

# Classification

## Exercise 5



# Naïve Bayes Classification

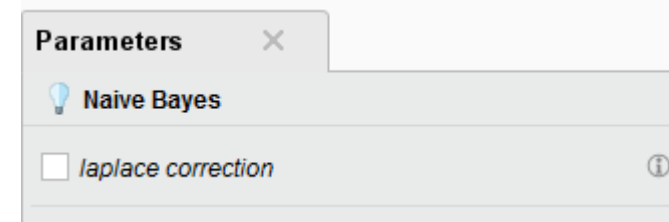
- If we know the prior probability and the likelihood
  - which we can estimate from the data
- Then we can calculate the posterior probability
  - Which we use for classification

$$P(C|A) = \frac{P(A|C)P(C)}{P(A)}$$

- Prior Probability
  - $P(A)$ ,  $P(C)$  “35.7% chance of rain”, “64.3% chance of play golf”
- Likelihood, given an observation
  - $P(A|C)$  “33% chance of rain if go play golf”
- Posterior Probability
  - $P(C|A)$  “66.7% chance of play golf if no rain”

# Operators: Naïve Bayes

- Input
  - Training data (Example Set)
- Output
  - Classification Model
  - Training data (Example Set)
- Parameters
  - Laplace Correction
- Distribution Table (in results) shows posterior probabilities



| Attribute | Parameter     | no    | yes   |
|-----------|---------------|-------|-------|
| Outlook   | value=no rain | 0.600 | 0.667 |
| Outlook   | value=rain    | 0.400 | 0.333 |
| Outlook   | value=unknown | 0     | 0     |

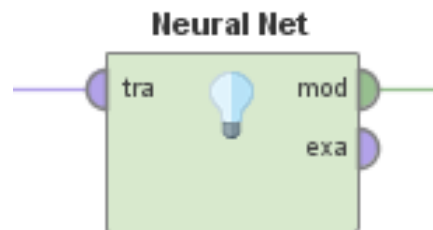
# Naïve Bayes in Rapid Miner

- Probabilities can be seen as confidences in the result

| Ro... | Play | prediction(Play) | confidence(no) | confidence(yes) | Outlook |
|-------|------|------------------|----------------|-----------------|---------|
| 1     | no   | yes              | 0.333          | 0.667           | no rain |
| 2     | no   | yes              | 0.333          | 0.667           | no rain |
| 3     | yes  | yes              | 0.333          | 0.667           | no rain |
| 4     | yes  | yes              | 0.400          | 0.600           | rain    |
| 5     | yes  | yes              | 0.400          | 0.600           | rain    |
| 6     | no   | yes              | 0.400          | 0.600           | rain    |
| 7     | yes  | yes              | 0.333          | 0.667           | no rain |
| 8     | no   | yes              | 0.333          | 0.667           | no rain |
| 9     | yes  | yes              | 0.333          | 0.667           | no rain |
| 10    | yes  | yes              | 0.400          | 0.600           | rain    |
| 11    | yes  | yes              | 0.333          | 0.667           | no rain |
| 12    | yes  | yes              | 0.333          | 0.667           | no rain |
| 13    | yes  | yes              | 0.333          | 0.667           | no rain |
| 14    | no   | yes              | 0.400          | 0.600           | rain    |

# Operators: Neural Net

- Input Port
  - Training data (Example Set)
- Output Ports
  - Classification Model
  - Training data (Example Set)
- Parameters
  - Hidden layers (amount & sizes)
  - Training cycles
  - Learning rate
  - Momentum
  - ...
- Requires numerical attributes

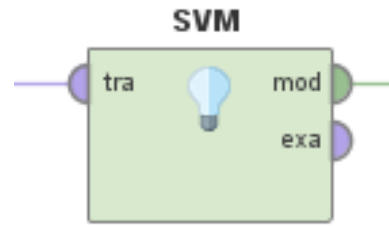


The screenshot shows a "Parameters" dialog box for a "Neural Net" operator. The dialog has a title bar with a close button (X). Below the title bar is a light blue lightbulb icon and the text "Neural Net". The parameters are listed in a table-like structure with labels, values, and information icons.

| Parameter             | Value                               | Info |
|-----------------------|-------------------------------------|------|
| hidden layers         | Edit List (0)...                    | ⓘ    |
| training cycles       | 500                                 | ⓘ    |
| learning rate         | 0.3                                 | ⓘ    |
| momentum              | 0.2                                 | ⓘ    |
| decay                 | <input type="checkbox"/>            | ⓘ    |
| shuffle               | <input checked="" type="checkbox"/> | ⓘ    |
| normalize             | <input checked="" type="checkbox"/> | ⓘ    |
| error epsilon         | 1.0E-5                              | ⓘ    |
| use local random seed | <input type="checkbox"/>            | ⓘ    |

# Operators: Support Vector Machine (LibSVM)

- Input Port
  - Training data (Example Set)
- Output Ports
  - Classification Model
  - Training data (Example Set)
- Parameters
  - SVM type
  - Kernel type
  - +more depending on SVM & Kernel type
- SVM Types
  - Last character indicates model type
  - ...C – Classification                      ...R - Regression
- Requires numerical attributes



The screenshot shows a 'Parameters' dialog box for the 'SVM (Support Vector Machine (LibSVM))' operator. The parameters are as follows:

| Parameter                 | Value                               |
|---------------------------|-------------------------------------|
| svm type                  | one-class                           |
| kernel type               | rbf                                 |
| gamma                     | 0.0                                 |
| nu                        | 0.5                                 |
| cache size                | 80                                  |
| epsilon                   | 0.001                               |
| class weights             | Edit List (0)...                    |
| shrinking                 | <input checked="" type="checkbox"/> |
| calculate confidences     | <input type="checkbox"/>            |
| confidence for multiclass | <input checked="" type="checkbox"/> |

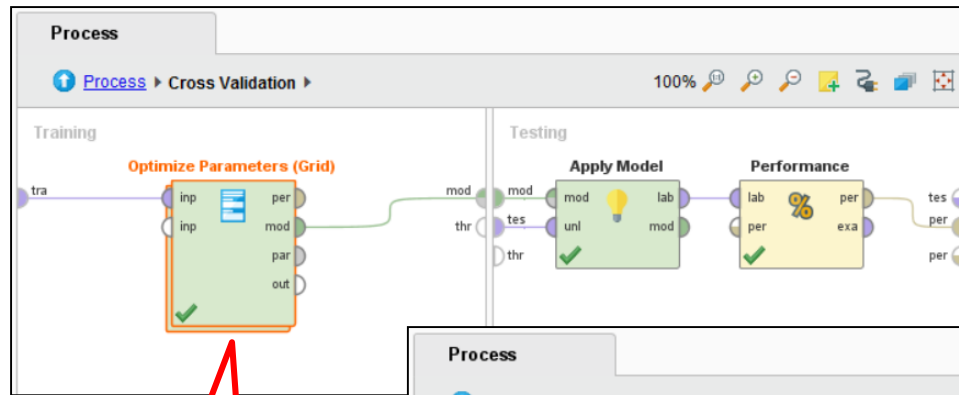
# Parameter Tuning

- Most learning algorithms have parameters
  - Systematic testing of parameter values is called optimisation
- There are different strategies for optimisation
  - Here we use Grid Search
  - Given a set of parameter values or ranges, test all possible combinations
- Attention!
  - We learn the best parameter values from the data
  - So we must **evaluate on a different dataset!!!**
  - Again, overfitting can be a problem



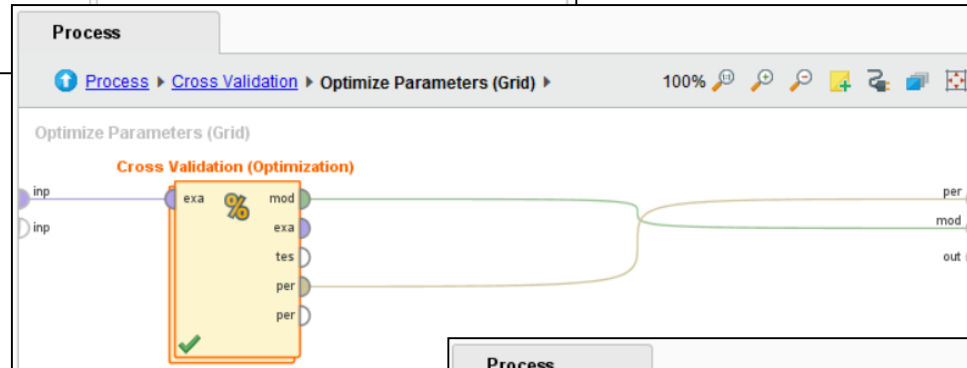
remember

# Nested Cross-Validation for Parameter Optimization

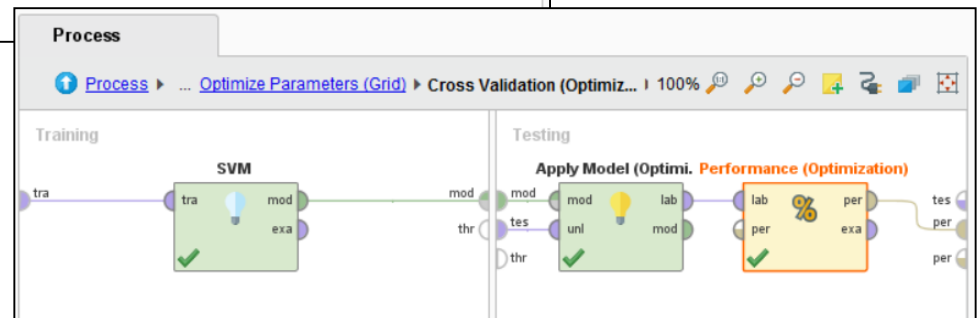


Outer Cross  
Validation

Optimize  
Parameters



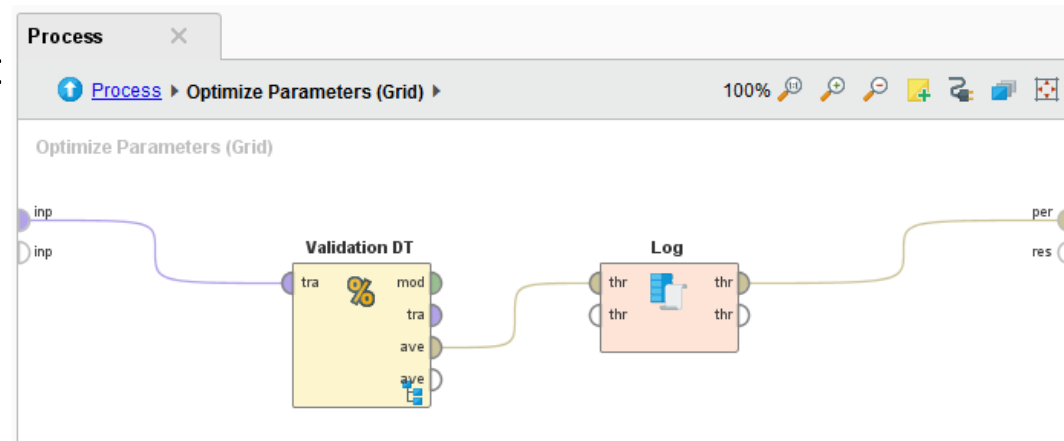
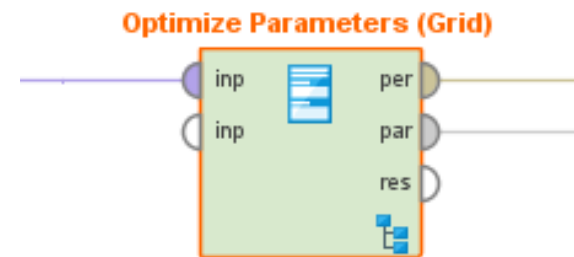
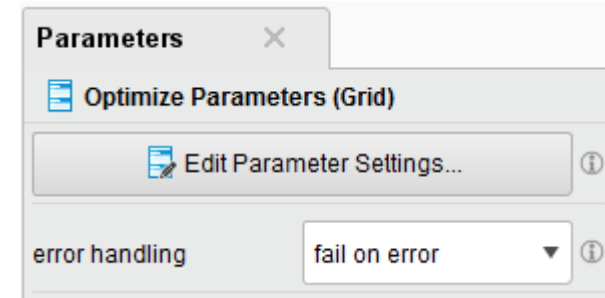
Inner Cross  
Validation





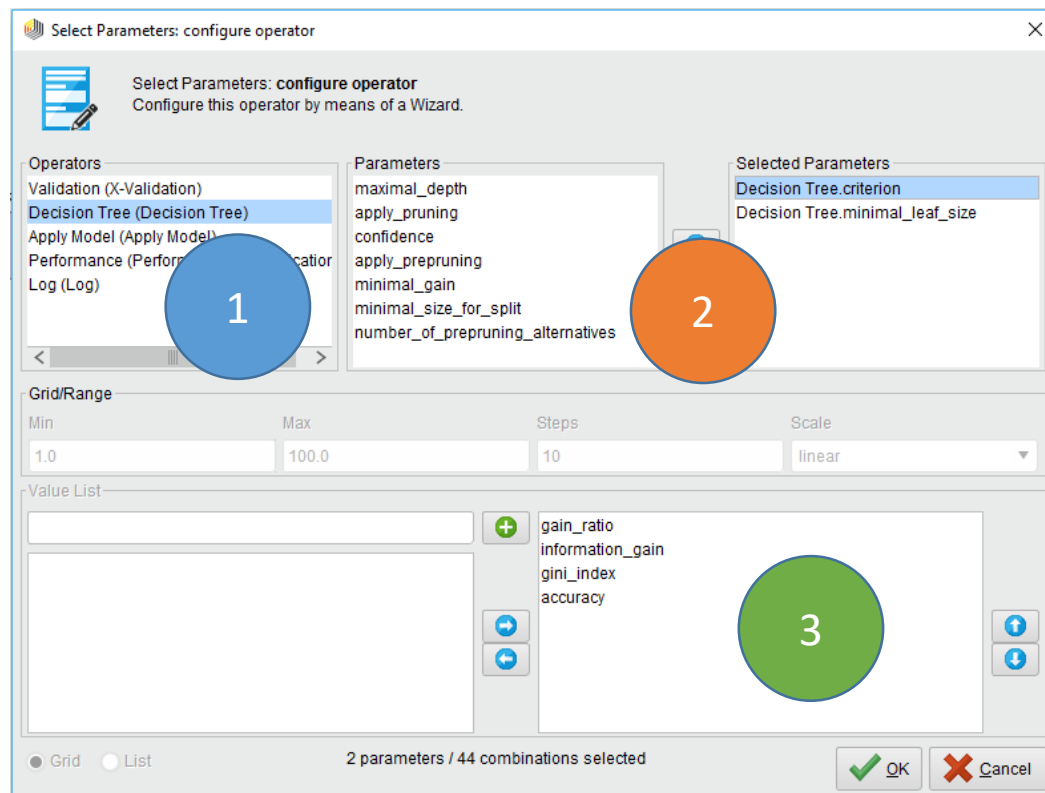
# Operators: Optimize Parameters (Grid)

- Input Ports
  - Any input (whatever you need in the nested process)
- Output Ports
  - Performance Vector (for optimal parameters)
  - Parameter Set (optimal values)
  - Any additional results (from the nested process)
- Parameters
  - Parameter values to test



# Operators: Optimize Parameters (Grid)

- Configuration steps:
  1. Select the operator you want to optimise
  2. Select the parameters of that operator
  3. Specify which values to test



# Operators: Optimize Parameters (Grid)

- Nominal parameter values
  - Select which values to use

Value List

Possible values

- information\_gain
- accuracy

Selected values

- gain\_ratio
- gini\_index

- Continuous parameter values

- Specify values by steps
  - Linear: 0 / 10 / 20 / 30 / ...
  - Quadratic: 0 / 1 / 4 / 9 / 16 / ...
  - Logarithmic: 1 / 2 / 3 / 4 / 6 / 10 / 16 / ...

Grid/Range

Min: 0

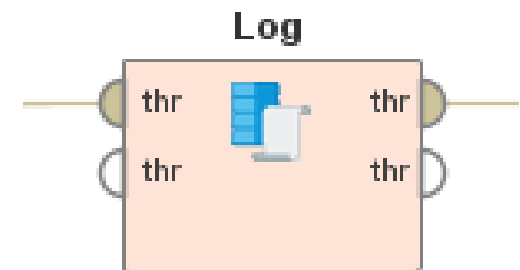
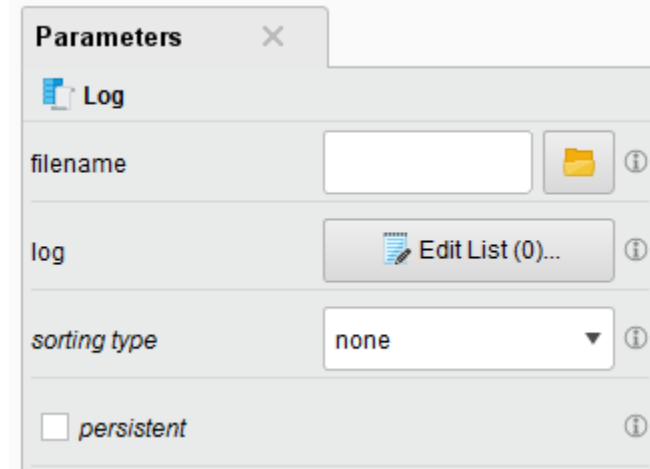
Max: 100.0

Steps: 10

Scale: linear

# Operators: Log

- Input Ports
  - Through
- Output Ports
  - Through (will simply forward what you connect to the input)
- Parameters
  - Filename
  - Log (what to log)
- Does not need any connections!
- But if connected, the order of execution is defined



# Operators: Log

- You can log parameters and output values of operators
  - Parameters can be changed by the optimise operator
  - Values are the results of operators

The diagram illustrates the logging process in a data mining tool. It features three main components:

- Performance Operator:** A yellow box with a percentage icon. It has four input/output ports labeled 'lab', 'per', 'per', and 'exa'.
- Parameters Dialog:** A window titled 'Parameters' showing 'Decision Tree' as the selected operator. The 'criterion' parameter is set to 'gain\_ratio'.
- Edit Parameter List Dialog:** A window titled 'Edit Parameter List: log' with a table for logging key-value pairs.

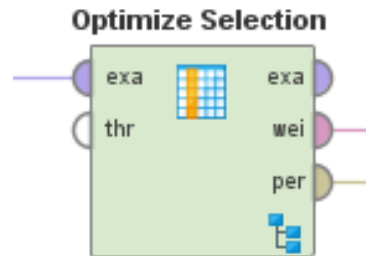
Two blue arrows indicate the flow of information:

- One arrow points from the 'criterion' parameter in the 'Parameters' dialog to the 'criterion' dropdown in the 'Edit Parameter List' table.
- Another arrow points from the 'value' dropdown in the 'Edit Parameter List' table to the 'Performance' operator.

| column name        | value                             |
|--------------------|-----------------------------------|
| my log value       | Decision Tree parameter criterion |
| my other log value | Performance value accuracy        |

# Operators: Optimize Selection

- Input Ports
  - Training data (Example Set)
  - Any Input
- Output Ports
  - Training data (Example Set)
  - Attribute weights
  - Performance Vector
- Parameters
  - Direction (add or remove attributes during optimisation)
- Finds the optimal selection of attributes



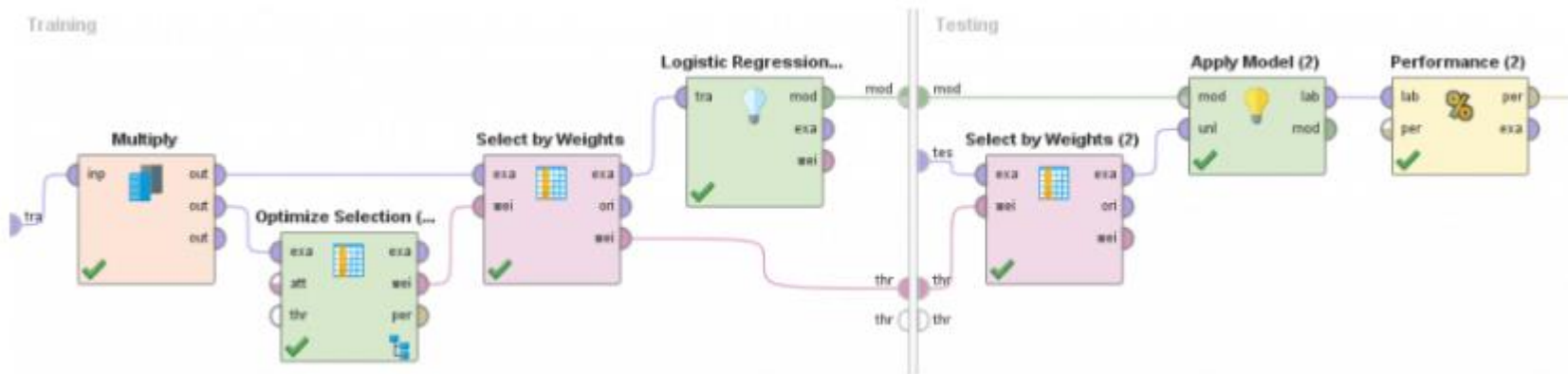
The screenshot shows the "Parameters" dialog for the "Optimize Selection" operator. The dialog has a title bar with a close button. The main area contains the following settings:

- selection direction**: A dropdown menu set to "forward" with a green checkmark icon.
- ☒ **limit generations without improvement**
- generations without impr...**: A text input field set to "1".
- ☐ **limit number of generations**
- keep best**: A text input field set to "1".
- ☒ **normalize weights**
- ☐ **use local random seed**
- ☐ **show stop dialog**
- ☐ **user result individual selection**
- ☐ **show population plotter**
- population criteria data fil...**: A text input field with a folder icon button.
- maximal fitness**: A text input field set to "Infinity".

# Optimize Selection with outer cross-validation

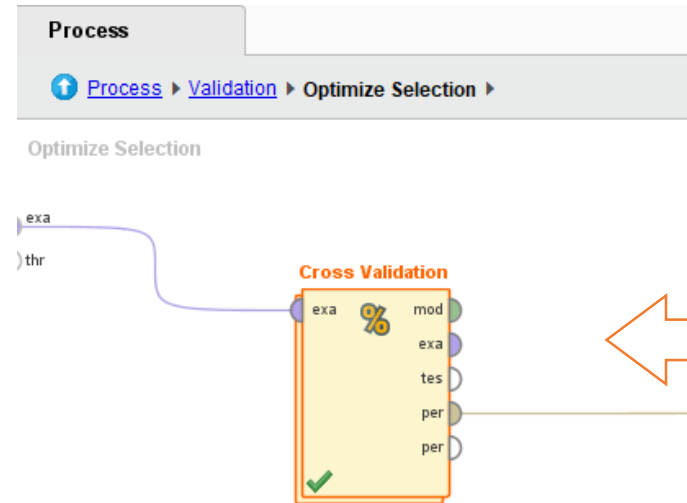
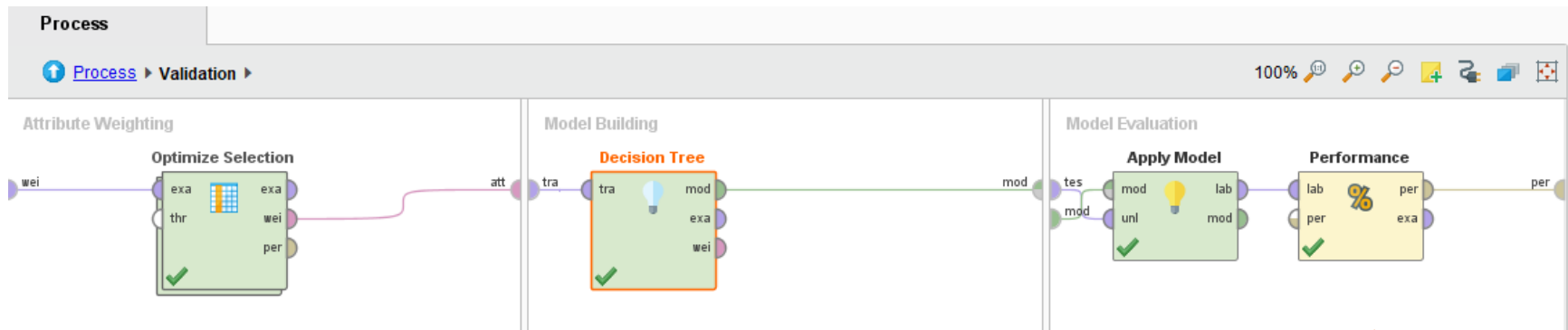
- Optimize Selection is a nested process
- In order to avoid „contaminating“ our results we would need to add an outer cross-validation (similarly to grid search)

Things become complicated...



source: <https://rapidminer.com/blog/learn-right-way-validate-models-part-4-accidental-contamination/> (accessed March 2020)

# Optimize Selection with outer cross-validation



Inner cross-validation for feature selection optimization

Use **Wrapper-X-Validation** which aligns the outer validation so that the test fold considers only the attributes selected by Optimize Selection operator.



# More Examples

- Rapidminer Tutorial on Accidental Contamination through Feature Selection and Parameter Optimization
  - <https://rapidminer.com/blog/learn-right-way-validate-models-part-4-accidental-contamination/>