Discovering High Utility patterns in Big Data Using EFIM Algorithm

In this tutorial, we will discuss two approaches to find High Utility patterns in big data using EFIM algorithm.

1. **Basic approach:** Here, we present the steps to discover High Utility patterns using a single minimum utility value
2. **Advanced approach:** Here, we generalize the basic approach by presenting the steps to discover High Utility patterns using multiple minimum utility values.

Basic approach: Executing EFIM on a single dataset at a particular minimum utility value

Step 1: Import the EFIM algorithm

In [1]:
```
from PAMI.highUtilityPatterns.basic import EFIM  as alg
```

Step 2: Specify the following input parameters

In [2]:
```
inputFile = 'retail_utility.txt'

  #Users can also specify this constraint between 0 to 1.
minimumUtility=50000
seperator=' '
```

Step 3: Execute the EFIM algorithm

In [3]:
```
obj = alg.EFIM(iFile=inputFile,minUtil=minimumUtility,   sep=seperator)    #initiali
obj.startMine()            #Start the mining process
```

```
High Utility patterns were generated successfully using EFIM algorithm
```

Step 4: Storing the generated patterns

Step 4.1: Storing the generated patterns in a file

In [4]:
```
obj.savePatterns(outFile='frequentPatternsMinSupCount100.txt')
```

Step 4.2. Storing the generated patterns in a data frame

In [5]:
```
frequentPatternsDF= obj.getPatternsAsDataFrame()
```

Step 5: Getting the statistics

Step 5.1: Total number of discovered patterns

In [6]:
```
print('Total No of patterns: ' + str(len(frequentPatternsDF)))
```

```
Total No of patterns: 47
```

Step 5.2: Runtime consumed by the mining algorithm

In [7]:
```
print('Runtime: ' + str(obj.getRuntime()))
```

```
Runtime: 15.24892783164978
```

In [8]:
```
##### Step 5.3: Total Memory consumed by the mining algorithm
```

In [9]:
```
print('Memory (RSS): ' + str(obj.getMemoryRSS()))
print('Memory (USS): ' + str(obj.getMemoryUSS()))
```

```
Memory (RSS): 195989504
Memory (USS): 157261824
```