LIsp-Miner installation

- Download [LIsp-Miner.Core.zip](https://lispminer.vse.cz/download/index.html), which is a windows version software.
- Install LIsp-Miner on Windows operation system.
- If you use Mac:
  1. Install [parallels desktop](https://www.parallels.com/de/usa/products/desktop/de/) or [virtualbox](https://www.virtualbox.org/)
  2. Install [windows operation system](https://www.microsoft.com/de-de/)
  3. Install LIsp-Miner.
Dealing with missing values

In LISp-Miner, there are three ways how to handle missing values
- Deleting – missing values are ignored, they are not involved in the algorithm
- Optimistic – missing values support the relationship we are interested in
- Secured – missing values do not support the relationship we are interested in
For each attribute, we should specify which set of literals will be created. This definition is determined by:

1. Minimal and maximal **length of a literal**.
2. **The type of coefficient** – subsets, intervals, cyclical intervals, left cuts, right cuts, cuts, one particular value
3. One from the following **options**:
   - Generate only positive literals – no literals with negation are created
   - Generate only negative literals – only literals with negation are created
   - Generate both positive and negative literals
The type of coefficient

- Subsets
- Intervals
- Cyclical intervals
- Left cuts
- Right cuts
- Cuts
- One particular value
Subsets.

Creation of all possible combinations of categories of the defined length (the order does not matter)

**Example:** Create literals of the attribute Type of therapy with its categories \{diet, medicaments, operation, none\} with minimal length 1 and maximal length 2: Type of therapy (diet), Type of therapy (medicaments), Type of therapy (operation), Type of therapy (none), Type of therapy (diet, medicaments), Type of therapy (diet, operation), Type of therapy (diet, none), Type of therapy (medicaments, operation), Type of therapy (medicaments, none), Type of therapy (operation, none)
Intervals.

Sequences of the defined length are created.

Example: Create literals of the attribute Age with its categories \{(20; 30), (30; 40), (40; 50), (50; 60), (60; 70)\} with minimal length 2 and maximal length 3.

Age [(20; 30), (30; 40)], Age [(30; 40), (40; 50)], Age [(40; 50), (50; 60)], Age [(50; 60), (60; 70)], Age [(20; 30), (30; 40), (40; 50)], Age [(30; 40), (40; 50), (50; 60)], Age [(40; 50), (50; 60), (60; 70)].
Cyclical intervals.

Sequences of the defined length are created, cycles are permitted. **Example:** Create literals of the attribute Day with its categories \{sun, mo, tue, we, thu, fri, sat\} with minimal length 3 and maximal length 4.

Day (sun, mo, tue), Day (mo, tue, we), Day (tue, we, thu), Day (we, thu, fri), Day (thu, fri, sat), Day (fri, sat, sun), Day (sat, sun, mo), Day (sun, mo, tue, we), Day (mo, tue, we, thu), Day (tue, we, thu, fri), Day (we, thu, fri, sat), Day (thu, fri, sat, sun), Day (fri, sat, sun, mo), day (sat, sun, mo, tue).
Left cuts.

Sequences containing only the first category are created.

**Example:** Create literals of the attribute Age with its categories \{(20; 30), (30; 40), (40; 50), (50; 60), (60; 70)\} with maximal length 4 (minimal length is by default 1).

Age \( (20; 30), \)

Age \([ (20; 30), (30; 40) ], \)

Age \([ [ (20; 30), (30; 40), (40; 50) ], \)

Age \([ (20; 30), (30; 40), (40; 50), (50; 60) ]]. \)
Right cuts.

Sequences containing only the last category are created.
Example: Create literals of the attribute Age with its categories \{(20; 30), (30; 40),
(40; 50), (50; 60), (60; 70)\} with maximal length 4 (minimal length is by default 1).
Age \((60; 70)\),

Age \([(60; 70), (50; 60)]\),

Age \([(60; 70), (50; 60), (40; 50)]\),

Age \([(60; 70), (50; 60), (40; 50), (30; 40)]\).
One particular value.

Only one literal with a particular category will be used.

**Example:** Create literal of the attribute Type of therapy with its category \{diet, medicaments, operation, none\} containing only operation.

Type of therapy (operation).
• A => B, where A (called antecedent) and B (called consequent) are sets of items.
R: \((A_1 = \omega_1) \land \ldots \land (A_Q = \omega_Q) \land (B_1, \alpha_1 \rightarrow \beta_1) \land \ldots \land (B_P, \alpha_P \rightarrow \beta_P) \Rightarrow (D, k_1 \rightarrow k_2)\)

- Where \((A_1, \ldots, A_Q)\) are stable attributes
- \((\omega_1, \ldots, \omega_Q)\) are values of stable attributes \((A_1, \ldots, A_Q)\)
- \(\{B_1, \ldots, B_P\}\) are flexible attributes
- \(\{(\alpha_1 \rightarrow \beta_1), \ldots, (\alpha_P \rightarrow \beta_P)\}\) are changes of values of flexible attributes \(\{B_1, \ldots, B_P\}\)
- \(D\) is a decision
- \((k_1 \rightarrow k_2)\) is a change of decision from \(k_1\) to \(k_2\)
CPL(R)...number of objects matching \((\omega_1, \ldots, \omega_Q, \alpha_1, \ldots, \alpha_p, k_1)\), i.e. number of objects matching the state before a change which also match the state of decision before the change

CPR(R): number of objects matching \((\omega_1, \ldots, \omega_Q, \beta_1, \ldots, \beta_p, k_2)\), i.e. number of objects matching the state after a change which also match the state of decision after the change

CVL(R): number of objects matching \((\omega_1, \ldots, \omega_Q, \alpha_1, \ldots, \alpha_p)\), i.e. number of all objects matching the state before a change

CVR(R): number of objects matching \((\omega_1, \ldots, \omega_Q, \beta_1, \ldots, \beta_p)\), i.e. number of all objects matching the state after a change

LeftSup(R) = \(\frac{CPL(R)}{n}\), where \(n\) is a total number of objects in the database

RightSup(R) = \(\frac{CPR(R)}{n}\)
<table>
<thead>
<tr>
<th>Nr.</th>
<th>Hypothesis</th>
<th>D%-Sum</th>
<th>Df-Conf</th>
<th>Df-AFUI</th>
<th>Df-FUE</th>
<th>Df-Avg</th>
<th>R-Conf</th>
<th>R-DFUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-FUE</td>
<td>R-Avg</td>
<td>H-Conf</td>
<td>H-DFUI</td>
<td>H-FUE</td>
<td>H-Avg</td>
<td>B:a</td>
<td>B:b</td>
<td></td>
</tr>
<tr>
<td>B:c</td>
<td>B:d</td>
<td>B:r</td>
<td>B:n</td>
<td>B:Conf</td>
<td>B:DFUI</td>
<td>B:EConf</td>
<td>B:Supp</td>
<td></td>
</tr>
<tr>
<td>B:Complete</td>
<td>B:AvgDF</td>
<td>B:Lift</td>
<td>B:LB</td>
<td>B:UBound</td>
<td>B:ELBound</td>
<td>B:EUBound</td>
<td>B:DLBound</td>
<td></td>
</tr>
<tr>
<td>B:DUBound</td>
<td>B:Fisher</td>
<td>B:Chi-Sq</td>
<td>B:PSep</td>
<td>B:bMean</td>
<td>B:bVAR</td>
<td>B:bStDev</td>
<td>B:P(&gt;=95%)</td>
<td></td>
</tr>
<tr>
<td>B:P(&gt;=95%)</td>
<td>A:a</td>
<td>A:b</td>
<td>A:c</td>
<td>A:d</td>
<td>A:r</td>
<td>A:n</td>
<td>A:Conf</td>
<td></td>
</tr>
<tr>
<td>A:bVAR</td>
<td>A:bStDev</td>
<td>A:P(&gt;=90%)</td>
<td>A:P(&gt;=95%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

255

Survey Type(Field) : (Benchmark__Service_Tech_Equipped_to_do_Job(9) & Benchmark__Service_Tech_Promised_in_Expected_Timeframe(9) -> Benchmark__Service_Tech_Equipped_to_do_Job(10) & Benchmark__Service_Tech_Promised_in_Expected_Timeframe(10)) => (CustomerStatus(Active) -> CustomerStatus(Leaving))

1.42

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9104716857</td>
<td>-0.0087952481</td>
<td>-0.6194382490</td>
<td>0.6520948122</td>
<td>40.8252873563</td>
<td>0.4668259929</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2874103066</td>
<td>-0.5644572759</td>
<td>40.8252873563</td>
<td>2.1421257923</td>
<td>3.4793463459</td>
<td>1.7716132693</td>
<td></td>
<td></td>
</tr>
<tr>
<td>154</td>
<td>11</td>
<td>19833</td>
<td>6455</td>
<td>165</td>
<td>26453</td>
<td>3.0333333333</td>
<td>0.0077007701</td>
</tr>
<tr>
<td>0.2498393377</td>
<td>0.0058216459</td>
<td>0.0077050083</td>
<td>0.2352762629</td>
<td>1.2352762629</td>
<td>0.9921864586</td>
<td>0.9473117288</td>
<td>0.9281437126</td>
</tr>
<tr>
<td>0.9999999999</td>
<td>0.0000000017</td>
<td>28.4098845456</td>
<td>127.8571428571</td>
<td>0.9281437126</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9281437126</td>
<td>0.019924405</td>
<td>0.9117630263</td>
<td>0.1291964958</td>
<td>58</td>
<td>2479</td>
<td>979</td>
<td>22937</td>
</tr>
</tbody>
</table>