Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Data Mining

Lab - 7 (Part 2)

NAME: AYUSH J. MARADIA

In [12]:
```python
import pandas as pd
```

Step 1: Load the Dataset

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

In [13]:
```python
df = pd.read_csv('Tdata.csv')
```

In [14]:
```python
df.head()
```

Out[14]:

	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

Step 2: Drop the 'Transaction' Column

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

In [15]:
```python
df_items = df.drop(columns = ['Transaction'])
```

In [16]:
```python
df_items.head()
```

Out[16]:

	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

Step 3: Count Single Items

See how many transactions include each item.

In [17]:
```python
df_items.sum()
```

Out[17]:
```
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 [23]:
```
from itertools import combinations

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

    for k in [1, 2, 3]: # 1-item, 2-item, 3-item sets
        for items in combinations(df.columns, k):
            mask = df[list(items)].all(axis=1)
            support = mask.sum() / n
            print(f"{frozenset(items)} -> {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 [24]:
```
frequent_itemsets = find_frequent_itemsets(df_items, min_support=0.5)

for itemset, support in frequent_itemsets:
    print(f"{set(itemset)}->support: {support}")
```

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

Step 6 Display as a DataFrame

In [26]:
```
result_df = pd.DataFrame(frequent_itemsets, columns = ['Itemset', 'Support'])
result_df
```

Out[26]:

	Itemset	Support
0	(bread)	0.83
1	(butter)	0.50
2	(milk)	0.50
3	(butter, bread)	0.50
4	(bread, milk)	0.50

In [ ]:

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

In [ ]:

Extra : - > Define Apriori Function without itertools

In [1]:

```python
def find_frequent_itemsets(df, min_support):
    n = len(df)
    result = []

    # Step 1: Frequent 1-itemsets (L1)
    L = []
    for col in df.columns:
        support = df[col].sum() / n
        if support >= min_support:
            itemset = frozenset([col])
            L.append(itemset)
            result.append((itemset, round(support, 2)))

    k = 2
    while L and k <= 3:  # Step 2: Only generate up to 3-itemsets
        Ck = []
        for i in range(len(L)):
            for j in range(i + 1, len(L)):
                union = L[i] | L[j]
                if len(union) == k and union not in Ck:
                    # Prune: all (k-1)-subsets must be frequent
                    subsets = [union - {item} for item in union]
                    if all(sub in L for sub in subsets):
                        Ck.append(union)

        # Step for Support Count
        Lk = []
        for cand in Ck:
            cols = list(cand)
            support = df[cols].all(axis=1).sum() / n
            if support >= min_support:
                Lk.append(cand)
                result.append((cand, round(support, 2)))

        L = Lk
        k += 1

    return result
```