

Classification

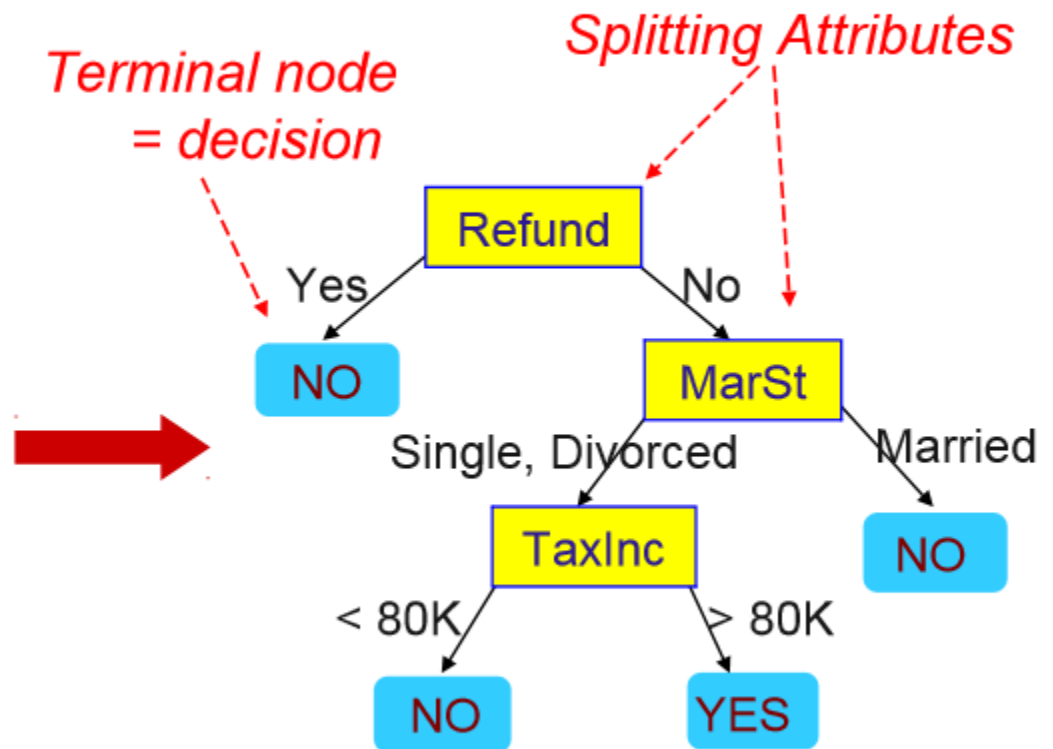
Exercise 4



Decision Tree Classifiers

<i>Tid</i>	<i>Refund</i>	<i>Marital Status</i>	<i>Taxable Income</i>	<i>Cheat</i>
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

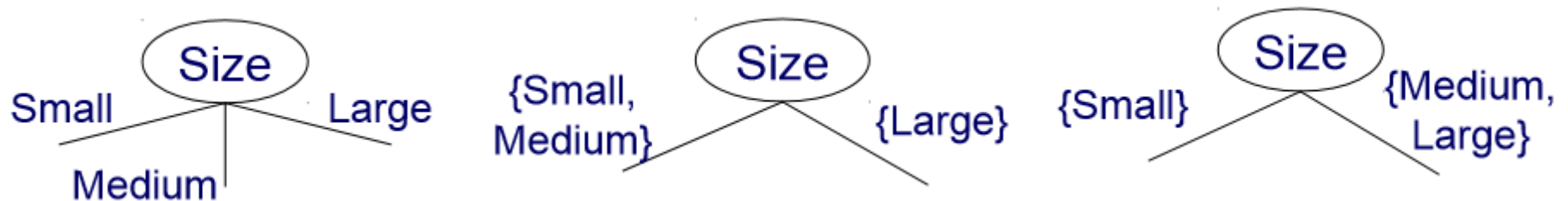
Training Data



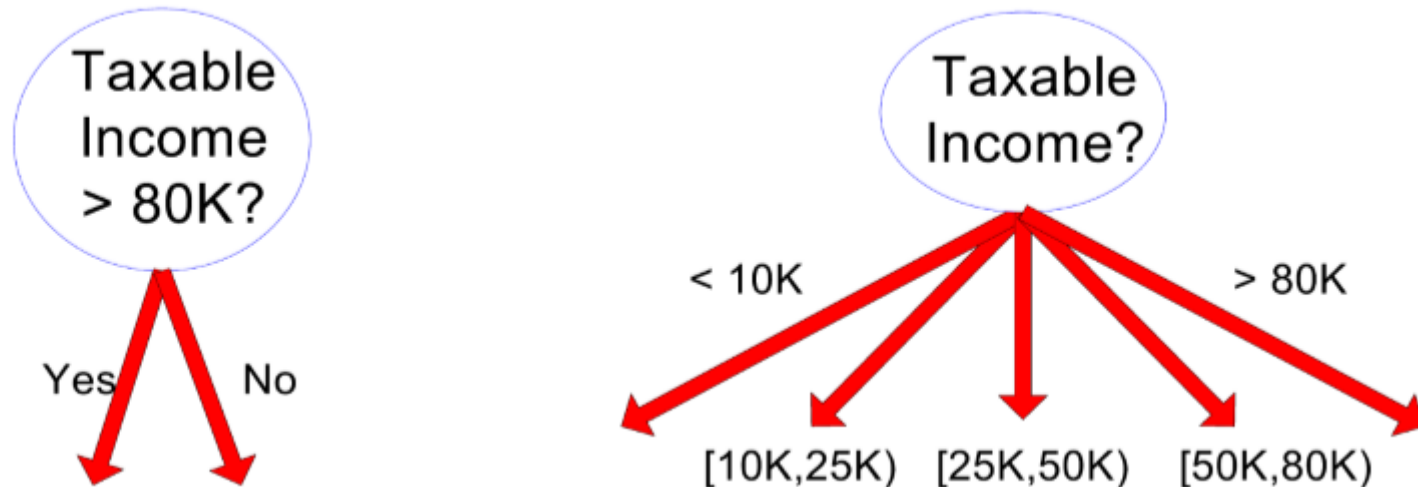
Model: Decision Tree

Attribute Test Conditions

- Depend on attribute types
 - Nominal values: Each partition contains a set of values



- Continuous values: Each partition contains a range of values



How to determine the best split?

- Measure the node **impurity** of the split
 - Gini Index
 - Information Gain
 - Gain Ratio
 - Classification Error
- The higher the measure, the less pure is the node
 - We want nodes to be as pure as possible

$$GINI(t) = 1 - \sum_i [p(i|t)]^2$$

$$Entropy(t) = -\sum_j p(j|t) \log_2 p(j|t)$$

$$GINI_{split} = \sum_i \frac{n_i}{n} GINI(i)$$

GainRAT

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^k \frac{n_i}{n} Entropy(i) \right)$$

SplitINFO =

$$SplitINFO = \sum_i \frac{n_i}{n} (1 - \max_j p(j|i))$$

Gini Index

- Gini Index for a given node t
 - 0 – all records belong to the same class
 - Max. (depends on number of classes) – records are equally distributed among classes
- Gini Index of a given split
 - The Gini index of each node in the split is weighted according to it's size



C1	5
C2	1
Gini=0.278	

C1	1
C2	5
Gini=0.278	

$$\frac{6}{12} \cdot 0.278 + \frac{6}{12} \cdot 0.278 = 0.278$$

C1	5
C2	2
Gini=0.408	

C1	1
C2	4
Gini=0.320	

$$\frac{7}{12} \cdot 0.408 + \frac{5}{12} \cdot 0.320 = 0.371$$

This is the better split!

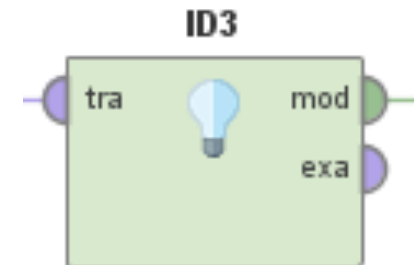
Operators: ID3

- Input Port
 - Training data (Example Set)
- Output Ports
 - Classification Model
 - Training data (Example Set)
- Parameters
 - Split criterion (measure)
 - Minimal size for split (examples)
 - Minimal leaf size (examples)
 - Minimal gain (for split)
- Can only handle nominal attributes!

Parameters ✕

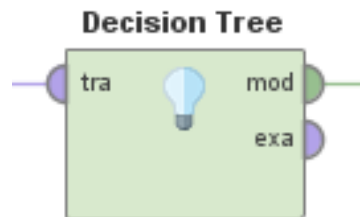
💡 ID3

criterion	gain_ratio ▼ ⓘ
minimal size for split	4 ⓘ
minimal leaf size	2 ⓘ
minimal gain	0.1 ⓘ



Operators: Decision Tree

- Input Port
 - Training data (Example Set)
- Output Ports
 - Classification Model
 - Training data (Example Set)
- Parameters
 - Split criterion (measure)
 - Maximal depth (-1 = unlimited)
 - Minimal size for split (examples)
 - Minimal leaf size (examples)
 - Minimal gain (for split)
 - Pruning



The screenshot shows a "Parameters" dialog box for a "Decision Tree" operator. The dialog has a title bar with a close button (X). Below the title bar, there is a lightbulb icon and the text "Decision Tree". The parameters are listed in a table-like structure with checkboxes for boolean options and input fields for numerical values. Each parameter has an information icon (i) to its right.

Parameter	Value
criterion	gain_ratio
maximal depth	20
apply pruning	☒
confidence	0.25
apply prepruning	☒
minimal gain	0.1
minimal leaf size	2
minimal size for split	4
number of prepruning alter...	3

Pre-processing for Classification

- After learning a classifier our results can be bad
- What can we do?
 - Change parameters
 - Change pre-processing
- We add some of our knowledge to the dataset by pre-processing
 - Change the range of values (normalisation)
 - Transform value types
 - Manipulate how attributes are split for decision trees (discretisation)

Operators: Discretize

- Input Port
 - Example Set
- Output Ports
 - Changed Example Set
 - Original Example Set
 - Pre-processing Model
- Transforms numerical attributes into nominal attributes

Parameters ×

 **Discretize (Discretize by Size)**

☐ create view ⓘ

attribute filter type  all ▼ ⓘ

☐ invert selection ⓘ

☐ include special attributes ⓘ

size of bins 10 ▼ ⓘ

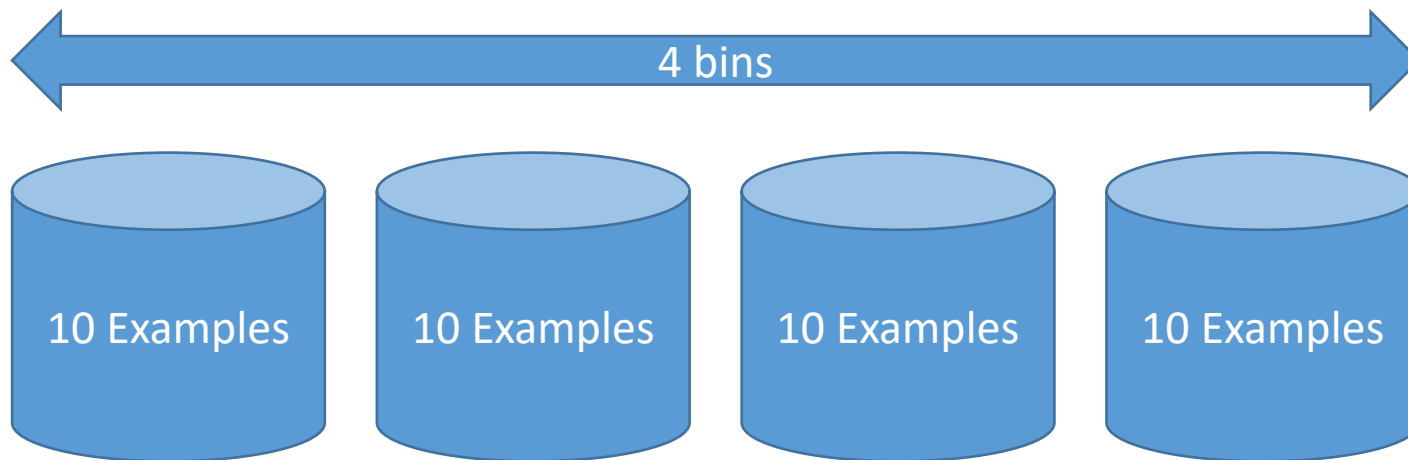
sorting direction decreasing ▼ ⓘ

range name type long ▼ ⓘ



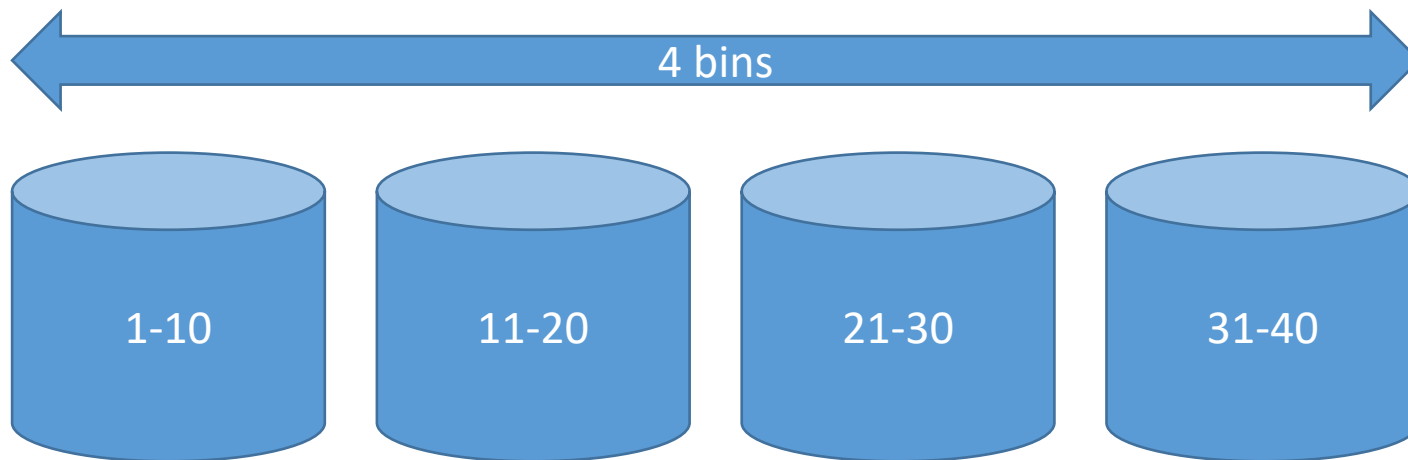
Discretization Techniques

- Equally sized number of examples per bin
 - **Discretize by Size:** Specify the size of the bins
 - **Discretize by Frequency:** Specify the number of bins



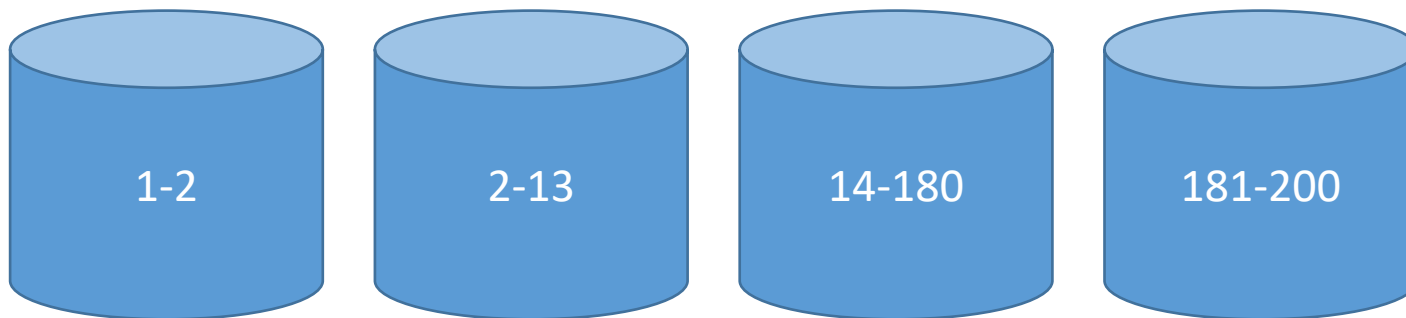
Discretization Techniques

- Equally sized data range per bin
 - **Discretize by Binning:** Specify the data range of bins



Discretization Techniques

- Varying data range per bin
 - Discretize by User Specification: Specify the range per bin
 - Discretize by Entropy: Don't Specify anything, minimise entropy

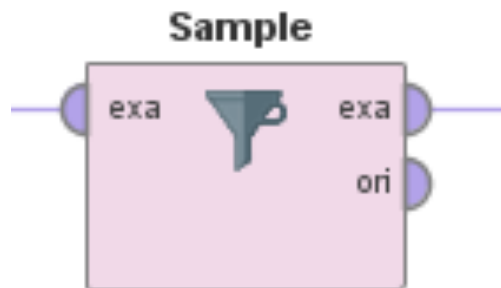


Balancing

- The number of instances per class affects model learning and evaluation
 - Same as with normalisation, we don't want to emphasize one specific class
 - So, we make sure all classes are evenly distributed (but only in the training set!)
- Example: 9,999 examples for class A, 1 for class B
 - Classify everything as class A has 99.99% accuracy
 - K-NN with $K \geq 3$ will always predict class A
- Counter-Example: 500 examples for class A and B
 - Classify everything as class A has 50% accuracy

Operators: Sample / Downsampling in Rapid Miner

- Input Port
 - Example Set
- Output Ports
 - Sampled data (Example Set)
 - Original Example Set
- Parameters
 - Sample (type)
 - Balance data (downsampling)
 - Sample size



Parameters window for the Sample operator. It includes a close button (X) and a filter icon. The settings are as follows:

Parameter	Value
sample	absolute
balance data	☐ (checked)
sample size	100
use local random seed	☐

Edit Parameter List: sample size per class

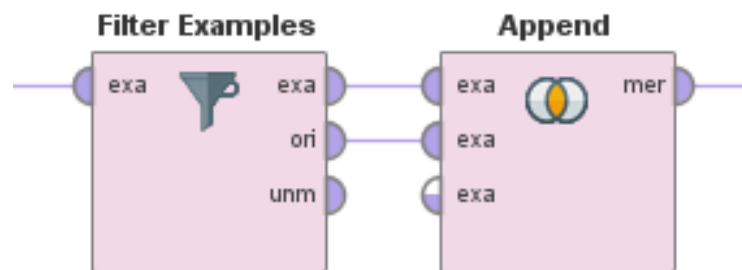
The absolute sample size per class.

class	size
class1	100
class2	100

Buttons: Add Entry, Remove Entry, Apply, Cancel

Upsampling in Rapid Miner

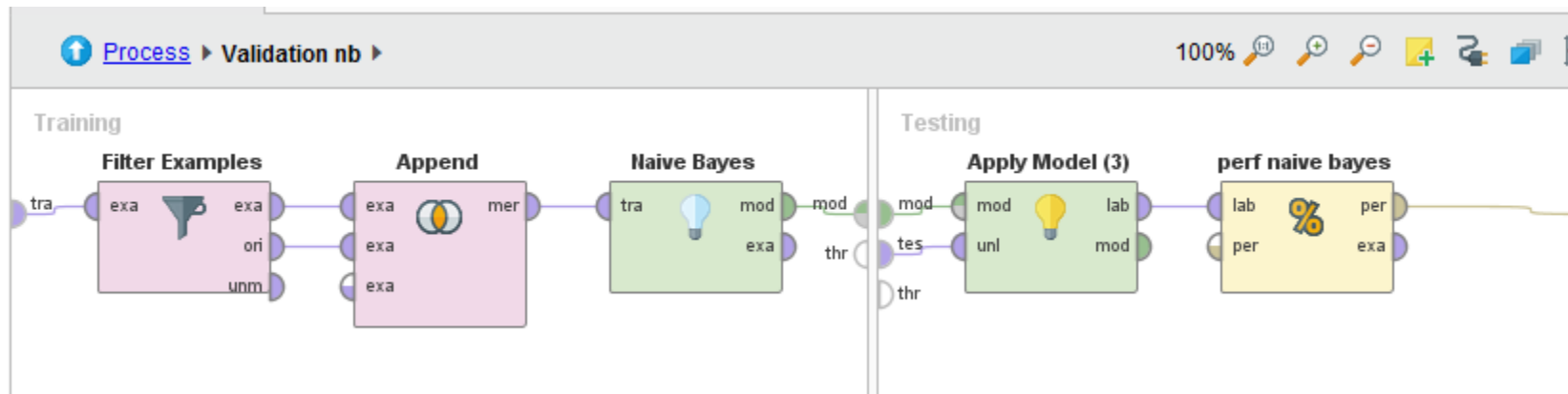
- Goal: Duplicate the examples of the under-represented class
 - Create a filter that only keeps the smaller class
 - Append the result to the original example set
- What's the effect?
 - Better relative distribution of the classes (more balanced)
 - But: no new information from duplicated examples
 - Better: find new examples for the small classes



Balancing is only done for training data!!!

- Otherwise, you would evaluate on duplicate examples!

Never balance your test data!

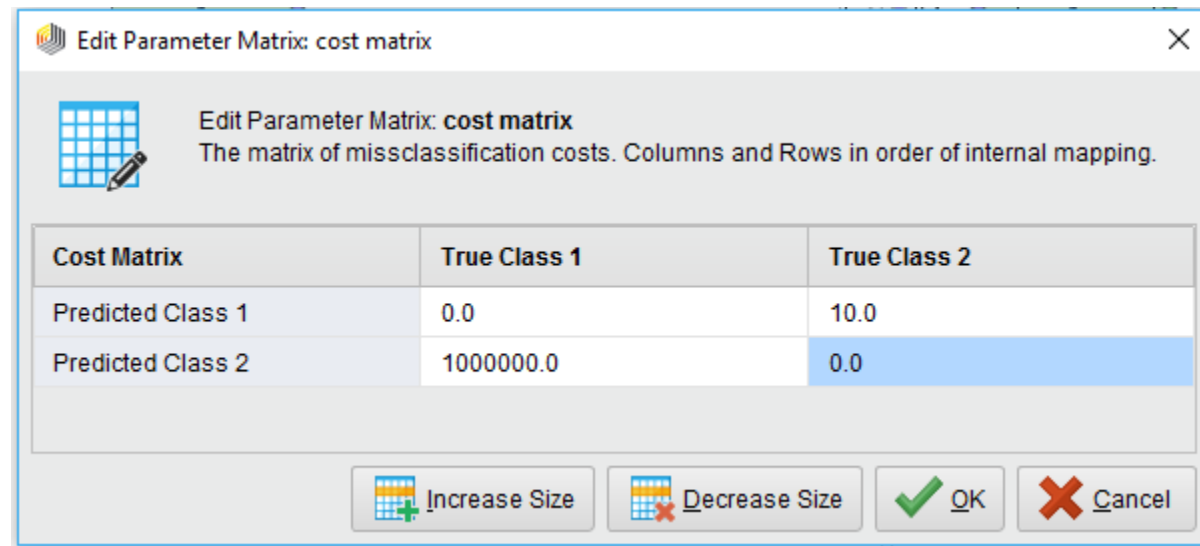
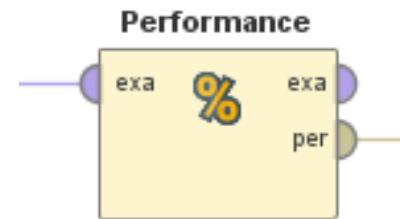
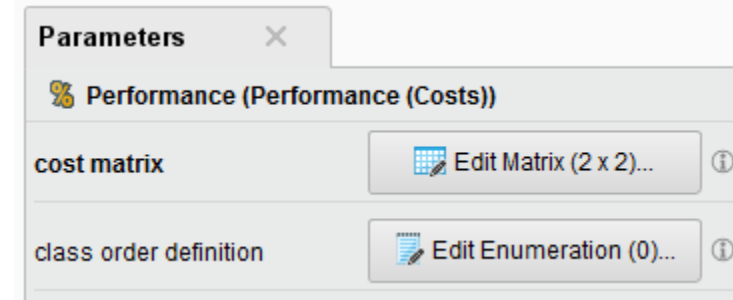


Additional Evaluation Methods

- Last exercise we looked at Accuracy, Precision & Recall
- We can also measure the cost of classification errors
 - Classifying a good part as broken costs the manufacturing cost
 - Classifying a broken part as good leading to an airplane crash may cost a little more!
- If we sum up the cost of the errors, we get an estimation of the total cost on the test data

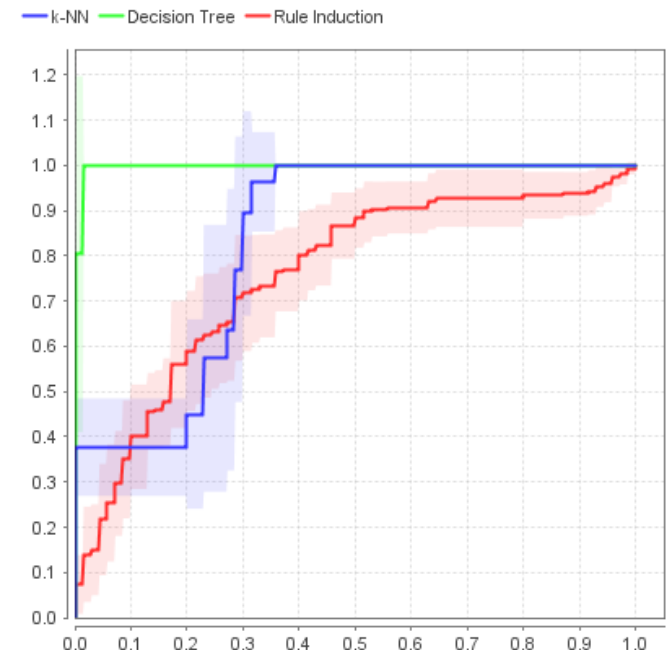
Operators: Performance (Costs)

- Input Port
 - Labelled Example Set
- Output Ports
 - Original Input data
 - Performance Vector
- Parameters
 - Cost Matrix
 - Class Order



Comparing Classification Results

- Receiver-Operating Characteristic (ROC) curve
 - Predictions ordered by confidence
 - Correct predictions increase steepness, incorrect predictions reduce it
 - Random guessing results in diagonal
 - Interpretation: Curve A above B means algorithm A better than B
 - See also: Area Under the Curve (AUC)
- Definitions
 - Only for binominal classification!
 - X-axis: False positive rate (=Fall-out)
 - False positive / actual negative
 - Y-axis: True positive rate (=Recall)
 - True positive / actual positive



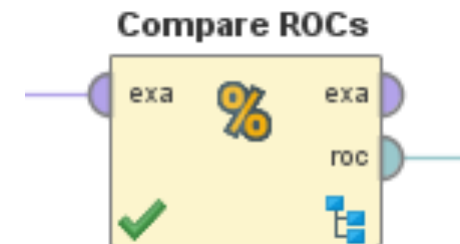
Operators: Compare ROC

- Input Port
 - Training data (Example Set)
- Output Port
 - Training data (Example Set)
 - ROC comparison
- Parameters
 - Number of folds (Cross-Validation)
 - Split ratio (Split-Validation)
 - Used if Number of folds = -1
 - ROC bias
 - Optimistic: correct predictions first
 - Pessimistic: wrong predictions first
 - Neutral: alternate

Parameters ✕

% Compare ROCs

number of folds	10	ⓘ
split ratio	0.7	ⓘ
sampling type	stratified sampling ▼	ⓘ
☐ use local random seed		ⓘ
☒ use example weights		ⓘ
roc bias	optimistic ▼	ⓘ



Operators: Compare ROC (nested process)

- Add learning operators for all models that should be compared

