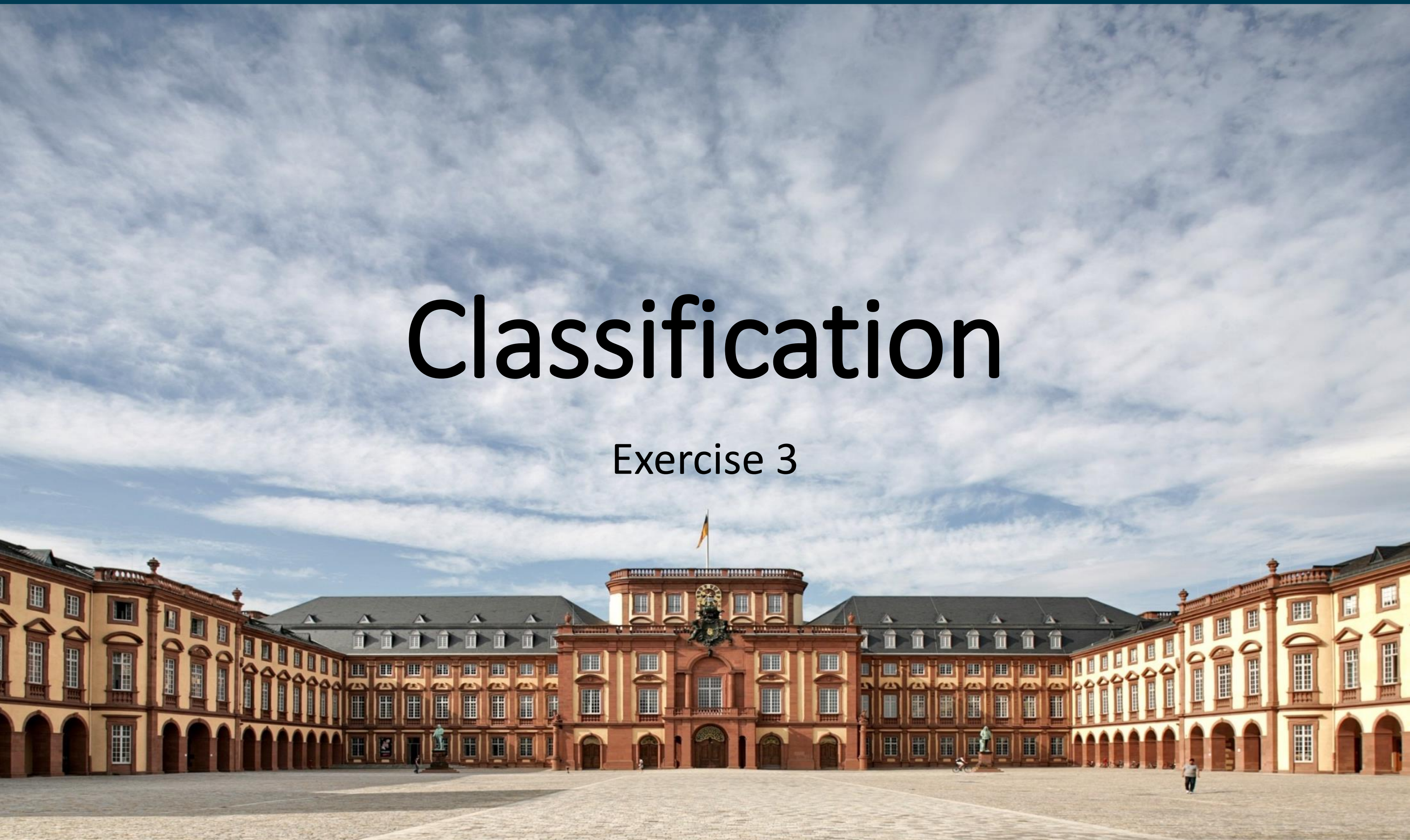
Classification

Exercise 3



Classification

"tree"

"tree"

"tree"

"not a tree"

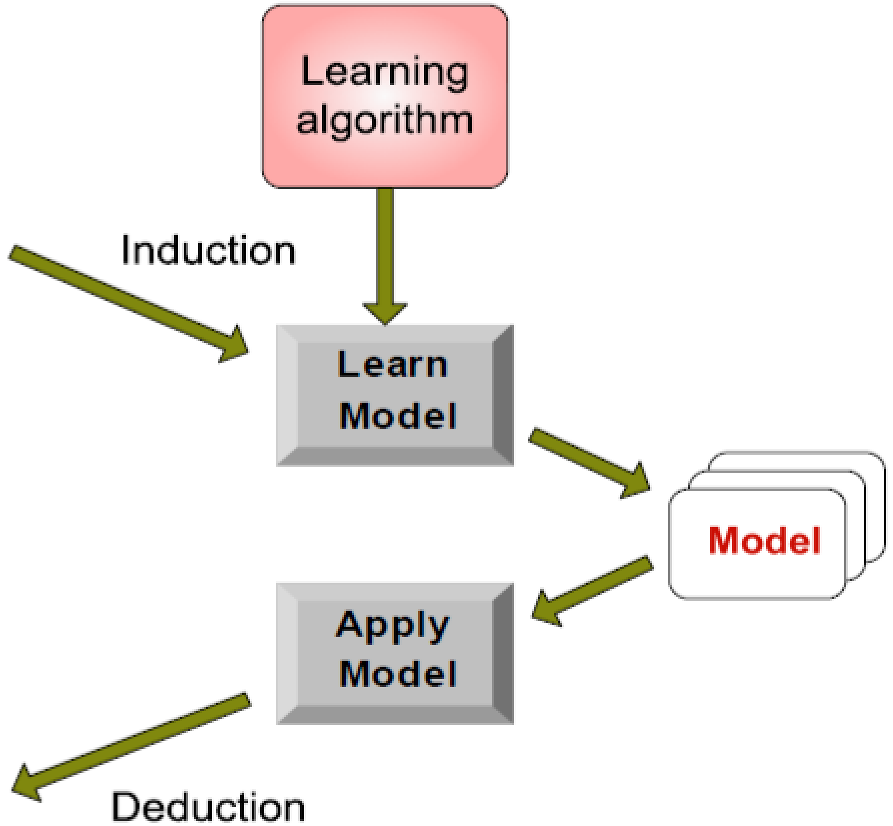
"not a tree"

"not a tree"

The Classification Workflow

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

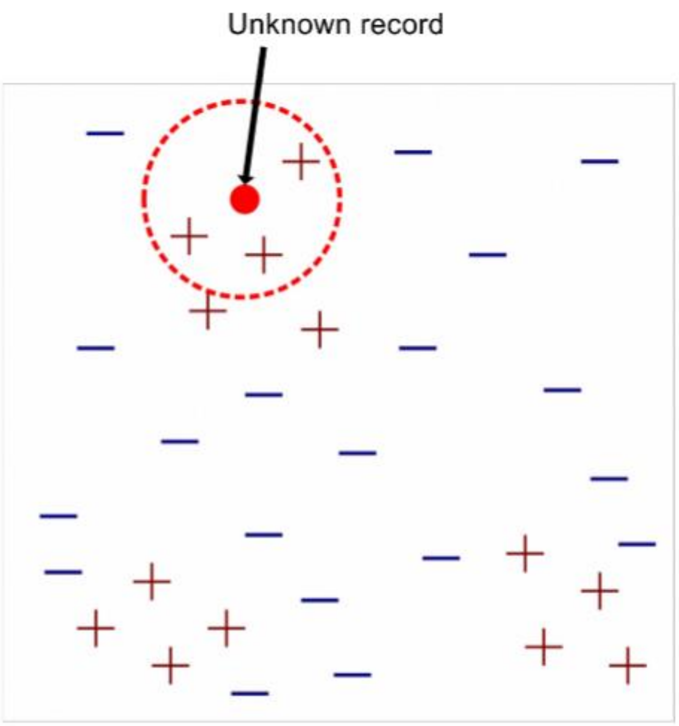


Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Unseen Records

K-Nearest-Neighbour

- Calculate the distance to all other points
- Choose the nearest K neighbours
- Let them vote for a class



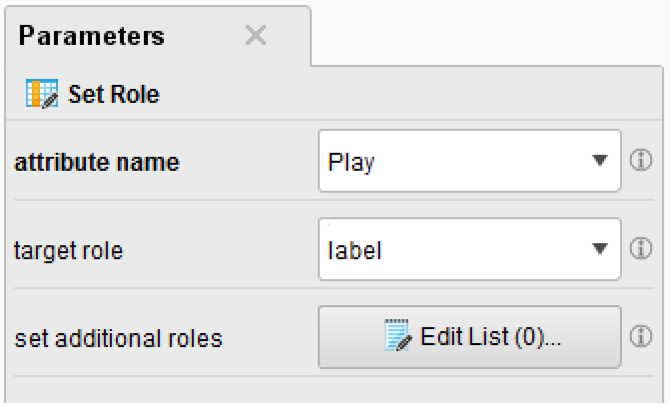
- Requires
  - All known records
  - Distance metric

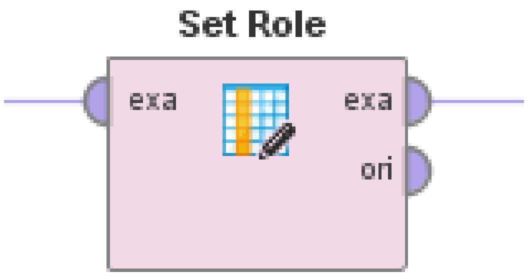
- Often very accurate
- But also slow

Operators: Set Role

- Input Port
  - Example Set
- Output Ports
  - Changed Example Set
  - Original Example Set
- Parameters
  - Attribute Name
  - Target Role

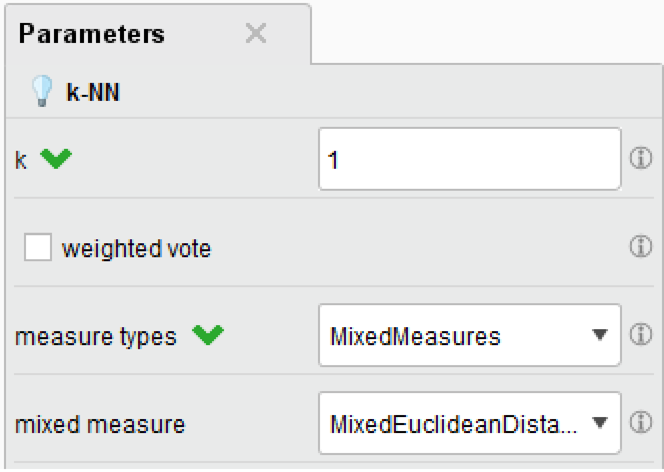
- Classification Operators need an attribute of type 'label'





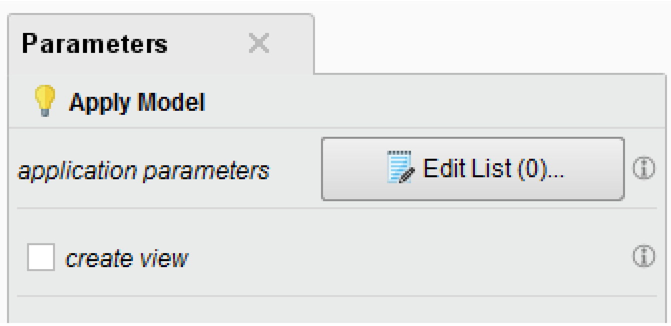
Operators: K-NN

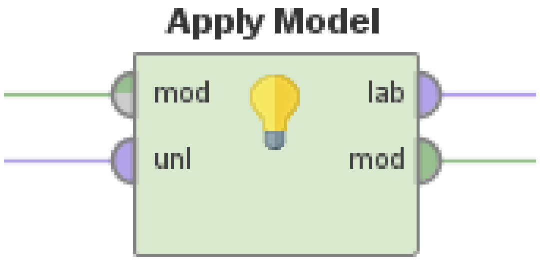
- Input Port:
  - Training data (Example Set)
- Output Ports
  - Classification Model
  - Training data (Example Set)
- Parameters
  - K
  - Weighted Vote
  - Similarity Measure



Operators: Apply Model

- Input Ports
  - Model
  - Unlabelled data (Example Set)
- Output Ports
  - Labelled data (Example Set)
  - Model

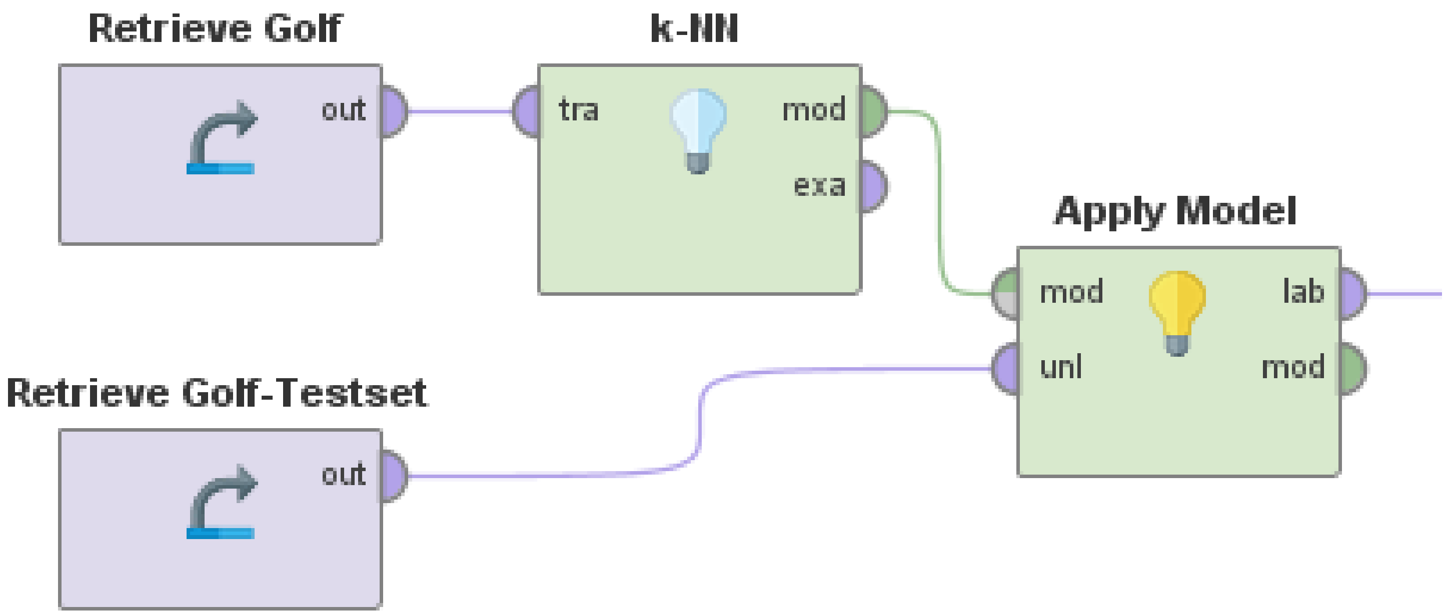




- Classification Operators do not apply the model they learn!
- You have to apply the model to ***a different*** example set

Process: Classification

- Learn the model from the training data
- Apply the model to the testing data
- Check the results

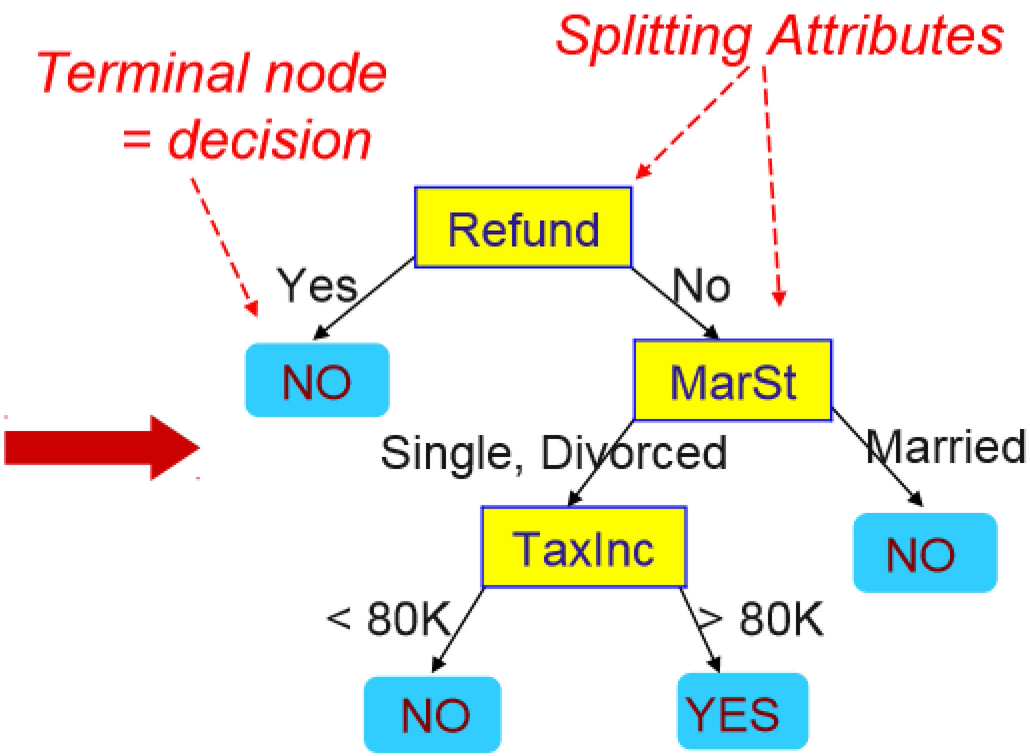


Decision Tree Classifiers

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
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

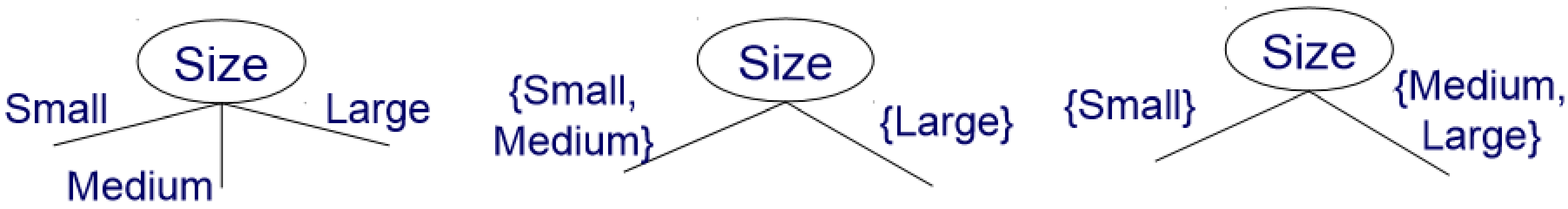
Training Data



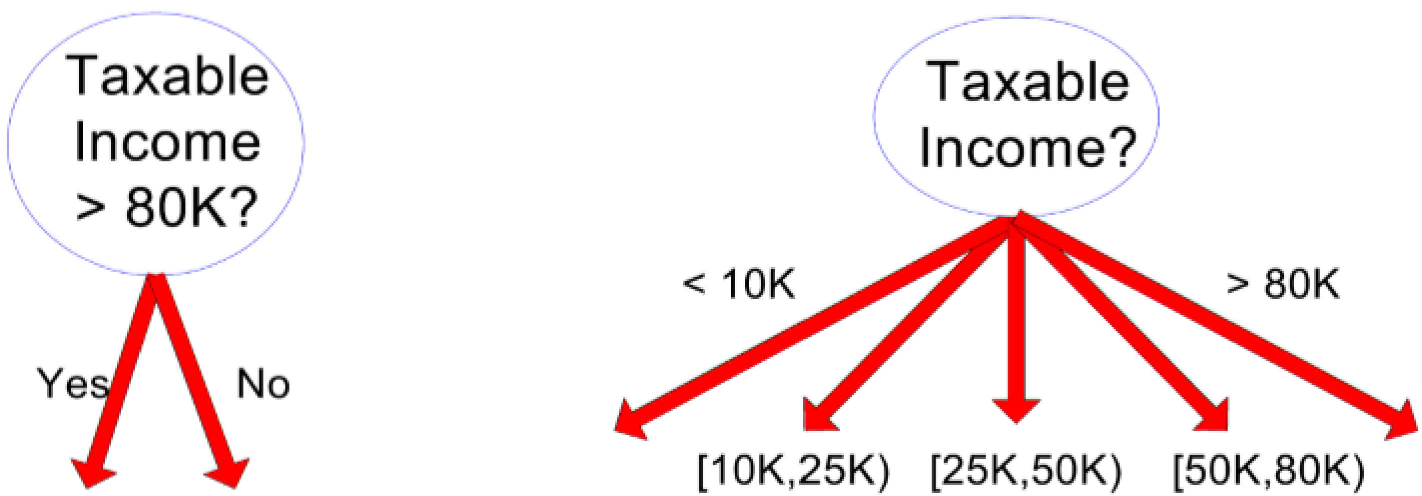
Model: Decision Tree

Attribute Test Conditions

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

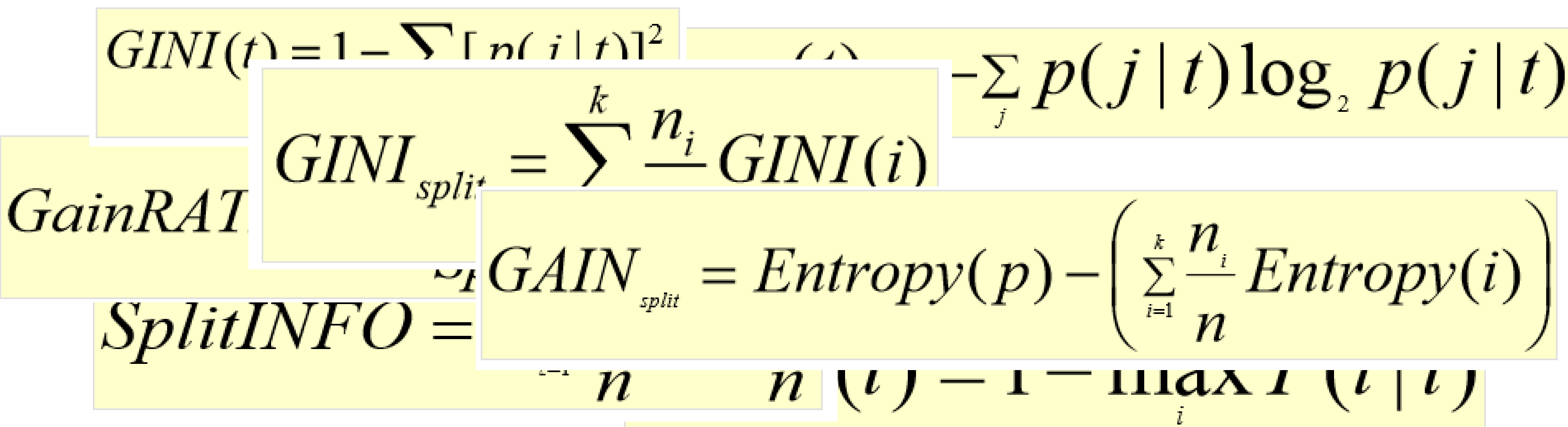


  - Continuous values: Each partition contains a range of values



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 Index

- Gini Index for a given node t
  - 0 – all records belong to the same class ← better
  - Max. (depends on number of classes) – records are equally distributed among classes

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

- 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	

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

This is the better split!

C1	5
C2	2
Gini=0.408	

C1	1
C2	4
Gini=0.320	

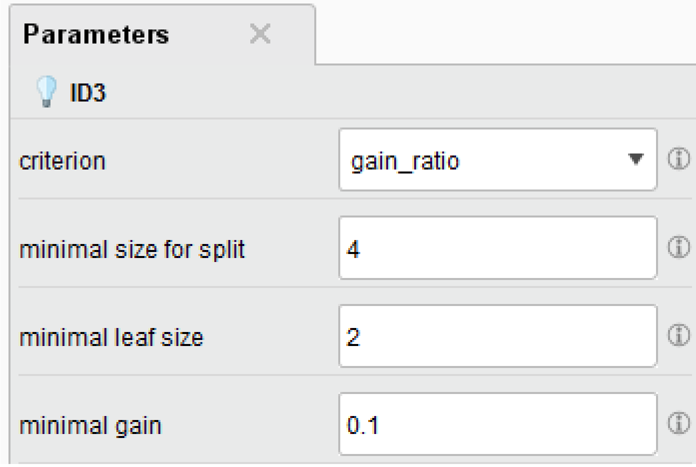
$$\frac{7}{12} \cdot 0.408 + \frac{5}{12} \cdot 0.320 = 0.371$$

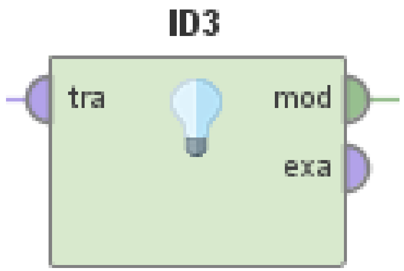
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

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)

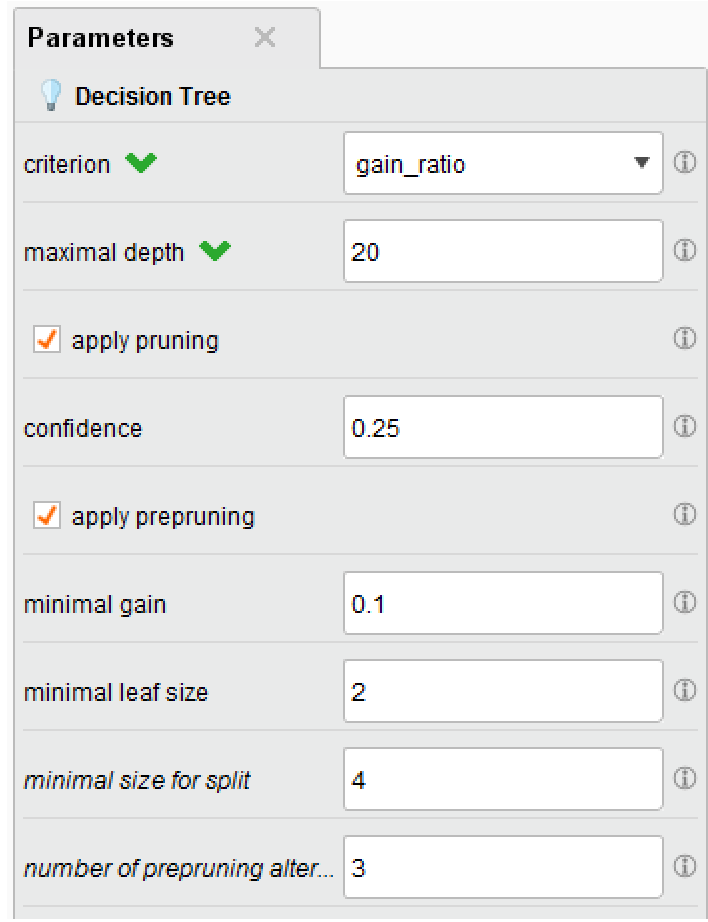
- Can only handle nominal attributes!

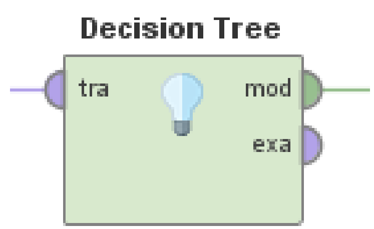




Operators: Decision Tree

- 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





Pre-processing for Classification

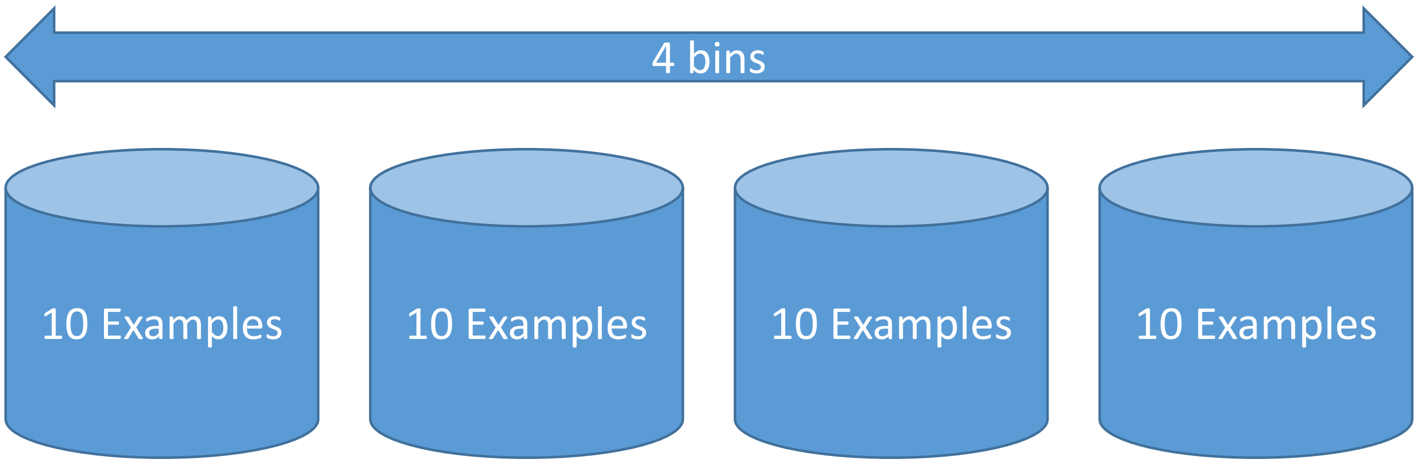
- After learning a classifier our results can be bad

- 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)

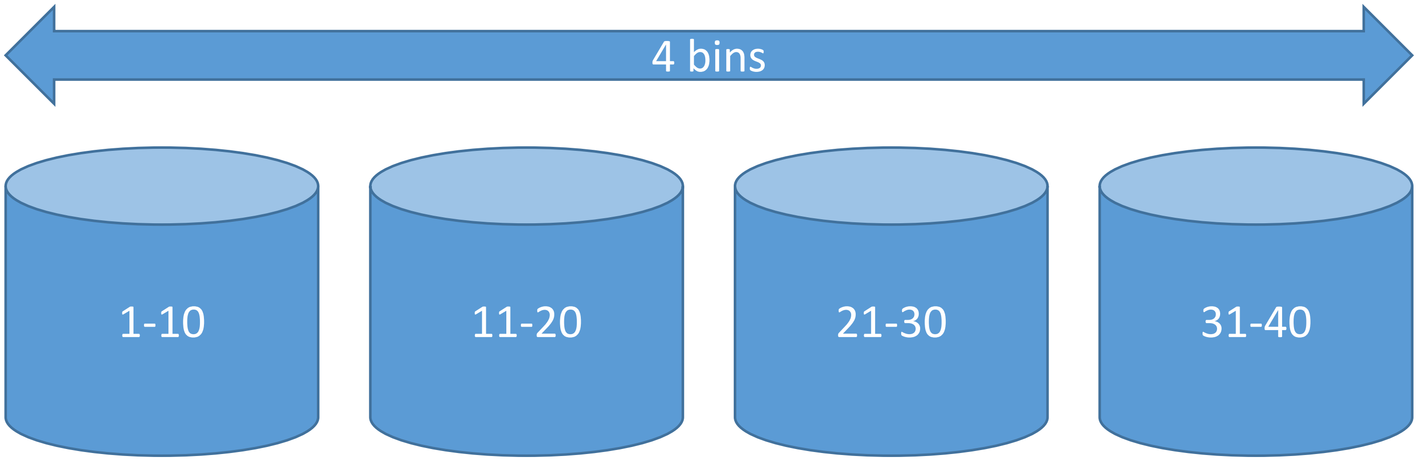
Discretization Techniques

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



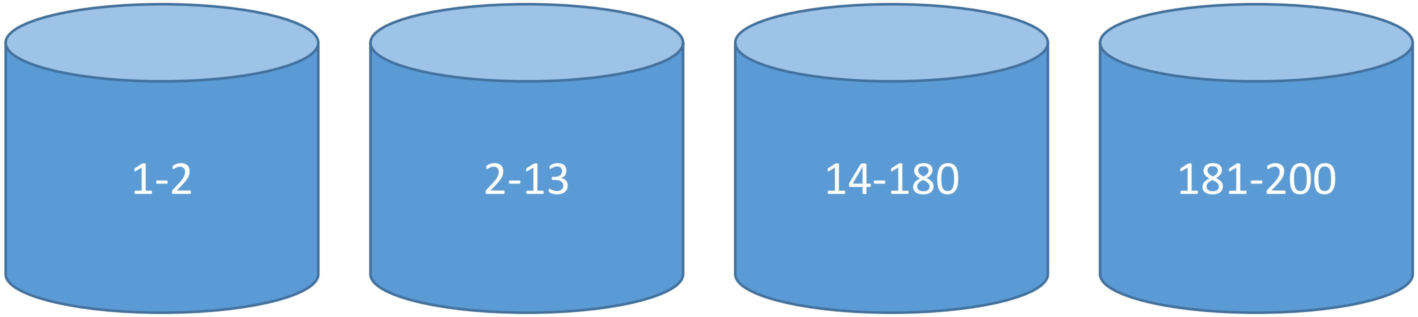
Discretization Techniques

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



Discretization Techniques

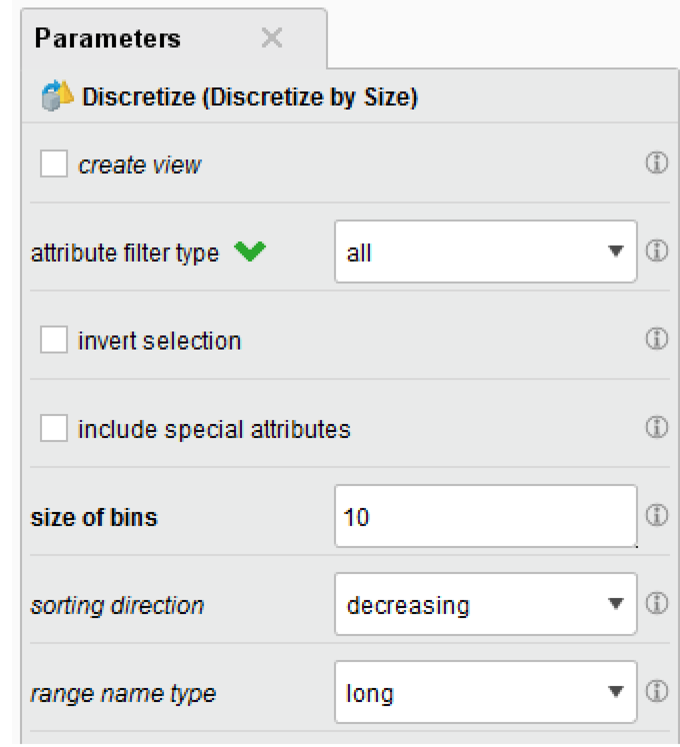
- Varying data range per bin
  - Discretize by User Specification: Specify the range per bin
  - Discretize by Entropy: Don’t Specify anything, minimise entropy



Operators: Discretize

- Input Port
  - Example Set
- Output Ports
  - Changed Example Set
  - Original Example Set
  - Pre-processing Model

- Transforms numerical attributes into nominal attributes



Evaluation

- How do we know the model actually works?
  - By counting the number of errors
  - On a ***different*** dataset

Important!!!

- What's the purpose of a model?
  - To apply it to new data where we don't know the label
- What happened if we used the same dataset?
  - How many errors for a K-NN classifier with K=1?
  - How good would that model be on a different dataset?

Evaluation: Confusion Matrix

- For every class in our dataset, the classifier can produce one of four possible results:

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	True Positive (TP)	False Negative (FN)
	Class=No	False Positive (FP)	True Negative (TN)

Evaluation Measures: Accuracy

- A single measure that tells you the overall accuracy of the result

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- "Number of correctly classified examples divided by the total number of examples."

- Problem: Unbalanced data
  - If 99% belong to class "yes"
  - And classifier always says "yes" – 99% Accuracy

Evaluation Measures: Precision and Recall

- Measure two aspects of the result for every class
- Precision: How many of the examples that were labelled "yes" are really "yes"?
  - "the number of correctly labelled examples divided by the number of all examples that were labelled with this class"
- Recall: How many of the examples that are really "yes" were labelled "yes'?
  - "the number of correctly labelled examples divided by the number of all examples that actually belong to this class"

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

Evaluation Measures: Precision and Recall

- An example:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

ID	Prediction	Actual Class
1	Yes	Yes
2	Yes	No
3	No	No
4	Yes	No

- For class "yes"
  - 1 true positive (ID 1)
  - 2 false positives (ID 2 & 4)
  - 1 true negative (ID 3)

$$Precision_{yes} = \frac{1}{1+2} = \frac{1}{3}$$

$$Recall_{yes} = \frac{1}{1+0} = \frac{1}{1}$$

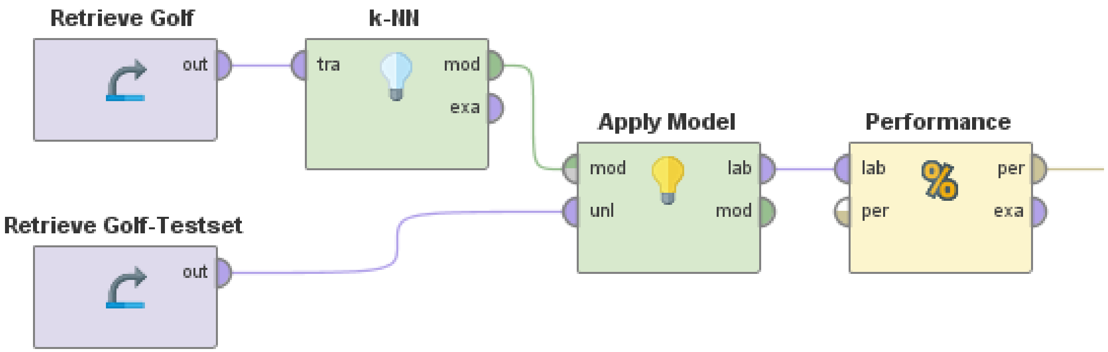
- For class "no"
  - 1 true negative (ID 1)
  - 2 false negatives (ID 2 & 4)
  - 1 true positive (ID 3)

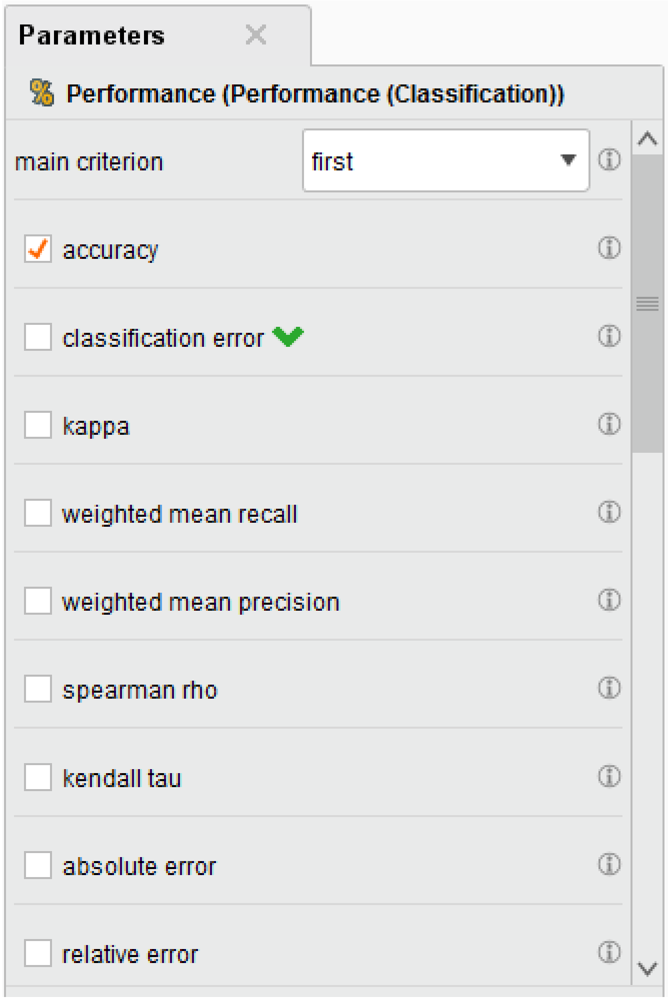
$$Precision_{no} = \frac{1}{1+0} = \frac{1}{1}$$

$$Recall_{no} = \frac{1}{1+2} = \frac{1}{3}$$

Operators: Performance (Classification)

- Input
  - Labelled Example Set
- Output
  - Performance
- Parameters
  - Performance Measures





Split-Validation / Cross-Validation

- What can you do if you only have one dataset?
  - Use one part of the data for training
  - Use *the other part* of the data for testing

- What if by accident all the easy examples are in the training set?
  - Then your model will not perform that good
  - Better to repeat the learning on different splits of the data

- X-Validation (Cross-Validation)
  - Split the dataset into X parts
  - Select one part for testing, use the rest for training
  - Repeat this until every part was used for training once

Just a reminder …

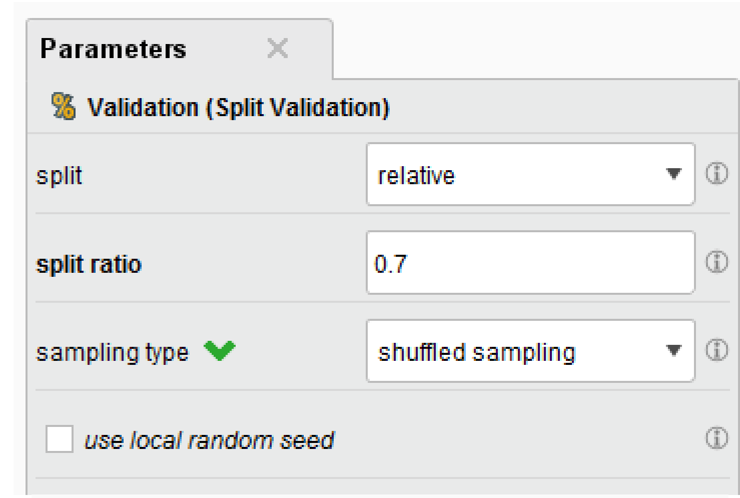
- If you use the same data for training and evaluation…

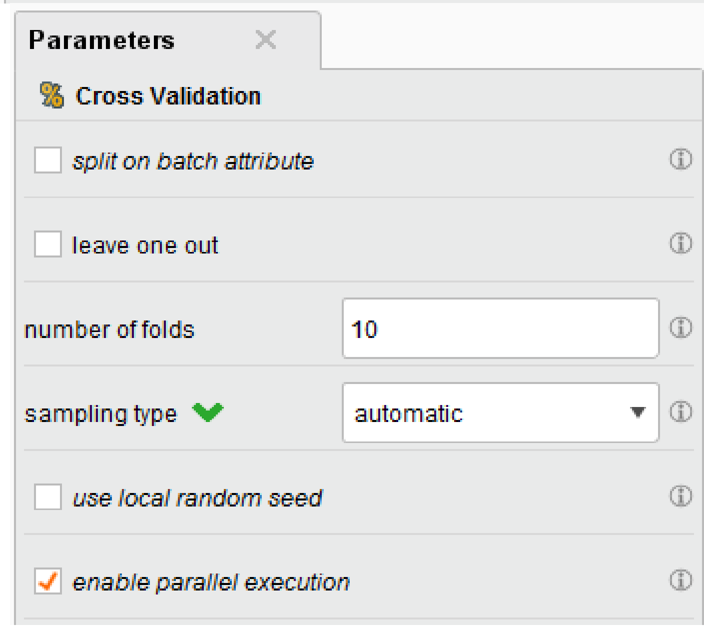
- … there will be no Easter!

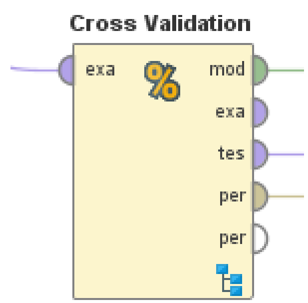
http://s198.photobucket.com/user/FuzzuBunny/media/MeMes/0584342a-0a3e-4ff7-8319-9615604c1203_zps26f59e45.jpg.html

Operators: Split Validation / Cross-Validation

- Input Port
  - Training data (Example Set)
- Output Ports
  - Classification Model
  - Training data (Example Set)
  - Test data (with prediction)
  - Performance
- Parameters
  - Split type
  - Split ratio
  - Sampling type

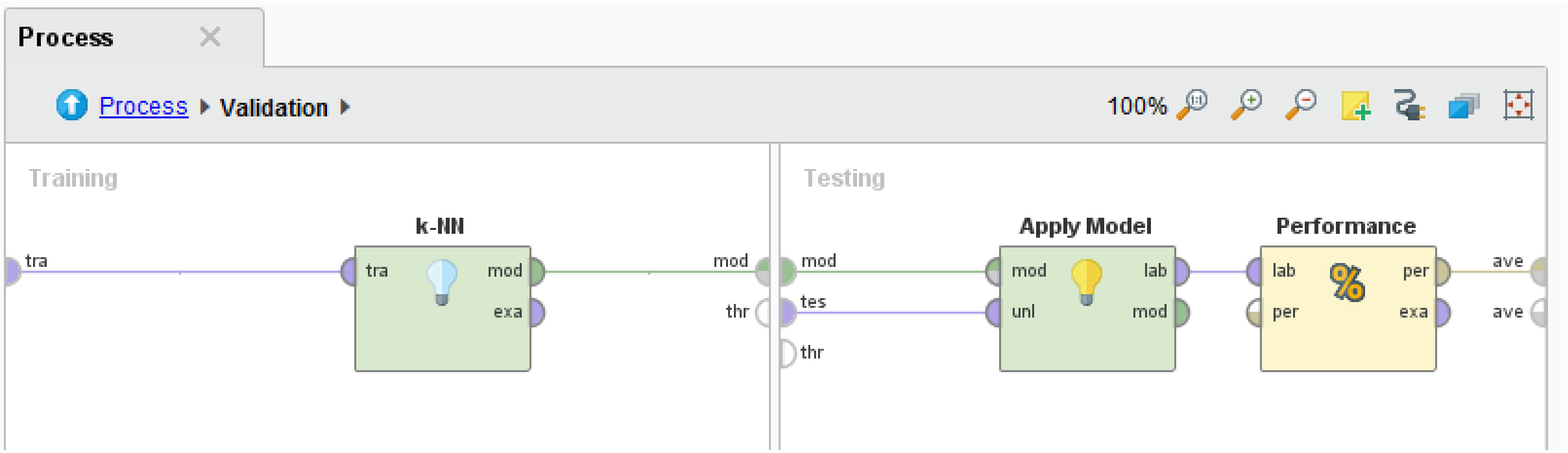




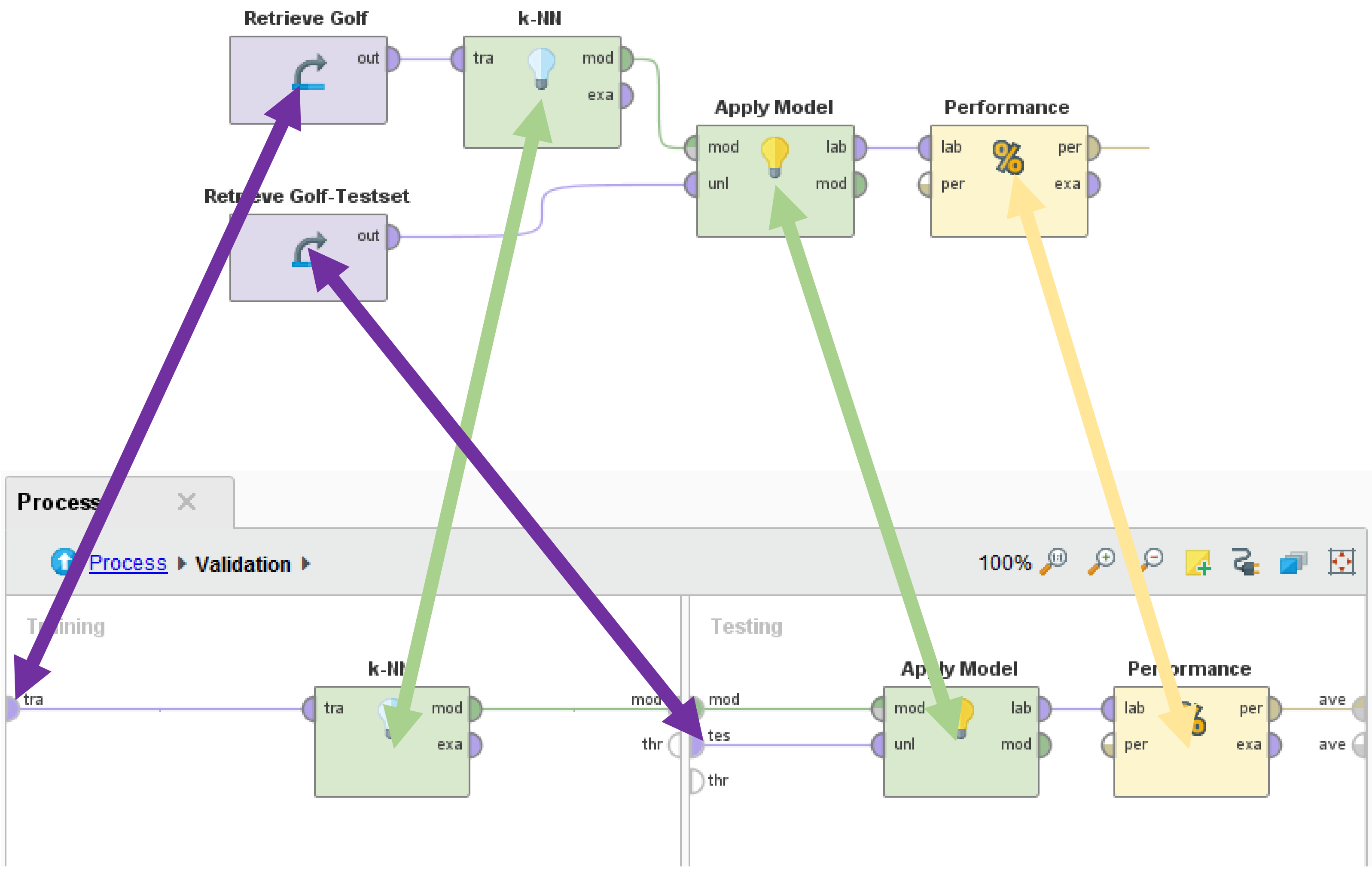


Nested Processes in Rapid Miner

- Operators can have “inner” processes that define their behaviour
- Split/Cross-Validation Operators have a “Training” and a “Testing” phase
  - Training: This is where you learn your model
  - Testing: This is where you evaluate



From two datasets to Split-Validation



The Mannheim RapidMiner Toolbox

- A Rapid Miner Extension with many great operators
- Developed by researchers from the Data and Web Science Group
- Contains the nearest centroid classifier