CSE 463 Data Warehousing and Mining

Date: 18-03-2025
Name - Debangan Ghosh
Roll - AP22110010984

AIM:

Write a Python Program to implement vertical mining Algorithm for generating Association rules

Topic explanation:

The **Vertical Mining Algorithm** is an approach for generating association rules in data mining. Unlike traditional horizontal mining methods (like Apriori), it represents transactions in a **vertical format**, where each item is associated with a list of transaction IDs (TIDs) in which it appears.

CODE:

```
from itertools import combinations

def build_vertical_db(transactions):
#convert transactions to vertical list (TID list)
    vertical_db = {}
    for tid, items in enumerate(transactions):
        for item in items:
            if item not in vertical_db:
                vertical_db[item] = set()
            vertical_db[item].add(tid)
    return vertical_db

def get_support(itemset, vertical_db):
#calculate the support of an itemset by intersecting itemlist (TID lists)
    common_tids = set.intersection(*(vertical_db[item] for item in
itemset))
    return len(common_tids), common_tids

def find_frequent_itemsets(vertical_db, min_support):
```

```python
    """ Generate frequent itemsets using vertical mining. """
    frequent_itemsets = {}

    # Start with single-item frequent sets
    for item, tids in vertical_db.items():
        if len(tids) >= min_support:
            frequent_itemsets[frozenset([item])] = tids

    #generate itemsets
    k = 2
    while True:
        new_itemsets = {}
        itemsets = list(frequent_itemsets.keys())

        for i in range(len(itemsets)):
            for j in range(i + 1, len(itemsets)):
                new_itemset = itemsets[i] | itemsets[j]  # Combine sets
                if len(new_itemset) == k:
                    support, tids = get_support(new_itemset, vertical_db)
                    if support >= min_support:
                        new_itemsets[frozenset(new_itemset)] = tids

        if not new_itemsets:
            break  # Stop when no more frequesnt itemsets

        frequent_itemsets.update(new_itemsets)
        k += 1

    return frequent_itemsets

def generate_association_rules(frequent_itemsets, min_confidence):
#generate association rules from frequent itemsets
    rules = []
    for itemset in frequent_itemsets:
        if len(itemset) > 1:
            for i in range(1, len(itemset)):
                for lhs in combinations(itemset, i):
                    lhs = frozenset(lhs)
                    rhs = itemset - lhs
                    support = len(frequent_itemsets[itemset])
                    confidence = support / len(frequent_itemsets[lhs])
                    if confidence >= min_confidence:
                        rules.append((lhs, rhs, support, confidence))
    return rules

#taking sample itemsets
data = [
    ['milk', 'bread', 'butter'],
    ['milk', 'bread'],
    ['milk', 'bread','butter'],
    ['milk', 'bread', 'butter', 'eggs'],
    ['milk','eggs','butter'],
    ['milk', 'bread', 'eggs'],
]

min_support = 2
min_confidence = 0.5

#algorithm run
vertical_db = build_vertical_db(data)
frequent_itemsets = find_frequent_itemsets(vertical_db, min_support)
rules = generate_association_rules(frequent_itemsets, min_confidence)

# Display Results
#print("Frequent Itemsets:")
#for itemset, tids in frequent_itemsets.items():
#    print(f"{set(itemset)} - Support: {len(tids)}")

print("\nAssociation Rules:")
for lhs, rhs, support, confidence in rules:
    print(f"{set(lhs)} => {set(rhs)} (Support: {support}, Confidence:
{confidence:.2f})")
```

Output:

```
Association Rules:
{'milk'} => {'bread'} (Support: 5, Confidence: 0.83)
{'bread'} => {'milk'} (Support: 5, Confidence: 1.00)
{'milk'} => {'butter'} (Support: 4, Confidence: 0.67)
{'butter'} => {'milk'} (Support: 4, Confidence: 1.00)
{'milk'} => {'eggs'} (Support: 3, Confidence: 0.50)
{'eggs'} => {'milk'} (Support: 3, Confidence: 1.00)
{'butter'} => {'bread'} (Support: 3, Confidence: 0.75)
{'bread'} => {'butter'} (Support: 3, Confidence: 0.60)
{'eggs'} => {'bread'} (Support: 2, Confidence: 0.67)
{'butter'} => {'eggs'} (Support: 2, Confidence: 0.50)
{'eggs'} => {'butter'} (Support: 2, Confidence: 0.67)
{'milk'} => {'butter', 'bread'} (Support: 3, Confidence: 0.50)
{'butter'} => {'milk', 'bread'} (Support: 3, Confidence: 0.75)
{'bread'} => {'milk', 'butter'} (Support: 3, Confidence: 0.60)
{'milk', 'butter'} => {'bread'} (Support: 3, Confidence: 0.75)
{'milk', 'bread'} => {'butter'} (Support: 3, Confidence: 0.60)
{'butter', 'bread'} => {'milk'} (Support: 3, Confidence: 1.00)
{'eggs'} => {'milk', 'bread'} (Support: 2, Confidence: 0.67)
{'milk', 'eggs'} => {'bread'} (Support: 2, Confidence: 0.67)
{'eggs', 'bread'} => {'milk'} (Support: 2, Confidence: 1.00)
{'butter'} => {'milk', 'eggs'} (Support: 2, Confidence: 0.50)
{'eggs'} => {'milk', 'butter'} (Support: 2, Confidence: 0.67)
{'milk', 'butter'} => {'eggs'} (Support: 2, Confidence: 0.50)
{'milk', 'eggs'} => {'butter'} (Support: 2, Confidence: 0.67)
{'butter', 'eggs'} => {'milk'} (Support: 2, Confidence: 1.00)
```

Google Colab Link (Code Run):

Link (Click here👆)

Brief Explanation for Report

Functions:

1. **build_vertical_db(transactions)**: Converts transactions into a **vertical database (TID-lists)**.
2. **get_support(itemset, vertical_db)**: Computes **support** by **intersecting TID-lists** of items.
3. **find_frequent_itemsets(vertical_db, min_support)**: Extracts **frequent itemsets** using **vertical mining**.
4. **generate_association_rules(frequent_itemsets, min_confidence)**: Derives **association rules** with confidence filtering.

Algorithm Steps:

1. Convert transactions into a **vertical format** (TID-lists).
2. Identify **frequent itemsets** using **TID-list intersections**.
3. Generate **association rules** from frequent itemsets.

Measures Used:

- **Support**: Fraction of transactions containing an itemset.
- **Confidence**: Strength of an association rule (likelihood of RHS given LHS).