp có thể. Minimum support có thể được xem như là một điều kiện giúp loại bỏ các tập không phổ biến trong bất kỳ cơ sở dữ liệu. Thường sử dụng Minimum support cho mô hình tỷ lệ phần trăm. Ta có công thức tính Minimum support:

$$\text{Minimum support} = \frac{Tổng\ Transaction * Tỉ\ lệ\ phần\ trăm}{100}$$

Frequent itemset (tập phổ biến) - các Itemset đáp ứng các tiêu chí điều kiện minimum support thì được gọi là tập phổ biến. Nó được ký hiệu là Li trong đó i chỉ là itemset. Độ phức tạp khi xử lý Li = O(n) với n là số giao dịch.

Candidate itemset (ứng viên tập phổ biến) - Candidate itemset là các item chỉ được xem xét xử lý. Ứng viên tập phổ biến là tất cả các kết hợp có thể có của tập phổ biến. Nó thường được ký hiệu Ci.

Support – Độ hữu dụng của một luật kết hợp có thể được đo với sự giúp đỡ của ngưỡng hỗ trợ. Ta có công thức tính độ Support giữa một luật kết hợp:

$$\text{Support}(A \rightarrow B) = \frac{Tổng\ Transaction\ có\ cả\ A\ và\ B}{Tổng\ Transaction}$$

Confidence – Confidence chỉ sự chắc chắn của các luật. Đây là tỉ lệ giữa giao dịch bao gồm cả A và B so với số lượng giao dịch chỉ A. Ta có công thức như sau:

$$\text{Confidence}(A \rightarrow B) = \frac{Support(AUB)}{Support(A)}$$

Lưu ý: $\text{Confidence}(A \rightarrow B) \neq \text{Confidence}(B \rightarrow A)$

2.1.2 Tính chất Apriori

Tính chất này còn được gọi là tính chất chống đơn điệu (anti-monotone) cho biết rằng nếu một tập con không vượt qua được ngưỡng Minimum support thì tất cả các tập cha của chúng (super set) không thể là tập phổ biến được.

2.1.3 Các giai đoạn của thuật toán

Đầu tiên, tìm tập phổ biến được tìm thấy ký hiệu là C_1.

Bước tiếp theo là tính minimum support.

Sau đó, bước chọn lọc được thực hiện trên C_1 trong đó những item được so sánh với thông số minimum support. Những item thỏa điều kiện minimum support mới được xem xét cho tiến trình tiếp theo ký hiệu là L_1.

Sau đó, bước phát sinh các bộ ứng viên được thực hiện trong đó tập phổ biến 2 được tạo ra ký hiệu là C_2.

Một lần nữa, cở sở dữ liệu được duyệt để tính toán support của 2 tập phổ biến. Theo minimum support, các bộ ứng viên tạo ra được kiểm tra và chỉ những tập phổ biến nào thỏa điều kiện minimum support thì tiếp tục được sử dụng tạo ra bô ứng viên tập phổ biến 3.

Bước trên tiếp tục cho đến khi không có tập phổ biến hoặc bộ ứng viên có thể được tạo ra.

Ví dụ:

$$\text{Minimum support} = \frac{4 * 50}{100} = 2$$

Database cho sẵn với 50%

TID	Items
1	A,C,D
2	B,C,E
3	A,B,C,E
4	B,E

Lần duyệt đầu

Lựa ra các con và tìm độ phổ biến của chúng

C_1

ItemSet	Support
A	2
B	3
C	3
D	1
E	3

L_1

So sánh từng Support của từng ItemSet và loại bỏ Support < minsup

ItemSet	Support
A	2
B	3
C	3
E	3

Tạo các tổ hợp ItemSet mới dựa trên các tập phổ biến đầu.

ItemSet
A,B
A,C
A,E
B,C
B,E
C,E

C_2

ItemSet	Support
A,B	1
A,C	2
A,E	1
B,C	2
B,E	3
C,E	2

L_2

ItemSet	Support
A,C	2
B,C	2
B,E	3
C,E	2

C_3

ItemSet	Support
A,C,E	0
A,C,B	0
B,E,C	2

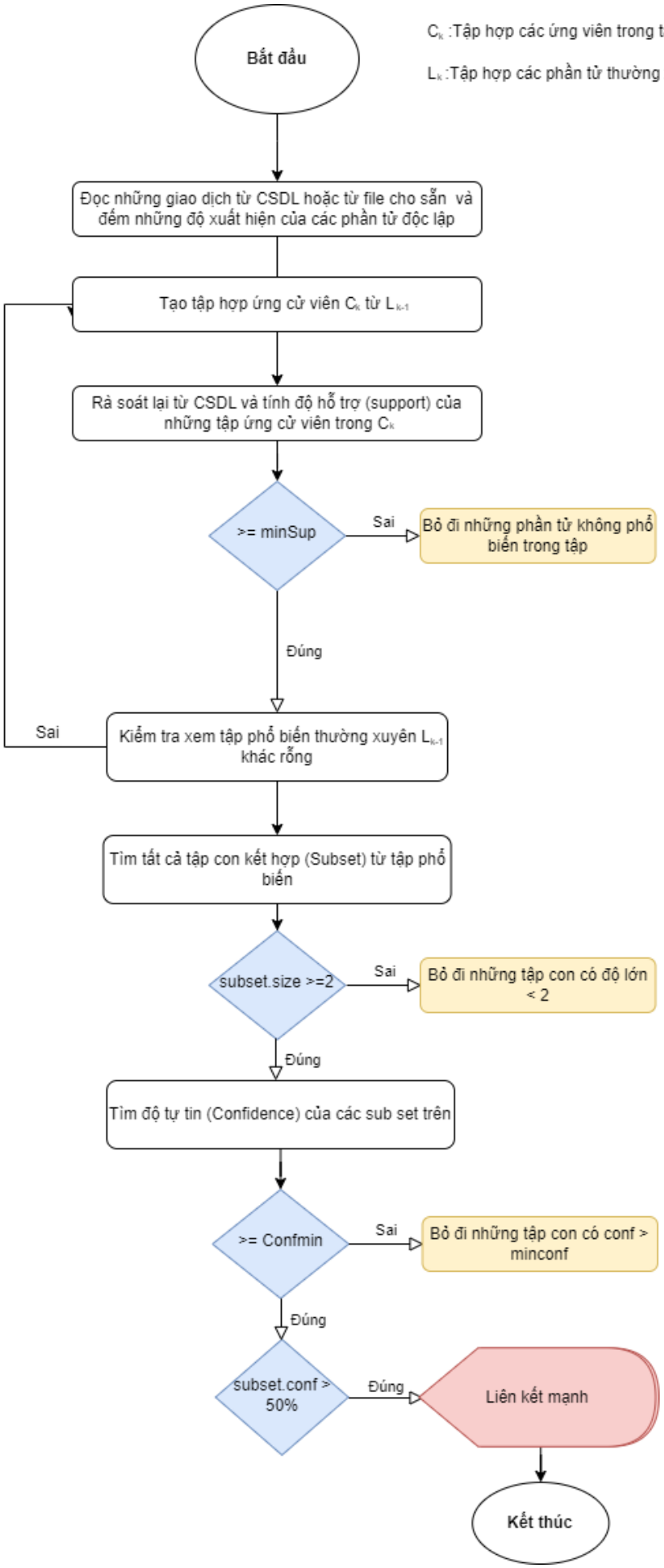
L_3

ItemSet	Support
B,C,E	2

Sau khi thực hiện xong bước tìm tập L phổ biến, ta bắt đầu ra kết quả tìm kiếm khai phá tập luật kết hợp của các tập phổ biến, ta được kết quả sau với min_confident lớn hơn 50% là kết hợp mạnh (Strong Association) với công thức:

$$\text{Confidence}(A \rightarrow B) = \frac{Support(AUB)}{Support(A)}$$

2.2 Sơ đồ giải thuật


Bắt đầu
C_k :Tập hợp các ứng viên trong tập itemset thứ k
L_k :Tập hợp các phần tử thường xuyên trong tập với độ dài là k
Đọc những giao dịch từ CSDL hoặc từ file cho sẵn và đếm những độ xuất hiện của các phần tử độc lập
Tạo tập hợp ứng cử viên C_k từ L_{k-1}
Rà soát lại từ CSDL và tính độ hỗ trợ (support) của những tập ứng cử viên trong C_k
>= minSup
Sai
Bỏ đi những phần tử không phổ biến trong tập
Đúng
Sai
Kiểm tra xem tập phổ biến thường xuyên L_{k-1} khác rỗng
Tìm tất cả tập con kết hợp (Subset) từ tập phổ biến
subset.size >=2
Sai
Bỏ đi những tập con có độ lớn < 2
Đúng
Tìm độ tự tin (Confidence) của các sub set trên
>= Confmin
Sai
Bỏ đi những tập con có conf > minconf
Đúng
subset.conf > 50%
Đúng
Liên kết mạnh
Kết thúc

2.3 Hiện thực bài toán

Hàm tính các phần tử độc lập trong khi k = 1 (chạy lần đầu tiên), và phải lớn hơn minSup

```
public Set<ItemSet> getCadidates() {
    Set<ItemSet> setLi = new HashSet();
    for (ItemSet it : this.dataSet.values()) {
        for (Integer number : it.itemset) {
            int count = countFrequency(number);
            Set<Integer> setItem = new HashSet();
            setItem.add(number);
            ItemSet itemset = new ItemSet(setItem, count);
            if (!checkContain(itemset, setLi)) {
                setLi.add(itemset)

            }
        }
        setLi = checkMinSupport(setLi);

    }
    return setLi;
}
```

Hàm tính phần tử ghép nối về sau khi k > 1 (Giai đoạn phát sinh)

```
public Set<Set<Integer>> getCandidates(Set<ItemSet> set, int size) {

    Set<Set<Integer>> result = new HashSet();
    int count = 0;
    int[] arr = new int[set.size()];
    int n = arr.length;
    for (ItemSet it : set) {
        for (Integer i : it.itemset) {
            arr[count] = i;
        }
        count++;
    }
    for (int i = 0; i < (1 << n); i++) {
        Set<Integer> setA = new HashSet();
        for (int j = 0; j < n; j++) {
            if ((i & (1 << j)) > 0) {
                setA.add(arr[j]);
            }
        }
        if (setA.size() == size) {
            result.add(setA);
        }
    }
    if (result.isEmpty()) {
        return null;
    }
    return result;
}
```

Hàm thực hiện chức năng của thuật toán Apriori

```
    public Set<ItemSet> apriorigen() {
        System.out.println("Growth number 1");
        Set<ItemSet> setCan = getCadidates();
        this.setL.addAll(setCan);
        printSetItem(this.setL);
        int size = 2;
        Set<Set<Integer>> setGen = new HashSet();
        while ((setGen = getCandidates(setCan, size)) != null) {
            System.out.println("Growth number " + size + ": ");
            Set<ItemSet> setItem = new HashSet();
            for (Set<Integer> item : setGen) {
                if (has_subnet(item)) {
                    int count = countFrequency(item);
                    setItem.add(new ItemSet(item, count));
                }
            }
            setItem = checkMinSupport(setItem);
            this.setL.addAll(setItem);
            printSetItem(this.setL);
            size++;
        }
        return setL;
    }
```

Sau khi tìm ra được tập L phổ biến, ta bắt đầu tới giai đoạn thứ 2, giai đoạn phát sinh luật kết hợp để tìm ra được liên kết mạnh

```
    public Set<Law> getLawGen(){
        Set<Law> setLaw = new HashSet();
        for (ItemSet it : this.setL) {
            if(it.itemset.size() >= 2){
                String[] arr = new String[it.itemset.size()];
                int count = 0;
                for (Integer integer : it.itemset) {
                   arr[count] = integer + "";
                   count++;
                }
                AssociationLaw aL = new AssociationLaw(arr);
                Set<Law> setTempLaw = aL.getLawGenerate();
                for (Law law : setTempLaw) {
                    int countASet = countFrequency(law.setA);
                    Set<Integer> setB = new HashSet();
                    setB.addAll(law.setB);
                    setB.addAll(law.setA);
                    int countBSet = countFrequency(setB);
                     double minconfidence =
getMinConfidence(countBSet,countASet);
                    law.setMin_conf(minconfidence);
                    setLaw.add(law);
```

```
                }
            }
        }
        printLaw(setLaw);
        return setLaw;
    }
```

Hàm tính min_confidence của một một liên kết

```
    private double getMinConfidence(int countASet, int countBSet) {
        return (double) countASet / countBSet;
    }
```

2.4 Kết quả và thảo luận

2.4.1 Kết quả đạt được

So với lý thuyết mà chúng em đã nói như trên, chúng em đã tìm ra được tập hợp L sau nhiều lần phát sinh, đã trả ra kết quả tập L phổ biến của chúng với độ minimum support của chúng đề lớn hơn 2, như trên

```
Item  Support
[2, 3, 5]  2.0
[3]  3.0
[3, 5]     2.0
[2, 3]     2.0
[2, 5]     3.0
[2]  3.0
[1]  2.0
[5]  3.0
[1, 3]     2.0
```

Ta bắt đầu đi tìm các liên kết mạnh. Liên kết mạnh là các liên kết min_confident > 50%:

$$2 \rightarrow 3,5:\ \text{Confidence}(A \rightarrow B) = \frac{Support(2,3,5)}{Support(2)} = \frac{2}{3} \sim 0.667$$

Có nghĩa là khi mua món đồ số 2 thì tỉ lệ mua món 3 và 5 là sấp xĩ 67%

$$2,3 \rightarrow 5:\ \text{Confidence}(A \rightarrow B) = \frac{Support(2,3,5)}{Support(2,3)} = \frac{2}{2} \sim 1$$

Có nghĩa là khi mua món 2 và 3 thì tỉ lệ mua món 5 sẽ sấp xĩ 100%

```
Growth association:
2 --> 35    min_conf= 66.667 %    Strong Association
5 --> 23    min_conf= 66.667 %    Strong Association
23 --> 5    min_conf= 100.0 %     Strong Association
3 --> 25    min_conf= 66.667 %    Strong Association
25 --> 3    min_conf= 66.667 %    Strong Association
35 --> 2    min_conf= 100.0 %     Strong Association
```

Áp dụng đoạn code example thầy gửi ta có:

```
!pip install mlxtend

dataset = [['1','3', '4'],
           ['2', '3', '5'],
           ['1', '2', '3', '5'],
           ['2', '5']]

import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)

from mlxtend.frequent_patterns import apriori

apriori(df, min_support=0.5)

apriori(df, min_support=0.5,use_colnames=True)
```

So sánh 2 ouput từ hai file

	support	itemsets
0	0.50	(1)
1	0.75	(2)
2	0.75	(3)
3	0.75	(5)
4	0.50	(3, 1)
5	0.50	(3, 2)
6	0.75	(2, 5)
7	0.50	(3, 5)
8	0.50	(3, 2, 5)

Item	Support
[1, 3]	2.0
[2, 3, 5]	2.0
[1]	2.0
[3]	3.0
[2, 5]	3.0
[2]	3.0
[3, 5]	2.0
[5]	3.0
[2, 3]	2.0

Kết quả có không có sự sai lệch, bởi vì độ support của hình bên trái được tính theo:

$$\text{Minimum support} = \frac{Tỉ\ lệ\ phần\ trăm}{100}$$

Của hình bên phải ta sẽ ra những support lớn hơn 1 bởi vì hình bên phải được tính theo công thức:

$$\text{Minimum support} = \frac{Tổng\ Transaction * Tỉ\ lệ\ phần\ trăm}{100}$$

Nếu ta lấy *Minimum Support/ Tổng transaction* thì ta sẽ ra kết quả giống hình bên trái

2.4.2 Hạn chế

Tổng quan, thuật toán Apriori còn có những hạn chế nhất định, và độ phức tạp của thuật toán bigO về thời gian khá lớn O(k * (k^2 + t * n)) với k là kích thước tập mục phổ biến, t là kích thước cơ sở dữ liệu và n là số tập mục của t. Vì thế khi muốn xác định độ support của các ứng viên, thuật toán luôn phải quét lại toàn bộ cơ sở dữ liệu đã được nhập vào, vì vậy sẽ tốn rất nhiều thời gian để quét.

2.4.3 Hướng phát triển

So với các thuật toán khác, với những mặt hạn chế về thuật toán Apriori đó, nhóm chúng em đã tìm hiểu và có được những hướng giải quyết sau đối với thuật toán:

- Cải tiến thuật toán Apriori dựa vào ma trận
- Cải tiến thuật toán theotrọng số Apriori
- Cải tiến thuật toán apriori dựa vào Interest itemset
- Cải tiến thuật toán Apriori dựa vào nén giao tác
- Cải tiến thuật toán Apriori dựa vào thuật toán OOO
- Cải tiến thuật toán Apriori dựa vào mô hình Map/reduce

3 Kết luận

3.1 Đối với dự án

Ngày nay, các thuật toán liên quan của các luật kết hợp đã thành công trong nhiều lĩnh vực. Thông qua chủ đề này, chúng ta có thể hiểu sâu hơn về quy trình cụ thể và cốt lõi của thuật toán Apriori, đồng thời hiểu rõ hơn về giá trị của các luật kết hợp. Ngoài ra, đã thành công mỹ mãn trong việc áp dụng thuật toán Apriori vào trong Java để có thể phân tích được tập luật phổ biến của dữ liệu được đưa vào trong chương trình.

3.1 Đối với bản thân nhóm

Thông qua bài toán Apriori tìm tập luật phổ biến, nhóm chúng em đã rút ra được nhiều bài học quý giá trong việc làm Assignment thứ 2 của môn CSD201:

- Quản lý và hiểu code một cách chặt chẽ.
- Xây dựng quy trình kế hoạch rõ rang trong lúc làm dự án.
- Luôn chuẩn bị những kế hoạch dự phòng trong lúc thực hiện dự án.
- Viết code luôn phải có ghi chú (note) để thuận tiện cho việc chỉnh sửa về sau, đồng thời cũng giúp mọi người trong dự án hiểu mình đang viết những dòng lệnh gì.
- Viết code luôn phải commit và push github liên tục, viết đủ task hôm đó để cả hai cùng bắt kịp tiến độ công việc.

TỰ ĐÁNH GIÁ – ĐỀ TÀI NHÓM

Nội dung	Điểm	Ghi chú
Giới thiệu về bài toán (0.25 đ)	0.25	
Mô tả cấu trúc dữ liệu (1.25 đ)	1.25	
Sơ đồ giải thuật (1.5 đ)	1.25	
Hiện thực (5 đ)	4.50	
Kết quả và thảo luận (0.5 đ)	0.50	
Điểm nhóm (0.75 đ)	0.75	
Các điều rút ra cho bản thân (0.25 đ)	0.25	
Báo cáo đúng theo mẫu (0.5 đ)	0.50	
Tổng điểm	9.30	

TÀI LIỆU THAM KHẢO

1. Sun, L. (2019, January 28). *An improved apriori algorithm based on support weight matrix for data mining in transaction database*. SpringerLink. https://link.springer.com/article/10.1007/s12652-019-01222-4?error=cookies_not_supported&code=d1eca95c-31d7-4c1f-9bda-ceb221bcf7be
2. Jiao, D. W. (2013). Research and Improvement of an Association Rule Algorithm. Ph.D. Thesis, Changchun: Changchun University of Science and Technology.
3. *An Improvement of Apriori Mining Algorithm using Linked List Based Hash Table*. (2020, November 4). IEEE Conference Publication. https://ieeexplore.ieee.org/abstract/document/9261804
4. Middleton A.M, Hanbook of Cloud Computing, Data-Intensive Technologies for Cloud Computing, FL, USA, Springer: 83-136, 2010.
5. Nilesh.S.Korde & Shailendra.W.Shende, Parallel Implementation of Apriori Algorithm, IOSR Journal of Computer Science (IOSR-JCE), e-ISSN: 2278-0661, p-ISSN: 2278-8727, PP 01-04, 2014.
6. Sinh, P. Đ., & Thành, T. T. (n.d.). *Cải tiến thuật toán Apriori dựa vào mô hình Map/reduce*.
7. Journal, I. (2014, June 21). AN IMPROVED APRIORI-BASED ALGORITHM FOR ASSOCIATION RULE MINING https://www.academia.edu/7419045/AN_IMPROVED_APRIORI_BASED_ALGORITHM_FOR_ASSOCIATION_RULE_MINING
8. GeeksforGeeks. (2022, January 13). Apriori Algorithm. https://www.geeksforgeeks.org/apriori-algorithm
9. edureka! (2019, June 19). *Apriori Algorithm Explained | Association Rule Mining | Finding Frequent Itemset | Edureka* [Video]. YouTube. https://www.youtube.com/watch?v=guVvtZ7ZClw