PAMI Release 2024.04.23

RAGE Uday Kiran

May 20, 2024

CONTENTS

1	Tran	isactional Database	1			
	1.1	Frequent Pattern mining	2			
	1.2	Relative Frequent Pattern	25			
	1.3	Frequent pattern With Multiple Minimum Support	29			
	1.4	Correlated Pattern Mining	37			
	1.5	Fault-Tolerant Frequent Pattern Mining	3			
	1.6	Coverage Pattern Mining	60			
2	Tem	poral Database 5	59			
	2.1	Periodic Frequent Pattern Mining	50			
	2.2	Local Periodic Pattern Mining)5			
	2.3	Partial Periodic Frequent Pattern Mining)9			
	2.4	Partial Periodic Pattern Mining	9			
	2.5	Periodic correlated pattern mining	13			
	2.6	Stable Periodic Pattern Mining	17			
	2.7	Recurring Pattern Mining	50			
3	Geo	referenced Pattern Mining	55			
·	3.1	Geo-referenced Frequent Pattern Mining	57			
	3.2	Geo-referenced Periodic Frequent Pattern Mining	/1			
	3.3	Geo-referenced Partial Periodic Pattern Mining	'5			
1	T 14:1:	ty Dettern mining	21			
4	4 1	Uigh Utility Dettern mining	ת ארי			
	4.1	High Utility Fraquent Dettern Mining)∠ 27			
	4.2	High Utility Coo referenced Frequent Pottern Mining	יי רו			
	4.5	High Utility Spatial Dattorn Mining	י∠ דו			
	4.4	Polotivo High Utility Dottorn Mining	' / 5			
	4.5	Weighted Frequent Dettern Mining	.) 16			
	4.0	Weighted Frequent Pagular Dattern Mining	.0 20			
	4.7	Weighted Frequent Neighbourhood Pottern Mining	25			
	4.0		5			
5	Fuzzy Pattern Mining 241					
	5.1	Fuzzy Frequent Pattern Mining 24	+2			
	5.2	Fuzzy Correlated Pattern Mining 24	+7			
	5.3	Fuzzy Geo-referenced Frequent Pattern Mining	1			
	5.4	Fuzzy Periodic Frequent Pattern Mining 25	6			
	5.5	Fuzzy Geo-referenced Periodic Frequent Pattern Mining 26	0			
6	Unce	ertain Database 26	5 5			
	6.1	Uncertain Frequent Pattern mining	57			

	6.2 Uncertain Periodic Frequent Pattern mining	294	
	6.3 Uncertain Geo-Referenced Frequent Pattern mining	303	
7	Sequential Database	309	
	7.1 Sequential Frequent Pattern mining	309	
	7.2 Geo-referenced Frequent Sequence Pattern mining	323	
8	Multiple Timeseries	325	
	8.1 Multiple Partial Periodic Pattern Mining	326	
9	Contiguous Patterns	331	
	9.1 Contiguous Frequent Patterns	331	
10	10 Indices and tables		
Py	Python Module Index		
Inc	Index		

CHAPTER

ONE

TRANSACTIONAL DATABASE

A transactional database is a set of transactions.

Each transaction contains a transaction-identifier (TID) and a set of items.

Example:

A sample transactional database containing the items from a to f is shown in below.

TID	Transactions
1	a, b, c
2	d, e
3	a, e, f

Rules to create a transactional database:

- Since the TID of a transaction directly represents its row number in a database, we the algorithms in PAMI ignore the TID information to save storage space and processing time.
- The items in a transactional database can be integers or strings.
- All items in a transaction must be seperated with a separator.
- 'Tab space' is the default seperator used by the mining algorithms in PAMI. However, transactional databases can also be constructed using other separators, such as comma and space.

Format:

```
>>> item1<sep>item2<sep>...<sep>itemN
```

Example:

1.1 Frequent Pattern mining

Frequent pattern mining is the process of identifying patterns or associations within a dataset that occur frequently. This is typically done by analyzing large datasets to find items or sets of items that appear together frequently.

Applications: DNA sequences, protein structures, leading to insights in genetics and drug design.

1.1.1 Basic

Apriori

class PAMI.frequentPattern.basic.Apriori.**Apriori**(*iFile*, *minSup*, *sep*=\t')

Bases: _frequentPatterns

About this algorithm

Description

Apriori is one of the fundamental algorithm to discover frequent patterns in a transactional database. This program employs apriori property (or downward closure property) to reduce the search space effectively. This algorithm employs breadth-first search technique to find the complete set of frequent patterns in a transactional database.

Reference

Agrawal, R., Imieli nski, T., Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD. pp. 207–216 (1993), https://doi.org/10.1145/170035.170072

Parameters

- **iFile** (*str or URL or dataFrame*) *Name of the Input file to mine complete set of frequent patterns.*
- **oFile** (*str*) *Name of the output file to store complete set of frequent patterns.*
- minSup (int or float or str) The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.
- **sep** (*str*) *This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.*

Attributes

- startTime (float) To record the start time of the mining process.
- endTime (float) To record the completion time of the mining process.
- finalPatterns (dict) Storing the complete set of patterns in a dictionary variable.
- memoryUSS (float) To store the total amount of USS memory consumed by the program.
- memoryRSS (float) To store the total amount of RSS memory consumed by the program.
- **Database** (*list*) *To store the transactions of a database in list.*

Execution methods

Terminal command

Format:

(.venv) \$ python3 Apriori.py <inputFile> <outputFile> <minSup>

Example Usage:

(.venv) \$ python3 Apriori.py sampleDB.txt patterns.txt 10.0

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

```
import PAMI1.frequentPattern.basic.Apriori as alg
iFile = 'sampleDB.txt'
minSup = 10  # can also be specified between 0 and 1
obj = alg.Apriori(iFile, minSup)
obj.mine()
frequentPattern = obj.getPatterns()
print("Total number of Frequent Patterns:", len(frequentPattern))
obj.save(oFile)
Df = obj.getPatternInDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits

The complete program was written by P. Likhitha and revised by Tarun Sreepada under the supervision of Professor Rage Uday Kiran.

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns() \rightarrow Dict[str, int]

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

$mine() \rightarrow None$

Frequent pattern mining process will start from here

$\texttt{printResults()} \rightarrow \text{None}$

This function is used to print the result

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (csvfile) - name of the output file

Returns None

 $startMine() \rightarrow None$

Frequent pattern mining process will start from here

ECLAT

class PAMI.frequentPattern.basic.ECLAT.**ECLAT**(*iFile*, *minSup*, *sep*=\t') Bases: _frequentPatterns

About this algorithm

Description

ECLAT is one of the fundamental algorithm to discover frequent patterns in a transactional database.

Reference

Mohammed Javeed Zaki: Scalable Algorithms for Association Mining. IEEE Trans. Knowl. Data Eng. 12(3): 372-390 (2000), https://ieeexplore.ieee.org/document/846291

Parameters

- **iFile** (*str or URL or dataFrame*) *Name of the Input file to mine complete set of frequent patterns.*
- **oFile** (*str*) *Name of the Output file to store the frequent patterns.*
- **minSup** (*int or float or str*) The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count.
- **sep** (*str*) This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

- startTime (float) To record the start time of the mining process.
- endTime (float) To record the end time of the mining process.
- finalPatterns (dict) Storing the complete set of patterns in a dictionary variable.
- memoryUSS (float) To store the total amount of USS memory consumed by the program.
- memoryRSS (float) To store the total amount of RSS memory consumed by the program.
- Database (list) To store the transactions of a database in list.

Execution methods

Terminal command

Format:

(.venv) \$ python3 ECLAT.py <inputFile> <outputFile> <minSup>

Example Usage:

(.venv) \$ python3 ECLAT.py sampleDB.txt patterns.txt 10.0

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

```
import PAMI.frequentPattern.basic.ECLAT as alg

iFile = 'sampleDB.txt'

minSup = 10  # can also be specified between 0 and 1

obj = alg.ECLAT(iFile, minSup)

obj.mine()

frequentPatterns = obj.getPatterns()

print("Total number of Frequent Patterns:", len(frequentPatterns))

obj.save(oFile)

Df = obj.getPatternInDataFrame()

memUSS = obj.getMemoryUSS()

print("Total Memory in USS:", memUSS)

memRSS = obj.getMemoryRSS()

print("Total Memory in RSS", memRSS)

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by Kundai and revised by Tarun Sreepada under the supervision of Professor Rage Uday Kiran.

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \rightarrow \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns() \rightarrow dict

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

$mine() \rightarrow None$

Frequent pattern mining process will start from here

$\texttt{printResults()} \rightarrow \text{None}$

Function used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (csvfile) - name of the output file

Returns None

$startMine() \rightarrow None$

Frequent pattern mining process will start from here

ECLATDiffset

class PAMI.frequentPattern.basic.ECLATDiffset.**ECLATDiffset**(*iFile*, *minSup*, *sep*=\t')

Bases: _frequentPatterns

Description

ECLATDiffset uses diffset to extract the frequent patterns in a transactional database.

Reference

KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining August 2003 Pages 326–335 https://doi.org/10.1145/956750.956788

Parameters

- **iFile** (*str or URL or dataFrame*) *Name of the Input file to mine complete set of frequent patterns.*
- oFile (str) Name of the output file to store complete set of frequent patterns
- minSup (int or float or str) The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count.
- sep (*str*) This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

- **startTime** (*float*) *To record the start time of the mining process.*
- endTime (float) To record the end time of the mining process.
- finalPatterns (dict) Storing the complete set of patterns in a dictionary variable.
- memoryUSS (float) To store the total amount of USS memory consumed by the program.
- memoryRSS (float) To store the total amount of RSS memory consumed by the program.
- Database (list) To store the transactions of a database in list.

Execution methods

Terminal command

```
Format:
(.venv) $ python3 ECLATDiffset.py <inputFile> <outputFile> <minSup>
Example Usage:
(.venv) $ python3 ECLATDiffset.py sampleDB.txt patterns.txt 10.0
```

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

<pre>import PAMI.frequentPattern.basic.ECLATDiffset as alg</pre>				
<pre>iFile = 'sampleDB.txt'</pre>				
<pre>minSup = 10 # can also be specified between 0 and 1</pre>				
obj = alg.ECLATDiffset(iFile, minSup)				
obj.mine()				
<pre>frequentPatterns = obj.getPatterns()</pre>				
<pre>print("Total number of Frequent Patterns:", len(frequentPatterns))</pre>				
obj.savePatterns(oFile)				
Df = obj.getPatternInDataFrame()				
<pre>memUSS = obj.getMemoryUSS()</pre>				
<pre>print("Total Memory in USS:", memUSS)</pre>				
<pre>memRSS = obj.getMemoryRSS()</pre>				
<pre>print("Total Memory in RSS", memRSS)</pre>				
<pre>run = obj.getRuntime()</pre>				
<pre>print("Total ExecutionTime in seconds:", run)</pre>				

Credits:

The complete program was written by Kundai and revised by Tarun Sreepada under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine()

Frequent pattern mining process will start from here

printResults()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (csvfile) - name of the output file

startMine()

Frequent pattern mining process will start from here

ECLATbitset

class PAMI.frequentPattern.basic.ECLATbitset.**ECLATbitset**(*iFile*, *minSup*, *sep*=\t')

Bases: _frequentPatterns

Description

ECLATbitset is one of the fundamental algorithm to discover frequent patterns in a transactional database.

Reference

Mohammed Javeed Zaki: Scalable Algorithms for Association Mining. IEEE Trans. Knowl. Data Eng. 12(3): 372-390 (2000), https://ieeexplore.ieee.org/document/846291

Parameters

- **iFile** (*str or URL or dataFrame*) *Name of the Input file to mine complete set of frequent patterns.*
- oFile (str) Name of the output file to store complete set of frequent patterns
- minSup (int or float or str) The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count.
- sep (*str*) This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

- startTime (float) To record the start time of the mining process.
- endTime (float) To record the end time of the mining process.
- finalPatterns (dict) Storing the complete set of patterns in a dictionary variable.
- memoryUSS (float) To store the total amount of USS memory consumed by the program.
- memoryRSS (float) To store the total amount of RSS memory consumed by the program.
- **Database** (*list*) *To store the transactions of a database in list.*

Execution methods

Terminal command

Format:

(.venv) \$ python3 ECLATbitset.py <inputFile> <outputFile> <minSup>

Example Usage:

(.venv) \$ python3 ECLATbitset.py sampleDB.txt patterns.txt 10.0

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

```
import PAMI.frequentPattern.basic.ECLATbitset as alg

iFile = 'sampleDB.txt'

minSup = 10  # can also be specified between 0 and 1

obj = alg.ECLATbitset(iFile, minSup)

obj.mine()

frequentPatterns = obj.getPatterns()

print("Total number of Frequent Patterns:", len(frequentPatterns))

obj.save(oFile)
```

(continues on next page)

(continued from previous page)

```
Df = obj.getPatternInDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by Yudai Masu and revised by Tarun Sreepada under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

mine() \rightarrow None

Frequent pattern mining process will start from here # Bitset implementation

printResults()

This function is used to print the result

save(outFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (file) – name of the outputfile

startMine()

Frequent pattern mining process will start from here

We start with the scanning the itemSets and store the bitsets respectively. We form the combinations of single items and check with minSup condition to check the frequency of patterns

FPGrowth

class PAMI.frequentPattern.basic.FPGrowth.**FPGrowth**(*iFile*, *minSup*, *sep*=\t')

Bases: _frequentPatterns

About this algorithm

Description

FPGrowth is one of the fundamental algorithm to discover frequent patterns in a transactional database. It stores the database in compressed fp-tree decreasing the memory usage and extracts the patterns from tree. It employs downward closure property to reduce the search space effectively.

Reference

Han, J., Pei, J., Yin, Y. et al. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. Data Mining and Knowledge Discovery 8, 53–87 (2004). https://doi.org/10.1023

Parameters

- **iFile** (*str or URL or dataFrame*) *Name of the Input file to mine complete set of frequent patterns.*
- oFile (str) Name of the output file to store complete set of frequent patterns.
- minSup (int or float or str) The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.
- **sep** (*str*) *This variable is used to distinguish items from one another in a transaction. The default seperator is tab space. However, the users can override their default separator.*

Attributes

- startTime (float) To record the start time of the mining process.
- endTime (float) To record the completion time of the mining process.
- **finalPatterns** (*dict*) *Storing the complete set of patterns in a dictionary variable.*
- memoryUSS (float) To store the total amount of USS memory consumed by the program.
- memoryRSS (float) To store the total amount of RSS memory consumed by the program.
- **Database** (*list*) *To store the transactions of a database in list.*
- mapSupport (Dictionary) To maintain the information of item and their frequency.
- **tree** (*class*) *it represents the Tree class*.

Execution methods

Terminal command

```
Format:
```

```
(.venv) $ python3 FPGrowth.py <inputFile> <outputFile> <minSup>
```

Example Usage:

```
(.venv) $ python3 FPGrowth.py sampleDB.txt patterns.txt 10.0
```

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

```
from PAMI.frequentPattern.basic import FPGrowth as alg

iFile = 'sampleDB.txt'

minSup = 10 # can also be specified between 0 and 1

obj = alg.FPGrowth(iFile, minSup)

obj.mine()

frequentPatterns = obj.getPatterns()

print("Total number of Frequent Patterns:", len(frequentPatterns))

obj.savePatterns(oFile)

Df = obj.getPatternInDataFrame()

memUSS = obj.getMemoryUSS()

print("Total Memory in USS:", memUSS)
```

(continues on next page)

(continued from previous page)

```
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P. Likhitha and revised by Tarun Sreepada under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

 $getPatterns() \rightarrow Dict[str, int]$ Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

```
mine() \rightarrow None
```

Main program to start the operation

 $printResults() \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (*csvfile*) – name of the output file

Returns None

startMine()

Starting the mining process

1.1.2 Closed

CHARM

class PAMI.frequentPattern.closed.CHARM.**CHARM**(*iFile*, *minSup*, *sep*=\t')

Bases: _frequentPatterns

Description

CHARM is an algorithm to discover closed frequent patterns in a transactional database. Closed frequent patterns are patterns if there exists no superset that has the same support count as this original itemset. This algorithm employs depth-first search technique to find the complete set of closed frequent patterns in a transactional database.

Reference

Mohammed J. Zaki and Ching-Jui Hsiao, CHARM: An Efficient Algorithm for Closed Itemset Mining, Proceedings of the 2002 SIAM, SDM. 2002, 457-473, https://doi.org/10.1137/1. 9781611972726.27

Parameters

- **iFile** (*str or URL or dataFrame*) *Name of the Input file to mine complete set of frequent patterns.*
- **oFile** (*str*) *Name of the output file to store complete set of frequent patterns.*
- **minSup** (*int or float or str*) *The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.*
- **sep** (*str*) *This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.*

Attributes

- startTime (float) To record the start time of the mining process.
- endTime (float) To record the completion time of the mining process.
- **finalPatterns** (*dict*) *Storing the complete set of patterns in a dictionary variable.*
- memoryUSS (float) To store the total amount of USS memory consumed by the program.
- memoryRSS (float) To store the total amount of RSS memory consumed by the program.

- Database (list) To store the transactions of a database in list.
- **mapSupport** (*Dictionary*) *To maintain the information of item and their frequency.*
- tree (class) It represents the Tree class.
- itemSetCount (int) It represents the total no of patterns.
- tidList (dict) Stores the timestamps of an item.
- hashing (dict) Stores the patterns with their support to check for the closed property.

Execution methods

Terminal command

```
Format:
(.venv) $ python3 CHARM.py <inputFile> <outputFile> <minSup>
```

Example Usage:

(.venv) \$ python3 CHARM.py sampleDB.txt patterns.txt 10.0

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

```
from PAMI.frequentPattern.closed import CHARM as alg
iFile = 'sampleDB.txt'
minSup = 10 # can also be specified between 0 and 1
obj = alg.CHARM(iFile, minSup)
obj.mine()
frequentPatterns = obj.getPatterns()
print("Total number of Closed Frequent Patterns:", len(frequentPatterns))
obj.savePatterns(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
```

(continues on next page)

(continued from previous page)

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by P.Likhitha and revised by Tarun Sreepada under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine()

Mining process will start from here by extracting the frequent patterns from the database. It performs prefix equivalence to generate the combinations and closed frequent patterns.

printResults()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (csvfile) - name of the output file

startMine()

Mining process will start from here by extracting the frequent patterns from the database. It performs prefix equivalence to generate the combinations and closed frequent patterns.

1.1.3 Maximal

MaxFPGrowth

class PAMI.frequentPattern.maximal.MaxFPGrowth.**MaxFPGrowth**(*iFile*, *minSup*, *sep*=\t')

Bases: _frequentPatterns

Description

MaxFP-Growth is one of the fundamental algorithm to discover maximal frequent patterns in a transactional database.

Reference

Grahne, G. and Zhu, J., "High Performance Mining of Maximal Frequent itemSets", http://users. encs.concordia.ca/~grahne/papers/hpdm03.pdf

Parameters

- iFile str : Name of the Input file to mine complete set of frequent patterns
- oFile str : Name of the output file to store complete set of frequent patterns
- **minSup** int or float or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count.
- **maxPer** float : The user can specify maxPer in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

finalPatterns

[dict] Storing the complete set of patterns in a dictionary variable

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

itemSetCount

[int] it represents the total no of patterns

finalPatterns

[dict] it represents to store the patterns

Methods to execute code on terminal

(.venv) \$ python3 MaxFPGrowth.py <inputFile> <outputFile> <minSup>

Example Usage:

Format:

(.venv) \$ python3 MaxFPGrowth.py sampleDB.txt patterns.txt 0.3

Note: minSup will be considered in percentage of database transactions

Importing this algorithm into a python program

```
from PAMI.frequentPattern.maximal import MaxFPGrowth as alg
obj = alg.MaxFPGrowth("../basic/sampleTDB.txt", "2")
obj.mine()
frequentPatterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(frequentPatterns))
obj.savePatterns("patterns")
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
```

(continues on next page)

(continued from previous page)

```
print("Total Memory in RSS", memRSS)
```

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function :return: returning RSS memory consumed by the mining process :rtype: float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function :return: returning USS memory consumed by the mining process :rtype: float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process :return: returning frequent patterns :rtype: dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe :return: returning frequent patterns in a dataframe :rtype: pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process :return: returning total amount of runtime taken by the mining process :rtype: float

mine()

Mining process will start from this function

printResults()

This functon is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to a output file :param outFile: name of the output file :type outFile: csvfile

startMine()

Mining process will start from this function

1.1.4 CUDA

cuApriori

cuAprioriBit

cuEclat

cuEclatBit

cudaAprioriGCT

cudaAprioriTID

cudaEclatGCT

1.1.5 Pyspark

parallelApriori

parallelECLAT

parallelFPGrowth

1.1.6 Top K

FAE

class PAMI.frequentPattern.topk.FAE.**FAE**(*iFile*, k, $sep=\forall t'$)

Bases: _frequentPatterns

Description

Top - K is and algorithm to discover top frequent patterns in a transactional database.

Reference

Zhi-Hong Deng, Guo-Dong Fang: Mining Top-Rank-K Frequent Patterns: DOI: 10.1109/ICMLC.2007.4370261 · Source: IEEE Xplore https://ieeexplore.ieee.org/document/ 4370261

Parameters

- iFile str : Name of the Input file to mine complete set of frequent patterns
- oFile str : Name of the output file to store complete set of frequent patterns
- **k** int : User specified count of top frequent patterns
- minimum int : Minimum number of frequent patterns to consider in analysis
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

startTime

[float] To record the start time of the mining process

[float] To record	the completion time of the mining process
finalPatterns [dict] Storing the	complete set of patterns in a dictionary variable
memoryUSS [float] To store th	ne total amount of USS memory consumed by the program
memoryRSS [float] To store th	ne total amount of RSS memory consumed by the program
finalPatterns	to to store the patterns

Methods to execute code on terminal

100.

Format:

```
(.venv) $ python3 FAE.py <inputFile> <outputFile> <K>
```

Example Usage:

(.venv) \$ python3 FAE.py sampleDB.txt patterns.txt 10

Note: k will be considered as count of top frequent patterns to consider in analysis

Importing this algorithm into a python program

```
import PAMI.frequentPattern.topK.FAE as alg
obj = alg.FAE(iFile, K)
obj.mine()
topKFrequentPatterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(topKFrequentPatterns))
obj.save(oFile)
Df = obj.getPatternInDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
```

(continues on next page)

(continued from previous page)

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

noa

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

mine()

Main function of the program

printTOPK()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (file) – name of the output file

startMine()

Main function of the program

1.2 Relative Frequent Pattern

Relative Frequency is an extension of frequency where each frequency is represented relative to all the present frequencies of different quantities. Frequency in mathematics represents the actual occurrence of quantities whereas relative frequency represents the occurrence of quantities relative to each other. Suppose if we have a term with frequency f and the total frequency of all the observation is n, then the relative frequency of the given observation is f/n.

Application: Market Basket Analysis, Web Usage Mining, Network Intrusion Detection, Manufacturing and Supply Chain.

Relative Frequency is an extension of frequency where each frequency is represented relative to all the present frequencies of different quantities. Frequency in mathematics represents the actual occurrence of quantities whereas relative frequency represents the occurrence of quantities relative to each other. Suppose if we have a term with frequency f and the total frequency of all the observation is n, then the relative frequency of the given observation is f/n.

Application: Market Basket Analysis, Web Usage Mining, Network Intrusion Detection, Manufacturing and Supply Chain.

1.2.1 Basic

RSFPGrowth

class PAMI.relativeFrequentPattern.basic.RSFPGrowth.**RSFPGrowth**(*iFile: str* | *DataFrame, minSup: int* | *float* | *str, minRS: float, sep:* $str = {t'}$)

Bases: _frequentPatterns

Description

Algorithm to find all items with relative support from given dataset

Reference

'Towards Efficient Discovery of Frequent Patterns with Relative Support' R. Uday Kiran and Masaru Kitsuregawa, http://comad.in/comad2012/pdf/kiran.pdf

Parameters

- iFile str : Name of the Input file to mine complete set of Relative frequent pattern's
- oFile str : Name of the output file to store complete set of Relative frequent patterns
- **minSup** str: Controls the minimum number of transactions in which every item must appear in a database.
- **minRS** float: Controls the minimum number of transactions in which at least one time within a pattern must appear in a database.

• **sep** – str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file to mine complete set of frequent patterns

oFile

[file] Name of the output file to store complete set of frequent patterns

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

minSup

[float] The user given minSup

minRS

[float] The user given minRS

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

itemSetCount

[int] it represents the total no of patterns

finalPatterns

[dict] it represents to store the patterns

itemSetBuffer

[list] it represents the store the items in mining

maxPatternLength

[int] it represents the constraint for pattern length

Methods

startMine()

Mining process will start from here

getFrequentPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getmemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

check(line)

To check the delimiter used in the user input file

creatingItemSets(fileName)

Scans the dataset or dataframes and stores in list format

frequentOneItem()

Extracts the one-frequent patterns from transactions

saveAllCombination(tempBuffer,s,position,prefix,prefixLength)

Forms all the combinations between prefix and tempBuffer lists with support(s)

saveItemSet(pattern,support)

Stores all the frequent patterns with their respective support

frequentPatternGrowthGenerate(frequentPatternTree,prefix,port)

Mining the frequent patterns by forming conditional frequentPatternTrees to particular prefix item. ____mapSupport represents the 1-length items with their respective support

Methods to execute code on terminal

Format:

(.venv) \$python3 RSFPGrowth.py <inputFile> <outputFile> <minSup> <__minRatio>

Example Usage :

(.venv) \$python3 python3 RSFPGrowth.py sampleDB.txt patterns.txt 0.23 0.2

Importing this algorithm into a python program

```
from PAMI.relativeFrequentPattern import RSFPGrowth as alg
obj = alg.RSFPGrowth(iFile, minSup, __minRatio)
obj.startMine()
frequentPatterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(frequentPatterns))
obj.save(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getmemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by Sai Chitra.B under the supervision of Professor Rage Uday Kiran.

$Mine() \rightarrow None$

Main program to start the operation :return: None

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \rightarrow \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns() \rightarrow Dict[str, str]

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

$getPatternsAsDataFrame() \rightarrow DataFrame$

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

 $printResults() \rightarrow None$

This function is used to print the results :return: None

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (*file*) – name of the output file.

Returns None

$startMine() \rightarrow None$

Main program to start the operation :return: None

1.3 Frequent pattern With Multiple Minimum Support

Frequent patterns with multiple minimum support refer to patterns in a dataset that occur frequently and meet multiple minimum support thresholds. Unlike traditional frequent pattern mining, which uses a single uniform minimum support threshold for all items, this approach considers varying levels of support for different items in the dataset. By using multiple minimum support thresholds, it allows for a more nuanced analysis, where the significance of each item is evaluated individually based on its characteristics and importance in the context of the dataset.

Applications: Network Traffic Analysis, Manufacturing Process Optimization, Healthcare Data Analysis, Retail Market Analysis.

Frequent patterns with multiple minimum support refer to patterns in a dataset that occur frequently and meet multiple minimum support thresholds. Unlike traditional frequent pattern mining, which uses a single uniform minimum support threshold for all items, this approach considers varying levels of support for different items in the dataset. By using multiple minimum support thresholds, it allows for a more nuanced analysis, where the significance of each item is evaluated individually based on its characteristics and importance in the context of the dataset.

Applications: Network Traffic Analysis, Manufacturing Process Optimization, Healthcare Data Analysis, Retail Market Analysis.

1.3.1 Basic

CFPGrowth

class PAMI.multipleMinimumSupportBasedFrequentPattern.basic.CFPGrowth.CFPGrowth(*iFile*, MIS,

sep = (t')

Bases: _frequentPatterns

Description

basic is one of the fundamental algorithm to discover frequent patterns based on multiple minimum support in a transactional database.

Reference

Ya-Han Hu and Yen-Liang Chen. 2006. Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism. Decis. Support Syst. 42, 1 (October 2006), 1–24. https://doi.org/10.1016/j.dss.2004.09.007

Parameters

- **iFile** str : Name of the Input file to mine complete set of Uncertain Minimum Support Based Frequent patterns
- **oFile** str : Name of the output file to store complete set of Uncertain Minimum Support Based Frequent patterns
- **minSup** str: minimum support thresholds were tuned to find the appropriate ranges in the limited memory
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Input file name or path of the input file

MIS: file or dictionary

Multiple minimum supports of all items in the database

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

oFile

[file] Name of the output file or the path of the output file

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

finalPatterns

[dict] it represents to store the patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to an output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets()

Scans the dataset or dataframes and stores in list format

frequentOneItem()

Extracts the one-frequent patterns from transactions

Executing the code on terminal:

```
Format:
(.venv) $ python3 CFPGrowth.py <inputFile> <outputFile>
Examples:
(.venv) $ python3 CFPGrowth.py sampleDB.txt patterns.txt MISFile.txt
```

.. note:: minSup will be considered in support count or frequency

Sample run of the importing code:

```
from PAMI.multipleMinimumSupportBasedFrequentPattern.basic import basic as alg
obj = alg.basic(iFile, mIS)
obj.startMine()
frequentPatterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(frequentPatterns))
obj.save(oFile)
Df = obj.getPatternInDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$Mine() \rightarrow None$

main program to start the operation :return: none

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float
getPatterns() \rightarrow Dict[str, int]

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

noat

$\texttt{printResults()} \rightarrow None$

this function is used to print the results :return: None

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (file) – name of the output file

Returns None

$startMine() \rightarrow None$

main program to start the operation :return: none

CFPGrowthPlus

class PAMI.multipleMinimumSupportBasedFrequentPattern.basic.CFPGrowthPlus.CFPGrowthPlus(*iFile*,

 $MIS, \\ sep = \t')$

Bases: _frequentPatterns

Description

Reference

R. Uday Kiran P. Krishna Reddy Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms. 11-20 2011 EDBT https://doi.org/10.1145/ 1951365.1951370

Parameters

• **iFile** – str : Name of the Input file to mine complete set of Uncertain Multiple Minimum Support Based Frequent patterns

- **oFile** str : Name of the output file to store complete set of Uncertain Minimum Support Based Frequent patterns
- **minSup** str: minimum support thresholds were tuned to find the appropriate ranges in the limited memory
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Input file name or path of the input file

MIS: file or dictionary

Multiple minimum supports of all items in the database

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

oFile

[file] Name of the output file or the path of the output file

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

finalPatterns

[dict] it represents to store the patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

savePatterns(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets()

Scans the dataset or dataframes and stores in list format

frequentOneItem()

Extracts the one-frequent patterns from transactions

Executing the code on terminal:

```
Format:
```

```
(.venv) $ python3 CFPGrowthPlus.py <inputFile> <outputFile>
```

Examples:

(.venv) \$ python3 CFPGrowthPlus.py sampleDB.txt patterns.txt MISFile.txt

.. note:: minSup will be considered in support count or frequency

Sample run of the importing code:

```
from PAMI.multipleMinimumSupportBasedFrequentPattern.basic import CFPGrowthPlus as_
    alg
    obj = alg.CFPGrowthPlus(iFile, mIS)
    obj.startMine()
    frequentPatterns = obj.getPatterns()
    print("Total number of Frequent Patterns:", len(frequentPatterns))
    obj.savePatterns(oFile)
    Df = obj.getPatternInDataFrame()
    memUSS = obj.getMemoryUSS()
    print("Total Memory in USS:", memUSS)
    memRSS = obj.getMemoryRSS()
```

(continued from previous page)

```
print("Total Memory in RSS", memRSS)
```

run = obj.getRuntime()

```
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

Mine()

main program to start the operation

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

$printResults() \rightarrow None$

this function is used to print the results :return: None

save(outFile)

Complete set of frequent patterns will be loaded in to a output file

Parameters outFile (file) – name of the output file

startMine()

main program to start the operation

1.4 Correlated Pattern Mining

Correlated patterns are specific types of regularities or associations that exist within a dataset, where the occurrence of certain items or attributes is statistically correlated with the occurrence of other items or attributes. These patterns represent meaningful relationships or dependencies between different sets of items or attributes, and their discovery can provide valuable insights into the underlying structure and behavior of the data.

Applications: Fraud Detection, Supply Chain Management, Healthcare Data Analysis, Retail Market Analysis.

Correlated patterns are specific types of regularities or associations that exist within a dataset, where the occurrence of certain items or attributes is statistically correlated with the occurrence of other items or attributes. These patterns represent meaningful relationships or dependencies between different sets of items or attributes, and their discovery can provide valuable insights into the underlying structure and behavior of the data.

Applications: Fraud Detection, Supply Chain Management, Healthcare Data Analysis, Retail Market Analysis.

1.4.1 Basic

CoMine

class PAMI.correlatedPattern.basic.CoMine.**CoMine**(*iFile: str* | *DataFrame*, *minSup: int* | *float* | *str*,

minAllConf: float, sep: $str = \langle t' \rangle$

Bases: _correlatedPatterns

About this algorithm

Description

CoMine is one of the fundamental algorithm to discover correlated patterns in a transactional database. It is based on the traditional FP-Growth algorithm. This algorithm uses depth-first search technique to find all correlated patterns in a transactional database.

Reference

Lee, Y.K., Kim, W.Y., Cao, D., Han, J. (2003). CoMine: efficient mining of correlated patterns. In ICDM (pp. 581–584).

parameters

iFile (str) – Name of the Input file to mine complete set of correlated patterns oFile (str) – Name of the output file to store complete set of correlated patterns minSup (*int or float or* str) – The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. minAllConf (*float*) – The user can specify minAllConf values within the range (0, 1). sep (str) – This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

memoryUSS (float) – To store the total amount of USS memory consumed by the program memoryRSS (float) – To store the total amount of RSS memory consumed by the program startTime (float) – To record the start time of the mining process endTime (float) – To record the completion time of the mining process minSup (int) – The user given minSup minAll-Conf (float) – The user given minimum all confidence Ratio(should be in range of 0 to 1) Database (list) – To store the transactions of a database in list mapSupport (*Dictionary*) – To maintain the information of item and their frequency lno (int) – it represents the total no of transactions tree (class) – it represents the Tree class itemSetCount (int) – it represents the total no of patterns finalPatterns (dict) – it represents to store the patterns itemSetBuffer (list) – it represents the store the items in mining maxPatternLength (int) – it represents the constraint for pattern length

Execution methods

Terminal command

Format:
(.venv) \$ python3 CoMine.py <inputFile> <outputFile> <minSup> <minAllConf> <sep>
Example Usage:

(.venv) \$ python3 CoMine.py sampleTDB.txt output.txt 0.25 0.2

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

```
from PAMI.correlatedPattern.basic import CoMine as alg
iFile = 'sampleTDB.txt'
minSup = 0.25 # can be specified between 0 and 1
minAllConf = 0.2 # can be specified between 0 and 1
obj = alg.CoMine(iFile, minSup, minAllConf,sep)
obj.mine()
patterns = obj.getPatterns()
print("Total number of Patterns:", len(patterns))
obj.savePatterns(oFile)
```

(continues on next page)

(continued from previous page)

df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)

Credits

The complete program was written by B.Sai Chitra under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$\texttt{getMemoryUSS()} \rightarrow \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$\texttt{getPatterns}() \rightarrow Dict[Tuple[int], List[int | float]]$

Function to send the set of correlated patterns after completion of the mining process

Returns

returning correlated patterns

Return type

dict

$\texttt{getPatternsAsDataFrame()} \rightarrow \texttt{DataFrame}$

Storing final correlated patterns in a dataframe

Returns

returning correlated patterns in a dataframe

Return type

pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

mine() \rightarrow None

main method to start

$\texttt{printResults()} \rightarrow \text{None}$

function to print the result after completing the process

Returns

None

$save(outFile) \rightarrow None$

Complete set of correlated patterns will be saved into an output file

Parameters outFile (file) – name of the outputfile

Returns None

 $\texttt{startMine}() \to \text{None}$

main method to start

CoMinePlus

class PAMI.correlatedPattern.basic.CoMinePlus.**CoMinePlus**(*iFile: str* | *DataFrame, minSup: int* | *float* | *str, minAllConf: str, sep: str* = `\t')

Bases: _correlatedPatterns

About this algorithm

Description

CoMinePlus is one of the efficient algorithm to discover correlated patterns in a transactional database. Using Item Support Intervals technique which is generating correlated patterns of higher order by combining only with items that have support within specified interval.

Reference

Uday Kiran R., Kitsuregawa M. (2012) Efficient Discovery of Correlated Patterns in Transactional Databases Using Items' Support Intervals. In: Liddle S.W., Schewe KD., Tjoa A.M., Zhou X. (eds) Database and Expert Systems Applications. DEXA 2012. Lecture Notes in Computer Science, vol 7446. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-32600-4_18

:parameters iFile (str) - Name of the Input file to mine complete set of correlated patterns

:oFile (str) – Name of the output file to store complete set of correlated patterns :minSup (*int or float* or str) – The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. :minAllConf (str) – Name of Neighbourhood file name :sep (str) – This variable is used to distinguish items from

one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

memoryUSS (float) – To store the total amount of USS memory consumed by the program memoryRSS (float) – To store the total amount of RSS memory consumed by the program startTime (float) – To record the start time of the mining process endTime (float) – To record the completion time of the mining process minSup (float) – The user given minSup minAll-Conf (float) – The user given minimum all confidence Ratio (should be in range of 0 to 1) Database (list) – To store the transactions of a database in list mapSupport (*Dictionary*) – To maintain the information of item and their frequency lno (int) – it represents the total no of transactions tree (class) – it represents the Tree class itemSetCount (int) – it represents the total no of patterns finalPatterns (dict) – it represents to store the patterns itemSetBuffer (list) – it represents the store the items in mining maxPatternLength (int) – it represents the constraint for pattern length

Execution methods

Terminal command

Format:
(.venv) \$ python3 CoMinePlus.py <inputFile> <outputFile> <minSup> <minAllConf> <sep>
Example Usage:
(.venv) \$ python3 CoMinePlus.py sampleTDB.txt patterns.txt 0.4 0.5 ','

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

```
from PAMI.correlatedPattern.basic import CoMinePlus as alg
obj = alg.CoMinePlus(iFile, minSup, minAllConf, sep)
obj.mine()
correlatedPatterns = obj.getPatterns()
print("Total number of correlated patterns:", len(correlatedPatterns))
obj.save(oFile)
df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
(continues on next page)
```

(continued from previous page)

```
print("Total Memory in RSS", memRSS)
```

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)

Credits

The complete program was written by B.Sai Chitra under the supervision of Professor Rage Uday Kiran.

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS() \rightarrow float

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns() \rightarrow Dict[Tuple[str], List[int | float]]

Function to send the set of correlated patterns after completion of the mining process

Returns

returning correlated patterns

Return type

dict

$\texttt{getPatternsAsDataFrame()} \rightarrow \texttt{DataFrame}$

Storing final correlated patterns in a dataframe

Returns

returning correlated patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None Main program to start the operation

 $printResults() \rightarrow None$

function to print the result after completing the process

save(*outFile: str*) \rightarrow None

Complete set of correlated patterns will be loaded in to an output file

Parameters outFile (csv file) – name of the output file

Returns None

 $startMine() \rightarrow None$

Main program to start the operation

1.5 Fault-Tolerant Frequent Pattern Mining

Fault-tolerant frequent pattern mining is a data mining approach aimed at discovering frequent patterns in large datasets containing both certain and uncertain records. Unlike traditional frequent pattern mining, which relies on exact matching based on support and confidence values, fault-tolerant mining employs approximate matching techniques to find patterns, thereby accommodating errors, missing information, or changes in the data. This approach allows for the discovery of frequent patterns even in the presence of uncertainties or faults in the dataset.

Applications: Geo-spatial Data Analysis, Remote Sensing Image Analysis, Weather Forecasting.

Fault-tolerant frequent pattern mining is a data mining approach aimed at discovering frequent patterns in large datasets containing both certain and uncertain records. Unlike traditional frequent pattern mining, which relies on exact matching based on support and confidence values, fault-tolerant mining employs approximate matching techniques to find patterns, thereby accommodating errors, missing information, or changes in the data. This approach allows for the discovery of frequent patterns even in the presence of uncertainties or faults in the dataset.

Applications: Geo-spatial Data Analysis, Remote Sensing Image Analysis, Weather Forecasting.

1.5.1 Basic

FTApriori

Bases: _faultTolerantFrequentPatterns

Description

FT-Apriori is one of the fundamental algorithm to discover fault-tolerant frequent patterns in a transactional database. This program employs apriori property (or downward closure property) to reduce the search space effectively.

Reference

Pei, Jian & Tung, Anthony & Han, Jiawei. (2001). Fault-Tolerant Frequent Pattern Mining: Problems and Challenges.

Parameters

- iFile str : Name of the Input file to mine complete set of fault Tolerant frequent patterns
- oFile str : Name of the output file to store complete set of falut Tolerant frequent patterns
- **minSup** float or int or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float
- **itemSup** int or float : Frequency of an item
- minLength int : minimum length of a pattern
- faultTolerance int
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

finalPatterns

[dict] Storing the complete set of patterns in a dictionary variable

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the transactions of a database in list

Methods to execute code on terminal

Format:

Example Usage:

(.venv) \$ python3 FTApriori.py sampleDB.txt patterns.txt 10.0 3.0 3 1

Note: minSup will be considered in times of minSup and count of database transactions

Importing this algorithm into a python program

```
from PAMI.faultTolerantFrequentPattern.basic import FTApriori as alg
obj = alg.FTApriori(inputFile,minSup,itemSup,minLength,faultTolerance)
obj.mine()
patterns = obj.getPatterns()
print("Total number of fault-tolerant frequent patterns:", len(patterns))
obj.save("outputFile")
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \rightarrow \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns() \rightarrow Dict[Tuple[str, ...], int]

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None

Fault-tolerant frequent pattern mining process will start from here

$printResults() \rightarrow None$

This is function is used to print the result

$save(outFile) \rightarrow None$

Complete set of frequent patterns will be loaded in to an output file

Parameters
 outFile (csvfile) - name of the output file

Returns None

$startMine() \rightarrow None$

Fault-tolerant frequent pattern mining process will start from here

FTFPGrowth

class PAMI.faultTolerantFrequentPattern.basic.FTFPGrowth.**FTFPGrowth**(*iFile: str* | *DataFrame*,

minSup: int | float | str, itemSup: float, minLength: int, faultTolerance: int, sep: str = \t')

Bases: _faultTolerantFrequentPatterns

Description

FPGrowth is one of the fundamental algorithm to discover frequent patterns in a transactional database. It stores the database in compressed fp-tree decreasing the memory usage and extracts the patterns from tree. It employs downward closure property to reduce the search space effectively.

Reference

Han, J., Pei, J., Yin, Y. et al. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. Data Mining and Knowledge Discovery 8, 53–87 (2004). https://doi.org/10.1023

Parameters

- **iFile** file : Name of the Input file to mine complete set of fault Tolerant frequent patterns
- oFile str : Name of the output file to store complete set of falut Tolerant frequent patterns
- **minSup** float or int or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

:param sep

[str :] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

Attributes

startTime: float :

To record the start time of the mining process

endTime: float :

To record the completion time of the mining process

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

finalPatterns

[dict] it represents to store the patterns

Methods

mine() Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to an output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets()

Scans the dataset or dataframes and stores in list format

frequentOneItem()

Extracts the one-frequent patterns from transactions

Executing the code on terminal:

```
Format:
```

(.venv) \$ python3 FPGrowth.py <inputFile> <outputFile> <minSup>

Example Usage:

(.venv) \$ python3 FPGrowth.py sampleDB.txt patterns.txt 10.0

Note: minSup will be considered in times of minSup and count of database transactions

Sample run of the importing code:

```
from PAMI.faultTolerantFrequentPattern.basic import FTFPGrowth as alg
obj = alg.FTFPGrowth(inputFile,minSup,itemSup,minLength,faultTolerance)
obj.mine()
patterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(patterns))
obj.save(oFile)
Df = obj.getPatternInDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
(continues on next page)
```

(continued from previous page)

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \to \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns() \rightarrow Dict[str, int]

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

$mine() \rightarrow None$

Main program to start the operation

$printResults() \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (csv file) – name of the output file

Returns None

 $startMine() \rightarrow None$

Main program to start the operation

1.6 Coverage Pattern Mining

Coverage pattern mining is a data mining technique focused on identifying patterns within a dataset that cover a substantial portion of the data, irrespective of their frequency of occurrence. Unlike traditional frequent pattern mining, which prioritizes patterns with high frequency, coverage pattern mining emphasizes patterns that have wide coverage across the dataset. These patterns are considered significant as they provide insights into the overall characteristics and trends present in the data. where understanding patterns that have broad coverage can inform decision-making processes, optimize operations, and improve overall efficiency and effectiveness.

Applications: Retail, Healthcare, Web Usage, Manufacturing, and Social Network Analysis.

Coverage pattern mining is a data mining technique focused on identifying patterns within a dataset that cover a substantial portion of the data, irrespective of their frequency of occurrence. Unlike traditional frequent pattern mining, which prioritizes patterns with high frequency, coverage pattern mining emphasizes patterns that have wide coverage across the dataset. These patterns are considered significant as they provide insights into the overall characteristics and trends present in the data. where understanding patterns that have broad coverage can inform decision-making processes, optimize operations, and improve overall efficiency and effectiveness.

Applications: Retail, Healthcare, Web Usage, Manufacturing, and Social Network Analysis.

1.6.1 Basic

CMine

class PAMI.coveragePattern.basic.CMine.CMine(iFile, minRF, minCS, maxOR, sep=\t')
Bases: _coveragePatterns

About this algorithm

Description

CMine algorithms aims to discover the coverage patterns in transactional databases.

Reference

Bhargav Sripada, Polepalli Krishna Reddy, Rage Uday Kiran: Coverage patterns for efficient banner advertisement placement. WWW (Companion Volume) 2011: 131-132 __https://dl.acm.org/doi/10.1145/1963192.1963259

param iFile

str : Name of the Input file to mine complete set of coverage patterns

param oFile

str : Name of the output file to store complete set of coverage patterns

param minRF

str: Controls the minimum number of transactions in which every item must appear in a database.

param minCS

str: Controls the minimum number of transactions in which at least one time within a pattern must appear in a database.

param maxOR

str: Controls the maximum number of transactions in which any two items within a pattern can reappear.

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

finalPatterns

[dict] Storing the complete set of patterns in a dictionary variable

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the transactions of a database in list

Execution methods

Terminal command

Format:

```
(.venv) $ python3 CMine.py <inputFile> <outputFile> <minRF> <minCS> <maxOR> <' '>
```

```
Example Usage:
```

```
(.venv) $ python3 CMine.py sampleTDB.txt patterns.txt 0.4 0.7 0.5 '
```

Calling from a python program

```
from PAMI.coveragePattern.basic import CMine as alg
obj = alg.CMine(iFile, minRF, minCS, maxOR, seperator)
obj.mine()
coveragePattern = obj.getPatterns()
print("Total number of coverage Patterns:", len(coveragePattern))
```

(continues on next page)

(continued from previous page)

```
obj.save(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$\textbf{creatingCoverageItems()} \rightarrow Dict[str, List[str]]$

This function creates coverage items from _database.

Returns

coverageTidData that stores coverage items and their tid list.

Return type

dict

genPatterns(prefix: Tuple[str, int], tidData: List[Tuple[str, int]]) \rightarrow None

This function generate coverage pattern about prefix.

Parameters

- prefix String
- tidData list

Returns

None

generateAllPatterns(coverageItems: Dict[str, int]) \rightarrow None

This function generates all coverage patterns.

Parameters

coverageItems - coverage items

Returns None

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$\texttt{getPatterns()} \rightarrow Dict[str, int]$

Function to send the set of coverage patterns after completion of the mining process

Returns

returning coverage patterns

Return type dict

$getPatternsAsDataFrame() \rightarrow DataFrame$

Storing final coverage patterns in a dataframe

Returns

returning coverage patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None

Main method to start

$printResults() \rightarrow None$

This function is used to print the result

save(*outFile: str*) \rightarrow None

Complete set of coverage patterns will be loaded in to an output file

Parameters

outFile (file) - name of the outputfile

Returns None

 $startMine() \rightarrow None$

Main method to start

tidToBitset(*item_set: Dict[str, int]*) → Dict[str, int]

This function converts tid list to bitset.

Parameters item_set -

Returns

Dictionary

Return type dict

CPPG

class PAMI.coveragePattern.basic.CPPG(*iFile*, *minRF*, *minCS*, *maxOR*, *sep*=\t')

Bases: _coveragePatterns

Description

CPPG algorithm discovers coverage patterns in a transactional database.

Reference

Gowtham Srinivas, P.; Krishna Reddy, P.; Trinath, A. V.; Bhargav, S.; Uday Kiran, R. (2015). Mining coverage patterns from transactional databases. Journal of Intelligent Information Systems, 45(3), 423–439. https://link.springer.com/article/10.1007/s10844-014-0318-3

Parameters

- iFile str : Name of the Input file to mine complete set of coverage patterns
- oFile str : Name of the output file to store complete set of coverage patterns
- **minRF** str: Controls the minimum number of transactions in which every item must appear in a database.
- **minCS** str: Controls the minimum number of transactions in which at least one time within a pattern must appear in a database.
- **maxOR** str: Controls the maximum number of transactions in which any two items within a pattern can reappear.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

finalPatterns

[dict] Storing the complete set of patterns in a dictionary variable

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the transactions of a database in list

'>

Methods to execute code on terminal

Format:

(.venv) \$ python3 CPPG.py <inputFile> <outputFile> <minRF> <minCS> <maxOR> <'

Example Usage:

(.venv) \$ python3 CPPG.py sampleTDB.txt patterns.txt 0.4 0.7 0.5 ','

Note: minSup will be considered in percentage of database transactions

Importing this algorithm into a python program

<pre>from PAMI.coveragePattern.basic import CPPG as alg</pre>
<pre>obj = alg.CPPG(iFile, minRF, minCS, maxOR)</pre>
obj.mine()
<pre>coveragePattern = obj.getPatterns()</pre>
<pre>print("Total number of coverage Patterns:", len(coveragePattern))</pre>
obj.save(oFile)
<pre>Df = obj.getPatternsAsDataFrame()</pre>
<pre>memUSS = obj.getMemoryUSS()</pre>
<pre>print("Total Memory in USS:", memUSS)</pre>
<pre>memRSS = obj.getMemoryRSS()</pre>
<pre>print("Total Memory in RSS", memRSS)</pre>
<pre>run = obj.getRuntime()</pre>
<pre>print("Total ExecutionTime in seconds:", run)</pre>

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \rightarrow \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$\texttt{getPatterns}() \rightarrow Dict[str, List[int]]$

Function to send the set of periodic-frequent patterns after completion of the mining process

Returns

returning periodic-frequent patterns

Return type

dict

$\texttt{getPatternsAsDataFrame()} \rightarrow \texttt{DataFrame}$

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None

Mining process will start from this function

$\texttt{printResults()} \rightarrow None$

Function used to print the result

save(*outFile: str*) \rightarrow None

Complete set of periodic-frequent patterns will be loaded in to an output file

Parameters outFile (file) – name of the outputfile

$\texttt{startMine}() \rightarrow \text{None}$

Mining process will start from this function

CHAPTER

TEMPORAL DATABASE

A temporal database is a collection of transactions ordered by their timestamps. A sample temporal database generated from the set of items, $I=\{a,b,c,d,e,f\}$, is shown in below table:

TID	Timestamp	Transactions
1	1	a, b, c
2	2	d, e
3	4	a, e, f
4	7	d, f, g

Types of temporal databases:

- Regular temporal database: Uniform time gap exists between any two transactions.
- Irregular temporal database: Non-uniform time gap exists between any two transactions.
 - Type-1 irregular temporal database: Every transaction will have a distinct timestamp.
 - Type-2 irregular temporal database: Multiple transactions can have a common timestamp.

Rules to create a temporal database:

- Since TID of a transaction implicitly represents the row number, this information can be ignored to save space.
- The first column in the database must represent a timestamp.
- The timestamp of the first transaction must always start from 1. The timestamps of remaining transactions follow thereafter. In other words, the timestamps in a temporal database must be relative to each other, rather than being absolute timestamps.
- Irregular time gaps can exist between the transactions.
- Multiple transactions can have a same timestamp. In other words, multiple transactions can occur at a particular timestamp. (Please note that some pattern mining algorithms, especially variants of ECLAT, may not work properly if multiple transactions share a common timestamp.)
- All items in a transaction must be seperated with a separator.
- The items in a temporal database can be integers or strings.
- 'Tab space' is the default seperator. However, temporal databases can be constructed using other seperators, such as comma and space.

Format of a temporal database:

>>> timestamp<sep>item1<sep>item2<sep>...<sep>itemN

Examples:

• Regular temporal database: Uniform time gap exists between the transactions.

1 a b c 2 d e 4 a e f 7 d f g

- Irregular temporal database (Type-1): Non-uniform time gap exists between the transactions. More important, every transaction contains a unique timestamp.
 - 1 abc 2 de 4 aef 7 dfg
- Irregular temporal database (Type-2): Non-uniform time gap exists between the transactions. More important, multiple transactions can have same timestamps.
 - 1 a b c 1 d e 4 a e f 8 d f g

2.1 Periodic Frequent Pattern Mining

Periodic frequent pattern mining involves identifying patterns that occur at regular intervals within a temporal database, where each record represents an event or observation associated with a specific timestamp. In this context, a pattern is considered periodic-frequent if it satisfies user-defined constraints on both the minimum support (minSup) and maximum periodicity (maxPer). The goal is to discover patterns that exhibit regular recurring behavior over time, providing insights into temporal trends, cyclic phenomena, or periodic events within the dataset. Unlike traditional frequent pattern mining, which focuses on static datasets, periodic frequent pattern mining specifically targets temporal databases, where time-related attributes play a crucial role in pattern discovery and analysis.

Applications: Temporal Data Analysis, Healthcare Monitoring, Retail Sales Forecasting, Network Traffic Analysis.

Periodic frequent pattern mining involves identifying patterns that occur at regular intervals within a temporal database, where each record represents an event or observation associated with a specific timestamp. In this context, a pattern is considered periodic-frequent if it satisfies user-defined constraints on both the minimum support (minSup) and maximum periodicity (maxPer). The goal is to discover patterns that exhibit regular recurring behavior over time, providing insights into temporal trends, cyclic phenomena, or periodic events within the dataset. Unlike traditional frequent pattern mining, which focuses on static datasets, periodic frequent pattern mining specifically targets temporal databases, where time-related attributes play a crucial role in pattern discovery and analysis.

Applications: Temporal Data Analysis, Healthcare Monitoring, Retail Sales Forecasting, Network Traffic Analysis.

2.1.1 Basic

PFPGrowth

class PAMI.periodicFrequentPattern.basic.PFPGrowth.**PFPGrowth**(*iFile*, *minSup*, *maxPer*, *sep*=\t') Bases: _periodicFrequentPatterns

Description

PFPGrowth is one of the fundamental algorithm to discover periodic-frequent patterns in a transactional database.

Reference

Syed Khairuzzaman Tanbeer, Chowdhury Farhan, Byeong-Soo Jeong, and Young-Koo Lee, "Discovering Periodic-Frequent Patterns in Transactional Databases", PAKDD 2009, https://doi.org/10.1007/978-3-642-01307-2_24

Parameters

- iFile str : Name of the Input file to mine complete set of periodic frequent pattern's
- oFile str : Name of the output file to store complete set of periodic frequent pattern's
- **minSup** str: Controls the minimum number of transactions in which every item must appear in a database.
- **maxPer** float: Controls the maximum number of transactions in which any two items within a pattern can reappear.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup

[int or float or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

maxPer

[int or float or str] The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

PeriodicFrequentOneItem()

Extracts the one-periodic-frequent patterns from database

updateDatabases()

Update the database by removing aperiodic items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

to convert the user specified value

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$Mine() \rightarrow None$

Mining process will start from this function :return: None

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$\texttt{getPatterns()} \rightarrow Dict[str, Tuple[int, int]]$

Function to send the set of periodic-frequent patterns after completion of the mining process

Returns

returning periodic-frequent patterns

Return type

dict

$getPatternsAsDataFrame() \rightarrow DataFrame$

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

$printResults() \rightarrow None$

This function is used to print the results :return: None

save(*outFile: str*) \rightarrow None

Complete set of periodic-frequent patterns will be loaded in to an output file

Parameters

outFile (csv file) - name of the output file

Returns None

$startMine() \rightarrow None$

Mining process will start from this function :return: None

PFPGrowthPlus

class PAMI.periodicFrequentPattern.basic.PFPGrowthPlus.PFPGrowthPlus(*iFile*, *minSup*, *maxPer*,

sep=(t')

Bases: _periodicFrequentPatterns

Description

PFPGrowthPlus is fundamental and improved version of PFPGrowth algorithm to discover periodic-frequent patterns in temporal database. It uses greedy approach to discover effectively

Reference

R. UdayKiran, MasaruKitsuregawa, and P. KrishnaReddyd, "Efficient discovery of periodic-frequent patterns in very large databases," Journal of Systems and Software February 2016 https://doi.org/10.1016/j.jss.2015.10.035

param iFile

str : Name of the Input file to mine complete set of periodic frequent pattern's

param oFile

str : Name of the output file to store complete set of periodic frequent pattern's

param minSup

str: Controls the minimum number of transactions in which every item must appear in a database.

param maxPer

str: Controls the maximum number of transactions in which any two items within a pattern can reappear.

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup

[int or float or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

maxPer

[int or float or str] The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transaction

tree

[class] it represents the Tree class

itemSetCount

[int] it represents the total no of patterns

finalPatterns

[dict] it represents to store the patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

check(line)

To check the delimiter used in the user input file

creatingItemSets(fileName)

Scans the dataset or dataframes and stores in list format

PeriodicFrequentOneItem()

Extracts the one-periodic-frequent patterns from Databases

updateDatabases()

update the Databases by removing aperiodic items and sort the Database by item decreased support

buildTree()

after updating the Databases ar added into the tree by setting root node as null

startMine()

the main method to run the program

Methods to execute code on terminal

(.venv) \$ python3 PFPGrowthPlus.py <inputFile> <outputFile> <minSup> <maxPer>

Example:

Format:

(.venv) \$ python3 PFPGrowthPlus.py sampleTDB.txt patterns.txt 0.3 0.4

.. note:: minSup will be considered in percentage of database transactions

Importing this algorithm into a python program

(continued from previous page)

```
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS() \rightarrow float

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns() \rightarrow Dict[str, Tuple[int, int]]

Function to send the set of periodic-frequent patterns after completion of the mining process

Returns

returning periodic-frequent patterns

Return type

dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

$printResults() \rightarrow None$

This function is used to print the results :return: None

save(*outFile: str*) \rightarrow None

Complete set of periodic-frequent patterns will be loaded in to an output file

Parameters
 outFile (csv file) - name of the output file

Returns None

$\texttt{startMine()} \rightarrow \text{None}$

Main method where the patterns are mined by constructing tree. :return: None

PSGrowth

class PAMI.periodicFrequentPattern.basic.PSGrowth.Node(item, children)
Bases: object

Bases: object

A class used to represent the node of frequentPatternTree

Attributes

item

[int] storing item of a node

timeStamps

[list] To maintain the timeStamps of Database at the end of the branch

parent

[node] To maintain the parent of every node

children

[list] To maintain the children of node

Methods

addChild(itemName)

storing the children to their respective parent nodes

$addChild(node) \rightarrow None$

Appends the children node details to a parent node

Parameters

 ${\color{blue}\textbf{node}}-{\color{blue}\textbf{children}}\ {\color{blue}\textbf{node}}$

Returns

appending children node to parent node

class PAMI.periodicFrequentPattern.basic.PSGrowth.**PSGrowth**(*iFile*, *minSup*, *maxPer*, *sep*=\t') Bases: _periodicFrequentPatterns

Description

PS-Growth is one of the fundamental algorithm to discover periodic-frequent patterns in a temporal database.

:Reference

[A. Anirudh, R. U. Kiran, P. K. Reddy and M. Kitsuregaway, "Memory efficient mining of periodic-frequent] patterns in transactional databases," 2016 IEEE Symposium Series on Computational Intelligence (SSCI), 2016, pp. 1-8, https://doi.org/10.1109/SSCI.2016.7849926
Parameters

- **iFile** str : Name of the Input file to mine complete set of periodic frequent pattern's
- oFile str : Name of the output file to store complete set of periodic frequent pattern's
- **minSup** str: Controls the minimum number of transactions in which every item must appear in a database.
- **maxPer** str: Controls the maximum number of transactions in which any two items within a pattern can reappear.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup: int or float or str

The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

maxPer: int or float or str

The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transaction

tree

[class] it represents the Tree class

itemSetCount

[int] it represents the total no of patterns

finalPatterns

[dict] it represents to store the patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to an output file

getConditionalPatternsInDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

OneLengthItems()

Scans the dataset or dataframes and stores in list format

buildTree()

after updating the Databases ar added into the tree by setting root node as null

Methods to execute code on terminal

Format:

(.venv) \$ python3 PSGrowth.py <inputFile> <outputFile> <minSup> <maxPer>

Example:

(.venv) \$ python3 PSGrowth.py sampleTDB.txt patterns.txt 0.3 0.4

```
.. note:: minSup will be considered in percentage of database \begin{smallmatrix} \b
```

Importing this algorithm into a python program

```
from PAMI.periodicFrequentPattern.basic import PSGrowth as alg
obj = alg.PSGrowth("../basic/sampleTDB.txt", "2", "6")
obj.startMine()
periodicFrequentPatterns = obj.getPatterns()
print("Total number of Patterns:", len(periodicFrequentPatterns))
obj.save("patterns")
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$Mine() \rightarrow None$

Mining process will start from this function :return: None

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \rightarrow \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns() \rightarrow dict

Function to send the set of periodic-frequent patterns after completion of the mining process

Returns

returning periodic-frequent patterns

Return type dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

$\texttt{printResults()} \rightarrow \text{None}$

This function is used to print the results :return: None

save(*outFile: str*) \rightarrow None

Complete set of periodic-frequent patterns will be loaded in to an output file

Parameters outFile (csv file) – name of the output file

Returns None

$startMine() \rightarrow None$

Mining process will start from this function :return: None

 $\texttt{PAMI.periodicFrequentPattern.basic.PSGrowth.conditionalTransactions(\textit{patterns, timestamp}) \rightarrow \texttt{PAMI.periodicFrequentPattern.basic.PSGrowth.conditionalTransactions(patterns, timestamp))} \rightarrow \texttt{PAMI.periodicFrequentPatterns, timestamp)} \rightarrow \texttt{PAMI.periodi$

Tuple[List[List[int]], List[List[_Interval]], Dict[int, Tuple[int, int]]]

To sort and update the conditional transactions by removing the items which fails frequency and periodicity conditions

Parameters

- patterns conditional patterns of a node
- timestamp timeStamps of a conditional pattern

Returns

conditional transactions with their respective timeStamps

PAMI.periodicFrequentPattern.basic.PSGrowth.getPeriodAndSupport(timeStamps) → List[int] Calculates the period and support of list of timeStamps

Parameters

timeStamps – timeStamps of a pattern or item

Returns

support and periodicity

PFECLAT

class PAMI.periodicFrequentPattern.basic.PFECLAT.**PFECLAT**(*iFile*, *minSup*, *maxPer*, *sep*=\t') Bases: _periodicFrequentPatterns

Description

PFECLAT is the fundamental approach to mine the periodic-frequent patterns.

Reference

P. Ravikumar, P.Likhitha, R. Uday kiran, Y. Watanobe, and Koji Zettsu, "Towards efficient discovery of periodic-frequent patterns in columnar temporal databases", 2021 IEA/AIE.

Parameters

- iFile str : Name of the Input file to mine complete set of periodic frequent pattern's
- oFile str : Name of the output file to store complete set of periodic frequent pattern's
- **minSup** str: Controls the minimum number of transactions in which every item must appear in a database.
- **maxPer** str: Controls the maximum number of transactions in which any two items within a pattern can reappear.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup

[int or float or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

maxPer

[int or float or str] The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

itemSetCount

[int] it represents the total no of patterns

finalPatterns

[dict] it represents to store the patterns

tidList

[dict] stores the timestamps of an item

hashing

[dict] stores the patterns with their support to check for the closed property

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingOneItemSets()

Scan the database and store the items with their timestamps which are periodic frequent

getPeriodAndSupport()

Calculates the support and period for a list of timestamps.

Generation()

Used to implement prefix class equivalence method to generate the periodic patterns recursively

Methods to execute code on terminal

```
Format:
(.venv) $ python3 PFECLAT.py <inputFile> <outputFile> <minSup>
Example usage:
(.venv) $ python3 PFECLAT.py sampleDB.txt patterns.txt 10.0
.. note:: minSup will be considered in percentage of database transactions
```

Importing this algorithm into a python program

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$Mine() \rightarrow None$

Mining process will start from this function :return: None

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \to \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

$\texttt{getPatterns()} \rightarrow \texttt{dict}$

Function to send the set of periodic-frequent patterns after completion of the mining process

Returns

returning periodic-frequent patterns

Return type

dict

$\texttt{getPatternsAsDataFrame()} \rightarrow \texttt{DataFrame}$

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

$\texttt{printResults()} \rightarrow \text{None}$

This function is used to print the results :return: None

save(*outFile: str*) \rightarrow None

Complete set of periodic-frequent patterns will be loaded in to a output file

Parameters outFile (*csv file*) – name of the output file

Returns None

$startMine() \rightarrow None$

Mining process will start from this function :return: None

PFPMC

class PAMI.periodicFrequentPattern.basic.PFPMC.**PFPMC**(*iFile*, *minSup*, *maxPer*, *sep*=\t') Bases: _periodicFrequentPatterns

Description

PFPMC is the fundamental approach to mine the periodic-frequent patterns.

Reference

(has to be added)

Parameters

- iFile str : Name of the Input file to mine complete set of periodic frequent pattern's
- oFile str : Name of the output file to store complete set of periodic frequent pattern's
- minSup str: Controls the minimum number of transactions in which every item must appear in a database.
- **maxPer** str: Controls the maximum number of transactions in which any two items within a pattern can reappear.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup

[int or float or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

maxPer

[int or float or str] The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

itemSetCount

[int] it represents the total no of patterns

finalPatterns

[dict] it represents to store the patterns

tidList

[dict] stores the timestamps of an item

hashing

[dict] stores the patterns with their support to check for the closed property

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to an output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingOneItemSets()

Scan the database and store the items with their timestamps which are periodic frequent

getPeriodAndSupport()

Calculates the support and period for a list of timestamps.

Generation()

Used to implement prefix class equivalence method to generate the periodic patterns recursively

Methods to execute code on terminal

Importing this algorithm into a python program

```
from PAMI.periodicFrequentPattern.basic import PFPMC as alg
obj = alg.PFPMC("../basic/sampleTDB.txt", "2", "5")
obj.startMine()
periodicFrequentPatterns = obj.getPatterns()
print("Total number of Periodic Frequent Patterns:", len(periodicFrequentPatterns))
obj.save("patterns")
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

 $getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns() \rightarrow dict

Function to send the set of periodic-frequent patterns after completion of the mining process

Returns

returning periodic-frequent patterns

Return type dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

$printResults() \rightarrow None$

This function is used to print the results :return: None

save(*outFile: str*) \rightarrow None

Complete set of periodic-frequent patterns will be loaded in to an output file

Parameters

outFile (csv file) – name of the output file

Returns

None

 $startMine() \rightarrow None$

Mining process will start from this function :return: None

2.1.2 closed

CPFPMiner

class PAMI.periodicFrequentPattern.closed.CPFPMiner.**CPFPMiner**(*iFile*, *minSup*, *maxPer*, *sep*=\t') Bases: _periodicFrequentPatterns

About this algorithm

Description

CPFPMiner algorithm is used to discover the closed periodic frequent patterns in temporal databases. It uses depth-first search.

Reference

P. Likhitha et al., "Discovering Closed Periodic-Frequent Patterns in Very Large Temporal Databases" 2020 IEEE International Conference on Big Data (Big Data), 2020, https://ieeexplore.ieee.org/document/9378215

param iFile

str : Name of the Input file to mine complete set of periodic frequent pattern's

param oFile

str : Name of the output file to store complete set of periodic frequent pattern's

param minSup

float: Controls the minimum number of transactions in which every item must appear in a database.

param maxPer

float: Controls the maximum number of transactions in which any two items within a pattern can reappear.

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

oFile

[str] Name of the output file or path of the input file

minSup: int or float or str

The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

maxPer: int or float or str

The user can specify maxPer either in count or proportion of database size. If the program

detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

finalPatterns: dict

Storing the complete set of patterns in a dictionary variable

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to an output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

Execution methods

Terminal command

Format:

(.venv) \$ python3 CPFPMiner.py <inputFile> <outputFile> <minSup> <maxPer>

Example:

(.venv) \$ python3 CPFPMiner.py sampleTDB.txt patterns.txt 0.3 0.4

(continues on next page)

(continued from previous page)

```
.. note:: minSup will be considered in percentage of database transactions
```

Calling from a python program

```
from PAMI.periodicFrequentPattern.closed import CPFPMiner as alg
obj = alg.CPFPMiner("../basic/sampleTDB.txt", "2", "6")
obj.startMine()
periodicFrequentPatterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(periodicFrequentPatterns))
obj.save("patterns")
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

Mine()

Mining process will start from here

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

Return type

returning frequent patterns in a dataframe

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

printResults()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (csv file) – name of the output file

startMine()

Mining process will start from here

2.1.3 maximal

MaxPFGrowth

Bases: _periodicFrequentPatterns

Description

MaxPF-Growth is one of the fundamental algorithm to discover maximal periodic-frequent patterns in a temporal database.

Reference

R. Uday Kiran, Yutaka Watanobe, Bhaskar Chaudhury, Koji Zettsu, Masashi Toyoda, Masaru Kitsuregawa, "Discovering Maximal Periodic-Frequent Patterns in Very Large Temporal Databases", IEEE 2020, https://ieeexplore.ieee.org/document/9260063

Parameters

- **iFile** str : Name of the Input file to mine complete set of periodic frequent pattern's
- oFile str : Name of the output file to store complete set of periodic frequent pattern's
- **minSup** str: Controls the minimum number of transactions in which every item must appear in a database.
- **maxPer** float: Controls the maximum number of transactions in which any two items within a pattern can reappear.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup: int or float or str

The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

maxPer: int or float or str

The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transaction

tree

[class] it represents the Tree class

itemSetCount

[int] it represents the total no of patterns

finalPatterns

[dict] it represents to store the patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset or dataframes and stores in list format

PeriodicFrequentOneItem()

Extracts the one-periodic-frequent patterns from Databases

updateDatabases()

update the Databases by removing aperiodic items and sort the Database by item decreased support

buildTree()

after updating the Databases ar added into the tree by setting root node as null

startMine()

the main method to run the program

Executing the code on terminal:

Format:

(.venv) \$ python3 maxpfrowth.py <inputFile> <outputFile> <minSup> <maxPer>

Examples usage :

(.venv) \$ python3 maxpfrowth.py sampleTDB.txt patterns.txt 0.3 0.4

.. note:: minSup will be considered in percentage of database transactions

Sample run of the imported code:

```
from PAMI.periodicFrequentPattern.maximal import MaxPFGrowth as alg
obj = alg.MaxPFGrowth("../basic/sampleTDB.txt", "2", "6")
obj.startMine()
Patterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(Patterns))
obj.save("patterns")
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

Mine() \rightarrow None

Mining process will start from this function :return: None

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \rightarrow \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$getPatterns() \rightarrow Dict[str, Tuple[int, int]]$

Function to send the set of periodic-frequent patterns after completion of the mining process

Returns

returning periodic-frequent patterns

Return type dict

$getPatternsAsDataFrame() \rightarrow DataFrame$

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

$printResults() \rightarrow None$

To print the results of the execution.

save(*outFile: str*) \rightarrow None

Complete set of periodic-frequent patterns will be loaded in to a output file

Parameters outFile (csv file) – name of the output file

Returns None

$startMine() \rightarrow None$

Mining process will start from this function :return: None

2.1.4 Top-K

kPFPMiner

class PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFPMiner.**kPFPMiner**(*iFile*, *k*, *sep*=\t') Bases: _periodicFrequentPatterns

Description

Top - K is and algorithm to discover top periodic-frequent patterns in a temporal database.

Reference

Likhitha, P., Ravikumar, P., Kiran, R.U., Watanobe, Y. (2022).

Discovering Top-k Periodic-Frequent Patterns in Very Large Temporal Databases. Big Data Analytics.

BDA 2022. Lecture Notes in Computer Science, vol 13773. Springer, Cham. https://doi.org/10. 1007/978-3-031-24094-2_14

Parameters

- **iFile** str : Name of the Input file to mine complete set of periodic frequent pattern's
- oFile str : Name of the output file to store complete set of periodic frequent pattern's
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

k: int

User specified counte of top-k periodic frequent patterns

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

oFile

[str] Name of the output file or the path of the output file

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

finalPatterns: dict

Storing the complete set of patterns in a dictionary variable

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Methods

startMine()
Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

savePatterns(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets()
 Scans the dataset or dataframes and stores in list format

frequentOneItem()

Generates one frequent patterns

eclatGeneration(candidateList) It will generate the combinations of frequent items

generateFrequentPatterns(tidList)

It will generate the combinations of frequent items from a list of items

Executing the code on terminal:

Format:

(.venv) \$ python3 kPFPMiner.py <inputFile> <outputFile> <k>

Examples :

(.venv) \$ python3 kPFPMiner.py sampleDB.txt patterns.txt 10

**Sample run of the importing code:

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getPer_Sup(tids)

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

lno = 0

printResults()

save(outFile)

Complete set of frequent patterns will be loaded in to a output file

Parameters

outFile (*file*) – name of the output file

startMine()

Main function of the program

TopkPFP

```
class PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP.TopkPFPGrowth(iFile, minSup, maxPer, sep=\t')
```

Bases: _periodicFrequentPatterns

Description

Top - K is and algorithm to discover top periodic frequent patterns in a temporal database.

Reference

Komate Amphawan, Philippe Lenca, Athasit Surarerks: "Mining Top-K Periodic-Frequent Pattern from Transactional Databases without Support Threshold" International Conference on Advances in Information Technology: https://link.springer.com/ chapter/10.1007/978-3-642-10392-6_3

param iFile

str : Name of the Input file to mine complete set of periodic frequent pattern's

param oFile

str : Name of the output file to store complete set of periodic frequent pattern's

param maxPer

str: Controls the maximum number of transactions in which any two items within a pattern can reappear.

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

k: int

User specified counte of top frequent patterns

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

oFile

[str] Name of the output file or the path of the output file

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

finalPatterns: dict

Storing the complete set of patterns in a dictionary variable

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets()

Scans the dataset or dataframes and stores in list format

frequentOneItem()

Generates one frequent patterns

eclatGeneration(candidateList)

It will generate the combinations of frequent items

generateFrequentPatterns (tidList)

It will generate the combinations of frequent items from a list of items

Format:

(.venv) \$ python3 TopkPFP.py <inputFile> <outputFile> <k> <maxPer>

Examples:

(.venv) \$ python3 TopkPFP.py sampleDB.txt patterns.txt 10 3

```
**Sample run of the importing code:**
```

```
-----
```

.. code-block:: python

import PAMI.periodicFrequentPattern.topk.TopkPFPGrowth as alg

```
obj = alg.TopkPFPGrowth(iFile, k, maxPer)
```

```
obj.startMine()
```

(continues on next page)

(continued from previous page)

Mine()

Main function of the program

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

noai

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

printResults()

To print the results of the execution.

save(outFile)

Complete set of frequent patterns will be loaded in to a output file

Parameters outFile (file) – name of the output file

startMine()

Main function of the program

2.2 Local Periodic Pattern Mining

Local Periodic Patterns (LPPs) which are patterns that have a periodic behavior in some non-predefined time-intervals. A pattern is said to be a local periodic pattern if it appears regularly and continuously in some time interval (s). A pattern is considered a local periodic pattern if it demonstrates periodic behavior within one or more distinct time intervals, indicating temporal regularity that may vary across different segments of the dataset. Unlike traditional periodic patterns, which assume consistent periodic behavior over time, LPPs are characterized by their regular and continuous appearance within certain time intervals.

Applications: Anomaly Detection, Time Series Forecasting, Resource Management.

Local Periodic Patterns (LPPs) which are patterns that have a periodic behavior in some non-predefined time-intervals. A pattern is said to be a local periodic pattern if it appears regularly and continuously in some time interval (s). A pattern is considered a local periodic pattern if it demonstrates periodic behavior within one or more distinct time intervals, indicating temporal regularity that may vary across different segments of the dataset. Unlike traditional periodic patterns, which assume consistent periodic behavior over time, LPPs are characterized by their regular and continuous appearance within certain time intervals.

Applications: Anomaly Detection, Time Series Forecasting, Resource Management.

2.2.1 Basic

LPPGrowth

class PAMI.localPeriodicPattern.basic.LPPGrowth.**LPPGrowth**(*iFile*, *maxPer*, *maxSoPer*, *minDur*, *sep=*\t')

Bases: _localPeriodicPatterns

Description

Local Periodic Patterns, which are patterns (sets of events) that have a periodic behavior in some non predefined time-intervals. A pattern is said to be a local periodic pattern if it appears regularly and continuously in some time-intervals. The maxSoPer (maximal period of spillovers) measure allows detecting time-intervals of variable lengths where a pattern is continuously periodic, while the minDur (minimal duration) measure ensures that those time-intervals have a minimum duration.

Reference

Fournier-Viger, P., Yang, P., Kiran, R. U., Ventura, S., Luna, J. M. (2020). Mining Local Periodic Patterns in a Discrete Sequence. Information Sciences, Elsevier, to appear. [ppt] DOI: 10.1016/j.ins.2020.09.044

Parameters

- iFile str : Name of the Input file to mine complete set of local periodic pattern's
- oFile str : Name of the output file to store complete set of local periodic patterns
- minDur str: Minimal duration in seconds between consecutive periods of time-intervals where a pattern is continuously periodic.
- **maxPer** float: Controls the maximum number of transactions in which any two items within a pattern can reappear.
- maxSoPer float: Controls the maximum number of time periods between consecutive periods of time-intervals where a pattern is continuously periodic.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

oFile

[str] Output file name or path of the output file

maxPer

[float] User defined maxPer value.

maxSoPer

[float] User defined maxSoPer value.

minDur

[float] User defined minDur value.

tsMin

[int / date] First time stamp of input data.

tsMax

[int / date] Last time stamp of input data.

startTime

[float] Time when start of execution the algorithm.

endTime

[float] Time when end of execution the algorithm.

finalPatterns

[dict] To store local periodic patterns and its PTL.

tsList

[dict] To store items and its time stamp as bit vector.

root

[Tree] It is root node of transaction tree of whole input data.

PTL

[dict] Storing the item and its PTL.

items

[list] Storing local periodic item list.

sep: str

separator used to distinguish items from each other. The default separator is tab space.

Methods

findSeparator(line)

Find the separator of the line which split strings.

creteLPPlist()

Create the local periodic patterns list from input data.

createTSList()

Create the tsList as bit vector from input data.

generateLPP()

Generate 1 length local periodic pattens by tsList and execute depth first search.

createLPPTree()

Create LPPTree of local periodic item from input data.

patternGrowth(tree, prefix, prefixPFList)

Execute pattern growth algorithm. It is important function in this program.

calculatePTL(tsList)

Calculate PTL from input tsList as integer list.

calculatePTLbit(tsList)

Calculate PTL from input tsList as bit vector.

mine()

Mining process will start from here.

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function.

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function.

getLocalPeriodicPatterns()

return local periodic patterns and its PTL

save(oFile)

Complete set of local periodic patterns will be loaded in to an output file.

getPatternsAsDataFrame()

Complete set of local periodic patterns will be loaded in to a dataframe.

Executing the code on terminal:

Format:

Example Usage:

(.venv) \$ python3 LPPMGrowth.py sampleDB.txt patterns.txt 0.3 0.4 0.5

Sample run of importing the code:

```
from PAMI.localPeriodicPattern.basic import LPPGrowth as alg
obj = alg.LPPGrowth(iFile, maxPer, maxSoPer, minDur)
obj.mine()
localPeriodicPatterns = obj.getPatterns()
print(f'Total number of local periodic patterns: {len(localPeriodicPatterns)}')
obj.save(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print(f'Total memory in USS: {memUSS}')
memRSS = obj.getMemoryRSS()
print(f'Total memory in RSS: {memRSS}')
runtime = obj.getRuntime()
print(f'Total execution time in seconds: {runtime})
```

Credits:

The complete program was written by So Nakamura under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$\texttt{getPatterns()} \rightarrow \texttt{Dict}$

Function to send the set of local periodic patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final local periodic patterns in a dataframe

Returns

returning local periodic patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

mine() \rightarrow None

Mining process start from here.

$printResults() \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of local periodic patterns will be loaded in to an output file

Parameters

outFile (csv file) – name of the output file

Returns None

$startMine() \rightarrow None$

Mining process start from here.

class PAMI.localPeriodicPattern.basic.LPPGrowth.Node

Bases: object

A class used to represent the node of localPeriodicPatternTree

Attributes

item

[int] storing item of a node

parent

[node] To maintain the parent of every node

child

[list] To maintain the children of node

nodeLink

[node] To maintain the next node of node

tidList

[set] To maintain timestamps of node

Methods

getChild(itemName)

storing the children to their respective parent nodes

getChild(*item: int*) \rightarrow *Node*

This function is used to get child node from the parent node

Parameters

item (int) – item of the parent node

Returns

if node have node of item, then return it. if node don't have return []

Return type

Node

class PAMI.localPeriodicPattern.basic.LPPGrowth.Tree

Bases: object

A class used to represent the frequentPatternGrowth tree structure

Attributes

root

[node] Represents the root node of the tree

nodeLinks

[dictionary] storing last node of each item

firstNodeLink

[dictionary] storing first node of each item

Methods

addTransaction(transaction,timeStamp)

creating transaction as a branch in frequentPatternTree

fixNodeLinks(itemName, newNode)

add newNode link after last node of item

deleteNode(itemName)

delete all node of item

createPrefixTree(path,timeStampList)
 create prefix tree by path

addTransaction(*transaction: List[int]*, *tid: int*) \rightarrow None

add transaction into tree

Parameters

• transaction (list) – it represents the one transaction in database

• tid (list or int) – represents the timestamp of transaction

Returns

None

 $createPrefixTree(path: List[int], tidList: List[int]) \rightarrow None$

create prefix tree by path

Parameters

- **path** (*list*) it represents path to root from prefix node
- tidList (list) it represents tid of each item

Returns

None

$\texttt{deleteNode}(\textit{item: int}) \rightarrow \text{None}$

delete the node from tree

Parameters

item (str) – it represents the item name of node

Returns

None

fixNodeLinks(*item: int, newNode:* Node) \rightarrow None

fix node link

Parameters

- item (*string*) it represents item name of newNode
- newNode (Node) it represents node which is added

Returns

None

LPPMBreadth

class PAMI.localPeriodicPattern.basic.LPPMBreadth.LPPMBreadth(*iFile*, *maxPer*, *maxSoPer*, *minDur*, *sep=**t'*)

Bases: _localPeriodicPatterns

Description

Local Periodic Patterns, which are patterns (sets of events) that have a periodic behavior in some non predefined time-intervals. A pattern is said to be a local periodic pattern if it appears regularly and continuously in some time-intervals. The maxSoPer (maximal period of spillovers) measure allows detecting time-intervals of variable lengths where a pattern is continuously periodic, while the minDur (minimal duration) measure ensures that those time-intervals have a minimum duration.

Reference

Fournier-Viger, P., Yang, P., Kiran, R. U., Ventura, S., Luna, J. M. (2020). Mining Local Periodic Patterns in a Discrete Sequence. Information Sciences, Elsevier, to appear. [ppt] DOI: 10.1016/j.ins.2020.09.044

Parameters

- iFile str : Name of the Input file to mine complete set of local periodic pattern's
- oFile str : Name of the output file to store complete set of local periodic patterns
- **minDur** str: Minimal duration in seconds between consecutive periods of time-intervals where a pattern is continuously periodic.
- **maxPer** float: Controls the maximum number of transactions in which any two items within a pattern can reappear.
- **maxSoPer** float: Controls the maximum number of time periods between consecutive periods of time-intervals where a pattern is continuously periodic.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

oFile

[str] Output file name or path of the output file

maxPer

[float] User defined maxPer value.

maxSoPer

[float] User defined maxSoPer value.

minDur

[float] User defined minDur value.

tsMin

[int / date] First time stamp of input data.

tsMax

[int / date] Last time stamp of input data.

startTime

[float] Time when start of execution the algorithm.

endTime

[float] Time when end of execution the algorithm.

finalPatterns

[dict] To store local periodic patterns and its PTL.

tsList

[dict] To store items and its time stamp as bit vector.

sep: str

separator used to distinguish items from each other. The default separator is tab space.

Methods

createTSList()

Create the tsList as bit vector from input data.

generateLPP()

Generate 1 length local periodic pattens by tsList and execute depth first search.

calculatePTL(tsList)

Calculate PTL from input tsList as bit vector

LPPMBreathSearch(extensionOfP)

Mining local periodic patterns using breadth first search.

mine()

Mining process will start from here.

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function.

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function.

getLocalPeriodicPatterns()

return local periodic patterns and its PTL

save(oFile)

Complete set of local periodic patterns will be loaded in to an output file.

getPatternsAsDataFrame()

Complete set of local periodic patterns will be loaded in to a dataframe.

Executing the code on terminal:

Format:

Example Usage:

(.venv) \$ python3 LPPMBreadth.py sampleDB.txt patterns.txt 0.3 0.4 0.5

Sample run of importing the code:

```
from PAMI.localPeriodicPattern.basic import LPPMBreadth as alg
obj = alg.LPPMBreadth(iFile, maxPer, maxSoPer, minDur)
obj.mine()
localPeriodicPatterns = obj.getPatterns()
print(f'Total number of local periodic patterns: {len(localPeriodicPatterns)}')
obj.save(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print(f'Total memory in USS: {memUSS}')
memRSS = obj.getMemoryRSS()
print(f'Total memory in RSS: {memRSS}')
runtime = obj.getRuntime()
print(f'Total execution time in seconds: {runtime})
```

Credits:

The complete program was written by So Nakamura under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$\texttt{getPatterns()} \rightarrow \texttt{Dict}[\texttt{Tuple[str, ...]} \mid \texttt{str, Set}[\texttt{Tuple[int, int]}]]$

Function to send the set of local periodic patterns after completion of the mining process
Returns

returning frequent patterns

Return type

dict

$\texttt{getPatternsAsDataFrame()} \rightarrow DataFrame$

Storing final local periodic patterns in a dataframe

Returns

returning local periodic patterns in a dataframe

Return type pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None

Mining process start from here.

$printResults() \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of local periodic patterns will be loaded in to an output file

Parameters outFile (csv file) – name of the output file

Returns None

$startMine() \rightarrow None$

Mining process start from here.

LPPMDepth

class PAMI.localPeriodicPattern.basic.LPPMDepth.**LPPMDepth**(*iFile*, *maxPer*, *maxSoPer*, *minDur*, *sep=\t'*)

Bases: _localPeriodicPatterns

Description

Local Periodic Patterns, which are patterns (sets of events) that have a periodic behavior in some non predefined time-intervals. A pattern is said to be a local periodic pattern if it appears regularly and continuously in some time-intervals. The maxSoPer (maximal period of spillovers) measure allows detecting time-intervals of variable lengths where a pattern is continuously periodic, while the minDur (minimal duration) measure ensures that those time-intervals have a minimum duration.

Reference

Fournier-Viger, P., Yang, P., Kiran, R. U., Ventura, S., Luna, J. M. (2020). Mining Local Periodic Patterns in a Discrete Sequence. Information Sciences, Elsevier, to appear. [ppt] DOI: 10.1016/j.ins.2020.09.044

Parameters

- iFile str : Name of the Input file to mine complete set of local periodic pattern's
- oFile str : Name of the output file to store complete set of local periodic patterns
- **minDur** str: Minimal duration in seconds between consecutive periods of time-intervals where a pattern is continuously periodic.
- **maxPer** float: Controls the maximum number of transactions in which any two items within a pattern can reappear.
- **maxSoPer** float: Controls the maximum number of time periods between consecutive periods of time-intervals where a pattern is continuously periodic.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

oFile

[str] Output file name or path of the output file

maxPer

[float] User defined maxPer value.

maxSoPer

[float] User defined maxSoPer value.

minDur

[float] User defined minDur value.

tsmin

[int / date] First time stamp of input data.

tsmax

[int / date] Last time stamp of input data.

startTime

[float] Time when start of execution the algorithm.

endTime

[float] Time when end of execution the algorithm.

finalPatterns

[dict] To store local periodic patterns and its PTL.

tsList

[dict] To store items and its time stamp as bit vector.

sep

[str] separator used to distinguish items from each other. The default separator is tab space.

Methods

createTSlist()

Create the TSlist as bit vector from input data.

generateLPP()

Generate 1 length local periodic pattens by TSlist and execute depth first search.

calculatePTL(tsList)

Calculate PTL from input tsList as bit vector

LPPMDepthSearch(extensionOfP)

Mining local periodic patterns using depth first search.

mine()

Mining process will start from here.

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function.

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function.

getLocalPeriodicPatterns()

return local periodic patterns and its PTL

save(oFile)

Complete set of local periodic patterns will be loaded in to an output file.

getPatternsAsDataFrame()

Complete set of local periodic patterns will be loaded in to a dataframe.

Executing the code on terminal:

Format:

(.venv) \$ python3 LPPMDepth.py <inputFile> <outputFile> <maxPer> <minSoPer> <minDur>

Example Usage:

(.venv) \$ python3 LPPMDepth.py sampleDB.txt patterns.txt 0.3 0.4 0.5

Sample run of importing the code:

```
from PAMI.localPeriodicPattern.basic import LPPMDepth as alg
obj = alg.LPPMDepth(iFile, maxPer, maxSoPer, minDur)
obj.mine()
localPeriodicPatterns = obj.getPatterns()
print(f'Total number of local periodic patterns: {len(localPeriodicPatterns)}')
obj.save(oFile)
```

(continues on next page)

(continued from previous page)

```
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print(f'Total memory in USS: {memUSS}')
memRSS = obj.getMemoryRSS()
print(f'Total memory in RSS: {memRSS}')
runtime = obj.getRuntime()
print(f'Total execution time in seconds: {runtime})
```

Credits:

The complete program was written by So Nakamura under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS() \rightarrow float

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

$getPatterns() \rightarrow Dict[Tuple[str, ...] | str, Set[Tuple[int, int]]]$

Function to send the set of local periodic patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

$\texttt{getPatternsAsDataFrame()} \rightarrow \texttt{DataFrame}$

Storing final local periodic patterns in a dataframe

Returns

returning local periodic patterns in a dataframe

Return type

pd.DataFrame

```
getRuntime() \rightarrow float
     Calculating the total amount of runtime taken by the mining process
          Returns
               returning total amount of runtime taken by the mining process
          Return type
               float
mine() \rightarrow None
     Mining process start from here. This function calls createTSlist and generateLPP.
printResults() \rightarrow None
     This function is used to print the results
save(outFile: str) \rightarrow None
     Complete set of local periodic patterns will be loaded in to an output file
          Parameters
               outFile (csv file) – name of the output file
          Returns
               None
startMine() \rightarrow None
     Mining process start from here. This function calls createTSlist and generateLPP.
```

2.3 Partial Periodic Frequent Pattern Mining

Partial Periodic-Frequent Patterns in a temporal database are recurring patterns that exhibit partial cyclic repetitions over time. Unlike full periodic-frequent patterns, which require complete cyclic repetitions, partial periodic-frequent patterns capture temporal regularities that may not follow a strict periodicity. These patterns represent recurring behaviors or events within specific time intervals, where the frequency of occurrence varies but still demonstrates a degree of periodicity. The interestingness of a partial periodic-frequent pattern is determined by its periodic ratio, which measures the proportion of cyclic repetitions it exhibits in the database.

Applications: Predictive Maintenance, Traffic Management, Environmental Monitoring.

Partial Periodic-Frequent Patterns in a temporal database are recurring patterns that exhibit partial cyclic repetitions over time. Unlike full periodic-frequent patterns, which require complete cyclic repetitions, partial periodic-frequent patterns capture temporal regularities that may not follow a strict periodicity. These patterns represent recurring behaviors or events within specific time intervals, where the frequency of occurrence varies but still demonstrates a degree of periodicity. The interestingness of a partial periodic-frequent pattern is determined by its periodic ratio, which measures the proportion of cyclic repetitions it exhibits in the database.

Applications: Predictive Maintenance, Traffic Management, Environmental Monitoring.

2.3.1 Basic

GPFgrowth

Bases: partialPeriodicPatterns

Description

GPFgrowth is algorithm to mine the partial periodic frequent pattern in temporal database.

Reference

R. Uday Kiran, J.N. Venkatesh, Masashi Toyoda, Masaru Kitsuregawa, P. Krishna Reddy, Discovering partial periodic-frequent patterns in a transactional database, Journal of Systems and Software, Volume 125, 2017, Pages 170-182, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2016. 11.035.

Parameters

- iFile str : Name of the Input file to mine complete set of frequent pattern's
- oFile str : Name of the output file to store complete set of frequent patterns
- minSup str: The user can specify minSup either in count or proportion of database size.
- **minPR** str: Controls the maximum number of transactions in which any two items within a pattern can reappear.
- **maxPer** str: Controls the maximum number of transactions in which any two items within a pattern can reappear.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

inputFile

[file] Name of the input file to mine complete set of frequent pattern

minSup

[float] The user defined minSup

maxPer

[float] The user defined maxPer

minPR

[float] The user defined minPR

finalPatterns

[dict] it represents to store the pattern

runTime

[float] storing the total runtime of the mining process

memoryUSS

[float] storing the total amount of USS memory consumed by the program

memoryRSS

[float] storing the total amount of RSS memory consumed by the program

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

storePatternsInFile(ouputFile)

Complete set of frequent patterns will be loaded in to an output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to an output file

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

Executing code on Terminal:

Format:

Examples:

>>> python3 GPFgrowth.py sampleDB.txt patterns.txt 10 10 0.5

Sample run of the importing code:

... code-block:: python

from PAMI.partialPeriodicFrequentPattern.basic import GPFgrowth as alg

obj = alg.GPFgrowth(inputFile, outputFile, minSup, maxPer, minPR)

obj.startMine()

partialPeriodicFrequentPatterns = obj.getPatterns()

print("Total number of partial periodic Patterns:", len(partialPeriodicFrequentPatterns))

obj.save(oFile)

Df = obj.getPatternInDf()

memUSS = obj.getMemoryUSS()

print("Total Memory in USS:", memUSS)

memRSS = obj.getMemoryRSS()

print("Total Memory in RSS", memRSS)

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by Nakamura under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function :return: returning RSS memory consumed by the mining process :rtype: float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function :return: returning USS memory consumed by the mining process :rtype: float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process :return: returning frequent patterns :rtype: dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe :return: returning frequent patterns in a dataframe :rtype: pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process :return: returning total amount of runtime taken by the mining process :rtype: float

mine()

printResults()

this function is used to print the results

runTime = 0

save(outFile)

Complete set of frequent patterns will be loaded in to an output file :param outFile: name of the output file :type outFile: csv file

class PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth.Node

Bases: object

A class used to represent the node of frequentPatternTree

Attributes

item

[int] storing item of a node

parent

[node] To maintain the parent of every node

child

[list] To maintain the children of node

nodeLink

[node] To maintain the next node of node

tidList

[set] To maintain timestamps of node

Methods

getChild(itemName)

storing the children to their respective parent nodes

getChild(item)

Parameters item –

Returns

if node have node of item, then return it. if node don't have return []

class PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth.PFgrowth(tree, prefix, PFList, minSup,

maxPer, minPR, last)

Bases: object

This class is pattern growth algorithm

Attributes

tree

[Node] represents the root node of prefix tree

prefix

[list] prefix is list of prefix items

PFList

[dict] storing time stamp each item

minSup

[float] user defined min Support

maxPer

[float] user defined max Periodicity

minPR

[float] user defined min PR

last

[int] represents last time stamp in database

Methods

run

it is pattern growth algorithm

run()

run the pattern growth algorithm :return: partial periodic frequent pattern in conditional pattern

class PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth.Tree

Bases: object

A class used to represent the frequentPatternGrowth tree structure

Attributes

root

[node] Represents the root node of the tree

nodeLinks

[dictionary] storing last node of each item

firstNodeLink

[dictionary] storing first node of each item

Methods

addTransaction(transaction,timeStamp)

creating transaction as a branch in frequentPatternTree

fixNodeLinks(itemName, newNode)

add newNode link after last node of item

deleteNode(itemName) delete all node of item

createPrefixTree(path, timeStampList)

create prefix tree by path

createConditionalTree(PFList, minSup, maxPer, minPR, last)

create conditional tree. Its nodes are satisfy IP / (minSup+1) >= minPR

addTransaction(transaction, tid)

add transaction into tree

Parameters

- **transaction** (*list*) it represents the one transactions in database
- **tid** (*list*) represents the timestamp of transaction

createConditionalTree(PFList, minSup, maxPer, minPR, last)

create conditional tree by PFlist

Parameters

- **PFList** (*dict*) it represents timestamp each item
- minSup it represents minSup
- maxPer it represents maxPer
- minPR it represents minPR
- **last** it represents last timestamp in database

Returns

return is PFlist which satisfy ip / (minSup+1) >= minPR

createPrefixTree(path, tidList)

create prefix tree by path

Parameters

- **path** (*list*) it represents path to root from prefix node
- tidList (list) it represents tid of each item

deleteNode(item)

delete the node from tree

Parameters

item (str) – it represents the item name of node

fixNodeLinks(item, newNode)

fix node link

Parameters

- item (string) it represents item name of newNode
- newNode (Node) it represents node which is added

class PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth.calculateIP(maxPer, timeStamp,

timeStampFinal)

Bases: object

This class calculate ip from timestamp

Attributes

maxPer

[float] it represents user defined maxPer value

timeStamp

[list] it represents timestamp of item

timeStampFinal

[int] it represents last timestamp of database

Methods

run

calculate ip from its timestamp list

run()

calculate ip from timeStamp list :return: it represents ip value

class PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth.generatePFListver2(Database,

minSup, maxPer, minPR)

Bases: object

generate time stamp list from input file

Attributes

inputFile

[str] it is input file name

minSup

[float] user defined minimum support value

maxPer

[float] user defined max Periodicity value

minPR

[float] user defined min PR value

PFList

[dict] storing timestamps each item

findSeparator(line)

find separator of line in database

Parameters

line (*list*) – it represents one line in database

Returns

return separator

run()

generate PFlist :return: timestamps and last timestamp

class PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth.generatePFTreever2(Database,

tidList)

Bases: object

create tree from tidList and input file

Attributes

inputFile [str] it represents input file name

tidList

[dict] storing tids each item

root

[Node] it represents the root node of the tree

Methods

run

it create tree

find separator(line) find separator in the line of database

findSeparator(line)

find separator of line in database

Parameters

line (*list*) – it represents one line in database

Returns

return separator

run()

create tree from database and tidList :return: the root node of tree

PPF_DFS

class PAMI.partialPeriodicFrequentPattern.basic.PPF_DFS.**PPF_DFS**(*iFile*, *minSup*, *maxPer*, *minPR*,

sep=(t')

Bases: partialPeriodicPatterns

Description

PPF_DFS is algorithm to mine the partial periodic frequent patterns.

References

(Has to be added)

Parameters

- iFile str : Name of the Input file to mine complete set of frequent pattern's
- oFile str : Name of the output file to store complete set of frequent patterns
- minSup str: The user can specify minSup either in count or proportion of database size.

- **minPR** str: Controls the maximum number of transactions in which any two items within a pattern can reappear.
- maxPer str: Controls the maximum number of transactions in which any two items within a pattern can reappear.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] input file path

oFile

[file] output file name

minSup

[float] user defined minSup

maxPer

[float] user defined maxPer

minPR

[float] user defined minPR

tidlist

[dict] it stores tids each item

last

[int] it represents last time stamp in database

lno

[int] number of line in database

mapSupport

[dict] to maintain the information of item and their frequency

finalPatterns

[dict] it represents to store the patterns

runTime

[float] storing the total runtime of the mining process

memoryUSS

[float] storing the total amount of USS memory consumed by the program

memoryRSS

[float] storing the total amount of RSS memory consumed by the program

Methods

getPer_Sup(tids)
 caluclate ip / (sup+1)

getPerSup(tids) caluclate ip

oneItems(path) scan all lines in database

save(prefix,suffix,tidsetx)

save prefix pattern with support and periodic ratio

Generation(prefix, itemsets, tidsets)

Userd to implement prefix class equibalence method to generate the periodic patterns recursively

startMine()

Mining process will start from here

getPartialPeriodicPatterns()

Complete set of patterns will be retrieved with this function

save(ouputFile)

Complete set of frequent patterns will be loaded in to an ouput file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to an ouput file

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

Executing code on Terminal:

Format:

Examples:

>>> python3 PPF_DFS.py sampleDB.txt patterns.txt 10 10 0.5

Sample run of the importing code:

... code-block:: python

from PAMI.partialPeriodicFrequentpattern.basic import PPF_DFS as alg

obj = alg.PPF_DFS(iFile, minSup)

obj.startMine()

frequentPatterns = obj.getPatterns()

print("Total number of Frequent Patterns:", len(frequentPatterns))

obj.save(oFile)

Df = obj.getPatternInDataFrame()

memUSS = obj.getMemoryUSS()

print("Total Memory in USS:", memUSS)

memRSS = obj.getMemoryRSS()

print("Total Memory in RSS", memRSS)

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by S. Nakamura under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function :return: returning RSS memory consumed by the mining process :rtype: float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function :return: returning USS memory consumed by the mining process :rtype: float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process :return: returning frequent patterns :rtype: dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe :return: returning frequent patterns in a dataframe :rtype: pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process :return: returning total amount of runtime taken by the mining process :rtype: float

mine()

Main program start with extracting the periodic frequent items from the database and performs prefix equivalence to form the combinations and generates closed periodic frequent patterns.

printResults()

this function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file :param outFile: name of the output file :type outFile: csv file

2.4 Partial Periodic Pattern Mining

Partial periodic pattern mining involves the identification of recurring patterns or sequences within a dataset that exhibit partial periodic behavior. Unlike traditional periodic pattern mining, where patterns repeat exactly at regular intervals, partial periodic patterns may exhibit variations or irregularities in their periodicity. These patterns may occur intermittently or periodically with some degree of variability, making them challenging to detect using conventional mining techniques. Applications: Healthcare Monitoring, Financial Time Series Analysis, Network Traffic Analysis.

Partial periodic pattern mining involves the identification of recurring patterns or sequences within a dataset that exhibit partial periodic behavior. Unlike traditional periodic pattern mining, where patterns repeat exactly at regular intervals,

partial periodic patterns may exhibit variations or irregularities in their periodicity. These patterns may occur intermittently or periodically with some degree of variability, making them challenging to detect using conventional mining techniques. Applications: Healthcare Monitoring, Financial Time Series Analysis, Network Traffic Analysis.

2.4.1 Basic

PPPGrowth

class PAMI.partialPeriodicPattern.basic.PPPGrowth.**PPPGrowth**(*iFile*, *minPS*, *period*, *sep=\t'*) Bases: _partialPeriodicPatterns

Description

3pgrowth is fundamental approach to mine the partial periodic patterns in temporal database.

Reference

Discovering Partial Periodic Itemsets in Temporal Databases, SSDBM '17: Proceedings of the 29th International Conference on Scientific and Statistical Database ManagementJune 2017 Article No.: 30 Pages 1–6https://doi.org/10.1145/3085504.3085535

Parameters

- iFile str : Name of the Input file to mine complete set of frequent pattern's
- oFile str : Name of the output file to store complete set of frequent patterns
- minPS float: Minimum partial periodic pattern...
- **period** float: Minimum partial periodic...
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minPS: float or int or str

The user can specify minPS either in count or proportion of database size. If the program detects the data type of minPS is integer, then it treats minPS is expressed in count. Otherwise, it will be treated as float. Example: minPS=10 will be treated as integer, while minPS=10.0 will be treated as float

period: float or int or str

The user can specify period either in count or proportion of database size. If the program detects the data type of period is integer, then it treats period is expressed in count. Otherwise, it will be treated as float. Example: period=10 will be treated as integer, while period=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

finalPatterns

[dict] it represents to store the patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets()

Scans the dataset or dataframes and stores in list format

partialPeriodicOneItem()

Extracts the one-frequent patterns from transactions

updateTransactions()

updates the transactions by removing the aperiodic items and sort the transactions with items by decreasing support

buildTree()

constrcuts the main tree by setting the root node as null

startMine()

main program to mine the partial periodic patterns

Executing the code on terminal:

```
Format:
(.venv) $python3 PPPGrowth.py <inputFile> <outputFile> <minPS> <period>
Examples:
(.venv) $ python3 PPPGrowth.py sampleDB.txt patterns.txt 10.0 2.0
```

Sample run of the importing code:

```
from PAMI.periodicFrequentPattern.basic import PPPGrowth as alg
obj = alg.PPPGrowth(iFile, minPS, period)
obj.startMine()
partialPeriodicPatterns = obj.getPatterns()
print("Total number of partial periodic Patterns:",_
-,len(partialPeriodicPatterns))
obj.save(oFile)
Df = obj.getPatternInDf()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

```
getMemoryRSS() \rightarrow float
```

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns returning RSS memory consumed by the mining process Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

$\texttt{getPatterns()} \rightarrow Dict[str, int]$

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

$\texttt{getPatternsAsDataFrame()} \rightarrow \text{DataFrame}$

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None

Main method where the patterns are mined by constructing tree. :return: None

$printResults() \rightarrow None$

This function is used to print the results :return: None

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to a output file

Parameters

outFile (csv file) - name of the output file

Returns None

$startMine() \rightarrow None$

Main method where the patterns are mined by constructing tree. :return: None

PPP_ECLAT

class PAMI.partialPeriodicPattern.basic.PPP_ECLAT.**PPP_ECLAT**(*iFile*, *minPS*, *period*, *sep=\t'*) Bases: _partialPeriodicPatterns

Descripition

3pEclat is the fundamental approach to mine the partial periodic frequent patterns.

Reference

R. Uday Kirana,b, , J.N. Venkateshd, Masashi Toyodaa , Masaru Kitsuregawaa,c , P. Krishna Reddy Discovering partial periodic-frequent patterns in a transactional database https://www.tkl.iis.u-tokyo.ac.jp/new/uploads/publication_file/file/774/JSS_2017.pdf

Parameters

- iFile str : Name of the Input file to mine complete set of frequent pattern's
- oFile str : Name of the output file to store complete set of frequent patterns
- minPS float: Minimum partial periodic pattern...
- period float: Minimum partial periodic...
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

self.iFile

[file] Name of the Input file or path of the input file

self. oFile

[file] Name of the output file or path of the output file

minPS: float or int or str

The user can specify minPS either in count or proportion of database size. If the program detects the data type of minPS is integer, then it treats minPS is expressed in count. Otherwise, it will be treated as float. Example: minPS=10 will be treated as integer, while minPS=10.0 will be treated as float

period: float or int or str

The user can specify period either in count or proportion of database size. If the program detects the data type of period is integer, then it treats period is expressed in count. Otherwise, it will be treated as float. Example: period=10 will be treated as integer, while period=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

finalPatterns

[dict] it represents to store the patterns

tidList

[dict] stores the timestamps of an item

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to an output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingOneitemSets()

Scan the database and store the items with their timestamps which are periodic frequent

getPeriodAndSupport()

Calculates the support and period for a list of timestamps.

Generation()

Used to implement prefix class equivalence method to generate the periodic patterns recursively

Executing the code on terminal:

```
Format:
(.venv) $ python3 PPP_ECLAT.py <inputFile> <outputFile> <minPS> <period>
Examples:
(.venv) $ python3 PPP_ECLAT.py sampleDB.txt patterns.txt 0.3 0.4
```

Sample run of importing the code:

... code-block:: python

from PAMI.periodicFrequentPattern.basic import PPP_ECLAT as alg

obj = alg.PPP_ECLAT(iFile, minPS,period)

obj.startMine()

Patterns = obj.getPatterns()

print("Total number of partial periodic patterns:", len(Patterns))

obj.save(oFile)

Df = obj.getPatternsAsDataFrame()

memUSS = obj.getMemoryUSS()

print("Total Memory in USS:", memUSS)

memRSS = obj.getMemoryRSS()

print("Total Memory in RSS", memRSS)

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by P.RaviKumar under the supervision of Professor Rage Uday Kiran.

Mine() \rightarrow None

Main program start with extracting the periodic frequent items from the database and performs prefix equivalence to form the combinations and generates partial-periodic patterns. :return: None

$\texttt{getMemoryRSS()} \to \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

$getPatterns() \rightarrow Dict[str, int]$

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

$\texttt{getPatternsAsDataFrame()} \rightarrow DataFrame$

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

noat

$printResults() \rightarrow None$

This function is used to print the results :return: None

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (file) – name of the output file

Returns

None

$startMine() \rightarrow None$

Main program start with extracting the periodic frequent items from the database and performs prefix equivalence to form the combinations and generates partial-periodic patterns. :return: None

GThreePGrowth

class PAMI.partialPeriodicPattern.basic.GThreePGrowth.GThreePGrowth(iFile: str, minPS: int | float | str, period: int | float | str, relativePS: bool, sep: str = \t')

Bases: _partialPeriodicPatterns

Description

3pgrowth is fundamental approach to mine the partial periodic patterns in temporal database.

Reference

Reference : Discovering Partial Periodic Itemsets in Temporal Databases, SSDBM '17: Proceedings of the 29th International Conference on Scientific and Statistical Database ManagementJune 2017 Article No.: 30 Pages 1–6https://doi.org/10.1145/3085504.3085535

Parameters

- iFile str : Name of the Input file to mine complete set of frequent pattern's
- oFile str : Name of the output file to store complete set of frequent patterns
- minPS float: Minimum partial periodic pattern...
- period float: Minimum partial periodic...
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

self.iFile

[file] Name of the Input file or path of the input file

self. oFile

[file] Name of the output file or path of the output file

minPS: float or int or str

The user can specify minPS either in count or proportion of database size. If the program detects the data type of minPS is integer, then it treats minPS is expressed in count. Otherwise, it will be treated as float. Example: minPS=10 will be treated as integer, while minPS=10.0 will be treated as float

period: float or int or str

The user can specify period either in count or proportion of database size. If the program detects the data type of period is integer, then it treats period is expressed in count. Otherwise, it will be treated as float. Example: period=10 will be treated as integer, while period=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

self.memoryUSS

[float] To store the total amount of USS memory consumed by the program

self.memoryRSS

[float] To store the total amount of RSS memory consumed by the program

self.startTime:float

To record the start time of the mining process

self.endTime:float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

finalPatterns

[dict] it represents to store the patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets()

Scans the dataset or dataframes and stores in list format

partialPeriodicOneItem()

Extracts the one-frequent patterns from transactions

updateTransactions()

updates the transactions by removing the aperiodic items and sort the transactions with items by decreasing support

buildTree()

constrcuts the main tree by setting the root node as null

startMine()

main program to mine the partial periodic patterns

Executing the code on terminal:

Format:

```
>>> python3 PPPGrowth.py <inputFile> <outputFile> <minPS> <period>
```

Examples:

>>> python3 PPPGrowth.py sampleDB.txt patterns.txt 10.0 2.0

Sample run of the importing code:

```
from PAMI.periodicFrequentPattern.basic import PPPGrowth as alg
obj = alg.PPPGrowth(iFile, minPS, period)
obj.startMine()
partialPeriodicPatterns = obj.getPatterns()
print("Total number of partial periodic Patterns:", len(partialPeriodicPatterns))
obj.save(oFile)
Df = obj.getPatternInDf()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

 $getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None

Main method where the patterns are mined by constructing tree.

$printResults() \rightarrow None$

this function is used to print the results

save(outFile: str)

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (csv file) – name of the output file

2.4.2 closed

PPPClose

class PAMI.partialPeriodicPattern.closed.PPPClose.PPPClose(*iFile*, periodicSupport, period,

sep = (t')

Bases: _partialPeriodicPatterns

Description

PPPClose algorithm is used to discover the closed partial periodic patterns in temporal databases. It uses depthfirst search.

Reference

R. Uday Kiran1, J. N. Venkatesh2, Philippe Fournier-Viger3, Masashi Toyoda1, P. Krishna Reddy2 and Masaru Kitsuregawa https://www.tkl.iis.u-tokyo.ac.jp/new/uploads/publication_file/file/799/PAKDD.pdf

Parameters

- iFile str : Name of the Input file to mine complete set of periodic frequent pattern's
- oFile str : Name of the output file to store complete set of periodic frequent pattern's
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.
- iFile str : Name of the Input file to mine complete set of frequent pattern's
- oFile str : Name of the output file to store complete set of frequent patterns
- period float: Minimum partial periodic...
- periodicSupport float: Minimum partial periodic...
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

oFile

[str] Name of the output file or path of the input file

periodicSupport: int or float or str

The user can specify periodicSupport either in count or proportion of database size. If the program detects the data type of periodicSupport is integer, then it treats periodicSupport is expressed in count. Otherwise, it will be treated as float. Example: periodicSupport=10 will be treated as integer, while periodicSupport=10.0 will be treated as float

period: int or float or str

The user can specify period either in count or proportion of database size. If the program detects the data type of period is integer, then it treats period is expressed in count. Otherwise, it will be treated as float. Example: period=10 will be treated as integer, while period=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

finalPatterns: dict

Storing the complete set of patterns in a dictionary variable

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

Executing the code on terminal:

Format:

(.venv) \$ python3 PPPClose.py <inputFile> <outputFile> <periodicSupport> <period>

Examples:

```
(.venv) $ python3 PPPClose.py sampleTDB.txt patterns.txt 0.3 0.4
```

Sample run of the imported code:

```
from PAMI.partialPeriodicPattern.closed import PPPClose as alg
obj = alg.PPPClose("../basic/sampleTDB.txt", "2", "6")
obj.startMine()
periodicFrequentPatterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(periodicFrequentPatterns))
obj.save("patterns")
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
```

(continues on next page)

(continued from previous page)

```
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine()

Mining process will start from here

printResults()

To print all the results of execution

save(outFile)

Complete set of frequent patterns will be loaded in to a output file

Parameters outFile (file) – name of the output file

startMine()

Mining process will start from here

2.4.3 maximal

Max3PGrowth

class PAMI.partialPeriodicPattern.maximal.Max3PGrowth.Max3PGrowth(*iFile*, *periodicSupport*, *period*, *sep*=\t')

Bases: _partialPeriodicPatterns

Description

Max3p-Growth algorithm IS to discover maximal periodic-frequent patterns in a temporal database. It extract the partial periodic patterns from 3p-tree and checks for the maximal property and stores all the maximal patterns in max3p-tree and extracts the maximal periodic patterns.

Reference

R. Uday Kiran, Yutaka Watanobe, Bhaskar Chaudhury, Koji Zettsu, Masashi Toyoda, Masaru Kitsuregawa, "Discovering Maximal Periodic-Frequent Patterns in Very Large Temporal Databases", IEEE 2020, https://ieeexplore.ieee.org/document/9260063

Parameters

- iFile str : Name of the Input file to mine complete set of frequent pattern's
- oFile str : Name of the output file to store complete set of frequent patterns
- period float: Minimum partial periodic...
- periodicSupport str: Minimum partial periodic...
- maximalTree str: Minimum partial periodic...
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

periodicSupport: float or int or str

The user can specify periodicSupport either in count or proportion of database size. If the program detects the data type of periodicSupport is integer, then it treats periodicSupport is

expressed in count. Otherwise, it will be treated as float. Example: periodicSupport=10 will be treated as integer, while periodicSupport=10.0 will be treated as float

period: float or int or str

The user can specify period either in count or proportion of database size. If the program detects the data type of period is integer, then it treats period is expressed in count. Otherwise, it will be treated as float. Example: period=10 will be treated as integer, while period=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

periodicSupport

[int/float] The user given minimum period-support

period

[int/float] The user given maximum period

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transaction

tree

[class] it represents the Tree class

itemSetCount

[int] it represents the total no of patterns

finalPatterns

[dict] it represents to store the patterns

Methods

startMine()

Mining process will start from here

getFrequentPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingitemSets(fileName)

Scans the dataset or dataframes and stores in list format

PeriodicFrequentOneItem()

Extracts the one-periodic-frequent patterns from Databases

updateDatabases()

update the Databases by removing aperiodic items and sort the Database by item decreased support

buildTree()

after updating the Databases ar added into the tree by setting root node as null

startMine()

the main method to run the program

Executing the code on terminal:

Format:

Examples:

>>> python3 Max3PGrowth.py sampleTDB.txt patterns.txt 0.3 0.4

Sample run of the importing code:

```
from PAMI.periodicFrequentPattern.maximal import ThreePGrowth as alg
obj = alg.ThreePGrowth(iFile, periodicSupport, period)
obj.startMine()
partialPeriodicPatterns = obj.partialPeriodicPatterns()
print("Total number of partial periodic Patterns:", len(partialPeriodicPatterns))
obj.save(oFile)
Df = obj.getPatternInDf()
```

(continues on next page)

(continued from previous page)

```
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
```

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of periodic-frequent patterns after completion of the mining process

Returns

returning periodic-frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

mine()

Mining process will start from this function

printResults()

This function is used to print the results

save(outFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

Parameters outFile (csv file) – name of the output file

2.4.4 TopK

k3PMiner

class PAMI.partialPeriodicPattern.topk.k3PMiner.**k3PMiner**(*iFile*, *k*, *period*, *sep*=\t')

Bases: partialPeriodicPatterns

Description

k3PMiner is and algorithm to discover top - k partial periodic patterns in a temporal database.

Reference

Palla Likhitha,Rage Uday Kiran, Discovering Top-K Partial Periodic Patterns in Big Temporal Databases https://dl.acm.org/doi/10.1007/978-3-031-39847-6_28

Parameters

- **iFile** str : Name of the Input file to mine complete set of periodic frequent pattern's
- oFile str : Name of the output file to store complete set of periodic frequent pattern's
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.
- iFile str : Name of the Input file to mine complete set of frequent pattern's
- oFile str : Name of the output file to store complete set of frequent patterns
- **period** str: Minimum partial periodic...
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

k: int

User specified count of top partial periodic patterns

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

oFile

[str] Name of the output file or the path of the output file

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

finalPatterns: dict

Storing the complete set of patterns in a dictionary variable

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets()

Scans the dataset or dataframes and stores in list format

frequentOneItem()

Generates one frequent patterns

eclatGeneration(candidateList)

It will generate the combinations of frequent items

generateFrequentPatterns(tidList)

It will generate the combinations of frequent items from a list of items
Executing the code on terminal:

Format:
python3 k3PMiner.py <iFile> <oFile> <k> <period>
Examples:
python3 k3PMiner.py sampleDB.txt patterns.txt 10 3

Sample run of the importing code:

... code-block:: python

import PAMI.partialPeriodicPattern.topk.k3PMiner as alg obj = alg.Topk_PPPGrowth(iFile, k, period) obj.startMine() partialPeriodicPatterns = obj.getPatterns() print("Total number of top partial periodic Patterns:", len(partialPeriodicPatterns)) obj.save(oFile) Df = obj.getPatternInDataFrame() memUSS = obj.getMemoryUSS() print("Total Memory in USS:", memUSS) memRSS = obj.getMemoryRSS() print("Total Memory in RSS", memRSS) run = obj.getRuntime() print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

ulet

$\verb+getPatternsAsDataFrame()$

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine()

Main function of the program

printResults()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (file) - name of the output file

startMine()

Main function of the program

2.4.5 Cuda

Pending

2.5 Periodic correlated pattern mining

Periodic correlated pattern mining is a data mining task aimed at discovering patterns within a temporal database that exhibit both periodic behavior and correlation between their occurrences. Unlike traditional periodic pattern mining, which focuses solely on periodicity, periodic correlated pattern mining considers the relationship between pattern occurrences over time. These patterns are characterized by their recurring nature and the presence of correlations between occurrences at specific time intervals.

Applications: Retail Sales Analysis, Web Usage Mining, Healthcare Monitoring.

Periodic correlated pattern mining is a data mining task aimed at discovering patterns within a temporal database that exhibit both periodic behavior and correlation between their occurrences. Unlike traditional periodic pattern mining, which focuses solely on periodicity, periodic correlated pattern mining considers the relationship between pattern occurrences over time. These patterns are characterized by their recurring nature and the presence of correlations between occurrences at specific time intervals.

Applications: Retail Sales Analysis, Web Usage Mining, Healthcare Monitoring.

2.5.1 Basic

EPCPGrowth

sep = (t')

Bases: _periodicCorrelatedPatterns

Description

EPCPGrowth is an algorithm to discover periodic-Correlated patterns in a temporal database.

Reference

http://www.tkl.iis.u-tokyo.ac.jp/new/uploads/publication_file/file/897/Venkatesh2018_ Chapter_DiscoveringPeriodic-Correlated.pdf

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup

[int or float or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

minAllConf

[int or float or str] The user can specify minAllConf either in count or proportion of database size. If the program detects the data type of minAllConf is integer, then it treats minAllCOnf is expressed in count. Otherwise, it will be treated as float. Example: minAllCOnf=10 will be treated as integer, while minAllConf=10.0 will be treated as float

maxPer

[int or float or str] The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in

count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

maxPerAllConf

[int or float or str] The user can specify maxPerAllConf either in count or proportion of database size. If the program detects the data type of maaxPerAllConf is integer, then it treats maxPerAllConf is expressed in count. Otherwise, it will be treated as float. Example : maxPerAllConf=10 will be treated as integer, while maxPerAllConf=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

PeriodicFrequentOneItem()

Extracts the one-periodic-frequent patterns from database

updateDatabases()

Update the database by removing aperiodic items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

to convert the user specified value

Executing the code on terminal:

Format:

>>> python3 PFPGrowth.py <inputFile> <outputFile> <minSup> <maxPer>

Examples:

>>> python3 PFPGrowth.py sampleTDB.txt patterns.txt 0.3 0.4

Sample run of importing the code:

(continues on next page)

(continued from previous page)

```
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \to \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$getPatterns() \rightarrow dict$

Function to send the set of periodic-frequent patterns after completion of the mining process

Returns

returning periodic-frequent patterns

Return type

dict

$\texttt{getPatternsAsDataFrame()} \rightarrow DataFrame$

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

 $printResults() \rightarrow None$

This function is used to print thr results

save(*outFile: str*) \rightarrow None

Complete set of periodic-frequent patterns will be loaded in to an output file

Parameters outFile (csv file) – name of the output file

```
\texttt{startMine()} \rightarrow None
```

Mining process will start from this function

2.6 Stable Periodic Pattern Mining

Stable Periodic Pattern Mining (SPPM) is a data mining task focused on discovering patterns within transactional databases that exhibit consistent and predictable periodic behavior. Unlike traditional periodic pattern mining approaches that may identify patterns with varying periodicity, SPPM specifically targets patterns with stable intervals between successive occurrences. These patterns, known as Stable Periodic-Frequent Patterns (SPPs), demonstrate stable repetition over time, making them more reliable and suitable for predictive modeling and analysis. SPPM aims to identify SPPs that adhere to user-defined constraints on periodicity stability, enabling the discovery of patterns with consistent periodic behavior.

Applications: Traffic Flow Optimization, Financial Market Analysis, Manufacturing Process Optimization.

Stable Periodic Pattern Mining (SPPM) is a data mining task focused on discovering patterns within transactional databases that exhibit consistent and predictable periodic behavior. Unlike traditional periodic pattern mining approaches that may identify patterns with varying periodicity, SPPM specifically targets patterns with stable intervals between successive occurrences. These patterns, known as Stable Periodic-Frequent Patterns (SPPs), demonstrate stable repetition over time, making them more reliable and suitable for predictive modeling and analysis. SPPM aims to identify SPPs that adhere to user-defined constraints on periodicity stability, enabling the discovery of patterns with consistent periodic behavior.

Applications: Traffic Flow Optimization, Financial Market Analysis, Manufacturing Process Optimization.

2.6.1 Basic

SPPGrowth

class PAMI.stablePeriodicFrequentPattern.basic.SPPGrowth.**SPPGrowth**(*inputFile*, *minSup*, *maxPer*, *maxLa*, *sep=\t'*)

Bases: object

Description

Stable periodic pattern mining aims to dicover all interesting patterns in a temporal database using three contraints minimum support, maximum period and maximum lability, that have support no less than the user-specified minimum support constraint and lability no greater than maximum lability.

Reference

Dao, H.N. et al. (2022). Towards Efficient Discovery of Stable Periodic Patterns in Big Columnar Temporal Databases. In: Fujita, H., Fournier-Viger, P., Ali, M., Wang, Y. (eds) Advances and Trends in Artificial Intelligence. Theory and Practices in Artificial Intelligence. IEA/AIE 2022. Lecture Notes in Computer Science(), vol 13343. Springer, Cham. https://doi.org/10. 1007/978-3-031-08530-7_70

Parameters

• iFile - str :

Name of the Input file to mine complete set of frequent pattern's

- oFile str : Name of the output file to store complete set of frequent patterns
- minSup str: Minimum number of frequent patterns to be included in the output file.
- maxLa float: Minimum number of ...
- maxPer float: Maximum number of frequent patterns to be included in the output file.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup: int or float or str

The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

maxPer

[int or float or str] The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

maxLa

[int or float or str] The user can specify maxLa either in count or proportion of database size. If the program detects the data type of maxLa is integer, then it treats maxLa is expressed in count. Otherwise, it will be treated as float. Example: maxLa=10 will be treated as integer, while maxLa=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

startMine()
Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

PeriodicFrequentOneItem()

Extracts the one-periodic-frequent patterns from database

updateDatabases()

Update the database by removing aperiodic items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

to convert the user specified value

Methods to execute code on terminal

Format:
(.venv) \$ python3 topk.py <inputFile> <outputFile> <minSup> <maxPer> <maxLa>
Example usage :
(.venv) \$ python3 topk.py sampleTDB.txt patterns.txt 0.3 0.4 0.3

Note: constraints will be considered in percentage of database transactions

Importing this algorithm into a python program

<pre>from PAMI.stablePeriodicFrequentPattern.basic import topk as alg</pre>
obj = alg.topk(iFile, minSup, maxPer, maxLa)
obj.startMine()
Patterns = obj.getPatterns()
<pre>print("Total number of Stable Periodic Frequent Patterns:", len(Patterns))</pre>
obj.save(oFile)
<pre>Df = obj.getPatternsAsDataFrame()</pre>
<pre>memUSS = obj.getMemoryUSS()</pre>
<pre>print("Total Memory in USS:", memUSS)</pre>
<pre>memRSS = obj.getMemoryRSS()</pre>
<pre>print("Total Memory in RSS", memRSS)</pre>
<pre>run = obj.getRuntime()</pre>
<pre>print("Total ExecutionTime in seconds:", run)</pre>

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

SPPList = {}

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of periodic-frequent patterns after completion of the mining process

Returns

returning periodic-frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine()

Mining process will start from this function

printResults()

This function is used to print the results

save(outFile)

Complete set of periodic-frequent patterns will be loaded in to an output file

Parameters
 outFile (csv file) - name of the output file

startMine()

Mining process will start from this function

SPPEclat

Bases: _stablePeriodicFrequentPatterns

Description

Stable periodic pattern mining aims to dicover all interesting patterns in a temporal database using three contraints minimum support, maximum period and maximum lability, that have support no less than the user-specified minimum support constraint and lability no greater than maximum lability.

Reference

Fournier-Viger, P., Yang, P., Lin, J. C.-W., Kiran, U. (2019). Discovering Stable Periodic-Frequent Patterns in Transactional Data. Proc. 32nd Intern. Conf. on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA AIE 2019), Springer LNAI, pp. 230-244

Parameters

- iFile str : Name of the Input file to mine complete set of stable periodic Frequent Pattern.
- oFile str : Name of the output file to store complete set of stable periodic Frequent Pattern.
- **minSup** float or int or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float
- itemSup int or float : Frequency of an item
- **maxLa** float : minimum loss of a pattern
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup

[int or float or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

maxPer

[int or float or str] The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

maxLa

[int or float or str] The user can specify maxLa either in count or proportion of database size. If the program detects the data type of maxLa is integer, then it treats maxLa is expressed in count. Otherwise, it will be treated as float. Example: maxLa=10 will be treated as integer, while maxLa=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

itemSetCount

[int] it represents the total no of patterns

finalPatterns

[dict] it represents to store the patterns

tidList

[dict] stores the timestamps of an item

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to an output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets()

Scan the database and store the items with their timestamps which are periodic frequent

calculateLa()

Calculates the support and period for a list of timestamps.

Generation()

Used to implement prefix class equivalence method to generate the periodic patterns recursively

Methods to execute code on terminal

```
Format:
(.venv) $ python3 basic.py <inputFile> <outputFile> <minSup> <maxPer> <maxLa>
Example usage:
(.venv) $ python3 basic.py sampleDB.txt patterns.txt 10.0 4.0 2.0
.. note:: constraints will be considered in percentage of database_
...transactions
```

Importing this algorithm into a python program

```
... code-block:: python
```

from PAMI.stablePeriodicFrequentPattern.basic import basic as alg

obj = alg.PFPECLAT("../basic/sampleTDB.txt", 5, 3, 3)

obj.startMine()

Patterns = obj.getPatterns()

print("Total number of Stable Periodic Frequent Patterns:", len(Patterns))

obj.save("patterns")

Df = obj.getPatternsAsDataFrame()

memUSS = obj.getMemoryUSS()

print("Total Memory in USS:", memUSS)

memRSS = obj.getMemoryRSS()

print("Total Memory in RSS", memRSS)

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function :return: returning RSS memory consumed by the mining process :rtype: float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function :return: returning USS memory consumed by the mining process :rtype: float

getPatterns()

Function to return the set of stable periodic-frequent patterns after completion of the mining process

Returns

returning stable periodic-frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine()

Method to start the mining of patterns

printResults()

This function is used to print the results

save(outFile)

Complete set of periodic-frequent patterns will be loaded in to an output file

Parameters

outFile (*csv file*) – name of the output file

startMine()

Method to start the mining of patterns

2.6.2 TopK

TSPIN

class PAMI.stablePeriodicFrequentPattern.topK.TSPIN.**TSPIN**(*iFile*, *maxPer*, *maxLa*, *k*, *sep*=\t') Bases: _stablePeriodicFrequentPatterns

Description

TSPIN is an algorithm to discover top stable periodic-frequent patterns in a transactional database.

Reference

Fournier-Viger, P., Wang, Y., Yang, P. et al. TSPIN: mining top-k stable periodic patterns. Appl Intell 52, 6917–6938 (2022). https://doi.org/10.1007/s10489-020-02181-6

Parameters

- iFile str : Name of the Input file to mine complete set of frequent pattern's
- oFile str : Name of the output file to store complete set of frequent patterns
- maxPer float: Maximum number of frequent patterns to be included in the output file.
- maxLa str: Maximum number of frequent patterns to be included in the output file.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

maxPer

[int or float or str] The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

maxLa

[int or float or str] The user can specify maxLa either in count or proportion of database size. If the program detects the data type of maxLa is integer, then it treats maxLa is expressed in count. Otherwise, it will be treated as float. Example: maxLa=10 will be treated as integer, while maxLa=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

PeriodicFrequentOneItem()

Extracts the one-periodic-frequent patterns from database

updateDatabases()

Update the database by removing aperiodic items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

to convert the user specified value

Methods to execute code on terminal

Format:

```
>>> python3 TSPIN.py <inputFile> <outputFile> <maxPer> <maxLa>
```

Example:

>>> python3 TSPIN.py sampleTDB.txt patterns.txt 0.3 0.4 0.6

Note: maxPer, maxLa and k will be considered in percentage of database transactions

Importing this algorithm into a python program

```
from PAMI.stablePeriodicFrequentPattern.basic import TSPIN as alg
obj = alg.TSPIN(iFile, maxPer, maxLa, k)
obj.startMine()
stablePeriodicFrequentPatterns = obj.getPatterns()
print("Total number of Periodic Frequent Patterns:",..
...len(stablePeriodicFrequentPatterns))
obj.savePatterns(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \rightarrow \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns() \rightarrow dict

Function to send the set of periodic-frequent patterns after completion of the mining process

Returns

returning periodic-frequent patterns

Return type

dict

$\texttt{getPatternsAsDataFrame()} \rightarrow \texttt{DataFrame}$

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type pd.DataFrame

$\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

$\texttt{printResults()} \rightarrow \text{None}$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of periodic-frequent patterns will be loaded in to an output file

Parameters outFile (file) – name of the output file

$\texttt{startMine()} \to \text{None}$

Mining process will start from this function

2.7 Recurring Pattern Mining

Recurring patterns refer to patterns within a dataset that demonstrate periodic behavior occurring only at specific time intervals within a series. The goal of recurring pattern mining is to discover meaningful and potentially predictive patterns that can provide insights into the underlying behavior of the time series data

Applications: Anomaly Detection, Predictive Maintenance, Financial Forecasting.

Recurring patterns refer to patterns within a dataset that demonstrate periodic behavior occurring only at specific time intervals within a series. The goal of recurring pattern mining is to discover meaningful and potentially predictive patterns that can provide insights into the underlying behavior of the time series data

Applications: Anomaly Detection, Predictive Maintenance, Financial Forecasting.

2.7.1 Basic

RPGrowth

class PAMI.recurringPattern.basic.RPGrowth.**RPGrowth**(*iFile*, *maxPer*, *minPS*, *minRec*, *sep*=t')

Bases: _recurringPatterns

Description

RPGrowth is one of the fundamental algorithm to discover recurring patterns in a transactional database.

Reference

R. Uday Haichuan Shang[†], Masashi Toyoda† Masaru Kitsure-Kiran[†], and Series, https://www.tkl.iis.ugawa† Discovering Recurring Patterns in Time tokyo.ac.jp/new/uploads/publication_file/file/693/Paper%2023.pdf

Parameters

- iFile str : Name of the Input file to mine complete set of Recurring patterns
- oFile str : Name of the output file to store complete set of Recurring patterns
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.
- **minPs** str : It could potentially represent a minimum parallelism percentage or some other value related to parallel processing.
- maxPer float : minRec It represent a maximum percentage or some other numeric value.
- **minRec** str : It could represent a minimum recommended value or some other string-based setting.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

maxPer

[int or float or str] The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

minPS

[int or float or str] The user can specify minPS either in count or proportion of database size. If the program detects the data type of minPS is integer, then it treats minPS is expressed in count. Otherwise, it will be treated as float. Example: minPS=10 will be treated as integer, while minPS=10.0 will be treated as float

minRec

[int or float or str] The user has to specify minRec in count.

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

OneItems()

Extracts the possible recurring items of size one from database

updateDatabases()

Update the database by removing non recurring items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

to convert the user specified value

Methods to execute code on terminal

Importing this algorithm into a python program



(continues on next page)

(continued from previous page)

memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by C. Saideep under the supervision of Professor Rage Uday Kiran.

Mine()

Mining process will start from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of periodic-frequent patterns after completion of the mining process

Returns

returning periodic-frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

printResults()

To print all the results of execution

save(outFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

Parameters

outFile (*file*) – name of the output file.

startMine()

Mining process will start from this function

CHAPTER

THREE

GEO-REFERENCED PATTERN MINING

A geo-referenced database represents the data gathered by a set of fixed sensors observing a particular phenomenon over a time period. It is a combination of spatial database and transactional/temporal/utility database .

Types of Geo-referenced databases

- · Geo-referenced transactional databases
- · Geo-referenced temporal databases
- Geo-referenced utility database

Basic topics

- · Location/spatial database
- Neighborhood database
- 1. Geo-referenced transactional database

A transactional database is said to be a geo-referenced transactional database if it contains spatial items. The format of this database is similar to that of transactional database . An example of a geo-referenced transactional database is as follows:

TID	Items
1	Point(0 0) Point(0 1) Point(1 0)
2	Point(0 0) Point(0 2) Point(5 0)
3	Point(5 0)
4	Point(4 0) Point(5 0)

Note: The rules to create a geo-referenced transactional database are same as the rules to create a transactional database. In other words, the format of creating a transaction in a geo-referential database is:

>>> spatialItem1<sep>spatialItem2<sep>...<sep>spatialItemN

An example:

Point(0 0) Point(0 1) Point(1 0) Point(0 0) Point(0 2) Point(5 0)

Point(50)

Point(4 0) Point(5 0)

2. Geo-referential temporal database

A temporal database is said to be a geo-referential temporal database if it contains spatial items. The format of this database is similar to that of temporal database . An example of a geo-referential temporal database is as follows:

TID	Timestamp	Items
1	1	Point(0 0) Point(0 1) Point(1 0)
2	2	Point(0 0) Point(0 2) Point(5 0)
3	4	Point(5 0)
4	5	Point(4 0) Point(5 0)

Note: The rules to create geo-referential temporal database are same as the rules to create a temporal database. In other words, the format to create geo-referential temporal database is as follows:

An example:

- 1 Point(0 0) Point(0 1) Point(1 0)
- 2 Point(0 0) Point(0 2) Point(5 0)
- 4 Point(5 0)
- 5 Point(4 0) Point(5 0)
- 3. Geo-referential utility database

A utility database is said to be a geo-referential utility database if it contains spatial items. The format of this database is similar to that of utility database . An example of a geo-referential utility database is as follows:

TID	Transactions (items and their prices)
1	(Point(0 0),100) (Point(0 1),42) (Point(1 0), 20)
2	(Point(0 0), 100) (Point(0 2), 10) (Point(5 0), 30)
3	(Point(5 0), 30)
4	(Point(4 0),30), (Point(5 0),40)

Note: The rules to create geo-referential utility database are same as the rules to create a utility database. In other words, the format to create geo-referential utility database is as follows:

An example:

1 Point(0 0) Point(0 1) Point(1 0):162:100 42 20

2 Point(0 0) Point(0 2) Point(5 0):140:100 10 30

4 Point(5 0):30:30

5 Point(4 0) Point(5 0):70:30 40

3.1 Geo-referenced Frequent Pattern Mining

Geo-referenced frequent pattern mining is the process of discovering frequent patterns, associations, or relationships among spatially and temporally referenced data. It involves analyzing datasets that contain geographic coordinates, timestamps, and possibly other attributes related to spatial and temporal events.

Applications: Location-Based Services, Environmental Monitoring and Conservation, Tourism and Hospitality.

Geo-referenced frequent pattern mining is the process of discovering frequent patterns, associations, or relationships among spatially and temporally referenced data. It involves analyzing datasets that contain geographic coordinates, timestamps, and possibly other attributes related to spatial and temporal events.

Applications: Location-Based Services, Environmental Monitoring and Conservation, Tourism and Hospitality.

3.1.1 Basic

SpatialECLAT

class PAMI.georeferencedFrequentPattern.basic.SpatialECLAT.**SpatialECLAT**(*iFile*, *nFile*, *minSup*, $sep = \forall t'$)

Bases: _spatialFrequentPatterns

Description

Spatial Eclat is a Extension of ECLAT algorithm, which stands for Equivalence Class Clustering and bottom-up Lattice Traversal. It is one of the popular methods of Association Rule mining. It is a more efficient and scalable version of the Apriori algorithm.

Reference

Rage, Uday & Fournier Viger, Philippe & Zettsu, Koji & Toyoda, Masashi & Kitsuregawa, Masaru. (2020). Discovering Frequent Spatial Patterns in Very Large Spatiotemporal Databases.

Parameters

- iFile str : Name of the Input file to mine complete set of Geo-referenced frequent patterns
- oFile str : Name of the output file to store complete set of Geo-referenced frequent patterns
- **minSup** int or float or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.
- **maxPer** float : The user can specify maxPer in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count.
- **nFile** str : Name of the input file to mine complete set of Geo-referenced frequent patterns
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

nFile

[str] Name of Neighbourhood file name

minSup

[int or float or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

finalPatterns

[dict] Storing the complete set of patterns in a dictionary variable

oFile

[str] Name of the output file to store complete set of frequent patterns

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the complete set of transactions available in the input database/file

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(iFileName)

Storing the complete transactions of the database/input file in a database variable

frequentOneItem()

Generating one frequent patterns

dictKeysToInt(iList)

Converting dictionary keys to integer elements

eclatGeneration(cList)

It will generate the combinations of frequent items

generateSpatialFrequentPatterns(tidList) It will generate the combinations of frequent items from a list of items

convert(value)

To convert the given user specified value

```
getNeighbourItems(keySet)
```

A function to get common neighbours of a itemSet

mapNeighbours(file)

A function to map items to their neighbours

Executing the code on terminal :

Format:

```
(.venv) $ python3 SpatialECLAT.py <inputFile> <outputFile> <neighbourFile> <minSup>
```

Example Usage:

(.venv) \$ python3 SpatialECLAT.py sampleTDB.txt output.txt sampleN.txt 0.5

Note: minSup will be considered in percentage of database transactions

Sample run of importing the code :

```
from PAMI.georeferencedFrequentPattern.basic import SpatialECLAT as alg
obj = alg.SpatialECLAT("sampleTDB.txt", "sampleN.txt", 5)
obj.mine()
spatialFrequentPatterns = obj.getPatterns()
print("Total number of Spatial Frequent Patterns:", len(spatialFrequentPatterns))
obj.save("outFile")
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by B.Sai Chitra under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

mine()

Frequent pattern mining process will start from here

printResults()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to a output file

Parameters
 outFile (csv file) - name of the output file

startMine()

Frequent pattern mining process will start from here

FSPGrowth

3.2 Geo-referenced Periodic Frequent Pattern Mining

Geo-referenced periodic frequent patterns describe consistent patterns of activity or events that occur in specific geographic areas at regular time intervals. These patterns may reveal recurring trends, behaviors, or phenomena in spatially and temporally referenced data, such as movement patterns, environmental changes, or human activities

Applications: Transportation and Logistics, Environmental Monitoring and Conservation, Urban Planning and Infrastructure Management.

Geo-referenced periodic frequent patterns describe consistent patterns of activity or events that occur in specific geographic areas at regular time intervals. These patterns may reveal recurring trends, behaviors, or phenomena in spatially and temporally referenced data, such as movement patterns, environmental changes, or human activities

Applications: Transportation and Logistics, Environmental Monitoring and Conservation, Urban Planning and Infrastructure Management.

3.2.1 Basic

GPFPMiner

class PAMI.geoReferencedPeriodicFrequentPattern.basic.GPFPMiner.**GPFPMiner**(*iFile*, *nFile*,

 $minSup, maxPer, sep=\langle t' \rangle$

Bases: _geoReferencedPeriodicFrequentPatterns

Description

GPFPMiner is an Extension of ÉCLAT algorithm, which stands for Equivalence Class Clustering and

bottom-up Lattice Traversal to mine the geo referenced periodic frequent patterns.

Reference

Parameters

- **iFile** str Name of the Input file to mine complete set of Geo-referenced periodic frequent patterns
- **oFile** str Name of the output file to store complete set of Geo-referenced periodic frequent patterns
- **minSup** int or float or str The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.
- **maxPer** float The user can specify maxPer in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count.
- **nFile** str Name of the input file to mine complete set of Geo-referenced periodic frequent patterns
- **sep** str This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

nFile

[str] Name of Neighbourhood file name

minSup

[float or int or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

maxPer

[float or int or str] The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

finalPatterns

[dict] Storing the complete set of patterns in a dictionary variable

oFile

[str] Name of the output file to store complete set of frequent patterns

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the complete set of transactions available in the input database/file

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrames()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(iFileName)

Storing the complete transactions of the database/input file in a database variable

frequentOneItem()

Generating one frequent patterns

convert(value)

To convert the given user specified value

getNeighbourItems(keySet)

A function to get common neighbours of a itemSet

mapNeighbours(file)

A function to map items to their neighbours

Executing the code on terminal :

Format:

Example Usage:

(.venv) \$ python3 GPFPMiner.py sampleTDB.txt output.txt sampleN.txt 0.5 0.3

Note: minSup & maxPer will be considered in percentage of database transactions

Sample run of importing the code :



(continued from previous page)

```
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.RaviKumar under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function :return: returning RSS memory consumed by the mining process :rtype: float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

mapNeighbours()

A function to map items to their Neighbours

mine()

Frequent pattern mining process will start from here

printResults()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to a output file

Parameters

outFile (*csv file*) – name of the output file

startMine()

Frequent pattern mining process will start from here

3.3 Geo-referenced Partial Periodic Pattern Mining

Geo-referenced partial periodic frequent pattern mining is a data mining technique that aims to discover recurring spatial-temporal patterns in datasets where events occur periodically but may not always cover the entire time period of interest. In other words, it focuses on identifying patterns that exhibit periodicity in both space and time, but allow for variations or partial occurrences within each period.

Applications: Agricultural Monitoring and Crop Management, Public Health Surveillance, Environmental Monitoring and Disaster Management.

Geo-referenced partial periodic frequent pattern mining is a data mining technique that aims to discover recurring spatial-temporal patterns in datasets where events occur periodically but may not always cover the entire time period of interest. In other words, it focuses on identifying patterns that exhibit periodicity in both space and time, but allow for variations or partial occurrences within each period.

Applications: Agricultural Monitoring and Crop Management, Public Health Surveillance, Environmental Monitoring and Disaster Management.

3.3.1 Basic

STEclat

class PAMI.georeferencedPartialPeriodicPattern.basic.STEclat.**STEclat**(*iFile*, *nFile*, *minPS*, *maxIAT*, *sep*=\t')

Bases: _partialPeriodicSpatialPatterns

Description

STEclat is one of the fundamental algorithm to discover georefereneced partial periodic-frequent patterns in a transactional database.

Reference

R. Uday Kiran, C. Saideep, K. Zettsu, M. Toyoda, M. Kitsuregawa and P. Krishna Reddy, "Discovering Partial Periodic Spatial Patterns in Spatiotemporal Databases," 2019 IEEE International

Conference on Big Data (Big Data), 2019, pp. 233-238, doi: 10.1109/Big-Data47090.2019.9005693.

Parameters

• **iFile** – str : Name of the Input file to mine complete set of Geo-referenced Partial Periodic patterns

- oFile str : Name of the output file to store complete set of Geo-referenced Partial Periodic patterns
- **minPS** int or float or str : The user can specify minPS either in count or proportion of database size. If the program detects the data type of minPS is integer, then it treats minPS is expressed in count. Otherwise, it will be treated as float.
- **maxIAT** int or float or str : The user can specify maxIAT either in count or proportion of database size. If the program detects the data type of maxIAT is integer, then it treats maxIAT is expressed in count. Otherwise, it will be treated as float.
- nFile str : Name of the input file to mine complete set of Geo-referenced Partial Periodic patterns
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

nFile

[str] Name of Neighbourhood file name

maxIAT

[float or int or str] The user can specify maxIAT either in count or proportion of database size. If the program detects the data type of maxIAT is integer, then it treats maxIAT is expressed in count. Otherwise, it will be treated as float. Example: maxIAT=10 will be treated as integer, while maxIAT=10.0 will be treated as float

minPS

[float or int or str] The user can specify minPS either in count or proportion of database size. If the program detects the data type of minPS is integer, then it treats minPS is expressed in count. Otherwise, it will be treated as float. Example: minPS=10 will be treated as integer, while minPS=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

finalPatterns

[dict] Storing the complete set of patterns in a dictionary variable

oFile

[str] Name of the output file to store complete set of frequent patterns

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the complete set of transactions available in the input database/file

Methods
mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrames()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(iFileName)

Storing the complete transactions of the database/input file in a database variable

frequentOneItem()

Generating one frequent patterns

convert(value):

To convert the given user specified value

getNeighbourItems(keySet)

A function to get common neighbours of a itemSet

mapNeighbours(file)

A function to map items to their neighbours

Executing the code on terminal :

Format:

Example Usage:

(.venv) \$ python3 STEclat.py sampleTDB.txt output.txt sampleN.txt 0.2 0.5

Note: maxIAT & minPS will be considered in percentage of database transactions

Sample run of importing the code :

```
import PAMI.georeferencedPartialPeriodicPattern.STEclat as alg
obj = alg.STEclat("sampleTDB.txt", "sampleN.txt", 3, 4)
obj.mine()
partialPeriodicSpatialPatterns = obj.getPatterns()
print("Total number of Periodic Spatial Frequent Patterns:",
olen(partialPeriodicSpatialPatterns))
obj.save("outFile")
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P. Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

mapNeighbours()

A function to map items to their Neighbours

mine()

Frequent pattern mining process will start from here

printResults()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (csv file) – name of the output file

startMine()

Frequent pattern mining process will start from here

UTILITY PATTERN MINING

A transactional/temporal database represents a binary database. It is because the items in these databases have values either 1 or 0. In contrast, a utility database is a non-binary database. In fact, a utility database is a quantitative database containing items and their utility values.

Utility databases are naturally produced by the real-world applications. Henceforth, most forms of the databases, such as transactional and temporal databases, are generated from utility databases.

In the utility database, the items have external utility values and internal utility values. External utility values, like prices of items in a supermarket, do not vary in the entire data. Internal utility values, like the number of items purchased by each customer, vary for every transaction in the database. The utility of an item in a transaction represents the product of its internal and external utility values.

Types

- Utility transactional databases
- Utility temporal databases

Utility transactional databases

Description

A utility transactional database consists of a transactional identifier (tid), items, and their corresponding utility values in a transaction. A sample utility transactional database generated from the set of items, I={Bread, Jam, Butter, Pen, Books, Bat}, is shown in below table:

TID	Transactions (items and their prices)
1	(Bread,1\$), (Jam,2\$), (Butter, 1.5\$)
2	(Bat, 100\$), (Ball, 10\$)
3	(Pen, 2\$), (Book, 5\$)

Format of a utility transactional database

The utility transactional database must exist in the following format:

The 'total utility' represents the total utility value of items in a transaction.

Rules to create a utility transactional database

• The default separator, i.e., , used in PAMI is tab space (or t). However, the users can override the default separator with their choice. Since spatial objects, such as Point, Line, and Polygon, are repre-

sented using space and comma, usage of tab space facilitates us to effectively distinguish the spatial objects.

• Items, total utility, and individual utilities of the items within a transaction have to be seperated by the symbol ':'

An example:

Bread Jam Butter:	4.5:1 2 1.5
Bat Ball:	110:100 10
Pen Book: 7:2 5	

Utility temporal databases

Description

A utility temporal database consists of timestamp, tid, items, and their corresponding utility values. A sample utility temporal database generated from the set of items, I={Bread, Jam, Butter, Pen, Books, Bat}, is shown in below table:

Format of a utility temporal database

The utility temporal database must exist in the following

Format:

The 'total utility' represents the total utility value of items in a transaction.

Rules to create a utility temporal database

- The default separator, i.e., , used in PAMI is tab space (or t). However, the users can override the default separator with their choice. Since spatial objects, such as Point, Line, and Polygon, are represented using space and comma, usage of tab space facilitates us to effectively distinguish the spatial objects.
- Timestamp, items, total utility, and individual utilities of the items within a transaction have to be seperated by the symbol ':'

Example:

1	Bread Jam Butter: 4.5:1	2 1.5
2	Bat Ball: 110:100 10	
3	Pen Book:	7:2 5

4.1 High-Utility Pattern mining

The aim of high-utility pattern mining (HUPM) is to discover meaningful patterns in medical databases that contribute to maximizing the utility from the perspective of diagnosis. However, HUPM pays less attention to the interpretability and explainability of these patterns in medical decision-making scenarios.

Applications: Clinical Decision Support, Drug Prescription and Therapy Planning, Disease Diagnosis and Prediction.

The aim of high-utility pattern mining (HUPM) is to discover meaningful patterns in medical databases that contribute to maximizing the utility from the perspective of diagnosis. However, HUPM pays less attention to the interpretability and explainability of these patterns in medical decision-making scenarios.

Applications: Clinical Decision Support, Drug Prescription and Therapy Planning, Disease Diagnosis and Prediction.

4.1.1 Basic

RHUIM

class PAMI.relativeHighUtilityPattern.basic.RHUIM.**RHUIM**(*iFile: str, minUtil: int, minUR: float, sep:* $str = \t'$)

Bases: _utilityPatterns

Description

RHUIM algorithm helps us to mine Relative High Utility itemSets from transactional databases.

Reference

R. U. Kiran, P. Pallikila, J. M. Luna, P. Fournier-Viger, M. Toyoda and P. K. Reddy, "Discovering Relative High Utility Itemsets in Very Large Transactional Databases Using Null-Invariant Measure,"

2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 2021, pp. 252-262, doi: 10.1109/BigData52589.2021.9672064.

Parameters

- iFile str : Name of the Input file to mine complete set of Relative High Utility patterns
- oFile str : Name of the output file to store complete set of Relative High Utility patterns
- **minSup** float or int or str : minSup measure constraints the minimum number of transactions in a database where a pattern must appear Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.
- minUtil int : The minimum utility threshold.

Attributes

[file] Name of the input file to mine complete set of patterns

oFile

iFile

[file] Name of the output file to store complete set of patterns

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

minUtil

[int] The user given minUtil value

minUR

[float] The user given minUR value

relativeHighUtilityItemSets

[map] set of relative high utility itemSets

candidateCount

[int] Number of candidates

utilityBinArrayLU

[list] A map to hold the local utility values of the items in database

utilityBinArraySU

[list] A map to hold the subtree utility values of the items is database

oldNamesToNewNames

[list] A map which contains old names, new names of items as key value pairs

newNamesToOldNames

[list] A map which contains new names, old names of items as key value pairs

maxMemory

[float] Maximum memory used by this program for running

patternCount

[int] Number of RHUI's

itemsToKeep

[list] keep only the promising items i.e items that can extend other items to form RHUIs

itemsToExplore

[list] list of items that needs to be explored

Methods

startMine()
Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

backTrackingRHUIM(transactionsOfP, itemsToKeep, itemsToExplore, prefixLength) A method to mine the RHUIs Recursively

useUtilityBinArraysToCalculateUpperBounds(transactionsPe, j, itemsToKeep)

A method to calculate the sub-tree utility and local utility of all items that can extend itemSet P and e

output(tempPosition, utility)

A method to output a relative-high-utility itemSet to file or memory depending on what the user chose

is_equal(transaction1, transaction2)

A method to Check if two transaction are identical

useUtilityBinArrayToCalculateSubtreeUtilityFirstTime(dataset) A method to calculate the sub tree utility values for single items

sortDatabase(self, transactions) A Method to sort transaction

- sort_transaction(self, trans1, trans2)
 A Method to sort transaction
- useUtilityBinArrayToCalculateLocalUtilityFirstTime(self, dataset) A method to calculate local utility values for single itemSets

Methods to execute code on terminal

```
Format:
(.venv) $ python3 RHUIM.py <inputFile> <outputFile> <minUtil> <sep>
Example usage:
(.venv) $ python3 RHUIM.py sampleTDB.txt output.txt 35 20
.. note:: minSup will be considered in times of minSup and count of_
→database transactions
```

Importing this algorithm into a python program

```
from PAMI.relativeHighUtilityPattern.basic import RHUIM as alg
obj=alg.RHUIM("input.txt", 35, 20)
obj.startMine()
frequentPatterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(frequentPatterns))
obj.savePatterns(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getmemoryUSS()
print("Total Memory in USS:", memUSS)
```

(continues on next page)

(continued from previous page)

```
memRSS = obj.getMemoryRSS()
```

```
print("Total Memory in RSS", memRSS)
```

run = obj.getRuntime()

```
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by Pradeep Pallikila under the supervision of Professor Rage Uday Kiran.

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \to \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$getPatterns() \rightarrow dict$

Function to send the set of patterns after completion of the mining process

Returns

returning patterns

Return type

dict

$\texttt{getPatternsAsDataFrame()} \rightarrow \texttt{DataFrame}$

Storing final patterns in a dataframe

Returns

returning patterns in a dataframe

Return type pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

$printResults() \rightarrow None$

This function is used to print the results :return: None

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (file) – name of the output file

Returns None

 $sortDatabase(transactions: list) \rightarrow None$

A Method to sort transaction

Attributes Parameters transactions (list) – transaction of items

Returns sorted transactions.

Return type Transactions or list

 $sort_transaction(trans1: _Transaction, trans2: _Transaction) \rightarrow int$

A Method to sort transaction

Attributes Parameters trans1 (Transaction) – the first transaction .

:param trans2:the second transaction. :type trans2: Transaction :return: sorted transaction. :rtype: Transaction

```
startMine() \rightarrow None
```

Mining process will start from this function :return: None

4.2 High-Utility Frequent Pattern Mining

High utility frequent pattern mining involves discovering patterns in transactional databases where each pattern consists of a set of items that occur frequently and contribute significantly to the overall utility of the dataset. These patterns are characterized by their high utility values, which reflect their importance or usefulness in the context of the application domain.

Applications: Market Basket Analysis, Healthcare Analytics, Web Usage Mining, Fraud Detection.

High utility frequent pattern mining involves discovering patterns in transactional databases where each pattern consists of a set of items that occur frequently and contribute significantly to the overall utility of the dataset. These patterns are characterized by their high utility values, which reflect their importance or usefulness in the context of the application domain.

Applications: Market Basket Analysis, Healthcare Analytics, Web Usage Mining, Fraud Detection.

4.2.1 Basic

HUFIM

Bases: _utilityPatterns

Description

HUFIM (High Utility Frequent Itemset Miner) algorithm helps us to mine High Utility Frequent ItemSets (HUFIs) from transactional databases.

Reference

Kiran, R.U., Reddy, T.Y., Fournier-Viger, P., Toyoda, M., Reddy, P.K., & Kitsuregawa, M. (2019). Efficiently Finding High Utility-Frequent Itemsets Using Cutoff and Suffix Utility. PAKDD 2019. DOI: 10.1007/978-3-030-16145-3_15

Parameters

- **iFile** str : Name of the Input file to mine complete set of Geo-referenced frequent sequence patterns
- **oFile** str : Name of the output file to store complete set of Geo-referenced frequent sequence patterns
- **minSup** int or float or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.
- minUtil int : The user given minUtil value.
- candidateCount int Number of candidates
- maxMemory int Maximum memory used by this program for running
- **nFile** str : Name of the input file to mine complete set of Geo-referenced frequent sequence patterns
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the input file to mine complete set of patterns

oFile

[file] Name of the output file to store complete set of patterns

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

minUtil

[int] The user given minUtil value

minSup

[float] The user given minSup value

highUtilityFrequentItemSets: map

set of high utility frequent itemSets

candidateCount: int

Number of candidates

utilityBinArrayLU: list

A map to hold the local utility values of the items in database

utilityBinArraySU: list

A map to hold the subtree utility values of the items is database

oldNamesToNewNames: list

A map which contains old names, new names of items as key value pairs

newNamesToOldNames: list

A map which contains new names, old names of items as key value pairs

singleItemSetsSupport: map

A map which maps from single itemsets (items) to their support

singleItemSetsUtility: map

A map which maps from single itemsets (items) to their utilities

maxMemory: float

Maximum memory used by this program for running

patternCount: int Number of RHUI's

itemsToKeep: list

keep only the promising items i.e items that can extend other items to form RHUIs

itemsToExplore: list

list of items that needs to be explored

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

backTrackingHUFIM (transactionsOfP, itemsToKeep, itemsToExplore, prefixLength)

A method to mine the RHUIs Recursively

```
useUtilityBinArraysToCalculateUpperBounds(transactionsPe, j, itemsToKeep)
A method to calculate the sub-tree utility and local utility of all items that can extend itemSet
P and e
```

output(tempPosition, utility)

A method to output a relative-high-utility itemSet to file or memory depending on what the user chose

isEqual(transaction1, transaction2)

A method to Check if two transaction are identical

use Utility Bin Array To Calculate Subtree Utility First Time (dataset)

A method to calculate the sub tree utility values for single items

sortDatabase(self, transactions)

A Method to sort transaction

sortTransaction(self, trans1, trans2)

A Method to sort transaction

useUtilityBinArrayToCalculateLocalUtilityFirstTime(self, dataset) A method to calculate local utility values for single itemSets

Executing the code on terminal

Format:
(.venv) \$ python3 HUFIM.py <inputFile> <outputFile> <minUtil> <sep>
Example Usage:
(.venv) \$ python3 HUFIM.py sampleTDB.txt output.txt 35 20
(.venv) \$ python3 HUFIM.py sampleTDB.txt output.txt 35 20

Note: minSup will be considered in percentage of database transactions

Sample run of importing the code



(continues on next page)

(continued from previous page)

```
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
```

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by pradeep pallikila under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$\texttt{getMemoryUSS()} \to \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$getPatterns() \rightarrow Dict[str, List[int | float]]$

Function to send the set of patterns after completion of the mining process

Returns

returning patterns

Return type

dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final patterns in a dataframe

Returns

returning patterns in a dataframe

Return type

pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float **mine()** \rightarrow None High Utility Frequent Pattern mining start here Returns None $printResults() \rightarrow None$ This function is used to print the results **save**(*outFile: str*) \rightarrow None Complete set of frequent patterns will be loaded in to an output file **Parameters** outFile (csv file) - name of the output file Returns None $startMine() \rightarrow None$ High Utility Frequent Pattern mining start here Returns

None

4.3 High-Utility Geo-referenced Frequent Pattern Mining

High utility georeferenced frequent pattern mining involves the discovery of spatial patterns in georeferenced datasets, where these patterns represent combinations of spatially distributed items or events that occur frequently and are associated with high utility values. These patterns are characterized by their high utility, reflecting their importance or usefulness in the context of the application domain.

Applications: Location-Based Services (LBS), Urban Planning and Development, Environmental Monitoring.

High utility georeferenced frequent pattern mining involves the discovery of spatial patterns in georeferenced datasets, where these patterns represent combinations of spatially distributed items or events that occur frequently and are associated with high utility values. These patterns are characterized by their high utility, reflecting their importance or usefulness in the context of the application domain.

Applications: Location-Based Services (LBS), Urban Planning and Development, Environmental Monitoring.

4.3.1 Basic

SHUFIM

Bases: _utilityPatterns

Description

Spatial High Utility Frequent ItemSet Mining (SHUFIM) aims to discover all itemSets in a spatioTemporal database that satisfy the user-specified minimum utility, minimum support and maximum distance constraints

Reference

10.1007/978-3-030-37188-3_17

Parameters

- iFile str : Name of the Input file to mine complete set of Geo-referenced frequent sequence patterns
- oFile str : Name of the output file to store complete set of Geo-referenced frequent sequence patterns
- **minSup** int or float or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.
- minUtil int : The user given minUtil value.
- candidateCount int Number of candidates
- maxMemory int Maximum memory used by this program for running
- nFile str : Name of the input file to mine complete set of Geo-referenced frequent sequence patterns
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the input file to mine complete set of frequent patterns

nFile

[file] Name of the Neighbours file that contain neighbours of items

oFile

[file] Name of the output file to store complete set of frequent patterns

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

minUtil

[int] The user given minUtil

minSup

[float] The user given minSup value

highUtilityFrequentSpatialItemSets: map

set of high utility itemSets

candidateCount: int

Number of candidates

utilityBinArrayLU: list

A map to hold the pmu values of the items in database

utilityBinArraySU: list

A map to hold the subtree utility values of the items is database

oldNamesToNewNames: list

A map to hold the subtree utility values of the items is database

newNamesToOldNames: list

A map to store the old name corresponding to new name

Neighbours

[map] A dictionary to store the neighbours of a item

maxMemory: float

Maximum memory used by this program for running

patternCount: int

Number of SHUFI's (Spatial High Utility Frequent Itemsets)

itemsToKeep: list

keep only the promising items ie items whose supersets can be required patterns

itemsToExplore: list

keep items that subtreeUtility grater than minUtil

:Methods :

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

calculateNeighbourIntersection(self, prefixLength)

A method to return common Neighbours of items

backtrackingEFIM(transactionsOfP, itemsToKeep, itemsToExplore, prefixLength) A method to mine the SHUIs Recursively

useUtilityBinArraysToCalculateUpperBounds(transactionsPe, j, itemsToKeep,

neighbourhoodList)

A method to calculate the sub-tree utility and local utility of all items that can extend itemSet P and e

output(tempPosition, utility)

A method ave a high-utility itemSet to file or memory depending on what the user chose

isEqual(transaction1, transaction2)

A method to Check if two transaction are identical

intersection(lst1, lst2)

A method that return the intersection of 2 list

useUtilityBinArrayToCalculateSubtreeUtilityFirstTime(dataset)

Scan the initial database to calculate the subtree utility of each items using a utility-bin array

sortDatabase(self, transactions)

A Method to sort transaction in the order of PMU

sortTransaction(self, trans1, trans2)

A Method to sort transaction in the order of PMU

useUtilityBinArrayToCalculateLocalUtilityFirstTime(self, dataset) A method to scan the database using utility bin array to calculate the pmus

Executing the code on terminal :

```
Format:
```

Example Usage:

(.venv) \$ python3 SHUFIM.py sampleTDB.txt output.txt sampleN.txt 35 20

Note: minSup will be considered in percentage of database transactions

Sample run of importing the code:

```
from PAMI.highUtilityGeoreferencedFrequentPattern.basic import SHUFIM as alg
obj=alg.SHUFIM("input.txt", "Neighbours.txt", 35,20)
obj.mine()
patterns = obj.getPatterns()
print("Total number of Spatial high utility frequent Patterns:", len(patterns))
obj.save("output")
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
```

(continued from previous page)

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by Pradeep Pallikila under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

not

getPatterns()

Function to send the set of patterns after completion of the mining process

Returns

returning patterns

Return type

dict

getPatternsAsDataFrame()

Storing final patterns in a dataframe :return: returning patterns in a dataframe :rtype: pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

mine()

High Utility Frequent Pattern mining start here

printResults()

This function is used to print the results

save(outFile)

Complete set of patterns will be loaded in to an output file

Parameters

outFile (csv file) - name of the output file

startMine()

High Utility Frequent Pattern mining start here

PAMI.highUtilityGeoreferencedFrequentPattern.basic.SHUFIM.main()

4.4 High-Utility Spatial Pattern Mining

High utility spatial pattern mining involves the identification of patterns in spatial datasets where each pattern has high utility, reflecting its significance or importance in the context of the application domain. These patterns consist of spatially distributed items or events that occur frequently and contribute significantly to a predefined utility measure.

Applications: Resource Management, Precision Agriculture, Emergency Response and Disaster Management.

High utility spatial pattern mining involves the identification of patterns in spatial datasets where each pattern has high utility, reflecting its significance or importance in the context of the application domain. These patterns consist of spatially distributed items or events that occur frequently and contribute significantly to a predefined utility measure.

Applications: Resource Management, Precision Agriculture, Emergency Response and Disaster Management.

4.4.1 Basic

HDSHUIM

class PAMI.highUtilitySpatialPattern.basic.HDSHUIM.HDSHUIM(*iFile: str*, *nFile: str*, *minUtil: int, sep:* $str = \t'$)

Bases: _utilityPatterns

Description

Spatial High Utility ItemSet Mining (SHUIM) [3] is an important model in data mining with many real-world applications. It involves finding all spatially interesting itemSets having high value in a quantitative spatio temporal database.

Reference

P. Pallikila et al., "Discovering Top-k Spatial High Utility Itemsets in Very Large Quantitative Spatiotemporal databases," 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 2021, pp. 4925-4935, doi: 10.1109/BigData52589.2021.9671912.

Parameters

- iFile str : Name of the Input file to mine complete set of High Utility Spatial patterns
- oFile str : Name of the output file to store complete set of High Utility Spatial patterns
- **minSup** int or float or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.
- **maxPer** float : The user can specify maxPer in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count.
- minUtil int : Minimum utility threshold given by User
- nFile str : Name of the input file to mine complete set of High Utility Spatial patterns
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Name of the input file to mine complete set of frequent patterns

oFile

[str] Name of the output file to store complete set of frequent patterns

nFile: str

Name of Neighbourhood items file

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

minUtil

[int] The user given minUtil

mapFMAP: list

EUCS map of the FHM algorithm

candidates: int candidates generated

huiCnt: int

huis created

neighbors: map keep track of neighbours of elements

mapOfPMU: map

a map to keep track of Probable Maximum utility(PMU) of each item

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

constructCUL(x, compactUList, st, minUtil, length, exNeighbours) A method to construct CUL's database

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

- **Explore_SearchTree(prefix, uList, exNeighbours, minUtil)** A method to find all high utility itemSets
- updateClosed(x, compactUList, st, exCul, newT, ex, eyTs, length) A method to update closed values
- saveItemSet(prefix, prefixLen, item, utility)
 A method to save itemSets
- updateElement(z, compactUList, st, exCul, newT, ex, duPrevPos, eyTs) A method to updates vales for duplicates

Executing the code on terminal:

```
Format:
```

Example Usage:

(.venv) \$ python3 HDSHUIM.py sampleTDB.txt output.txt sampleN.txt 35 ','

Note: minSup will be considered in percentage of database transactions

Sample run of importing the code:

```
from PAMI.highUtilityGeoreferencedFrequentPattern.basic import HDSHUIM as alg
obj=alg.HDSHUIM("input.txt","Neighbours.txt",35)
obj.mine()
Patterns = obj.getPatterns()
print("Total number of Spatial High-Utility Patterns:", len(Patterns))
obj.save("output")
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
(continues on next page)
```

(continued from previous page)

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by B.Sai Chitra under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS() \rightarrow float

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

$getPatterns() \rightarrow Dict[str, str]$

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

getPatternsAsDataFrame() \rightarrow Dict[str, str]

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None

main program to start the operation

$printResults() \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (*csv file*) – name of the output file

Returns None

startMine() \rightarrow None main program to start the operation

SHUIM

class PAMI.highUtilitySpatialPattern.basic.SHUIM.**SHUIM**(*iFile: str, nFile: str, minUtil: int, sep: str* = $\langle t' \rangle$

Bases: _utilityPatterns

Description

Spatial High Utility itemSet Mining (SHUIM) aims to discover all itemSets in a spatioTemporal database that satisfy the user-specified minimum utility and maximum distance constraints

Reference

Rage, Uday & Veena, Pamalla & Penugonda, Ravikumar & Raj, Bathala & Dao, Minh & Zettsu, Koji & Bommisetti, Sai. (2023). HDSHUI-miner: a novel algorithm for discovering spatial highutility itemsets in high-dimensional spatiotemporal databases. Applied Intelligence. 53. 1-26. 10.1007/s10489-022-04436-w.

Parameters

- iFile str : Name of the Input file to mine complete set of High Utility Spatial patterns
- oFile str : Name of the output file to store complete set of High Utility Spatial patterns
- **minSup** int or float or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.
- **maxPer** float : The user can specify maxPer in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count.
- minUtil int : Minimum utility threshold given by User
- maxMemory int : Maximum memory used by this program for running
- **candidateCount** int : Number of candidates to consider when calculating a high utility spatial pattern
- nFile str : Name of the input file to mine complete set of High Utility Spatial patterns
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the input file to mine complete set of frequent patterns

nFile

[file] Name of the Neighbours file that contain neighbours of items

oFile

[file] Name of the output file to store complete set of frequent patterns

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

minUtil

[int] The user given minUtil

highUtilityItemSets: map set of high utility itemSets

candidateCount: int

Number of candidates

utilityBinArrayLU: list

A map to hold the pmu values of the items in database

utilityBinArraySU: list

A map to hold the subtree utility values of the items is database

oldNamesToNewNames: list

A map to hold the subtree utility values of the items is database

newNamesToOldNames: list

A map to store the old name corresponding to new name

Neighbours

[map] A dictionary to store the neighbours of a item

maxMemory:Maximum memory used by this program for running patternCount: int

Number of SHUI's

itemsToKeep: list

keep only the promising items ie items having twu >= minUtil

itemsToExplore: list

keep items that subtreeUtility grater than minUtil

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

calculateNeighbourIntersection(self, prefixLength)

A method to return common Neighbours of items

backtrackingEFIM(transactionsOfP, itemsToKeep, itemsToExplore, prefixLength) A method to mine the SHUIs Recursively

useUtilityBinArraysToCalculateUpperBounds(transactionsPe, j, itemsToKeep,

neighbourhoodList)

A method to calculate the sub-tree utility and local utility of all items that can extend itemSet P and e

output(tempPosition, utility)

A method ave a high-utility itemSet to file or memory depending on what the user chose

_isEqual(transaction1, transaction2)

A method to Check if two transaction are identical

intersection(lst1, lst2)

A method that return the intersection of 2 list

useUtilityBinArrayToCalculateSubtreeUtilityFirstTime(dataset)

Scan the initial database to calculate the subtree utility of each items using a utility-bin array

sortDatabase(self, transactions)

A Method to sort transaction in the order of PMU

sort_transaction(self, trans1, trans2)

A Method to sort transaction in the order of PMU

useUtilityBinArrayToCalculateLocalUtilityFirstTime(self, dataset)

A method to scan the database using utility bin array to calculate the pmus

Executing the code on terminal:

```
Format:
```

(.venv) \$ python3 SHUIM.py <inputFile> <outputFile> <Neighbours> <minUtil> <sep>

Example Usage:

(.venv) \$ python3 SHUIM.py sampleTDB.txt output.txt sampleN.txt 35

Note: minSup will be considered in percentage of database transactions

Sample run of importing the code:

```
from PAMI.highUtilitySpatialPattern.basic import SHUIM as alg
obj=alg.SHUIM("input.txt","Neighbours.txt",35)
obj.mine()
frequentPatterns = obj.getPatterns()
print("Total number of Spatial high utility Patterns:", len(frequentPatterns))
obj.save("output")
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by Pradeep Pallikila under the supervision of Professor Rage Uday Kiran.

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns() \rightarrow Dict[str, str]

Function to send the set of patterns after completion of the mining process

Returns

returning patterns

Return type dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final patterns in a dataframe

Returns

returning patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

nou

$mine() \rightarrow None$

main program to start the operation

$printResults() \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of patterns will be loaded in to an output file

Parameters outFile (csv file) - name of the output file

Returns None

startMine() \rightarrow None

main program to start the operation

4.4.2 Top-K

TKSHUIM

class PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Dataset(datasetpath, sep)

Bases: object

A class represent the list of transactions in this dataset

Attributes

transactions: the list of transactions in this dataset

maxItem:

the largest item name

Methods

createTransaction(line):

Create a transaction object from a line from the input file

getMaxItem():

return Maximum Item

getTransactions():

return transactions in database

createTransaction(line)

A method to create Transaction from dataset given

Parameters

line (*string*) – represent a single line of database

:return : Transaction. :rtype: int

getMaxItem()

A method to return name of the largest item

getTransactions()

A method to return transactions from database

maxItem = 0

transactions = []

class PAMI.highUtilitySpatialPattern.topk.TKSHUIM.**TKSHUIM**(*iFile*, *nFile*, *k*, *sep*=\t')

Bases: utilityPatterns

Description

Top K Spatial High Utility ItemSet Mining (TKSHUIM) aims to discover Top-K Spatial High Utility Itemsets (TKSHUIs) in a spatioTemporal database

Reference

P. Pallikila et al., "Discovering Top-k Spatial High Utility Itemsets in Very Large Quantitative Spatiotemporal databases," 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 2021, pp. 4925-4935, doi: 10.1109/BigData52589.2021.9671912.

Parameters

- iFile str : Name of the Input file to mine complete set of High Utility Spatial patterns
- oFile str : Name of the output file to store complete set of High Utility Spatial patterns
- minUtil int : Minimum utility threshold given by User
- maxMemory int : Maximum memory used by this program for running
- **candidateCount** int : Number of candidates to consider when calculating a high utility spatial pattern
- nFile str : Name of the input file to mine complete set of High Utility Spatial patterns
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the input file to mine complete set of frequent patterns

nFile

[file] Name of the Neighbours file that contain neighbours of items

oFile

[file] Name of the output file to store complete set of frequent patterns

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

k

[int] The user given k value

candidateCount: int

Number of candidates

utilityBinArrayLU: list

A map to hold the pmu values of the items in database

utilityBinArraySU: list

A map to hold the subtree utility values of the items is database

oldNamesToNewNames: list

A map to hold the subtree utility values of the items is database

newNamesToOldNames: list

A map to store the old name corresponding to new name

Neighbours

[map] A dictionary to store the neighbours of a item

maxMemory: float

Maximum memory used by this program for running

itemsToKeep: list

keep only the promising items ie items having twu >= minUtil

itemsToExplore: list

keep items that subtreeUtility grater than minUtil

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

calculateNeighbourIntersection(self, prefixLength)

A method to return common Neighbours of items

- **backtrackingEFIM(transactionsOfP, itemsToKeep, itemsToExplore, prefixLength)** A method to mine the TKSHUIs Recursively
- useUtilityBinArraysToCalculateUpperBounds(transactionsPe, j, itemsToKeep, neighbourhoodList)

A method to calculate the sub-tree utility and local utility of all items that can extend itemSet P and e

output(tempPosition, utility)

A method ave a high-utility itemSet to file or memory depending on what the user chose

is_equal(transaction1, transaction2)

A method to Check if two transaction are identical

intersection(lst1, lst2)

A method that return the intersection of 2 list

useUtilityBinArrayToCalculateSubtreeUtilityFirstTime(dataset)

Scan the initial database to calculate the subtree utility of each items using a utility-bin array

sortDatabase(self, transactions)

A Method to sort transaction in the order of PMU

sort_transaction(self, trans1, trans2)

A Method to sort transaction in the order of PMU

useUtilityBinArrayToCalculateLocalUtilityFirstTime(self, dataset)

A method to scan the database using utility bin array to calculate the pmus

Executing the code on terminal:

Format:

(.venv) \$ python3 TKSHUIM.py <inputFile> <outputFile> <Neighbours> <k> <sep>

Example Usage:

(.venv) \$ python3 TKSHUIM.py sampleTDB.txt output.txt sampleN.txt 35

Note: maxMemory will be considered as Maximum memory used by this program for running

Sample run of importing the code:

```
from PAMI.highUtilitySpatialPattern.topk import TKSHUIM as alg
obj=alg.TKSHUIM("input.txt","Neighbours.txt",35)
obj.mine()
Patterns = obj.getPatterns()
obj.save("output")
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by Pradeep Pallikila under the supervision of Professor Rage Uday Kiran.

Neighbours = {}

```
additemset(itemset, utility)
```

adds the itemset to the priority queue

Parameters

- **itemset** (*str*) the itemset to be added
- **utility** (*numpy.array*) utility matrix for the itemset to be added

backtrackingEFIM(*transactionsOfP*, *itemsToKeep*, *itemsToExplore*, *prefixLength*)

A method to mine the TKSHUIs Recursively

Parameters

- transactionsOfP (list) the list of transactions containing the current prefix P
- itemsToKeep (list) the list of secondary items in the p-projected database
- itemsToExplore (list) the list of primary items in the p-projected database
- **prefixLength** (*int*) current prefixLength

calculateNeighbourIntersection(prefixLength)

A method to find common Neighbours

Parameters

prefixLength - the prefix itemSet

:type prefixLength:int

candidateCount = 0

endTime = 0.0

finalPatterns = {}

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns()

Function to send the set of patterns after completion of the mining process

Returns

returning patterns

Return type

dict

getPatternsAsDataFrame()

Storing final patterns in a dataframe

Returns

returning patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

heapList = []

iFile = ' '

intTostr = {}

intersection(lst1, lst2)

A method that return the intersection of 2 list

Parameters

- **lst1** (*list*) items neighbour to item1
- 1st2 (list) items neighbour to item2

:return :intersection of two lists :rtype : list

is_equal(transaction1, transaction2)

A method to Check if two transaction are identical

Parameters

- **transaction1** (Transaction) the first transaction.
- transaction2 (Transaction) the second transaction.

:return : whether both are identical or not :rtype: bool

maxMemory = 0

memoryRSS = 0.0

```
memoryUSS = 0.0
```

```
minUtil = 0
```

mine()

Main function of the program.

nFile = ' '

newNamesToOldNames = {}

oFile = ' '

```
oldNamesToNewNames = {}
```

```
output(tempPosition, utility)
```

A method save all high-utility itemSet to file or memory depending on what the user chose

```
Parameters
```

tempPosition – position of last item

:type tempPosition : int :param utility: total utility of itemSet :type utility: int

printResults()

This function is used to print the results

save(outFile)

Complete set of patterns will be loaded in to an output file

```
Parameters
```

outFile (csv file) – name of the output file

sep = ' t'

sortDatabase(transactions)

A Method to sort transaction in the order of PMU

Parameters

transactions (Transaction) – transaction of items

Returns

sorted transaction

Return type

Transaction

sort_transaction(trans1, trans2)

A Method to sort transaction in the order of PMU

Parameters

trans1 (Transaction) – the first transaction.

:param trans2:the second transaction. :type trans2: Transaction :return: sorted transaction. :rtype: int

startMine()

Main function of the program.

startTime = 0.0

strToint = {}
useUtilityBinArrayToCalculateLocalUtilityFirstTime(dataset)

A method to scan the database using utility bin array to calculate the pmus

Parameters

dataset (*database*) – the transaction database.

useUtilityBinArrayToCalculateSubtreeUtilityFirstTime(dataset)

Scan the initial database to calculate the subtree utility of each item using a utility-bin array

Parameters

dataset (Dataset) – the transaction database

useUtilityBinArraysToCalculateUpperBounds(*transactionsPe*, *j*, *itemsToKeep*, *neighbourhoodList*)

A method to calculate the sub-tree utility and local utility of all items that can extend itemSet P U {e}

Parameters

transactionsPe (*list*) – transactions the projected database for P U {e}

:param j:the position of j in the list of promising items :type j:int :param itemsToKeep :the list of promising items :type itemsToKeep: list :param neighbourhoodList: list of neighbourhood elements :type neighbourhoodList: list

utilityBinArrayLU = {}

utilityBinArraySU = {}

Bases: object

A class to store Transaction of a database

Attributes

items: list

A list of items in transaction

utilities: list

A list of utilites of items in transaction

transactionUtility: int

represent total sum of all utilities in the database

pmus: list

represent the pmu (probable maximum utility) of each element in the transaction

prefixutility:

prefix Utility values of item

offset:

an offset pointer, used by projected transactions

Methods

projectedTransaction(offsetE):

A method to create new Transaction from existing till offsetE

getItems():

return items in transaction

getUtilities():

return utilities in transaction

getPmus():

return pmus in transaction

getLastPosition():

return last position in a transaction

removeUnpromisingItems():

A method to remove items with low Utility than minUtil

insertionSort():

A method to sort all items in the transaction

getItems()

A method to return items in transaction

getLastPosition()

A method to return last position in a transaction

getPmus()

A method to return pmus in transaction

getUtilities()

A method to return utilities in transaction

insertionSort()

A method to sort items in order

offset = 0

prefixUtility = 0

projectTransaction(offsetE)

A method to create new Transaction from existing till offsetE

Parameters

offsetE (int) – an offset over the original transaction for projecting the transaction

removeUnpromisingItems(oldNamesToNewNames)

A method to remove items with low Utility than minUtil

Parameters

oldNamesToNewNames (map) - A map represent old namses to new names

PAMI.highUtilitySpatialPattern.topk.TKSHUIM.main()

4.5 Relative High-Utility Pattern Mining

Relative high utility pattern mining involves the discovery of patterns in datasets where each pattern has a high utility relative to other patterns in the dataset. These patterns represent itemsets, sequences, or other structured data elements that contribute significantly to a predefined utility measure compared to other patterns in the dataset.

Applications: Retail and Market Basket Analysis, Recommendation Systems, Financial Transaction Analysis.

Relative high utility pattern mining involves the discovery of patterns in datasets where each pattern has a high utility relative to other patterns in the dataset. These patterns represent itemsets, sequences, or other structured data elements that contribute significantly to a predefined utility measure compared to other patterns in the dataset.

Applications: Retail and Market Basket Analysis, Recommendation Systems, Financial Transaction Analysis.

4.5.1 Basic

EFIM

class PAMI.highUtilityPattern.basic.EFIM.**EFIM**(*iFile*, *minUtil*, *sep*=\t')

Bases: _utilityPatterns

Description

EFIM is one of the fastest algorithm to mine High Utility ItemSets from transactional databases.

Reference

Zida, S., Fournier-Viger, P., Lin, J.CW. et al. EFIM: a fast and memory efficient algorithm for high-utility itemset mining. Knowl Inf Syst 51, 595–625 (2017). https://doi.org/10.1007/s10115-016-0986-0

Parameters

- iFile str : Name of the Input file to mine complete set of High Utility patterns
- oFile str : Name of the output file to store complete set of High Utility patterns
- minUtil int : The user given minUtil value.
- candidateCount int Number of candidates specified by user
- maxMemory int Maximum memory used by this program for running
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the input file to mine complete set of high utility patterns

oFile

[file] Name of the output file to store complete set of high utility patterns

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

minUtil

[int] The user given minUtil value

highUtilityitemSets: map set of high utility itemSets

candidateCount: int

Number of candidates

utilityBinArrayLU: list

A map to hold the local utility values of the items in database

utilityBinArraySU: list

A map to hold the subtree utility values of the items is database

oldNamesToNewNames: list

A map which contains old names, new names of items as key value pairs

newNamesToOldNames: list

A map which contains new names, old names of items as key value pairs

maxMemory: float

Maximum memory used by this program for running

patternCount: int

Number of HUI's

itemsToKeep: list

keep only the promising items ie items having local utility values greater than or equal to minUtil

itemsToExplore: list

list of items that have subtreeUtility value greater than or equal to minUtil

:Methods :

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

backTrackingEFIM(transactionsOfP, itemsToKeep, itemsToExplore, prefixLength)

A method to mine the HUIs Recursively

useUtilityBinArraysToCalculateUpperBounds(transactionsPe, j, itemsToKeep)

A method to calculate the sub-tree utility and local utility of all items that can extend itemSet P and e

output(tempPosition, utility)

A method to output a high-utility itemSet to file or memory depending on what the user chose

is_equal(transaction1, transaction2)

A method to Check if two transaction are identical

useUtilityBinArrayToCalculateSubtreeUtilityFirstTime(dataset)

A method to calculate the sub tree utility values for single items

sortDatabase(self, transactions)

A Method to sort transaction

sort_transaction(self, trans1, trans2)

A Method to sort transaction

useUtilityBinArrayToCalculateLocalUtilityFirstTime(self, dataset)

A method to calculate local utility values for single itemsets

Executing the code on terminal:

```
Format:
(.venv) $ python3 EFIM.py <inputFile> <outputFile> <minUtil> <sep>
Example Usage:
(.venv) $ python3 EFIM sampleTDB.txt output.txt 35
```

Note: maxMemory will be considered as Maximum memory used by this program for running

Sample run of importing the code:

```
from PAMI.highUtilityPattern.basic import EFIM as alg
obj=alg.EFIM("input.txt",35)
obj.mine()
Patterns = obj.getPatterns()
print("Total number of high utility Patterns:", len(Patterns))
obj.save("output")
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by pradeep pallikila under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function :return: returning RSS memory consumed by the mining process :rtype: float

$\texttt{getMemoryUSS()} \to \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function :return: returning USS memory consumed by the mining process :rtype: float

getPatterns() \rightarrow dict

Function to send the set of patterns after completion of the mining process :return: returning patterns :rtype: dict

getPatternsAsDataFrame() \rightarrow _pd.DataFrame

Storing final patterns in a dataframe :return: returning patterns in a dataframe :rtype: pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process :return: returning total amount of runtime taken by the mining process :rtype: float

mine() \rightarrow None

Start the EFIM algorithm. :return: None

$printResults() \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file :param outFile: name of the output file :type outFile: csv file :return: None

sort_transaction(*trans1: _Transaction*, *trans2: _Transaction*) \rightarrow int

A Method to sort transaction :param trans1: the first transaction :type trans1: Trans :param trans2:the second transaction :type trans2: Trans :return: sorted transaction :rtype: int

$startMine() \rightarrow None$

Start the EFIM algorithm. :return: None

HMiner

class PAMI.highUtilityPattern.basic.HMiner.**HMiner**(*iFile1*, *minUtil*, *sep*='\t')

Bases: _utilityPatterns

Description

High Utility itemSet Mining (HMIER) is an importent algorithm to miner High utility items from the database.

Reference

Parameters

- iFile str : Name of the Input file to mine complete set of High Utility patterns
- oFile str : Name of the output file to store complete set of High Utility patterns

- **minUtil** int : The user given minUtil value.
- **minSup** int or float or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.
- **maxPer** float : The user can specify maxPer in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the input file to mine complete set of frequent patterns

oFile

[file] Name of the output file to store complete set of frequent patterns

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

minUtil

[int] The user given minUtil

mapFMAP: list

EUCS map of the FHM algorithm

candidates: int

candidates genetated

huiCnt: int

huis created

neighbors: map

keep track of nighboues of elements

Methods

mine() Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

- **Explore_SearchTree(prefix, uList, minUtil)** A method to find all high utility itemSets
- **UpdateCLosed**(**x**, **culs**, **st**, **excul**, **newT**, **ex**, **ey_ts**, **length**) A method to update closed values
- saveitemSet(prefix, prefixLen, item, utility)
 A method to save itemSets
- updateElement(z, culs, st, excul, newT, ex, duppos, ey_ts) A method to updates vales for duplicates
- **construcCUL**(**x**, **culs**, **st**, **minUtil**, **length**, **exnighbors**) A method to construct CUL's database

Executing the code on terminal:

Format:

(.venv) \$ python3 HMiner.py <inputFile> <outputFile> <minUtil>

Example Usage:

(.venv) \$ python3 HMiner.py sampleTDB.txt output.txt 35

Note: minSup will be considered in percentage of database transactions

Sample run of importing the code:

from PAMI.highUtilityPattern.basic import HMiner as alg
obj = alg.HMiner("input.txt",35)
obj.mine()
Patterns = obj.getPatterns()
print("Total number of high utility Patterns:", len(Patterns))
obj.save("output")
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)

(continues on next page)

(continued from previous page)

```
memRSS = obj.getMemoryRSS()
```

```
print("Total Memory in RSS", memRSS)
```

run = obj.getRuntime()

```
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by B.Sai Chitra under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function :return: returning RSS memory consumed by the mining process :rtype: float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function :return: returning USS memory consumed by the mining process :rtype: float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process :return: returning frequent patterns :rtype: dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe :return: returning frequent patterns in a dataframe :rtype: pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process :return: returning total amount of runtime taken by the mining process :rtype: float

mine()

Main program to start the operation

printResults()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file :param outFile: name of the output file :type outFile: csv file

startMine()

Main program to start the operation

UPGrowth

```
class PAMI.highUtilityPattern.basic.UPGrowth.UPGrowth(iFile: str, minUtil: int, sep: str = '\t')
Bases: _utilityPatterns
```

Description

UP-Growth is two-phase algorithm to mine High Utility Itemsets from transactional databases.

Reference

Vincent S. Tseng, Cheng-Wei Wu, Bai-En Shie, and Philip S. Yu. 2010. UP-Growth: an efficient algorithm for high utility itemset mining. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '10). Association for Computing Machinery, New York, NY, USA, 253–262. DOI:https://doi.org/10.1145/1835804.1835839

Parameters

- iFile str : Name of the Input file to mine complete set of High Utility patterns
- oFile str : Name of the output file to store complete set of High Utility patterns
- **minUtil** int : The user given minUtil value.
- candidateCount int Number of candidates specified by user
- maxMemory int Maximum memory used by this program for running
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the input file to mine complete set of frequent patterns

oFile

[file] Name of the output file to store complete set of frequent patterns

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

minUtil

[int] The user given minUtil

NumberOfNodes

[int] Total number of nodes generated while building the tree

ParentNumberOfNodes

[int] Total number of nodes required to build the parent tree

MapItemToMinimumUtility

[map] A map to store the minimum utility of item in the database

phuis

[list] A list to store the phuis

MapItemToTwu

[map] A map to store the twu of each item in database

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

createLocalTree(tree, item)

A Method to Construct conditional pattern base

UPGrowth(tree, alpha)

A Method to Mine UP Tree recursively

PrintStats()

A Method to print number of phuis

save(oFile)

Complete set of frequent patterns will be loaded in to an output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

Executing the code on terminal:

Format:

(.venv) \$ python3 UPGrowth <inputFile> <outputFile> <Neighbours> <minUtil> <sep>

Example Usage:

(.venv) \$ python3 UPGrowth sampleTDB.txt output.txt sampleN.txt 35

Note: maxMemory will be considered as Maximum memory used by this program for running

Sample run of importing the code:

```
from PAMI.highUtilityPattern.basic import UPGrowth as alg
```

```
obj=alg.UPGrowth("input.txt",35)
```

```
obj.mine()
```

```
highUtilityPattern = obj.getPatterns()
```

(continues on next page)

(continued from previous page)

```
print("Total number of Spatial Frequent Patterns:", len(highUtilityPattern))
obj.save("output")
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by Pradeep pallikila under the supervision of Professor Rage Uday Kiran.

$PrintStats() \rightarrow None$

A Method to print number of phuis :return: None

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function :return: returning RSS memory consumed by the mining process :rtype: float

$\texttt{getMemoryUSS()} \to \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function :return: returning USS memory consumed by the mining process :rtype: float

getPatterns() \rightarrow dict

Function to send the set of frequent patterns after completion of the mining process :return: returning frequent patterns :rtype: dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final frequent patterns in a dataframe :return: returning frequent patterns in a dataframe :rtype: pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process return: returning total amount of runtime taken by the mining process :rtype: float

mine() \rightarrow None

Mining process will start from here :return: None

$printResults() \rightarrow None$

This function is used to print the results :return: None

```
save(outFile: str) \rightarrow None
```

Complete set of frequent patterns will be loaded in to an output file :param outFile: name of the output file :type outFile: csv file :return: None

```
startMine() \rightarrow None
```

Mining process will start from here :return: None

4.6 Weighted Frequent Pattern Mining

Weighted frequent pattern mining involves the discovery of patterns in datasets where items are assigned different weights based on their significance. These patterns represent combinations of items that occur frequently and have a high cumulative weight relative to other patterns in the dataset. The main focus in weighted frequent pattern mining is to satisfy the downward closure property, which ensures that any subset of a frequent pattern is also frequent.

Applications: Market Basket Analysis, Healthcare Analytics, Network Traffic Analysis.

Weighted frequent pattern mining involves the discovery of patterns in datasets where items are assigned different weights based on their significance. These patterns represent combinations of items that occur frequently and have a high cumulative weight relative to other patterns in the dataset. The main focus in weighted frequent pattern mining is to satisfy the downward closure property, which ensures that any subset of a frequent pattern is also frequent.

Applications: Market Basket Analysis, Healthcare Analytics, Network Traffic Analysis.

4.6.1 Basic

WFIM

class PAMI.weightedFrequentPattern.basic.WFIM.WFIM(*iFile: str, wFile: str, minSup: str, minWeight: int, sep: str = \t'*)

Bases: _weightedFrequentPatterns

About this algorithm

Description

- WFMiner is one of the fundamental algorithm to discover weighted frequent patterns in a transactional database.
- It stores the database in compressed fp-tree decreasing the memory usage and extracts the patterns from tree. It employs employs downward closure property to reduce the search space effectively.

Reference

U. Yun and J. J. Leggett, "Wfim: weighted frequent itemset mining with a weight range and a minimum weight," In: Proceedings of the 2005 SIAM International Conference on Data Mining. SIAM, 2005, pp. 636–640.

https://epubs.siam.org/doi/pdf/10.1137/1.9781611972757.76

param iFile

str : Name of the Input file to mine complete set of weighted Frequent Patterns.

param oFile

str : Name of the output file to store complete set of weighted Frequent Patterns.

param minSup

str or int or float: minimum support thresholds were tuned to find the appropriate ranges in the limited memory

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

:Attributes :

iFile

[file] Input file name or path of the input file

minSup: float or int or str

The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

minWeight: float or int or str

The user can specify minWeight either in count or proportion of database size. If the program detects the data type of minWeight is integer, then it treats minWeight is expressed in count. Otherwise, it will be treated as float. Example: minWeight=10 will be treated as integer, while minWeight=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

oFile

[file] Name of the output file or the path of the output file

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

finalPatterns

[dict] it represents to store the patterns

:Methods :

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets()

Scans the dataset or dataframes and stores in list format

frequentOneItem()

Extracts the one-frequent patterns from transactions

Execution methods

Terminal command

```
Format:
```

Example Usage:

(.venv) \$ python3 basic.py sampleDB.txt weightSample.txt patterns.txt 10.0 3.4

Note: minSup and maxPer will be considered in support count or frequency

Calling from a python program

```
from PAMI.weightFrequentPattern.basic import basic as alg
iFile = 'sampleDB.txt'
minSup = 10  # can also be specified between 0 and 1
obj = alg.basic(iFile, wFile, minSup, minWeight)
obj.mine()
```

(continues on next page)

(continued from previous page)

```
frequentPatterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(frequentPatterns))
obj.savePatterns(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getmemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function.

Returns

returning RSS memory consumed by the mining process

Return type float

$\texttt{getMemoryUSS()} \rightarrow \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$\texttt{getPatterns()} \rightarrow Dict[str, int]$

Function to send the set of frequent patterns after completion of the mining process.

Returns

returning frequent patterns

Return type

dict

$\texttt{getPatternsAsDataFrame()} \rightarrow \texttt{DataFrame}$

Storing final frequent patterns in a dataframe.

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process.

Returns

returning total amount of runtime taken by the mining process

Return type float

mine() \rightarrow None

main program to start the operation

Returns None

 $printResults() \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file.

Parameters outFile (*csv file*) – name of the output file

Returns None

```
\texttt{startMine()} \to \text{None}
```

main program to start the operation

Returns

None

4.7 Weighted Frequent Regular Pattern Mining

Weighted frequent regular pattern mining involves the discovery of regular patterns in a dataset where items are assigned different weights based on their significance. Regular patterns are sequences of itemsets that occur frequently and exhibit a regular or repeating structure. In weighted frequent regular pattern mining, the significance of a pattern is determined not only by its frequency but also by the cumulative weights of its constituent itemsets.

Applications: Retail Analytics, Healthcare Data Analysis, Manufacturing Process Optimization.

Weighted frequent regular pattern mining involves the discovery of regular patterns in a dataset where items are assigned different weights based on their significance. Regular patterns are sequences of itemsets that occur frequently and exhibit a regular or repeating structure. In weighted frequent regular pattern mining, the significance of a pattern is determined not only by its frequency but also by the cumulative weights of its constituent itemsets.

Applications: Retail Analytics, Healthcare Data Analysis, Manufacturing Process Optimization.

4.7.1 Basic

WFRIMiner

class PAMI.weightedFrequentRegularPattern.basic.WFRIMiner.**WFRIMiner**(*iFile*, _*wFile*, *WS*,

regularity, *sep*=t')

Bases: _weightedFrequentRegularPatterns

About this algorithm

Description

WFRIMiner is one of the fundamental algorithm to discover weighted frequent regular patterns in a transactional database. * It stores the database in compressed WFRI-tree decreasing the memory usage and extracts the patterns from tree. It employs downward closure property to reduce the search space effectively.

Reference

K. Klangwisan and K. Amphawan, "Mining weighted-frequent-regular itemsets from transactional database," 2017 9th International Conference on Knowledge and Smart Technology (KST), 2017, pp. 66-71, doi: 10.1109/KST.2017.7886090.

param iFile

str : Name of the Input file to mine complete set of Weighted Frequent Regular Patterns.

param oFile

str : Name of the output file to store complete set of Weighted Frequent Regular Patterns.

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

param wFile

str : This is a weighted file.

Attributes

iFile

[file] Input file name or path of the input file

WS: float or int or str

The user can specify WS either in count or proportion of database size. If the program detects the data type of WS is integer, then it treats WS is expressed in count. Otherwise, it will be treated as float. Example: WS=10 will be treated as integer, while WS=10.0 will be treated as float

regularity: float or int or str

The user can specify regularity either in count or proportion of database size. If the program detects the data type of regularity is integer, then it treats regularity is expressed in count. Otherwise, it will be treated as float. Example: regularity=10 will be treated as integer, while regularity=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

oFile

[file] Name of the output file or the path of the output file

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

finalPatterns

[dict] it represents to store the patterns

Methods

startMine()

Mining process will start from here

getPatterns() Complete

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to an output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets()

Scans the dataset or dataframes and stores in list format

frequentOneItem()

Extracts the one-frequent patterns from transactions

Execution methods

Terminal command

Format:

```
(.venv) $ python3 WFRIMiner.py <inputFile> <outputFile> <weightSupport> <regularity>
```

Example Usage:

(.venv) \$ python3 WFRIMiner.py sampleDB.txt patterns.txt 10 5

Note: WS & regularity will be considered in support count or frequency

Calling from a python program

```
from PAMI.weightedFrequentRegularpattern.basic import WFRIMiner as alg

iFile = 'sampleDB.txt'

minSup = 10  # can also be specified between 0 and 1

obj = alg.WFRIMiner(iFile, WS, regularity)

obj.mine()

weightedFrequentRegularPatterns = obj.getPatterns()

print("Total number of Frequent Patterns:", len(weightedFrequentRegularPatterns))

obj.save(oFile)

Df = obj.getPatternInDataFrame()

memUSS = obj.getMemoryUSS()

print("Total Memory in USS:", memUSS)

memRSS = obj.getMemoryRSS()

print("Total Memory in RSS", memRSS)

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)
```

Credits

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \rightarrow \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$getPatterns() \rightarrow Dict[str, float]$

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

$mine() \rightarrow None$

Frequent pattern mining process will start from here

$\texttt{printResults()} \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (csv file) – name of the output file

Returns None

```
startMine() \rightarrow None
```

Frequent pattern mining process will start from here

4.8 Weighted Frequent Neighbourhood Pattern Mining

Weighted frequent neighborhood pattern mining involves the discovery of frequent patterns in a spatial dataset where patterns are assigned different weights based on their significance and proximity to other patterns. Neighborhood patterns refer to groups of spatially related objects or events that occur frequently and exhibit a regular or repeating structure. In weighted frequent neighborhood pattern mining, the significance of a pattern is determined not only by its frequency but also by the cumulative weights of its neighboring patterns.

Applications: Urban Planning, Environmental Monitoring, Transportation Planning.

Weighted frequent neighborhood pattern mining involves the discovery of frequent patterns in a spatial dataset where patterns are assigned different weights based on their significance and proximity to other patterns. Neighborhood patterns refer to groups of spatially related objects or events that occur frequently and exhibit a regular or repeating structure. In weighted frequent neighborhood pattern mining, the significance of a pattern is determined not only by its frequency but also by the cumulative weights of its neighboring patterns.

Applications: Urban Planning, Environmental Monitoring, Transportation Planning.

4.8.1 Basic

SWFPGrowth

class PAMI.weightedFrequentNeighbourhoodPattern.basic.SWFPGrowth.**SWFPGrowth**(*iFile: str* |

DataFrame, nFile: str | DataFrame, minWS: int | float | str, sep=\t')

Bases: _weightedFrequentSpatialPatterns

About this algorithm

Description

SWFPGrowth is an algorithm to mine the weighted spatial frequent patterns in spatiotemporal databases.

Reference

R. Uday Kiran, P. P. C. Reddy, K. Zettsu, M. Toyoda, M. Kitsuregawa and P. Krishna Reddy, "Discovering Spatial Weighted Frequent Itemsets in Spatiotemporal Databases," 2019 International Conference on Data Mining Workshops (ICDMW), 2019, pp. 987-996, doi: 10.1109/ICDMW.2019.00143.

param iFile

str : Name of the Input file to mine complete set of weighted Frequent Neighbourhood Patterns.

param oFile

str: Name of the output file to store complete set of weighted Frequent Neighbourhood Patterns.

param minSup

int or str or float: minimum support thresholds were tuned to find the appropriate ranges in the limited memory

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

param maxper

floot : where maxPer represents the maximum periodicity threshold value specified by the user.

Attributes

iFile

[file] Input file name or path of the input file

minWS: float or int or str

The user can specify minWS either in count or proportion of database size. If the program detects the data type of minWS is integer, then it treats minWS is expressed in count. Otherwise, it will be treated as float. Example: minWS=10 will be treated as integer, while minWS=10.0 will be treated as float

minWeight: float or int or str

The user can specify minWeight either in count or proportion of database size. If the program detects the data type of minWeight is integer, then it treats minWeight is expressed in count. Otherwise, it will be treated as float. Example: minWeight=10 will be treated as integer, while minWeight=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

oFile

[file] Name of the output file or the path of the output file

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] it represents the total no of transactions

tree

[class] it represents the Tree class

finalPatterns

[dict] it represents to store the patterns

:Methods :

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to an output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets()

Scans the dataset or dataframes and stores in list format

frequentOneItem()

Extracts the one-frequent patterns from transactions

Execution methods

Terminal command

```
Format:
(.venv) $ python3 SWFPGrowth.py <inputFile> <weightFile> <outputFile> <minSup>
→<minWeight>
```

Example usage :

(.venv) \$ python3 SWFPGrowth.py sampleDB.txt weightFile.txt patterns.txt 10 2

Note: minSup will be considered in support count or frequency

Calling from a python program

```
from PAMI.weightFrequentNeighbourhoodPattern.basic import SWFPGrowth as alg
obj = alg.SWFPGrowth(iFile, wFile, nFile, minSup, minWeight, seperator)
iFile = 'sampleDB.txt'
minSup = 10  # can also be specified between 0 and 1
```

(continues on next page)

(continued from previous page)

```
obj.mine()
frequentPatterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(frequentPatterns))
obj.save(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getmemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

$getPatterns() \rightarrow Dict[str, float]$

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

$\texttt{getPatternsAsDataFrame()} \rightarrow DataFrame$

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

$\texttt{mine()} \rightarrow \text{None}$

Frequent pattern mining process will start from here

$printResults() \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (*csv file*) – name of the output file

Returns

None

$\texttt{startMine()} \rightarrow \text{None}$

Frequent pattern mining process will start from here

CHAPTER

FUZZY PATTERN MINING

A fuzzy database represents the data generated from a non-binary transactional or temporal database using fuzzy logic.

Types

- Fuzzy transactional databases
- Fuzzy temporal databases

Fuzzy transactional databases

A fuzzy transactional database represents a set of transactions, where each transaction consists of a transactional identifier (tid), items, and their fuzzy occurrences values. Please note that the fuzzy occurrence values of an item lie between 0 and 1. If the fuzzy value of an item is close zero, it implies less chance of occurrence of an item in a database. If the fuzzy value of an item is close one, it implies high chance of occurrence of an item in a database. A sample fuzzy transactional database generated from the set of items, $I=\{Bread, Jam, Butter, Pen, Books, Bat\}$, is shown in below table:

TID	Transactions (items and their fuzzy values)
1	(Bread.High, 0.6), (Bread.Low, 0.4), (Jam.High, 0.2), (Jam.Low, 0.8), (Butter.High, 0.8), (Butter.Low, 0.2)
2	(Bat.High, 0.5), (Bat.Low, 0.5), (Ball.High, 0.6), (Ball.Low, 0.4)
3	(Pen.High, 0.2), (Pen.Low, 0.8), (Book.High, 0.3), (Book.Low, 0.7)

Format of a fuzzy transactional database

The fuzzy transactional database must exist in the following format:

The 'total fuzzy value' represents the sum of fuzzy values of all items in a transaction.

Rules to create a fuzzy database

- The default separator, i.e., , used in PAMI is tab space (or t). However, the users can override the default separator with their choice. Since spatial objects, such as Point, Line, and Polygon, are represented using space and comma, usage of tab space facilitates us to effectively distinguish the spatial objects.
- Items, total utility, and individual utilities of the items within a transaction have to be seperated by the symbol ':'

An example

Bread.High Bread.Low Jam.High Jam.Low Butter.High Butter.Low:3:0.6 0.4 0.2 0.8 0.8 0.2

Bat.High Bat.Low Ball.High Ball.Low:2:0.5 0.5 0.6 0.4

Pen Book:2:0.2 0.8 0.3 0.7

Fuzzy temporal databases

A fuzzy temporal database consists of timestamp, tid, items, and their corresponding fuzzy values. A sample fuzzy temporal database generated from the set of items, I={Bread, Jam, Butter, Pen, Books, Bat}, is shown in below table:

Times- tamp	tid	Transactions (items and their fuzzy values)
1	1	(Bread.High, 0.6), (Bread.Low, 0.4), (Jam.High, 0.2), (Jam.Low, 0.8), (Butter.High, 0.8), (Butter.Low, 0.2)
2	2	(Bat.High, 0.5), (Bat.Low, 0.5), (Ball.High, 0.6), (Ball.Low, 0.4)
5	3	(Pen.High, 0.2), (Pen.Low, 0.8), (Book.High, 0.3), (Book.Low, 0.7)

Format of fuzzy temporal database

The fuzzy temporal database must exist in the following format:

The 'total fuzzy value' represents the total fuzzy value of all items in a transaction.

Rules to create a fuzzy temporal database

- The default separator, i.e., , used in PAMI is tab space (or t). However, the users can override the default separator with their choice. Since spatial objects, such as Point, Line, and Polygon, are represented using space and comma, usage of tab space facilitates us to effectively distinguish the spatial objects.
- Timestamp, items, total utility, and individual utilities of the items within a transaction have to be seperated by the symbol ':'

An example

1:Bread Jam Butter: 3:0.6 0.4 0.2 0.8 0.8 0.2

2:Bat Ball:110:100 10

5:Pen Book:7:2 5

5.1 Fuzzy Frequent Pattern Mining

Fuzzy frequent patterns (FFPs) are patterns that capture the inherent uncertainty or fuzziness in data by allowing for partial matching of items or events. Unlike traditional frequent patterns, which require exact matches between items or events, fuzzy frequent patterns accommodate variations in the degree of membership or similarity between items, making them suitable for data with uncertain or imprecise information

Applications: Medical Data Mining, Financial Analysis, Manufacturing and Quality Control.

Fuzzy frequent patterns (FFPs) are patterns that capture the inherent uncertainty or fuzziness in data by allowing for partial matching of items or events. Unlike traditional frequent patterns, which require exact matches between items

or events, fuzzy frequent patterns accommodate variations in the degree of membership or similarity between items, making them suitable for data with uncertain or imprecise information

Applications: Medical Data Mining, Financial Analysis, Manufacturing and Quality Control.

5.1.1 Basic

FFIMiner

class PAMI.fuzzyFrequentPattern.basic.FFIMiner.**FFIMiner**(*iFile: str, minSup: float, sep: str* = \t')

Bases: _fuzzyFrequentPattenrs

Description

Fuzzy Frequent Pattern-Miner is desired to find all frequent fuzzy patterns which is on-trivial and challenging problem to its huge search space.we are using efficient pruning techniques to reduce the search space.

Reference

Lin, Chun-Wei & Li, Ting & Fournier Viger, Philippe & Hong, Tzung-Pei. (2015). A fast Algorithm for mining fuzzy frequent itemsets. Journal of Intelligent & Fuzzy Systems. 29. 2373-2379. 10.3233/IFS-151936. https://www.researchgate.net/publication/286510908_ A_fast_Algorithm_for_mining_fuzzy_frequent_itemSets

Parameters

- iFile str : Name of the Input file to mine complete set of frequent patterns
- oFile str : Name of the output file to store complete set of frequent patterns
- **minSup** int or float or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.
- **maxPer** float : The user can specify maxPer in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count.
- fuzFile str : The user can specify fuzFile.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[string] Name of the input file to mine complete set of fuzzy frequent patterns

fmFile

[string] Name of the fuzzy membership file to mine complete set of fuzzy frequent patterns

oFile

[string] Name of the oFile file to store complete set of fuzzy frequent patterns

minSup

[float] The user given minimum support

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

itemsCnt: int

To record the number of fuzzy spatial itemSets generated

mapItemSum: map

To keep track of sum of Fuzzy Values of items

joinsCnt: int

To keep track of the number of ffi-list that was constructed

BufferSize: int represent the size of Buffer

itemSetBuffer list

to keep track of items in buffer

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

convert(value)

To convert the given user specified value

compareItems(01, 02)

A Function that sort all ffi-list in ascending order of Support

FSFIMining(prefix, prefixLen, FSFIM, minSup)

Method generate ffi from prefix

construct(px, py)

A function to construct Fuzzy itemSet from 2 fuzzy itemSets

findElementWithTID(uList, tid)

To find element with same tid as given

WriteOut(prefix, prefixLen, item, sumIUtil)

To Store the patten

Executing the code on terminal :

Format:

(.venv) \$ python3 FFIMiner.py <inputFile> <outputFile> <minSup> <separator>

Example Usage:

(.venv) \$ python3 FFIMiner.py sampleTDB.txt output.txt 6

Note: minSup will be considered in percentage of database transactions

Sample run of importing the code:

```
from PAMI.fuzzyFrequentPattern import FFIMiner as alg
obj = alg.FFIMiner("input.txt", 2)
obj.mine()
fuzzyFrequentPattern = obj.getPatterns()
print("Total number of Fuzzy Frequent Patterns:", len(fuzzyFrequentPattern))
obj.save("outputFile")
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by B.Sai Chitra under the supervision of Professor Rage Uday Kiran.

```
getMemoryRSS() \rightarrow float
```

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

 $getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$\texttt{getPatterns()} \rightarrow \texttt{dict}$

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

$getPatternsAsDataFrame() \rightarrow DataFrame$

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None

fuzzy-Frequent pattern mining process will start from here

$printResults() \rightarrow None$

This function is used to print the results

$save(outFile) \rightarrow dict$

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (csv file) - name of the output file

Returns

dictionary of frequent patterns

Return type dict

$\texttt{startMine()} \rightarrow \text{None}$

fuzzy-Frequent pattern mining process will start from here

5.2 Fuzzy Correlated Pattern Mining

Fuzzy correlated pattern mining involves the exploration of associations between fuzzy itemsets that exhibit linear relationships, as assessed through fuzzy correlation analysis. Instead of solely relying on co-occurrence frequencies, this approach considers the strength and type of correlation between fuzzy itemsets to uncover meaningful patterns.

Applications: Market Basket Analysis, Healthcare Analytics, Financial Forecasting.

Fuzzy correlated pattern mining involves the exploration of associations between fuzzy itemsets that exhibit linear relationships, as assessed through fuzzy correlation analysis. Instead of solely relying on co-occurrence frequencies, this approach considers the strength and type of correlation between fuzzy itemsets to uncover meaningful patterns.

Applications: Market Basket Analysis, Healthcare Analytics, Financial Forecasting.

5.2.1 Basic

FCPGrowth

class PAMI.fuzzyCorrelatedPattern.basic.FCPGrowth.Element(tid: int, IUtil: float, RUtil: float)

Bases: object

A class represents an Element of a fuzzy list

Attributes

tid

[int] keep tact of transaction id

IUtils: float the utility of a fuzzy item in the transaction

RUtil

[float] the neighbourhood resting value of a fuzzy item in the transaction

Bases: _corelatedFuzzyFrequentPatterns

Description

FCPGrowth is the algorithm to discover Correlated Fuzzy-frequent patterns in a transactional database. it is based on traditional fuzzy frequent pattern mining.

Reference

Lin, N.P., & Chueh, H. (2007). Fuzzy correlation rules mining. https://citeseerx.ist.psu.edu/ viewdoc/download?doi=10.1.1.416.6053&rep=rep1&type=pdf

Parameters

- iFile str : Name of the Input file to mine complete set of frequent patterns
- oFile str : Name of the output file to store complete set of frequent patterns
- **minSup** int or float or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.
- **maxPer** float : The user can specify maxPer in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count.

- **minAllConf** float : The user can specify minAllConf values within the range (0, 1).
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the input file to mine complete set of fuzzy spatial frequent patterns

oFile

[file] Name of the oFile file to store complete set of fuzzy spatial frequent patterns

minSup

[int] The user given support

minAllConf: float

user Specified minAllConf(should be in range 0 and 1)

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTimeTime:float

To record the startTime time of the mining process

endTime:float

To record the completion time of the mining process

itemsCnt: int

To record the number of fuzzy spatial itemSets generated

mapItemsLowSum: map

To keep track of low region values of items

mapItemsMidSum: map

To keep track of middle region values of items

mapItemsHighSum: map

To keep track of high region values of items

mapItemSum: map

To keep track of sum of Fuzzy Values of items

mapItemRegions: map

To Keep track of fuzzy regions of item

jointCnt: int

To keep track of the number of FFI-list that was constructed

BufferSize: int

represent the size of Buffer

itemBuffer list

to keep track of items in buffer

Methods

startTimeMine()

Mining process will startTime from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file
getPatternsAsDataFrame() Complete set of frequent patterns will be loaded in to a dataframe getMemoryUSS() Total amount of USS memory consumed by the mining process will be retrieved from this function getMemoryRSS() Total amount of RSS memory consumed by the mining process will be retrieved from this function getRuntime() Total amount of runtime taken by the mining process will be retrieved from this function getRatio(self, prefix, prefixLen, item) Method to calculate the ration of itemSet convert(value): To convert the given user specified value FSFIMining(prefix, prefixLen, fsFim, minSup) Method generate FFI from prefix

construct(px, py)

A function to construct Fuzzy itemSet from 2 fuzzy itemSets

findElementWithTID(uList, tid)

To find element with same tid as given

WriteOut(prefix, prefixLen, item, sumIUtil, ratio)

To Store the patten

Executing the code on terminal :

Format:

(.venv) \$ python3 FCPGrowth.py <inputFile> <outputFile> <minSup> <minAllConf> <sep>

Example Usage:

(.venv) \$ python3 FCPGrowth.py sampleTDB.txt output.txt 2 0.2

Note: minSup will be considered in percentage of database transactions

Sample run of importing the code:

```
from PAMI.fuzzyCorrelatedPattern.basic import FCPGrowth as alg
```

obj = alg.FCPGrowth("input.txt",2,0.4)

obj.mine()

correlatedFuzzyFrequentPatterns = obj.getPatterns()

(continues on next page)

(continued from previous page)

```
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by B.Sai Chitra under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \to \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns() \rightarrow Dict[str, List[float]]

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

$\texttt{getPatternsAsDataFrame()} \rightarrow \texttt{DataFrame}$

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type pd.DataFrame

 $\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

mine() \rightarrow None

Frequent pattern mining process will startTime from here

 $printResults() \rightarrow None$

This function is used to print the result

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (csv file) – name of the output file

 $startMine() \rightarrow None$

Frequent pattern mining process will startTime from here

PAMI.fuzzyCorrelatedPattern.basic.FCPGrowth.main()

5.3 Fuzzy Geo-referenced Frequent Pattern Mining

Fuzzy geo-referenced frequent pattern mining refers to the process of discovering patterns in spatial data that occur frequently and exhibit fuzzy relationships or uncertainties. These patterns are identified based on their geographical references and may involve fuzzy spatial attributes or relationships between spatial objects.

Applications: Retail and Marketing, Healthcare and Epidemiology, Environmental Monitoring.

Fuzzy geo-referenced frequent pattern mining refers to the process of discovering patterns in spatial data that occur frequently and exhibit fuzzy relationships or uncertainties. These patterns are identified based on their geographical references and may involve fuzzy spatial attributes or relationships between spatial objects.

Applications: Retail and Marketing, Healthcare and Epidemiology, Environmental Monitoring.

5.3.1 Basic

FFSPMiner

class PAMI.fuzzyGeoreferencedFrequentPattern.basic.FFSPMiner.**FFSPMiner**(*iFile: str*, *nFile: str*,

minSup: float, sep: str = t')

Bases: _fuzzySpatialFrequentPatterns

About this algorithm

Description

Fuzzy Frequent Spatial Pattern-Miner is desired to find all Spatially frequent fuzzy patterns which is on-trivial and challenging problem to its huge search space.we are using efficient pruning techniques to reduce the search space.

Reference

Reference: P. Veena, B. S. Chithra, R. U. Kiran, S. Agarwal and K. Zettsu, "Discovering Fuzzy Frequent Spatial Patterns in Large Quantitative Spatiotemporal databases," 2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2021, pp. 1-8, doi: 10.1109/FUZZ45933.2021.9494594.

param iFile

str : Name of the Input file to mine complete set of frequent patterns

param oFile

str : Name of the output file to store complete set of frequent patterns

param minSup

int or float or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.

param maxPer

float : The user can specify maxPer in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count.

param nFile

str : Name of the input file to mine complete set of frequent patterns

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the input file to mine complete set of fuzzy spatial frequent patterns

oFile

[file] Name of the oFile file to store complete set of fuzzy spatial frequent patterns

minSup

[float] The user given minimum support

neighbors

[map] keep track of neighbours of elements

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

itemsCnt

[int] To record the number of fuzzy spatial itemSets generated

mapItemSum

[map] To keep track of sum of Fuzzy Values of items

mapItemRegions

[map] To Keep track of fuzzy regions of item

joinsCnt

[int] To keep track of the number of FFI-list that was constructed

BufferSize

[int] represent the size of Buffer

itemSetBuffer

[list] to keep track of items in buffer

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

convert(value)

To convert the given user specified value

FSFIMining(prefix, prefixLen, fsFim, minSup)

Method generate FFI from prefix

construct(px, py)

A function to construct Fuzzy itemSet from 2 fuzzy itemSets

Intersection(neighbourX,neighbourY)

Return common neighbours of 2 itemSet Neighbours

findElementWithTID(uList, tid)

To find element with same tid as given

WriteOut(prefix, prefixLen, item, sumIUtil,period)

To Store the patten

Execution methods

Terminal command

```
Format:
```

```
(.venv) $ python3 FFSPMiner.py <inputFile> <outputFile> <neighbours> <minSup> <sep>
```

Example Usage:

(.venv) \$ python3 FFSPMiner.py sampleTDB.txt output.txt sampleN.txt 3

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

Credits

The complete program was written by B.Sai Chitra under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \to \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$\texttt{getPatterns()} \rightarrow Dict[str, str]$

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

$getPatternsAsDataFrame() \rightarrow DataFrame$

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None

Frequent pattern mining process will start from here

Returns None

$printResults() \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (csv file) – name of the output file

Returns None

$startMine() \rightarrow None$

Frequent pattern mining process will start from here

Returns None

5.4 Fuzzy Periodic Frequent Pattern Mining

Fuzzy periodic frequent patterns refer to recurring patterns in temporal data where the occurrences exhibit fuzzy relationships or uncertainties. These patterns are characterized by their periodic nature and may involve imprecise or fuzzy temporal attributes or relationships between events.

Applications: Financial Time Series Analysis, Manufacturing and Production Processes, Network Traffic Analysis.

Fuzzy periodic frequent patterns refer to recurring patterns in temporal data where the occurrences exhibit fuzzy relationships or uncertainties. These patterns are characterized by their periodic nature and may involve imprecise or fuzzy temporal attributes or relationships between events.

Applications: Financial Time Series Analysis, Manufacturing and Production Processes, Network Traffic Analysis.

5.4.1 Basic

FPFPMiner

Bases: _fuzzyPeriodicFrequentPatterns

Description

Fuzzy Periodic Frequent Pattern Miner is desired to find all fuzzy periodic frequent patterns which is on-trivial and challenging problem to its huge search space.we are using efficient pruning techniques to reduce the search space.

Reference

R. U. Kiran et al., "Discovering Fuzzy Periodic-Frequent Patterns in Quantitative Temporal Databases," 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Glasgow, UK, 2020, pp. 1-8, doi: 10.1109/FUZZ48607.2020.9177579.

Parameters

- iFile str : Name of the Input file to mine complete set of frequent patterns
- oFile str : Name of the output file to store complete set of frequent patterns
- **minSup** int or float or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.
- **maxPer** float : The user can specify maxPer in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count.
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the input file to mine complete set of fuzzy spatial frequent patterns

oFile

[file] Name of the oFile file to store complete set of fuzzy spatial frequent patterns

minSup

[float] The user given support

period: int

periodicity of an element

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

itemsCnt: int

To record the number of fuzzy spatial itemSets generated

mapItemsLowSum: map To keep track of low region values of items

mapItemsMidSum: map To keep track of middle region values of items

mapItemsHighSum: map

To keep track of high region values of items

mapItemSum: map To keep track of sum of Fuzzy Values of items

mapItemRegions: map To Keep track of fuzzy regions of item

jointCnt: int

To keep track of the number of FFI-list that was constructed

BufferSize: int

represent the size of Buffer

itemBuffer list

to keep track of items in buffer

maxTID: int

represent the maximum tid of the database

lastTIDs: map

represent the last tid of fuzzy items

itemsToRegion: map

represent items with respective regions

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

convert(value):

To convert the given user specified value

FSFIMining(prefix, prefixLen, fsFim, minSup)

Method generate FFI from prefix

construct(px, py)

A function to construct Fuzzy itemSet from 2 fuzzy itemSets

findElementWithTID(UList, tid)

To find element with same tid as given

WriteOut(prefix, prefixLen, item, sumIUtil, period)

To Store the patten

Executing the code on terminal :

Format:

```
(.venv) $ python3 FPFPMiner.py <inputFile> <outputFile> <minSup> <maxPer> <sep>
```

Example Usage:

(.venv) \$ python3 FPFPMiner.py sampleTDB.txt output.txt 2 3

Note: minSup will be considered in percentage of database transactions

Sample run of importing the code:

from PAMI.fuzzyPeriodicFrequentPattern.basic import FPFPMiner as alg obj =alg.FPFPMiner("input.txt",2,3) obj.mine() periodicFrequentPatterns = obj.getPatterns() print("Total number of Fuzzy Periodic Frequent Patterns:", len(periodicFrequentPatterns)) obj.save("output.txt") memUSS = obj.getMemoryUSS() print("Total Memory in USS:", memUSS) memRSS = obj.getMemoryRSS() print("Total Memory in RSS", memRSS)

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by Sai Chitra.B under the supervision of Professor Rage Uday Kiran.

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$\texttt{getMemoryUSS()} \to \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

$\texttt{getPatterns()} \rightarrow Dict[str, str]$

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

$getPatternsAsDataFrame() \rightarrow DataFrame$

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

mine() \rightarrow None

Fuzzy periodic Frequent pattern mining process will start from here

$printResults() \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (csv file) – name of the output file

Returns None

 $\texttt{startMine()} \to \text{None}$

Fuzzy periodic Frequent pattern mining process will start from here

5.5 Fuzzy Geo-referenced Periodic Frequent Pattern Mining

Fuzzy geo-referenced periodic frequent pattern mining involves the discovery of recurring patterns in spatial-temporal data where the occurrences exhibit fuzzy relationships or uncertainties and are associated with geographical locations. These patterns capture the repetitive nature of events or phenomena over time and space, while considering imprecise or fuzzy attributes and relationships between spatial-temporal entities.

Applications: Traffic Flow Analysis, Environmental Monitoring, Epidemiological Studies.

Fuzzy geo-referenced periodic frequent pattern mining involves the discovery of recurring patterns in spatial-temporal data. where the occurrences exhibit fuzzy relationships or uncertainties and are associated with geographical locations. These patterns capture the repetitive nature of events or phenomena over time and space, while considering imprecise or fuzzy attributes and relationships between spatial-temporal entities.

Applications: Traffic Flow Analysis, Environmental Monitoring, Epidemiological Studies.

5.5.1 Basic

FGPFPMiner

class PAMI.fuzzyGeoreferencedPeriodicFrequentPattern.basic.FGPFPMiner.FGPFPMiner(*iFile*,

nFile, minSup, maxPer, sep)

Bases: _fuzzySpatialFrequentPatterns

About this algorithm

Description

Fuzzy Frequent Spatial Pattern-Miner is desired to find all Spatially frequent fuzzy patterns which is on-trivial and challenging problem to its huge search space.we are using efficient pruning techniques to reduce the search space.

Reference

param iFile

str : Name of the Input file to mine complete set of frequent patterns

param oFile

str : Name of the output file to store complete set of frequent patterns

param minSup

int or float or str : The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float.

param maxPer

float : The user can specify maxPer in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count.

param nFile

str : Name of the input file to mine complete set of frequent patterns

param FuzFile

str: The user can specify fuzFile.

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the input file to mine complete set of fuzzy spatial frequent patterns

oFile

[file] Name of the oFile file to store complete set of fuzzy spatial frequent patterns

minSup

[float] The user given minimum support

neighbors

[map] keep track of neighbours of elements

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

itemsCnt

[int] To record the number of fuzzy spatial itemSets generated

mapItemSum

[map] To keep track of sum of Fuzzy Values of items

joinsCnt

[int] To keep track of the number of FFI-list that was constructed

BufferSize

[int] represent the size of Buffer

itemSetBuffer list

to keep track of items in buffer

Methods

mine()

Mining process will start from here

```
getPatterns()
```

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

convert(value)

To convert the given user specified value

FSFIMining(prefix, prefixLen, fsFim, minSup)

Method generate FFI from prefix

construct(px, py)

A function to construct Fuzzy itemSet from 2 fuzzy itemSets

Intersection(neighbourX,neighbourY)

Return common neighbours of 2 itemSet Neighbours

findElementWithTID(uList, tid)

To find element with same tid as given

WriteOut(prefix, prefixLen, item, sumIUtil,period) To Store the patten

Example Usage:

(.venv) \$ python3 FGPFPMiner.py sampleTDB.txt output.txt sampleN.txt 3 4

Note: minSup will be considered in percentage of database transactions

Calling from a python program

```
from PAMI.fuzzyGeoreferencedPeriodicFrequentPattern import FGPFPMiner as alg
obj = alg.FFSPMiner("input.txt", "neighbours.txt", 3, 4)
obj.mine()
```

(continues on next page)

(continued from previous page)

```
print("Total number of fuzzy frequent spatial patterns:", len(obj.getPatterns()))
obj.save("outputFile")
print("Total Memory in USS:", obj.getMemoryUSS())
print("Total Memory in RSS", obj.getMemoryRSS())
print("Total ExecutionTime in seconds:", obj.getRuntime())
```

Credits

The complete program was written by B.Sai Chitra under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

mine()

Frequent pattern mining process will start from here

printResults()

This function is used to print the result

save(outFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (*csv file*) – name of the output file

startMine()

Frequent pattern mining process will start from here

CHAPTER

UNCERTAIN DATABASE

An uncertain database is a non-binary database, where an occurrence of an item in a transaction is associated with a probabilistic value that lies between zero and one. The value zero represents the complete non-occurrence of an item, while the value represents the perfect occurrence of an item in a transaction.

Currently, the algorithms in PAMI support the discovery of knowledge hidden in two types of uncertain databases, namely uncertain transactional database and uncertain temporal database. We now describe each of these databases.

Types

- Uncertain transactional database
- Uncertain temporal database

Uncertain transactional database

An uncertain transactional database consists of a transactional identifier (tid), items, and their occurrence probability value. A sample uncertain transactional database generated from the set of items, I={Bread, Jam, Butter, Pen, Books, Bat}, is shown in below table:

TID	Transactions (items and their prices)
1	(Bread,0.9), (Jam,0.7), (Butter, 0.1)
2	(Bat, 1), (Ball, 0.5)
3	(Pen, 0.2), (Book, 0.5)

Note: The above uncertain database represents an uncertain transactional database. If every transaction in an uncertain database is associated with a timestamp, then we call that database an uncertain temporal database.

Format to create uncertain transactional databases in PAMI

An utility transactional database must exist in the following format:

The 'total probability' represents the sum of probabilities of all items in a transaction.

Rules to create a uncertain transactional databases

• The default separator, i.e., , used in PAMI is tab space (or t). However, the users can override the default separator with their choice. Since spatial objects, such as Point, Line, and Polygon, are represented using space and comma, usage of tab space facilitates us to effectively distinguish the spatial objects.

- Items, total probability, and individual probabilities of the items within a transaction have to be seperated by the symbol ':'
- The probability values of an item must be within the range [0,1].

An example of an uncertain transactional database

Bread Jam Butter: 1.7:0.9 0.7 0.1 Bat Ball: 1.5:1 0.5 Pen Book: 0.7:0.2

Uncertain temporal database

Introduction

An uncertain temporal database consists of a transactional identifier (tid), a timestamp, items, and their occurrence probability value. A sample uncertain temporal database generated from the set of items, $I=\{Bread, Jam, Butter, Pen, Books, Bat\}$, is shown in below table:

TID	TS	Transactions (items and their prices)
1	1	(Bread,0.9), (Jam,0.7), (Butter, 0.1)
2	4	(Bat, 1), (Ball, 0.5)
3	5	(Pen, 0.2), (Book, 0.5)

Format to create an uncertain temporal databases in PAMI

An utility temporal database must exist in the following format:

The 'total probability' represents the sum of probabilities of all items in a transaction.

Rules to create an uncertain temporal databases

- First element in every transaction must be a timestamp.
- The default separator, i.e., , used in PAMI is tab space (or t). However, the users can override the default separator with their choice. Since spatial objects, such as Point, Line, and Polygon, are represented using space and comma, usage of tab space facilitates us to effectively distinguish the spatial objects.
- Items, total probability, and individual probabilities of the items within a transaction have to be seperated by the symbol ':'
- The probability values of an item must be within the range [0,1].

An example of an uncertain temporal database

1 Bread Jam Butter: 1.7:0.9 0.7 0.1

2 Bat Ball:1.5:1 0.5

3 Pen Book:0.7:0.2 0.5

6.1 Uncertain Frequent Pattern mining

Uncertain frequent pattern mining is a data mining task that involves the discovery of frequent patterns from datasets containing uncertain or probabilistic data. Unlike traditional frequent pattern mining, where the data is precise and deterministic, uncertain frequent pattern mining deals with data in which the values or attributes have associated probabilities or uncertainties.

Applications: Healthcare, Finance, Environmental Science.

Uncertain frequent pattern mining is a data mining task that involves the discovery of frequent patterns from datasets containing uncertain or probabilistic data. Unlike traditional frequent pattern mining, where the data is precise and deterministic, uncertain frequent pattern mining deals with data in which the values or attributes have associated probabilities or uncertainties.

Applications: Healthcare, Finance, Environmental Science.

6.1.1 Basic

CUFPTree

class PAMI.uncertainFrequentPattern.basic.CUFPTree.**CUFPTree**(*iFile*, *minSup*, *sep*='\t')

Bases: _frequentPatterns

About this algorithm

Description

It is one of the fundamental algorithm to discover frequent patterns in a uncertain transactional database using CUFP-Tree.

Reference

Chun-Wei Lin Tzung-PeiHong, 'new mining approach for uncertain databases using CUFP trees', Expert Systems with Applications, Volume 39, Issue 4, March 2012, Pages 4084-4093, https://doi.org/10.1016/j.eswa.2011.09.087

param iFile

str : Name of the Input file to mine complete set of Uncertain Frequent Patterns

param oFile

str : Name of the output file to store complete set of Uncertain frequent patterns

param minSup

int or float or str : minimum support thresholds were tuned to find the appropriate ranges in the limited memory

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup: float or int or str

The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

frequentOneItem()

Extracts the one-length frequent patterns from database

updateTransactions()

Update the transactions by removing non-frequent items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

to convert the user specified value

startMine()

Mining process will start from this function

Execution methods

Terminal command

Format:
(.venv) \$ python3 CUFPTree.py <inputFile> <outputFile> <minSup>

Example Usage:

(.venv) \$ python3 CUFPTree.py sampleTDB.txt patterns.txt 3

Note: minSup will be considered in support count or frequency

Calling from a python program

```
from PAMI.uncertainFrequentPattern.basic import CUFPTree as alg
iFile = 'sampleDB.txt'
minSup = 10 # can also be specified between 0 and 1
obj = alg.CUFPTree(iFile, minSup)
obj.mine()
frequentPatterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(frequentPatterns))
obj.save(oFile)
```

(continues on next page)

(continued from previous page)

```
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$getMemoryRSS() \rightarrow float$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$\texttt{getPatterns()} \rightarrow \texttt{dict}$

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

$\texttt{getPatternsAsDataFrame()} \rightarrow \texttt{DataFrame}$

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

mine() \rightarrow None

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns.

Returns

None

 $\texttt{printResults()} \rightarrow \text{None}$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (csv file) – name of the output file

Returns None

$startMine() \rightarrow None$

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns.

Returns

None

PUFGrowth

class PAMI.uncertainFrequentPattern.basic.PUFGrowth.**PUFGrowth**(*iFile*, *minSup*, *sep*=\t') Bases: _frequentPatterns

About this algorithm

Description

It is one of the fundamental algorithm to discover frequent patterns in a uncertain transactional database using PUF-Tree.

Reference

Carson Kai-Sang Leung, Syed Khairuzzaman Tanbeer, "PUF-Tree: A Compact Tree Structure for Frequent Pattern Mining of Uncertain Data", Pacific-Asia Conference on Knowledge Discovery and Data Mining(PAKDD 2013), https://link.springer.com/chapter/10.1007/ 978-3-642-37453-1_2

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup

[float or int or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

frequentOneItem()

Extracts the one-length frequent patterns from database

updateTransactions()

Update the transactions by removing non-frequent items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

to convert the user specified value

startMine()

Mining process will start from this function

Execution methods

Terminal command

Format:

(.venv) \$ python3 PUFGrowth.py <inputFile> <outputFile> <minSup>

Example Usage:

```
(.venv) $ python3 PUFGrowth.py sampleDB.txt patterns.txt 10.0
```

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

```
from PAMI.uncertainFrequentPattern.basic import puf as alg
iFile = 'sampleDB.txt'
minSup = 10  # can also be specified between 0 and 1
obj = alg.PUFGrowth(iFile, minSup)
obj.startmine()
frequentPatterns = obj.getPatterns()
```

(continues on next page)

(continued from previous page)

```
print("Total number of Frequent Patterns:", len(frequentPatterns))
obj.save(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getmemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$getMemoryUSS() \rightarrow float$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$\texttt{getPatterns()} \rightarrow \texttt{dict}$

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns

$\texttt{printResults()} \rightarrow \text{None}$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (csv file) – name of the output file

$\texttt{startMine()} \to \text{None}$

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns

TUFP

class PAMI.uncertainFrequentPattern.basic.TUFP.**TUFP**(*iFile*, *minSup*, *sep*=\t') Bases: _frequentPatterns

About this algorithm

Description

It is one of the fundamental algorithm to discover top-k frequent patterns in a uncertain transactional database using CUP-Lists.

Reference

Tuong Le, Bay Vo, Van-Nam Huynh, Ngoc Thanh Nguyen, Sung Wook Baik 5, "Mining top-k frequent patterns from uncertain databases", Springer Science+Business Media, LLC, part of Springer Nature 2020, https://doi.org/10.1007/s10489-019-01622-1

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup

[float or int or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

storePatternsInFile(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsInDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

frequentOneItem()

Extracts the one-length frequent patterns from database

updateTransactions()

Update the transactions by removing non-frequent items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

to convert the user specified value

startMine()

Mining process will start from this function

Execution methods

Terminal command

Format:

(.venv) \$ python3 TUFP.py <inputFile> <outputFile> <minSup>

Example Usage:

(.venv) \$ python3 TUFP.py sampleDB.txt patterns.txt 0.6

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

<pre>from PAMI.uncertainFrequentPattern.basic import TUFP as alg</pre>				
<pre>iFile = 'sampleDB.txt'</pre>				
<pre>minSup = 10 # can also be specified between 0 and 1</pre>				
<pre>obj = alg.TUFP(iFile, minSup)</pre>				
obj.startMine()				
<pre>frequentPatterns = obj.getPatterns()</pre>				
<pre>print("Total number of Frequent Patterns:", len(frequentPatterns))</pre>				
obj.save(oFile)				
<pre>Df = obj.getPatternsAsDataFrame()</pre>				
<pre>memUSS = obj.getMemoryUSS()</pre>				
<pre>print("Total Memory in USS:", memUSS)</pre>				
<pre>memRSS = obj.getMemoryRSS()</pre>				
	(continues on next page)			

(continued from previous page)

```
print("Total Memory in RSS", memRSS)
```

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)

Credits

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \to \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

$\texttt{getMemoryUSS()} \to \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns() \rightarrow Dict[str, float]

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$getRuntime() \rightarrow float$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns

$\texttt{printResults()} \rightarrow \text{None}$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (file) – name of the output file

$startMine() \rightarrow None$

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns

TubeP

class PAMI.uncertainFrequentPattern.basic.TubeP.**TUFP**(*iFile*, *minSup*, *sep*=\t')

Bases: _frequentPatterns

About this algorithm

Description

It is one of the fundamental algorithm to discover top-k frequent patterns in a uncertain transactional database using CUP-Lists.

Reference

Tuong Le, Bay Vo, Van-Nam Huynh, Ngoc Thanh Nguyen, Sung Wook Baik 5, "Mining top-k frequent patterns from uncertain databases", Springer Science+Business Media, LLC, part of Springer Nature 2020, https://doi.org/10.1007/s10489-019-01622-1

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup

[float or int or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

storePatternsInFile(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsInDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

frequentOneItem()

Extracts the one-length frequent patterns from database

updateTransactions()

Update the transactions by removing non-frequent items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert() to convert the user specified value

startMine()
Mining process will start from this function

Execution methods

Terminal command

Format:

(.venv) \$ python3 TUFP.py <inputFile> <outputFile> <minSup>

Example Usage:

(.venv) \$ python3 TUFP.py sampleDB.txt patterns.txt 10.0

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

```
from PAMI.uncertainFrequentPattern.basic import TUFP as alg

iFile = 'sampleDB.txt'

minSup = 10 # can also be specified between 0 and 1

obj = alg.TUFP(iFile, minSup)

obj.mine()

frequentPatterns = obj.getPatterns()

print("Total number of Frequent Patterns:", len(frequentPatterns))

obj.save(oFile)

Df = obj.getPatternsAsDataFrame()

memUSS = obj.getmemoryUSS()

print("Total Memory in USS:", memUSS)

memRSS = obj.getMemoryRSS()

print("Total Memory in RSS", memRSS)

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)
```

Credits

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

$\texttt{getMemoryUSS()} \rightarrow \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$getPatterns() \rightarrow Dict[str, float]$

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type pd.DataFrame

$\texttt{getRuntime()} \rightarrow \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns

$printResults() \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (file) – name of the output file

$startMine() \rightarrow None$

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns

TubeS

PAMI.uncertainFrequentPattern.basic.TubeS.Second(transaction, i)

To calculate the second probability of a node in transaction

Parameters

- transaction transaction in a database
- **i** index of item in transaction

Returns

second probability of a node

class PAMI.uncertainFrequentPattern.basic.TubeS.**TubeS**(*iFile*, *minSup*, *sep*=\t')

Bases: _frequentPatterns

About this algorithm

Description

TubeS is one of the fastest algorithm to discover frequent patterns in a uncertain transactional database.

Reference

Carson Kai-Sang Leung and Richard Kyle MacKinnon. 2014. Fast Algorithms for Frequent Itemset Mining from Uncertain Data. In Proceedings of the 2014 IEEE International Conference on Data Mining (ICDM '14). IEEE Computer Society, USA, 893–898. https://doi.org/10.1109/ ICDM.2014.146

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup

[float or int or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

frequentOneItem()

Extracts the one-length frequent patterns from database

updateTransactions()

Update the transactions by removing non-frequent items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

to convert the user specified value
Execution methods

Terminal command

Format:

(.venv) \$ python3 TubeS.py <inputFile> <outputFile> <minSup>

Example Usage:

(.venv) \$ python3 TubeS.py sampleDB.txt patterns.txt 10.0

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

```
from PAMI.uncertainFrequentPattern.basic import TubeS as alg
iFile = 'sampleDB.txt'
minSup = 10  # can also be specified between 0 and 1
obj = alg.TubeS(iFile, minSup)
obj.mine()
frequentPatterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(frequentPatterns))
obj.save(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine()

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns

printResults()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (file) – name of the output file

updateTransactions(dict1)

Remove the items which are not frequent from transactions and updates the transactions with rank of items

:param dict1 : frequent items with support :type dict1 : dictionary

PAMI.uncertainFrequentPattern.basic.TubeS.printTree(root)

To print the tree with root node through recursion

Parameters

root – root node of tree

Returns

details of tree

UFGrowth

class PAMI.uncertainFrequentPattern.basic.UFGrowth.**UFGrowth**(*iFile*, *minSup*, *sep*=\t')

Bases: _frequentPatterns

Description

It is one of the fundamental algorithm to discover frequent patterns in a uncertain transactional database using PUF-Tree.

Reference

Carson Kai-Sang Leung, Syed Khairuzzaman Tanbeer, "PUF-Tree: A Compact Tree Structure for Frequent Pattern Mining of Uncertain Data", Pacific-Asia Conference on Knowledge Discovery and Data Mining(PAKDD 2013), https://link.springer.com/chapter/10.1007/ 978-3-642-37453-1_2

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup

[float or int or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

frequentOneItem()

Extracts the one-length frequent patterns from database

updateTransactions()

Update the transactions by removing non-frequent items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

to convert the user specified value

startMine()

Mining process will start from this function

Methods to execute code on terminal

Format:

>>> python3 PUFGrowth.py <inputFile> <outputFile> <minSup>

Example:

>>> python3 PUFGrowth.py sampleTDB.txt patterns.txt 3

Note: minSup will be considered in support count or frequency

Importing this algorithm into a python program

```
from PAMI.uncertainFrequentPattern.basic import UFGrowth as alg
obj = alg.UFGrowth(iFile, minSup)
obj.startMine()
frequentPatterns = obj.getPatterns()
print("Total number of Frequent Patterns:", len(frequentPatterns))
obj.save(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getmemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function :return: returning RSS memory consumed by the mining process :rtype: float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function :return: returning USS memory consumed by the mining process :rtype: float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process :return: returning frequent patterns :rtype: dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe :return: returning frequent patterns in a dataframe :rtype: pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process :return: returning total amount of runtime taken by the mining process :rtype: float

mine()

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns

printResults()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file :param outFile: name of the output file :type outFile: csv file

UVECLAT

class PAMI.uncertainFrequentPattern.basic.UVECLAT.**UVEclat**(*iFile*, *minSup*, *sep*='\t') Bases: _frequentPatterns

About this algorithm

•

Description

It is one of the fundamental algorithm to discover frequent patterns in an uncertain transactional database using PUF-Tree.

Reference

Carson Kai-Sang Leung, Lijing Sun: "Equivalence class transformation based mining of frequent itemsets from uncertain data", SAC '11: Proceedings of the 2011 ACM Symposium on Applied ComputingMarch, 2011, Pages 983–984, https://doi.org/10.1145/1982185.1982399

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup

[float or int or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represent the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

storePatternsInFile(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsInDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

frequentOneItem()

Extracts the one-length frequent patterns from database

Execution methods

Terminal command

Format:

(.venv) \$ python3 uveclat.py <inputFile> <outputFile> <minSup>

Example Usage:

(.venv) \$ python3 uveclat.py sampleDB.txt patterns.txt 3

Note: minSup can be specified in support count or a value between 0 and 1.

Calling from a python program

<pre>from PAMI.uncertainFrequentPattern.basic import UVECLAT as alg</pre>
<pre>iFile = 'sampleDB.txt'</pre>
<pre>minSup = 10 # can also be specified between 0 and 1</pre>
<pre>obj = alg.UVEclat(iFile, minSup)</pre>
obj.mine()
<pre>frequentPatterns = obj.getPatterns()</pre>
<pre>print("Total number of Frequent Patterns:", len(frequentPatterns))</pre>
obj.save(oFile)
<pre>Df = obj.getPatternsAsDataFrame()</pre>
<pre>memUSS = obj.getmemoryUSS()</pre>
<pre>print("Total Memory in USS:", memUSS)</pre>

(continues on next page)

(continued from previous page)

```
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine()

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns

printResults()

This function is used to print the results

save(oFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters oFile (*csv file*) – name of the output file

6.2 Uncertain Periodic Frequent Pattern mining

Uncertain periodic frequent pattern mining is a data mining task that involves the discovery of periodic patterns from datasets containing uncertain or probabilistic data. Unlike traditional periodic frequent pattern mining, which deals with deterministic data, uncertain periodic frequent pattern mining addresses the challenges posed by uncertainty in the data, where each item or attribute may have associated probabilities or uncertainties.

Applications: Healthcare, Environmental Monitoring, Financial Forecasting.

Uncertain periodic frequent pattern mining is a data mining task that involves the discovery of periodic patterns from datasets containing uncertain or probabilistic data. Unlike traditional periodic frequent pattern mining, which deals with deterministic data, uncertain periodic frequent pattern mining addresses the challenges posed by uncertainty in the data, where each item or attribute may have associated probabilities or uncertainties.

Applications: Healthcare, Environmental Monitoring, Financial Forecasting.

6.2.1 Basic

UPFPGrowth

class PAMI.uncertainPeriodicFrequentPattern.basic.UPFPGrowth.UPFPGrowth(*iFile*, *minSup*, *maxPer*,

sep = (t')

Bases: _periodicFrequentPatterns

About this algorithm

Description

Basic is to discover periodic-frequent patterns in a uncertain temporal database.

Reference

Uday Kiran, R., Likhitha, P., Dao, MS., Zettsu, K., Zhang, J. (2021).Discovering Periodic-Frequent Patterns in Uncertain Temporal Databases. In: Mantoro, T., Lee, M., Ayu, M.A., Wong, K.W., Hidayanto, A.N. (eds) Neural Information Processing.

ICONIP 2021. Communications in Computer and Information Science, vol 1516. Springer, Cham. https://doi.org/10.1007/978-3-030-92307-5_83

param iFile

str : Name of the Input file to mine complete set of Uncertain Periodic Frequent Patterns

param oFile

str: Name of the output file to store complete set of Uncertain Periodic Frequent patterns

param minSup

float: minimum support thresholds were tuned to find the appropriate ranges in the limited memory

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

param maxper

float : where maxPer represents the maximum periodicity threshold value specified by the user.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of output file

minSup: int or float or str

The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

maxPer: int or float or str

The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

sep: str

This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS: float

To store the total amount of USS memory consumed by the program

memoryRSS: float

To store the total amount of RSS memory consumed by the program

startTime: float

To record the start time of the mining process

endTime: float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

_lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

finalPatterns

[dict] To store the complete patterns

Methods

mine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets()

Scans the dataset and stores in a list format

PeriodicFrequentOneItem()

Extracts the one-periodic-frequent patterns from database

updateTransaction()

Update the database by removing aperiodic items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

To convert the user specified value

removeFalsePositives()

To remove the false positives in generated patterns

Execution methods

Terminal command

```
Format:
```

(.venv) \$ python3 basic.py <inputFile> <outputFile> <minSup> <maxPer>

```
Example Usage:
```

(.venv) \$ python3 basic.py sampleTDB.txt patterns.txt 0.3 4

Note: minSup and maxPer will be considered in support count or frequency

Calling from a python program

<pre>from PAMI.uncertainPeriodicFrequentPattern.basic import UPFPGrowth as alg</pre>
<pre>iFile = 'sampleDB.txt'</pre>
<pre>minSup = 10 # can also be specified between 0 and 1</pre>
<pre>maxPer = 2 # can also be specified between 0 and 1</pre>
obj = alg.UPFPGrowth(iFile, minSup, maxPer)
obj.mine()
<pre>periodicFrequentPatterns = obj.getPatterns()</pre>
<pre>print("Total number of Periodic Frequent Patterns:", len(periodicFrequentPatterns))</pre>
obj.save(oFile)
Df = obj.getPatternsAsDataFrame()
<pre>memUSS = obj.getMemoryUSS()</pre>
<pre>print("Total Memory in USS:", memUSS)</pre>
<pre>memRSS = obj.getMemoryRSS()</pre>
<pre>print("Total Memory in RSS", memRSS)</pre>
<pre>run = obj.getRuntime()</pre>
<pre>print("Total ExecutionTime in seconds:", run)</pre>

Credits

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

$\texttt{getMemoryRSS()} \rightarrow \texttt{float}$

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

```
Return type
float
```

$\texttt{getMemoryUSS()} \rightarrow \texttt{float}$

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

$\texttt{getPatterns()} \rightarrow Dict[str, List[float]]$

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

getPatternsAsDataFrame() \rightarrow DataFrame

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

$\texttt{getRuntime()} \to \texttt{float}$

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine() \rightarrow None

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns.

Returns None

$printResults() \rightarrow None$

This function is used to print the results

save(*outFile: str*) \rightarrow None

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (csv file) - name of the output file

Returns

None

$\texttt{startMine()} \rightarrow None$

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns.

Returns

None

UPFPGrowthPlus

class PAMI.uncertainPeriodicFrequentPattern.basic.UPFPGrowthPlus.UPFPGrowthPlus(*iFile*,

minSup, maxPer, sep=\t')

Bases: _periodicFrequentPatterns

About this algorithm

Description

Basic Plus is to discover periodic-frequent patterns in a uncertain temporal database.

Reference

Palla Likhitha, Rage Veena, Rage Uday Kiran, Koji Zettsu, Masashi Toyoda, Philippe Fournier-Viger, (2023). UPFP-growth++: An Efficient Algorithm to Find Periodic-Frequent Patterns in Uncertain Temporal Databases. ICONIP 2022. Communications in Computer and Information Science, vol 1792. Springer, Singapore. https://doi.org/10.1007/978-981-99-1642-9_16

param iFile

str : Name of the Input file to mine complete set of Uncertain Periodic Frequent Patterns

param oFile

str : Name of the output file to store complete set of Uncertain Periodic Frequent patterns

param minSup

str: minimum support thresholds were tuned to find the appropriate ranges in the limited memory

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

param maxper

floot : where maxPer represents the maximum periodicity threshold value specified by the user.

Attributes

iFile: file

Name of the Input file or path of input file

oFile: file

Name of the output file or path of output file

minSup: int or float or str

The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

maxPer: int or float or str

The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

sep: str

This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS: float

To store the total amount of USS memory consumed by the program

memoryRSS: float

To store the total amount of RSS memory consumed by the program

startTime: float

To record the start time of the mining process

endTime: float

To record the completion time of the mining process

Database: list

To store the transactions of a database in list

mapSupport: Dictionary

To maintain the information of item and their frequency

lno: int

To represent the total no of transaction

tree: class

To represents the Tree class

itemSetCount: int

To represents the total no of patterns

finalPatterns: dict

To store the complete patterns

Methods

startMine()
Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

savePatterns(oFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

updateDatabases()

Update the database by removing aperiodic items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

to convert the user specified value

PeriodicFrequentOneItems()

To extract the one-length periodic-frequent items

Execution methods

Terminal command

Format:

(.venv) \$ python3 UPFPGrowthPlus.py <inputFile> <outputFile> <minSup> <maxPer>

Examples Usage:

(.venv) \$ python3 UPFPGrowthPlus.py sampleTDB.txt patterns.txt 0.3 4

Note: minSup and maxPer will be considered in support count or frequency

Calling from a python program

```
from PAMI.uncertainPeriodicFrequentPattern import UPFPGrowthPlus as alg
iFile = 'sampleDB.txt'
minSup = 10  # can also be specified between 0 and 1
maxPer = 2  # can also be specified between 0 and 1
obj = alg.UPFPGrowthPlus(iFile, minSup, maxPer)
obj.mine()
periodicFrequentPatterns = obj.getPatterns()
print("Total number of uncertain Periodic Frequent Patterns:",________elen(periodicFrequentPatterns))
obj.save(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
```

(continued from previous page)

```
print("Total Memory in RSS", memRSS)
```

run = obj.getRuntime()

print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function.

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine()

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns

printResults()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (*csv file*) – name of the output file

startMine()

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns

PAMI.uncertainPeriodicFrequentPattern.basic.UPFPGrowthPlus.printTree(root)

To print the tree with nodes with item name, probability, timestamps, and second probability respectively.

Parameters

root – Node

Returns

print all Tree with nodes with items, probability, parent item, timestamps, second probability respectively.

6.3 Uncertain Geo-Referenced Frequent Pattern mining

Uncertain geo-referenced frequent pattern mining is a data mining task that involves the discovery of frequent patterns from datasets containing uncertain or probabilistic data with geographic references. In uncertain geo-referenced data, each item or attribute is associated with a geographical location, and uncertainty arises from the probabilistic nature of the data, where the occurrence of events or patterns may vary with associated probabilities or uncertainties.

Applications: Location-based Services, Urban Planning and Development, Emergency Response and Disaster Management.

Uncertain geo-referenced frequent pattern mining is a data mining task that involves the discovery of frequent patterns from datasets containing uncertain or probabilistic data with geographic references. In uncertain geo-referenced data, each item or attribute is associated with a geographical location, and uncertainty arises from the probabilistic nature of the data, where the occurrence of events or patterns may vary with associated probabilities or uncertainties.

Applications: Location-based Services, Urban Planning and Development, Emergency Response and Disaster Management.

6.3.1 Basic

GFPGrowth

class PAMI.uncertainGeoreferencedFrequentPattern.basic.GFPGrowth.GFPGrowth(*iFile*, *nFile*,

minSup, *sep*=t')

Bases: _frequentPatterns

About this algorithm

Description

GFPGrowth algorithm is used to discover geo-referenced frequent patterns in a uncertain transactional database using GFP-Tree.

Reference

Palla Likhitha,Pamalla Veena, Rage, Uday Kiran, Koji Zettsu (2023). "Discovering Georeferenced Frequent Patterns in Uncertain Geo-referenced Transactional Databases". PAKDD 2023. https://doi.org/10.1007/978-3-031-33380-4_3

param iFile

str : Name of the Input file to mine complete set of uncertain Geo referenced Frequent Patterns

param oFile

str : Name of the output file to store complete set of Uncertain Geo referenced frequent patterns

param minSup

str: minimum support thresholds were tuned to find the appropriate ranges in the limited memory

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup: float or int or str

The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

savePatterns(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

frequentOneItem()

Extracts the one-length frequent patterns from database

updateTransactions()

Update the transactions by removing non-frequent items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

to convert the user specified value

startMine()

Mining process will start from this function

Execution methods

Terminal command

Format:

```
(.venv) $ python3 GFPGrowth.py <inputFile> <neighborFile> <outputFile> <minSup>
```

Examples usage:

(.venv) \$ python3 GFPGrowth.py sampleTDB.txt sampleNeighbor.txt patterns.txt 3

Note: minSup will be considered in support count or frequency

Calling from a python program:

```
from PAMI.uncertainGeoreferencedFrequentPattern.basic import GFPGrowth as_
⊶alg
iFile = 'sampleDB.txt'
minSup = 10 # can also be specified between 0 and 1
obj = alg.GFPGrowth(iFile, nFile, minSup)
obj.mine()
Patterns = obj.getPatterns()
print("Total number of Patterns:", len(Patterns))
obj.save(oFile)
Df = obj.getPatternsAsDataFrame()
memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)
```

Credits

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

mine()

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns

printResults()

This function is used to print the result

save(outFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters

outFile (csv file) – name of the output file

startMine()

Main method where the patterns are mined by constructing tree and remove the false patterns by counting the original support of a patterns

CHAPTER

SEVEN

SEQUENTIAL DATABASE

A sequence represents a collection of itemsets (or transactions) in a particular order. A sequence database is a collection of sequences and their sequence identifiers. An example of a geo-referenced transactional database is as follows:

Rules to create a sequence database:

- Items in an itemset have to be seperated by a tab space.
- Itemsets in a sequence are seperated using '-1' as a seperator.
- Each sequence is represented as a line
- The sequence identifier, sid, is not needed to create a sequence database.

Format of a sequence:

```
>>> item1<sep>item2<sep>...<sep>itemA : item1<sep>item2<sep>...<sep>itemB : item1<sep>
_____item2<sep>...<sep>itemC
```

Example:

```
>>> a b c d : a d e : a e f
a b c : b d e : c d e
a e f : c
a e f : a c d : c e
```

7.1 Sequential Frequent Pattern mining

Sequential frequent pattern mining is a data mining technique focused on identifying patterns or subsequences of events that frequently occur together in ordered sequences of data. It involves analyzing datasets where data instances are presented sequentially over time, such as transaction sequences, web clickstreams, biological sequences, or event logs.

Applications: Marketing and User Retention, Process Optimization, Healthcare Monitoring.

Sequential frequent pattern mining is a data mining technique focused on identifying patterns or subsequences of events that frequently occur together in ordered sequences of data. It involves analyzing datasets where data instances are presented sequentially over time, such as transaction sequences, web clickstreams, biological sequences, or event logs.

Applications: Marketing and User Retention, Process Optimization, Healthcare Monitoring.

7.1.1 Basic

SPADE

class PAMI.sequentialPatternMining.basic.SPADE.**SPADE**(*iFile*, *minSup*, *sep*=\t')

Bases: _sequentialPatterns

Description

- SPADE is one of the fundamental algorithm to discover sequential frequent patterns in a transactional database.
- This program employs SPADE property (or downward closure property) to reduce the search space effectively.
- This algorithm employs breadth-first search technique when 1-2 length patterns and depth-first serch when above 3 length patterns to find the complete set of frequent patterns in a transactional database.

Reference

Mohammed J. Zaki. 2001. SPADE: An Efficient Algorithm for Mining Frequent Sequences. Mach. Learn. 42, 1-2 (January 2001), 31-60. DOI=10.1023/A:1007652502315 http://dx.doi.org/10.1023/A:1007652502315

Parameters

- iFile str : Name of the Input file to mine complete set of Sequential frequent patterns
- oFile str : Name of the output file to store complete set of Sequential frequent patterns
- **minSup** float or int or str : minSup measure constraints the minimum number of transactions in a database where a pattern must appear Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

oFile

[str] Name of the output file or the path of output file

minSup: float or int or str

The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

finalPatterns: dict

Storing the complete set of patterns in a dictionary variable

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the transactions of a database in list

_xLenDatabase: dict

To store the datas in different sequence separated by sequence, rownumber, length.

_xLenDatabaseSame

[dict] To store the datas in same sequence separated by sequence, rownumber, length.

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

savePatterns(oFile)

Complete set of frequent patterns will be loaded in to an output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

candidateToFrequent(candidateList)

Generates frequent patterns from the candidate patterns

frequentToCandidate(frequentList, length)

Generates candidate patterns from the frequent patterns

Methods to execute code on terminal



(continues on next page)

(continued from previous page)

```
.. note:: minSup will be considered in times of minSup and count of {\scriptstyle \sqcup} {\scriptstyle {\scriptstyle { \mbox{--}}}} database transactions
```

Importing this algorithm into a python program

<pre>import PAMI.sequentialPatternMining.basic.SPADE as alg</pre>
<pre>obj = alg.SPADE(iFile, minSup)</pre>
obj.startMine()
<pre>sequentialPatternMining = obj.getPatterns()</pre>
<pre>print("Total number of Frequent Patterns:", len(frequentPatterns))</pre>
obj.save(oFile)
<pre>Df = obj.getPatternInDataFrame()</pre>
<pre>memUSS = obj.getMemoryUSS()</pre>
<pre>print("Total Memory in USS:", memUSS)</pre>
<pre>memRSS = obj.getMemoryRSS()</pre>
<pre>print("Total Memory in RSS", memRSS)</pre>
<pre>run = obj.getRuntime()</pre>
<pre>print("Total ExecutionTime in seconds:", run)</pre>

Credits:

The complete program was written by Suzuki Shota under the supervision of Professor Rage Uday Kiran.

Mine()

Frequent pattern mining process will start from here

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type

float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

make1LenDatabase()

To make 1 length frequent patterns by breadth-first search technique and update Database to sequential database

make2LenDatabase()

To make 2 length frequent patterns by joining two one length patterns by breadth-first search technique and update xlen Database to sequential database

make3LenDatabase()

To call each 2 length patterns to make 3 length frequent patterns depth-first search technique

makeNextRow(bs, latestWord, latestWord2)

To make pattern row when two patterns have the latest word in different sequence

:param bs : previous pattern without the latest one :param latestWord : latest word of one previous pattern :param latestWord2 : latest word of other previous pattern

makeNextRowSame(bs, latestWord, latestWord2)

To make pattern row when one pattern have the latestWord1 in different sequence and other(latestWord2) in same

:param bs : previous pattern without the latest one :param latestWord : latest word of one previous pattern in same sequence :param latestWord2 : latest word of other previous pattern in different sequence

makeNextRowSame2(bs, latestWord, latestWord2)

To make pattern row when two patterns have the latest word in same sequence

:param bs : previous pattern without the latest one :param latestWord : latest word of one previous pattern :param latestWord2 : latest word of the other previous pattern

makeNextRowSame3(bs, latestWord, latestWord2)

To make pattern row when two patterns have the latest word in different sequence and both latest word is in same sequence

:param bs : previous pattern without the latest one :param latestWord : latest word of one previous pattern :param latestWord2 : latest word of other previous pattern

makexLenDatabase(rowLen, bs, latestWord)

To make "rowLen" length frequent patterns from pattern which the latest word is in same seq by joining "rowLen"-1 length patterns by depth-first search technique and update xlenDatabase to sequential database

Parameters

rowLen – row length of patterns.

:param bs : patterns without the latest one :param latestWord : latest word of patterns

makexLenDatabaseSame(rowLen, bs, latestWord)

To make 3 or more length frequent patterns from pattern which the latest word is in different seq by depthfirst search technique and update xlenDatabase to sequential database

Parameters

rowLen - row length of previous patterns.

:param bs : previous patterns without the latest one :param latestWord : latest word of previous patterns

printResults()

This function is used to prnt the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters
 outFile (csv file) - name of the output file

startMine()

Frequent pattern mining process will start from here

SPAM

class PAMI.sequentialPatternMining.basic.SPAM.**SPAM**(*iFile*, *minSup*, *sep*=\t')

Bases: _sequentialPatterns

Description

SPAM is one of the fundamental algorithm to discover sequential frequent patterns in a transactional database. This program employs SPAM property (or downward closure property) to reduce the search space effectively. This algorithm employs breadth-first search technique to find the complete set of frequent patterns in a sequential database.

Reference

J. Ayres, J. Gehrke, T.Yiu, and J. Flannick. Sequential Pattern Mining Using Bitmaps. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Edmonton, Alberta, Canada, July 2002.

Parameters

- iFile str : Name of the Input file to mine complete set of Sequential frequent patterns
- oFile str : Name of the output file to store complete set of Sequential frequent patterns

- **minSup** float or int or str : minSup measure constraints the minimum number of transactions in a database where a pattern must appear Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

oFile

[str] Name of the output file or the path of output file

minSup

[float or int or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

finalPatterns

[dict] Storing the complete set of patterns in a dictionary variable

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the sequences of a database in list

_idDatabase

[dict] To store the sequences of a database by bit map

_maxSeqLen:

the maximum length of subsequence in sequence.

Methods

_creatingItemSets():

Storing the complete sequences of the database/input file in a database variable

_convert(value):

To convert the user specified minSup value

make2BitDatabase():

To make 1 length frequent patterns by breadth-first search technique and update Database to sequential database

DfsPruning(items,sStep,iStep):

the main algorithm of spam. This can search sstep and istep items and find next patterns, its

sstep, and its istep. And call this function again by using them. Recursion until there are no more items available for exploration.

Sstep(s):

To convert bit to ssteo bit. The first time you get 1, you set it to 0 and subsequent ones to 1.(like 010101=>001111, 00001001=>00000111)

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

savePatterns(oFile)

Complete set of frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

candidateToFrequent(candidateList)

Generates frequent patterns from the candidate patterns

frequentToCandidate(frequentList, length)

Generates candidate patterns from the frequent patterns

Executing the code on terminal:

```
Format:
(.venv) $ python3 SPAM.py <inputFile> <outputFile> <minSup> (<separator>)
Examples usage:
(.venv) $ python3 SPAM.py sampleDB.txt patterns.txt 10.0
.. note:: minSup will be considered in times of minSup and count of...
```

```
\hookrightarrow database transactions
```

Sample run of the importing code:

import PAMI.sequentialPatternMining.basic.SPAM as alg obj = alg.SPAM(iFile, minSup) obj.startMine() sequentialPatternMining = obj.getPatterns() print("Total number of Frequent Patterns:", len(frequentPatterns)) obj.savePatterns(oFile) Df = obj.getPatternInDataFrame() memUSS = obj.getMemoryUSS() print("Total Memory in USS:", memUSS) memRSS = obj.getMemoryRSS() print("Total Memory in RSS", memRSS) run = obj.getRuntime() print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by Shota Suzuki under the supervision of Professor Rage Uday Kiran.

DfsPruning(items, sStep, iStep)

the main algorithm of spam. This can search sstep and istep items and find next patterns, its sstep, and its istep. And call this function again by using them. Recursion until there are no more items available for exploration.

Attributes

items

[str] The pattrens I got before

sStep

[list] Items presumed to have "sstep" relationship with "items".(sstep is What appears later like a-b and a-c)

iStep

[list] Items presumed to have "istep" relationship with "items" (istep is What appears in same time like ab and ac)

Sstep(s)

To convert bit to Sstep bit. The first time you get 1, you set it to 0 and subsequent ones to 1.(like 010101=>001111, 00001001=>0000111)

:param s:list

to store each bit sequence

Returns

nextS:list to store the bit sequence converted by sstep

countSup(n)

count support

:param n:list

to store each bit sequence

Returns

count: int support of this list

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function :return: returning RSS memory consumed by the mining process :rtype: float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function :return: returning USS memory consumed by the mining process :rtype: float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process :return: returning frequent patterns :rtype: dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe :return: returning frequent patterns in a dataframe :rtype: pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process :return: returning total amount of runtime taken by the mining process :rtype: float

make2BitDatabase()

To make 1 length frequent patterns by breadth-first search technique and update Database to sequential database

printResults()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file :param outFile: name of the output file :type outFile: file

startMine()

Frequent pattern mining process will start from here

prefixSpan

class PAMI.sequentialPatternMining.basic.prefixSpan.**prefixSpan**(*iFile*, *minSup*, *sep*=\t')

Bases: _sequentialPatterns

Description

- Prefix Span is one of the fundamental algorithm to discover sequential frequent patterns in a transactional database.
- This program employs Prefix Span property (or downward closure property) to reduce the search space effectively.

• This algorithm employs depth-first search technique to find the complete set of frequent patterns in a transactional database.

Reference

J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, M. Hsu: Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. IEEE Trans. Knowl. Data Eng. 16(11): 1424-1440 (2004)

Parameters

- iFile str : Name of the Input file to mine complete set of Sequential frequent patterns
- oFile str : Name of the output file to store complete set of Sequential frequent patterns
- minSup float or int or str : minSup measure constraints the minimum number of transactions in a database where a pattern must appear Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float
- **sep** str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[str] Input file name or path of the input file

oFile

[str] Name of the output file or the path of output file

minSup

[float or int or str] The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

startTime

[float] To record the start time of the mining process

endTime

[float] To record the completion time of the mining process

finalPatterns

[dict] Storing the complete set of patterns in a dictionary variable

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

Database

[list] To store the transactions of a database in list

Methods

startMine()
Mining process will start from here

getPatterns() Complete set of patterns will be retrieved with this function savePatterns(oFile) Complete set of frequent patterns will be loaded in to a output file getPatternsAsDataFrame() Complete set of frequent patterns will be loaded in to a dataframe getMemoryUSS() Total amount of USS memory consumed by the mining process will be retrieved from this function getMemoryRSS() Total amount of RSS memory consumed by the mining process will be retrieved from this function getRuntime() Total amount of runtime taken by the mining process will be retrieved from this function candidateToFrequent(candidateList) Generates frequent patterns from the candidate patterns frequentToCandidate(frequentList, length)

Methods to execute code on terminal

Format:

```
(.venv) $ python3 prefixSpan.py <inputFile> <outputFile> <minSup>
```

Generates candidate patterns from the frequent patterns

Example usage:

```
(.venv) $ python3 prefixSpan.py sampleDB.txt patterns.txt 10
```

.. note:: minSup will be considered in support count or frequency

Importing this algorithm into a python program



(continues on next page)
(continued from previous page)

memUSS = obj.getMemoryUSS()
print("Total Memory in USS:", memUSS)
memRSS = obj.getMemoryRSS()
print("Total Memory in RSS", memRSS)
run = obj.getRuntime()
print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by Suzuki Shota under the supervision of Professor Rage Uday Kiran.

Mine()

Frequent pattern mining process will start from here

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns()

Function to send the set of frequent patterns after completion of the mining process

Returns

returning frequent patterns

Return type dict

getPatternsAsDataFrame()

Storing final frequent patterns in a dataframe

Returns

returning frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type float

getSameSeq(startrow)

To get words in the latest sequence

Parameters startrow – the patterns get before

makeNext(sepDatabase, startrow)

To get next pattern by adding head word to next sequence of startrow

Parameters

- **sepDatabase** dict what words and rows startrow have to add it.
- **startrow** the patterns get before

makeNextSame(sepDatabase, startrow)

To get next pattern by adding head word to the latest sequence of startrow

Parameters

- sepDatabase dict what words and rows startrow have to add it
- **startrow** the patterns get before

makeSeqDatabaseFirst(database)

To make 1 length sequence dataset list which start from same word. It was stored only 1 from 1 line.

Parameters

database – To store the transactions of a database in list

makeSeqDatabaseSame(database, startrow)

To make sequence dataset list which start from same word(head). It was stored only 1 from 1 line. And it separated by having head in the latest sequence of startrow or not.

Parameters

- database To store the transactions of a database in list
- **startrow** the patterns get before

makeSupDatabase(database, head)

To delete not frequent words without words in the latest sequence

Parameters

database - list database of lines having same startrow and head word

:param head:list

words in the latest sequence

Returns

changed database

printResults()

This function is used to print the results

save(outFile)

Complete set of frequent patterns will be loaded in to an output file

Parameters outFile (csv file) – name of the output file

serchSame(database, startrow, give)

To get 2 or more length patterns in same sequence.

Parameters

- **database** list To store the transactions of a database in list which have same startrow and head word
- **startrow** list the patterns get before
- give list the word in the latest sequence of startrow

startMine()

Frequent pattern mining process will start from here

7.1.2 closed

bide

7.2 Geo-referenced Frequent Sequence Pattern mining

Geo-referenced frequent sequential pattern mining is a data mining technique focused on discovering patterns or sequences of events that frequently occur in geo-referenced time series data while preserving the spatial and temporal ordering information. It involves analyzing datasets where data instances are geo-referenced.

Applications: Transportation, Environmental Monitoring, Urban Planning or Geographical Phenomena.

Geo-referenced frequent sequential pattern mining is a data mining technique focused on discovering patterns or sequences of events that frequently occur in geo-referenced time series data while preserving the spatial and temporal ordering information. It involves analyzing datasets where data instances are geo-referenced.

Applications: Transportation, Environmental Monitoring, Urban Planning or Geographical Phenomena.

CHAPTER

EIGHT

MULTIPLE TIMESERIES

A timeseries represents an ordered collection of values of an event (or item) over time. A multiple timeseries represents the collection of multiple timeseries gathered from multiple items over a particular duration. Depending on the values stored in a series, a multiple timeseries can be broadly classified into two types:

- Binary multiple timeseries and
- (non-binary) multiple timeseries .

Binary Multiple Timeseries

A binary multiple time series represents the binary data of multiple items split into temporal windows. An example of this series is shown below.

windowID	binary sequences
1	(a,1) (a,3) (b,2) (b,3) (c,2) (c,3)
2	(a,1) (b,1) (b,2) (b,3) (c,1)
3	(a,1) (a,2) (b,1) (b,3) (c,2)
4	(a,1) (b,1) (b,2) (c,3)
5	(a,1)(a,3)(b,3)(c,2)(c,2)
6	(a,1) (a,2) (b,2) (b,3)

Rules to create a binary multiple time series.

- First column must contain an integer representing an windowID.
- Remaining columns must contain items and their timestamps within braces.
- In the braces, starting from left hand side, the first word/letter represents an item and the other word/letter represents an timestamp.
- Columns are seperated with a seperator.
- 'Tab space' is the default seperator. However, transactional databases can be constructed using other seperators, such as comma and space.

Format of a binary multiple time series:

```
>>> windowID<sep>(item,timestamp)<sep>(item,timestamp)<sep>...<sep>(item,_

→timestamp)
```

An example

1	(a,1) (a,3) (b,2) (b,3) (c,2) (c,3)
2	(a,1) (b,1) (b,2) (b,3) (c,1)
3	(a,1) (a,2) (b,1) (b,3) (c,2)
4	(a,1) (b,1) (b,2) (c,3)
5	(a,1) (a,3) (b,3) (c,2) (c,2)
6	(a,1)(a,2)(b,2)(b,3)

8.1 Multiple Partial Periodic Pattern Mining

Multiple partial periodic pattern mining is a data mining technique focused on identifying recurring patterns or sequences of events that occur periodically but may not cover the entire duration of the periodic cycle. It involves analyzing datasets where multiple partial periodic patterns exist, with each pattern representing a subset of events recurring at regular intervals.

Applications: Stock Market Analysis, Healthcare Monitoring, Internet Traffic Analysis.

Multiple partial periodic pattern mining is a data mining technique focused on identifying recurring patterns or sequences of events that occur periodically but may not cover the entire duration of the periodic cycle. It involves analyzing datasets where multiple partial periodic patterns exist, with each pattern representing a subset of events recurring at regular intervals.

Applications: Stock Market Analysis, Healthcare Monitoring, Internet Traffic Analysis.

8.1.1 Basic

PPGrowth

Bases: _partialPeriodicPatterns

About this algorithm

Description

PPGrowth is one of the fundamental algorithm to discover periodic-frequent patterns in a transactional database.

Reference

C. Saideep, R. Uday Kiran, K. Zettsu, P. Fournier-Viger, M. Kitsuregawa and P. Krishna Reddy, "Discovering Periodic Patterns in Irregular Time Series," 2019 International Conference on Data Mining Workshops (ICDMW), 2019,

pp. 1020-1028, doi: 10.1109/ICDMW.2019.00147.

param iFile

str : Name of the Input file to mine complete set of periodic frequent pattern's

param oFile

str : Name of the output file to store complete set of periodic frequent pattern's

param sep

str : This variable is used to distinguish items from one another in a transaction. The default separator is tab space. However, the users can override their default separator.

Attributes

iFile

[file] Name of the Input file or path of the input file

oFile

[file] Name of the output file or path of the output file

minSup: int or float or str

The user can specify minSup either in count or proportion of database size. If the program detects the data type of minSup is integer, then it treats minSup is expressed in count. Otherwise, it will be treated as float. Example: minSup=10 will be treated as integer, while minSup=10.0 will be treated as float

maxPer: int or float or str

The user can specify maxPer either in count or proportion of database size. If the program detects the data type of maxPer is integer, then it treats maxPer is expressed in count. Otherwise, it will be treated as float. Example: maxPer=10 will be treated as integer, while maxPer=10.0 will be treated as float

sep

[str] This variable is used to distinguish items from one another in a transaction. The default separator is tab space or . However, the users can override their default separator.

memoryUSS

[float] To store the total amount of USS memory consumed by the program

memoryRSS

[float] To store the total amount of RSS memory consumed by the program

startTime:float

To record the start time of the mining process

endTime:float

To record the completion time of the mining process

Database

[list] To store the transactions of a database in list

mapSupport

[Dictionary] To maintain the information of item and their frequency

lno

[int] To represent the total no of transaction

tree

[class] To represents the Tree class

itemSetCount

[int] To represents the total no of patterns

finalPatterns

[dict] To store the complete patterns

Methods

startMine()

Mining process will start from here

getPatterns()

Complete set of patterns will be retrieved with this function

save(oFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

getPatternsAsDataFrame()

Complete set of periodic-frequent patterns will be loaded in to a dataframe

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

getRuntime()

Total amount of runtime taken by the mining process will be retrieved from this function

creatingItemSets(fileName)

Scans the dataset and stores in a list format

PeriodicFrequentOneItem()

Extracts the one-periodic-frequent patterns from database

updateDatabases()

Update the database by removing aperiodic items and sort the Database by item decreased support

buildTree()

After updating the Database, remaining items will be added into the tree by setting root node as null

convert()

to convert the user specified value

Execution methods

Terminal command

Format:

```
(.venv) $ python3 PPGrowth.py <inputFile> <outputFile> <minSup> <maxPer>
```

Examples:

(.venv) \$ python3 PPGrowth.py sampleTDB.txt patterns.txt 0.3 0.4

Sample run of importing the code:

from PAMI.periodicFrequentPattern.basic import PPGrowth as alg obj = alg.PPGrowth(iFile, minSup, maxPer) obj.startMine() periodicFrequentPatterns = obj.getPatterns() print("Total number of Periodic Frequent Patterns:", len(periodicFrequentPatterns)) obj.save(oFile) Df = obj.getPatternsAsDataFrame() memUSS = obj.getMemoryUSS() print("Total Memory in USS:", memUSS) memRSS = obj.getMemoryRSS() print("Total Memory in RSS", memRSS) run = obj.getRuntime() print("Total ExecutionTime in seconds:", run)

Credits:

The complete program was written by P.Likhitha under the supervision of Professor Rage Uday Kiran.

Mine()

Mining process will start from this function

getMemoryRSS()

Total amount of RSS memory consumed by the mining process will be retrieved from this function

Returns

returning RSS memory consumed by the mining process

Return type

float

getMemoryUSS()

Total amount of USS memory consumed by the mining process will be retrieved from this function

Returns

returning USS memory consumed by the mining process

Return type float

getPatterns()

Function to send the set of periodic-frequent patterns after completion of the mining process

Returns

returning periodic-frequent patterns

Return type

dict

getPatternsAsDataFrame()

Storing final periodic-frequent patterns in a dataframe

Returns

returning periodic-frequent patterns in a dataframe

Return type

pd.DataFrame

getRuntime()

Calculating the total amount of runtime taken by the mining process

Returns

returning total amount of runtime taken by the mining process

Return type

float

printResults()

This function is used to print the results

save(outFile)

Complete set of periodic-frequent patterns will be loaded in to a output file

Parameters outFile (file) – name of the output file

startMine()

Mining process will start from this function

CHAPTER

NINE

CONTIGUOUS PATTERNS

Contiguous Pattern Mining Definition here

9.1 Contiguous Frequent Patterns

Contiguous Frequent Patterns

Contiguous Frequent Patterns Def Here

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

р	102
PAMI.correlatedPattern.basic.CoMine,37	PAMI.localPeriodicPattern.basic.LPPMDepth,
PAMI.correlatedPattern.basic.CoMinePlus,40	105
PAMI.coveragePattern.basic.CMine, 50	PAMI.multipleMinimumSupportBasedFrequentPattern.basic.CFP
PAMI.coveragePattern.basic.CPPG, 54	30
PAMI.faultTolerantFrequentPattern.basic.FTApr	.iOTHI.multipleMinimumSupportBasedFrequentPattern.basic.CFP 33
PAMI.faultTolerantFrequentPattern.basic.FTFPG 46	rPAMI, partialPeriodicFrequentPattern.basic.GPFgrowth, 110
PAMI.frequentPattern.basic.Apriori,2	<pre>PAMI.partialPeriodicFrequentPattern.basic.PPF_DFS,</pre>
PAMI.frequentPattern.basic.ECLAT, 5	116
PAMI.frequentPattern.basic.ECLATbitset, 10	PAMI.partialPeriodicPattern.basic.GThreePGrowth,
PAMI.frequentPattern.basic.ECLATDiffset, 8	128
PAMI.frequentPattern.basic.FPGrowth, 13	PAMI.partialPeriodicPattern.basic.PPP_ECLAT,
PAMI.frequentPattern.closed.CHARM, 16	124
PAMI.frequentPattern.maximal.MaxFPGrowth, 19	PAMI.partialPeriodicPattern.basic.PPPGrowth,
PAMI.frequentPattern.topk.FAE, 22	120
<pre>PAMI.fuzzyCorrelatedPattern.basic.FCPGrowth, 247</pre>	<pre>PAMI.partialPeriodicPattern.closed.PPPClose, 131</pre>
PAMI.fuzzvFrequentPattern.basic.FFIMiner.243	PAMI.partialPeriodicPattern.maximal.Max3PGrowth,
PAMI.fuzzyGeoreferencedFrequentPattern.basic. 251	FFSPMiner ^{1,35} PAMI.partialPeriodicPattern.topk.k3PMiner,
PAMI.fuzzyGeoreferencedPeriodicFrequentPatter 260	n.basic.fGPFPMiner, PAMI.partialPeriodicPatternInMultipleTimeSeries.PPGrowth,
PAMI.fuzzyPeriodicFrequentPattern.basic.FPFPM 256	liner, ³²⁶ PAMI.periodicCorrelatedPattern.basic.EPCPGrowth,
<pre>PAMI.georeferencedPartialPeriodicPattern.basi 175</pre>	.c.STEclat ^{1,43} PAMI.periodicFrequentPattern.closed.CPFPMiner,
PAMI.geoReferencedPeriodicFrequentPattern.bas	sic.GPFPM ^{frl} er, PAMI.periodicFrequentPattern.maximal.MaxPFGrowth.
DAMT highlitilityFrequentDattern basic HIFTM	84
188	PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFPMiner,
PAMI.highUtilityGeoreferencedFrequentPattern.	basic.SHOFIM, PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP,
PAMI.highUtilitvSpatialPattern.basic.HDSHUIM.	92
197	<pre>PAMI.recurringPattern.basic.RPGrowth, 160</pre>
PAMI.highUtilitySpatialPattern.basic.SHUIM, 201	<pre>PAMI.relativeFrequentPattern.basic.RSFPGrowth, 25</pre>
PAMI.highUtilitySpatialPattern.topk.TKSHUIM, 205	<pre>PAMI.relativeHighUtilityPattern.basic.RHUIM,</pre>
PAMI.localPeriodicPattern.basic.LPPGrowth,96 PAMI.localPeriodicPattern.basic.LPPMBreadth,	<pre>PAMI.sequentialPatternMining.basic.prefixSpan, 318</pre>

- PAMI.sequentialPatternMining.basic.SPADE, 310 PAMI.sequentialPatternMining.basic.SPAM, 314 PAMI.sequentialPatternMining.closed.bide, 323 PAMI.stablePeriodicFrequentPattern.topK.TSPIN, 156 PAMI.uncertainFrequentPattern.basic.CUFPTree, 267 PAMI.uncertainFrequentPattern.basic.PUFGrowth, 271 PAMI.uncertainFrequentPattern.basic.TubeP, 279 PAMI.uncertainFrequentPattern.basic.TubeS, 283 PAMI.uncertainFrequentPattern.basic.TUFP, 275 PAMI.uncertainFrequentPattern.basic.UFGrowth, 287 PAMI.uncertainFrequentPattern.basic.UVECLAT, 290 PAMI.uncertainGeoreferencedFrequentPattern.basic.GFPGrowth, 303 PAMI.uncertainPeriodicFrequentPattern.basic.UPFPGrowth, 294 PAMI.uncertainPeriodicFrequentPattern.basic.UPFPGrowthPlus, 299
- PAMI.weightedFrequentNeighbourhoodPattern.basic.SWFPGrowth, 235
- PAMI.weightedFrequentPattern.basic.WFIM, 226
- PAMI.weightedFrequentRegularPattern.basic.WFRIMiner,

231

INDEX

A	CPFPMiner (class in PAMI.periodicFrequentPattern.closed.CPFPMiner),
addChild() (PAMI.periodicFrequentPattern.basic.PSGrow	vth.Node ⁸¹
<i>method</i>), 68	CPPG (class in PAMI.coveragePattern.basic.CPPG), 54
<pre>additemset() (PAMI.highUtilitySpatialPattern.topk.TKSI</pre>	GreateConduitionalTree() (PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth.Tree
addTransaction() (PAMI.localPeriodicPattern.basic.LPL	PGrowth.Theethod), 114
method), 101	<pre>createPrefixTree() (PAMI.localPeriodicPattern.basic.LPPGrowth.Tree</pre>
addTransaction() (PAMI.partialPeriodicFrequentPattern method), 114	n.basic.GP F gfbwdh:Tree createPrefixTree() (PAMI.partialPeriodicFrequentPattern.basic.GPFg
Apriori (class in PAMI.frequentPattern.basic.Apriori), 2	<i>method</i>), 114
	createTransaction()
В	(PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Dataset
backtrackingEFIM() (PAMI.highUtilitySpatialPattern.to)	pk.TKSHUIM.TKSHUIM creatingCoverageItems()
memou), 209	(PAMI.coveragePattern.basic.CMine.CMine
С	method), 52
	CUFPTree (class in PAMI.uncertainFrequentPattern.basic.CUFPTree),
CalculatelP (class in	267
	growin),
()	U
(PAMI.highUtilitySpatialPattern.topk.TKSHUIM.	Dataset (class in PAMI.highUtilitySpatialPattern.topk.TKSHUIM), TKSHUIM ₂₀₅
candidateCount (PAMI.highUtilitySpatialPattern.topk.TK	deleteNode() (PAMI.localPeriodicPattern.basic.LPPGrowth.Tree XSHUIM.TKSHUIM method), 101
attribute), 210	deleteNode() (PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth.T
CFPGrowth (class in PAMI.multipleMinimumSupportBased	FrequentPattern basic (FPGrowth), method), 114
30 CEDCount h Disc	DfsPruning() (PAMI.sequentialPatternMining.basic.SPAM.SPAM
CFPGFOWthPlus (class in DAML multiple Minimum Support Daved Encourant D	method), 317
	ulern.busic.CFFGrowinPlus),
CUADM (class in PAMI frequent Pattern closed CHAPM)	E
16	ECLAT (class in PAMI.frequentPattern.basic.ECLAT), 5
CMino (class in PAMI coverage Pattern basic CMine) 50	ECLATbitset (class in
Comine (class in PAMI correlated Pattern basic Comine), 50	PAMI.frequentPattern.basic.ECLATbitset),
37	10
CoMinePlus (class in PAMI correlated Pattern basic CoMin	ECLATDiffset (class in
	PAMI.frequentPattern.basic.ECLATDiffset), 8
conditionalTransactions() (in module	EFIM (class in PAMI.highUtilityPattern.basic.EFIM), 216
PAMI.periodicFrequentPattern.basic.PSGrowth),	Element (<i>class in PAMI.fuzzyCorrelatedPattern.basic.FCPGrowth</i>), 247
count Sun() (PAMI sequential Pattern Mining basic SPAM	sendTime (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHUIM
method), 318	attribute), 210

EPCPGrowth (class in PAMI.periodicCorrelatedPattern.bas 143	sigetMeRoryR\$5)() (PAMI.correlatedPattern.basic.CoMine.CoMine method), 39
-	getMemoryRSS() (PAMI.correlatedPattern.basic.CoMinePlus.CoMinePlus
F	<i>method</i>), 42
FAE (class in PAMI.frequentPattern.topk.FAE), 22	getMemoryRSS() (PAMI.coveragePattern.basic.CMine.CMine
FCPGrowth (class in PAMI.fuzzyCorrelatedPattern.basic.Fe	CPGrowth)method), 52
247	<pre>getMemoryRSS() (PAMI.coveragePattern.basic.CPPG.CPPG</pre>
FFIMiner (class in PAMI.fuzzyFrequentPattern.basic.FFIM	<i>Ainer</i>), <i>method</i>), 56
243	getMemoryRSS() (PAMI.faultTolerantFrequentPattern.basic.FTApriori.FT.
FFSPMiner (class in PAMI.fuzzyGeoreferencedFrequentPat	ttern.basic.FRESPONDAD),
$\frac{251}{100000000000000000000000000000000000$	getMemoryRSS() (PAMI.fault Iolerant Frequent Pattern. basic. FIFPGrowth.
FGPFPMIner (class in PAMI.juzzyGeoreferenceaPeriodicFr	equentFattemalDasuc. #UPFPMiner),
200 final Dattorne (DAMI high Utility Spatial Dattorn took TV	getnemoryRSS()(FAMI.Jrequentratern.busic.Aprion.Aprion
attribute) 210	GetMemoryBSS() (PAMI frequentPattern basic FCI AT FCI AT
findSeparator() (PAMI partialPeriodic FrequentPattern	havic GPEnnethnikhanarataPFListvar?
method) 115	aetMemoryRSS() (PAMI frequentPattern basic FCLAThitset FCLAThitset
findSenarator() (PAMI partialPeriodicFrequentPattern	basic GPFmethal polierate PFTreever?
method) 116	getMemoryRSS() (PAML frequent Pattern basic ECLATDiffset ECLATDiffs
fixNodeLinks() (PAML local Periodic Pattern basic LPPC	Growth Treemethod). 9
method). 101	getMemorvRSS() (PAMI.frequentPattern.basic.FPGrowth.FPGrowth
fixNodeLinks() (PAML partial Periodic Frequent Pattern.	basic.GPFemethbdrek5
method). 114	getMemoryRSS() (PAMI.frequentPattern.closed.CHARM.CHARM
FPFPMiner (class in PAMI.fuzzyPeriodicFrequentPattern.b	asic.FPFPMintbod), 18
256	getMemoryRSS() (PAMI.frequentPattern.maximal.MaxFPGrowth.MaxFPG
FPGrowth (class in PAMI.frequentPattern.basic.FPGrowth)), method), 21
13	getMemoryRSS() (PAMI.frequentPattern.topk.FAE.FAE
FTApriori (class in PAMI.faultTolerantFrequentPattern.bd	asic.FTAprimeil/10d), 24
43	getMemoryRSS() (PAMI.fuzzyCorrelatedPattern.basic.FCPGrowth.FCPG
${\tt FTFPGrowth} \ (class \ in \ PAMI. fault Tolerant Frequent Pattern. line of the state of the$	basic.FTFPGethouh), 250
46	getMemoryRSS() (PAMI.fuzzyFrequentPattern.basic.FFIMiner.FFIMiner
G	<i>method</i>), 245
G	getMemoryRSS() (PAMI.fuzzyGeoreferencedFrequentPattern.basic.FFSPM
generateAllPatterns()	method), 254
(PAMI.coveragePattern.basic.CMine.CMine method), 52	getMemoryRSS() (PAMI.fuzzyGeoreferencedPeriodicFrequentPattern.basic method), 263
generatePFListver2 (class in	getMemoryRSS() (PAMI.fuzzyPeriodicFrequentPattern.basic.FPFPMiner.I
PAMI. partial Periodic Frequent Pattern. basic. GPF	Fgrowth), method), 259
115	getMemoryRSS() (PAMI.georeferencedFrequentPattern.basic.SpatialECLA
generatePFTreever2 (class in	method), 1/0
PAMI.partialPeriodicFrequentPattern.basic.GPF	<i>got Memory RSS()</i> (PAMI. georeferenced Partial Periodic Pattern. basic. STEch
	method), 1/8
genPatterns() (PAMI.coveragePattern.basic.CMine.CMi	method) 174
method), 52	memory DSS() (PAMI high Utility Frequent Pattern basic HUEIM HUEIN
getCh11d() (PAMI.localPerioalcPattern.basic.LPPGrowt	method) 191
method), 100	(method), 191
getChild() (PAMI.partialPerioaiCFrequentPattern.basic.	method) 196
memou), 115	InetMemoryRSS() (PAMI highUtilityPattern basic EFIM EFIM
method) 215	method). 219
getLastPosition() (PAMI high I tility Spatial Pattern ton	kger Nenony RSS6 APAMI. high Utility Pattern. basic. HMiner. HMiner
method), 215	method), 222
getMaxItem() (PAMI.highUtilitvSpatialPattern.tonk.TKS	HIGHMANNEY RSS() (PAMI.highUtilityPattern.basic.UPGrowth.UPGrowth
method), 206	method), 225

act Momenty DEE () (DAMI bigh Utility, Spatial Dattom basis UDSHMH	ALLOGY IN MAM I as summing a Dattern has is DDC now the DDC now the
gethemoryRSS() (PAMI.nigholilliySpallalPallern.basic.Hgshtem	arykasta (Maami. recurringPuttern. basic. KPGrowin. KPGrowin method) 162
memory DSS() (DAML high Litility, Spatial Dattom hasis SLILI Meet	memou), 105
getnemoryRSS() (PAMI.nignOniitySpanaiPattern.basic.SFgetimen	GLYRDS() (PAMI.relativeFrequentPattern.basic.KSFPGrowtn.KSFF
method), 204	method), 28
getMemoryRSS() (PAMI.highUtilitySpatialPattern.topk.TKgettWen	difysky blog (PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUIM
<i>method</i>), 210	<i>method</i>), 186
getMemoryRSS() (PAMI.localPeriodicPattern.basic.LPPGrgettHem	ሪዮርጽፍፍስ) (PAMI.sequentialPatternMining.basic.prefixSpan.prefixS
method), 99	<i>method</i>), 321
getMemoryRSS() (PAMI.localPeriodicPattern.basic.LPPMBetMbh	dryRSSB()e4HAMI.sequentialPatternMining.basic.SPADE.SPADE
<i>method</i>), 104	<i>method</i>), 312
getMemoryRSS() (PAMI.localPeriodicPattern.basic.LPPMDepMeh	BPYRDs ()) (PAMI.sequentialPatternMining.basic.SPAM.SPAM
method), 108	<i>method</i>), 318
getMemorvRSS() (PAMI.multipleMinimumSupportBasedFractivent	atteRs.\$63idPAMIGrabielP.CFRCGrowthuentPattern.basic.SPPEclat.SF
method) 32	method) 155
aet Memory RSS() (PAMI multiple Minimum Support Rased Freet Matthe	with Rf. Solid Pa MI Gradule Poling di HPG manuti Politern basic SPP Growth
method) 36	method) 151
nemou), 50	memou), 151 EmpRe (ADDA Malythable Deviedie Energy ent Dattawa ton V TSDIN TSDI
gethemory KSS() (FAMI.paritairenouicriequentratiern.ouged.tem	ogy x x x x y r r z y www.studierentouicriequentruttern.topK.15r11.15r11
method), 112	methoa), 159
getMemoryRSS() (PAMI.partialPeriodicFrequentPattern.bdged.Hem	omyRSS(B)(PHMIS.uncertainFrequentPattern.basic.CUFPIree.CUFI
method), 119	<i>method</i>), 270
getMemoryRSS() (PAMI.partialPeriodicPattern.basic.GThgePNem	outhRSEKipe(PAMInuthcertainFrequentPattern.basic.PUFGrowth.PUI
<i>method</i>), 130	<i>method</i>), 274
getMemoryRSS() (PAMI.partialPeriodicPattern.basic.PPPgECMem	б Р�FR<u>S</u>£ЮЦЯТ MI.uncertainFrequentPattern.basic.TubeP.TUFP
method), 126	<i>method</i>), 282
getMemoryRSS() (PAMI.partialPeriodicPattern.basic.PPPGetMem	BPVRS& (WthPAMI.uncertainFrequentPattern.basic.TubeS.TubeS
method), 122	method). 286
aet Memory RSS() (PAML partial Periodic Pattern closed PPRet Men	CHRRSS (e) (PAML uncertain Frequent Pattern basic. TUFP. TUFP
method) 134	method) 278
ant Memory RSS () (PAMI nartial Periodic Pattern maximal Mar RMan	anorth Staff 3 PCM with certain Frequent Pattern basic UFGrowth UFG
wethod) 129	worthod) 200
memora), 130	memou), 290
getnemoryRSS() (PAMI.partialPerioaicPattern.topk.kSPMgeenkenk	outyress() (PANII.uncertainFrequentPattern.basic.UVECLAI.UVEC
method), 141	method), 293
getMemoryRSS() (PAMI.partialPeriodicPatternInMultiple/Igenessen	or.yRSSf@w(tPAIMPIGinnoeth ainGeoreferencedFrequentPattern.basic.GI
method), 329	<i>method</i>), 307
getMemoryRSS() (PAMI.periodicCorrelatedPattern.basic.EpecMem	onyRSHORPAMIthncertainPeriodicFrequentPattern.basic.UPFPG
method), 146	<i>method</i>), 297
getMemoryRSS() (PAMI.periodicFrequentPattern.basic.PFgetMem	dHyRSLAT(PAMI.uncertainPeriodicFrequentPattern.basic.UPFPG
method), 76	<i>method</i>), 302
getMemoryRSS() (PAMI.periodicFrequentPattern.basic.PFdetMem	brPRBGydwPAMI.weightedFrequentNeighbourhoodPattern.basic.SV
method), 63	method). 238
aetMemoryRSS() (PAMI periodic Frequent Pattern basic PERATIMENT	hPhRSRFP(PAMIhPinshtedFrequentPattern basic WFIM WFIM
method) 67	method) 229
ant Momony DSS() (PAMI pariodia Fraquent Pattern hasia PEAD # More	FEMENS() (DAMI weighted Erequent Pequilar Pattern hasis WEPIMir
getriemory KSS() (I Ami. periodici requenti ditern. ousic. I regentiem	woth od) 224
meinoa), 80	meinoa), 254
getMemoryRSS() (PAMI.periodicFrequentPattern.basic.PSgetawem	orycosymp (PAMI.correlatedPattern.basic.Comine.Comine
method), 71	method), 39
getMemoryRSS() (PAMI.periodicFrequentPattern.closed.CgetMem	aryCIBEPMiN&MI.correlatedPattern.basic.CoMinePlus.CoMinePlus
method), 83	method), 42
$\verb+getMemoryRSS()(PAMI.periodicFrequentPattern.maximalgletamentPattern.maxima$	GiryUS5.(MaHAMIncovidragePattern.basic.CMine.CMine
method), 87	method), 53
getMemoryRSS() (PAMI.periodicFrequentPattern.topk.kPFgPeMMmem	da PyTISM(i) (i)HAMTIPMinen gePattern.basic.CPPG.CPPG
method), 91	method), 56
getMemoryRSS() (PAMI.periodicFrequentPattern.topk.TopkBETPeriodicFrequentPattern.topkBETPeriodicFrequentPattern.topk.TopkBETPeriodicFrequentPattern.topk.TopkBETPeriodicFrequentPattern.topk.TopkBETPeriodicFrequentPattern.topk.TopkBETPeriodicFrequentPattern.topk.TopkBETPeriodicFrequentPattern.topkBETPeriodicFrequentPattern.topkBETPeriodicFrequentPattern.topkBETPeriodicFrequentPattern.topkBETPeriodicFrequentPattern.topkBETPeriodicFrequentPattern.topkBETPeriodicFrequentPattern.topkBETPeriodicFrequentPattern.topkBETPeriodicFrequentPattern.topkBETPeriodicFrequentPattern	orkyUSSTopHHAMPCfnarkflolerantFrequentPattern.basic.FTApriori.FT
method). 94	method), 45
····// ·	

<pre>gettemoryUSS () (PML fjequemPattern.basic_Apriori.AprigetNemoryUSS () (PML multipleMinimumSupporfBasedFrequentPattern method), 36 gettemoryUSS () (PML frequemPattern.basic_ECLAT.ECIgEtNemoryUSS () (PML multipleMinimumSupporfBasedFrequentPattern method), 35 gettemoryUSS () (PML frequentPattern.basic_ECLAT.ECIgEtNemoryUSS () (PML multipleMinimumSupporfBasedFrequentPattern method), 35 gettemoryUSS () (PML frequentPattern.basic_ECLAT.DijsgetEdBas/DSSC) (PML multipleMinimumSupporfBasedFrequentPattern method), 12 gettemoryUSS () (PML frequentPattern.basic_ECLAT.DijsgetEdBas/DSSC) (PML martialPeriodicFrequentPattern.basic.GPF.pDF method), 12 gettemoryUSS () (PML frequentPattern.basic_FPG rowth.gGbasedFullSSC) (PML martialPeriodicFattern.basic.GThreePG rowth. method), 13 gettemoryUSS () (PML frequentPattern.basic.FPG rowth.gGbasedFullSSC) (PML martialPeriodicPattern.basic.FPF_DE method), 15 gettemoryUSS () (PML frequentPattern.basic.FPG rowth.gGbasedFullSSC) (PML partialPeriodicPattern.basic.FPF_DE method), 13 gettemoryUSS () (PML frequentPattern.maximal.MaxFPGgevtIndsyPUSS () (PML partialPeriodicPattern.basic.FPF_DECLAT.PPF method), 13 gettemoryUSS () (PML frequentPattern.basic.FFG gettemoryUSS () (PML partialPeriodicPattern.basic.FPF DC or method), 24 gettemoryUSS () (PML frequentPattern.basic.FFG gettemoryUSS () (PML partialPeriodicPattern.basic.FPF DC or method), 250 gettemoryUSS () (PML fizzyCorrelatedPattern.basic.FFG gettemoryUSS () (PML partialPeriodicPattern.basic.FPF DC or method), 24 gettemoryUSS () (PML fizzyCorrelatedPattern.basic.FFG gettemoryUSS () (PML partialPeriodicPattern.basic.FFFG gettemoryUSS () (PML partialPeriodicPattern.basic.FPFC DC rese.PPC method), 25 gettemoryUSS () (PML fizzyCorrelatedPattern.basic.FFG gettemoryUSS () (PML partialPeriodicPattern.basic.FPFC DC rese.PPC method), 25 gettemoryUSS () (PML fizzyCorrelatedPattern.basic.FFG gettemoryUSS () (PML periodicPattern.basic.FFFG rowth.FF method), 25 gettemoryUSS () (PML fizzyCorrelatedPattern.basic.FFG gettemoryUSS () (PML periodicFr</pre>		
 method), 49 method), 40 method), 4 method), 4 method), 4 method), 4 method), 4 method), 32 getKemoryUSC (PAML/requentPattern.basic.ECLAT.EClgFtHemoryUSC) (PAML/nullipleMinimumSupportBasedFrequentPattern method), 7 method), 7 method), 10 getKemoryUSC (PAML/requentPattern.basic.ECLAT.DiffsgetKhas/byKSC) (PAML partialPeriodicFrequentPattern.basic.CPF_pDF method), 19 method), 10 method), 10 method), 11 getKemoryUSC (PAML/requentPattern.basic.ECLATDiffsgetKhas/byKSC) (PAML partialPeriodicFrequentPattern.basic.CPF_pDF method), 10 method), 10 method), 10 getKemoryUSC (PAML/requentPattern.basic.FPGrowth.JBg60EemoryUSSC) (PAML partialPeriodicPattern.basic.CPF_pDF method), 21 method, 12 getKemoryUSC (PAML/requentPattern.closed.CHARM.CgbtMboryUSSC) (PAML partialPeriodicPattern.basic.PPF_DCLAT.PPF method), 21 getKemoryUSSC) (PAML/requentPattern.closed.CHARM.CgbtMboryUSSC) (PAML partialPeriodicPattern.basic.PPP_Close.PPPC method), 21 getKemoryUSSC) (PAML/requentPattern.basic.FCPfgetNaskyBFSS) (PAML partialPeriodicPattern.basic.PPPClose.PPPC method), 23 getKemoryUSSC) (PAML/grequentPattern.basic.FCPfgetNaskyBFSS) (PAML partialPeriodicPattern.basic.PPPClose.PPPC method), 24 getKemoryUSSC (PAML/grequentPattern.basic.FCPfgetNaskyBFSS) (PAML partialPeriodicPattern.basic.PPPClose.PPPC method), 25 getKemoryUSSC (PAML/grequentPattern.basic.FCPfgetNaskyBFSS) (PAML partialPeriodicPattern.basic.FPPCgrowth.PF method), 25 getKemoryUSSC (PAML/greq/FequentPattern.basic.FCPfgetNaskyBFSSS) (PAML partialPeriodicPattern.basic.FPPCGrowth method), 13 getKemoryUSSC (PAML/greq/FequentPattern.basic.FCPfgetNaskyBFSSSS) (PAML partialPeriodicPattern.basic.FPPCGrowth method), 14 ge	getMemoryUSS() (PAMI.faultTolerantFrequentPattern.basigetMemo	ryWBSETHPMUWabalPeriodicPattern.basic.LPPMDepth.LPPMDe
getXenoryUSS () (PMM frequentPattern basic. Apriori. Aprior Aprior Aprior Aprior Aprice Construction of the construction of th	method), 49	method), 108
method), 32 method), 7 method), 7 method), 7 method), 7 method), 7 method), 12 method), 12 method), 19 gettemoryUSSO (PAMI, frequentPattern. basic. ECLATDistsget@BakabbyeSO (PAMI, partialPeriodicFrequentPattern. basic. GPF grow method), 19 gettemoryUSSO (PAMI, frequentPattern. basic. ECLATDistsget@BakabbyeSO (PAMI, partialPeriodicFrequentPattern. basic. GPF grow method), 19 gettemoryUSSO (PAMI, frequentPattern. basic. ECLATDistsget@BakabbyeSO (PAMI, partialPeriodicFrequentPattern. basic. PFF_DF method), 19 gettemoryUSSO (PAMI, frequentPattern. basic. EFG growth. Jgb0EmetryUSSO (PAMI, partialPeriodicPattern. basic. GPF growth. Jgb0EmetryUSSO (PAMI, frequentPattern. basic. PFG rowth. Jgb0EmetryUSSO (PAMI, partialPeriodicPattern. basic. PFF_CLAT.PFF method), 13 gettemoryUSSO (PAMI, frequentPattern. closed. CHARM. CgbAtMetoryUSSO (PAMI, partialPeriodicPattern. basic. PFP_CLAT.PFF method), 21 gettemoryUSSO (PAMI, frequentPattern. topk.FAE.FAE getNemoryUSSO (PAMI, partialPeriodicPattern. closed. PFPC close. PFPC method), 24 gettemoryUSSO (PAMI, frequentPattern. topk.FAE.FAE getNemoryUSSO (PAMI, partialPeriodicPattern. closed. PFPC close. PFPC method), 24 gettemoryUSSO (PAMI, frequentPattern. basic. FCHYperNethodYpTEGOvEMMI, partialPeriodicPattern. topk.k3PM inter.k3PM int	getMemoryUSS() (PAMI.frequentPattern.basic.Apriori.Apr iget Memo:	ryUSS() (PAMI.multipleMinimumSupportBasedFrequentPattern.l
gettemoryUSS () (PMJ, IrequentPattern, basic, FCLATE/GgetHemoryUSS () (PMJ, partialPeriodicFrequentPattern, basic, GPF grow method), 12 gettemoryUSS () (PMJI, IrequentPattern, basic, FCLATDit/sgdtfUkbaGyUKSS () (PMI, partialPeriodicFrequentPattern, basic, GPF grow method), 12 gettemoryUSS () (PMII, IrequentPattern, basic, FCLATDit/sgdtfUkbaGyUKSS () (PAMI, partialPeriodicFrequentPattern, basic, GPF grow method), 13 gettemoryUSS () (PAMI, IrequentPattern, basic, FFG rowth, IGCHMI, partialPeriodicPattern, basic, GPF growth, GPMI, partialPeriodicPattern, basic, GPF growth, GPMI, partialPeriodicPattern, basic, CPP ECLAT.PFF method), 15 gettemoryUSS () (PAMI, IrequentPattern, closed, CHARM, GPtMWoryUSS () (PAMI, partialPeriodicPattern, basic, PPP Growth, DPPP method), 16 gettemoryUSS () (PAMI, IrequentPattern, topk, FAE, FAE gettemoryUSS () (PAMI, IrequentPattern, basic, FFIM gettiRMMY fly656 () PAMI, partialPeriodicPattern, topk, k3P Miner, k3PMiner, k3PG, (PAMI, Jaczy GeoreferencedFrequentPattern, basic, FFIM gettRMMMY fleXiBMI fieldWinerIndWinerLifePattern, basic, PFPC Growth, PF method), 265 gettemoryUSS () (PAMI, Jaczy GeoreferencedFrequentPattern, basic, MEMMiner, basic, DEPCGrowth, method), 255 gettemoryUSS () (PAMI, Jaczy GeoreferencedFrequentPattern, basic, SFIMMiner, basic, CFPC Growth, PF method), 269 gettemoryUSS () (PAMI, Jaczy GeoreferencedFrequentPattern, basic, SFIMMiner, basic, PFPG crowth, PF method), 259 get	method), 4	method), 32
method), 7 method), 7 method), 7 method), 7 method), 12 getKenoryUSSO (PAMI, frequentPattern, basic. ECLATDissgd201448451948SO (PAMI, partialPeriodicFrequentPattern, basic. CPF grow method), 112 getKenoryUSSO (PAMI, frequentPattern, basic. ECLATDisfgd20144851948SO (PAMI, partialPeriodicFrequentPattern, basic. PPF_DF method), 15 getKenoryUSSO (PAMI, frequentPattern, basic. FFG owth, LigktManoryUSSO (PAMI, partialPeriodicPattern, basic. GThreePGrowth, C method), 15 getKenoryUSSO (PAMI, frequentPattern, closed, CHARM CigktManoryUSSO (PAMI, partialPeriodicPattern, basic. PPP_ECLAT.PPF method), 15 getKenoryUSSO (PAMI, frequentPattern, closed, CHARM CigktManoryUSSO (PAMI, partialPeriodicPattern, basic.PPP_GOVEN, PPP_method), 21 getKenoryUSSO (PAMI, frequentPattern, topk, FAE_FAE getKenoryUSSO (PAMI, partialPeriodicPattern, closed, PPPClose, PPPC method), 21 getKenoryUSSO (PAMI, frequentPattern, topk, FAE_FAE getKenoryUSSO (PAMI, partialPeriodicPattern, topk, S2PMinerk3PMine	getMemoryUSS() (<i>PAMI.frequentPattern.basic.ECLAT.ECIg</i> &#Memo:	ryUSS() (PAMI.multipleMinimumSupportBasedFrequentPattern.l
getXmoryUSS () (PML/requentPattern.basic.FCLATDifsgetXikaXiyWSS () (PAML partialPeriodicFrequentPattern.basic.GPF grow method,) 12 method, 20 method, 21 method, 20 method,	method), 7	method), 36
 method), 12 method), 112 method), 112 method), 119 getMemoryUSSO (PAMI, frequentPattern, basic. EPCLATDiff getMemoryUSSO (PAMI, partialPeriodicFrequentPattern, basic. GThreePGrowth. GetMemoryUSSO (PAMI, partialPeriodicPattern, basic. GThreePGrowth. Common VUSSO (PAMI, partialPeriodicPattern, basic. GThreePGrowth. Common VUSSO (PAMI, partialPeriodicPattern, basic. PPP_ECLAT.PPF method), 130 getMemoryUSSO (PAMI, frequentPattern, anximal. Max FP GevNih degVBEG2040M II, partialPeriodicPattern, basic. PPP_ECLAT.PPF method), 21 method), 24 method), 24 method), 134 method), 132 getMemoryUSSO (PAMI, frequentPattern, topk. FAE.FAE getMemoryUSSO (PAMI, partialPeriodicPattern, aximal.Max3PGrowth method), 250 getMemoryUSSO (PAMI, fizzyOrrelatePattern, basic. FFIMigetNitiAtMyRSSD (PAMI, partialPeriodicPattern, topk.k3PMiner.k3PMine method), 250 getMemoryUSSO (PAMI, fizzyGeoreferencedFrequentPattergbMsmdf)SSBSD (PAMIMperiodicOviPAttern.topk.k3PMiner.k3PMiner.k3PMiner.k3PMine, 255 getMemoryUSSO (PAMI, fizzyGeoreferencedFrequentPattergbMsmdf)SSBSD (PAMIMperiodicOviPAttern.topk.k3PMiner.EPFCPGrowth. method), 263 getMemoryUSSO (PAMI, fizzyGeoreferencedFrequentPattern, basig SplentoffyUSSBSD, PAMIMperiodicOviPAttern.topsic.EPFCPGrowth.PF method), 265 getMemoryUSSO (PAMI, georeferencedFrequentPattern, basig SplentoffyUSSBSD, PAMIMperiodicOviPAttern.topsic.EPFCPGrowth.PF method), 265 getMemoryUSSO (PAMI, fizzyPeriodicFrequentPattern, basig SplentoffyUSSBSD, PAMIMperiodicPerquentPattern, basic.EPFCPGrowth.PF <	getMemoryUSS() (PAMI.frequentPattern.basic.ECLATbitseg.eicle.ho	bitises() (PAMI.partialPeriodicFrequentPattern.basic.GPF growth
<pre>getMemoryUSS () (PAML/requentPattern.basic.ECLATD/iffgetNikib/FUJSS0 () (PAML.partialPeriodicFrequentPattern.basic.PFP_DF method), 19 getMemoryUSS () (PAML/requentPattern.basic.FPGrowth.fgR0MemdryUSS () (PAML.partialPeriodicPattern.basic.GThreePGrowth.C method), 13 getMemoryUSS () (PAML/requentPattern.closed.CHARM.CgbtMinoryUSS () (PAML.partialPeriodicPattern.basic.PFP_ECLAT.PPF method), 18 getMemoryUSS () (PAML/requentPattern.maximal.MaxFPCgenNikudAyUBSC/04MML.partialPeriodicPattern.basic.PPPGrowth.PPO method), 21 getMemoryUSS () (PAML/requentPattern.maximal.MaxFPCgenNikudAyUBSC/04MML.partialPeriodicPattern.basic.PPPGrowth.PPO method), 21 getMemoryUSS () (PAML/requentPattern.basic.FFIMgetNikudAyUBSC/04MML.partialPeriodicPattern.closed.PPPClose.PPPC method), 24 getMemoryUSS () (PAML/fuzyCorrelatedPattern.basic.FFIMgetNikufAyUSS () (PAML.partialPeriodicPattern.toksdPMPRChose.PPPC method), 25 getMemoryUSS () (PAML/fuzyFrequentPattern.basic.FFIMgetNikufAyUSS () (PAML.partialPeriodicPattern.toksdPMPRChose. PPC method), 250 getMemoryUSS () (PAML/fuzyGeoreferencedFrequentPattergebKaufAyUSMAyUEANHIPMinoidPeriodicPattern.toksic.EPCPGrowth. method), 250 getMemoryUSS () (PAML/fuzyGeoreferencedFrequentPattergebKaufAyUSMAyUEANHIPMinoidFordicPattern.hasic.FFECLAT.PFE method), 255 getMemoryUSS () (PAML/fuzyGeoreferencedPeriodicFrequentPattern.basic.PFCCLAT.PFE method), 150 getMemoryUSS () (PAML/fuzyGeoreferencedPeriodicFrequentPattern.basic.FFECLAT.PFE method), 170 getMemoryUSS () (PAML/fuzyGeoreferencedPrequentPattern.basic.ffENDHIMI/GENHIPEriodicFrequentPattern.basic.FFECLAT.PFE method), 170 getMemoryUSS () (PAML/fuzyGeoreferencedPeriodicFrequentPattern.basic.FFECLAT.PFE method), 174 getMemoryUSS () (PAML/fuzyGeoreferencedPeriodicFrequentPattern.basic.YFFOInter.FFE method), 174 getMemoryUSS () (PAML/fuzyGeoreferencedPeriodicFrequentPattern.basic.FFFMiner.FSFECLAT.PFE method), 174 getMemoryUSS () (PAML/fuzyGeoreferencedPeriodicFrequentPattern.basic.PFPMiner.FPF method), 194 getMemoryUSS () (PAML/fuzyGeoreferencedFreque</pre>	<i>method</i>), 12	<i>method</i>), 112
method), 9 method), 19 method), 10 method), 15 method), 15 method), 15 method), 15 method), 15 method), 15 method), 15 method), 16 method), 16 method), 12 getMemoryUSS () (PAML/partialPeriodicPattern.basic.PPP_ECLAT.PPF method), 18 getMemoryUSS () (PAML/requentPattern.inaximal.MaxFPCgetMethodyUSS) (PAML/partialPeriodicPattern.basic.PPP_ECLAT.PPF method), 12 getMemoryUSS () (PAML/requentPattern.inaximal.MaxFPCgetMethodyUSS) (PAML/partialPeriodicPattern.basic.PPPGrowth.PPPC method), 12 getMemoryUSS () (PAML/requentPattern.inaximal.MaxFPCgetMethodyUSSC) (PAML/partialPeriodicPattern.closed.PPPClose.PPPC method), 13 getMemoryUSS () (PAML/itzzyCorrelatedPattern.basic.FCPgetMethodyUSSC) (PAML/partialPeriodicPattern.inaximal.Max3PGrowth method), 25 getMemoryUSS () (PAML/itzzyCorrelatedPattern.basic.FFIMigetMethodyTr5GOvMetAMLpartialPeriodicPattern.topk.k3PMiner.k3PMine method), 24 getMemoryUSS () (PAML/itzzyCoreferencedFrequentPattergebMethodyTr5GOvMetAMLpartialPeriodicPattern.topk.k3PMiner.k3PMine method), 25 getMemoryUSS () (PAML/itzzyGeoreferencedFrequentPattergebMethodyTr5GOvMetAMLpartialPeriodicPattern.hasic.EPCPGrowth. method), 263 getMemoryUSS () (PAML/itzzyGeoreferencedFrequentPattergebMethodyTr5GOVMethodPhotiodEV#iPMAinedPattern.basic.EPCPGrowth. method), 263 getMemoryUSS () (PAML/itzzyGeoreferencedFrequentPattern.basigSthatiofyUSS() (PAML/itzerGeoreferencedFrequentPattern.basigSthatiofyUSS() (PAML/itzerGeoreferencedFrequentPattern.basigSthatiofyUSS() (PAML/itzerGeoreferencedFrequentPattern.basigSthatiofyUSS() (PAML/itzerGeoreferencedFrequentPattern.basigSthatiofyUSS() (PAML/itzerGeoreferencedFrequentPattern.basigSthatiofyUSS() (PAML/itzerGeoreferencedFrequentPattern.basigSthatiofyUSS() (PAML/itzerGeoreferencedFrequentPattern.basigSthatiofyUSS() (PAML/itzerGeoreferencedFrequentPattern.basig.SthatiofyUSS() (PAML/itzerGeoreferencedFrequentPattern.basigSthatiofyUSS() (PAML/itzerGeorefPattern.basig.PFPCrOPFM method), 170 method), 170 method), 170 getMemoryUSS () (PAML/itglWithigFattern.basig.EFIMEtfWMEthogVUSS ()	getMemoryUSS() (PAMI.frequentPattern.basic.ECLATDiffsgre#N@md	FDJ\$Se() (PAMI.partialPeriodicFrequentPattern.basic.PPF_DFS
<pre>getMemoryUSS () (PAMI,frequentPattern.basic.FPGrowth.fgetShemdryUSS () (PAMI,partialPeriodicPattern.basic.GThreePGrowth.C method), 15 method), 15 method), 15 method), 15 method), 12 getMemoryUSS () (PAMI,frequentPattern.closed.CHARM.CgetMtMoryUSS () (PAMI,partialPeriodicPattern.basic.PPP_ECLAT.PPF method), 13 getMemoryUSS () (PAMI,frequentPattern.maximal.MaxFPGetMtMatgEtEXEQNUMMI,partialPeriodicPattern.basic.PPPGrowth.PPPC method), 24 getMemoryUSS () (PAMI,frequentPattern.topk.FAE.FAE getMemoryUSS () (PAMI,partialPeriodicPattern.closed.PPPClose.PPPC method), 25 getMemoryUSS () (PAMI,fuzzyCorrelatedPattern.basic.FFIM getMtEmotyUSS () (PAMI,partialPeriodicPattern.maximal.Max3PGrowth method), 134 getMemoryUSS () (PAMI,fuzzyFrequentPattern.basic.FFIM getMtEmotyUSS () (PAMI,partialPeriodicPattern.topk.k3PMinerk3PMine method), 25 getMemoryUSS () (PAMI,fuzzyGeoreferencedFrequentPattergeMtemotifySESMtPatterNMIthiniafPeriodicPattern.topk.k3PMinerk3PMine method), 25 getMemoryUSS () (PAMI,fuzzyGeoreferencedPeriodicFrequentPattergeMtemotifySESMtPatterNMIthiniafPeriodicPattern.basic.EPCPGrowth. method), 26 getMemoryUSS () (PAMI,fuzzyGeoreferencedPeriodicFrequentPattern.basic&GUMMtEmotyUSS () (PAMI,fuzzyFeriodicFrequentPattern.basic.FFECLAT.PFE method), 259 getMemoryUSS () (PAMI,fuzzyGeoreferencedPeriodicFrequentPattern.basic/BCMtMtEmotyUSSQ) (PAMI,fuzzyFeriodicFrequentPattern.basic.FFECLAT.PFE method), 70 getMemoryUSS () (PAMI,fuzzyGeoreferencedFrequentPattern.basic/SUMMtEmotyUSSQ) (PAMI,fuzzyFeriodicFrequentPattern.basic.FFFCLAT.PFE method), 71 getMemoryUSS () (PAMI,fuzzyGeoreferencedFrequentPattern.basic/SUMMtEmotyUSSQ) (PAMI,fuzzyFeriodicFrequentPattern.basic.FFFCLAT.PFE method), 70 getMemoryUSS () (PAMI,fuzzyHighUtilityFrequentPattern.basic.JEMMtEmotyUSSQ) (PAMI,fuzzyFeriodicFrequentPattern.basic.FFFCLAT.FFE method), 71 getMemoryUSS () (PAMI,fuzyHighUtilityFrequentPattern.basic.JEMEMtEmotyUSSQ) (PAMI,fuzyHighUtilityFeriodicFrequentPattern.basic.FFFPQFFFT method), 71 getMemoryUSS () (PAMI,fuzyHighUtilityFeriodicFreq</pre>	method), 9	method), 119
method), 13 method), 13 getMemoryUSSO (PAMI/frequentPattern.closed.CHARM.CGEtMMmoryUSSO (PAMI.partialPeriodicPattern.basic.PPP_ECLAT.PPF method), 12 getMemoryUSSO (PAMI/frequentPattern.topk.FAE.FAE getMemoryUSSO (PAMI.partialPeriodicPattern.basic.PPPGrowth.PPPG method), 24 getMemoryUSSO (PAMI/frequentPattern.topk.FAE.FAE getMemoryUSSO (PAMI.partialPeriodicPattern.closed.PPPClose.PPPC method), 12 getMemoryUSSO (PAMI/frequentPattern.topk.FAE.FAE getMemoryUSSO (PAMI.partialPeriodicPattern.tosed.PPPClose.PPPC method), 13 getMemoryUSSO (PAMI/fuzzyCorrelatedPattern.basic.FCPGetMethodyBEtSO (PAMI.partialPeriodicPattern.topk.k3PMiner.k3PMiner method), 138 getMemoryUSSO (PAMI/fuzzyCorrelatedPattern.basic.FFIMigetMitidEtSO (PAMI.partialPeriodicPattern.topk.k3PMiner.k3PMiner method), 138 getMemoryUSSO (PAMI.fuzzyCorrelatedPattern.basic.FFIMigetMitidEtSO (PAMI.partialPeriodicPattern.topk.k3PMiner.k3PMiner method), 138 getMemoryUSSO (PAMI.fuzzyCorreferencedFrequentPattegekMathtySKSMetAtMMIIpMitidPeriodicPattern.topk.k3PMiner.k3PMiner method), 255 getMemoryUSSO (PAMI.fuzzyCoreferencedFrequentPattern.basiceFCPGrowth method), 163 getMemoryUSSO (PAMI.fuzzyCoreferencedFrequentPattern.basiceFCPGrowth method), 176 getMemoryUSSO (PAMI.fuzzyCoreferencedFrequentPattern.basiceFCPGrowth.PFC method), 176 getMemoryUSSO (PAMI.fuzzyCoreferencedFrequentPattern.basiceFFEGXQSUBAMMI.fieldMit.feriodicFrequentPattern.basic.FFFGrowthPut method), 178 getMemoryUSSO (PAMI.fuzyCoreferencedFrequentPattern.basic.FFEMCSQSUBAMMI.fieldMit.feriodicFrequentPattern.basic.FFFMC.FFFPM method), 178 getMemoryUSSO (PAMI.fugUtilityFerquentPattern.basic.FFIM.fefMitMethodySUSGWMIMALFIElMMit.feriodicFrequentPattern.basic.FFFMC.FFFPM method), 178 getMemoryUSSO (PAMI.fugUtilityPattern.basic.EFIM.EFGMMEthodySUSGWMIMALFIElMMit.feriodicFrequentPattern.basic.FFFMC.FFPM method), 179 getMemoryUSSO (PAMI.highUtilityPattern.basic.EFIM.EFGMMethodySUSGWMIMALFIElMMit.feriodicFrequentPattern.topk.kPFPMiner.KPFF method), 191 getMemoryUSSO (PAMI.hig	getMemoryUSS() (PAMI.frequentPattern.basic.FPGrowth.FgeGMemory	ryUSS() (PAMI.partialPeriodicPattern.basic.GThreePGrowth.GT
<pre>getHemoryUSS() (PAMI.frequentPattern.closed.CHARM.CBERMEMOryUSS() (PAMI.partialPeriodicPattern.basic.PPP_ECLAT.PPF method), 18 method), 18 getHemoryUSS() (PAMI.frequentPattern.maximal.MaxFPGgetMinduyBEKS/wfMMI.partialPeriodicPattern.basic.PPPGrowth.PPPC method), 21 getHemoryUSS() (PAMI.frequentPattern.topk.FAE.FAE getMemoryUSS() (PAMI.partialPeriodicPattern.closed.PPPClose.PPPC method), 24 getHemoryUSS() (PAMI.fuzzyCorrelatedPattern.basic.FFIMigetMinduyBEKS() (PAMI.partialPeriodicPattern.topk.k3PMiner.k3PMine method), 134 getHemoryUSS() (PAMI.fuzzyCorrelatedPattern.basic.FFIMigetMinduyBESS() (PAMI.partialPeriodicPattern.topk.k3PMiner.k3PMine method), 250 getHemoryUSS() (PAMI.fuzzyGeoreferencedFrequentPattegekMetadryVSSS() detXMIPIMticdPeriodicPatternInMultipleTimeSeries.PFI method), 250 getHemoryUSS() (PAMI.fuzzyGeoreferencedPeriodicFrequgentBetweedVyVSSS() detXMIPIMticdPeriodicPatternInMultipleTimeSeries.PFI method), 253 getHemoryUSS() (PAMI.fuzzyGeoreferencedPeriodicFrequgentBetweedVyVSSS() detXMIPIMitedPattern.basic.PFPCCMorwth. method), 263 getHemoryUSS() (PAMI.fuzzyFeriodicFrequentPattern.basigeRPEMIMityVSSQRIPAMIRperiodicFrequentPattern.basic.PFPCCMorwth. method), 160 getHemoryUSS() (PAMI.fuzzyFeriodicFrequentPattern.basigeRPEMIMityVSSQRIPAMIRperiodicFrequentPattern.basic.PFPGrowth.PF. method), 170 getHemoryUSS() (PAMI.georeferencedPartialPeriodicPattggtMixatoryVSSRIVMMIRperiodicFrequentPattern.basic.PFPGrowthPIL method), 171 getHemoryUSS() (PAMI.georeferencedPartialPeriodicFattggtMixatoryVSKRIVMMIRperiodicFrequentPattern.basic.PFPGrowthPIL method), 174 getHemoryUSS() (PAMI.highUtilityFrequentPattern.basic.FFIM.ergMXHemoryUSS() (PAMI.georeferencedFrequentPattern.basic.VFPGN.method), 170 getHemoryUSS() (PAMI.highUtilityFrequentPattern.basic.FFIM.ErgMXHemoryUSS() (PAMI.highUtilityFrequentPattern.basic.FFFMC.pFPM method), 174 getHemoryUSS() (PAMI.highUtilityFrequentPattern.basic.FFIM.ErgMXHemoryUSS() (PAMI.highUtilityPattern.basic.FFFMC.pFPM method), 190 getHemoryUSS() (PAMI.highUtilityPattern.basic.HFF</pre>	method). 15	method). 130
 method), 18 method), 126 method), 126 method), 121 method), 121 method), 21 method), 21 method), 21 getKenoryUSS () (PAMI frequentPattern.topk.FAE.FAE getKenoryUSS () (PAMI frequentPattern.topk.FAE.FAE getKenoryUSS () (PAMI fuzzyCorrelatedPattern.basic.FCPQerMethod), 134 getKenoryUSS () (PAMI fuzzyCorrelatedPattern.basic.FCPQerMethod), 135 getKenoryUSS () (PAMI fuzzyCorrelatedPattern.basic.FFIM getXinitidSyDESG) (PAMI partialPeriodicPattern.topk.k3PMinerk3PMine method), 240 getKenoryUSS () (PAMI fuzzyCorrelatedPattern.basic.FFIM getXinitidSyDESG) (PAMI partialPeriodicPattern.topk.k3PMinerk3PMine method), 250 getKenoryUSS () (PAMI fuzzyGeoreferencedFrequentPattergehKinitidSyDESG) (PAMI partialPeriodicPattern.topk.k3PMinerk3PMine method), 255 getKenoryUSS () (PAMI fuzzyGeoreferencedFrequentPattergehKinitidSyDESG) (PAMI fuzzyGeoreferencedPrequentPattern.basigeKEtHMittidePeriodicFoEtHMintePattern.basic.EPCPGrowth. method), 255 getKenoryUSS () (PAMI fuzzyPeriodicFrequentPattern.basigeKEtHMittyDSSQD(PAMI fuzzyPeriodicFrequentPattern.basic.PFECLAT.PFE: method), 253 getKenoryUSS () (PAMI fuzzyPeriodicFrequentPattern.basigeKEtHMittyDSSQD(PAMI fuzieFieldIcFrequentPattern.basic.PFPGrowth.PF method), 170 getKenoryUSS () (PAMI georeferencedFrequentPattern.basigeStHmitidyDSSQD(PAMI periodicFrequentPattern.basic.PFPGrowthPE method), 174 getKenoryUSS () (PAMI georeferencedFrequentPattern.basic.EFUM method), 63 getKenoryUSS () (PAMI highUtilityPattern.basic.EFUM EFightKendYWSSQ) (PAMI periodicFrequentPattern.basic.PFPGrowthPE method), 174 getKenoryUSS () (PAMI highUtilityPattern.basic.EFUM EFightKendYWSSQ) (PAMI periodicFrequentPattern.basic.PFPGrowthPEG method), 174 getKenoryUSS () (PAMI highUtilityPattern.basic.EFUM EFightKendYWSSQ)	getMemoryUSS() (PAMI frequent Pattern closed. CHARM. Chert Memo	rvUSS() (PAML partial Periodic Pattern, basic, PPP ECLAT, PPP 1
get HemoryUSS () (PAMI.frequentPattern.maximal.MaxFPGpentIndingUBSG) (dfMI.frequentPattern.basic.PPPGrowth.PPPC method), 121 get NemoryUSS () (PAMI.frequentPattern.topk.FAE.FAE get NemoryUSS () (PAMI.partialPeriodicPattern.closed.PPPClose.PPPC method), 134 get NemoryUSS () (PAMI.fuzzyCorrelatedPattern.basic.FCPGetKnehtGyDEGO (PAMI.partialPeriodicPattern.maximal.Max3PGrowth method), 250 get NemoryUSS () (PAMI.fuzzyCorrelatedPattern.basic.FCPGetKnehtGyDEGO (PAMI.partialPeriodicPattern.topk.k3PMiner.k3PMiner.method), 250 get NemoryUSS () (PAMI.fuzzyFrequentPattern.basic.FFIM getXEikGtyDEGO (PAMI.partialPeriodicPattern.topk.k3PMiner.k3PMiner.method), 250 get NemoryUSS () (PAMI.fuzzyGeoreferencedFrequentPattergebKsindifySESG) (PRMIPhiedofKoPDFbAinadPattern.basic.EPCPGrowth. method), 255 get NemoryUSS () (PAMI.fuzzyGeoreferencedPrequentPattergebKsindifySESG) (PRMIPhiedofKoPDFbAinadPattern.basic.EPCPGrowth. method), 250 get NemoryUSS () (PAMI.fuzzyPeriodicFrequentPattern.basigeREikikty/USSG) (PRMIPhiedofKoPDFbAinadPattern.basic.EPCPGrowth. method), 150 get NemoryUSS () (PAMI.georeferencedPrequentPattern.basigeREikikty/USSG) (PRMIPhiedofKoPDFbAinadPattern.basic.FFFGCLAT.PFEV method), 170 get NemoryUSS () (PAMI.georeferencedPrequentPattern.basigeStatiatifyUSSG) (PRMIFigeriodicFrequentPattern.basic.FFFGrowth.PFF method), 178 get NemoryUSS () (PAMI.georeferencedPreiodicFrequentPattern.basicSfBEGOSfBEAMI.fifeFbBlidFirequentPattern.basic.FFFMC.PFPM method), 178 get NemoryUSS () (PAMI.georeferencedPreiodicFrequentPattern.basicSfBEGOSfBEAMI.fifeFbBlidFirequentPattern.basic.PFPGrowth.PFF method), 178 get NemoryUSS () (PAMI.georeferencedPreiodicFrequentPattern.basicSfBEGOSfBEAMI.fifeFbBlidFirequentPattern.basic.PFPG.PFPM method), 178 get NemoryUSS () (PAMI.highUtilityFrequentPattern.basic.HightMethoryUSS () (PAMI.periodicFrequentPattern.basic.PFPG.pFPM method), 179 get NemoryUSS () (PAMI.highUtilityPattern.basic.FFIM.EFightMethoryUSS () (PAMI.periodicFrequentPattern.topk.KPFPMiner.KPFI method), 220 get NemoryUSS () (PAMI.hig	method) 18	method) 126
 methody, 21 methody, 21 methody, 21 methody, 24 methody, 24 methody, 24 getMenoryUSS () (PAMI, frequentPattern.topk.FAE.FAE getMenoryUSS () (PAMI, frequentPattern.topk.FAE.FAE getMenoryUSS () (PAMI, fuzzyCorrelatedPattern.basic.FCHGetNehchGyTSEGO(MFAMI, partialPeriodicPattern.maximal.Max3PGrowth method), 24 getMenoryUSS () (PAMI, fuzzyFrequentPattern.basic.FFIM getSitiAfVisS () (PAMI, partialPeriodicPattern.topk.k3PMiner.k3PMine method), 25 getMenoryUSS () (PAMI, fuzzyGeoreferencedFrequentPattergekMenthyStS () (PAMI partialPeriodicPattern.topk.k3PMiner.k3PMine method), 25 getMenoryUSS () (PAMI, fuzzyGeoreferencedFrequentPattergekMenthyStS () (PAMI, fuzizyGeoreferencedPeriodicFrequentPattergekMenthyStS () (PAMI, fuzizyGeoreferencedPeriodicFrequentPattern.basic.EPCPGrowth. method), 253 getMenoryUSS () (PAMI, fuzzyPeriodicFrequentPattern.basigeKREikMyINSGO(PAMIfigEikMicFrequentPattern.basic.PFEGrowth.PF method), 76 getMenoryUSS () (PAMI, georeferencedFrequentPattern.basigeKREikMyINSGO(PAMIfigEikMicFrequentPattern.basic.PFPGrowth.PF method), 76 getMenoryUSS () (PAMI, georeferencedFrequentPattern.basigeKREikMyINSGO(PAMIfigEikMicFrequentPattern.basic.PFPGrowth.PF method), 174 getMenoryUSS () (PAMI, georeferencedFrequentPattern.basic.EffMEMiofioStyDSGD(PAMIfigEikMicFrequentPattern.basic.PFPGrowthPluc method), 174 getMenoryUSS () (PAMI, highUtilityFrequentPattern.basic.EffMEMiofiyUSS () (PAMI, periodicFrequentPattern.basic.PFPMioreAppi method), 20 getMenoryUSS () (PAMI, highUtilityFattern.basic.EffM.EffMtMemoryUSS () (PAMI, periodicFrequentPattern.basic.PFPMioreAppi method), 19 getMenoryUSS () (PAMI, highUtilityFattern.basic.EFIM.EffMtMemoryUSS () (PAMI, periodicFrequentPattern.basic.PFPMioreAppi method), 21 getMenoryUSS () (PAMI, highUtilityPattern.basic.UPGrowth	aetMemoryUSS() (PAMI frequentPattern maximal MaxFP Creater Mem	nemou), 120 ny HRS/rbuPA MI partial Periodic Pattern basic PPPG rowth PPPGr
 Michael, 124 Michael, 134 getMenozyUSS () (PAMI, frequentPattern. topk.FAE.FAE getMenozyUSS () (PAMI, partialPeriodicPattern.closed.PPPClose.PPPC method), 24 method), 134 getMenozyUSS () (PAMI, fuzzyCorrelatedPattern.basic.FFIMgetHikidKiyBS () (PAMI, partialPeriodicPattern.maximal.Max3PGrowth method), 250 getMenozyUSS () (PAMI, fuzzyFrequentPattern.basic.FFIMgetHikidKiyBS () (PAMI, partialPeriodicPattern.topk.k3PMiner.k3PMiner.method), 241 getMenozyUSS () (PAMI, fuzzyGeoreferencedFrequentPattergebMenofKySESKiyAERNRIPMinelPeriodicPattern.topk.k3PMiner.k3PMiner.method), 255 getMenozyUSS () (PAMI, fuzzyGeoreferencedPeriodicFrequgetBetwaw KisS () (PAMIperiodicPetiternInMultipleTimeSeries.PI method), 263 getMenozyUSS () (PAMI, fuzzyGeoreferencedPeriodicFrequgetBetwaw KisS () (PAMIperiodicFrequentPattern.basic.PFECLAT.PFE: method), 159 getMenozyUSS () (PAMI, georeferencedFrequentPattern.basigeReitbidKyUSSG/KIAMKiperiodicFrequentPattern.basic.PFECLAT.PFE: method), 170 getMenozyUSS () (PAMI, georeferencedPeriodicFrequentPattern.basicSylItAMKiperiodicFrequentPattern.basic.PFFGrowth.PF. method), 173 getMenozyUSS () (PAMI, georeferencedPeriodicFrequentPattern.basicSylItAMKipeFibMitirequentPattern.basic.PFPGrowth.PF. method), 173 getMenozyUSS () (PAMI, georeferencedPeriodicFrequentPattern.basic.SylItAMKI periodicFrequentPattern.basic.PFPGrowth.PF. method), 174 getMenozyUSS () (PAMI, highUtilityFrequentPattern.basic.HightHiddHyUSS() (PAMI, periodicFrequentPattern.basic.PFPGrowth.PF. method), 180 getMenozyUSS () (PAMI, highUtilityFrequentPattern.basic.LightMethodKyUSS() (PAMI, periodicFrequentPattern.basic.PFPM.perPM method), 80 getMenozyUSS () (PAMI, highUtilityFattern.basic.LightMethodYUSS () (PAMI, periodicFrequentPattern.basic.PFPM.periodicFrequentPattern.basic.PFPM.periodicFrequentPattern.basic.PFPM method), 91	method) 21	method) 122
Sectientary USS O (FAML) requent alternation. FEIN Sectientary USS O (FAML) artial Periodic Pattern. toxical FFF Closer FFE method), 124 Sectientary USS O (FAML) fuzzy Correlated Pattern. basic. FFI Miget Nethod YOSS O (FAML) partial Periodic Pattern. topk.k3P Miner.k3P Miner method), 250 Sectientary USS O (FAML) fuzzy Correlated Pattern. basic. FFI Miget Nethod YOSS O (FAML) partial Periodic Pattern. topk.k3P Miner.k3P Miner method), 246 Sectientary USS O (FAML) fuzzy Georeferenced Frequent Patterge bit Kein GYSSS Kight & RMI Phineid Periodic Pattern. InMultiple Time Series. Pl method), 253 Sectientary USS O (FAML) fuzzy Georeferenced Frequent Patterge bit Kein GYSSS Kight & RMI Phineid Periodic Pattern. InMultiple Time Series. Pl method), 263 Sectientary USS O (FAML) fuzzy Georeferenced Frequent Pattern. basige KEinking YUSS GY (FAML) Frequent Pattern. basic. EPC PG rowth. method), 263 Sectientary USS O (FAML) georeferenced Frequent Pattern. basige KEinking YUSS GYRAMILGE Frequent Pattern. basic. PFF CLAT. PFE method), 170 Sectientary USS O (FAML) georeferenced Frequent Pattern. basige Specific Mixing SPTS & Kiggy Wath Mit Gei YuKang Kiggy Wath Mit georeferenced Periodic Prequent Pattern. basic SPTS & Kiggy Wath Mit Gei YuKang Kiggy Wath Mit Ge	actMomorpulies() (DAMI frequentDattern tonk FAF FAF actMomo	memou), 122
 metinolo, 1:24 metinolo, 1:24 metinolo, 1:35 gettemoryUSS () (PAMI.fuzzyCorrelatedPattern.basic.FCPlgetMethodyTBS(6)(PRAMI.partialPeriodicPattern.maximal.Max3PGrowth method), 250 gettemoryUSS () (PAMI.fuzzyFrequentPattern.basic.FFIMigetINiitoMyNSS () (PAMI.partialPeriodicPattern.topk.k3PMiner.k3PMine method), 255 gettemoryUSS () (PAMI.fuzzyGeoreferencedFrequentPattegebMethodyStSK)()(PRMMIperiodIcOeffedMatinedPattern.basic.EPCPGrowth. method), 263 gettemoryUSS () (PAMI.fuzzyGeoreferencedPeriodicFrequgetEtwows/NSK)() (PRMIMperiodIcOeffedMatinedPattern.basic.FFECLAT.PFE method), 263 gettemoryUSS () (PAMI.fuzzyPeriodicFrequentPattern.basigeREitMyUSS()(PRMMIperiodIcOeffedMatinedPattern.basic.PFECLAT.PFE method), 176 gettemoryUSS () (PAMI.georeferencedPrequentPattern.basigeSteniotfyUSS()(SMAMILperiodicFrequentPattern.basic.PFECLAT.PFE method), 170 gettemoryUSS () (PAMI.georeferencedPartialPeriodicPrequentPattern.basic.PFPGrowth.PF. method), 170 gettemoryUSS () (PAMI.georeferencedPartialPeriodicPrequentPattern.basic.PFPGrowthPIue method), 174 gettemoryUSS () (PAMI.georeferencedPartialPeriodicFrequentPattern.basic.PFPGrowthPIue method), 178 gettemoryUSS () (PAMI.highUtilityFrequentPattern.basic.LFFMC.pFPM method), 174 gettemoryUSS () (PAMI.highUtilityGeoreferencedFrequentPattern.basic.SS(1) (WMMILperiodicFrequentPattern.basic.PFPMC.PFPM method), 191 gettemoryUSS () (PAMI.highUtilityGeoreferencedFrequentFielMethemoxyUSS () (PAMI.periodicFrequentPattern.closed.CPFPMiner.CF method), 196 gettemoryUSS () (PAMI.highUtilityPattern.basic.LFIM.EFightMetmoryUSS () (PAMI.highUtilityPattern.basic.LFIM.EFightMetmoryUSS () (PAMI.highUtilityPattern.basic.LFIM.EFightMetmoryUSS () (PAMI.highUtilityPattern.basic.LFIM.EFightMetmoryUSS () (PAMI.highUtilityPattern.basic.LFIM.EFightMetmoryUSS () (PAMI.highUtilityPattern.basic.LFIM.EFigh	gethemoly035() (IAMI.Jrequenti utern.topk.TAE.TAE gethemo.	method) 124
get temory USS () (PAMI, Juzy Correlated and term, Dasis, CFC, Pget Chemicary USS () (PAMI, Juzy Correlated and term, Dasis, CFC Juget temory USS () (PAMI, Juzy Frequent Pattern, Dasis, CFFIM get third), 141 get temory USS () (PAMI, Juzy Georeferenced Frequent Pattern, Dasis, CFFIM get third), 141 get temory USS () (PAMI, Juzy Georeferenced Frequent Pattern, Dasis, CFC Juget temory, JuSS () (PAMI, Juzy Georeferenced Periodic Frequent Pattern, Dasis, C) (PAMI, Juzy Ceoreferenced Periodic Frequent Pattern, Dasis, C) (PAMI, Juzy Ceoreferenced Periodic Frequent Pattern, Dasis, C) (PAMI, Juzy Ceoreferenced Prequent Pattern, Dasis, C) (PAMI, Juzy Ceoreferenced Frequent Pattern, Juzy C) (PAMI, Jugy Utility Pattern, Dasis, C) (PAMI, Juzy Ceoreferenced Frequent Pattern, Juzy C) (PAMI, Juzy Ceoreferenced Frequent Pattern, Juzy C) (PAMI, Juzy Ceoreferenced Frequent Pattern, Juzy C) (PAMI, Juzy C) (P	meinou), 24	memore), 154
method), 138 method), 138 method), 240 method), 240 method), 241 method), 243 method), 244 method), 245 method), 255 method), 255 getMemoryUSS () (PAMI.fuzzyGeoreferencedFrequentPattergekkeudfiySSSKi)(CPMMIprinielFeriodicPattern.topk.k3PMiner.k3PMine method), 253 getMemoryUSS () (PAMI.fuzzyGeoreferencedPeriodicFrequentPattergekkeudfiySSKi)(CPMMIprinielFeriodicPattern.basic.EPCPGrowth. method), 253 getMemoryUSS () (PAMI.fuzzyGeoreferencedPrequentPattern.basigetReinkdiyISSKI)(PAMIAperiodicFrequentPattern.basic.EPCPGrowth. method), 253 getMemoryUSS () (PAMI.fuzzyPeriodicFrequentPattern.basigetReinkdiyISSKI)(PAMIAperiodicFrequentPattern.basic.EPCPGrowth. method), 253 getMemoryUSS () (PAMI.georeferencedFrequentPattern.basigetReinkdiyUSSKI)(PAMIAperiodicFrequentPattern.basic.PFPGrowth.PF. method), 170 getMemoryUSS () (PAMI.georeferencedPartialPeriodicPrequentPattern.basic.PFPGrowth.PF. method), 178 getMemoryUSS () (PAMI.georeferencedPeriodicFrequentPattern.basic.FFMC.PFPM method), 178 getMemoryUSS () (PAMI.georeferencedPeriodicFrequentPattern.basic.FFMC.PFPM method), 179 getMemoryUSS () (PAMI.georeferencedPeriodicFrequentPattern.basic.EFVC.PFPM method), 179 getMemoryUSS () (PAMI.lighUtilityFrequentPattern.basic.EfVENdditfyWISK) (PAMI.periodicFrequentPattern.basic.PFPGrowth.PFG method), 179 getMemoryUSS () (PAMI.highUtilityFrequentPattern.basic.EfVENdditfyWISK) (PAMI.periodicFrequentPattern.closed.CFFPMiner.CI method), 196 getMemoryUSS () (PAMI.highUtilityPattern.basic.EFIM.EFiMtMemoryUSS () (PAMI.periodicFrequentPattern.maximal.MaxPFGrow method), 215 getMemoryUSS () (PAMI.highUtilityPattern.basic.EFIM.EFiMtMemoryUSS () (PAMI.periodicFrequentPattern.topk.TFFPMiner.KPFI method), 225 getMemoryUSS () (PAMI.highUtilityPattern.basic.IPGrowthStiftWisteryUSS () (PAMI.periodicFrequentPattern.topk.TFFPMiner.KPFI method), 225 getMemoryUSS () (PAMI.highUtilityPattern.basic.IPGrowthStiftWisteryUSS () (PAMI.periodicFrequentPattern.topk.TFFPMiner.KPFI method), 225 getMemoryUSS () (PAMI.highUtilitySpatialPattern.basic.IPGrowthStiftW	getmemoryUSS() (PAMI.juzzyCorrelatedPattern.basic.FCPgertnemor	Lybbown In 120
get Memory USS () (PAMI, fuzzy Frequent Pattern, basic, FFIM get Medi VisiS () (PAMI, partial Periodic Pattern, topk, k3PM inter, k3PM inter, method), 245 method), 329 get Memory USS () (PAMI, fuzzy Georeferenced Periodic FrequentPattergeb Meind VisiS () (PAMI, fuzind Pattern, basic, EPCPG rowth, method), 255 method), 146 get Memory USS () (PAMI, fuzzy Periodic FrequentPattern, basiget Neind VisiS () (PAMI Minial Pattern, basic, EPCPG rowth, method), 259 method), 146 get Memory USS () (PAMI, georeferenced FrequentPattern, basiget Neind VisiS () (PAMI, Might Eind Terguent Pattern, basic, PFPG rowth, PF, method), 170 method), 170 method), 163 get Memory USS () (PAMI, georeferenced Partial Periodic Patteget Memory USS () (PAMI, georeferenced Periodic Frequent Pattern, basic, STB USK () STANMI (ALL), 174 method), 174 get Memory USS () (PAMI, high Utility Frequent Pattern, basic, Life With Model VisiS () (PAMI, periodic Frequent Pattern, basic, Life With Minial VisiS () (PAMI, high Utility Frequent Pattern, basic, Life With Model VisiS () (PAMI, periodic Frequent Pattern, basic, Life With Model VisiS () (PAMI, periodic Frequent Pattern, basic, Life With Model VisiS () (PAMI, periodic Frequent Pattern, basic, Life With Model VisiS () (PAMI, periodic Frequent Pattern, basic, Life With Minial VisiS () (PAMI, periodic Frequent Pattern, basic, Life With Model VisiS () (PAMI, periodic Frequent Pattern, basic, Life With Model VisiS () (PAMI, periodic Frequent Pattern, basic, Life With Minial VisiS () (PAMI, periodic Frequent Pattern, basic, Life With Model VisiS () (PAMI, periodic Frequent Pattern, basic, Life With Minial VisiS () (PAMI, periodic Frequent Pattern,	method), 250	method), 138
method), 246 method), 141 getMemoryUSS () (PAMI,fuzzyGeoreferencedFrequentPatte gebMeindFyMSMf)defRsMf)MiniedPeriodicPatternInMultipleTimeSeries.PI method), 255 method), 329 getMemoryUSS () (PAMI,fuzzyGeoreferencedPeriodicFrequgerfMethody, 329 getMemoryUSS () (PAMI,fuzzyPeriodicFrequentPattern.basigetRefixedvaryMSS () GPH/NP,fireia/HGG/#MMiniedPattern.basic.PFECLAT.PFEv method), 263 method), 146 getMemoryUSS () (PAMI,georeferencedFrequentPattern.basigetRefixedvaryMSS () GPM/NEgeriodicFrequentPattern.basic.PFECLAT.PFEv method), 259 method), 76 getMemoryUSS () (PAMI,georeferencedFrequentPattern.basigetRefixedvaryMSS () GPM/NEgeriodicFrequentPattern.basic.PFFGrowth.PF. method), 170 method), 63 getMemoryUSS () (PAMI,georeferencedPartialPeriodicPattegetMethodyS0TMAMLperiodicFrequentPattern.basic.PFPGrowthPlus method), 178 method), 63 getMemoryUSS () (PAMI,georeferencedPeriodicFrequentPagetMethodyS0TMAMLperiodicFrequentPattern.basic.PFPGrowthPlus method), 174 method), 63 getMemoryUSS () (PAMI.georeferencedPeriodicFrequentPagetMethodyS0TMAMLpeFibilioFrequentPattern.basic.PFPMC.PFPM method), 174 method), 80 getMemoryUSS () (PAMI.highUtilityFrequentPattern.basic.JEftMethdfiy03SK () (PAMI.periodicFrequentPattern.closed.CFFPMI.PFG method), 191 getMemoryUSS () (PAMI.highUtilityGeoreferencedFrequentfettemdary0ISSK () (PAMI.periodicFrequentPattern.closed.CFFPMiner.Cr method), 196 getMemoryUSS () (PAMI.highUtilityPattern.basic.FFIM.EFfMtMemoryUSS () (PAMI.periodicFrequentPattern.topk.kPFPMiner.KPFI method), 219 getMemoryUSS () (PAMI.highUtilityPattern.basic.HMiner.HyttMetmoryUSS () (PAMI.periodicFrequentPattern.topk.kPFPMiner.KPFI method), 22 getMemoryUSS () (PAMI.highUtilityPattern.basic.HMiner.HyttMetmoryUSS () (PAMI.periodicFrequentPattern.topk.kPFPMiner.KPFI method), 22 getMemoryUSS () (PAMI.highUtilityPattern.basic.HMiner.HyttMetmoryUSS () (PAMI.periodicFrequentPattern.basic.RSFPGrowth.RPG method), 22 getMemoryUSS () (PAMI.highUtilityPattern.basic.HMiner.HyttMetmoryUSS () (PAMI.periodicFrequentPattern.basic.RSFPGrowth.RF	getMemoryUSS() (<i>PAMI.fuzzyFrequentPattern.basic.FFIMi</i> gærMemdø	figlesS() (PAMI.partialPeriodicPattern.topk.k3PMiner.k3PMiner
getMemoryUSS() (PAMI.fuzzyGeoreferencedFrequentPattergebMeindfi%MSMMA/MKNIJMiriedPeriodicPatternInMultipleTimeSeries.PI method), 255 getMemoryUSS() (PAMI.fuzzyGeoreferencedPeriodicFrequentWisterWistSG)(UMNMI.fuzionIIiGO#bibMainedPattern.basic.EPCPGrowth. method), 263 getMemoryUSS() (PAMI.fuzzyPeriodicFrequentPattern.basigeRietWistyUSSG)(PAMIlifeEtWiteFrequentPattern.basic.FFECLAT.PFE/ method), 259 getMemoryUSS() (PAMI.georeferencedFrequentPattern.basigeRietWistyUSSG)(SpRAMIlifeEtWiteFrequentPattern.basic.PFPGrowth.PF. method), 170 getMemoryUSS() (PAMI.georeferencedPartialPeriodicPattegetHistoryUSSG)(SpRAMILifeEtWiteFrequentPattern.basic.PFPGrowth.PF. method), 170 getMemoryUSS() (PAMI.georeferencedPartialPeriodicPattegetHistoryUSSG)(SpRAMILifeEtWiteFrequentPattern.basic.PFPGrowth.PFF method), 178 getMemoryUSS() (PAMI.geoReferencedPartialPeriodicPattegetHistoryUSSG)(SpRAMILifeEtWiteFrequentPattern.basic.PFPGrowth.PFG method), 178 getMemoryUSS() (PAMI.highUtilityFrequentPattern.basic.HistRMddfVyUSSU) (PAMI.periodicFrequentPattern.basic.PFPMC.PFPMC method), 191 getMemoryUSS() (PAMI.highUtilityFrequentPattern.basic.HistRMddfVyUSSU) (PAMI.periodicFrequentPattern.closed.CPFPMiner.Cl method), 191 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFIMtMemoryUSS() (PAMI.periodicFrequentPattern.closed.CPFPMiner.Cl method), 193 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFIMtMemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.KPFI method), 219 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.HightWiteFrequentYUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.KPFI method), 225 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowthgetWiteFrequentVUSS() (PAMI.highUtilityPattern.basic.UPGrowthgetWiteFrequentVUSS() (PAMI.highUtilitySpatialPattern.basic.HJGHWiteFrequentVUSS() (PAMI.highUtilitySpatialPattern.basic.HJGHWiteFrequentPattern.basic.RSFPGrowth.RS method), 204 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.HJGHWiteFrequentPattern.basic.RSFPGrowth.RS method), 204 getMemoryUSS() (PAMI.highUtilityS	<i>method</i>), 246	<i>method</i>), 141
method), 255 method), 329 getMemoryUSS() (PAMI.fuzzyGeoreferencedPeriodicFrequentMises(); (JPMM/IneriodicGebitMandPattern.basic.EPCPGrowth. method), 259 method), 146 getMemoryUSS() (PAMI.fuzzyPeriodicFrequentPattern.basigetReixMiyUSS(); (JPMM/IneriodicFrequentPattern.basic.PFECLAT.PFEV method), 259 method), 63 getMemoryUSS() (PAMI.georeferencedFrequentPattern.basiceStressionSyTUSS(); (SMIMMIGEixMicFrequentPattern.basic.PFPGrowth.PF, method), 170 method), 63 getMemoryUSS() (PAMI.georeferencedFrequentPattern.basiceStressionSyTUSS(); (SMIMMIGEixMicFrequentPattern.basic.PFPGrowth.PF, method), 170 method), 67 getMemoryUSS() (PAMI.georeferencedPeriodicFrequentPattern.basic.SFPGrowthPlus method), 174 method), 80 getMemoryUSS() (PAMI.highUtilityFrequentPattern.basic.HighWindfYVSX() (PAMI.periodicFrequentPattern.basic.PSGrowth.PSGr method), 191 getMemoryUSS() (PAMI.highUtilityGeoreferencedFrequentFequentPattern.stasic.SUMMMIfibHifiMicFrequentPattern.closed.CPFPMiner.Cl method), 191 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFiJM.MemoryUSS() (PAMI.periodicFrequentPattern.closed.CPFPMiner.Cl method), 219 method), 83 getMemoryUSS() (PAMI.highUtilityPattern.basic.HighWinemoryUSS() (PAMI.periodicFrequentPattern.aximal.MaxPFGrow method), 219 method), 87 getMemoryUSS() (PAMI.highUtilityPattern.basic.HighWinemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 222 method), 87 getMemoryUSS() (PAMI.highUtilityPattern.basic.HighWinemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 222 method), 84 getMemoryUSS() (PAMI.highUtilityPattern.basic.HighWinemoryUSS() (PAMI.periodicFrequentPattern.topk.RPFPMiner.kPFI method), 220 method), 84 getMemoryUSS() (PAMI.highUtilityPattern.basic.HighWinemoryUSS() (PAMI.periodicFrequentPattern.topk.RPFPTopkPF method), 220 method), 84 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SFIGFWineKiNgYUSS() (PAMI.periodicFrequentPattern.basic.RSFPGrowth.RS method), 200 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SFIGFWineKiNgYUSS() (PAMI	getMemoryUSS()(PAMI.fuzzyGeoreferencedFrequentPattergebMemory	TYSIBM())(PANIPMitien PeriodicPatternInMultipleTimeSeries.PPG
<pre>getMemoryUSS() (PAMI.fuzzyGeoreferencedPeriodicFrequegetMemoryUSS() (JPAMI.fuzialFibMinadPattern.basic.EPCPGrowth. method), 263 method), 146 getMemoryUSS() (PAMI.fuzzyPeriodicFrequentPattern.basigetMethod), 176 getMemoryUSS() (PAMI.georeferencedFrequentPattern.basigetMethod), 76 getMemoryUSS() (PAMI.georeferencedFrequentPattern.basigetMethod), 63 getMemoryUSS() (PAMI.georeferencedPartialPeriodicPattegetMethod), 63 getMemoryUSS() (PAMI.georeferencedPartialPeriodicPattegetMethod), 67 getMemoryUSS() (PAMI.georeferencedPartialPeriodicPattegetMethod), 67 getMemoryUSS() (PAMI.georeferencedPeriodicFrequentPattern.basic.Jft/MithigPitMiofirequentPattern.basic.PFPGrowthPlus method), 178 method), 67 getMemoryUSS() (PAMI.lighUtilityFrequentPattern.basic.Jft/MithigPitMiofirequentPattern.basic.PFPMC.PFPM method), 174 method), 80 getMemoryUSS() (PAMI.highUtilityFrequentPattern.basic.Jft/MithigNUTMMI.fift/HiMcFrequentPattern.basic.PSGrowth.PSGr method), 19 getMemoryUSS() (PAMI.highUtilityGeoreferencedFrequentGetMethody, 171 getMemoryUSS() (PAMI.highUtilityGeoreferencedFrequentGetMethody, 171 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFiftMtMemoryUSS() (PAMI.periodicFrequentPattern.closed.CPFPMiner.CL method), 19 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFiftMtMemoryUSS() (PAMI.periodicFrequentPattern.topk.KPFPMiner.KPFI method), 219 method), 87 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.HyttMemoryUSS() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 222 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.HBGHMiMfJVSSUSUXAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 204 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SFiftMethfYDSSUSUSUSUXAMI.relativeFrequentPattern.basic.RPFProphPF method), 204 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SFiftMethfYDSSUSUSUSUXAMI.relativeHighUtilityPattern.basic.RPFGrowth.RS method), 204 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.LPPGigetMethoffySSUSUSUSUAAMI.relativeHighUtilityPattern.basic.RPFGrowth.RS method), 204</pre>	<i>method</i>), 255	<i>method</i>), 329
method), 146 method), 146 getMemoryUSS (PAMI.fuzzyPeriodicFrequentPattern.basigetRefRMirfuSSG)(PAMIleperiodicFrequentPattern.basic.PFECLAT.PFE method), 259 getMemoryUSS (PAMI.georeferencedFrequentPattern.basigetRefIndirfyUSSQ)(pAMIleperiodicFrequentPattern.basic.PFPGrowth.PF, method), 170 getMemoryUSS (PAMI.georeferencedPartialPeriodicPattegetMemoryUSSQ)(pAMI.georeferencedPartialPeriodicFrequentPattern.basic.PFPGrowthPlus method), 178 getMemoryUSS (PAMI.georeferencedPartialPeriodicFrequentPattern.basic.JFPPGrowthPlus method), 178 getMemoryUSS (PAMI.georeferencedPeriodicFrequentPattern.basicJSG)MirAMI.geFibMiofibrequentPattern.basic.PFPMC.PFPM method), 178 getMemoryUSS (PAMI.highUtilityFrequentPattern.basic.JffMirMatryUSSG)(PAMI.periodicFrequentPattern.basic.PSGrowth.PSGri method), 191 getMemoryUSS (PAMI.highUtilityFrequentPattern.basic.JffMirMatryUSSG)(PAMI.periodicFrequentPattern.closed.CPFPMiner.CL method), 191 getMemoryUSS (PAMI.highUtilityGeoreferencedFrequentJettMemdaryUSSG)(PAMI.periodicFrequentPattern.closed.CPFPMiner.CL method), 196 getMemoryUSS (PAMI.highUtilityPattern.basic.EFIM.EFjMtMemoryUSS (PAMI.periodicFrequentPattern.tosk.CPFPMiner.CL method), 222 method), 91 getMemoryUSS (PAMI.highUtilityPattern.basic.HIMiner.HjMtMemoryUSS (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 225 method), 94 getMemoryUSS (PAMI.highUtilityPattern.basic.UPGrowthethTethoryUSS (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 200 getMemoryUSS (PAMI.highUtilitySpatialPattern.basic.SHjMtMtMityDSSU(JMAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 210 getMemoryUSS (PAMI.highUtilitySpatialPattern.basic.SHjMtMtMtMtSQUMANI.recurringPattern.basic.RPGrowth.RSG method), 210 getMemoryUSS (PAMI.highUtilitySpatialPattern.basic.LPPGrgettRefIndfWdMtGySUS(MAMI.relativeHighUtilityPattern.basic.RPGrowth.RSG method), 321 getMemoryUSS (PAMI.localPeriodicPattern.basic.LPPG gettRefIndfWdSUS(MAMI.relativeHighUtilityPattern.basic.SPADE.SPADE method), 321 getMemoryUSS (PAMI.localPeriod	getMemoryUSS()(PAMI.fuzzyGeoreferencedPeriodicFrequ getNermo n	astassa (F GPPANP Miewind i G Poli Pollained Pattern. basic. EPCPG row th. E.
<pre>getMemoryUSS () (PAMI.fuzzyPeriodicFrequentPattern.basigetReinkärgits:St())(PAMI/keperiodicFrequentPattern.basic.PFECLAT.PFEV method), 259 method), 76 getMemoryUSS () (PAMI.georeferencedFrequentPattern.basiesfinatiotyUSSQ)(StatAMILifeEiikiTicFrequentPattern.basic.PFPGrowth.PF, method), 170 method), 63 getMemoryUSS () (PAMI.georeferencedPartialPeriodicPattagetHismosFyTisEd(StH2AMILifeFibMitifeequentPattern.basic.PFPGrowthPlus method), 178 method), 67 getMemoryUSS () (PAMI.geoReferencedPeriodicFrequentPatternbasic.YFPMC.PFPMV method), 174 method), 80 getMemoryUSS () (PAMI.highUtilityFrequentPattern.basic.HetMitifetHiMCIPFibMitifeequentPattern.basic.PFPMC.PFPMV method), 191 method), 71 getMemoryUSS () (PAMI.highUtilityFrequentPattern.basic.HetMitifetHiMCIPFibMitifetHiMCFrequentPattern.basic.PSGrowth.PSGi method), 191 method), 83 getMemoryUSS () (PAMI.highUtilityGeoreferencedFrequentFitMemdryUSSKI(JMMILifetHiMCFrequentPattern.closed.CPFPMiner.Cl method), 196 method), 83 getMemoryUSS () (PAMI.highUtilityPattern.basic.EFIM.EFitMemoryUSS () (PAMI.periodicFrequentPattern.maximal.MaxPFGrow method), 219 method), 87 getMemoryUSS () (PAMI.highUtilityPattern.basic.HMiner.Hj#tMemoryUSS () (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 222 method), 91 getMemoryUSS () (PAMI.highUtilityPattern.basic.UPGrowthetTitMemoryUSS () (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 225 method), 94 getMemoryUSS () (PAMI.highUtilitySpatialPattern.basic.SHj#tMethofMAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 method), 163 getMemoryUSS () (PAMI.highUtilitySpatialPattern.basic.SHj#tMethofMXISUSUAMI.relativeFrequentPattern.basic.RPFGrowth.RPG method), 200 method), 28 getMemoryUSS () (PAMI.highUtilitySpatialPattern.basic.SHj#tMethofMXISUSUAMI.relativeFrequentPattern.basic.RFPGrowth.RS method), 200 method), 28 getMemoryUSS () (PAMI.highUtilitySpatialPattern.basic.LPPG/getMthafMXISUSUS(PAMI.relativeHighUtilityPattern.basic.RPFGrowth.RS method), 200 method), 321 getMemoryUSS () (PAMI.highUtilitySpatialPatte</pre>	method). 263	method). 146
method), 259 method), 76 getMemoryUSS () (PAMI.georeferencedFrequentPattern.basige 5) aniol 5) OSS () SyRAMIL fight bill for the fight of the fight	getMemorvUSS() (PAMI.fuzzyPeriodicFrequentPattern.basigetPeriod	tinuss(PRPANNeperiodicFrequentPattern.basic.PFECLAT.PFECL
<pre>getMemoryUSS() (PAMI.georeferencedFrequentPattern.basge 6fianitPUESQ() (PAMI.georeferencedFrequentPattern.basic.PFPGrowth.PF. method), 170 method), 63 getMemoryUSS() (PAMI.georeferencedPartialPeriodicPattagetMetriosfyUSK()/URAMILperiodicFrequentPattern.basic.PFPGrowthPlus method), 178 getMemoryUSS() (PAMI.geoReferencedPeriodicFrequentPattern.basic;USY()/URAMILpeFibMibiBrequentPattern.basic.PFPMC.PFPM(method), 174 method), 80 getMemoryUSS() (PAMI.highUtilityFrequentPattern.basic.IfetMibidifyUSX() (PAMI.periodicFrequentPattern.basic.PSGrowth.PSGr method), 191 getMemoryUSS() (PAMI.highUtilityGeoreferencedFrequentFattern.basic.IfetMibidifyUSX() (PAMI.periodicFrequentPattern.closed.CPFPMiner.CF method), 196 method), 83 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFigMtMemoryUSS() (PAMI.periodicFrequentPattern.closed.CPFPMiner.CF method), 219 method), 87 getMemoryUSS() (PAMI.highUtilityPattern.basic.Himer.HightMemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 222 method), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.Himer.HightMemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 225 method), 94 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowthetTeinsonyUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 220 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.HIGHMtHTDSXUMAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHjetMethifyUSS() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHjetMethifyUSS() (PAMI.relativeHighUtilityPattern.basic.RFIGrowth.RSG method), 204 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrogetHighMtFWGS65() (PAMI.sequentialPatternMining.basic.sPADE.SPADE method), 201 method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBetHighWithWfSS() MithalLsequentialPatternMining.basic.SPADE.SPADE method), 321</pre>	method). 259	method). 76
(getMemoryUSS () (FMM:georeferencedPartialPeriodicPattagetMasinoSYUSS (a)S(HAMM: periodicFrequentPattern.basic.PFPGrowthPlus method), 178 method), 67 (getMemoryUSS () (PAMI.geoReferencedPeriodicFrequentPattern.basic(YUSY(D)MAMLfiPFiPMibfiPequentPattern.basic.PFPMC.PFPM(method), 174 method), 80 (getMemoryUSS () (PAMI.highUtilityFrequentPattern.basic.HightMathfiPefiPMibfiPequentPattern.basic.PFPMC.PFPM(method), 191 method), 80 getMemoryUSS () (PAMI.highUtilityFrequentPattern.basic.HightMathfiPefiPMibfiPequentPattern.basic.PSGrowth.PSGr method), 191 method), 71 getMemoryUSS () (PAMI.highUtilityGeoreferencedFrequentFattern.basic.SEG(VMM.SEG) (PAMI.periodicFrequentPattern.closed.CPFPMiner.CL method), 196 method), 83 getMemoryUSS () (PAMI.highUtilityPattern.basic.EFIM.EFifMetmoryUSS () (PAMI.periodicFrequentPattern.maximal.MaxPFGrow method), 219 method), 87 getMemoryUSS () (PAMI.highUtilityPattern.basic.HIMiner.HythMemoryUSS () (PAMI.periodicFrequentPattern.topk.KPFPMiner.kPFI method), 222 method), 91 getMemoryUSS () (PAMI.highUtilityPattern.basic.UPGrowthethTethody, 91 getMemoryUSS () (PAMI.highUtilityPattern.basic.UPGrowthethTethonyUSS () (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 200 getMemoryUSS () (PAMI.highUtilitySpatialPattern.basic.SHightMethfiPUSS () (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 200 getMemoryUSS () (PAMI.highUtilitySpatialPattern.basic.SHightMethfiPUSS () (PAMI.relativeFrequentPattern.basic.RPGrowth.RPGrowth method), 200 getMemoryUSS () (PAMI.highUtilitySpatialPattern.basic.SHightMethfiPUSS () (PAMI.relativeFrequentPattern.basic.RFGrowth.RPGrowth method), 200 getMemoryUSS () (PAMI.highUtilitySpatialPattern.basic.SHightMethfiPUSS () (PAMI.relativeFrequentPattern.basic.RFFGrowth.RFG method), 200 getMemoryUSS () (PAMI.highUtilitySpatialPattern.basic.LFPGFgethfithQFGWSG () (PAMI.relativeHighUtilityPattern.basic.RFFGrowth.RFG method), 210 method), 163 getMemor	aetMemoryUSS() (PAMI georeferencedFrequentPattern basie fMenia	KMRSMMBAMIGGeiaTicFrequentPattern basic PFPGrowth PFPG
<pre>method, 970 method, 178 method), 178 getMemoryUSS() (PAMI.georeferencedPartialPeriodicPattegetMemoryUSS()(PAMI.geoReferencedPeriodicFrequentPattern.basic.PFPGrowthPlus method), 178 getMemoryUSS() (PAMI.geoReferencedPeriodicFrequentPattern.basic,USFQ/WAMLfiteFibMiof6requentPattern.basic.PFPMC.PFPM method), 174 getMemoryUSS() (PAMI.highUtilityFrequentPattern.basic.it/it/Wisk() (PAMI.periodicFrequentPattern.basic.PSGrowth.PSGr method), 191 getMemoryUSS() (PAMI.highUtilityGeoreferencedFrequentFieltMemdaryUSS() (PAMI.periodicFrequentPattern.closed.CPFPMiner.Cr method), 190 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFiltMemoryUSS() (PAMI.periodicFrequentPattern.closed.CPFPMiner.Cr method), 219 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFiltMemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 219 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.HighUtilemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 222 method), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowthgetHeinoryUSS() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 225 method), 24 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHighHeimHift)DSS() (PAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHighHeimHift)DSS() (PAMI.relativeFrequentPattern.basic.RFFGrowth.RPGrowth method), 200 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHighHeimHift)DSS() (PAMI.relativeFrequentPattern.basic.RFFGrowth.RFG method), 200 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHighHeimHift)DSS() (PAMI.relativeFrequentPattern.basic.RFFGrowth.RFG method), 210 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.LPPGgetHiftMeSHift)DSS() (PAMI.sequentialPattern.basic.RFFGrowth.RFUIM.RFUI method), 210 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPG getHiftMeSHift)DSS() (PAMI.sequentialPatternMining,basic.SPADE.SPADE method), 321</pre>	method) 170	method) 63
<pre>getMemoryUSS() (PAMI.georgerencedPeriodicFrequentPattern.basic.JjetAlskuSyJSuZimperiodicFrequentPattern.basic.FFPMC.PFPM(method), 178 method), 67 getMemoryUSS() (PAMI.highUtilityFrequentPattern.basic.JjetAlskuSyJSUZIMPAAL[jPFiDMipFidequentPattern.basic.PSGrowth.PSGn method), 191 method), 71 getMemoryUSS() (PAMI.highUtilityGeoreferencedFrequentJettitemdoryUSS() (PAMI.periodicFrequentPattern.closed.CPFPMiner.CF method), 196 method), 83 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFijktMemoryUSS() (PAMI.periodicFrequentPattern.maximal.MaxPFGrow method), 219 method), 83 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.HjetMemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 222 method), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowthjetMemoryUSS() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 225 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.HJjetHideMitJUSSU()MAMI.recurringPattern.basic.RFGrowth.RFG method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHjetHeiMitJUSSU()MAMI.recurringPattern.basic.RFFGrowth.RS method), 210 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHjetHeiMitJUSSU()MAMI.relativeFrequentPattern.basic.RFFGrowth.RS method), 210 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHjetHeiMitJUSSU()MAMI.relativeFrequentPattern.basic.RFFGrowth.RS method), 210 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.LPPGrgestHeiMitJWSS() (PAMI.relativeHighUtilityPattern.basic.RFFGrowth.RF method), 210 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.LPPGrgestHeiMithfYSSS() (PAMI.sequentialPatternMining.basic.SPADE.SPADE method), 9 method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPM BectHindfryWSSG) (PAMI.sequentialPatternMining,basic.SPADE.SPADE method), 321</pre>	aetMomoryUISS() (PAMI appreferencedPartialPariodicPattaretMomo	STAS (d'SUFAM) pariodic Fraquent Pattern basic PEPC rowth Plus E
<pre>method, 07 getMemoryUSS() (PAMI.geoReferencedPeriodicFrequentPqgetMemoryUSS() (PAMI.geoReferencedPeriodicFrequentPqtetrn.basic.PFPMC.PFPMC method), 174 getMemoryUSS() (PAMI.highUtilityFrequentPattern.basic.LfdtMedfJyUSM() (PAMI.periodicFrequentPattern.basic.PSGrowth.PSGr method), 191 getMemoryUSS() (PAMI.highUtilityGeoreferencedFrequentfetNemdxyUSS(J)(JPMMShfbffdMcFrequentPattern.closed.CPFPMiner.CL method), 196 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFfjktMemoryUSS() (PAMI.periodicFrequentPattern.maximal.MaxPFGrow method), 219 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.HjbftMemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 22 method), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.HjbftMemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 22 method), 91 getMemoryUSS() (PAMI.highUtilitySpattern.basic.UPGrowthgetTfCtnoxyUSS() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 225 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.Hlj6HtMetMitjUSSKU(MAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 method), 200 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHj6HtMetMitjUSSKU(MPAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 204 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKfjHUMetMitjKSFSKUMPAMI.relativeFrequentPattern.basic.RFPGrowth.RS method), 210 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKfjHUMetMitjKSFSKUMPAMI.relativeFrequentPattern.basic.RFUIM.RHUI method), 186 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKfjHUMetMitjKSFSKUMPAMI.relativeHighUtilityPattern.basic.RFUIM.RHUI method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgetMthdfSKSFSKUMPAMI.sequentialPatternMining.basic.SPADE.SPADE method), 99 method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBgetMthdfSKSKSKUMI.sequentialPatternMining.basic.SPADE.SPADE method), 312</pre>	method) 179	method) 67
<pre>getMemoryUSS() (PAMI.kighUtilityPattern.basic.FFIM.erFIM.complexenterlygerMathingSt(), WAMALfierMothonequentPattern.basic.PFFM.c.FFFM.complexenterlygerMathingSt(), WAMI.periodicFrequentPattern.basic.PSGrowth.PSGreathermoryUSS() (PAMI.highUtilityFrequentPattern.basic.EFIM.efiftMethod), 71 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.efiftMethod), 83 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.efiftMethod), 87 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.HytriMemoryUSS() (PAMI.periodicFrequentPattern.maximal.MaxPFGrowmethod), 219 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.HytriMemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFImethod), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowthgetiffChomyUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFImethod), 222 method), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowthgetiffChomyUSS() (PAMI.periodicFrequentPattern.topk.kPFP.TopkPF method), 225 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.HightHethodityDSSU()WAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHightHethodityDSSU()WAMI.recurringPattern.basic.RSFPGrowth.RS method), 204 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKightWethodityDSSU()WAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 210 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKightWethodityDSSU()WAMI.relativeHighUtilityPattern.basic.RFFIGROWth.RHUI method), 210 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrowtHefWEthOfWSG() (PAMI.sequentialPatternMining.basic.SPADE.SPADE method), 93 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrowtHefWEthodityDSS() (PAMI.sequentialPatternMining.basic.SPADE.SPADE method), 221 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrowtHefWEthOfWSG() (PAMI.sequentialPatternMining.basic.SPADE.SPADE method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrowtHefWethOfWSG() (PAMI.sequentialPatternMining.basic.SPADE.SPADE method), 312 m</pre>	memou), 170	memou), 07
method), 80 getMemoryUSS () (PAMI.highUtilityFrequentPattern.basic. Hj&HMedifyWSM () (PAMI.periodicFrequentPattern.basic.PSGrowth.PSGr method), 191 getMemoryUSS () (PAMI.highUtilityGeoreferencedFrequent fextivendaxy USS UJ/PMMSHj#iFilM cFrequentPattern.closed.CPFPMiner.CI method), 196 getMemoryUSS () (PAMI.highUtilityPattern.basic.EFIM.EF j# MemoryUSS () (PAMI.periodicFrequentPattern.maximal.MaxPFGrow method), 219 getMemoryUSS () (PAMI.highUtilityPattern.basic.HMiner.H yWikemory USS () (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 222 method), 91 getMemoryUSS () (PAMI.highUtilityPattern.basic.UPGrowt j et WEinory USS () (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 225 getMemoryUSS () (PAMI.highUtilitySpatialPattern.basic.UPGrowt j et WEinory USS () (PAMI.periodicFrequentPattern.basic.RPGrowth.RPGrowth method), 200 getMemoryUSS () (PAMI.highUtilitySpatialPattern.basic.SH j et HeinfHjWjMjUiWiMiUiUiWiMiUUiUUUUUUUUUUUUU	getMemoryUSS() (PAMI.geoReferenceaPeriodicFrequentPagernemos	
getMemoryUSS() (PAMI.highUtilityFrequentPattern.basic. HeimidifyWSM () (PAMI.periodicFrequentPattern.basic.PSGrowth.PSGr method), 191 method), 71 getMemoryUSS() (PAMI.highUtilityGeoreferencedFrequent GetMemoryUSS() (PAMI.periodicFrequentPattern.closed.CPFPMiner.CF method), 83 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EF jht MemoryUSS() (PAMI.periodicFrequentPattern.maximal.MaxPFGrow method), 219 method), 87 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner. HydiMem oryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 222 method), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowt jet/Fexion syUSS() (PAMI.periodicFrequentPattern.topk.topkPFP.TopkPF method), 225 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.H ightHeimHighUSS() (PAMI.relativeFrequentPattern.basic.RPGrowth method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SH ightMeimHighUSS() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TK §HUHM:HighUtilityExs () (PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TK §HUHM:HighUtilityExs () (PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 28 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGr gestHeimPGFWSG () (PAMI.sequentialPatternMining.basic.prefixSpan.prefix method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPM BexHimMirPMSM ; 4 4 4 4 4 4 4 4 4 4	method), 1/4	method), 80
method), 191 method), 71 getMemoryUSS() (PAMI.highUtilityGeoreferencedFrequentfettMendersyUSSKI/UPIMMStheifelMcFrequentPattern.closed.CPFPMiner.Cli method), 196 method), 83 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFijkAMemoryUSS() (PAMI.periodicFrequentPattern.maximal.MaxPFGrow method), 219 method), 87 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.HighThemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 222 method), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowthgetTReinonsyUSS() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 225 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.HightHightHemitHighUsS() (PAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHightHightHightS() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 200 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKjHUMethoftKSTSU(MAMI.relativeHighUtilityPattern.basic.RFPGrowth.RS method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKjHUMethoftKSTSU(MAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrogentHightQFGBSG() (PAMI.sequentialPatternMining.basic.SPADE.SPADE method), 99 method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPM getMithdffgGBSG) 412	getMemoryUSS() (PAMI.highUtilityFrequentPattern.basic. getMemo	tybssk() (PAMI.periodicFrequentPattern.basic.PSGrowth.PSGrov
<pre>getMemoryUSS() (PAMI.highUtilityGeoreferencedFrequentFetIvemdxyElSkfU(PFM/SlfEiFild/CFrequentPattern.closed.CPFPMiner.CL method), 196 method), 83 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFild/CMemoryUSS() (PAMI.periodicFrequentPattern.maximal.MaxPFGrow method), 219 method), 87 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.HyeliMemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 222 method), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowthelt/FeinomyUSS() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 225 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.H)getMethod hyDSStQUMAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHjelfMethilfyDSStQUMAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKjelfUHethilfyDfStQUMAMI.relativeHighUtilityPattern.basic.RFPGrowth.RS method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKjelfUHethilfyDfStQUMPAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgeetHiethOftyDStG) (PAMI.sequentialPatternMining.basic.prefixSpan.prefix method), 99 method), 312</pre>	<i>method</i>), 191	<i>method</i>), 71
method), 196 method), 83 getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFightMemoryUSS() (PAMI.periodicFrequentPattern.maximal.MaxPFGrow method), 219 method), 87 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.HjefiMemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 222 method), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowthetiPfEmonyUSS() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 225 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.HJGEHIEIMAryUSS(UMAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHJEHIEIMATJUSS(UMAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKGEHUTEIMENTJUSS(UMAMI.relativeHighUtilityPattern.basic.RFPGrowth.RS method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKGEHUTEIMENTJUSS(UMAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgeetHetiDFGrgeetHe	getMemoryUSS() (<i>PAMI.highUtilityGeoreferencedFrequent</i> GentMemda	xxxxSKLXPAMSHeFAMCFrequentPattern.closed.CPFPMiner.CPF
getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EF jMt MemoryUSS() (PAMI.periodicFrequentPattern.maximal.MaxPFGrow method), 219 method), 87 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.H jVtiMe moryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 222 method), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowt bet/Texnony USS() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 225 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.H J6HteiMityDSSL() MAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SH J6HteiMityUSS () (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TK 5HUVedtGHySIS ()(MPAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPG rowthMethol , 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPG rowtHeimCHySIS ()(PAMI.sequentialPatternMining.basic.SPADE.SPADE method), 312	<i>method</i>), 196	method), 83
method), 219 method), 87 getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.HydriMemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFN method), 222 method), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowthetTremonyUSS() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 225 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.HPGHteiMidyDSSLUNMAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHgEtMetMidyDSSLONMAMI.reclativeFrequentPattern.basic.RSFPGrowth.RS method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHgEtMetMidyDSSLONMAMI.reclativeFrequentPattern.basic.RSFPGrowth.RS method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TK§HUYetAcHighSSSUM(PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgetHetMidfighUtilityGitsSUM (PAMI.sequentialPatternMining.basic.prefixSpan.prefix method), 99 method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBetAtHetMidfighUtilSE)(#MMI.sequentialPatternMining.basic.SPADE.SPADE method), 312 method), 312	getMemoryUSS() (PAMI.highUtilityPattern.basic.EFIM.EFg&tMemo:	ryUSS() (PAMI.periodicFrequentPattern.maximal.MaxPFGrowth
<pre>getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.HyeriMemoryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFI method), 222 method), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowthetITreinonyIiSS() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 225 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.HIGEHMEMILYDSSLOJAMMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHGEIMEMILYDSS() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKSEHUMAttifySTSS()/(PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgestHeinEPGWSSG) (PAMI.sequentialPatternMining.basic.prefixSpan.prefix method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBetXHeinGrfySUSG) (HAMI.sequentialPatternMining.basic.SPADE.SPADE method), 312</pre>	<i>method</i>), 219	method), 87
<pre>method), 222 method), 91 getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowthetIRGnonyUSS() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 225 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.HIGEHIEIMIEIMUSS() (PAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHJEEIMEEIMEEIMEEIMEEIMEEIMEEIMEEIMEEIMEEI</pre>	getMemoryUSS() (PAMI.highUtilityPattern.basic.HMiner.Hydimemo:	ryUSS() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFPN
<pre>getMemoryUSS() (PAMI.highUtilityPattern.basic.UPGrowthetThemoryUSS() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPF method), 225 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.HIGhtMethdfyDShtQtMAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHdetMethdfyDShtQtMAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKgHtYHAtGtKySFS(QM(PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgettHethoft@USSG) (PAMI.sequentialPatternMining.basic.prefixSpan.prefi method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBetaMthdfyBASSG) (MEtAMI.sequentialPatternMining.basic.SPADE.SPADE method), 312</pre>	method), 222	method). 91
<pre>method), 225 method), 94 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.HIghHeiMityDSSLOUMAMI.recurringPattern.basic.RPGrowth.RPGrowth</pre>	aetMemoryUSS() (PAMI high Utility Pattern basic UPGrowthet Prenow	whiss() (PAMI periodic Frequent Pattern tonk TonkPFP TonkPFP
<pre>getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.HighHeiMityDSst()(MAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHightMemityJWSS() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKghtIMemithySTSS(M/PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgeetMemIPGWSS()) (PAMI.sequentialPatternMining.basic.prefixSpan.prefix method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBetAMIthdIfySUSS()e(MAMI.sequentialPatternMining.basic.SPADE.SPADE method), 312</pre>	method) 225	method) 94
<pre>getMemoryUSS() (PAMI.highOntitySpatialPattern.basic.HighItendifyDSbQQMPAMI.recurringPattern.basic.RFPGrowth.RFC method), 200 method), 163 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHighItendifySHSQ() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RSC method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TK§HtYHethCHKSHSV(M(PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgeetthetuPGVBSS() (PAMI.sequentialPatternMining.basic.prefixSpan.prefix method), 99 method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBeetAlthulrgVBSS()edtHaMI.sequentialPatternMining.basic.SPADE.SPADE method), 312</pre>	actMomoryUISS() (PAMI high Utility Spatial Pattern basic HOSEMANN	HINSTON MAN MI requiring Dattern basic PDC rowth PDC rowth
method), 200 method), 165 getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SHjetMeshitzjWSS() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS. method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TKSjHUMeshiftySFS/(M(PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgeetthemProvise(f) (PAMI.sequentialPatternMining.basic.prefixSpan.prefix method), 99 method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBeetAlthodrySSS() (MHAMI.sequentialPatternMining.basic.SPADE.SPADE method), 321 method), 312	gethemoty 055() (TAMI.nighotiniyspanan anen.basic.11 getremo .	with a b 162
<pre>getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SFgetMemoryUSS() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 204 method), 28 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TK§HUMedGHySTSS(M(PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgeetMetholPGV86(f) (PAMI.sequentialPatternMining.basic.prefixSpan.prefix method), 99 method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBeetMithdlPgV85(f) (PAMI.sequentialPatternMining.basic.SPADE.SPADE method), 312</pre>	methoa), 200	methoa), 105
method), 204 getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TK §HUHA d iKKSTSV() (PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgeetMetholPGV8&(f) (PAMI.sequentialPatternMining.basic.prefixSpan.prefix method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBeetMetholPgV8&G) (PAMI.sequentialPatternMining.basic.SPADE.SPADE method), 312	getMemoryUSS() (PAMI.highUtilitySpatialPattern.basic.SFgetMemo	Cyux S() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RSFF
<pre>getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TK§HYMAdifySTS(M(PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUI method), 210 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgewtMethOPGV86(f) (PAMI.sequentialPatternMining.basic.prefixSpan.prefix method), 99 method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBeadMithdrfySSSG)@diffaMI.sequentialPatternMining.basic.SPADE.SPADE method) 104 method) 312</pre>	<i>method</i>), 204	method), 28
method), 210 method), 186 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgreatHendPGUSS() (PAMI.sequentialPatternMining.basic.prefixSpan.prefix method), 99 method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBeatHendrHyUSSG) (AltHendI.sequentialPatternMining.basic.SPADE.SPADE method) 104 method) 312	getMemoryUSS() (PAMI.highUtilitySpatialPattern.topk.TK §HIWeh d	KASIS SAM (PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUIM
<pre>getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPGrgreatMetuRPGWSS() (PAMI.sequentialPatternMining.basic.prefixSpan.prefix method), 99 method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBetAtHendrffWSSG)@dHAMI.sequentialPatternMining.basic.SPADE.SPADE method) 104 method) 312</pre>	<i>method</i>), 210	<i>method</i>), 186
method), 99 method), 321 getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBeddithdufPtMSSG)edithaMI.sequentialPatternMining.basic.SPADE.SPADE method) 104 method) 312	getMemoryUSS()(PAMI.localPeriodicPattern.basic.LPPGrgeentHeinfol	ዮፍሁሪፍ(f) (PAMI.sequentialPatternMining.basic.prefixSpan.prefixS
getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBeadilthdufyEVSE) @(HAMI.sequentialPatternMining.basic.SPADE.SPADE method) 104 method) 312	method), 99	<i>method</i>), 321
method) 104 method) 312	getMemoryUSS() (PAMI.localPeriodicPattern.basic.LPPMBeaMand	RyBSBCredHAMI.sequentialPatternMining.basic.SPADE.SPADE
	method), 104	method), 312

getMemoryUSS() (PAMI.sequentialPatternMining.basic.SPgetBatt	frns() (PAMI.frequentPattern.basic.FPGrowth.FPGrowth
<i>method</i>), 318	method), 15
getMemoryUSS() (PAMI.stablePeriodicFrequentPattern.basgetSPattern.basgetSPattern.basgetSpattern.basg	ehmsPPEAMI.frequentPattern.closed.CHARM.CHARM
<i>method</i>), 155	method), 18
getMemoryUSS() (PAMI.stablePeriodicFrequentPattern.ba.getSPatte	econsh(SRPAMIythquentPattern.maximal.MaxFPGrowth.MaxFPG
<i>method</i>), 151	method), 21
getMemorvUSS() (PAMI.stablePeriodicFrequentPattern.tor	ETSISIN (PAMI.frequentPattern.topk.FAE.FAE
method), 159	method), 24
aet Memory USS () (PAMI uncertain Frequent Pattern hasic CherpErter	active of the second
method) 270	method) 250
act Momonuliss () (DAML uncertain Enguent Dattorn basic DAHEO att	memou), 250
gethemory USS() (PAMI.uncertainFrequentPattern.basic.PgetuPaul	Here in the second s
method), 214	method), 246
getMemoryUSS() (PAMI.uncertainFrequentPattern.basic.TiggetPAtt	Herns() (PAMI.fuzzyGeoreferencedFrequentPattern.basic.FFSPMi
method), 282	<i>method</i>), 255
getMemoryUSS() (PAMI.uncertainFrequentPattern.basic.TugetPEnth	cfrns() (PAMI.fuzzyGeoreferencedPeriodicFrequentPattern.basic.
<i>method</i>), 286	<i>method</i>), 263
getMemoryUSS() (PAMI.uncertainFrequentPattern.basic.TbetPath	Farns() (PAMI.fuzzyPeriodicFrequentPattern.basic.FPFPMiner.F.
<i>method</i>), 278	<i>method</i>), 259
getMemoryUSS() (PAMI.uncertainFrequentPattern.basic.UperRauth	ht Fisio (PAMI.georeferencedFrequentPattern.basic.SpatialECLAT
method), 290	method), 170
aet Memory USS () (PAMI uncertain Frequent Pattern basic UNATIPAT	Att Schut PAMI georeferenced Partial Periodic Pattern basic STEclar
method) 293	method) 178
act Momony USE () (DAMI uncentain Cooreforence dEneruent Brathenth	memory, 170
gethemory 055() (FAMI.uncentumGeorejerenceurrequentrgener H.Do	BLAIST FORMUTEROTE FORMAGE FOR OUT FOR THE STATE ST
method), 307	method), 1/4
getMemoryUSS() (PAMI.uncertainPeriodicFrequentPatterngetstatt	errestoward Philocountry Frequent Pattern. basic. HUFIM. HUFIM
<i>method</i>), 297	<i>method</i>), 191
getMemoryUSS() (PAMI.uncertainPeriodicFrequentPatterngetPatt	EFREN d (# AMII.InlgPEPIGityGuthPdjer encedFrequentPattern.basic.SH
<i>method</i>), 302	<i>method</i>), 196
getMemoryUSS() (PAMI.weightedFrequentNeighbourhood Better ath	æxin SWKPGMMhtlgBW#IPGPottehn.basic.EFIM.EFIM
<i>method</i>), 238	<i>method</i>), 219
getMemoryUSS() (PAMI.weightedFrequentPattern.basic.WEMPWH	Marns() (PAMI.highUtilityPattern.basic.HMiner.HMiner
method), 229	method), 222
getMemoryUSS() (PAMI.weightedFrequentRegularPattern.betPAVE	Refusion (WHWRIMigner/tilityPattern.basic.UPGrowth.UPGrowth
method) 234	method) 225
aetPatterns() (PAMI correlated Pattern basic CoMine CaldtPatt	erns () (PAMI highUtilitySpatialPattern basic HDSHUIM HDSH
method) 30	method) 200
act Datt orne () (DAMI correlated Dattorn basic CoMine Direct DMit	oPha () (DAMI high Utility Spatial Dattern basic SHUM SHUM
getPatterns() (FAMI.contenterfulerti.dasic.com/iner/ugetatut	erians () (FAMI.nighouniyspanaranern.basic.shoum.shou
method), 42	metnoa), 204
getPatterns() (PAMI.coveragePattern.basic.CMine.CMingetPatt	erns() (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHUI
method), 53	<i>method</i>), 210
getPatterns() (PAMI.coveragePattern.basic.CPPG.CPPGetPatt	erns() (PAMI.localPeriodicPattern.basic.LPPGrowth.LPPGrowt
method), 56	method), 99
getPatterns() (PAMI.faultTolerantFrequentPattern.basic.getApria	reithEApr(BAMI.localPeriodicPattern.basic.LPPMBreadth.LPPMB
method), 45	1 1 104
getPatterns() (PAMI.faultTolerantFrequentPattern.basic.getPAGet	method), 104
	methoa), 104 wwwsffffffffffffffffffffffffffffffffff
method), 49	methoa), 104 wrths ET (HPCMbWthalPeriodicPattern.basic.LPPMDepth.LPPMDep method), 108
method), 49 getPatterns() (PAMI frequentPattern basic Apriori Apriori Apriori Patt	methoa), 104 wrths fy (HACMbWthalPeriodicPattern.basic.LPPMDepth.LPPMDep method), 108 erns() (PAMI multipleMinimumSupportBasedFrequentPattern ba
method), 49 getPatterns() (PAMI.frequentPattern.basic.Apriori.ApriogétPatt method) 4	method), 104 certis ET (HACMbWth alPeriodicPattern.basic.LPPMDepth.LPPMDep method), 108 cerns () (PAMI.multipleMinimumSupportBasedFrequentPattern.ba method), 32
method), 49 getPatterns() (PAMI.frequentPattern.basic.Apriori.ApriogetPatt method), 4 getPatterns() (PAMI.frequentPattern basic ECLATECLATECLATECT	method), 104 Sewils FJ (HAMbAth alPeriodicPattern.basic.LPPMDepth.LPPMDep method), 108 erns () (PAMI.multipleMinimumSupportBasedFrequentPattern.ba method), 32 orns () (PAMI multipleMinimumSupportBasedFrequentPattern ba
<pre>method), 49 getPatterns() (PAMI.frequentPattern.basic.Apriori.ApriogetPatt</pre>	method), 104 courts ET (HAMbAth alPeriodicPattern.basic.LPPMDepth.LPPMDep method), 108 erns () (PAMI.multipleMinimumSupportBasedFrequentPattern.ba method), 32 erns () (PAMI.multipleMinimumSupportBasedFrequentPattern.ba method), 26
<pre>method), 49 getPatterns() (PAMI.frequentPattern.basic.Apriori.ApriogetPatt</pre>	method), 104 cerris ET (HACMAWtha lPeriodicPattern.basic.LPPMDepth.LPPMDep method), 108 erns () (PAMI.multipleMinimumSupportBasedFrequentPattern.ba method), 32 erns () (PAMI.multipleMinimumSupportBasedFrequentPattern.ba method), 36
<pre>method), 49 getPatterns() (PAMI.frequentPattern.basic.Apriori.ApriogetPatt</pre>	method), 104 wrths ET (HPCMbWtbalPeriodicPattern.basic.LPPMDepth.LPPMDep method), 108 erns() (PAMI.multipleMinimumSupportBasedFrequentPattern.ba method), 32 erns() (PAMI.multipleMinimumSupportBasedFrequentPattern.ba method), 36 wrths() (PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth.
<pre>method), 49 getPatterns() (PAMI.frequentPattern.basic.Apriori.ApriogetPatt</pre>	method), 104 certis ET (HACMbWtb alPeriodicPattern.basic.LPPMDepth.LPPMDep method), 108 erns () (PAMI.multipleMinimumSupportBasedFrequentPattern.ba method), 32 erns () (PAMI.multipleMinimumSupportBasedFrequentPattern.ba method), 36 certers () (PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth.org) method), 112
<pre>method), 49 getPatterns() (PAMI.frequentPattern.basic.Apriori.ApriogetPatt</pre>	method), 104 certis EJ (HACMisktba lPeriodicPattern.basic.LPPMDepth.LPPMDep method), 108 erns () (PAMI.multipleMinimumSupportBasedFrequentPattern.ba method), 32 erns () (PAMI.multipleMinimumSupportBasedFrequentPattern.ba method), 36 certsets () (PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth.org) method), 112 Epiffse() (PAMI.partialPeriodicFrequentPattern.basic.PPF_DFS.P.

getPatterns() (PAMI.partialPeriodicPattern.basic.GThregetPranthinEl())e(PAMI.wuthcertainFrequentPattern.basic.PUFGrowth.PUI	F
meinou), 151 meinou), 274	
getPatterns() (PAMI.partialPerioaicPattern.basic.PPP_getPatternsgQLRAMI.uncertainFrequentPattern.basic.TubeP.TUFP	
melmou), 12/ melmou), 282	
getracterins() (PAMI.parialPerioalcrallern.basic.PPP GgewirderenbeswyllerAMI.uncertainFrequentPatiern.basic.Tubes.Tubes	
method), 125 method), 280	
getPatterns() (PAMI.parnalPerioalcPattern.closea.PPP (gesearcecnsse) (PAMI.uncertainFrequentPattern.basic.IUFP.IUFP	
method), 134 method), 278	
getPatterns() (PAMI.partialPeriodicPattern.maximal.MagetRarotenneux(PAMDwincertainFrequentPattern.basic.UFGrowth.UFG	rc
method), 138 method), 290	,
getPatterns() (PAMI.partialPeriodicPattern.topk.k3PMugetReatwienzns() (PAMI.uncertainFrequentPattern.basic.UVECLAT.UVEC	:le
method), 142 method), 293	-
getPatterns() (PAMI.partialPeriodicPatternInMultipleTigesPeretAffiskow(InAMAGimmethainGeoreferencedFrequentPattern.basic.GI	FI
method), 329 method), 307	
getPatterns() (PAMI.periodicCorrelatedPattern.basic.ERGERPathernAFGRPAthIncertainPeriodicFrequentPattern.basic.UPFPG	rc
<i>method</i>), 146 <i>method</i>), 298	
getPatterns() (PAMI.periodicFrequentPattern.basic.PFEgetPEFGEAT(PAMI.uncertainPeriodicFrequentPattern.basic.UPFPG	ro
method), 76 method), 302	
$\verb+getPatterns() (PAMI. periodic Frequent Pattern. basic. PFP \verb+getPatternSt0 + PAMI. weighted Frequent Neighbourhood Pattern. basic. SV + PAMI. weighted Fre$	W
<i>method</i>), 63 <i>method</i>), 238	
getPatterns() (PAMI.periodicFrequentPattern.basic.PFPGerRathPermBFP(PAMIhPeightedFrequentPattern.basic.WFIM.WFIM	
method), 67 method), 229	
getPatterns() (PAMI. periodic Frequent Pattern. basic. PFP getPattern() (PAMI. weighted Frequent Regular Pattern. basic. WFRIMinistry () (PAMI. weighted Frequent Regular Pattern. basic. WFRIMinistry () () () () () () () () () () () () ()	ıe
<i>method</i>), 80 <i>method</i>), 234	
getPatterns()(<i>PAMI.periodicFrequentPattern.basic.PSG</i> getPlatSGmsAtbDataFrame()	
method), 71 (PAMI.correlatedPattern.basic.CoMine.CoMine	
getPatterns() (PAMI.periodicFrequentPattern.closed.CPFPMiner.GPCfhBM)inter	
<pre>method), 84 getPatternsAsDataFrame()</pre>	
getPatterns() (PAMI.periodicFrequentPattern.maximal.MaxPFGr@ PAtMLaxpFeCinedPla ttern.basic.CoMinePlus.CoMinePlus	
method), 87 method), 42	
getPatterns() (PAMI.periodicFrequentPattern.topk.kPFRMtrRatkternMasDattaFPAtime()	
method), 91 (PAMI.coveragePattern.basic.CMine.CMine	
getPatterns() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPrefiled)kEEPGrowth	
method), 94 getPatternsAsDataFrame()	
getPatterns() (PAMI.recurringPattern.basic.RPGrowth.RPGrowth(PAMI.coveragePattern.basic.CPPG.CPPG	
<i>method</i>), 163 <i>method</i>), 56	
getPatterns() (PAMI.relativeFrequentPattern.basic.RSFRGtPartRSFRStSparthFrame()	
method), 28 (PAML fault Tolerant Frequent Pattern, basic, FTA priori, FTA priori	
getPatterns() (PAML relative High Utility Pattern, basic, RHUIM, RHUMA hod), 46	
method), 186 getPatternsAsDataFrame()	
get Patterns() (PAMI sequential Pattern Mining basic prefix Span p	\overline{a}
method) 321 method) 49	
aetPatterns() (PAMI sequential Pattern Mining basic SPADER RPH/DErnsAsDataFrame()	
method) 313 (PAMI frequentPattern basic Apriori Apriori	
getPatterns() (PAMI sequentialPatternMining basic SPAM SPAM method) 4	
method) 318 (actPatterns(sDataFrame())	
get atternssbatariane() get atternssbatariane()	
method) 155 method) 7	
memou), 155 memou), 7	
getratterns() (PAMI.stablePerioaicFrequentPattern.basigestmacheevitsABBestmacheevit	
memou), 131 (PAMI.JrequentPattern.dusic.ECLAIDUset.ECLAIDUset	
getracterns() (rawii.stadiererioaicrrequentratiern.topk.isPilv.isPilv.a), 12	
memou), 139 getratternsAsuatarrame()	
getratterns() (PAMI.uncertainFrequentPattern.basic.CUFPIree.Q(PAMInprequentPattern.basic.ECLAIDiffset.ECLAIDiffset	
method), 210 method), 10	

getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.frequentPattern.basic.FPGrowth.FPGrowth (PAMI.highUtilitySpatialPattern.basic.SHUIM.SHUIM method), 15 method), 205 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.frequentPattern.closed.CHARM.CHARM (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHUIM method), 18 method), 210 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.frequentPattern.maximal.MaxFPGrowth.MaxFPGrowth.axFPGrowth.localPeriodicPattern.basic.LPPGrowth.LPPGrowth method), 21 method), 99 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.frequentPattern.topk.FAE.FAE method), (PAMI. local Periodic Pattern. basic. LPPMB readth. LPPMB readthmethod), 105 24 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.fuzzyCorrelatedPattern.basic.FCPGrowth.FCPGrowthRAMI.localPeriodicPattern.basic.LPPMDepth.LPPMDepthmethod), 250 method), 108 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.fuzzyFrequentPattern.basic.FFIMiner.FFIMiner (PAMI.multipleMinimumSupportBasedFrequentPattern.basic.CF) method), 246 method), 33 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.fuzzyGeoreferencedFrequentPattern.basic.FFSPMin&PAMInwinePattern.basic.CFL method), 255 method), 36 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.fuzzyGeoreferencedPeriodicFrequentPattern.basic.FGHAMPManetidfEdFtoEdMEnerguentPattern.basic.GPFgrowth.GPFgro method), 263 method), 112 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.fuzzyPeriodicFrequentPattern.basic.FPFPMiner.FPF(PMWhepartialPeriodicFrequentPattern.basic.PPF_DFS.PPF_DF method), 259 method), 119 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.georeferencedFrequentPattern.basic.SpatialECLAT.StpRAMIEGLAMIPeriodicPattern.basic.GThreePGrowth.GThreePGr method), 131 method), 170 getPatternsAsDataFrame() getPatternsAsDataFrame() $(PAMI.georeferenced Partial Periodic Pattern.basic.STEclat.S(IHAMM.partial Periodic Pattern.basic.PPP_ECLAT.PPP_EC$ method), 179 method), 127 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.geoReferencedPeriodicFrequentPattern.basic.GPFPMHAML;RHFHMPeeiodicPattern.basic.PPPGrowth.PPPGrowth method), 174 method), 123 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.highUtilityFrequentPattern.basic.HUFIM.HUFIM (PAMI.partialPeriodicPattern.closed.PPPClose.PPPClose method), 191 method), 134 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.highUtilityGeoreferencedFrequentPattern.basic.SHU(PAMISHUHFilMPeriodicPattern.maximal.Max3PGrowth.Max3PGro method), 196 method), 138 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.highUtilityPattern.basic.EFIM.EFIM (PAMI.partialPeriodicPattern.topk.k3PMiner.k3PMiner method), 219 method), 142 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.partial Periodic Pattern In Multiple Time Series. PPG rowth. PICON PARTICULAR Series (PAMI.partial Periodic Pattern In Multiple Time Series) (PAMI.partial Pattern In Multiple Time Series(PAMI.highUtilityPattern.basic.HMiner.HMiner method), 222 method), 329 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.periodicCorrelatedPattern.basic.EPCPGrowth.EPCPGrowth.EPCPGrowth.epcP(PAMI.highUtilityPattern.basic.UPGrowth.UPGrowth method), 225 method), 146 getPatternsAsDataFrame() getPatternsAsDataFrame() (PAMI.highUtilitySpatialPattern.basic.HDSHUIM.HDSHUIMAMI.periodicFrequentPattern.basic.PFECLAT.PFECT.PFECLAT.PFECLAT.PFECT.PFECT.PFECT.PFECT.PFECT.Pmethod), 200 method), 76

getPatternsAsDataFrame() (PAML pariodiaEnguantPattern hasia PEPC row	getPatt	ernsAsDataFrame()
(PAMI.perioaicFrequentFattern.basic.FFFGrow)	IN.PFPGR	math ad 274
methoa), 63	****D***	method), 274
getPatternsAsDataFrame()	getPatt	ernsaspataframe()
(PAMI.periodicFrequentPattern.basic.PFPGrow)	thPlus.PFI	P (Provintumas rtainFrequentPattern.basic.1ubeP.1UFP
method), 67		method), 282
getPatternsAsDataFrame()	getPatt	ernsAsDataFrame()
(PAMI.periodicFrequentPattern.basic.PFPMC.P	FPMC	(PAMI.uncertainFrequentPattern.basic.TubeS.TubeS
method), 80		<i>method</i>), 286
getPatternsAsDataFrame()	getPatt	ernsAsDataFrame()
(PAMI.periodicFrequentPattern.basic.PSGrowth	.PSGrowth	n (PAMI.uncertainFrequentPattern.basic.TUFP.TUFP
method), 72		<i>method</i>), 278
getPatternsAsDataFrame()	getPatt	ernsAsDataFrame()
(PAMI.periodicFrequentPattern.closed.CPFPMi	ner.CPFPl	MiPAMI.uncertainFrequentPattern.basic.UFGrowth.UFGrowth
method), 84		<i>method</i>), 290
<pre>getPatternsAsDataFrame()</pre>	getPatt	ernsAsDataFrame()
(PAMI.periodicFrequentPattern.maximal.MaxPI)	⁷ Growth.M	la EANG noneth rtainFrequentPattern.basic.UVECLAT.UVEclat
method), 88		<i>method</i>), 293
getPatternsAsDataFrame()	getPatt	ernsAsDataFrame()
(PAMI.periodicFrequentPattern.topk.kPFPMinet	r.kPFPMin	http://www.comment.com/ansatzanter/an
method), 91		<i>method</i>), 307
<pre>getPatternsAsDataFrame()</pre>	getPatt	ernsAsDataFrame()
(PAMI.periodicFrequentPattern.topk.TopkPFP.Te	opkPFP.To	p(#PAMPIGmoverhainPeriodicFrequentPattern.basic.UPFPGrowth.UF
method), 94	•	method), 298
getPatternsAsDataFrame()	getPatt	ernsAsDataFrame()
(PAMI.recurringPattern.basic.RPGrowth.RPGro	wth	(PAMI.uncertainPeriodicFrequentPattern.basic.UPFPGrowthPlu
method). 163		method). 302
getPatternsAsDataFrame()	getPatt	ernsAsDataFrame()
(PAMI.relativeFrequentPattern.basic.RSFPGrow	th.RSFPG	ropAML weighted FrequentNeighbourhood Pattern basic. SWFPGrov
method), 29		method). 238
getPatternsAsDataFrame()	getPatt	ernsAsDataFrame()
(PAMI relativeHighUtilityPattern basic RHUIM	RHIJIM	(PAMI weightedFrequentPattern basic WFIM WFIM
method) 186	ini o na	method) 229
actPatterns(sDataFrame()	no+Pott	erns ()
(PAMI sequential Pattern Mining basic prefix Spar	geti att	ref PAMI weighted Frequent Regular Pattern basic WFRIMiner WFRI
(1 AMI.sequential alternitining.basic.prejisspar method) 321	і.ргелілэри	method) 234
actPatterns(sDataFrame()	aot Dor	Sup() (PAMI pariodic Fraquent Pattern tonk kPEPMinar kPEPMin
(DAMI sequential Pattern Mining basic SDADE SI	UNDE	mothod) 01
(1 AMI.sequentian anermining.basic.si ADE.SI	ADE	adAndSupport () (in module
memou), 515	getreri	DAMInania dia Encauent Datterre hagia DEC nouth)
(DAML as quantial Dattern Mining hasis SDAM SD	A 1 A	PAMI.perioaicFrequeniPatiern.basic.PSGrowin),
(PAMI.sequentiaiPatternMining.basic.SPAM.SPA	1///	$\frac{12}{12}$
method), 318	getPmus	() (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Transaction
getPatternsAsDataFrame()		method), 215
(PAMI.stablePeriodicFrequentPattern.basic.SPP	Egentkun	Liwed) (PAMI.correlatedPattern.basic.CoMine.CoMine
method), 155	_	method), 39
getPatternsAsDataFrame()	getRunt	ime() (PAMI.correlatedPattern.basic.CoMinePlus.CoMinePlus
(PAMI.stablePeriodicFrequentPattern.basic.SPP	Growth.SF	PAGetooveth, 42
<i>method</i>), 151	getRunt	ime() (PAMI.coveragePattern.basic.CMine.CMine
getPatternsAsDataFrame()		method), 53
(PAMI.stablePeriodicFrequentPattern.topK.TSPI	!∕∕ġ €£Ri ûNt	<pre>ime() (PAMI.coveragePattern.basic.CPPG.CPPG</pre>
method), 159		<i>method</i>), 56
getPatternsAsDataFrame()	getRunt	ime() (PAMI.faultTolerantFrequentPattern.basic.FTApriori.FTAp
(PAMI.uncertain Frequent Pattern.basic.CUFPTrestore)	ee.CUFPT	reneethod), 46
<i>method</i>), 270	getRunt	ime() (PAMI.faultTolerantFrequentPattern.basic.FTFPGrowth.FT

method), 49	<i>method</i>), 108
getRuntime() (PAMI.frequentPattern.basic.Apriori.ApriorigetRunt	ime() (PAMI.multipleMinimumSupportBasedFrequentPattern.bas
method), 4	<i>method</i>), 33
getRuntime() (PAMI.frequentPattern.basic.ECLAT.ECLATgetRunt	ime() (PAMI.multipleMinimumSupportBasedFrequentPattern.bas
method), 7	method), 36
getRuntime() (PAMI.frequentPattern.basic.ECLATbitset.EGLRTht	titate() (PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth.G
method), 12	<i>method</i>), 112
getRuntime() (PAMI.frequentPattern.basic.ECLATDiffset.GetRafe	<code>Diffect) (PAMI.partialPeriodicFrequentPattern.basic.PPF_DFS.PP</code>
<i>method</i>), 10	<i>method</i>), 119
getRuntime() (PAMI.frequentPattern.basic.FPGrowth.FPGetRunt	ime() (PAMI.partialPeriodicPattern.basic.GThreePGrowth.GTh
method), 15	<i>method</i>), 131
getRuntime() (PAMI.frequentPattern.closed.CHARM.CHAgenRunt	ime() (PAMI.partialPeriodicPattern.basic.PPP_ECLAT.PPP_EC
method), 18	<i>method</i>), 127
getRuntime() (PAMI.frequentPattern.maximal.MaxFPGrogetRMnt	ERECTO (PAMI.partialPeriodicPattern.basic.PPPGrowth.PPPGrow
<i>method</i>), 21	<i>method</i>), 123
getRuntime() (PAMI.frequentPattern.topk.FAE.FAE getRunt	<pre>ime() (PAMI.partialPeriodicPattern.closed.PPPClose.PPPClose</pre>
method), 24	<i>method</i>), 134
getRuntime() (PAMI.fuzzyCorrelatedPattern.basic.FCPG getRunc	PGrGy(F AMI.partialPeriodicPattern.maximal.Max3PGrowth.Max
<i>method</i>), 251	<i>method</i>), 138
getRuntime() (PAMI.fuzzyFrequentPattern.basic.FFIMinegetRNAM	izme() (PAMI.partialPeriodicPattern.topk.k3PMiner.k3PMiner
<i>method</i>), 246	<i>method</i>), 142
getRuntime() (PAMI.fuzzyGeoreferencedFrequentPattern.betRuffe	SRM()) (PRMIPMitical Periodic Pattern In Multiple Time Series. PPG ro
<i>method</i>), 255	<i>method</i>), 330
getRuntime() (PAMI.fuzzyGeoreferencedPeriodicFrequentPetRunt	inne(FOPPHNPMpieneindFiGOBiPAllainedPattern.basic.EPCPGrowth.EPC
<i>method</i>), 263	<i>method</i>), 146
getRuntime() (PAMI.fuzzyPeriodicFrequentPattern.basic. GEERMA	ieme (P)F(PANitheperiodicFrequentPattern.basic.PFECLAT.PFECLAT
method), 259	method), 76
getRuntime() (PAMI.georeferencedFrequentPattern.basic.getRuff	GhA())StRAMIEtCELiaTi cFrequentPattern.basic.PFPGrowth.PFPGro
<i>method</i>), 170	<i>method</i>), 63
getRuntime() (PAMI.georeferencedPartialPeriodicPatterndetRuff	Ement SOFFAMIL periodicFrequentPattern.basic.PFPGrowthPlus.PFF
<i>method</i>), 179	<i>method</i>), 67
getRuntime() (PAMI.geoReferencedPeriodicFrequentPattegretRunt	GRACE/PAMCiPeFileMinFenequentPattern.basic.PFPMC.PFPMC
<i>method</i>), 174	method), 80
getRuntime() (PAMI.highUtilityFrequentPattern.basic.HUgeror.htt	File () (PAMI.periodicFrequentPattern.basic.PSGrowth.PSGrowth
<i>method</i>), 191	method), 72
getRuntime() (PAMI.highUtilityGeoreferencedFrequentPagetRunds	imski)(IPAMISIPEriducFrequentPattern.closed.CPFPMiner.CPFPM
method), 196	method), 84
getRuntime() (PAMI.highUtilityPattern.basic.EFIM.EFIMgetRunt	ime() (PAMI.periodicFrequentPattern.maximal.MaxPFGrowth.M
method), 219	method), 88
getRuntime() (PAMI.highUtilityPattern.basic.HMiner.HMgedRunt	ime() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFPMin
method), 222	method), 91
getRuntime() (PAMI.highUtilityPattern.basic.UPGrowth.lgetRunt	time() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP.To
method), 225	method), 95
getkuntime() (PAMI.nighUtilitySpatialPattern.basic.HDSgetkant	LIDER UNITAMI. recurringPattern.basic.KPGrowth.KPGrowth
method), 200	method), 103
getkuntime() (PAMI.nighUtilitySpatialPattern.basic.SHUgetkunt	mae() (PAMI.relativeFrequentPattern.basic.KSFPGrowth.KSFPG
method), 205	metnoa), 29
getkuntime() (PAMI.nighUtilitySpatialPattern.topk.1KSHgentikunk	Sme (M/PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUIM
metnoa), 210	methoa), 180
yetxuittine()(FAMI.iocalFerioalCFattern.basic.LFPGrowge.tkcinc	rentering (FAMII.sequentialFatternMining.basic.prefixSpan.prefixSpan
meinua), yy	memou), 521 BMC + 6 De MI seguential Dettern Mining hearin CDADE CDADE
yethal () (FAMI. IOCUIFE IOUCFUIER DUSIC. LFF MBregenRuife	mothod) 212
memory), 103	memou), 515
yetruittiie() (FAMI.iocairenoaicFailern.basic.LFFMDe getKEnt	nume gun (1 AN11.Sequennan Fanernin Inning.Dasic.SFAN1.SFAM

method), 318	HMiner (class in PAMI.highUtilityPattern.basic.HMiner),
getRuntime() (PAMI.studiePerioaicr requentPattern.basic	C.SPPECIAL209PECIAL
meinoa), 155	HOF IN (class in FAMI.nigho lilliyFrequentPallern.basic.hOF IM),
getRuntime() (PAMI.stablePerioaicFrequentPattern.basic	c.SPPGrowin SPPGrowin
methoa), 151	TODIN TODIN
getRuntime() (PAMI.stablePerioalCFrequentPattern.topK	
methoa), 159	1F11e (PAMI.highUtilitySpatialPattern.topk.1KSHUIM.1KSHUIM
getRuntime() (PAMI.uncertainFrequentPattern.basic.CU	FPIree.Cuttorionate), 210
methoa), 270	insertionSort() (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Transe
getRuntime() (PAMI.uncertainFrequentPattern.basic.PU	FGrowth.Phylethorapyth15
method), 2/5	intersection() (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHU
getRuntime() (PAMI.uncertainFrequentPattern.basic.lub	PP.10FP method), 210
method), 282	intTostr (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHUIM
getRuntime() (PAMI.uncertainFrequentPattern.basic.lub	es. Tubes attribute), 210
method), 286	1s_equal() (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHUIM
getRuntime() (PAMI.uncertainFrequentPattern.basic.IU.	FP.IUFP method), 211
method), 278	
getRuntime() (PAMI.uncertainFrequentPattern.basic.UF	Gr owth. UFGrowth
method), 290	k3PMiner (class in PAMI.partialPeriodicPattern.topk.k3PMiner),
getRuntime() (PAMI.uncertainFrequentPattern.basic.UV)	ECLAT.UVEgjat
method), 293	kPFPMiner (class in PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFP.
getRuntime() (PAMI.uncertainGeoreferencedFrequentPa	ttern.basic.gFPGrowth.GFPGrowth
method), 307	
getRuntime() (PAMI.uncertainPeriodicFrequentPattern.b	d <u>s</u> ic.UPFPGrowth.UPFPGrowth
method), 298	lno (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPfPMiner.kPfPMiner.kPfpmin
getRuntime() (PAMI.uncertainPeriodicFrequentPattern.b	asic.UPFPGrowthPlus.UPFPGrowthPlus
method), 302	LPPGrowth (class in PAMI.localPeriodicPattern.basic.LPPGrowth),
getRuntime() (PAMI.weightedFrequentNeighbourhoodPa	ittern.basic ₉ WFPGrowth.SWFPGrowth
methoa), 239	LPPMBreadth (class in
getRuntime() (PAMI.weightedFrequentPattern.basic.wFI	M.WFIM PAMI.localPeriodicPattern.basic.LPPMBreadth),
method), 230	102
getRuntime() (PAMI. weigniear requent Regular Pattern. bo	¹² PPMDEptin(cluss in PAMI.localPeriodicPattern.basic.LPPMDepth),
methoa), 234	105
getSameSeq() (PAMI.sequentialPatternMining.basic.prefi	xSpan.prefixSpan N
method), 322	IVI
getIransactions() (PAMI.nignUnitySpanalPattern.top)	main() (in module PAMI.fuzzyCorrelatedPattern.basic.FCPGrowth),
memoa), 200	251
getotilities() (PAMI.nignOnniySpanarPanern.iopk.TR	main() (in module PAMI.highUtilityGeoreferencedFrequentPattern.basic.Si
meinoa), 215 CEDC reasth (alaga in DAML un contain C constance of Energy of	197 ADattern hasis CEDCrowth
302	main() (in module PAMI.highUtilitySpatialPattern.topk.TKSHUIM),
505 CDE anorith (alaga in DAML nantial Davia dia Enaguant Dattaun	hasis CREanouth)
	make ILenDatabase() (PAMI.sequentialPatternMining.basic.SPADE.SPAI
CDEDMinor (class in DAMI and Pateron ad Pariodic Fraquent	method), 313 Pattern basis (CPEPMinar)
171	make2BitDatabase() (PAMI.sequentialPatternMining.basic.SPAM.SPAM
Three DC routh (alass in	<i>method</i>), 318
GIIII EEPGI OW LII (Cluss In PAMI partial Davia dia Pattern hasia CThree PC no	make2LenDatabase() (PAMI.sequentialPatternMining.basic.SPADE.SPAI
	(<i>vin</i>), method), 313
120	<pre>make3LenDatabase() (PAMI.sequentialPatternMining.basic.SPADE.SPAI</pre>
Н	<i>method</i>), 313
UDSUITM (alass in DA MI biah I tility. So at al Dattane have a	makeNext() (PAMI.sequentialPatternMining.basic.prefixSpan.prefixSpan
ווטסרכעה (class in FAMI.nignOlilliySpallalFallern.basiC.HL 107	, method), 322
17/ hoopiict (DAMI high Iltility Spatial Dattoms tonk TVSIIIII)	makeNextRow() (PAMI.sequentialPatternMining.basic.SPADE.SPADE
attribute), 210	(in Kill of Method), 313

makeNextRow	Same() (PAMI.sequentic	ılPatternMining.basi	mSRADE.	SPA DÆ MI.frequentPattern.basic.Apriori.Apriori
meth	od), 313			method), 4
makeNextRow	Same2() (PAMI.sequent	ialPatternMining.bas	miseade	E.SRRAMII.frequentPattern.basic.ECLAT.ECLAT
meth	od), 313			method), /
makeNextRow	Same3() (PAMI.sequent	ialPatternMining.bas	maeap	PSMAIme quentPattern.basic.ECLATbitset.ECLATbitset
meth	(od), 314			method), 13
makenextSam	e() (PAMI.sequentialPa	tternMining.basic.pr	eyaxopqn.y	Hajus prenquent Pattern. basic. ECLAIDiffset. ECLAIDiffset
metro	(0a), 522		mina()(methoa), 10 PAMI from out Pattern hasis EPC routh EPC routh
	Daser ITSL() Al soquential Pattern Min	ing basic profix Span	mine()()	PAMI.Jrequentratiern.basic.FFGrowin.FPGrowin
(I Al	an.sequentian anerman ad) 322	ing.basic.prejixspan.	prejixspui mino()	(PAMI frequentPattern closed CHARM CHARM
makeSeqData	baseSame()		mine()	method) 18
INARESEQUATA	M soquontialPatternMin	ing basic prefixSpan	miatio Cindu	RAMI frequentPattern maximal MaxEPGrowth MaxEPGrowth
(1711 metl	(a) 322	ing.busic.prejixspun.	han na han an han h	method) 21
makeSunData	base() (PAMI sequentic	lPatternMining basi	minateSn	(PAMAIffrSamentPattern tonk FAF FAF method)
meth	(ad) 322	in anermining.basi	μμειομωp	24
makexLenDat	ahase()(PAMI sequent	ialPatternMining bay	mi SPAD A	PARATOF zvCorrelatedPattern basic FCPGrowth FCPGrowth
meth	(ad) 314			method) 251
makexLenDat	abaseSame()		mine()(PAML fuzzy Frequent Pattern, basic, FFIMiner, FFIMiner
(PA)	AL sequentialPatternMin	ing.basic.SPADE.SPA	ADE	method), 246
meth	(od), 314		mine()(PAML fuzzyGeoreferencedFrequentPattern.basic.FFSPMiner.FFSI
mapNeighbou	rs() (PAMI.georeferenc	edPartialPeriodicPat	tern.basic	The second states and the se
meth	od), 179		mine()(PAMI.fuzzyGeoreferencedPeriodicFrequentPattern.basic.FGPFPN
mapNeighbou	rs() (PAMI.geoReferend	edPeriodicFrequent	Pattern.ba	sine.GBBPPMiner.GPFPMiner
meth	od), 174		mine()(PAMI.fuzzyPeriodicFrequentPattern.basic.FPFPMiner.FPFPMine
Max3PGrowth	(class	in		method), 259
PAN	I.partialPeriodicPattern	.maximal.Max3PGro	mihne()(PAMI.georeferencedFrequentPattern.basic.SpatialECLAT.SpatialPattern.basic.SpatialPattern.ba
135				<i>method</i>), 170
MaxFPGrowth	(class	in	mine()(PAMI.geore ferenced Partial Periodic Pattern.basic. STE clat. STE clat
PAN	I.frequentPattern.maxim	al.MaxFPGrowth),		<i>method</i>), 179
19			mine()(PAMI.geo Reference dPeriodic Frequent Pattern.basic.GPFPM iner.Commence of the theory of the term of ter
<pre>maxItem(PAA</pre>	11.highUtilitySpatialPatt	ern.topk.TKSHUIM.I	Dataset	<i>method</i>), 174
attri	bute), 206		mine()(PAMI.highUtilityFrequentPattern.basic.HUFIM.HUFIM
<pre>maxMemory(#</pre>	AMI.highUtilitySpatialPo	attern.topk.TKSHUII	M.TKSHU	(<i>Method</i>), 192
attri	bute), 211		mine()(PAMI.highUtilityGeore ferencedFrequentPattern.basic.SHUFIM.Structures Structures Struc
MaxPFGrowth	(class	in		<i>method</i>), 196
PAN	I.periodicFrequentPatte	rn.maximal.MaxPFG	mineh())	(PAMI.highUtilityPattern.basic.EFIM.EFIM
84				<i>method</i>), 219
memoryRSS(P	AMI.highUtilitySpatialPo	attern.topk.TKSHUII	ALTHESH U	(IFAMI.highUtilityPattern.basic.HMiner.HMiner
attri	<i>bute</i>), 211			<i>method</i>), 222
memoryUSS (F	AMI.highUtilitySpatialPo	attern.topk.TKSHUII	ahimkexp ()	PAMI .highUtilityPattern.basic.UPGrowth.UPGrowth
attri	<i>bute</i>), 211	~		<i>method</i>), 225
mine() (PAN	II.correlatedPattern.basi	c.CoMine.CoMine	mine()(PAMI.highUtilitySpatialPattern.basic.HDSHUIM.HDSHUIM
meth	od), 40		N I () ()	method), 200
mine()(PAM	correlatedPattern.basic	.CoMinePlus.CoMin	antines () (I	PAMI.highUtilitySpatialPattern.basic.SHUIM.SHUIM
. meth	(od), 42		· 0.0	method), 205
mine() (PAMI.coveragePattern.be	asic.CMine.CMine	mine()(PAMI.highUtilitySpatialPattern.topk.1KSHUIM.1KSHUIM
meth	(0a), 55	The CDDC CDDC		method), 211
mine()	PAMI.coveragePattern.b	asic.CPPG.CPPG	mine()(PAMI.localPerioalCPattern.basic.LPPGrowth.LPPGrowth
meth	(00), 30 I faultTolonantEnsing (D	attom basis FTA-	wiFFK	Internou), 99 BAMI local David in Dattoring having I DDMD
mine() (PAM	.jauii10ierantFrequentPo	uuern.basic.r 1Aprio	uu uu ue Aprili	method) 105
mine()(DAM	00), 40 [faultTolerantFraguer+D	attern basic FTEDC.	mi <i>n</i> le ETE	memou), 100 PPMMhuthalPariodicPattern havie I DDMD anth I DDMD anth
matle () (I AM	ad) 49		unik 115 () ()	method) 109
men	οu), τ <i>γ</i>			<i>incurva</i>), 107

Mine() (PAMI.multipleMinimumSupportBasedFrequentPatmin method) 22	b(i)i(PANHGinexthaGilFPGguentR attern.basic.TUFP.TUFP
memoa), 52 Mino () (DAMI multinlo Minimum Sunn out Data dEns au out Dat min	memou), 270
mine() (PAMI.mullipleminimumSupportBasearrequentPathatan	eusiceante and 200
memou), 50 mino() (DAMI nantialDaviodiaEnaquentDattown basia CDEamim	memou), 290
mille() (FAMI.partialFerioalCFTequentFullern.busic.OFFg	B.G.F.F.g.KOWMIN CETIAINFTEQUENIFALIETT.DUSIC.OVECLAT.OVECIAI mothod) 203
memou), 112 mino() (PAMI partialPariodicEreauentPattern basic DDE DES	memou), 295
mille() (IAMI.partial enouicrequent altern.basic.111_imili method) 110	(1)(<u>1</u> HWIS.uncertainGeorejerencearrequenti altern.basic.OFT Growin.
memou), 119 mino() (PAMI nartialPariodicPattern basic CThreePCrowthig)	meinoa), 507 FK 206 DOM unh eart ain Pario die Fraquent Pattern basie UDE DC rowth UDE
mille() (IAMI.partial enouici alern.basic.OThreef Orowi m.p	method) 208
Mino() (PAMI partialPariodicPattern basic PPP ECIAT PPP	memou), 296
method) 126	method) 302
memou), 120	memou), 502
mille() (IAMI.partial enourier alern.basic.111 Orowin.11	method) 230
memou), 125	(PAMI weighted Frequent Pattern basic WFIM WFIM
method) 134	method) 230
mine() (PAMI partialPeriodicPattern maximal Max3PGrowthm	AGY 3 PGMMstb ightedFrequentRegularPattern basic WFRIMiner WFRIM
method) 139	method) 234
mine() (PAMI partialPeriodicPattern tonk k3PMiner k3PMinim	IItil (PAML highUtilitySpatialPattern topk TKSHUIM TKSHUIM
method). 142	attribute). 211
Mine() (PAMI partialPeriodicPatternInMultipleTimeSeries	And the PPG rowth
method). 329	PAMI.correlatedPattern.basic.CoMine.37
Mine() (PAMI.periodicFrequentPattern.basic.PFECLAT.PFEC	/PAMT.correlatedPattern.basic.CoMinePlus
method). 76	40
Mine() (PAMI periodicFrequentPattern basic PFPGrowth PFP	(PAMThcoveragePattern.basic.CMine.50)
method) 63	PAMI coveragePattern basic CPPG 54
Mine() (PAMI.periodicFrequentPattern.basic.PSGrowth.PSGro	WMMI.faultTolerantFrequentPattern.basic.FTApriori.
method). 71	43
Mine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.CP.	FPAMInefaultTolerantFrequentPattern.basic.FTFPGrowth.
method). 83	46
Mine() (PAMI.periodicFrequentPattern.maximal.MaxPFGrowt	hPMMIPFfequentPattern.basic.Apriori.2
method). 87	PAMI.frequentPattern.basic.ECLAT.5
Mine() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFI	P. TAMAP. FP Convents Pattern. basic. ECLAT bitset.
method), 94	10
Mine() (PAMI.recurringPattern.basic.RPGrowth.RPGrowth	PAMI.frequentPattern.basic.ECLATDiffset,
method), 163	8
Mine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RSF	PCAMI wthrequentPattern.basic.FPGrowth, 13
method), 28	PAMI.frequentPattern.closed.CHARM, 16
Mine() (PAMI.sequentialPatternMining.basic.prefixSpan.prefix.	SPAMI.frequentPattern.maximal.MaxFPGrowth,
<i>method</i>), 321	19
Mine() (PAMI.sequentialPatternMining.basic.SPADE.SPADE	PAMI.frequentPattern.topk.FAE, 22
<i>method</i>), 312	PAMI.fuzzyCorrelatedPattern.basic.FCPGrowth,
mine() (PAMI.stablePeriodicFrequentPattern.basic.SPPEclat.S	PPEclat7
<i>method</i>), 155	PAMI.fuzzyFrequentPattern.basic.FFIMiner,
mine() (PAMI.stablePeriodicFrequentPattern.basic.SPPGrowth	n.SPPG#@wth
<i>method</i>), 151	PAMI.fuzzyGeoreferencedFrequentPattern.basic.FFSPMiner
mine() (PAMI.uncertainFrequentPattern.basic.CUFPTree.CUF	PTree251
<i>method</i>), 271	PAMI.fuzzyGeoreferencedPeriodicFrequentPattern.basic.F
mine() (PAMI.uncertainFrequentPattern.basic.PUFGrowth.PU	FGrowth
method), 275	PAMI.fuzzyPeriodicFrequentPattern.basic.FPFPMiner,
mine() (PAMI.uncertainFrequentPattern.basic.TubeP.TUFP	256
<i>method</i>), 282	<pre>PAMI.georeferencedFrequentPattern.basic.SpatialECLAT,</pre>
mine() (PAMI.uncertainFrequentPattern.basic.TubeS.TubeS	167
<i>method</i>), 286	PAMI.georeferencedPartialPeriodicPattern.basic.STEclat

175	77
PAMI.geoReferencedPeriodicFrequentPattern.bas	siRAMPFMMinadi,cFrequentPattern.basic.PSGrowth,
PAMI.highUtilityFrequentPattern.basic.HUFIM,	PAMI.periodicFrequentPattern.closed.CPFPMiner,
PAMI.highUtilityGeoreferencedFrequentPattern.	baamE., penditi, cFrequentPattern.maximal.MaxPFGrowth,
192	84
PAMI.highUtilityPattern.basic.EFIM, 216 PAMI.highUtilityPattern.basic.HMiner, 219	PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFPMiner, 88
PAMI.highUtilityPattern.basic.UPGrowth, 223	<pre>PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP, 92</pre>
PAMI.highUtilitySpatialPattern.basic.HDSHUIM,	PAMI.recurringPattern.basic.RPGrowth, 160
197 PAMI.highUtilitySpatialPattern.basic.SHUIM.	25
201	PAMI.relativeHighUtilitvPattern.basic.RHUIM
PAMI.highUtilitySpatialPattern.topk.TKSHUIM,	183
205	PAMI.sequentialPatternMining.basic.prefixSpan,
PAMI.localPeriodicPattern.basic.LPPGrowth,	318
96	PAMI.sequentialPatternMining.basic.SPADE,
PAMI.localPeriodicPattern.basic.LPPMBreadth,	310
102	PAMI.sequentialPatternMining.basic.SPAM,
PAMI.localPeriodicPattern.basic.LPPMDepth,	314
105	PAMI.sequentialPatternMining.closed.bide,
PAMI.multipleMinimumSupportBasedFrequentPatte	ern.basic.CFPGrowth,
30	<pre>PAMI.stablePeriodicFrequentPattern.basic.SPPEclat,</pre>
PAMI.multipleMinimumSupportBasedFrequentPatte	ern.balsidc.CFPGrowthPlus,
33	PAMI.stablePeriodicFrequentPattern.basic.SPPGrowth,
PAMI.partialPeriodicFrequentPattern.basic.GPF	growth,7
110	<pre>PAMI.stablePeriodicFrequentPattern.topK.TSPIN,</pre>
PAMI.partialPeriodicFrequentPattern.basic.PPF	F_DFS,156
116	PAMI.uncertainFrequentPattern.basic.CUFPTree,
PAMI.partialPeriodicPattern.basic.GThreePGrow	rth, 267
128	PAMI.uncertainFrequentPattern.basic.PUFGrowth,
PAMI.partialPeriodicPattern.basic.PPP_ECLAT,	271
124	<pre>PAMI.uncertainFrequentPattern.basic.TubeP,</pre>
PAMI.partialPeriodicPattern.basic.PPPGrowth,	279
120	<pre>PAMI.uncertainFrequentPattern.basic.TubeS,</pre>
<pre>PAMI.partialPeriodicPattern.closed.PPPClose,</pre>	283
131	PAMI.uncertainFrequentPattern.basic.TUFP,
PAMI.partialPeriodicPattern.maximal.Max3PGrow	rth, 275
135	PAMI.uncertainFrequentPattern.basic.UFGrowth,
PAMI.partialPeriodicPattern.topk.k3PMiner,	287
139	PAMI.uncertainFrequentPattern.basic.UVECLAT,
PAMI.partialPeriodicPatternInMultipleTimeSeri	es.PPCGrowth.
326	PAMI.uncertainGeoreferencedFrequentPattern.basic.GFPGr
PAMI.periodicCorrelatedPattern.basic.EPCPGrow	rth. 303
143	PAMI.uncertainPeriodicFrequentPattern.basic.UPFPGrowth
PAMI.periodicFrequentPattern.basic.PFECLAT.	294
73	PAMI_uncertainPeriodicFrequentPattern_basic_UPFPGrowth
PAMI.periodicFrequentPattern basic PFPGrowth	299
61	PAMT weightedFrequentNeighbourhoodPattern basic SWFPGr
PAMI periodicFrequentPattern basic PFPCrowthF	Plus 235
64	PAMT weightedFrequentPattern basic WFTM
PAMI periodicFrequentPattern basic PFPMC	226
per rourer requester accertiousrent i i inc,	

PAMI.weightedFrequentRegularPattern.bas	ic.WARIMfnequentPattern.topk.FAE
231	module, 22
N	PAMI.fuzzyCorrelatedPattern.basic.FCPGrowth
	module, 247
Neighbours (PAMI.highUtilitySpatialPattern.topk.TKSI attribute), 209	<pre>With the second se</pre>
newNamesToOldNames(PAMI.highUtilitySpatialPattern	tor RAMES FUZZA GROSHE FUZZA CON FUZZA CON CONTRACTOR CON CONTRACTOR CONTRAC
attribute). 211	module, 251
nFile (PAMI.highUtilitySpatialPattern.topk.TKSHUIM. attribute) 211	<pre>TK\$PAUTMfuzzyGeoreferencedPeriodicFrequentPattern.basic.FGPFI module.260</pre>
Node (<i>class in PAMI.localPeriodicPattern.basic.LPPGro</i>	wthP,AMI.fuzzyPeriodicFrequentPattern.basic.FPFPMiner
Node (<i>class in PAMI.partialPeriodicFrequentPattern.bas</i>	<i>ic.OFAMGrogenty</i> ,eferencedFrequentPattern.basic.SpatialECLAT
Node (class in PAMI periodic Frequent Pattern basic PSG	rou PAMI georeferencedPartialPeriodicPattern basic STEclat
68	module 175
00	PAMI, geoReferencedPeriodicFrequentPattern, basic, GPFPMiner
0	module 171
offsot (PAMI high Utility Spatial Pattern tonk TKSHUM	TrBAMI abighUtilitvFrequentPattern.basic.HUFIM
attribute) 215	module. 188
oFile (PAMI high Utility Spatial Pattern tonk TKSHIIIM	TK @AMJAhighUtilityGeoreferencedFrequentPattern.basic.SHUFIM
attribute) 211	module. 192
oldNamosToNouNamos (PAMI high Utility Spatial Pattern	tor PAMES by abut this get Pattern basic FFIM
attributa) 211	module 216
output () (PAMI high Utility Spatial Pattern tonk TKSHI	IIMPhAMTLIhinhIItilityPattern basic HMiner
method) 211	module 219
meinoa), 211	PAMI highlitilityPattern basic IIPGrowth
Р	module 223
	PAMI, highlitilitySpatialPattern, basic, HDSHUIM
PAMI.correlatedPattern.basic.Comine	module 197
module, 37	PAMI_highUtilitySpatialPattern_basic_SHUIM
PAMI.correlatedPattern.basic.CoMinePlus	module, 201
PAMT coveragePattern basic (Mine	PAMI.highUtilitySpatialPattern.topk.TKSHUIM
module 50	module, 205
PAMT coveragePattern basic CPPC	PAMI.localPeriodicPattern.basic.LPPGrowth
module 54	module, 96
PAMI faultTolerantFrequentPattern hasic FTA	nriPAMI.localPeriodicPattern.basic.LPPMBreadth
module 43	module, 102
PAMI faultTolerantFrequentPattern hasic FTF	PC_PAMI.localPeriodicPattern.basic.LPPMDepth
module 46	module, 105
PAMI frequentPattern basic Apriori	PAMI.multipleMinimumSupportBasedFrequentPattern.basic.CFP
module 2	module, 30
PAMI frequentPattern basic FCLAT	PAMI.multipleMinimumSupportBasedFrequentPattern.basic.CFP
module 5	module, 33
PAMT frequentPattern basic FCI AThitset	PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth
module 10	module, 110
PAMI frequentPattern basic FCI ATDiffset	PAMI.partialPeriodicFrequentPattern.basic.PPF_DFS
module 8	module, 116
PAMI frequentPattern basic FPGrowth	PAMI.partialPeriodicPattern.basic.GThreePGrowth
module 13	module, 128
PAMI frequentPattern closed CHARM	PAMI.partialPeriodicPattern.basic.PPP_ECLAT
module 16	module, 124
PAMI frequentPattern maximal MaxFDCrowth	PAMI.partialPeriodicPattern.basic.PPPGrowth
module. 19	module, 120

PAMI.partialPeriodicPattern.closed.PPPClose module, 131	PAMI.uncertainFrequent module, 283	Pattern.basic.Tube	S
PAMI.partialPeriodicPattern.maximal.Max3PGrow	tPAMI.uncertainFrequent	Pattern.basic.TUFP	
module, 155	DAMI uncontainEncouont	Dattorn hasis UECr	a+h
module, 139	module, 287	Pattern.basic.urgr	οωτη
<pre>PAMI.partialPeriodicPatternInMultipleTimeSeri module, 326</pre>	eBAMPPGnocethtainFrequent module, 290	Pattern.basic.UVEC	LAT
<pre>PAMI.periodicCorrelatedPattern.basic.EPCPGrow module, 143</pre>	tPAMI.uncertainGeorefer module, 303	encedFrequentPatte	rn.basic.GFPGrowth
<pre>PAMI.periodicFrequentPattern.basic.PFECLAT module,73</pre>	PAMI.uncertainPeriodic module, 294	FrequentPattern.ba	sic.UPFPGrowth
<pre>PAMI.periodicFrequentPattern.basic.PFPGrowth module, 61</pre>	PAMI.uncertainPeriodic module, 299	FrequentPattern.ba	sic.UPFPGrowthPlus
<pre>PAMI.periodicFrequentPattern.basic.PFPGrowthP module, 64</pre>	<pre>IRAMI.weightedFrequentN module, 235</pre>	eighbourhoodPatter	n.basic.SWFPGrowth
<pre>PAMI.periodicFrequentPattern.basic.PFPMC module, 77</pre>	PAMI.weightedFrequentPamodule, 226	attern.basic.WFIM	
<pre>PAMI.periodicFrequentPattern.basic.PSGrowth module, 68</pre>	PAMI.weightedFrequentR module, 231	egularPattern.basi	c.WFRIMiner
<pre>PAMI.periodicFrequentPattern.closed.CPFPMiner module, 81</pre>	PFECLAT (<i>class in PAMI.perio</i> 73	odicFrequentPattern.bas	sic.PFECLAT),
<pre>PAMI.periodicFrequentPattern.maximal.MaxPFGro module, 84</pre>	w Pth growth (<i>class in PAMI.par</i> 113	rtialPeriodicFrequentPa	ttern.basic.GPFgrowth
<pre>PAMI.periodicFrequentPattern.topk.kPFPMiner.k module, 88</pre>	PIFFFFGineerth (<i>class in PAMI.pe</i> 61	eriodicFrequentPattern.	basic.PFPGrowth),
<pre>PAMI.periodicFrequentPattern.topk.TopkPFP.Top module.92</pre>	k PFFP GrowthPlus PAMI.periodicFrea	(class quentPattern.basic.PFP(in GrowthPlus),
PAMI.recurringPattern.basic.RPGrowth	64	:	· · ·
module, 160	PFPMC (class in PAMI.periodi	icFrequentPattern.basic.	PFPMC),
PAMI.relativeFrequentPattern.basic.RSFPGrowth	77		
module, 25	PPF_DFS (class in PAMI.part)	ialPeriodicFrequentPatt	ern.basic.PPF_DFS),
PAMI.relativeHighUtilityPattern.basic.RHUIM	116		
module, 183	PPGrowth (class in PAMI.par	rtialPeriodicPatternInM	ultipleTimeSeries.PPG
<pre>PAMI.sequentialPatternMining.basic.prefixSpan module, 318</pre>	326 PPP_ECLAT (<i>class in PAMI.pd</i>	artialPeriodicPattern.ba	sic.PPP_ECLAT),
PAMI.sequentialPatternMining.basic.SPADE	124 PPPClose (alass in PAMI not	utial Davia dia Dattauna alaa	ad DDDClose)
PAMI sequentialPatternMining basic SPAM	131	mairenouicraitern.clos	eu.FFFClose),
module 314	PPPCrowth (class in PAMI n	artialPeriodicPattern bo	sic PPPGrowth)
PAMI.sequentialPatternMining.closed.bide	120		
module, 323 PAMI.stablePeriodicFrequentPattern.basic.SPPE	prefixSpan(<i>class in PAMI.s</i> clat 318	sequentialPatternMining	g.basic.prefixSpan),
module, 152	<pre>prefixUtility(PAMI.high</pre>	UtilitySpatialPattern.to	pk.TKSHUIM.Transact
<pre>PAMI.stablePeriodicFrequentPattern.basic.SPPG module 147</pre>	rowth <i>attribute</i>), 215	rrelatedPattern basic Co	Mine CoMine
PAMI stablePeriodicErequentPattern tonk TSPIN	method) 40	retatear attern.basic.Co	mine.Comme
module, 156	printResults() (PAMI.con	rrelatedPattern.basic.Co	oMinePlus.CoMinePlus
PAMI.uncertainFrequentPattern.basic.CUFPTree	method), 43		in a CMin -
module, 20/	printkesuits() (PAMI.cov	eragePattern.basic.CM	ine.CMine
module, 271	<pre>printResults() (PAMI.cov</pre>	veragePattern.basic.CPI	PG.CPPG
PAMI.uncertainFrequentPattern.basic.TubeP	method), 56		1
module, 279	printResults() (PAMI.fau	ltTolerantFrequentPatte	rn.basic.FTApriori.FT

<i>method</i>), 46	method), 109
printResults() (PAMI.faultTolerantFrequentPattern.basip	Ssulvi/sfyf#2MbwthltipleMinimumSupportBasedFrequentPattern.b
method), 49	method), 33
printResults() (PAMI.frequentPattern.basic.Apriori.Apr pri ntRe	esults() (PAMI.multipleMinimumSupportBasedFrequentPattern.t
method), 4	method), 3/
printResults() (PAMI. frequentPattern. basic. ECLAI. EC. printRe	esults() (PAMI.partialPerioalCFrequentPattern.basic.GPF growth
method), /	method), 112
princesuits() (PAMI. JrequentPattern. basic. ECLAT bitseptextucke	Busids () (PAMI.parital Ferioal CF requent Pattern. Dasic. FFF_DFS.
memou), 15 printPosults() (PAMI frequentPattern basic ECI ATDiffert FatPA	memou), 119 Thiffed) (PAMI partialPariodicPattern basic CThreePGrowth GT
method) 10	method) 131
nrintResults () (PAMI frequentPattern basic EPGrowth Refine Refin	memory, 151
method) 16	method) 127
nrintResults() (PAMI frequentPattern closed CHARM CHARMR	sults() (PAMI partialPeriodicPattern basic PPPGrowth PPPGr
method) 18	method) 123
nrintResults() (PAMI frequentPattern maximal MaxFPG mointhRe	Inchiou), 125
method) 21	method) 135
nrintResults() (PAMI fuzzyCorrelatedPattern basic FCPGrintile	KGPGKwtPAMI partialPeriodicPattern maximal Max3PGrowth M
method). 251	method). 139
printResults() (PAML fuzzy Frequent Pattern basic FFIM internation	() (PAML partial Periodic Pattern topk k3PMiner k3PMiner
method). 246	method). 142
printResults() (PAMI.fuzzyGeoreferencedFrequentPatteprhatRe	FESTRAN APENDMitter Periodic Pattern In Multiple Time Series. PPG
method). 255	method). 330
printResults() (PAMI.fuzzyGeoreferencedPeriodicFrequentiPartRet	B. Blasic (FOR Mienin di Cebil Pellaned Pattern. basic. EPCPGrowth. El
method), 264	method), 146
printResults() (PAMI.fuzzyPeriodicFrequentPattern.basiprERFR	Sinler: Stor PMM eperiodic Frequent Pattern. basic. PFECLAT. PFECL
method), 259	method), 76
printResults() (PAMI.georeferencedFrequentPattern.baspc.SptRe	http://www.astronaction.com/astronaction/astronaction/astronaction/astronaction/astronaction/astronaction/astro
<i>method</i>), 170	method), 63
printResults() (PAMI.georeferencedPartialPeriodicPattepnintRe	STELEA, SALAMAL periodic Frequent Pattern. basic. PFPGrowthPlus. F
<i>method</i>), 179	method), 67
printResults() (PAMI.geoReferencedPeriodicFrequentPaperintRe	scuOtSCOMPAMIGPEFiBMinFenequentPattern.basic.PFPMC.PFPMC
<i>method</i>), 175	method), 80
printResults() (PAMI.highUtilityFrequentPattern.basic. ##1#####	HUFIM() (PAMI.periodicFrequentPattern.basic.PSGrowth.PSGrov
<i>method</i>), 192	method), 72
printResults() (PAMI.highUtilityGeoreferencedFrequentPrintRe	baselet.SKIXIPHMISHeFAMCFrequentPattern.closed.CPFPMiner.CPF
<i>method</i>), 196	method), 84
printResults() (PAMI.highUtilityPattern.basic.EFIM.EF Øbr intRe	esults() (PAMI.periodicFrequentPattern.maximal.MaxPFGrowth
<i>method</i>), 219	method), 88
printResults() (PAMI.highUtilityPattern.basic.HMiner.HMiner	esults() (PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFPM
<i>method</i>), 222	<i>method</i>), 91
printResults() (PAMI.highUtilityPattern.basic.UPGrowtprUPCRe	psull ts() (PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP.
method), 225	method), 95
printResults() (PAMI.highUtilitySpatialPattern.basic.H pshtiRe	IstuDist () (MAMI. recurring Pattern. basic. RPG rowth. RPG rowth
method), 200	method), 164
printResults() (PAMI.highUtilitySpatialPattern.basic.SFpo1mtRe	IsuMs() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RSFF
method), 205	method), 29
printResults() (PAMI.highUtilitySpatialPattern.topk.TK printke	ISUM(PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUIM
method), 211	method), 186
printkesuits() (PAMI.localPeriodicPattern.basic.LPPG ponsiti.the	Esurasin (PAMI.sequentialPatternMining.basic.prefixSpan.prefixS
meinoa), yy	memou), 323
printresuits() (FAMI. local rerioal crattern. basic. LPPM predates	usuniussu gaataan sequeniiairaiierniviining.dasic.SPADE.SPADE
memory), 103	memou), 514 Dell'Epsold DAMI sequential Dattern Mining Lagis CDAM CDAM
princkesuits() (PAMI. local Periodic Pattern. basic. LPPM Depitt Re	STALLES WING FAMI. SEQUENTIAL FATTERNMINING. DASIC. SPAM. SPAM

method), 318	method), 215
printResults() (PAMI.stablePeriodicFrequentPattern.ba	RHISTY (Eldss. Spp: Allat elative High Utility Pattern. basic. RHUIM),
method), 155	183
printResults() (PAMI.stablePeriodicFrequentPattern.ba	RPCSHCRGh(whasSPPEArMithecurringPattern.basic.RPGrowth),
method), 151	160
<pre>printResults() (PAMI.stablePeriodicFrequentPattern.top</pre>	HSTBBHN/HSP (Hass in PAMI.relativeFrequentPattern.basic.RSFPGrowth),
method), 159	25
nrintResults () (PAMI uncertain Frequent Pattern basic (+IFUT voDMIFDFrim Periodic Frequent Pattern basic GPE growth calculate II
method) 271	method) 115
memou), 2/1	memou), 115
printkesuits() (PAMI.uncertainFrequentPattern.basic.P	EXECTION OF THE AND A CONTRACT OF THE AND A CONTRACT OF THE ADDRESS OF THE ADDRE
method), 275	method), 116
<pre>printResults() (PAMI.uncertainFrequentPattern.basic.Tell</pre>	advarf(JUPPAMI.partialPeriodicFrequentPattern.basic.GPFgrowth.generatePI
<i>method</i>), 282	<i>method</i>), 116
<pre>printResults() (PAMI.uncertainFrequentPattern.basic.Tell</pre>	index (Jul Person Section 2019) In the section of t
method), 286	<i>method</i>), 113
printResults() (PAML uncertain Frequent Pattern, basic, T	WHETTWEPPAML partial Periodic Frequent Pattern, basic, GPF growth, GPF gro
method) 279	attribute) 112
nrin+Posults() (PAMI uncertain Frequent Pattern basic I	IFGrowth LIEGrowth
princkesures() (TAMI. uncertaint requent allern. basic. 0	S
meinoa), 290	
printResults() (PAMI.uncertainFrequentPattern.basic.U	Save(3) (PXMI.CorrelatedPattern.basic.CoMine.CoMine
method), 294	method), 40
<pre>printResults() (PAMI.uncertainGeoreferencedFrequentI</pre>	Store & PAME Contract Contra
<i>method</i>), 307	method), 43
<pre>printResults() (PAMI.uncertainPeriodicFrequentPattern</pre>	basic UPFP Growth UPFP Growth basic CMine CMine
method). 298	method) 53
nrintResults() (PAMI uncertainPeriodicFrequentPattern	heriou), 55
method) 303	Save STIT (FAMILOVERAgeranern.ousie:CPFG.CPFG
memory, 505	method), 50 Dattern hemia SWEBGrowth SWEDC routh
princkesuits() (PAMI. weighted Frequentiveightournood)	Save (PAMPI fault Poler and Prequent Pattern.basic.FTApriori.FTApriori
method), 239	method), 46
printResults() (PAMI.weightedFrequentPattern.basic.W	\$#VeW} (PAMI.faultTolerantFrequentPattern.basic.FTFPGrowth.FTFPGro
<i>method</i>), 230	method), 49
<pre>printResults() (PAMI.weightedFrequentRegularPattern.</pre>	sasie WFRIM prenty FRIMingrattern basic Apriori Apriori
method), 234	method) 4
<pre>PrintStats() (PAMI.highUtilityPattern.basic.UPGrowth.</pre>	UBGrowth (PAMI frequentPattern basic ECI AT ECI AT
method) 225	save() (TAMI.Jrequenti utern.basic.ECLAT.ECLAT
PAMI frequent Pattern tonk FAF FAF	meinou), /
method) 24	save() (PAMI.frequentPattern.basic.ECLATbitset.ECLATbitset
meinoa), 24	method), 13
printlree() (in module	<pre>save() (PAMI.frequentPattern.basic.ECLATDiffset.ECLATDiffset</pre>
PAMI.uncertainFrequentPattern.basic.TubeS),	method), 10
287	save() (PAMI.frequentPattern.basic.FPGrowth.FPGrowth
printTree() (in module	method), 16
PAMI.uncertainPeriodicFrequentPattern.basic.UI	ERGrowth Palus) frequent Pattern closed CHARM CHARM
303	method) 10
nrojectTransaction()	memou), 19
(DAML high Utility Spatial Pattern tonk TKSHUIM	Save() (PAMI. frequent Pattern. maximal. MaxFPGrowth. MaxFPGrowth
	Transactiomethod), 21
method), 215	save() (PAMI.frequentPattern.topk.FAE.FAE method),
PSGrowth (class in PAMI.periodicFrequentPattern.basic.PS	Growth), 25
68	<pre>save() (PAMI.fuzzyCorrelatedPattern.basic.FCPGrowth.FCPGrowth</pre>
${\tt PUFGrowth} (class in PAMI.uncertainFrequentPattern.basic.methods) and the second $	PUFGrowthethod), 251
271	save() (PAMI.fuzzyFrequentPattern.basic_FFIMiner_FFIMiner
_	method) 246
К	souch (PAMI fuzzy Coorderon cod Frequent Dattorn basic FESDMiner FESI
removelInpromisingTtems()	Save (I AMII. JULLY OCOTEJETENCEUF TEQUENT FUTURET IL DUSIC. FF SF MIMER. FF SF
(DAMI high Utility, Spatial Dattern tork TVSIIIIIM)	meinoa), 233

(PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Transaction

save()	(PAMI.fuzzyGeoreferencedPeriodicFrequentPattern.sassic(F) method), 264	JPANPMiewindliG@difPMiateed Pattern.basic.EPCPGrowth.EPCPGrowt method), 147
save()	(PAMI.fuzzyPeriodicFrequentPattern.basic.FPFPMbavef()) method) 260	PMM/lep eriodicFrequentPattern.basic.PFECLAT.PFECLAT method) 76
save()	(PAMI.georeferencedFrequentPattern.basic.Spatial Esalt A(),S method) 170	RAMIEGENATIC FrequentPattern.basic.PFPGrowth.PFPGrowth
save()	(PAMI.georeferencedPartialPeriodicPattern.basic.SEEvela.S. method) 179	THEMUL periodicFrequentPattern.basic.PFPGrowthPlus.PFPGrowth method) 68
save()	(PAMI.geoReferencedPeriodicFrequentPattern.basics GAPACE) method), 175	RAM GREFIDM in Fine quent Pattern. basic. PFPMC. PFPMC method), 80
save()	(PAMI.highUtilityFrequentPattern.basic.HUFIM.HERVM() (method), 192	(PAMI.periodicFrequentPattern.basic.PSGrowth.PSGrowth method), 72
save()	(PAMI.highUtilityGeoreferencedFrequentPattern.basion & MU	PAMISHEF And CFrequentPattern.closed.CPFPMiner.CPFPMiner
save()	method), 196 (PAMI.highUtilityPattern.basic.EFIM.EFIM save() (method), 210	method), 84 [PAMI.periodicFrequentPattern.maximal.MaxPFGrowth.MaxPFGi method) 88
save()	(PAMI.highUtilityPattern.basic.HMiner.HMiner save() (method) 222	[PAMI.periodicFrequentPattern.topk.kPFPMiner.kPFPMiner.kPFF method) 91
save()	(PAMI.highUtilityPattern.basic.UPGrowth.UPGrowshve() (method), 225	[PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP.TopkPFPC method), 95
save()	(PAMI.highUtilitySpatialPattern.basic.HDSHUIM.HSDSHUM method), 201	MAMI.recurringPattern.basic.RPGrowth.RPGrowth method), 164
save()	(PAMI.highUtilitySpatialPattern.basic.SHUIM.SHUSMre() (method), 205	[PAMI.relativeFrequentPattern.basic.RSFPGrowth.RSFPGrowth method), 29
save()	(PAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHUM method), 211	PAMI.relativeHighUtilityPattern.basic.RHUIM.RHUIM method), 187
save()	(PAMI.localPeriodicPattern.basic.LPPGrowth.LPPGaue()) method), 99	PAMI.sequentialPatternMining.basic.prefixSpan.prefixSpan method), 323
save()	(PAMI.localPeriodicPattern.basic.LPPMBreadth.LPSAMED() d method), 105	HAMI .sequentialPatternMining.basic.SPADE.SPADE method), 314
save()	(PAMI.localPeriodicPattern.basic.LPPMDepth.LPP MDepth method), 109	PAMI.sequentialPatternMining.basic.SPAM.SPAM method), 318
save()	(PAMI.multipleMinimumSupportBasedFrequentPattsan.b()) method), 33	PANH Gudole P. G. F. R. Guloret J uent Pattern. basic. SPPE clat. SPPE clat method), 155
save()	(PAMI.multipleMinimumSupportBasedFrequentPattsan.b()) method), 37	PANH Guaddel Pelius di diffe que nul Haltus rn. basic. SPPG rowth. SPPG rowt method), 151
save()	(PAMI.partialPeriodicFrequentPattern.basic.GPFgsawb.QK method), 112	Pf3Miwsh ablePeriodicFrequentPattern.topK.TSPIN.TSPIN method), 159
save()	(PAMI.partialPeriodicFrequentPattern.basic.PPF_ BESERD method), 119	PPMS .uncertainFrequentPattern.basic.CUFPTree.CUFPTree method), 271
save()	(PAMI.partialPeriodicPattern.basic.GThreePGrowt bacyEl (ret method), 131	PAMIwh certainFrequentPattern.basic.PUFGrowth.PUFGrowth method), 275
save()	(PAMI.partialPeriodicPattern.basic.PPP_ECLAT.P BR v E QU method), 127	RX MI.uncertainFrequentPattern.basic.TubeP.TUFP method), 282
save()	(PAMI.partialPeriodicPattern.basic.PPPGrowth.PPSCiec())th method), 123	PAMI.uncertainFrequentPattern.basic.TubeS.TubeS method), 286
save()	(PAMI.partialPeriodicPattern.closed.PPPClose.PPB6/wese) (method), 135	PAMI.uncertainFrequentPattern.basic.TUFP.TUFP method), 279
save()	(PAMI.partialPeriodicPattern.maximal.Max3PGrovshyld();3 method), 139	PSMI with certain Frequent Pattern. basic. UFG rowth. UFG rowth method), 290
save()	(PAMI.partialPeriodicPattern.topk.k3PMiner.k3PMiseve() (method) 142	(PAMI.uncertainFrequentPattern.basic.UVECLAT.UVEclat method) 294
save()	(PAMI.partialPeriodicPatternInMultipleTimeSeries. SPRG for method), 330	(PAMPGroweth ainGeoreferencedFrequentPattern.basic.GFPGrowth. method), 307

save() (PAMI.uncertainPeriodicFrequentPattern.basic.UPFP	GrowthmLeHoElGreewth
<i>method</i>), 298 sta	artMine() (PAMI.faultTolerantFrequentPattern.basic.FTFPGrowth.FTI
<pre>save() (PAMI.uncertainPeriodicFrequentPattern.basic.UPFP</pre>	Growth PlathddP,FP GrowthPlus
method), 303	artMine() (PAMI.frequentPattern.basic.Apriori.Apriori
<pre>save() (PAMI.weightedFrequentNeighbourhoodPattern.basic.</pre>	SWFPGnenhthd S,WFPGrowth
method), 239 sta	artMine() (PAMI.frequentPattern.basic.ECLAT.ECLAT
save() (PAMI.weightedFrequentPattern.basic.WFIM.WFIM	method). 8
method), 230 st	artMine() (PAML frequent Pattern, basic, ECLAT bitset, ECLAT bitset
save() (PAML weighted Frequent Regular Pattern basic, WFRIM	liner. Wfm Rill/ bither13
method). 234	artMine() (PAMI frequentPattern basic ECLATDiffset ECLATDiffset
Second() (in module PAMI uncertainFrequentPattern basic Tu	bes) method) 10
283 st	artMine() (PAMI frequentPattern basic FPGrowth FPGrowth
sen (PAMI highUtilitySpatialPattern tonk TKSHUIM TKSHUI	M method) 16
attribute) 211	artMino() (PAMI frequent Pattern closed CHAPM CHAPM
conchesting () (PAMI sequential Pattern Mining basic prefix Spa	n prefixentend) 10
method) 222	n. presinage and u), 19
memoa), 525 SL	ar Linne() (PAMI, JrequentPattern.maximat.maxr PGrowin.maxr PGrow
SHUF IM (class in PAMI.nignUtilityGeoreJerenceaFrequentPatte	
$\frac{192}{192}$	artMine() (PAMI.frequentPattern.topk.FAE.FAE
SHUIM (class in PAMI.highUtilitySpatialPattern.basic.SHUIM),	method), 25
201 sta	artMine() (PAMI.fuzzyCorrelatedPattern.basic.FCPGrowth.FCPGrowt
<pre>sort_transaction() (PAMI.highUtilityPattern.basic.EFIM.</pre>	EFIM method), 251
method), 219 st	artMine() (PAMI.fuzzyFrequentPattern.basic.FFIMiner.FFIMiner
<pre>sort_transaction() (PAMI.highUtilitySpatialPattern.topk.T</pre>	KSHUHMATTEKKSHTLAKM
<i>method</i>), 212 sta	artMine() (PAMI.fuzzyGeoreferencedFrequentPattern.basic.FFSPMine
<pre>sort_transaction() (PAMI.relativeHighUtilityPattern.basic</pre>	r.RHUIMeRHUUM255
<i>method</i>), 187 sta	artMine() (PAMI.fuzzyGeoreferencedPeriodicFrequentPattern.basic.FC
<pre>sortDatabase() (PAMI.highUtilitySpatialPattern.topk.TKSH</pre>	UIM.TK&HUJM 264
<i>method</i>), 211 sta	artMine() (PAMI.fuzzyPeriodicFrequentPattern.basic.FPFPMiner.FPF
<pre>sortDatabase() (PAMI.relativeHighUtilityPattern.basic.RHU</pre>	JIM.RHuelmod), 260
method), 187 sta	artMine() (PAMI.georeferencedFrequentPattern.basic.SpatialECLAT.Spatia
SPADE (class in PAMI.sequentialPatternMining.basic.SPADE).	method). 171
310 st	artMine() (PAMI.georeferencedPartialPeriodicPattern.basic.STEclat.S
SPAM (class in PAMI sequentialPatternMining basic SPAM)	method) 179
314 st	artMine() (PAMI geoReferencedPeriodicFrequentPattern basic GPFPN
SnatialECLAT (class in	method) 175
PAMI georaforenced Frequent Pattern basic Spatial	memou), 175
	method) 102
SPDE al at (alaga in DAMI at able Danie die Energy aut Dattern harit	memou), 192 SDME where (DAMI high Ittility Constants and Enservent Pattorn havin SIIII
SPPECIAL (class in PAMI.studiePerioalCFrequentPattern.bassc)	artificter () (PAMI.nignOlliliyGeorejerenceur requentraliern.busic.SHOL
	method), 197
SPPGrowth (class in PAMI.stablePeriodicFrequentPattern.bastic	a. Stringeowith AMI. high Utility Pattern. basic. EFIM. EFIM
147	method), 219
SPPList (PAMI.stablePeriodicFrequentPattern.basic.SPPGsoc	ath MR Ref MAMI.highUtilityPattern.basic.HMiner.HMiner
attribute), 151	<i>method</i>), 222
Sstep() (PAMI.sequentialPatternMining.basic.SPAM.SPAMst	artMine() (PAMI.highUtilityPattern.basic.UPGrowth.UPGrowth
<i>method</i>), 317	<i>method</i>), 226
<pre>startMine()(PAMI.correlatedPattern.basic.CoMine.CoMister</pre>	artMine() (PAMI.highUtilitySpatialPattern.basic.HDSHUIM.HDSHUI
method), 40	<i>method</i>), 201
<pre>startMine()(PAMI.correlatedPattern.basic.CoMinePlus.Gat</pre>	MineRhos() (PAMI.highUtilitySpatialPattern.basic.SHUIM.SHUIM
method), 43	<i>method</i>), 205
<pre>startMine() (PAMI.coveragePattern.basic.CMine.CMine startMine()</pre>	artMine() (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHUIM
method), 53	method), 212
<pre>startMine() (PAMI.coveragePattern.basic.CPPG.CPPG startMine()</pre>	artMine() (PAMI.localPeriodicPattern.basic.LPPGrowth.LPPGrowth
method), 56	method). 100
startMine() (PAML faultTolerantFrequentPattern basic FRan	artaNiKEADuBAMLlocalPeriodicPattern.basic LPPMRreadth LPPMRrea
	g, the antice can be required another based in the required in the first of the fir

<i>method</i>), 105	<i>method</i>), 159
startMine()(PAMI.localPeriodicPattern.basic.LPPMDep st&PPM	IDefyh(PAMI.uncertainFrequentPattern.basic.CUFPTree.CUFPTre
method), 109	method), 271
startMine()(PAMI.multipleMinimumSupportBasedFrequentAntMi	mb(1)i(PANHGineethaGilFPegreentleattern.basic.PUFGrowth.PUFGro
method), 33	method), 275
startMine()(PAMI.multipleMinimumSupportBasedFrequentAptMi	mba)icPAMHanoethRhEseChiPattenPalbasic.TubeP.TUFP
method). 37	method), 283
startMine() (PAMI nartialPeriodicPattern basic PPP_EC& ########	PhefOM(PATMI uncertainFrequentPattern basic TUFP TUFP
method) 127	method) 279
startMine() (PAML partial Periodic Pattern basic, PPPG rosta PPP	GenWthPAML uncertainGeoreferencedFrequentPattern basic GFPG
method) 123	method) 308
startMine() (PAMI nartialPeriodicPattern closed PPPClast and the start Mine() (PAMI nartialPeriodicPattern closed PPPClast and the start and t	(PAMI uncertainPeriodicFrequentPattern basic UPFPGrowt
method) 135	method) 298
startMino() (PAMI nartialPariodicPattern tonk k3PMinardt3PMM	memore), 200
method) 142	method) 202
memou), 142	memoa), 505
startmine() (PAMI.partialPerioaicPatterninMultiple1imeseaes.m	Harow have no weight ear requentive ignoournoo a Pattern. basic. Sw F P
methoa), 330	methoa), 239
startMine() (PAMI.periodicCorrelatedPattern.basic.EPCBtaptM	Lift (OR Exploring the definition of the second sec
method), 14/	method), 230
startMine()(PAMI.periodicFrequentPattern.basic.PFECIstartM	GEAY (PAMI.weightedFrequentRegularPattern.basic.WFRIMiner.V
method), 77	<i>method</i>), 235
startMine()(PAMI.periodicFrequentPattern.basic.PFPG rsnabtPI	RE(BAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHUIM
<i>method</i>), 64	attribute), 212
startMine()(<i>PAMI.periodicFrequentPattern.basic.PFPG</i> STEbPa	ts(ElERGmdRAthH lgsoreferencedPartialPeriodicPattern.basic.STEclat
method), 68	175
startMine()(<i>PAMI.periodicFrequentPattern.basic.PFPM</i> 6tPFBM	${f t} {f t} $
method) 80	attributa) 212
memoa), 80	
startMine() (PAMI.periodicFrequentPattern.basic.PSGro SWE.PSfc	workthclass in PAMI.weightedFrequentNeighbourhoodPattern.basic.
startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPSC method), 72	www.hclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235
startMine() (PAMI.periodicFrequentPattern.basic.PSGro SWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFP <u>M</u> iner.C	www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 PFPMiner
startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84	worldhclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 PFPMiner
startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWF.PSG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxBfG(P)	www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 PFPMiner WMINIWIRFICingSplatialPattern.topk.TKSHUIM.TKSHUIM
startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPSG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEmG(P), method), 88	www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 PFPMiner WMIMigHPfflingSplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212
startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWFPSG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxBfS(PA method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinidTable)	annoule, 212 www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 PFPMiner wMINItgHUMIngSplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 RMinGNKIPHIMIngeragePattern.basic.CMine.CMine
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEmG(PM method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMiniedTobE method), 91</pre>	www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 PFPMiner WMINIgHPffingSplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 RMin(i)k[PARMineveragePattern.basic.CMine.CMine method), 53
startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEmG(PA method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMiniedTobE method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFPKEWEP	 annoule, 212 annoule, 212 annoule, 212 annoule, 213 PFPMiner Annoule, 212 Restance (PARMinewerage Pattern.basic.CMine.CMine method), 53 FRIONER PRONOW th Utility Spatial Pattern topk TKSHUIM)
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEEG(PA method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMiniedToBE method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFTKEWET method), 95</pre>	www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 PFPMiner WMIMigHUGingSplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 RMin(PARMineveragePattern.basic.CMine.CMine method), 53 FCETagskIPPRMOWIghUtilitySpatialPattern.topk.TKSHUIM), 206
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEmG(PA method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinicdTobE method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFTKEWET method), 95 startMine() (PAMI.recurringPattern basic RPGrowth RPGrowtFET</pre>	www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 PFPMiner WMINig/REfilingSplutialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 RMin()/(PARMineveragePattern.basic.CMine.CMine method), 53 FC:RayskiPPRONOWIGhUtilitySpatialPattern.topk.TKSHUIM), 206 PGrowth (class in
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEmG(PA method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinicdTdB method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFTXSuuTM method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGopkPFT method), 164</pre>	withoute, 212 www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 'PFPMiner wMINighTiGingSplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 'RMin()/(PARMineveragePattern.basic.CMine.CMine method), 53 'FCRayshi PRONOWighUtilitySpatialPattern.topk.TKSHUIM), 206 'Growth (class 'PAMI.periodicFrequentPattern
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxBEG(PA method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinidTdBE method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFTKSWETM method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGopkPFT method), 164 startMine() (PAMI.relativeFrequentPattern basic RSEPGrowth RS</pre>	www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 PFPMiner WMINigHUiGingSphatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 RMin(t)kIPARMineveragePattern.basic.CMine.CMine method), 53 FREMoveBhUtilitySpatialPattern.topk.TKSHUIM), 206 PGrowth (class in PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP), SEPGrowth
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxBEG(PM method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinicdTdB method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFTKSWETM method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGopkPFT method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 164</pre>	annoule, 212 www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 PFPMiner WMINigHPfGingSplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 RMin(r)k[PARMineweragePattern.basic.CMine.CMine method), 53 FGERayskIPFRMowighUtilitySpatialPattern.topk.TKSHUIM), 206 PGrowth (class in PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP), SEP2Growth class in
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEmG(PAMI.periodicFrequentPattern.topk.kPFPMinedTdF method), 88 startMine() (PAMI.periodicFrequentPattern.topk.tPFPMinedTdF method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFTKSubT method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGopkPFT method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS</pre>	announe, 212 www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 PFPMiner whiNighthfhingsplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 RMineweragePattern.basic.CMine.CMine method), 53 FREMoverful UtilitySpatialPattern.topk.TKSHUIM), 206 PGrowth (class PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP), SFP2Growth ction (class und (class und (class und (class SFP2Growth (class Ction (class Communication (class Class in
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEmG(PAMI.periodicFrequentPattern.topk.kPFPMinedTkdF method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinedTkdF method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFTKSWETM method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGopkFFF method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RHUIM.RHU method), 29 startMine() (PAMI.relativeHighUtilityPattern.basic.RHUIM.RHU method), 187</pre>	withbule, 212 withclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 "PFPMiner willNigiRFGingSplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 RSkind)k[PARMineweragePattern.basic.CMine.CMine method), 53 FRERMANNIGHUtilitySpatialPattern.topk.TKSHUIM), 206 OGrowth (class PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP), SFP2Growth Ction (class IMAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEEG(PA method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinedTkdE method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFTKSWET method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGopkPFT method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RFFGrowth.RS method), 187</pre>	announe, 212 www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 "PFPMiner whiMightfiftingSplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 "RSin(i)k[PARMineveragePattern.basic.CMine.CMine method), 53 FRETagskiPPRMonvighUtilitySpatialPattern.topk.TKSHUIM), 206 "Growth (class in PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP), SFP2Growth Ction (class in IMAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEmG(PAMI.periodicFrequentPattern.topk.kPFPMinidTkdB method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinidTkdB method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFTKSWEFT method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGopkFFT method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeHighUtilityPattern.basic.RHUIM.RHU method), 187 startMine() (PAMI.sequentialPatternMining.basic.prefixSpanned method), 187</pre>	announe, 212 www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 "PFPMiner whiNightfiCingSplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 "RSin(r)k(PARMineweragePattern.basic.CMine.CMine method), 53 "FcRapskitPROMOWIGhUtilitySpatialPattern.topk.TKSHUIM), 206 "Growth (class in PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP), SPP_Growth "Ction (class in UMAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214 "tSpens (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Dataset
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEmG(PAMI.periodicFrequentPattern.topk.kPFPMinedTable method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinedTable method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFTKSINTM method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGookPFT method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeHighUtilityPattern.basic.RHUIM.RHU method), 187 startMine() (PAMI.sequentialPatternMining.basic.prefixSpransa method), 323</pre>	announe, 212 www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 "PFPMiner whiNightficingsplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 "RMin()k(PARMineweragePattern.basic.CMine.CMine method), 53 "F@Rayshif PROMOWIShUtilitySpatialPattern.topk.TKSHUIM), 206 "Growth (class in PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP), SP_Growth "Ction (class in IMAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214 "KSpons (PAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214 "KSpons (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Dataset attribute), 206
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEmG(PA method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinedTdB method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFTXSMUTM method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGopkPFT method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 187 startMine() (PAMI.relativeHighUtilityPattern.basic.PrefixSpansal method), 187 startMine() (PAMI.sequentialPatternMining.basic.prefixSpansal method), 323</pre>	announe, 212 www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 "PFPMiner whiNityHUfflingsplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 "RMin(n)K[PARMineveragePattern.basic.CMine.CMine method), 53 "FERMINGNUMUMUNUTURITYSpatialPattern.topk.TKSHUIM), 206 "Growth (class in PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP), SFP2Growth "Ction (class in INFAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214 "Sports (PAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214 "Statistic (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Dataset attribute), 206 "ass in PAMI.localPeriodicPattern.basic.LPPGrowth), 100
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEmG(PA method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinicdTdB method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFTXSuUTM method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGookPFT method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 187 startMine() (PAMI.relativeHighUtilityPattern.basic.RHUIM.RHU method), 187 startMine() (PAMI.sequentialPatternMining.basic.prefixSpannad method), 323</pre>	announe, 212 www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 "PFPMiner wmiNitglfUfflingsplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 "RMindy/RFMMineveragePattern.basic.CMine.CMine method), 53 "Fellayshif PRMMowighUtilitySpatialPattern.topk.TKSHUIM), 206 "Growth (class in PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP), SFP2Growth "Ction (class in IMAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214 "tSpens (PAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214 "tSpens (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Dataset attribute), 206 "ass in PAMI.localPeriodicPattern.basic.LPPGrowth), 100
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEEG(PA method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMiniedToFF method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFTKSWETM method), 91 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGoopkPFF method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 187 startMine() (PAMI.relativeHighUtilityPattern.basic.RHUIM.RHU method), 187 startMine() (PAMI.sequentialPatternMining.basic.prefixSpannad method), 323 startMine() (PAMI.sequentialPatternMining.basic.SPADESEACME method), 314 startMine() (PAMI.sequentialPatternMining.basic.SPAM:SPAM(Clate))</pre>	announe, 212 www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 'PFPMiner wmiNitgHUfflingSphitialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 'RMindy/PARMineweragePattern.basic.CMine.CMine method), 53 'Ffellagshif PROMOWIGHUtilitySpatialPattern.topk.TKSHUIM), 206 'OG 'OG 'PGrowth class 'PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP), SFP2Growth 'Ction (class 'IMAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214 'CSports (PAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214 'StSports (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Dataset attribute), 206 'ass in PAMI.localPeriodicPattern.basic.LPPGrowth), 100 ass in PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth),
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxBEG(PA method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinidTdBE method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFFKSHBEF method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGookPFF method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 187 startMine() (PAMI.relativeHighUtilityPattern.basic.RHUIM.RHU method), 187 startMine() (PAMI.sequentialPatternMining.basic.prefixSpansal method), 323 startMine() (PAMI.sequentialPatternMining.basic.SPADESEACH method), 314 startMine() (PAMI.sequentialPatternMining.basic.SPAM:SFEACH method), 318</pre>	announe, 212 www.thclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 "PFPMiner whilWighthingsphatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 RMan(a)k(PARMineweragePattern.basic.CMine.CMine method), 53 Foreward Pattern.basic.CMine.CMine method), 53 Foreward Pattern.topk.TKSHUIM), 206 Ogrowth (class in PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP), SFP2Growth Ction (class in IMAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214 itSpons (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Dataset attribute), 206 ass in PAMI.localPeriodicPattern.basic.LPPGrowth), 100 ass in PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth), 113
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPGG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEEG(PM method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinidTdBE method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFPKSWEP method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGopkPFF method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 187 startMine() (PAMI.relativeHighUtilityPattern.basic.RHUIM.RHU method), 187 startMine() (PAMI.sequentialPatternMining.basic.prefixSpransa method), 323 startMine() (PAMI.sequentialPatternMining.basic.SPADEtSPACE method), 314 startMine() (PAMI.sequentialPatternMining.basic.SPAM:SPEACE method), 318 startMine() (PAMI.stablePeriodicFrequentPattern.basic.SFSPEN(cd))</pre>	announe), 212 workth(class in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 'PFPMiner WMINigHTrfingSplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 'RMin()(PARMLineweragePattern.basic.CMine.CMine method), 53 'F(RasskRPRMMineweragePattern.basic.CMine.CMine 'Soft 'Soft 'Stoms (requentPattern.topk.TKSHUIM), '214 'Soft 'Soft 'Soft 'attribute), 206 'ass in PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Dataset attribute), 206 'ass in PAMI.localPeriodicPattern.basic.LPPGrowth), '100 ass in PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth), '113 'BSPEMEMAMI.stablePeriodicFrequentPattern.topK.TSPIN),
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPSG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxBhGGP method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinedTdB method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFFKSuptP method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFFKSuptP method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGoopkFFF method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RFU method), 187 startMine() (PAMI.sequentialPatternMining.basic.prefixSpansad method), 323 startMine() (PAMI.sequentialPatternMining.basic.SPADETSPECted method), 314 startMine() (PAMI.sequentialPatternMining.basic.SPAM.SFECTED method), 318 startMine() (PAMI.sequentialPatternMining.basic.SPAM.SFECTED method), 318</pre>	announe), 212 workth(class in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 "PFPMiner whilNigifFffingSplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 "RMin() (PARMLineweragePattern.basic.CMine.CMine method), 53 "FfeRmskifFffingSplatialPattern.basic.CMine.CMine method), 53 "FfeRmskifFffingSplatialPattern.basic.CMine.CMine method), 53 "FfeRmskifFffingSplatialPattern.basic.CMine.CMine method), 53 "FfeRmskifFffingSplatialPattern.topk.TKSHUIM), 206 "PGrowth (class in PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP), SPPGrowth "ction (class in WMAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214 "XSpens (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Dataset attribute), 206 Sass in PAMI.localPeriodicPattern.basic.LPPGrowth), 100 ass in PAMI.partialPeriodicFrequentPattern.topK.TSPIN), 113 "WRSPERFMAMI.stablePeriodicFrequentPattern.topK.TSPIN), 156
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPSG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxBhGGP method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMinedTdB method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFFKSubT method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFFKSubT method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGogNEFF method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RHU method), 187 startMine() (PAMI.sequentialPatternMining.basic.prefixSpansa method), 323 startMine() (PAMI.sequentialPatternMining.basic.SPADE.SPACE method), 314 startMine() (PAMI.sequentialPatternMining.basic.SPAM.SFEA(cl. method), 318 startMine() (PAMI.stablePeriodicFrequentPattern.basic.SFtPEFF(cl. method), 155 startMine() (PAMI.stablePeriodicFrequentPattern.basic.SFtPEFEFFEACE) startMine() (PAMI.stablePeriodicFrequentPattern.basic.SFtPEFEACE) method), 155</pre>	announe), 212 workitkclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 PFPMiner whiNitgitffilingSplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 Reference Refere
<pre>startMine() (PAMI.periodicFrequentPattern.basic.PSGroSWEPSG method), 72 startMine() (PAMI.periodicFrequentPattern.closed.CPFPMiner.C method), 84 startMine() (PAMI.periodicFrequentPattern.maximal.MaxEfford method), 88 startMine() (PAMI.periodicFrequentPattern.topk.kPFPMiniaTidf method), 91 startMine() (PAMI.periodicFrequentPattern.topk.TopkPFFKSWEFF method), 95 startMine() (PAMI.recurringPattern.basic.RPGrowth.RPGrowthFF method), 164 startMine() (PAMI.relativeFrequentPattern.basic.RSFPGrowth.RS method), 29 startMine() (PAMI.relativeFrequentPattern.basic.RHUIM.RHU method), 187 startMine() (PAMI.relativeHighUtilityPattern.basic.RHUIM.RHU method), 187 startMine() (PAMI.sequentialPatternMining.basic.prefixSpannsa method), 323 startMine() (PAMI.sequentialPatternMining.basic.SPADEfSEA(22 method), 314 startMine() (PAMI.stablePeriodicFrequentPattern.basic.SPAM.SFEA(cl method), 155 startMine() (PAMI.stablePeriodicFrequentPattern.basic.SPAMESEA(cl method), 155</pre>	anihome), 212 workfikiclass in PAMI.weightedFrequentNeighbourhoodPattern.basic. 235 PFPMiner whiMigiRFitingSplatialPattern.topk.TKSHUIM.TKSHUIM attribute), 212 Reserve agePattern.basic.CMine.CMine method), 53 FiteRaskR PRMMemeragePattern.basic.CMine.CMine method), 53 FiteRaskR PRMMemetragePattern.basic.CMine.CMine method), 53 FiteRaskR PRMMemetragePattern.basic.CMine.CMine method), 53 FiteRaskR PRMMemetragePattern.basic.CMine.CMine method), 53 FiteRaskR PRMMemetragePattern.basic.CMine.CMine method), 53 FiteRaskR PRMMemetragePattern.topk.TKSHUIM), 206 OG Paron (class in PAMI.periodicFrequentPattern.topk.TopkPFP.TopkPFP), SFP2Growth Ction (class in IMAMI.highUtilitySpatialPattern.topk.TKSHUIM), 214 Stass in PAMI.highUtilitySpatialPattern.topk.TKSHUIM.Dataset attribute), 206 Sass in PAMI.localPeriodicPattern.basic.LPPGrowth), 100 ass in PAMI.partialPeriodicFrequentPattern.basic.GPFgrowth), 113 StassRFEATMI.stablePeriodicFrequentPattern.topK.TSPIN), 156 StassRFPATMI.funcertainFrequentPattern.basic.TubeS), 283
TUFP (class in PAMI.uncertainFrequentPattern.basic.TubeP), 279 TUFP (class in PAMI.uncertainFrequentPattern.basic.TUFP), 275

U

UFGrowth (class in PAMI.uncertainFrequentPattern.basic.UFGrowth), 287 updateTransactions() (PAMI.uncertainFrequentPattern.basic.TubeS.TubeS method), 287 UPFPGrowth (class in PAMI.uncertainPeriodicFrequentPattern.basic.UPFPGrowth), 294 **UPFPGrowthPlus** (class in PAMI.uncertainPeriodicFrequentPattern.basic.UPFPGrowthPlus), 299 UPGrowth (class in PAMI.highUtilityPattern.basic.UPGrowth), 223 useUtilityBinArraysToCalculateUpperBounds() (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHUIM method), 214 useUtilityBinArrayToCalculateLocalUtilityFirstTime() (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHUIM method). 214 useUtilityBinArrayToCalculateSubtreeUtilityFirstTime() (PAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHUIM method), 214 utilityBinArrayLU(PAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHUIM attribute), 214 utilityBinArraySU(PAMI.highUtilitySpatialPattern.topk.TKSHUIM.TKSHUIM attribute), 214 UVEclat (class in PAMI.uncertainFrequentPattern.basic.UVECLAT),

290

W

WFIM(class in PAMI.weightedFrequentPattern.basic.WFIM), 226 WFRIMiner(class in PAMI.weightedFrequentRegularPattern.basic.WFRIMiner), 231