



University of  
**Reading**

CSMDM21 - Data Analytics and Mining

# Advanced KNIME - Programming with KNIME

Module convenor

**Dr. Carmen Lam**


[carmen.lam@reading.ac.uk](mailto:carmen.lam@reading.ac.uk)

Department of Computer Science

Lecture notes and videos created by

Prof. Giuseppe Di Fatta

# Overview

- 
- L1
- Advanced features, programmability
    - Data table and data types, I/O, data manipulation (ETL) nodes
    - Updates and extensions
    - KNIME SDK
    - Loops and control structures
    - (Recursion)
    - Flow variables
- L2
- Modularisation
    - Metanodes
    - Workflow 'Components' and 'Abstractions'
  - KNIME Server and Web Portal
- L3
- Advanced analytics (ML and data mining)
    - Example with tree-based predictive models
    - Parameter optimization

# Data Table

- Contains meta information (spec)
  - data types
  - domains
  - # of rows/cols
- Large tables are buffered on disc
- Blob cell support for large data cells e.g. images



Bit vector data - 4.0 - Bitvector Generator

Table "Default" - Rows: 150 | Spec - Columns: 5 | Properties

Row ID	D Double ...	S String Col	I Integer ...	[...] Collect...	[...] BitVectors
Row0	0.2	iris-setosa	1	[0.2,1]	10
Row1	0.2	iris-setosa	1	[0.2,1]	10
Row2	0.2	iris-setosa	1	[0.2,1]	10
Row3	0.2	iris-setosa	1	[0.2,1]	10
Row4	0.2	iris-setosa	1	[0.2,1]	10
Row5	0.4	iris-setosa	1	[0.4,1]	10
Row6	0.3	iris-setosa	1	[0.3,1]	10
Row7	0.2	iris-setosa	1	[0.2,1]	10
Row8	0.2	iris-setosa	1	[0.2,1]	10
Row9	0.1	iris-setosa	1	[0.1,1]	10
Row10	0.2	iris-setosa	1	[0.2,1]	10
Row11	0.2	iris-setosa	1	[0.2,1]	10
Row12	0.1	iris-setosa	1	[0.1,1]	10
Row13	0.1	iris-setosa	1	[0.1,1]	10
Row14	0.2	iris-setosa	1	[0.2,1]	10
Row15	0.4	iris-setosa	1	[0.4,1]	10
Row16	0.4	iris-setosa	1	[0.4,1]	10
Row17	0.3	iris-setosa	1	[0.3,1]	10
Row18	0.3	iris-setosa	1	[0.3,1]	10
Row19	0.3	iris-setosa	1	[0.3,1]	10
Row20	0.2	iris-setosa	1	[0.2,1]	10

# Data Types

Manually created table - 0:7 - Table Creator

File

Table "default" - Rows: 1 Spec - Columns: 6 Properties Flow Variables

Row ID	I int	D double	L long	B bool	S string	[00] bit_vect
Row0	0	3.5	696	false	hello	0001

- Common data types

- Double Value

D Double ...  
0.2

- Int Value

I Integer ...  
1

- String Value

S String Col  
Iris-setosa

- Collections

- Sets
- Lists

[...] Collecti...  
[0.2,1]

- Bit vectors

[00] BitVectors  
10

- Additional data types

- Terms and Documents

- Image

- Network

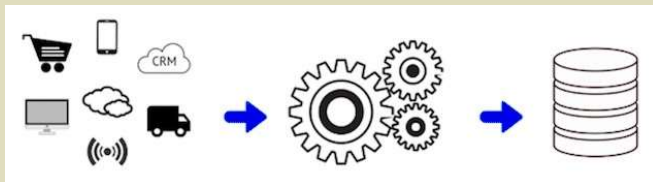
- Chemical types

- Molecules i.e. CDK, Smiles, SDF, ...

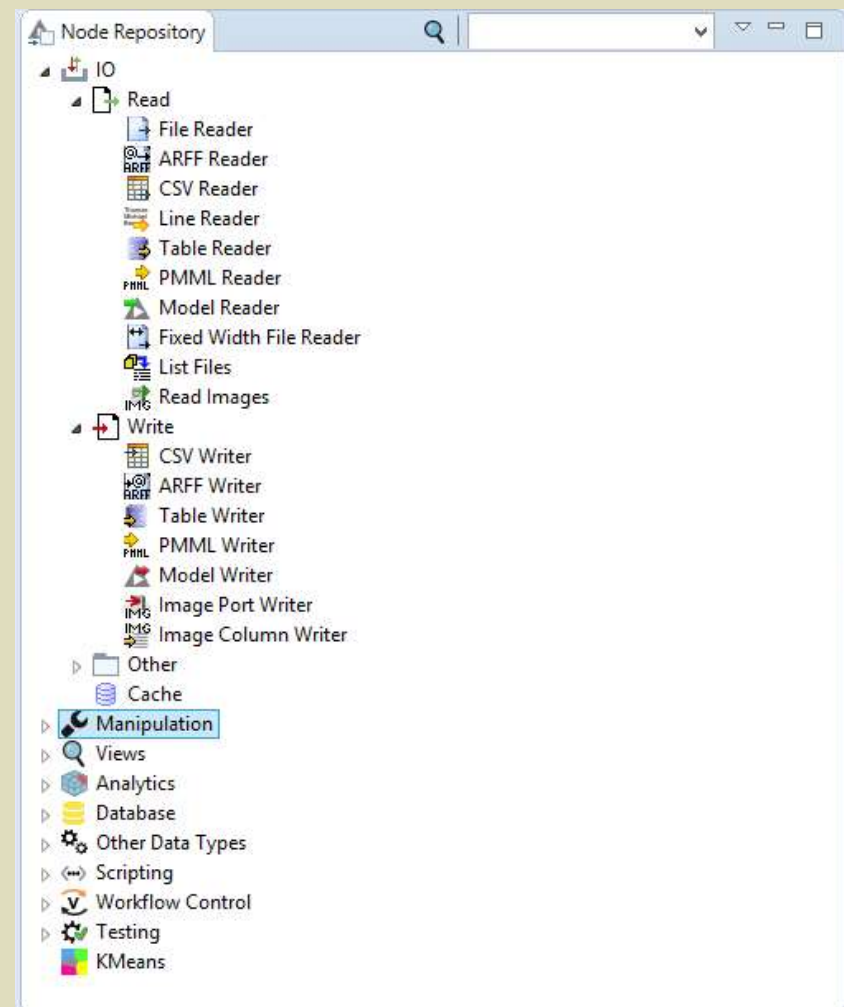
- Distance Matrix

- Custom data types

# I/O and Extract-Transform-Load (ETL)

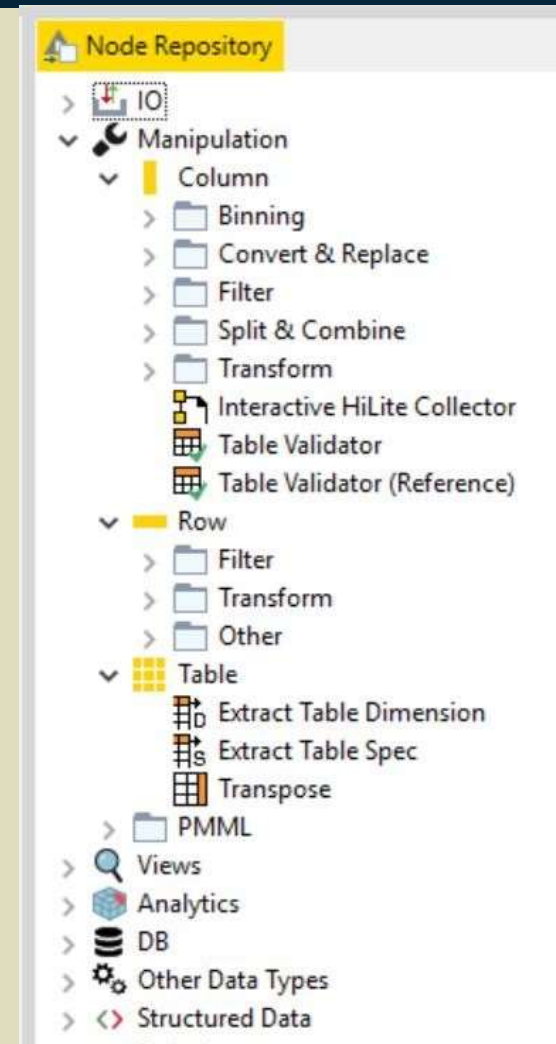


- read (load) data into the workflow
- store (write) the results of the analysis somewhere, such as in a file in your local file system.
- import/export data to DBs
- import/export data models in PMML format
- data manipulation and transformation



# Data Manipulation Nodes

- Large number of useful nodes to manipulate and transform data
  - E.g., Row/Column Filter/Transform
- Available documents in Bb:
  - KNIME cheat sheets  
<https://www.knime.com/cheat-sheets>
  - ETL in KNIME - useful nodes
  - Advanced KNIME - useful nodes









# KNIME Extensions (additional features)

Some available extensions include:

- Chemistry types and features
- Distance Matrix
- Ensemble Learning, Itemset Mining
- OpenStreetMap
- R Statistics Integration
- Python integration
- Weka Data Mining Integration
- HTML/PDF Writer
- Report Designer
- Webservice Client
- XLS Support
- XML Processing
- Cloud connectors (Amazon, Azure, etc.)
- etc.



# KNIME Extensions

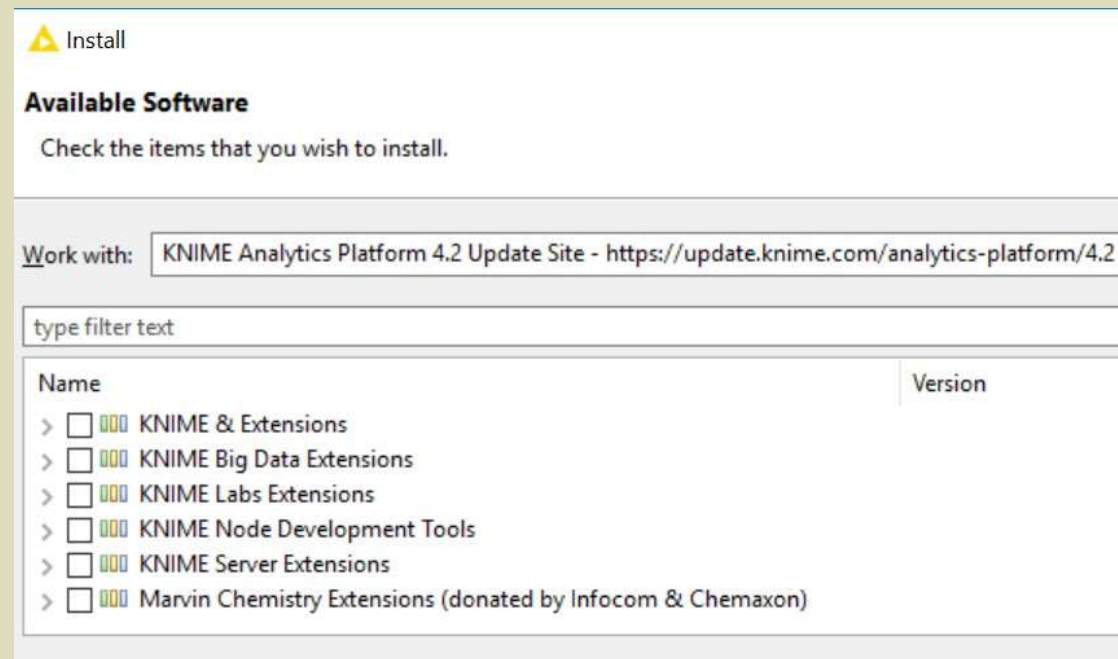
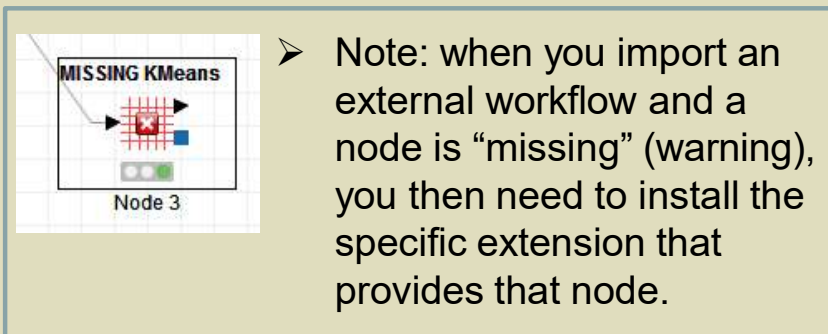
- > ☐  KNIME & Extensions
- > ☐  KNIME Big Data Extensions
- > ☐  KNIME Labs Extensions
- > ☐  KNIME Node Development Tools
- > ☐  KNIME Server Extensions
- > ☐  Marvin Chemistry Extensions

- KNIME & Extensions
  - Not all features are included in the standard installation
- “Labs” extensions: [knime.com/knime-labs](https://knime.com/knime-labs)
  - Network Mining, Active Learning, Text Processing
  - R/Groovy/Matlab/Python/Perl Scripting, STARK, etc.
  - Integrations: Weka, Keras, H2O, Tableau, MongoDB, Amazon ML, ONNX, TensorFlow, etc.
- Community contributions: [knime.com/community](https://knime.com/community)
  - Chemoinformatics
  - High Content Screening
  - Image Processing
  - Next Generation Sequencing
  - etc.
- KNIME is designed to be extended!
  - You can create your own KNIME nodes (extensions) by using the KNIME SDK version.



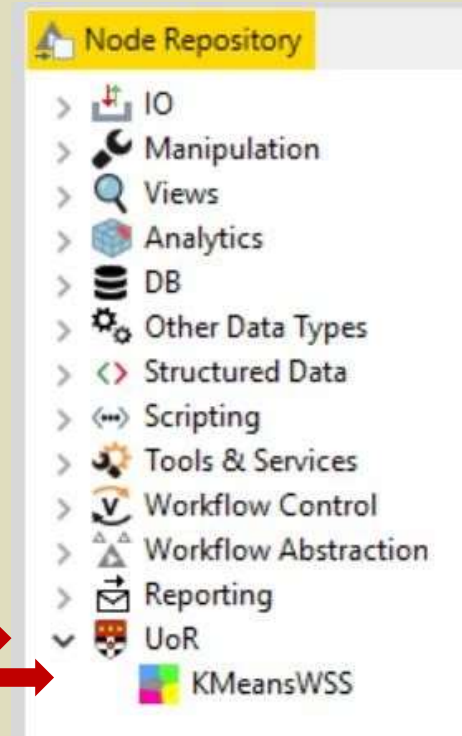
# KNIME Updates and Extensions

- Extensions are installed via the **KNIME update site**.
  - In KNIME, menu “Help” -> “Install New Software...”
  - select the KNIME update site (<http://update.knime.com/analytics-platform/4.2>)
  - select the features you want to install in the dialog.
  - You need to restart KNIME after installing new extensions in order to get them activated.



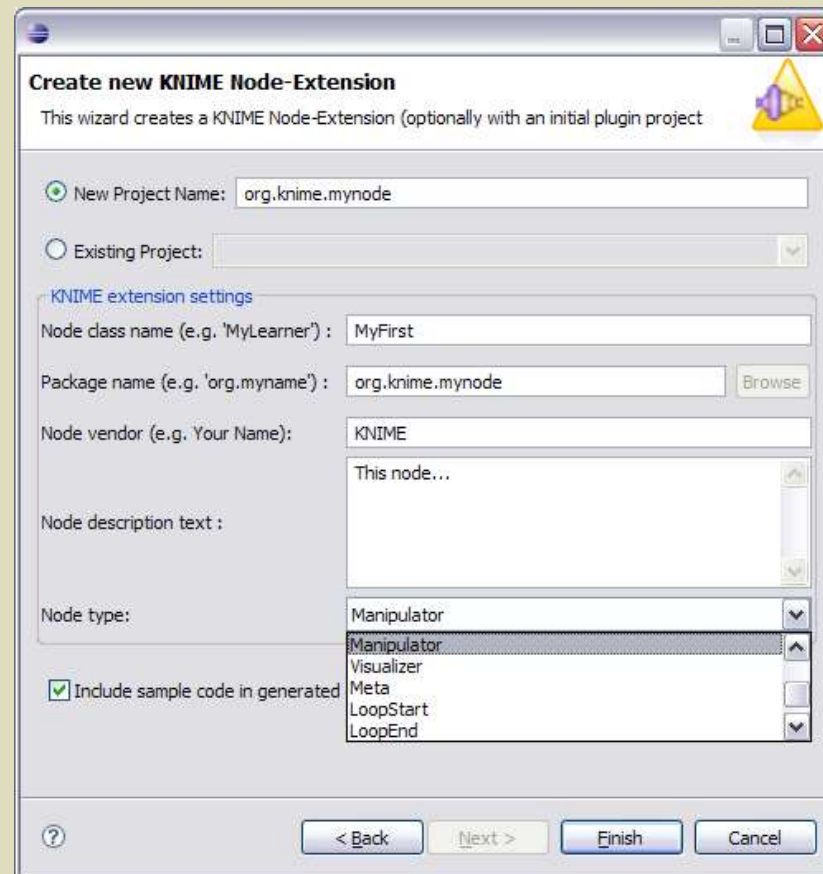
# KNIME SDK: user-defined extensions

- Eclipse programming framework
- KNIME extension points
  - “**Category**”: a folder in the node repository
  - “**Node**”: implementation of a specific algorithm or functionality
- New node wizard
- Metaprogramming paradigm: Java Reflection



➤ [www.knime.com/developers](http://www.knime.com/developers)

# KNIME SDK: new node wizard



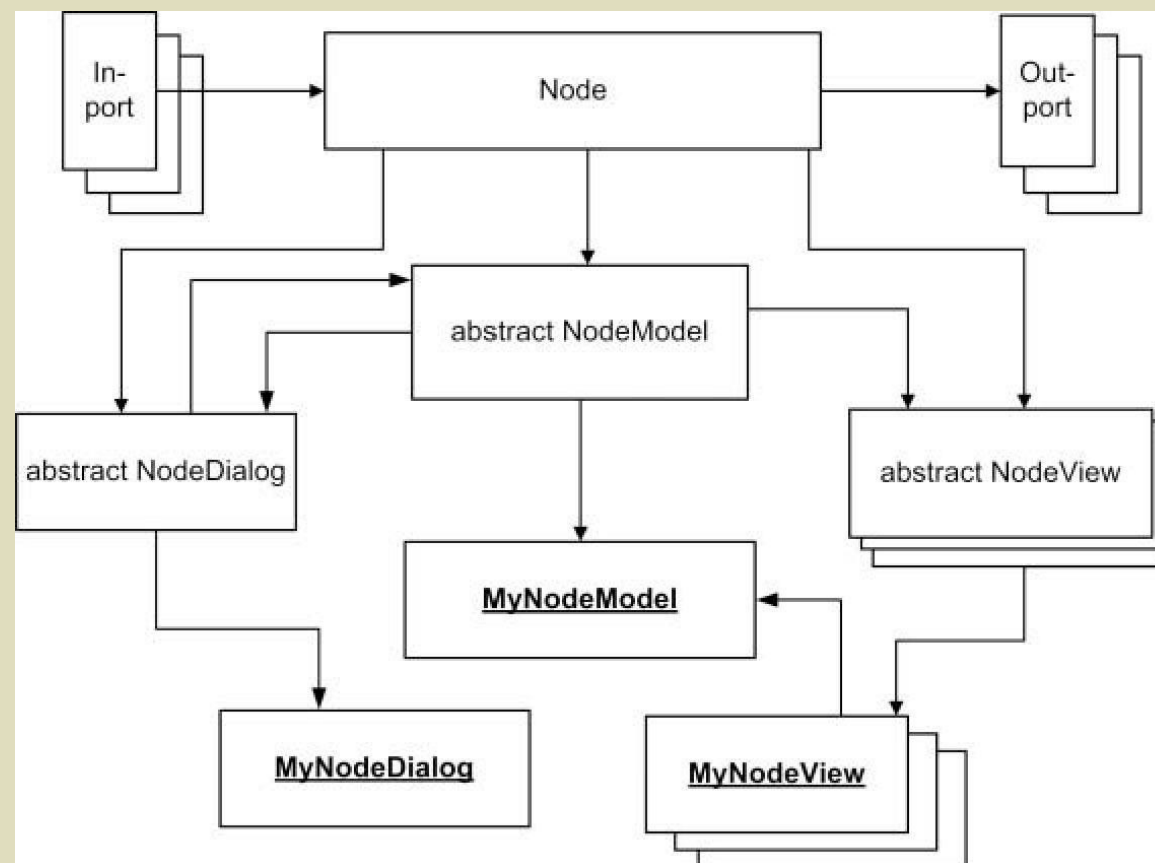
The screenshot shows a Java-style dialog box titled "Create new KNIME Node-Extension". Below the title bar, there is a subtitle: "This wizard creates a KNIME Node-Extension (optionally with an initial plugin project)".

The main content area is divided into sections:

- Project Selection:** Two radio buttons are present. The first, "New Project Name:", is selected and followed by a text field containing "org.knime.mynode". The second, "Existing Project:", is unselected and followed by a dropdown menu.
- KNIME extension settings:** A section header in blue text.
- Node class name (e.g. 'MyLearner'):** A text field containing "MyFirst".
- Package name (e.g. 'org.myname'):** A text field containing "org.knime.mynode" and a "Browse" button to its right.
- Node vendor (e.g. Your Name):** A text field containing "KNIME".
- Node description text:** A large text area containing the placeholder text "This node...".
- Node type:** A dropdown menu with "Manipulator" selected. The list of options includes: Manipulator, Visualizer, Meta, LoopStart, and LoopEnd.
- Include sample code in generated:** A checkbox that is checked.

At the bottom of the dialog, there is a row of buttons: a help button (question mark icon), "< Back", "Next >", "Finish", and "Cancel".

# KNIME Node Model



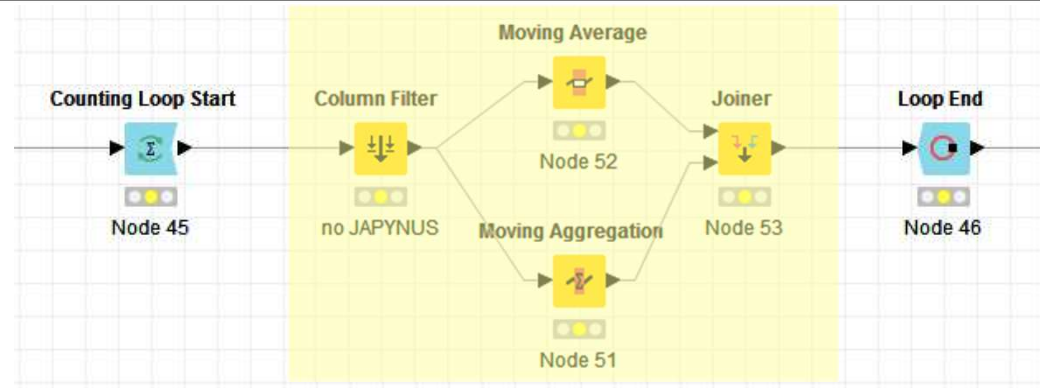
# Workflow Control Structures #1: Loops

## Looping

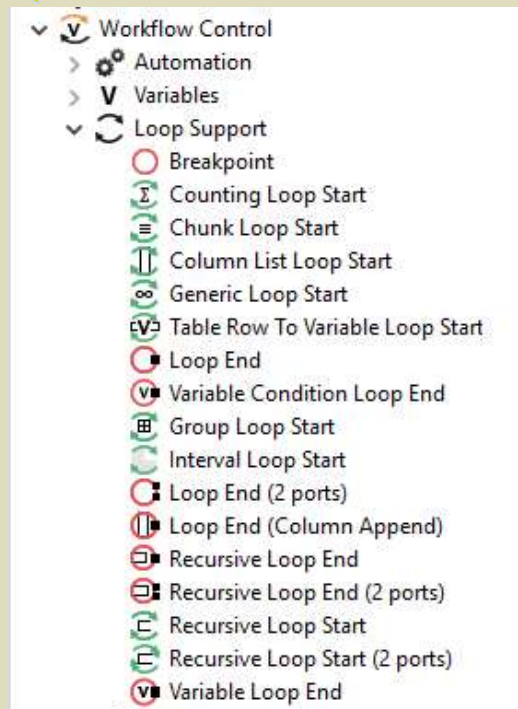
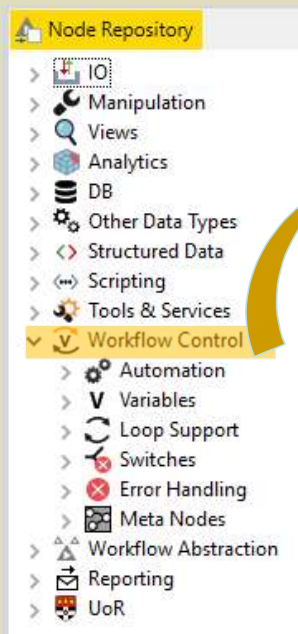
- A loop repeatedly executes the same processes
- Typical loop constructs are while and switch-case in Java

## Loops in KNIME

- all nodes are executed one after the other
- for a loop a start and a end is needed. (Hence, e./g. the do and the while)
- therefore specific loop start and end nodes are necessary



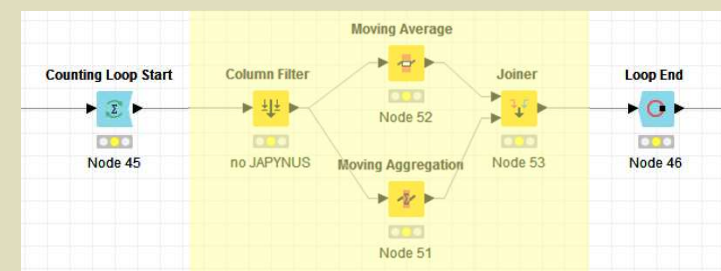
# KNIME Nodes for Loop Support



loop start  
node

Workflow block  
to be repeated

loop end  
node

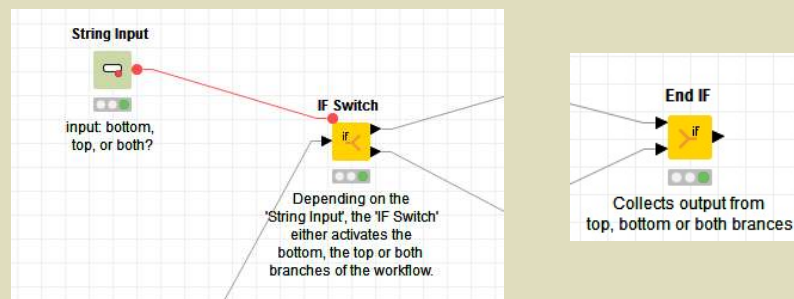


# Workflow Control Structures #2: Switches

- If Switch

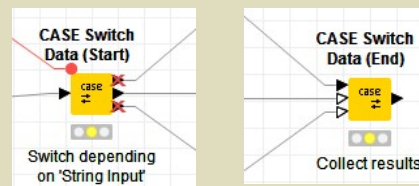
- See example at:

- <https://www.knime.com/nodeguide/control-structures/switches/using-a-if-switch>
    - Server Example: 06\_Control\_Structures/05\_Switches/02\_Using\_a\_If\_Switch



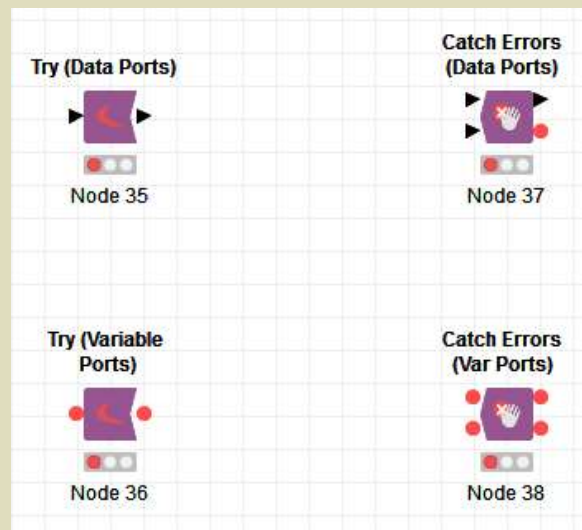
- CASE Switch

- <https://www.knime.com/nodeguide/control-structures/switches/case-switch>
  - Server Example: 06\_Control\_Structures/05\_Switches/01\_Case\_Switch



# Workflow Control Structures #3: Try-Catch

- Try-Catch
  - Used to handle workflow branches that may fail during execution (you do not know before execution they will fail).
  - Useful when it is not possible to know if a node will execute successfully.
    - For example, when connecting to an external Web Service
  - KNIME tries to execute the nodes in the main branch, but if one node in it fails, then it will fall back to a safe branch that provides some alternative output in order to continue the execution of the rest of the workflow.

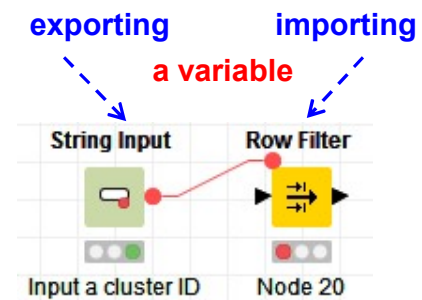
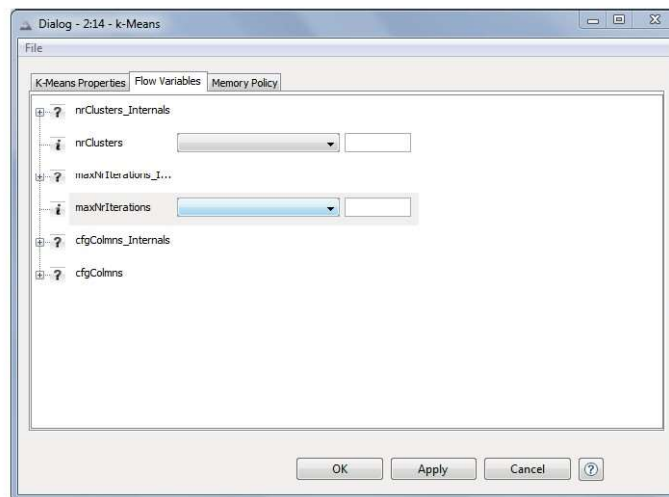




# Flow Variables

Flow Variables allow you to pass information between nodes using programming variables represented by red dots on nodes and red edges between nodes.

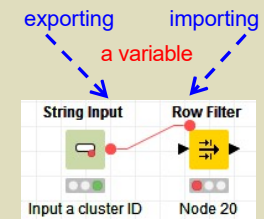
In most scenarios, we want to change something in the loop. This can be done via FlowVariables.



The Flow Variable Tab is available in each configuration pane.

# Flow Variables Declaration

- What is a Flow Variable and what is used for?
  - Programming variables that can be used to pass information between KNIME nodes (software components) within the same workflow.
  - One node declares (initialises) and export a new (named) variable: red circle
  - Drag a red edge from red circle to red circle
  - The importing node can use the variable to set one of its configuration parameters.
- How to create (declare and initialise) a Flow Variable
  - Flow Variables can be of type String, Integer, or Double
  - They can be created
    1. in the “Flow Variables” tab of any node
    2. using QuickForms nodes and other workflow abstractions
    3. using the node “Table Row to Variable” or “Table Column to Variable”



cell of 1<sup>st</sup> row is converted into a variable

- column name → var name
- 1<sup>st</sup> row value → var value
- other rows and other columns are not used: only the 1<sup>st</sup> row of a specified column.

Index	Owner ID	Name	Value
0	2:33	myStrVar	value1
0	2:33	RowID	row0
0		knime.workspace	C:\Users\giuseppe\Do

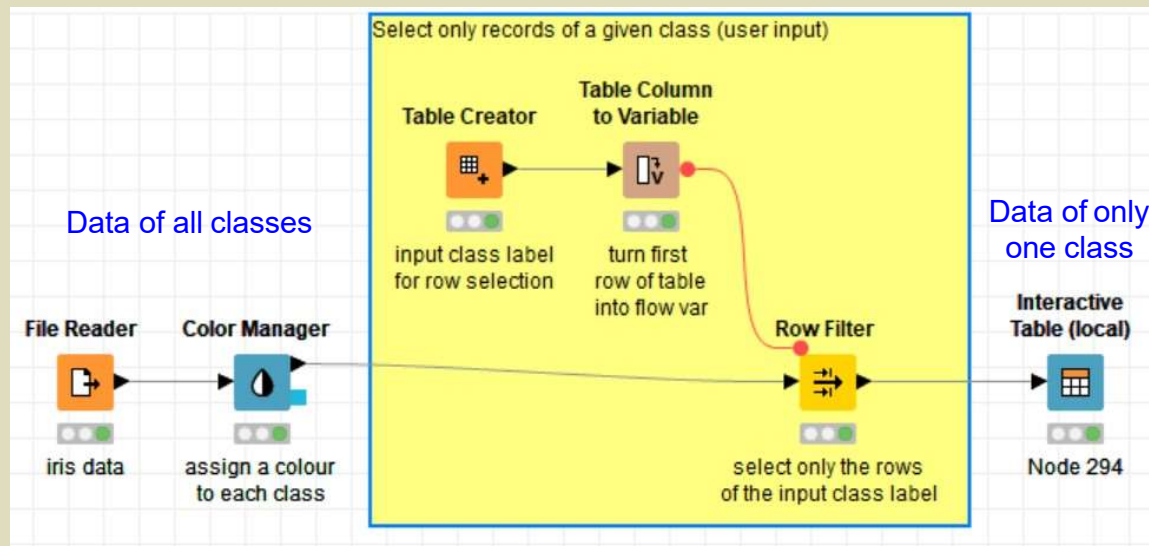
# Flow Variables Propagation

- How to propagate a Flow Variable
  - Implicitly: Flow Variables are carried along workflow branches (data flows, black edges connecting data ports). They are not implicitly shared with nodes in other workflow branches.
  - Explicitly: connecting the output Flow Variable port (red circle) of a node to the input Flow Variable port of a node by means of a red edge.
- A Flow Variable can be used to control loops.
- A Flow Variable can be used to set a parameter in some node settings.
  - This is done in the configuration dialog of the node.
- Workflow Abstractions (Quickforms) nodes allow to create a Flow Variable to parameterise a workflow Component (Wrapped Metanode).
  - The Flow Variable is used to set a parameter of some node inside the Component.
  - The configuration dialog of the Component inherits the configuration pane of the Abstraction node.

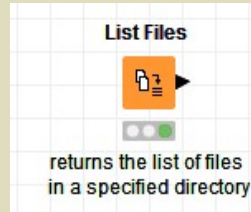
# Example: Flow Variables

## Example of use of a flow var to control 'Row Filter'

- Load some data
- Use the node “*Table Creator*” to create and edit an input table with some string value, e.g. “Iris-versicolor”
- Convert the value in the string input table to a **flow variable** by using the node “*Table Column to Variable*”.
- Use the node “*Row Filter*” to select only rows associated with the input string (one of the class values). The flow variable is used to control the parameter of the node “*Row Filter*” for the matching criterion in the configuration dialog.



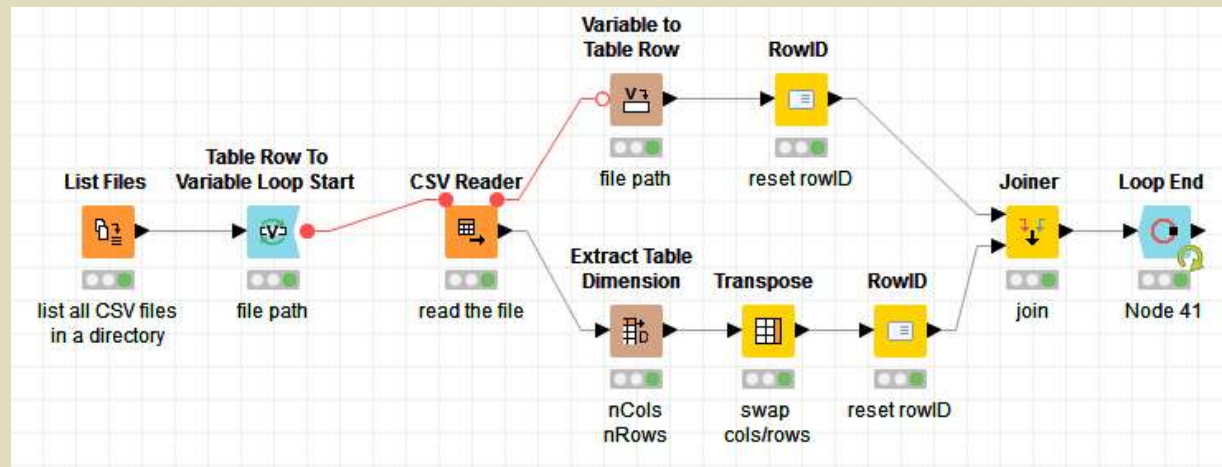
# Node “List Files”



- List all files in a directory
- Restricted to the top level directory (not recursive)
  - Specify file extensions to filter
  - Matching name patterns (regex or wildcard)
- The output provides file references as a data table of two (string) columns with URLs and absolute paths.

# Example: a Loop with a Flow Variable

- Extract the number of rows and columns of all CSV files in a directory.
  - At each iteration, convert the file path into a flow variable, read the file and collect the results.



Collected results - 2:41 - Loop End

File Hilite Navigation View

Table "default" - Rows: 5 Spec - Columns: 4 Properties Flow Variables

Row ID	Location	Number Rows	Number Columns	Iteration
Row0#0	C:\Users\giuseppe\Dropbox\AA_Work\Teaching\csdm16\data\all\adult.csv	32561	15	0
Row0#1	C:\Users\giuseppe\Dropbox\AA_Work\Teaching\csdm16\data\all\iris.csv	150	5	1
Row0#2	C:\Users\giuseppe\Dropbox\AA_Work\Teaching\csdm16\data\all\wine.csv	178	14	2
Row0#3	C:\Users\giuseppe\Dropbox\AA_Work\Teaching\csdm16\data\all\winequality-red.csv	1599	12	3
Row0#4	C:\Users\giuseppe\Dropbox\AA_Work\Teaching\csdm16\data\all\winequality-white.csv	4898	12	4



University of  
**Reading**

CSMDM16 - Data Analytics and Mining

# Modularisation in KNIME

Module convenor

**Dr. Carmen Lam**

[carmen.lam@reading.ac.uk](mailto:carmen.lam@reading.ac.uk)

Department of Computer Science

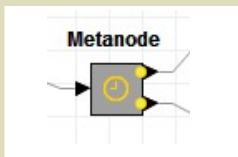
Lecture notes and videos created by

Prof. Giuseppe Di Fatta

# Metanodes

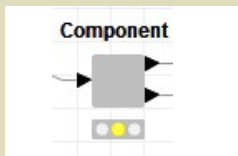
❑ Metanodes introduce **modularity** and **reusability** of workflows.

- They allow to build new reusable nodes that encapsulate an arbitrarily complex workflow inside them.
- Similar to user-defined functions (procedures, methods) in programming languages.



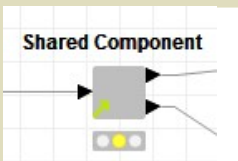
➤ Metanodes are the older and simpler version:

- they provide modularity and reusability at a basic level within a workflow.
- They do not have a “useful” configuration dialog.



➤ Components (Wrapped Metanodes) are Metanodes that can be configured via Quickforms/Abstractions:

- a Quickform node provides a proper configuration GUI for input parameters.
- Their input and output ports can be renamed.
- They can filter flow variables they import and export from/to the rest of the workflow they are used in.
- They are backward compatible: they can be used as simple metanodes.
- They can be used in the **KNIME Webportal (server)**: each wrapped metanode of a workflow generates one Web page on the KNIME Webportal.



➤ Shared Components (Metanode Templates) are stored Components

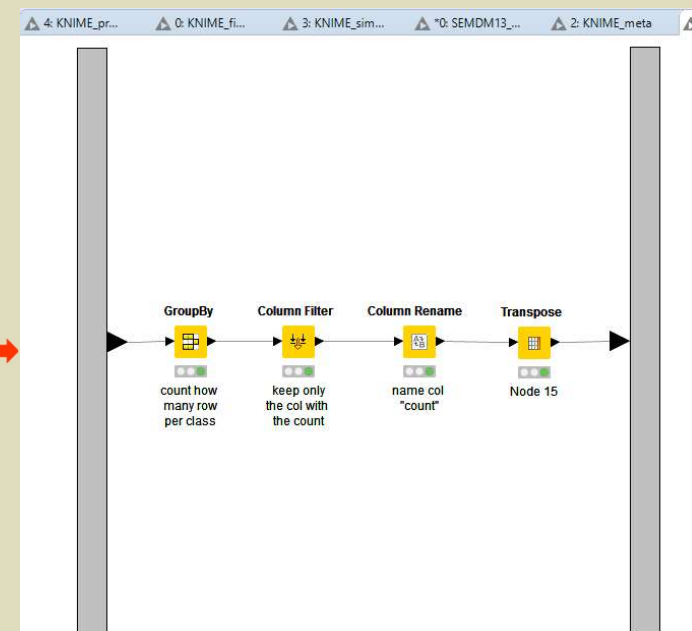
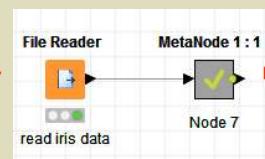
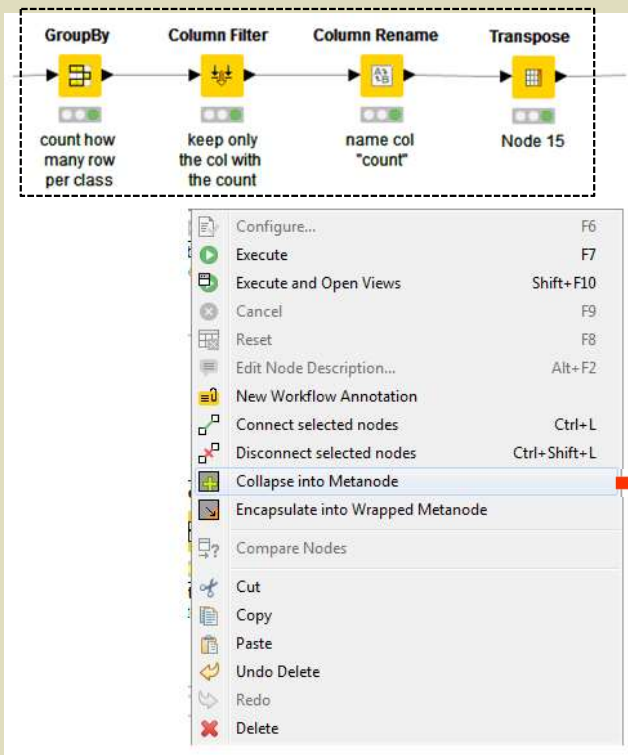
- the recommended way to share a metanode.
- They can be stored in your workspace or in a KNIME Server, ready to be reused.

[www.knime.com/blog/wrapped-metanodes-and-metanode-templates-in-knime-analytics-platform](http://www.knime.com/blog/wrapped-metanodes-and-metanode-templates-in-knime-analytics-platform)

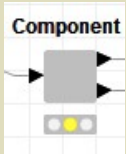


# Metanodes

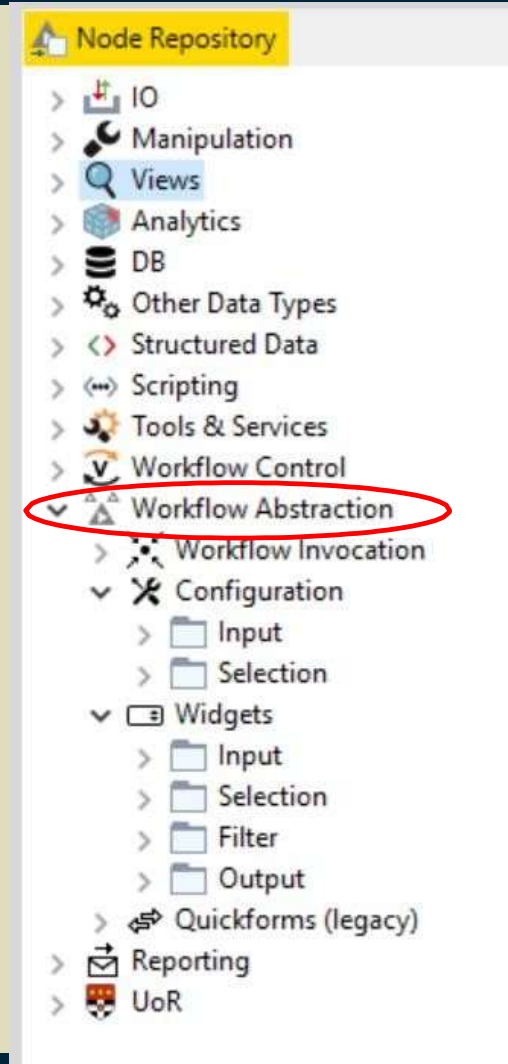
- Metanodes are used to create a new node from a workflow or part of it.
- Metanodes are nodes that contains a sub-workflow: a metanode looks like a single node, although it contains many nodes forming the sub-workflow.
- Metanodes may also contain other Metanodes.



# Workflow Components and Abstractions



- Abstractions allow to customise Workflow Components
  - **Configuration** define the parameters of the Component (the component's configuration),
  - **Component I/O** to define which data get passed in (or out) of the component,
  - **Widget nodes** are used to compose the component's view
  - **Invocation** mechanisms (e.g., nodes that can call local/remote workflows)

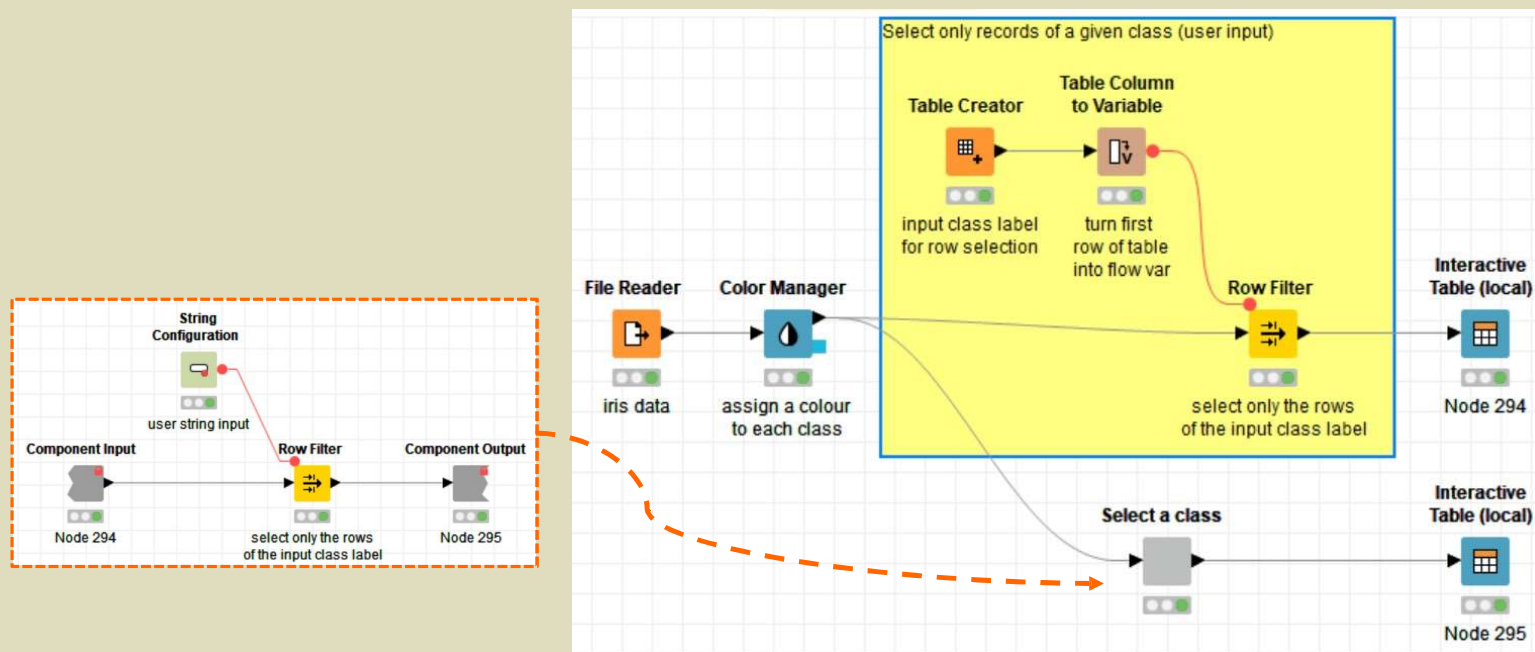


# Example: Flow Variables and Components

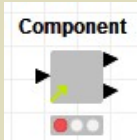
Given the previous example of use of a flow var to control 'Row Filter':

## Turn the solution into a reusable component:

- Perform the same row filter task using the Abstraction node “*String Input*” instead of the nodes “Table Creator”+“Table Column to Variable”.
- Wrap together the nodes “String Input” and “Row Filter” to create a new **Component** and name it “Select a class”.

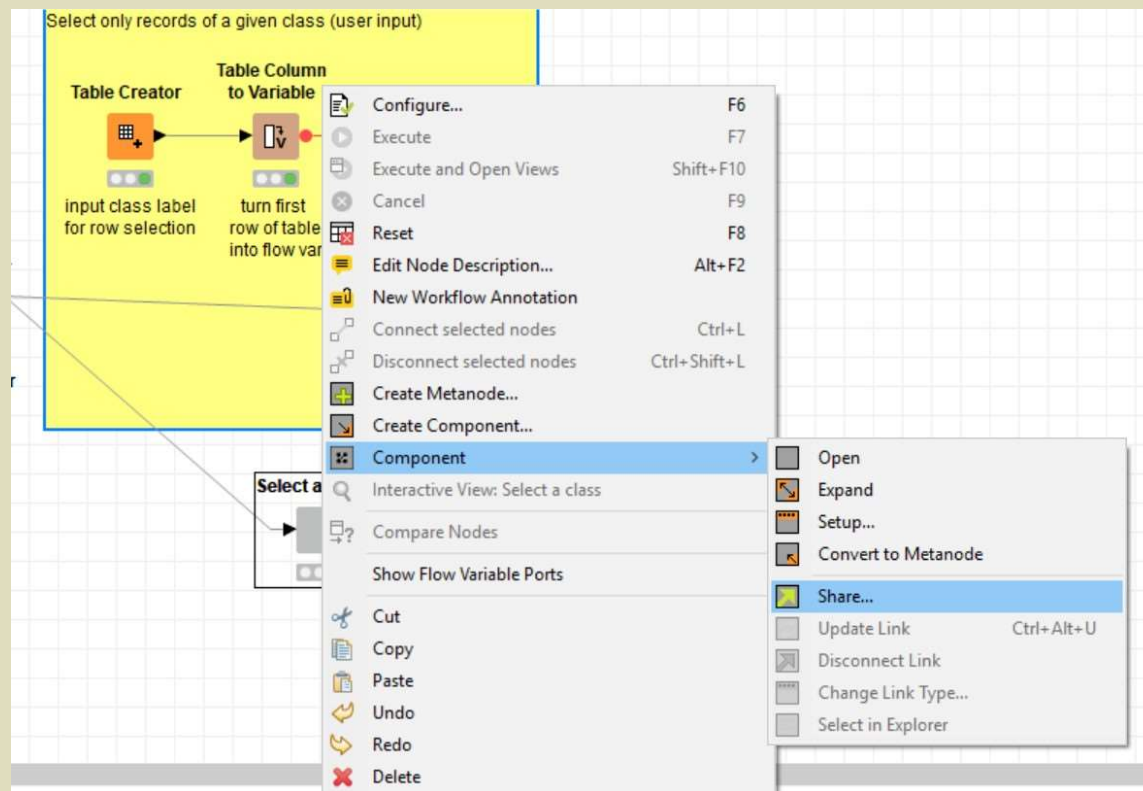


# Shared Component



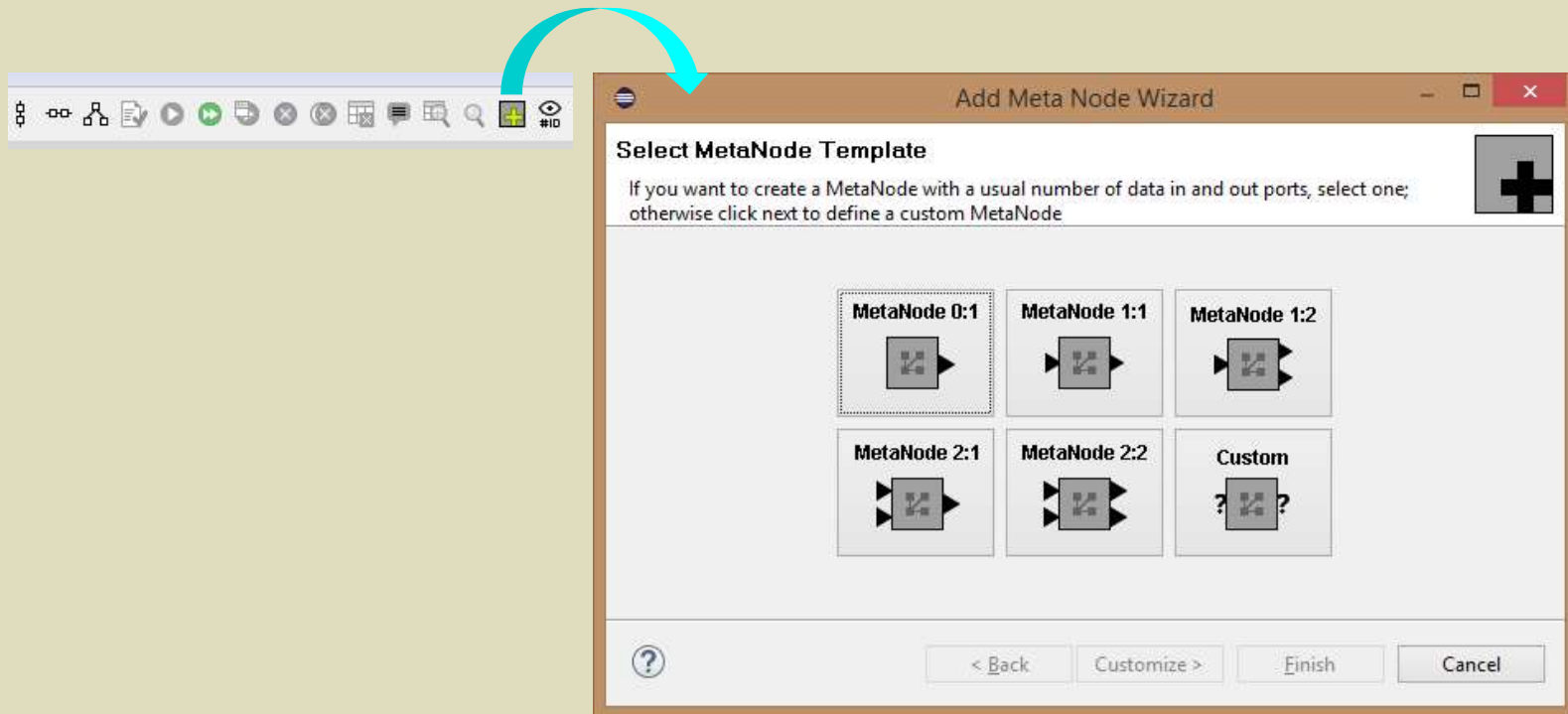
- A “Shared Component” is a Metanode that is stored into your local workspace for later reuse. (prev. called “Metanode Templates”)

The recommended way to share a metanode. A Metanode can become a Component and then can be stored (‘Shared’) to reside in your workspace or in a KNIME Server.



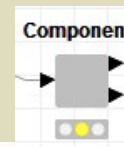
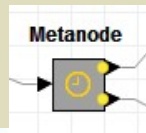
# The Metanode Wizard

- Metanode Wizard allows you to create an empty Metanode to work on.
- It has a number of predefined wizard templates with varying number of I/O ports: not to be confused with the “Metanode Template”, which is a stored Metanode.
- From the menu “Node” or from the toolbar:



# Metanodes vs Components

(Note: in KNIME v.4.0.0 “Wrapped Metanodes” have been renamed “Components”. Metanodes are still called Metanodes.)

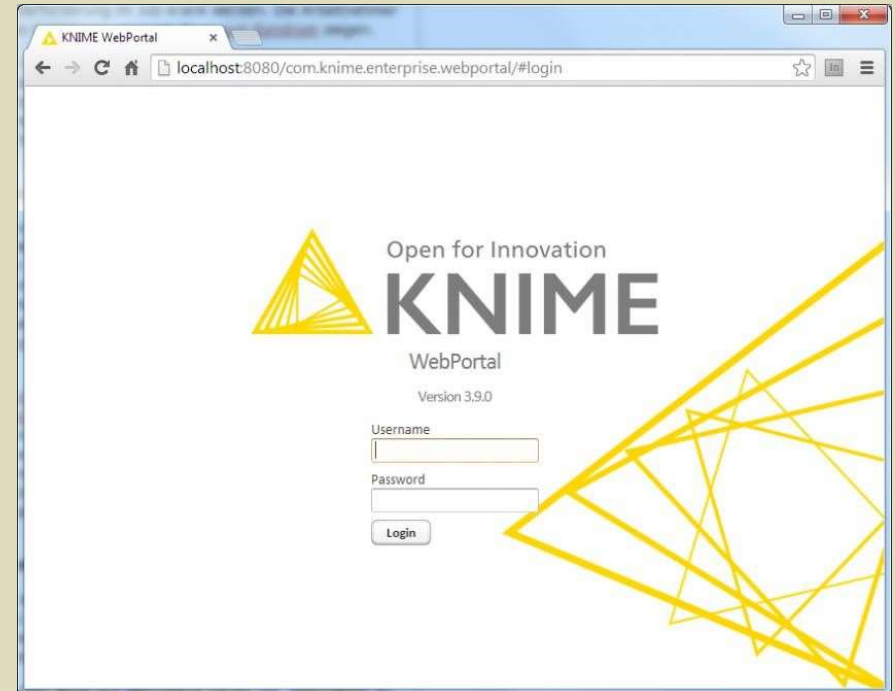


	Metanodes	Wrapped Metanodes*
Quick Forms	Legacy	Standard
Variable scope	Global	Local
WebPortal Execution	Old	New (work with loops/switches)
JavaScript views in WebPortal	Not supported	Supported
WebPortal Usage	Quickforms used globally	Views/Quickforms must be embedded in a Wrapped Metanode
Recommended uses	Legacy workflows	New developments
Compatibility	KNIME Server 3.x/4.1.x	KNIME Server 4.2+

\* Valid for KNIME Analytics Platform 3.1 and above

# Abstractions and Web Applications

- Abstractions nodes (prev. Quickforms) are typically used to create a flow variable and set the parameters of another node in the workflow.
- These nodes can also be used in conjunction with the [KNIME WebPortal \(Server\)](#) to rendered in the browser the interface for user input/output.
- The KNIME WebPortal allows to execute workflows interactively directly from a Web browser.
  - transfer the workflow into the KNIME server
  - use a web browser to run the workflow
  - the Abstraction nodes provide the user interface, e.g., to set the parameters of the workflow



Note: [KNIME Server](#) refers to the enterprise software for team-based collaboration, automation, management, and deployment of data science projects.

# Framework for Web Applications

- Abstraction, Flow Variable, Component and (optionally) Web application



Abstraction  
(configuration)

Flow Variable

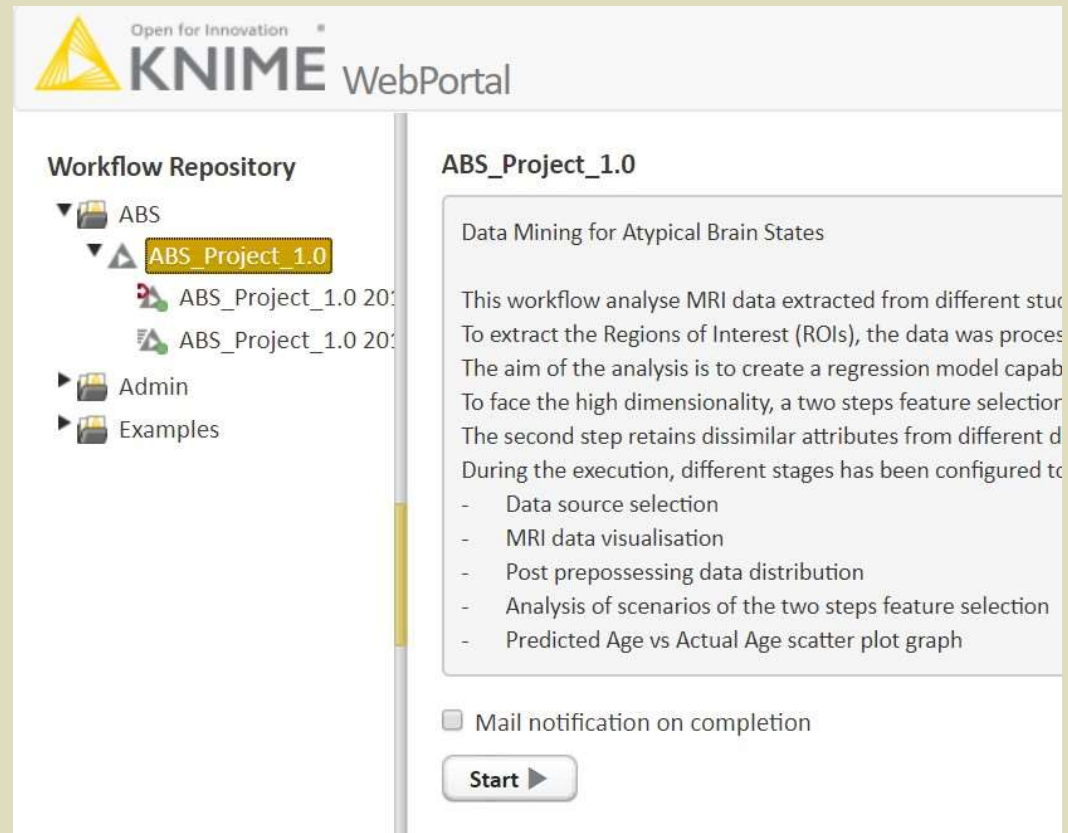
Workflow Component

Web application



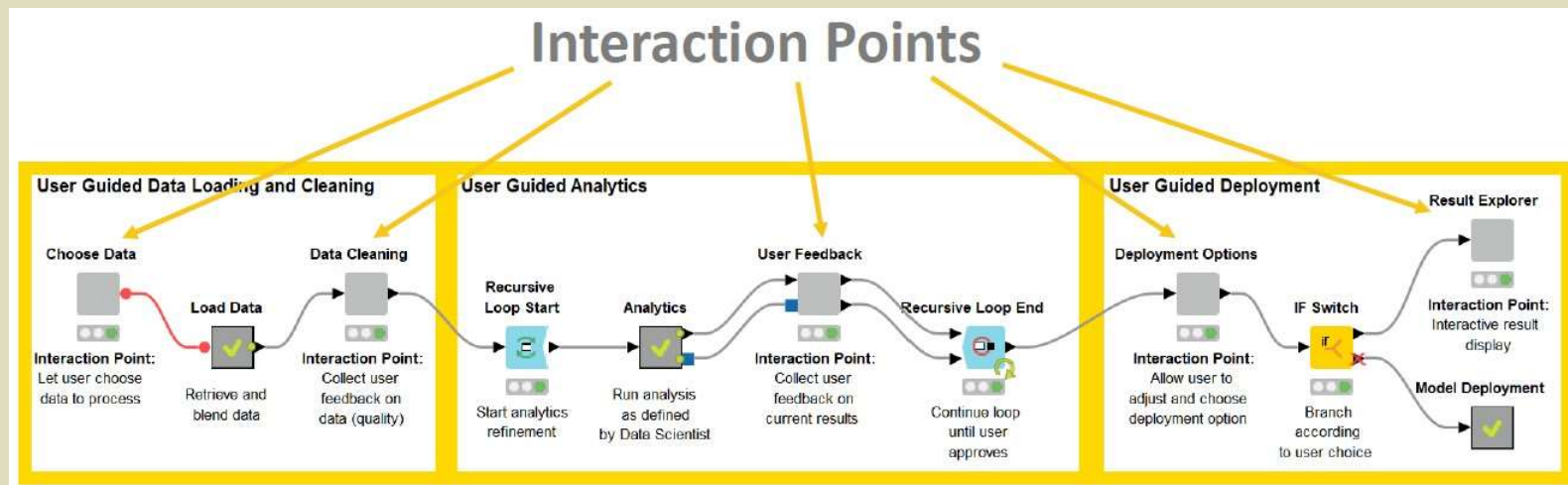
# KNIME Web Portal

- Workflows on KNIME Server as Web apps
  - Step-by-step execution of workflows from any browser
- Simple, clean interface for end users
  - ‘Guided Analytics’
- Customize layout to match corporate design

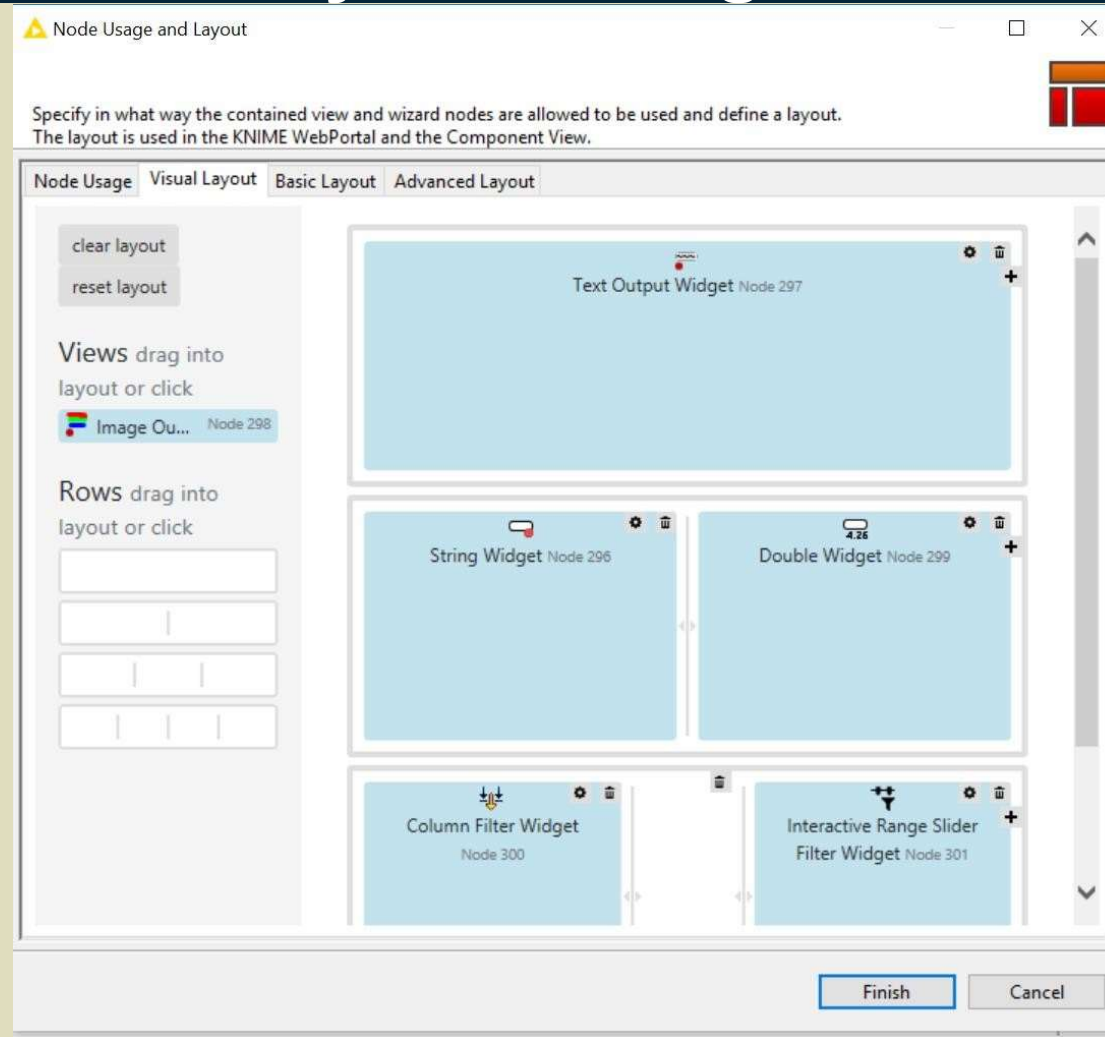


# Guided Analytics: Interactive Data Science

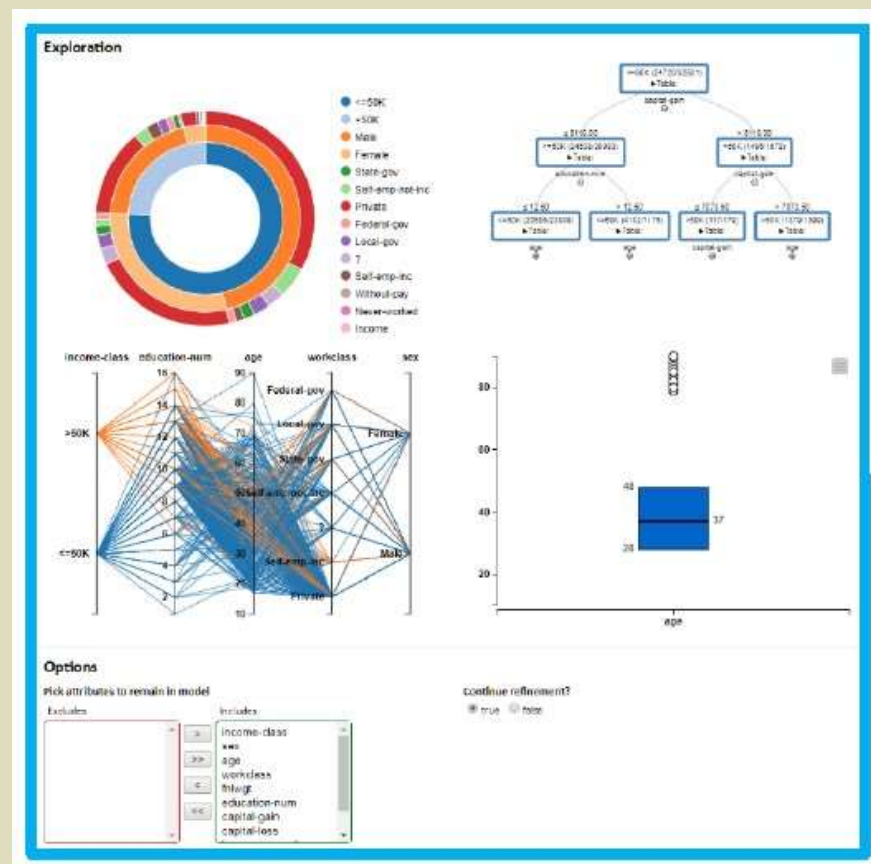
- The workflow 'Components' define the interaction points by means of the widgets embedded in them and a layout manager.



# Layout Manager



# Interaction Point Web Interface - Example





University of  
**Reading**

CSMDM16 - Data Analytics and Mining

# Advanced Predictive Analytics in KNIME

Module convenor

**Dr. Carmen Lam**

[carmen.lam@reading.ac.uk](mailto:carmen.lam@reading.ac.uk)

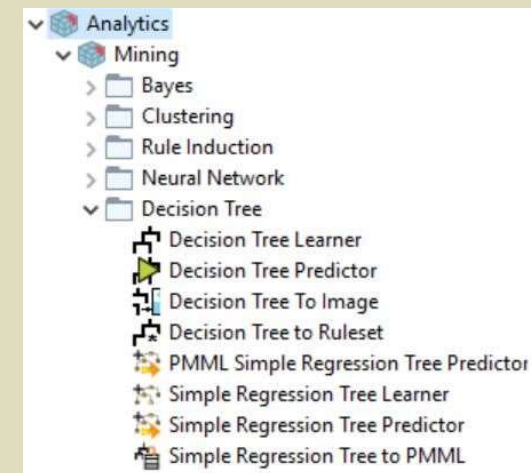
Department of Computer Science

Lecture notes and videos created by

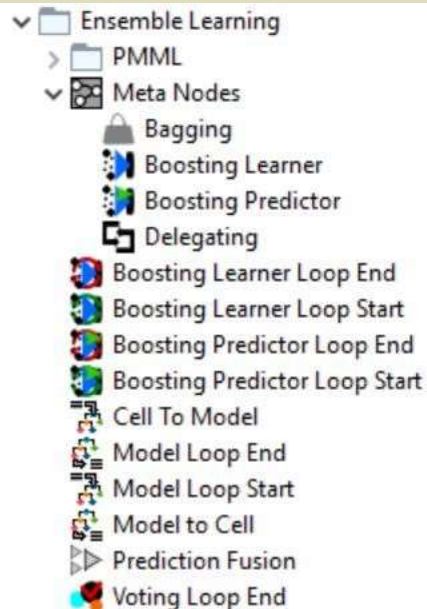
Prof. Giuseppe Di Fatta

# Advanced Predictive Analytics (Data Mining)

- Advanced analytics nodes for many data mining tasks
  - Many algorithms directly supported and many available via extensions/integrations
- Example of Classification methods
  - Decision Trees
    - Random Forest
    - Tree Ensemble
    - Gradient Boosted Trees
  - etc. etc. etc.
- Parameter optimization
- Hold-out and Cross-validation methods for estimating model performance



# Ensemble Methods

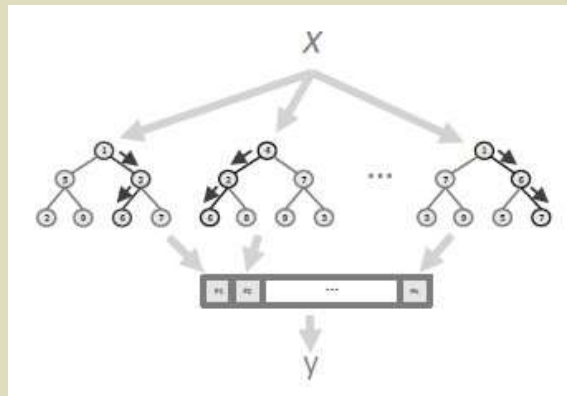


- Ensemble methods employ a set of learning algorithms (alternative predictive models) with some form of orchestration.
  - E.g., each model in the ensemble votes with equal weight.
  - They can achieve better predictive performance than can be obtained from each of the algorithms alone.
- Types of ensembles:
  - **Bagging** (Bootstrap aggregating)
    - attribute bagging
      - Each model is trained on a random subspace (attribute subset) to promote model variance (e.g. Random Forest).
    - sample bagging
      - Each model is trained on a random sample (with replacement) of the records in the training set.
  - **Boosting**
    - Start from training a model
    - Identify the training records that currently are not classified correctly
    - Iteratively train another model to correctly classified these cases and add it to the ensemble
    - ...and others.

# Tree Ensemble Models

## Bagging

- The general idea is to take advantage of the “wisdom of the crowd”
  - Ensemble models: Combining predictions from a large number of weak predictors, e.g. Decision Trees
  - Leads to a more accurate and robust model
  - In Classification (typically) the individual models vote and the majority wins.
    - Pick a different random subset of the training data for each model in the ensemble (bag)
  - In Regression (typically) the individual predictions are averaged.

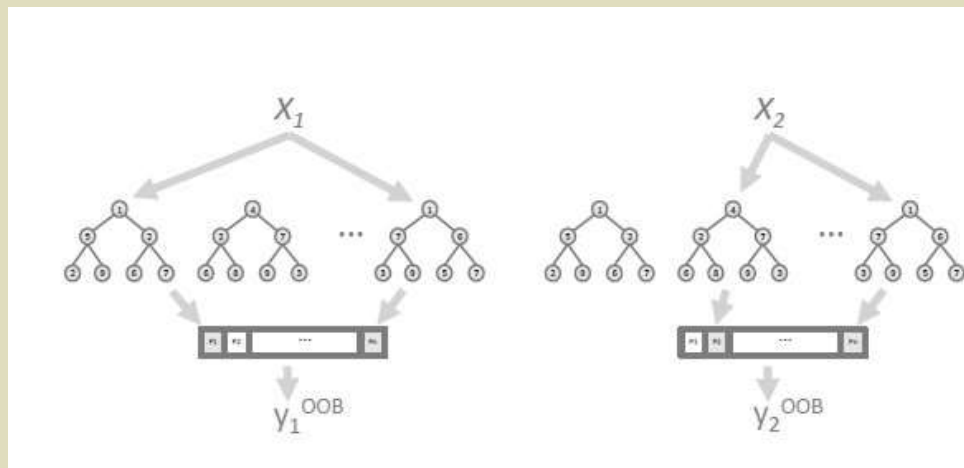




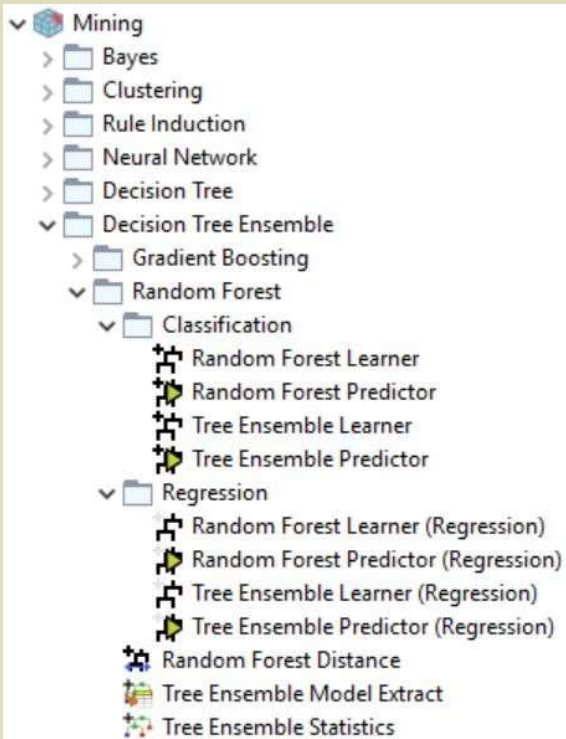
# Tree Ensemble Models

## Out of Bag Estimation

- Extra advantage: it allows testing the model using the training data:
- When validating, each model should only vote on data points that were not used to train it.



# Tree Ensemble Models in KNIME



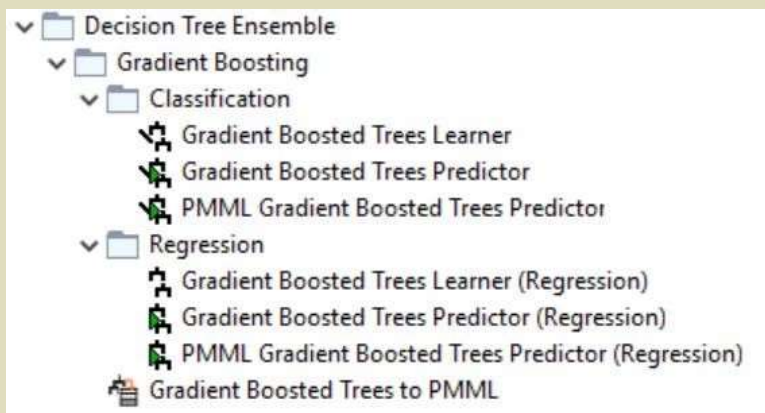
- Nodes for “Random Forest”:

- bag of decision trees, with an extra element of randomization when building the trees: each node in the decision tree adopts a subset of the input columns.
- Advantages:
  - Random forests tend to be robust w.r.t. overfitting.
    - Although the individual trees are almost certainly overfit
  - Training each model is faster than using all attributes (Decision Tree).

- Nodes for “Tree Ensemble”:

- a Random Forest variant providing both attribute and record bagging: random subspace method and bootstrap aggregating
  - Trees may be trained using subsets of rows and/or columns
  - It may lead to greater accuracy.
  - Complex model optimisation because of the many parameters:
    - number of models, number of columns, number of rows, tree depth, etc.

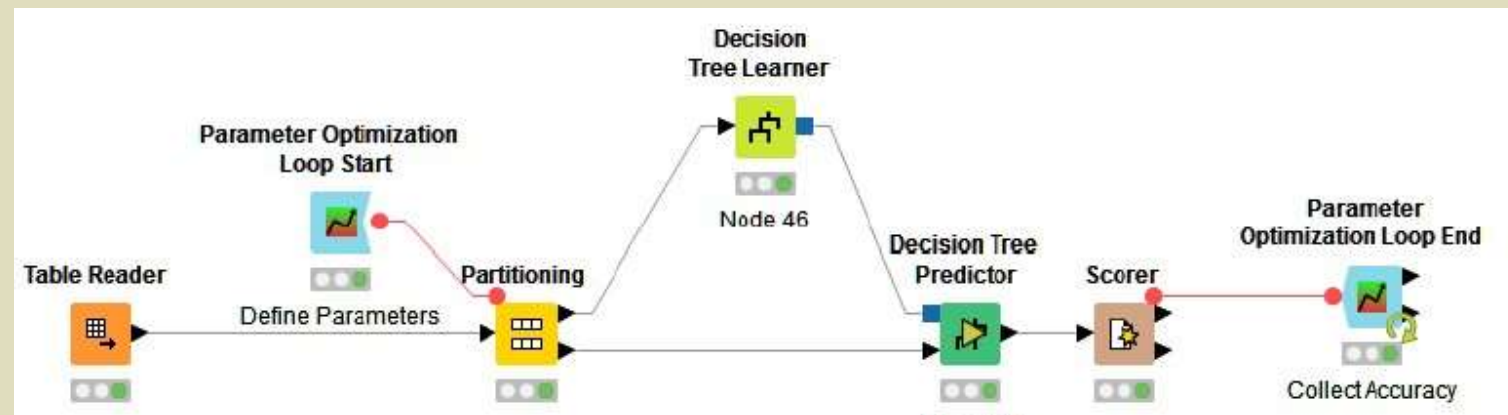
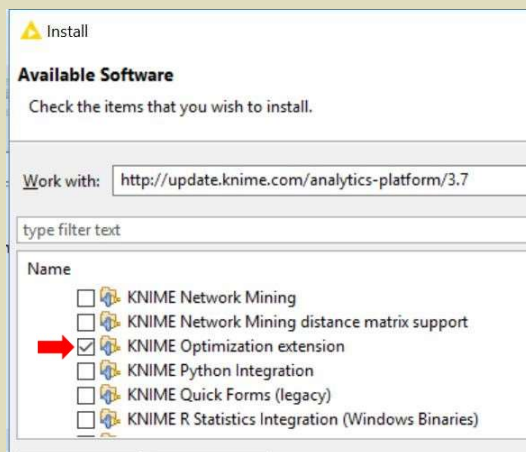
# Tree Ensemble Models in KNIME



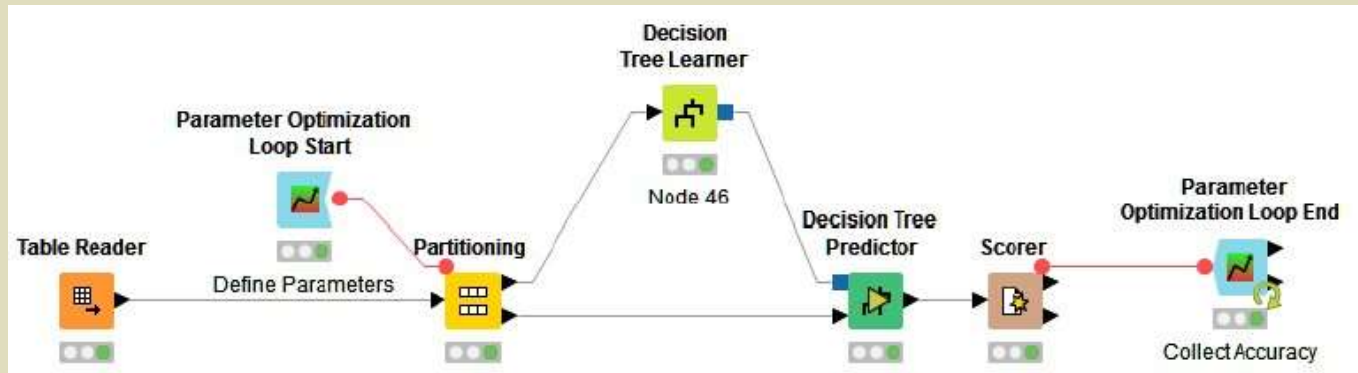
- Nodes for “Gradient Boosted Trees”:
  - Another meta-heuristic for creating ensembles of decision trees
    - Starts with a tree built on a subset of the data
    - Builds additional trees to fit the residual errors
    - Typically uses fairly shallow trees
    - Can introduce randomness in choice of data subsets (“stochastic gradient boosting”) and in variable choice (advanced options)

# Parameter Optimization (KNIME Extension)

- Model learners can be sensitive to their configuration parameters.
  - Calculating “optimal” settings is not always easy or even possible.
  - The **Parameter Optimization Loop** may help to find a good configuration.



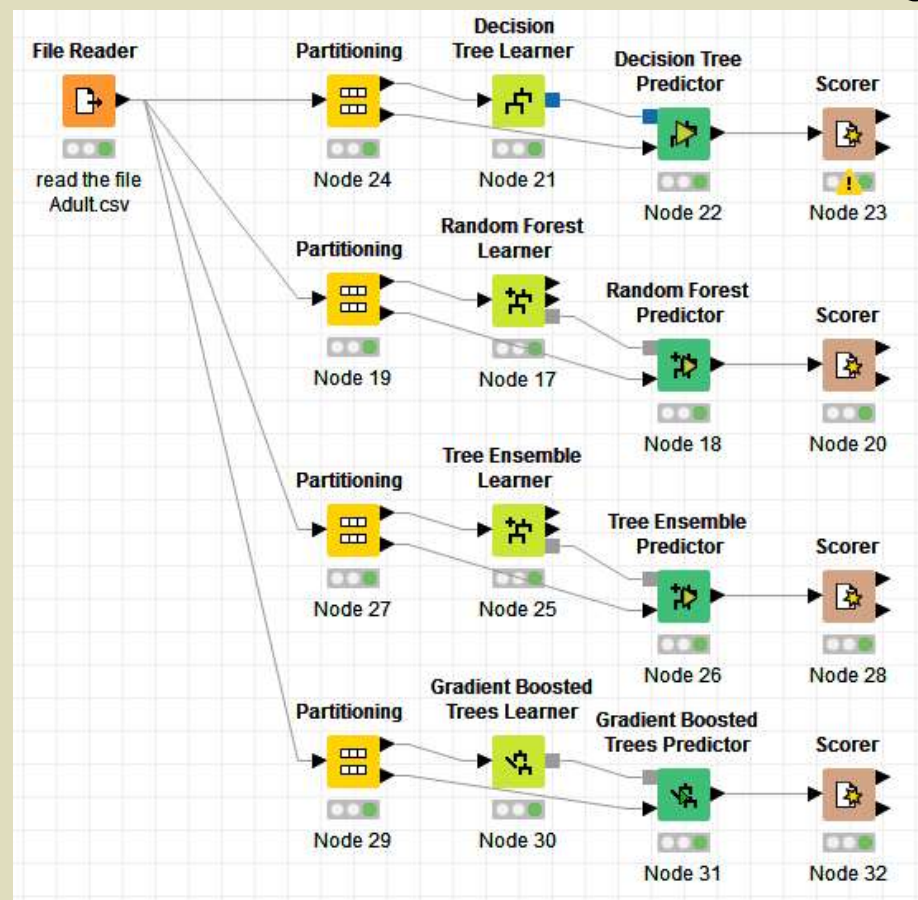
# Parameter Optimization (KNIME Extension)



- Searching for optimal values of configuration parameters.
  - Calculating “optimal” settings is not always easy or even possible.
  - The **Parameter Optimization Loop** may help to find a good configuration.
  - Loop Start:
    - choose the parameters to be optimised
    - set upper/lower bounds and step sizes (and flag integers)
    - choose an optimisation method
      - **brute force** for maximum accuracy but slower computation
      - **hill climbing** for faster runtimes but may get stuck in local minima
  - Loop End:
    - collects some value to be optimised in a flow variable.
    - the value may be maximised (e.g., accuracy) or minimised (e.g., error).

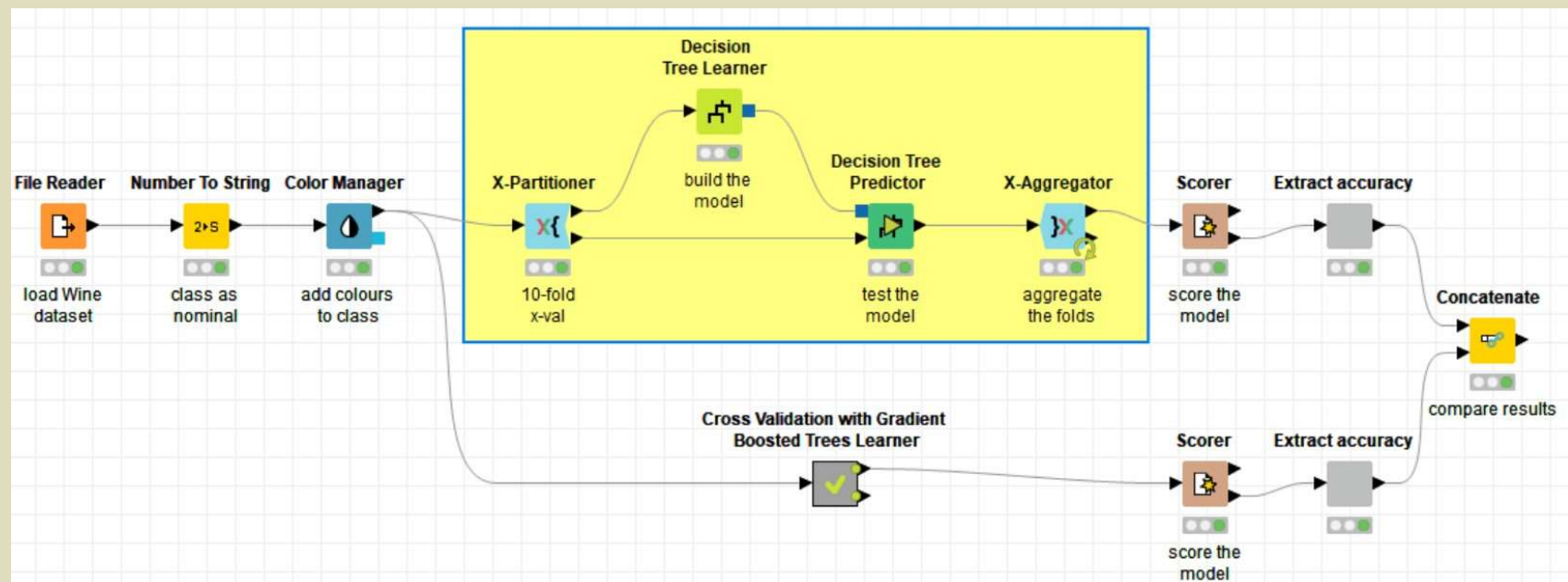
# Model Evaluation: Hold-out

- Hold-out method: the node 'Partitioning'



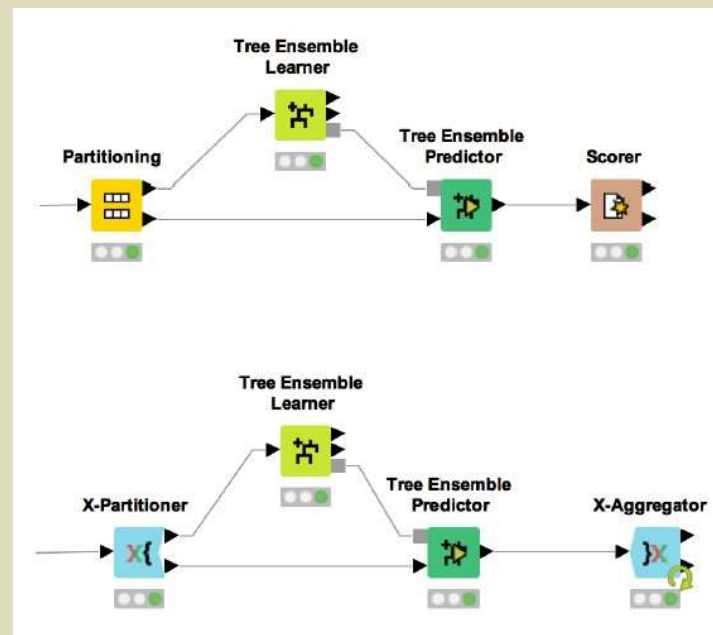
# Model Evaluation: Cross-validation

- The cross-validation method (e.g., 10-fold xVal or LOOCV):  
X-Partitioner and X-Aggregator create a loop to repeat training and testing for each fold.



# Hold-out and Cross-validation

- (For more details on this topic please refer to the lecture on Model Evaluation.)
- The hold-out method: use the node “Partitioning”
- The cross-validation method (10-fold xVal or LOOCV): use the loop nodes “X-Partitioner” (loop start) and “X-Aggregator” (loop end) to create a loop for repeating training and testing for each fold.



- How can you implement the resubstitution method?



# Final Conclusions on KNIME

- Strong vision for 'end to end' Data Science
  - KNIME Analytics Platform: open source, GPL license
  - KNIME SDK (Eclipse): extendibility
  - KNIME Server and Web Portal: deployment via Web apps and REST
- User-friendly workflow management system with modularity
- Two main releases a year for
  - enhanced GUI and performance, more algorithms and features
- Open platform
  - seamless integration of many popular tools, libs, languages
- Automation
- 'Guided Analytics' application authoring
- Large community with contributions of workflows and nodes

## Next:

- P05: practical on Advanced KNIME

## Next week:

- Introduction to R