## **BÁO CÁO**

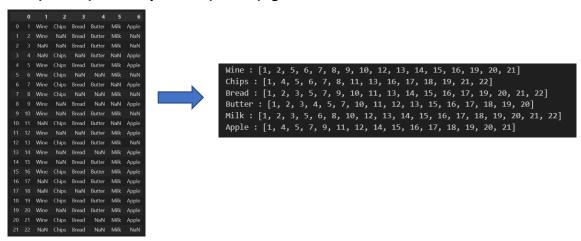
## Thực hành Khai thác dữ liệu - Tuần 05

MSSV - Họ tên : 20280071 - Nguyễn Tiến Phong

### Thuật toán Vertical Apriori:

#### Các bước thực hiện:

Đọc dữ liệu và chuyển dữ liệu từ dạng horizontal thành vertical



- Tính Support của các frequent\_itemsets:

Trong đó, hàm count item để đếm tần suất xuất hiện của itemset:

```
# Đếm các phần tử giống nhau của array trong dict

def count_item(cnt):
    common_elements = set()

for array_name, array_data in cnt.items():
    if not common_elements:
    | common_elements.update(array_data)
    else:
    | common_elements.intersection_update(array_data)

num_common_elements = len(common_elements)
    return num_common_elements

1    ✓ 0.0s
```

Tìm các frequent\_itemsets có kích thước 1

Sinh các ứng viên có kích thước lớn hơn từ các frequent\_itemsets trước đó

```
# Sinh các ứng viên cho frequent_itemsets có kích thước lớn hơn từ các frequent_itemsets trước đó

def generate_candidate_itemsets(pre_itemsets):

candidate_itemsets = set()

for itemset1 in pre_itemsets:

| for itemset2 in pre_itemsets:

| if itemset1 != itemset2 :

| new_itemset2 :

| new_itemset = tuple(sorted(set(itemset1) | set(itemset2))) # Kết hợp các itemset lại

if len(new_itemset) == len(itemset1) + 1: # Kích thước được tăng lên 1

| candidate_itemsets.add(new_itemset)

return candidate_itemsets
```

- Triển khai thuật toán Vertical Apriori

```
def vertical_apriori(data, min_support, max_length):
      frequent_itemsets = find_frequent_itemsets(data, min_support)
      k = 0
      k = max(len(itemset) for itemset in frequent_itemsets)
      while k < max_length:</pre>
          frequent_itemsets = generate_candidate_itemsets(frequent_itemsets)
          k = max(len(itemset) for itemset in frequent_itemsets)
      support_itemsets = calculate_support(frequent_itemsets, data)
      support = {}
      frequent_itemset = []
      for itemset, supp in support_itemsets.items():
          if supp > min_support:
              frequent_itemset.append(itemset)
              support[itemset] = supp
      return support
✓ 0.1s
```

- In ra kết quả

```
# In ket qua'
min_support = 0.3
max_length = 3
support = vertical_apriori(vertical_df, min_support, max_length)
data = pd.DataFrame()
data['Frequent_Itemset'] = support.keys()
data['Support'] = support.values()
data
```

# So sánh với kết quả khi sử dụng thư viện pyECLAT : Kết quả tương tự nhau

	Frequent_Itemset	Support
0	(Apple, Bread, Milk)	0.409091
1	(Bread, Butter, Milk)	0.500000
2	(Butter, Milk, Wine)	0.454545
	(Bread, Chips, Wine)	0.318182
4	(Apple, Chips, Milk)	0.318182
5	(Apple, Bread, Butter)	0.409091
6	(Apple, Milk, Wine)	0.409091
7	(Apple, Butter, Wine)	0.363636
8	(Bread, Chips, Milk)	0.363636
9	(Bread, Milk, Wine)	0.500000
10	(Apple, Butter, Chips)	0.409091
11	(Apple, Bread, Wine)	0.454545
12	(Bread, Butter, Wine)	0.454545
13	(Apple, Butter, Milk)	0.409091
14	(Chips, Milk, Wine)	0.363636
15	(Butter, Chips, Milk)	0.318182
16	(Apple, Bread, Chips)	0.363636
17	(Bread, Butter, Chips)	0.363636

```
eclat = ECLAT(data=df, verbose=True)
frequent_itemsets, support = eclat.fit(min_support=0.3, min_combination=3, separator=' & ', verbose=True)

support

✓ 0.7s

100%| | 28/28 [00:00<00:00, 139.96it/s]
100%| | 28/28 [00:00<00:00, 1025.25it/s]
100%| | 28/28 [00:00<00, 1025.25it/s]
100%| | 28/28 [00:00<00, 1025.25it/s]
100%|
```