

# Discovering Frequent Patterns in Big Data Using Apriori Algorithm

In this tutorial, we will discuss two approaches to find frequent patterns in big data using Apriori algorithm.

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

---

## Basic approach: Executing Apriori on a single dataset at a particular minimum support value

### Step 1: Import the Apriori algorithm

```
In [1]: from PAMI.frequentPattern.basic import Apriori as alg
```

### Step 2: Specify the following input parameters

```
In [2]: inputFile = 'transactional_T10I4D100K.csv'
minimumSupportCount=1000 #Users can also specify this constraint between 0 to 1.
seperator='¥t'
```

### Step 3: Execute the Apriori algorithm

```
In [3]: obj = alg.Apriori(iFile=inputFile, minSup=minimumSupportCount, sep=seperator) #i
obj.startMine() #Start the mining process
```

Frequent patterns were generated successfully using Apriori algorithm

### Step 4: Storing the generated patterns

#### Step 4.1: Storing the generated patterns in a file

```
In [4]: obj.savePatterns(outFile='frequentPatternsMinSupCount1000.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: 385

### Step 5.2: Runtime consumed by the mining algorithm

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

Runtime: 516.1182169914246

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

```
In [8]: print('Memory (RSS): ' + str(obj.getMemoryRSS()))  
print('Memory (USS): ' + str(obj.getMemoryUSS()))
```

Memory (RSS): 263667712

Memory (USS): 225132544