**DEPARTMENT OF INFORMATION TECHNOLOGY**

**Course: Data Mining & Business Intelligence Lab (ITL601)**

**B.Tech. (Information Technology) – Semester VI**

**Academic Year: 2023-24 (Even Semester)**

**PRACTICAL 8**

**Aim:** Implement and evaluate any Frequent Pattern Mining Algorithm using Python

**Lab Objective:** To learn how to gather and analyse large sets of data to gain useful business understanding.

**Theory:**

Frequent pattern mining algorithms are essential in data mining for identifying recurring patterns within datasets. These algorithms play a crucial role in uncovering relationships and dependencies among items. Here is an explanation of frequent pattern mining algorithms based on the provided sources:

1. **Apriori Algorithm**:
   * The Apriori algorithm is a classic method for mining frequent patterns in large datasets. It operates by iteratively generating candidate itemsets and pruning those that do not meet the minimum support threshold. This algorithm is efficient but can be computationally expensive for datasets with many infrequent itemsets.
2. **FP-growth Algorithm**:
   * The FP-growth algorithm is another popular approach for frequent pattern mining. It constructs a tree-like data structure called the FP tree, where each node represents an item in a frequent pattern. By scanning the dataset efficiently, FP-growth can mine all frequent itemsets without explicitly generating candidate itemsets. It is particularly suitable for datasets with long patterns and low support thresholds.

**Applications**

1. **Market Basket Analysis**:
   * Frequent pattern mining algorithms are extensively used in market basket analysis to identify associations between items frequently purchased together. This analysis helps businesses understand customer behavior, optimize product placement, and enhance sales strategies.
2. **Bioinformatics**:
   * In bioinformatics, frequent pattern mining is applied to DNA sequences, protein structures, and gene expressions to identify common patterns. This aids in gaining insights into genetics, drug design, and understanding biological processes at a molecular level.
3. **Web Mining**:
   * Frequent pattern mining algorithms play a crucial role in web mining by discovering navigational patterns, user preferences, and collaborative filtering recommendations on the web. This helps improve user experience, personalize content, and enhance web services.
4. **Network Traffic Analysis**:
   * Frequent pattern mining is utilized in network traffic analysis to identify patterns in data transmission, detect anomalies, and enhance network security. By analyzing frequent patterns, network administrators can optimize network performance and detect potential threats.
5. **Customer Behavior Analysis**:
   * Frequent pattern mining algorithms are employed in customer behavior analysis to understand purchasing patterns, preferences, and trends. This analysis enables businesses to tailor marketing strategies, improve customer satisfaction, and enhance customer retention.
6. **Recommender Systems**:
   * Frequent pattern mining is crucial in building recommender systems that provide personalized recommendations to users based on their preferences and behavior. By identifying frequent patterns, these systems can suggest relevant items or content to users, enhancing user experience and engagement.

**Output:**

Source:

from itertools import combinations

def load\_data():

return [

['A', 'B', 'C', 'D'],

['A', 'B', 'C', 'D', 'E', 'G'],

['A', 'C', 'G', 'H', 'K'],

['B', 'C', 'D', 'E', 'K'],

['D', 'E', 'F', 'H', 'L'],

['A', 'B', 'C', 'D', 'L'],

['B', 'I', 'E', 'K', 'L'],

['A', 'B', 'D', 'E', 'L', 'K'],

['A', 'E', 'E', 'H', 'L'],

['B', 'C', 'D', 'F'],

]

def create\_candidates(dataset, k):

candidates = []

for transaction in dataset:

for itemset in combinations(transaction, k):

if itemset not in candidates:

candidates.append(itemset)

return candidates

def support\_count(dataset, candidate):

count = 0

for transaction in dataset:

if set(candidate).issubset(set(transaction)):

count += 1

return count

def apriori(dataset, min\_support):

itemsets = []

k = 1

while True:

candidates = create\_candidates(dataset, k)

frequent\_itemsets = []

for candidate in candidates:

support = support\_count(dataset, candidate)

if support >= (min\_support \* len(dataset)):

frequent\_itemsets.append((candidate, support))

if not frequent\_itemsets:

break

itemsets.extend(frequent\_itemsets)

k += 1

return itemsets

def generate\_association\_rules(frequent\_itemsets, min\_confidence, max\_items = 2):

association\_rules = []

for itemset, support in frequent\_itemsets:

if len(itemset) < max\_items:

continue

for i in range(1, len(itemset)):

for antecedent in combinations(itemset, i):

consequent = tuple(set(itemset) - set(antecedent))

antecedent = tuple(antecedent)

if not consequent or not antecedent:

continue

confidence = support / dict(frequent\_itemsets)[antecedent]

if confidence >= min\_confidence:

association\_rules.append((antecedent, consequent, confidence))

return association\_rules

dataset = load\_data()

max\_items = int(input("Enter maximum number of items in itemset: "))

min\_support = float(input("Enter Minimum Support: "))

min\_confidence = float(input("Enter Minimum Confidence: "))

frequent\_itemsets = apriori(dataset, min\_support)

print()

association\_rules = generate\_association\_rules(frequent\_itemsets, min\_confidence, max\_items)

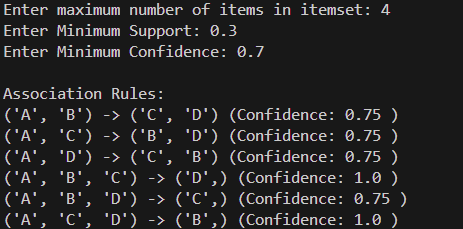
print("Association Rules:")

for antecedent, consequent, confidence in association\_rules:

print(antecedent, "->", consequent, "(Confidence:", confidence, ")")



Output:



**Conclusion:**

We learnt about Frequent Mining Algorithms and their different types. We also implemented the Apriori algorithm in Python, generating strong association rules for a given problem.

**Lab Outcome:** Implement various data mining algorithms from scratch using languages like Python / Java/R, etc.

**Submitted Details -**

**Name of Student: Parth Malviya**

**Roll No.: 22**

**Date of Performance:**

**Date of Submission:**