Data Mining

Mohil Parmar

23010101192

Lab - 7 (Part 2)

```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

Step 1: Load the Dataset

Load the `Tdata.csv` file and display the first few rows.

```
In [4]: Tdata = pd.read_csv('Tdata.csv')
        Tdata
```

Out[4]:

	Transaction	bread	butter	coffee	eggs	jam	milk
0	T1	1	1	0	0	0	1
1	T2	1	1	0	0	1	0
2	T3	1	0	0	1	0	1
3	T4	1	1	0	0	0	1
4	T5	1	0	1	0	0	0
5	T6	0	0	1	1	1	0

Step 2: Drop the 'Transaction' Column

We're only interested in the items (not the transaction IDs).

In [8]:
```
Tdata = Tdata.drop(columns="Transaction")
Tdata
```

Out[8]:

	bread	butter	coffee	eggs	jam	milk
0	1	1	0	0	0	1
1	1	1	0	0	1	0
2	1	0	0	1	0	1
3	1	1	0	0	0	1
4	1	0	1	0	0	0
5	0	0	1	1	1	0

Step 3: Count Single Items

See how many transactions include each item.

```
In [11]: Tdata.sum()
```

```
Out[11]: bread     5
         butter    3
         coffee    2
         eggs      2
         jam       2
         milk      3
         dtype: int64
```

Step 4: Define Apriori Function

This function finds frequent itemsets of size 1, 2, and 3 with minimum support.

```
In [14]: from itertools import combinations

         def find_frequent_itemsets(Tdata, min_support):
             n = len(Tdata)
             result = []

             for k in [1, 2, 3]:
                 for items in combinations(Tdata.columns, k):
                     mask = Tdata[list(items)].all(axis=1)
                     support = mask.sum() / n
                     print(f"{set(items)} -> support: {round(support, 2)}")

                     if support >= min_support:
                         result.append((frozenset(items), round(support, 2)))
             return result
```

Step 5: Run Apriori

Set `min_support = 0.6` and display the frequent itemsets.

```
In [17]: frequent_itemsets = find_frequent_itemsets(Tdata, 0.6)
```

```
{'bread'} -> support: 0.83
{'butter'} -> support: 0.5
{'coffee'} -> support: 0.33
{'eggs'} -> support: 0.33
{'jam'} -> support: 0.33
{'milk'} -> support: 0.5
{'butter', 'bread'} -> support: 0.5
{'coffee', 'bread'} -> support: 0.17
{'eggs', 'bread'} -> support: 0.17
{'jam', 'bread'} -> support: 0.17
{'bread', 'milk'} -> support: 0.5
{'coffee', 'butter'} -> support: 0.0
{'eggs', 'butter'} -> support: 0.0
{'jam', 'butter'} -> support: 0.17
{'butter', 'milk'} -> support: 0.33
{'coffee', 'eggs'} -> support: 0.17
{'coffee', 'jam'} -> support: 0.17
{'coffee', 'milk'} -> support: 0.0
{'jam', 'eggs'} -> support: 0.17
{'eggs', 'milk'} -> support: 0.17
{'jam', 'milk'} -> support: 0.0
{'coffee', 'butter', 'bread'} -> support: 0.0
{'eggs', 'butter', 'bread'} -> support: 0.0
{'butter', 'jam', 'bread'} -> support: 0.17
{'butter', 'bread', 'milk'} -> support: 0.33
{'eggs', 'coffee', 'bread'} -> support: 0.0
{'coffee', 'jam', 'bread'} -> support: 0.0
{'coffee', 'bread', 'milk'} -> support: 0.0
{'eggs', 'jam', 'bread'} -> support: 0.0
{'eggs', 'bread', 'milk'} -> support: 0.17
{'jam', 'bread', 'milk'} -> support: 0.0
{'eggs', 'coffee', 'butter'} -> support: 0.0
{'coffee', 'jam', 'butter'} -> support: 0.0
{'coffee', 'butter', 'milk'} -> support: 0.0
{'eggs', 'jam', 'butter'} -> support: 0.0
{'eggs', 'butter', 'milk'} -> support: 0.0
{'jam', 'butter', 'milk'} -> support: 0.0
{'coffee', 'jam', 'eggs'} -> support: 0.17
{'coffee', 'eggs', 'milk'} -> support: 0.0
{'coffee', 'jam', 'milk'} -> support: 0.0
{'jam', 'eggs', 'milk'} -> support: 0.0
```

Step 6 Display as a DataFrame

In [20]:
```
formatted_itemsets = [(tuple(itemset), support) for itemset, support in frequent_itemsets]

frequent_df = pd.DataFrame(formatted_itemsets, columns=["Itemset", "Support"])

frequent_df
```

Out[20]:

	Itemset	Support
0	(bread,)	0.83

Orange Tool : - >Generate Same Frequent Patterns in Orange tools

Extra : - > Define Apriori Function without itertools

In [24]:
```
def apriori_without_itertools(Tdata, min_support):
    n = len(Tdata)
    result = []

    columns = list(Tdata.columns)

    # 1-itemsets
    for i in range(len(columns)):
        item1 = columns[i]
        support = Tdata[item1].sum() / n
        if support >= min_support:
            result.append((frozenset([item1]), round(support, 2)))
            print(f"{set([item1])} -> support: {round(support, 2)}")

    # 2-itemsets
    for i in range(len(columns)):
        for j in range(i + 1, len(columns)):
            item1, item2 = columns[i], columns[j]
            support = Tdata[item1].mul(Tdata[item2]).sum() / n
            if support >= min_support:
                result.append((frozenset([item1, item2]), round(support, 2)))
                print(f"{set([item1, item2])} -> support: {round(support, 2)}")

    # 3-itemsets
    for i in range(len(columns)):
        for j in range(i + 1, len(columns)):
            for k in range(j + 1, len(columns)):
                item1, item2, item3 = columns[i], columns[j], columns[k]
                support = Tdata[item1].mul(Tdata[item2]).mul(Tdata[item3]).sum() / n
                if support >= min_support:
                    result.append((frozenset([item1, item2, item3]), round(support, 2)))
                    print(f"{set([item1, item2, item3])} -> support: {round(support, 2)}")

    return result
```

In [26]:
```
min_support = 0.6
frequent_itemsets = apriori_without_itertools(Tdata, min_support)
```

```
{'bread'} -> support: 0.83
```