Cheat Sheet: Building a KNIME Workflow for Beginners

Open for Innovation
KNIME

Getting started with KNIME Analytics Platform

- Read through the installation guide at knime.com/downloads
- Check out the 7 things you should do after installing KNIME Analytics Platform at www.knime.com/blog/seven-things-to-do-after-installing-knime
- Take the E-Learning Course at www.knime.com/knime-self-paced-courses

Understanding the traffic light system:

Not configured: Node is not yet configured and cannot be executed with its current settings

Configured: Node has been correctly configured and may be executed at any time

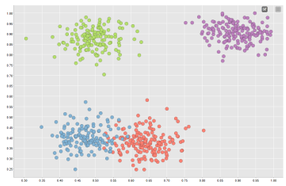
Executed: Node has been successfully executed and results can be viewed and used in downstream nodes.

Dynamic ports: Additional input ports can be added by clicking the three dots in the bottom left corner of a node.

EXPLORE

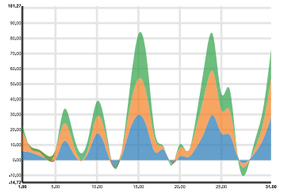
(All visualizations are interactive)

Scatter Plot: Represents input data rows as points in a two dimensional plot. Input dimensions (columns) on the x-y axis plot and graphical properties can be changed in the configuration window or interactively in the node view.



Sunburst Chart: Displays categorical columns through a hierarchy of rings. Each ring is sliced according to the nominal values in the corresponding column and to the selected hierarchy. This is a powerful chart for multivariate analysis.

Stacked Area Chart: Plots multiple numerical data columns on top of each other using the previous line as the base reference. The areas in between lines are colored for easier comparison. This chart is commonly used to visualize trending topics.



Line Plot: Plots numerical values in data columns (y-axis) against values in a reference column (x-axis). Data points are connected via colored lines. If the reference column on the x-axis contains sorted time values, the line plot graphically represents the evolution of a time series.

Color Manager: Assigns a color property to each input row based on the row's value in a selected column. This color property affects the graphical representation in the upcoming views.

Pie Chart: Visualizes one aggregated metric for different data partitions with colored slices on a circle where the areas are proportional to the metric values. The partitions are defined by a categorical column.

Data Explorer: Provides an interactive view to summarize the statistics of the input data via statistical measures and histograms - for both numerical and nominal columns.

Box Plot: Visualizes numeric columns using the quartile statistics. Watch out for the points at the end of the whiskers - they might mark outliers!

Bar Chart: Visualizes one or more aggregated metrics for different data partitions with rectangular bars where the heights are proportional to the metric values. The partitions are defined by a categorical column.

READ

CSV Reader: Reads CSV files. It has an auto-detect function to automatically guess the file structure. As for other reader nodes, clicking the three dots in the lower left corner allows to add one input port to connect to external data sources.

Excel Reader (XLS): Reads content from sheets in Excel files (XLS, XLSX). Sheet and cells to be read can be defined in the configuration window.

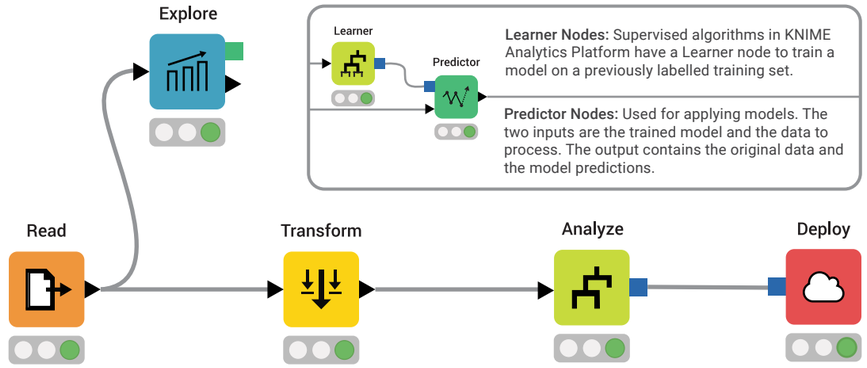
Table Creator: Allows users to manually create a data table in its configuration window as a data sheet. Data cells can be copied and pasted in the sheet. Perfect for generating small data sets.

Model Reader: Reads machine learning models generated with any of the Learner nodes. Models are usually saved after training and reused in deployment.

Table Reader: Reads data from a .table file. .table files are organized using a KNIME proprietary format, including the full file structure and are optimized for space and speed - providing maximum performance with minimum configuration!

Google Sheets Reader: Reads data from a Google Sheet file. Authentication occurs on the Google site. Google credentials are not saved within the KNIME workflow.

In reader and writer nodes, the file path is expressed relatively to a key location of the current KNIME installation, like workflow, workflow data area, and mountpoint.



Learner Nodes: Supervised algorithms in KNIME Analytics Platform have a Learner node to train a model on a previously labelled training set.

Predictor Nodes: Used for applying models. The two inputs are the trained model and the data to process. The output contains the original data and the model predictions.

ANALYZE

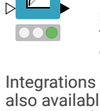
Decision Tree: The Learner node trains a C4.5 or a CART decision tree. The configuration window includes options for pruning, early stopping, information measures, splitting values, and more. Both the Learner and the Predictor node provide an interactive view where the decision tree is displayed together with the input data propagation.

k-Means: Implements the k-Means clustering algorithm. Number of clusters must be set prior to node execution. This node builds the clusters. The Cluster Assigner node finds the closest cluster and assigns it to the input data row. Being an unsupervised algorithm, this node pair doesn't follow the classic Learner - Predictor scheme.

Logistic Regression: The Learner node trains a logistic regression model to predict categorical target values. The configuration window includes options for solver, input feature choice, regularization functions to avoid overfitting, & more.

Scorer: Calculates a number of performance measures such as accuracy, F1-score, or Cohen's Kappa, to quantify the quality of a classifier.

Numeric Scorer: Calculates a number of numerical error measures, such as root mean squared error, mean absolute error, or R^2, to quantify the quality of a numerical predictor model.



ROC Curve: Displays the Receiver Operating Characteristic (ROC) curve of a classifier working on a binary class problem. One of the two classes is arbitrarily chosen as the positive class and the ROC curve is built on the probabilities/scores produced for that class on the input data set.

Integrations to many open source data analytics tools are also available. Some use the KNIME node GUI (H2O, Weka, Keras, Spark MLlib). Others offer nodes with a development environment for scripting and debugging (R, Python, Java).

TRANSFORM

GroupBy: Groups the rows of a table by the unique values in selected columns and calculates aggregation and statistical measures for the defined groups. Despite its simple name, it offers powerful functionality and has many unsuspected usages. For example - row deduplication.

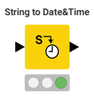
Pivoting: Extends the aggregation functionality of the GroupBy node by creating an output data table with columns and rows for the unique values in selected input columns. Note: the unique values of the grouping column become rows and the unique values of the pivoting column become columns.

Rule Engine: Applies a set of rules to each row of the input data table. All Rule Engine operators are also available in the Column Expressions node.

Partitioning: Splits data into two subsets according to a sampling strategy. This node is generally used to produce a training and a test set to train and evaluate a machine learning model.

Row Filter: Filters rows in or out from the input data table according to a filtering rule. The filtering rule can match a value in a selected column or numbers in a numerical range.

Math Formula: Implements a number of math operations across multiple input columns, from simple sum and average, to logarithms and exponentials. All Math Formula operators are also available in the Column Expressions node.



String to Date&Time: Converts values in a String column into Date&Time values. The Date&Time format contained in the String values can be manually defined or auto guessed.

Cell Splitter: Splits values in a selected column into two or more substrings, as defined by a delimiter match. Delimiter is a set character, such as a comma, space, or any other character or character sequence.

Column Filter: Filters columns in or out from the input data table according to a filtering rule. Columns to be retained can be manually picked or selected according to their type, or of a regex expression matching their name.

Column Rename: Assigns new names and types to selected columns, as configured in the dialog.

Joiner: Joins rows from two data tables based on common values in one or more key columns. The output - inner join, left outer join, right outer join, full outer join, or the respective antijoins - can be split into multiple output tables.

Sorter: Sorts the table in ascending or descending order based on the values of a chosen column. In addition, it is possible to sort based on multiple columns.

Concatenate: Merges two or more data tables vertically by piling up cells in columns with the same name. Cells in not overlapping columns are filled with missing values.

Missing Value: Defines a strategy to deal with missing values in the input data table - either globally on all columns, or individually for each single column.

String Manipulation: Performs operations on String values in columns, such as combining two or more Strings together, extracting one or more substrings, trimming blank spaces, and so on. All operators are also available in the Column Expressions node.

DEPLOY

Data to Report: Marks the data table to be exported to BIRT - a partially open source reporting tool integrated within KNIME. When switching from KNIME to BIRT, the marked data sets are imported into BIRT. The Image To Report node marks the input images to be exported to BIRT.

Excel Writer (XLS): Writes the input data table to a sheet in an Excel file (XLS or XLSX).

Table Writer: Writes the input data table to a file using the .table KNIME proprietary format. This format includes the full file structure and is optimized for space and speed. Including the table structure in the file is a great advantage - especially when exchanging data files among users.

CSV Writer: Writes out input data table into a CSV file or to a remote location denoted by an URL

Google Sheets Writer: Writes the input data table into a Google Sheet file. Authentication occurs on the Google site. Google credentials are not saved within the KNIME workflow.

Connectors to Tableau: Export input data table into a Tableau file or server for reporting.

Resources

- **KNIME Forum:** Join our global community and engage in conversations at forum.knime.com
- **KNIME Books:** More tips, ideas, and lessons from knime.com/knimepress
- **KNIME Events:** Take a course, attend a workshop, or join a meetup at knime.com/events
- **KNIME Blog:** Engaging topics, challenges, industry news, and knowledge at knime.com/blog
- **KNIME Hub:** Browse and share workflows, nodes, and components, with the KNIME community. Add ratings, or comments to other workflows at hub.knime.com
- **More Guides:** Still using SAS or Excel? Transition to KNIME Analytics Platform with these handy guides at knime.com/knimepress
- **KNIME Server:** For team-based collaboration, automation, management, and deployment check out KNIME Server at knime.com/knime-server
- **Beginners Space on KNIME Hub:** Find the collection of example workflows using these cheat sheet nodes.

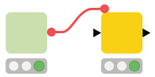
tinyurl.com/KNIME-Beginner

Cheat Sheet: Control and Orchestration with KNIME Analytics Platform

Open for Innovation
KNIME

CONTROL

Flow Variables allow for the parameterization of a workflow. A Flow Variable is a parameter that can assume different values at different execution points in the workflow & overwrite configuration settings in upcoming nodes.



Hidden Flow Variable Ports

Each node has two hidden Flow Variable ports to accept incoming Flow Variables & to propagate them to the upcoming nodes. To make these ports visible, right-click the node & select ***Show Flow Variable Ports***. Only ports of the same type can be connected.

Creating a Flow Variable

1. Use a Configuration or a Widget node to create a Flow Variable at any point in your workflow.
1. Use any of the nodes converting data into Flow Variables.
3. Via the node configuration window in the ***Flow Variables*** tab, fill in a blank box with the name of the Flow Variable.

String Manipulation (Variable): This node is the Flow Variables version of the String Manipulation node. Similarly, other nodes have their own version for Flow Variables like the Rule Engine Variable node & the Math Formula (Variable) node.

Merge Variables: Combines Flow Variables from two or more separate branches. To add a branch click the three dots in the bottom left corner. If Flow Variables with the same name are collected, the Flow Variable in the top most connection is retained.

Inject Variables (Data): Adds (injects) the Flow Variables at its Flow Variable input port into the data table at the top input port. The input data table is forwarded (unaltered) in the node output port.

Extract Variables (Data): Extracts the Flow Variables coming in through the input data port & produces them as standalone Flow Variables at the output port.

Widget and Configuration nodes create one or more new flow variables & make them available at the output port. Widget nodes create a UI item for the composite view or the KNIME WebPortal (textbox, radio button, etc.) to create and control the flow variable. Configuration nodes create a UI item in the configuration dialog of a component & are not visible in the composite view or the KNIME WebPortal.

Column Selection Widget: Creates a list of selectable columns from the input data table in the form of a menu or radio buttons. The node produces the name of the selected column in a flow variable at its output port.

Interactive Range Slider Filter Widget: Creates a slider to filter data to only include rows with values in the selected column within the specified range. The slider can interact with views from other JavaScript based nodes in the same composite view.

Text Output Widget: Creates a paragraph of either free, preformatted, or HTML text.

Credentials Widget: Creates fields to enter credentials (username & password) in the form of text boxes. The text in the password box is masked. The node produces these credentials in a flow variable at its output port.

Single Selection Configuration: Creates a list of options of type String in the form of a menu or radio buttons. These options can be defined in the configuration dialog together with the selected default value. The node produces the value of the selected option in a flow variable at its output port.

Boolean Configuration: Creates a boolean selection for an enabled/disabled flag (1/0) in the form of a checkbox. The node produces the value of the selected option in a flow variable at its output port (1 if the checkbox is enabled, 0 if disabled).

A Metanode or Component is a node that contains other nodes.

Creating a Metanode or Component

Select all relevant nodes, right-click and select ***Create Metanode...*** for a Metanode or ***Create Component...*** for a Component. Right-clicking a Metanode or Component opens the context menu with a number of options such as open, expand, setup, or reconfigure, and save as template.

Metanodes just collect nodes inside and are an efficient way to clean up your workflow.

Components encapsulate & abstract functionality, can have their own dialog and can have their own sophisticated, interactive views. They can be reused in your own workflows but also shared with others: via KNIME Server or the KNIME Hub. They can also represent web pages in a Data App deployed to others via KNIME Server. Flow Variables cannot enter or exit a Component, unless explicity configured in the component's input and output nodes.

A loop is a sequence of operations that is repeated until a condition is met. It has a start, an end, & a loop body of operations. A loop is implemented via a Loop Start node, a Loop End node, & a number of nodes in between for the body of operations. Different Loop Start nodes provide alternative ways to iterate on the input data. Different Loop End nodes provide alternative ways to collect results. The end condition can be defined in either the Loop Start node or the Loop End node, depending on the kind of loop. Some nodes to start & end a loop work on data & others on Flow Variables. These nodes can be paired up freely - loops can start with data & end up with Flow Variables & vice versa.

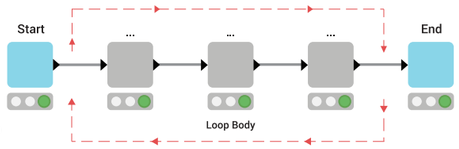


Table Row to Variable Loop Start: Starts a loop iterating over input data rows one at a time i.e. each iteration is dedicated to just one row. At each iteration, the row values are converted into Flow Variables & named after the column headers.

Group Loop Start: Starts a loop iterating over groups of input data rows i.e. each iteration works on a different group of the input data. Groups are extracted from values in selected columns, as in the configuration window of the GroupBy node.

Column List Loop Start: Starts a loop iterating over a selected list of columns. At each iteration, the current column & the remaining columns are passed into the loop body.

Generic Loop Start: Starts a loop. It should be paired with a loop end node defining the end condition.

Parallel Chunk Start: Starts a parallel loop iterating over input data rows in chunks of equal size which get processed in parallel in the loop body. The number of parallel iterations, (chunks), is defined either automatically based on the size of the input data, or manually.

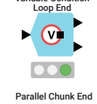
Recursive Loop Start: Starts a recursive loop iterating on the updated input table. The input data table for each iteration is the output data table from the previous iteration. The first iteration works on the data table provided at the input port of the node. This is the only loop where the updated data table feeds the next iteration.

Feature Selection Loop Start (1:1): Starts a feature selection loop iterating on a set of input columns (features) to extract the subset that optimizes a given Flow Variable. Used for feature selection against a given model & against a given performance metric. Together with the Feature Selection Loop End node, it's used in the Backward/Forward Feature Selection metanode.

Loop End: Ends a loop by concatenating the resulting rows from each iteration.

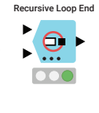
Loop End (Column Append): Ends a loop by joining together the resulting columns from each iteration on the column containing the row IDs.

Variable Loop End: Loop End nodes do not only work on data. At the end of a loop you might want to pass the results as a Flow Variable - like in the Variable Loop End node.



Variable Condition Loop End: Ends a loop when a condition is met, i.e. a specific value in a Flow Variable.

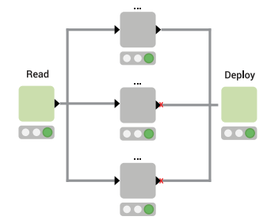
Parallel Chunk End: Ends a parallel loop by concatenating the resulting data rows from each chunk.



Recursive Loop End: Ends a recursive loop. The top input port collects the resulting data rows. The lower input port collects the updated data table to be passed back to the Recursive Loop Start node. Also defines the loop end condition, such as max. number of iterations, min. number of rows, & a specific value for a Flow Variable.

Feature Selection Loop End: Ends a feature selection loop. It sets the variable with the metric to be maximized/minimized for best feature selection. The top output port of the node produces the evaluation metrics for all column subsets. The bottom output port produces a summary of the different feature sets with the associated scores.

A switch construct allows you to conditionally execute different sequences of operations via nodes located on different workflow branches. All start with a Switch Start node and optionally end with a Switch End node. In between, a number of parallel branches implement various operation sequences.

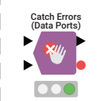


A switch construct works on data, flow variables, database queries, or any other port type defined via the dynamic ports of the start and end nodes. You can pair up different port types.

CASE Switch Start: Selectively activates only one of its three output ports, enabling three alternative paths for the input. The active output port can be configured manually or automatically via the value of a Flow Variable. The IF Switch node performs the same task but with only two alternative output ports (both can be active at the same time).

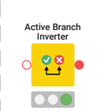
CASE Switch End: Collects the results from the active one among the branches connected to its input ports. The End IF node works similarly & is paired with the IF Switch node.

Try (Data Ports): Starts a try-catch construct to enable an alternative path for the data flow in case of failure in the main branch. One branch is defined as the main branch while the other is set as the secondary branch. If execution fails in the main branch, the secondary branch is activated. It must be closed by a Catch node.



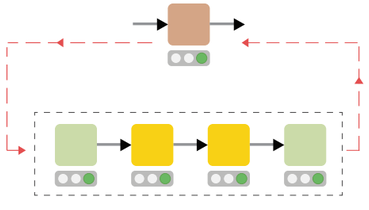
Catch Errors (Data Ports): Closes a try-catch construct started with a Try node & collects the results from the active branch.

Empty Table Switch: Provides an alternative path for the data flow in case the main branch has no data rows. It activates the top output port & deactivates the bottom output port if the input table has at least one data row. It deactivates the top output port & activates the bottom output port if the input table is empty.



Active Branch Inverter: Changes the activity status of the branch. If the input port is active, the output port becomes inactive & vice versa. It's often used to force a branch to produce an output even if it's inactive & vice versa (to deactivate a branch even if it's active).

ORCHESTRATION



Call Workflow Service: Triggers the execution of a workflow stored in the LOCAL workspace or on KNIME Server. Data exchange with the triggered workflow can happen via any port type via the Workflow Service Input/Output nodes.

Workflow Service Input: Receives data via any port type from the caller workflow via the Call Workflow Service node. Enables an efficient data exchange between workflows only, excluding third-party software. The corresponding "Workflow Service Output" node returns the results to the caller workflow.

Call Workflow (Table Based): Triggers the execution of a workflow stored in the LOCAL workspace or on KNIME Server. Data exchange with the triggered workflow can happen via data tables, Flow Variables, or credentials via the respective Container Input/Output nodes.

Container Input (Table): Receives a data table from the caller workflow. If no input is provided, the template default data table is used. Similar nodes are available to exchange Flow Variables & credentials. The corresponding "Container Output (Table)" node returns the results as a data table.

Call Remote Workflow (Row Based): Triggers the execution of a remote external workflow via REST. Remote means stored on a KNIME Server accessed using the server URL & credentials. Data exchange with the triggered workflow happens via JSON format.

Container Input (JSON): Receives a JSON data structure from the caller workflow. If no input is provided, the template default JSON structure is used. The corresponding "Container Output (JSON)" node returns the results as a JSON structure.

GET Request: Calls a REST service in GET mode. The node can send one single service request set in the configuration window, or multiple service requests stored in a column of the input table. Responses are saved in the output data table. Options to set authentication, request header, & response header are available.

KNIME Server Connector: Connects to a KNIME Server using the server URL & credentials. After the connection has been created, new directories on the server can be created & remote files can be accessed, created, & deleted.

Timer Info: Reports the number of executions & execution times for each node in a workflow. Both single node & total workflow execution time are reported. Execution times for nodes inside metanodes can also be reported.

Send Email: Sends HTML or text formatted emails using an external SMTP to a recipient - including the message & possible attachments.

Save Workflow: Saves the (also partially - up to here) executed workflow.

Create Folder: Creates a new folder and outputs the folder location as a flow variable of type Path.

Create File/Folder Variables: Creates a list of Path type Flow Variables pointing to files/folders relative to a selected base location.

Node ports

Different types of data pass through different node ports. Only ports of the same type can be connected. Here are some examples of ports for frequently used data types.

Data Table

PMML Model

Flow Variable

Tree Ensemble Model

Database Connection

Spark Context

Database Query

Spark Data

Image

HDFS

Resources

E-Books: KNIME Advanced Luck covers these advanced features & more. Practicing Data Science is a collection of data science case studies from past projects. Both available at knime.com/knimepress

KNIME Blog: Engaging topics, challenges, industry news, & knowledge nuggets at knime.com/blog

E-Learning Course: Take our free online self-paced coursesto learn about the different steps in a data science project (with exercises & solutions to test your knowledge) at www.knime.com/knime-self-paced-courses

KNIME Hub: Browse and share workflows, nodes, and components. Add ratings, or comments to other workflows at hub.knime.com

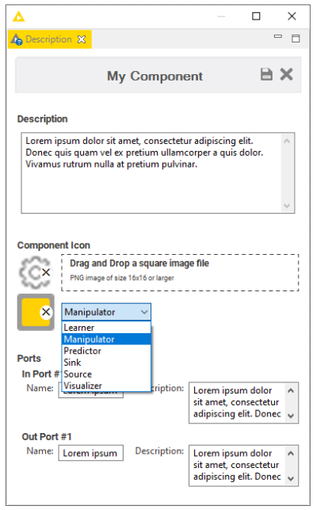
KNIME Forum: Join our global community & engage in conversations at forum.knime.com

KNIME Server: For team-based collaboration, automation, management, & deployment check out KNIME Server at www.knime.com/knime-server

Cheat Sheet: Components with KNIME Analytics Platform

Open for Innovation
KNIME

COMPONENTS DESCRIPTION



Documents the purpose and usage of the component and defines the appearance of the component. To access and edit the component description, open the Description panel from inside the component and select the empty canvas.

Description: Provides the text describing the use case, requirements, licensing and copyrights, disclaimers, and any other extra information you would like to add.

Icon: Drag and drop a PNG 16x16 px from your local file system. The icon will appear on the component, giving it a unique look.

Color: Usually the color of nodes and components is associated with a precise category. Pick the category that best fits your use case: The selected color appears behind the logo.

In/Out Ports: Describe the ports requirements here. For example: What kind of input column types are supported? Are there new rows/columns in the output? Does the input/output connect only to a precise other node or component?

Options: The component description also lists all the settings available inside the component dialog. To add text describing the settings usage open each configuration node dialog and enter it there node by node.

CONFIGURATION DESCRIPTION

Creates a list of options of type String for a menu or radio buttons. Define options in the configuration dialog together with the selected default value. This node produces the value of the selected option, which can be used inside the component to configure other node settings.

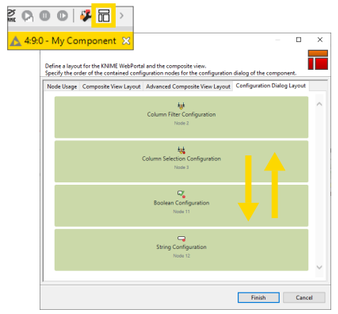
Creates a boolean selection for an enabled/disabled flag (1/0) in the form of a checkbox. This node produces the value of the selected option in a flow variable at its output port. Usually adopted to configure switch nodes and trigger different component modes.

Selects the columns of the component input table. Set the node to include or exclude all columns by default. Useful to remove columns that the component should not use in its execution.

Creates a string flow variable for configuring other nodes inside the component. Pick a default value so that the user understands how this component setting works. The string flow variable is useful to determine how to rename the component output columns or for text displayed in the component's composite view.

DIALOGUE LAYOUT PANEL

The component dialog stacks one setting for each configuration node in the component. From inside the component, select the Node Usage and Layout icon from the top toolbar. In the panel select the Configuration Dialog Layout tab. Drag and drop tiles to change the component settings based on sequential steps the user should follow, from top to bottom.



WIDGET NODES

Displays an SVG or PNG image in the composite view of the component. The static image can be displayed in different pixel sizes and it usually comes from a Table to Image node connected to this widget.

Creates a slider to filter data to only include rows with values in the selected column within the specified range. The slider can interact with most of the other views inside the component when connected to its output.

Creates a paragraph of either free, preformatted, or HTML text. This widget is useful to show styled text in the component composite view, guiding the user on how to interact with other widget and views.

Creates a list of selectable columns from the input data table in the form of a menu or radio buttons. The node produces the name of the selected column in a flow variable at its output port.

Offers a button to be placed in the composite view to trigger workflow execution. When clicked, the nodes after this widget re-execute and if any of them are widgets or views they are also updated.

VIEW NODES

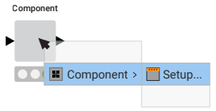
Displays one tile/card per row. Useful to browse information row after row esp. when displaying image or text data. It can automatically publish and subscribe to interactive events when it shares the same input with other views and/or interactive widgets in the component.

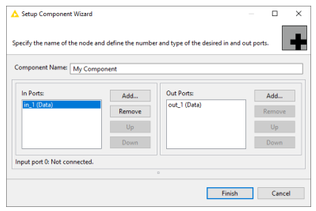
Displays one curve for every row and one parallel axis for each included column, both numerical and categorical. Useful to explore data points over several dimensions, looking for interesting patterns. It can automatically publish and subscribe to interactive events when it shares the same input with other views and/or interactive widgets in the component.

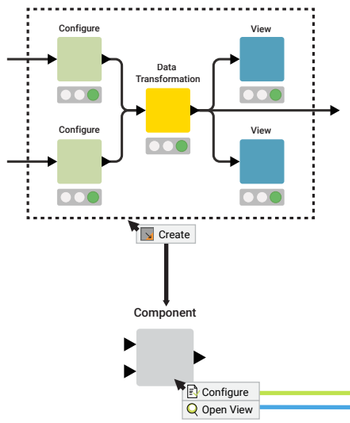
Displays a curve for each selected column on the y axis. The x axis in between the curves is based on another column or the RowID. This view comes from the KNIME Plotly Integration, a JavaScript based open source visualization library.

COMPONENT SETUP

You can still change component names and ports after component creation. The panel offers a text field to edit the title, buttons to change the order, remove or add. When adding a new port, a drop down menu appears to define the port type.

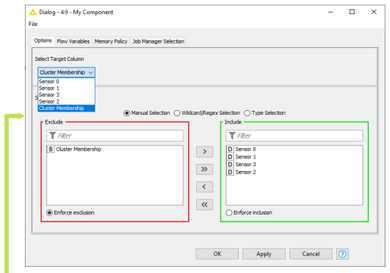




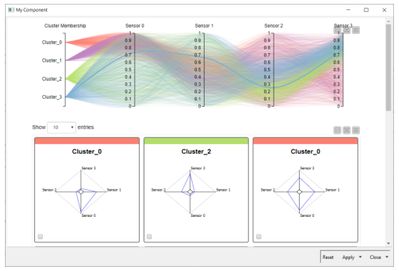


Create: To create a component multiple-select all the nodes you want to encapsulate in the component. Now right-click the selection and select "Create Component".

Configure: If you included a Configuration node in the component, open the configuration dialog. Change the component settings here before executing, just like any KNIME node.

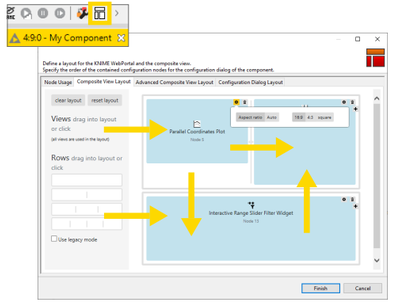


Open View: After executing a component that contains a Widget or View node, you can open its interactive view. In the opened composite view, interactivity across the single views and widgets nodes in the component is activated by default: visualizations sharing the same input table are usually connected via selection and filter events.



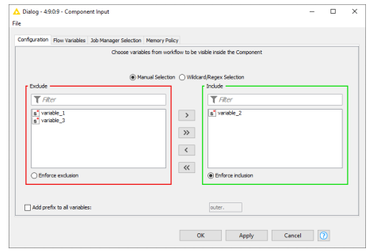
COMPOSITE VIEW LAYOUT

By default KNIME stacks views and widgets from top to bottom in the composite view. Access the Composite View Layout tab via the Node Usage Layout panel from inside the component. Define the layout of the composite view with tables: reset/clear the current layout (top left buttons); drag rows from the left; add columns using "+"; drag the views/widgets on the left into the empty structure. Optionally set the ratio or pixel sizes of individual views/widgets via the gear button.



FLOW VARIABLE FILTER

Filters the flow variables at the input and output of the component. Double-click the Input and Output ports inside the component to access a dialog. By default, flow variables pre-existing upstream of the component input cannot be used inside. Similarly, by default, flow variables generated within the component cannot be used downstream of the component output.



ERROR HANDLING

Fails on purpose when a certain condition is met. A custom error message appears on the node and on the outside of the component. Use it to detect whether the input or configurations of the component satisfy minimum requirements and provide the user with an intuitive message on what should be fixed after making the component fail on purpose.

VERIFIED COMPONENTS

Verified Components behave like actual nodes and are regularly released on the KNIME Hub. Browse them all at knime.com/verified-component. Each new component is assigned to one of the 9 categories, represented by a color and symbol.

Automation Example:
Trains classification models and outputs the best one in an automated fashion. The component adopts Integrated Deployment to capture the end-to-end process as an optimized workflow.

Guided Analytics Example:
Once a classifier model is trained and deployed, it can be monitored. The component generates a chart with performance over time and controls for re-training of the model.

Time Series Example:
Trains in Python a SARIMA (Seasonal ARIMA) model. This model is a powerful option when designing forecasts on data with seasonal or cyclic patterns.

Visualizations Example:
Visualizes a bar chart changing over time via a Generic JavaScript View node. Each visual bar represents a different category, racing in range with the others in a smooth animation.

Finance Analytics Example:
Computes the fraction of the year of the difference between two dates, similar to the YEARFRAC function in Microsoft Excel.

Text Processing Example:
Uses a Java based library to extract the main textual content from a web page to analyze online trends for Search Engine Optimization (SEO).

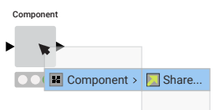
Data Manipulation Example:
A series of transformations have been implemented via the KNIME Python Integration. Use it to learn how to reliably package your Python scripts for other KNIME users.

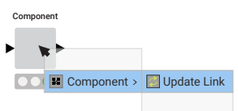
Model Interpretability Example:
Computes and visualizes global feature importance for an input model with four available model-agnostic XAI techniques.

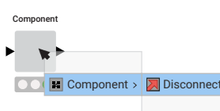
Life Sciences Example:
Selects a subset of molecules based on molecular properties via RDKit nodes. The Interactive View depicts the properties and structural formula of the selected molecules.

SHARED COMPONENTS

Components can be reused as your personal customized KNIME nodes or they can also be shared with others via KNIME Hub and KNIME Server.







To share a component choose the destination for the shared component in the window that opens. Now the component is shared and it can only be edited in its new location: A green arrow appears on top of the component to represent the link between the instance and the shared component in the KNIME Explorer. The link can be of relative type when the component is shared locally.

Manual update is useful to make sure your component is still up to date. A check for updates can be automated or prompted based on your KNIME Preferences. If an update is found but you opt out, the green arrow becomes red. If the link between the instance and the shared component is no longer valid the green arrow will become a red cross.

The owner of the shared component can edit its new location on the KNIME Explorer. After editing, KNIME can update the shared component instances in other workflows. If you are using someone else's shared component you can still edit it, but first you need to break the link: note that then you won't be able to get updates anymore.

SCRIPTED COMPONENTS

Component builders can optionally implement component functionalities through coding. KNIME supports a number of coding frameworks as well as ways to ship dependencies with the component.

Offers a code editor for JavaScript to implement a custom view. Optionally feed in data to visualize it based on your implementation. The node offers checkboxes for a few dependencies (d3.js, ..) as well as a CSS editor.

Offers a code editor for R to process a KNIME table. KNIME executes the R installation configured either in the node settings and/or in KNIME Preferences. More nodes with different ports are available from the KNIME Interactive R Statistics Integration

Offers a code editor for Python to process any number and type of inputs into outputs: three dots on the outside of the node means you can add ports. KNIME executes the Python installation configured either in the node settings and/or in KNIME Preferences.

Automatically installs the Conda environment necessary for your component to execute the downstream R/Python nodes. The environment usually includes the R/Python installation plus precise versions of the libraries. Useful to share scripted components with the necessary dependencies. Requires installation of Anaconda or Miniconda and minimal configuration in KNIME Preferences.

Resources

E-Books: KNIME Advanced Luck covers advanced features & more. Practicing Data Science is a collection of data science case studies from past projects. Both available at knime.com/knimepress
KNIME Blog: Engaging topics, challenges, industry news, & knowledge nuggets at knime.com/blog
E-Learning Courses: Take our free online self-paced courses to learn about the different steps in a data science project (with exercises & solutions to test your knowledge) at www.knime.com/knime-self-paced-courses
KNIME Hub: Browse and share workflows, nodes, and components. Add ratings, or comments to other workflows at hub.knime.com
KNIME Forum: Join our global community & engage in conversations at forum.knime.com
KNIME Server: For team-based collaboration, automation, management, & deployment check out KNIME Server at www.knime.com/knime-server

Cheat Sheet: Data Wrangling with KNIME Analytics Platform

Open for Innovation
KNIME

ACCESS DATA

CSV Reader — Reads a CSV file from either your local file system or another connected file system. Click the three dots in the lower left corner to add a dynamic connection input port to connect to an external file system, like Amazon S3, Azure Blob Storage, etc.

Excel Reader — Reads sheet(s) from one or more Excel files. One sheet from each Excel file. A loop can be used to read multiple sheets from one Excel file.

Table Reader — Reads data from a .table file. The .table files are organized using a KNIME proprietary format, including the full file structure, and are optimized for space and speed - providing maximum performance with minimum configuration.

SAP Reader (Theobald Software) — Loads data from various SAP systems (e.g. SAP S/4HANA, SAP BW, SAP R/3).

Amazon S3 Connector — Connects to Amazon S3 and points to a working directory (with a UNIX-like syntax, e.g., /mybucket/myfolder/myfile). Allows downstream reader nodes to access data from Amazon S3 as a file system.

Common settings of Reader and Writer nodes

File path: All Reader and Writer nodes require a file path. The file path can be expressed as an absolute path in the local file system, a relative path to a key location in the current KNIME installation, or a path defined in an external file system if such a connection is used.

Multiple files: Reader nodes can read and concatenate multiple files, according to a selected file extension or file name pattern.

Transformation tab: Reader nodes include a Transformation tab for renaming, filtering, re-ordering, and type changing of the columns.

DATABASES

DB Connector — Connects to any JDBC-compliant database. The JDBC driver must be added in the KNIME Preferences and then selected in the node configuration window.

H2 Connector — Connects to an H2 database. Similar dedicated connector nodes connect to other databases, such as MySQL or PostgreSQL.

DB Reader — Executes the input SQL query on the database and exports the results into a KNIME data table.

DB Table Selector — Creates a SQL query to access the database table selected in the configuration window. The table can be selected either via browsing the database metadata or via a custom SQL query.

COMBINE DATA

Concatenate — Concatenates the rows of all input tables by writing them below each other. This is especially useful for tables with shared column headers.

Joiner — Joins the columns of the two input tables based on one or multiple joining columns. Allows you to select between different joiner modes and to use multiple joining columns.

Full outer join · Inner join · Right outer join · Left outer join

Cell Replacer — Replaces the values in one column of the table at the top input port with values from a look up table provided at the bottom input port.

DB Joiner — Expands the input SQL query to include the join of two tables. It has a similar configuration window as the joiner node. No SQL coding required. There are more DB nodes, all expanding the input SQL query with additional SQL instructions. Besides the SQL Query node, no DB nodes require SQL coding.

FILTER DATA

Row Filter — Filters rows in or out of the input table according to a filtering rule. The filtering rule can match a value in a selected column or numbers in a numerical range.

Rule-based Row Filter — Filters rows in or out according to a set of rules, defined in its configuration window. Rules are evaluated from top to bottom. Using TRUE as the antecedent applies the rule to all unmatched rows.

Reference Row Filter — Filters rows in or out from the top input table according to matching values in the selected column of the lower input table.

Column Filter — Filters columns in or out from the input table according to a filtering rule. Columns to be retained can be manually picked or selected according to their type, or based on a regex expression matching their name.

DB Row Filter — Expands the input SQL query to include the row filter criteria defined in the configuration window. Grouping of multiple conditions with an AND or OR conjunctions is also supported. No SQL coding required.

DB Query — Modifies the input SQL query using custom SQL. The input SQL query is represented by the place holder #table#.

WRITE DATA

Excel Writer — Writes the input table(s) to sheet(s) in an Excel file (XLS or XLSX). Click the three dots in the lower left corner to add a dynamic sheet input port to write multiple data tables into multiple sheets.

CSV Writer — Writes the input data table to a CSV file. Click the three dots in the lower left corner to add a dynamic connection input port to write to an external file system, like Amazon S3, Azure Blob Storage, etc.

Send to Tableau Server — Uploads the input table to a Tableau server for reporting.

Send to Power BI — Uploads the input table to Microsoft Power BI for reporting.

DB Writer — Inserts the data rows from the top input port into a table in the database specified by the input connection port. If the database table does not exist it will be created.

DB Connection Table Writer — Writes the resulting rows from the input SQL query into a new table inside the database.

DATE&TIME

String to Date&Time — Parses the strings in the selected columns according to a date/time format and converts them into Date&Time cells. Four Date&Time forms are supported: only date, only time, date&time, and date&time plus time zone.

Date&Time-based Row Filter — Extracts rows where the time value in the selected column lies within a given time window. The time window is specified either by a start and /or an end date or by a start date and a duration.

Date&Time Difference — Calculates the difference between two date&time objects e.g., from two selected columns, from a selected column and a fixed value, from a selected column and the current execution time, or from one cell and the cell in the previous row for a selected column.

Extract Date&Time Fields — Extracts selected time and date fields from a selected column of type date&time and appends their values in new columns.

CLEAN DATA

Missing Value — Defines and applies a strategy to replace missing values in the input table - either globally on all columns, or individually for each single column.

Duplicate Row Filter — Detects duplicate rows and applies the selected operation, e.g. removes duplicate rows. Duplicates are rows that have the same value in all selected columns.

Numeric Outliers — Detects and treats numerical outliers for each of the selected columns individually using the interquartile range (IQR).

RESHAPE AND AGGREGATE DATA

GroupBy — Groups the rows of a table by the unique values in selected columns and calculates aggregation and statistical measures for the defined groups. Despite its simple name, it offers powerful functionality and has many unsuspected usages.

Pivoting — Extends the aggregation functionality of the GroupBy node by creating an output table with columns and rows for the unique values in the selected input columns. The unique values of the grouping columns become rows and the unique values of the pivoting columns become columns.

Category to Number — Maps the categorical values in the selected columns to integer values and exports the mapping rules to the model output port. The Category to Number (Apply) and Number to Category (Apply) nodes apply the mapping rule in both directions.

One to Many — Creates one new column for each value in the selected input column. These values become the column headers. Cells in the newly created columns are set to 0 if the value is not present, otherwise 1. This type of encoding is called one-hot vector.

Transpose — Converts the rows to columns and the columns to rows.

Table Manipulator — Performs several transformations at once, such as renaming, filtering, re-ordering and type changing, on the input columns. By adding dynamic ports it can replace a concatenate node.

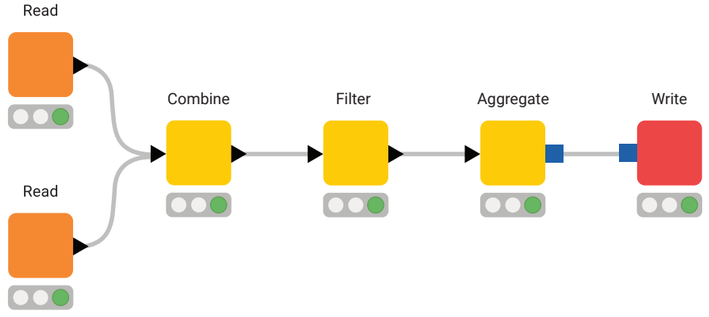
Cell Splitter — Splits values in the selected column into two or more substrings, as defined by a delimiter match. A delimiter is a defined character, such as a comma, space, or any other character or character sequence.

Ungroup — Ungroups a collection-type cell by creating one row for each value in the collection cell. Other columns from the input table are left unaltered.

Unpivoting — Stacks the cells of the selected value columns into one column. The cells of the selected retained columns are appended to the corresponding output rows.

Sorter — Sorts the table in ascending or descending order based on the values of one or more columns.

Value Counter — Counts the number of occurrences of all values in a selected column from the input table.



DATA TYPES & CONVERSIONS

- **String**: Sequence of characters, e.g. "This is a string"
- **Integer**: Whole real valued number, e.g. -100 or 345
- **Double**: Real valued number, e.g. -0.432 or 45.39
- **Date&Time**: A data format for date, time, date&time, or date&time plus time zone.
- **Boolean**: Two possible values only, e.g. TRUE and FALSE
- **Collection Cell**: Collection of multiple values of either the same or different types e.g., can be a list of values or a set of values. In a set each value occurs only once.
- **Document/Image**: KNIME Analytics Platform supports many more data types like text documents, images, fingerprints, etc.

Number to String — Converts the data type of the selected columns from a number format, e.g. integer or double, to string.

String to Number — Converts the data type of the selected columns from string to either double or integer.

CREATE COLUMNS

Math Formula — Implements a number of math operations across multiple input columns. The math operations can be applied to multiple columns with the Math Formula (Multi Column) node.

Rule Engine — Applies a set of rules to each row of the input table. Rules are applied from top to bottom. The first rule that matches is used.

Counter Generation — Creates a new column with a counter. The start value and step size are defined in the configuration window.

String Manipulation — Performs operations on string values in columns, such as combining two or more strings together, extracting one or more substrings, trimming blank spaces, and so on.

String Replacer — Replaces values in a selected string column if they match a defined pattern.

Column Expressions — Combines the functionality of the Math Formula, Rule Engine, and String Manipulation nodes. More than one expression can be defined to modify or add multiple columns at the same time.

DYNAMIC PORT

Concatenate

Dynamic ports: Additional input ports can be added by clicking the three dots in the bottom left corner of a node.

FORMAT EXCEL SHEETS

The Continental Nodes for KNIME extension allows you to automatically format an existing Excel sheet. The key is an additional data table of the same size as the original Excel sheet, where each cell contains one or more comma separated tag values e.g., header, border, etc. Based on these tags, the XLS Formatter nodes add new formatting instructions to the existing instructions, as available at the lower (optional) input port.

XLS Control Table Generator — Transforms the input table to an XLS Control Table, meaning it exchanges the column names to A, B, C, ... and the row IDs to 1, 2, 3, ... It is the kickoff node to collect formatting instructions for an Excel sheet and feeds all XLS formatter nodes.

XLS Background Colorizer — Adds background color and/or pattern fill formatting instructions to all cells with a specified tag in the XLS Control Table at the top input port.

XLS Border Formatter — Adds border formatting instructions for a given range specified by a tag in the XLS control table at the top input port.

XLS Cell Merger — Adds formatting instructions to merge all cells with a specified tag in the XLS control table at the top input port.

XLS Conditional Formatter — Adds formatting instructions to color cell backgrounds according to their numeric value for all cells specified by a tag in the XLS control table at the top

XLS Formatter (apply) — Applies all formatting instructions to an existing Excel sheet.

Resources

E-Books: KNIME Advanced Luck covers advanced features & more. Practicing Data Science is a collection of data science case studies from past projects. Both available at knime.com/knimepress

KNIME Blog: Engaging topics, challenges, industry news, & knowledge nuggets at knime.com/blog

E-Learning Courses: Take our free online self-paced courses to learn about the different steps in a data science project (with exercises & solutions to test your knowledge) at www.knime.com/knime-self-paced-courses

KNIME Hub: Browse and share workflows, nodes, and components. Add ratings, or comments to other workflows at hub.knime.com

KNIME Forum: Join our global community & engage in conversations at forum.knime.com

KNIME Server: For team-based collaboration, automation, management, & deployment check out KNIME Server at www.knime.com/knime-server

Cheat Sheet: Connectors with KNIME Analytics Platform

Open for Innovation
KNIME

ON PREMISE

ACCESS

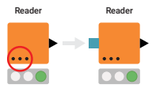
Environments

Creates a fully functional local big data environment including Apache Hive, Apache Spark, and HDFS. Allows opening Spark WebUI and sharing Spark contexts between KNIME workflows. Ideal for test runs.

Creates a Databricks Environment connected to an existing Databricks cluster.

Creates a new Spark context via Apache Livy. Requires access to a remote file system in order to exchange temporary files between KNIME and the Spark context running on the cluster.

Most Reader nodes can read both local and remote data. They connect to remote data sources via dynamic ports. These can be activated by clicking the three dots in the node lower left corner.



Most Reader nodes have their corresponding Writer nodes. Similar to Reader nodes, most Writer nodes support writing the data directly to remote locations via dynamic ports.

FILE SYSTEMS

Files

Reads data from a .table file. .table files are saved using a KNIME proprietary format, include the file structure, and are optimized for space and speed. Other nodes are available to read tabular formatted files, e.g., Parquet or ORC files.

Reads all text files, particularly character separated files, such as CSV files. Other similar reader nodes are dedicated to reading special file formats, like Excel or CSV files.

Parses textual content and metadata and extracts embedded files and attachments from more than 280 file formats. Also provides an authentication option for encrypted files.

Reads textual data straight out of document copies or photos using the Tesseract OCR library.

Reads either the whole JSON document or the selected part of the document, specified with a JSONPath query. The XML Reader node reads XML documents.

Reads Apache log files.

Reads PNG and SVG images, as well as ZIP files containing images, by browsing over the file system. Similar reader nodes read images from URLs, URIs, or Paths in the input table.

Reads audio files into a data cell. It is often used together with the Audio Viewer node to play audio files.

Reads and creates a network saved to a graph file with the Network Writer node.

Reads networks from visone and Cytoscape.

Reads the measurement data of one or more channels of an ASAM MDF file, either fully or in part.

Loads molecules from MDL Structure-Data Files (SDF).

Integrations

Executes a Python script in a local Python environment. Supports Python 2 and 3 and Jupyter notebooks import.

Reads Python pickle objects. Supports Python 2 and 3 and Jupyter notebooks import.

Reads diverse data sources from R into a KNIME table.

Reads Lucene table index.

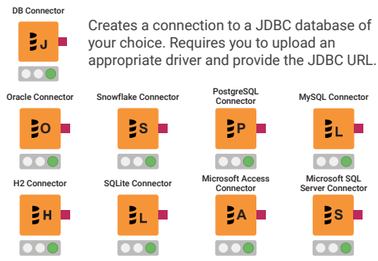
Reads H2O's Generated MOJO models.

Reads data from sas7bdat files.

DATABASES

Dedicated Connector nodes connect to a specific SQL, noSQL, or big data platform, and require a limited number of settings e.g., hostname and credentials.

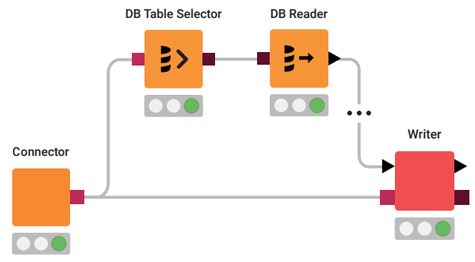
SQL



Creates a connection to a JDBC database of your choice. Requires you to upload an appropriate driver and provide the JDBC URL.

NoSQL

Big Data



WEB SERVICES

Services

Calls a REST service in the GET, POST, PUT, DELETE, or PATCH mode. Can send one single service request set in the configuration window, or multiple service requests stored in a column of the input table. Options to set authentication, request header, & response header are available.

Connects to a SPARQL endpoint. Can be then used with Semantic Web nodes.

Provides an in-memory Semantic Web endpoint. Supports default and named graphs and works with all offered Semantic Web nodes.

Connect to Twitter's API, retrieve tweets, users, or post new tweets. Require credentials for the Twitter's Developer account.

Reads triples stored in a file (.ttl, .rdf. .rj, .nt, .trig, .trix).

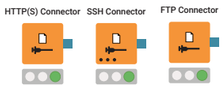
Interact with Salesforce's REST API performing authentication and SOQL queries execution.

Web

Retrieves web pages by issuing HTTP GET requests and parsing the requested HTML webpage from one or more URLs. The output can be returned in an XHTML or String format.

Connects to an RSS Feed URL, parses the RSS feeds, and retrieves the metadata. The results can be saved as String, Document, XML, or HTTP response code columns.

These nodes connect to web servers and specify a working directory with a UNIX-like syntax. The downstream nodes can then access the files on the server (FTP, SSH protocols) or read single files from a server (HTTP(S) protocol). The connection is closed when the Connector node is reset or the workflow is closed.



MODELS

Reads KNIME formatted models generated with any of the Learner nodes. The PMML Reader node reads PMML formatted models.

Reads a Keras deep learning network. A pre-trained network can be read from an HDF5 (.h5) file. A network specification without weights can be read from JSON or YAML files.

Reads a TensorFlow deep learning network of the SavedModel format from a directory or zip file.

Reads a TensorFlow 2 deep learning network from a file or folder. The model should be saved as an HDF5 (.h5) file, a SavedModel file, or zip file of a SavedModel.

Reads word vector models saved by the Word Vector Writer Node, models in .txt, .csv, or .bin.gz formats.

Downloads BERT models from TensorFlow Hub and HuggingFace to the disk. The cashed model can be then used with the BERT Classification Learner node.

Reads OpenNLP named entity tagging models.

ON THE CLOUD

Authentication

Authenticates against Google API services, via the "Authenticate" button's pop-up window. The Google Authentication (API Key) node performs the same authentication via a P12 key.

Authenticates against Microsoft Azure and Office 365 cloud services via a number of interactive authentication options.

Authenticates against Amazon services.

Cloud Storage Systems

Dedicated Connector nodes connect to remote file systems, specify the working directory with a UNIX-like syntax, and allow downstream nodes to access the remote file system just as a local one, e.g., to read or write files and folders, browse, list files, copy, move, etc. The connection is closed when the Connector node is reset or the workflow is closed.

Distributed File Systems

Dedicated Connector nodes connect to a specific distributed file system (HDFS, WebHDFS, HTTPFS, Databricks, ...), and require a limited number of settings e.g., hostname and credentials.

CLOUD

SERVERS

Connects to a KNIME Server using the server URL & credentials. Allows downstream nodes to access the server as a file system.

Accesses and loads data from various SAP systems (e.g., SAP S/4HANA, SAP BW, SAP R/3) via the Theobald Xtract Universal Server.

Connects to a remote SMB server. Allows downstream nodes to access the server as a file system.

Cloud Services

Connects to Google Sheets. Depending on the authentication method, the sheet should be either opened with a Google account or shared with a service account.

Connects to Google Analytics API.

Interacts with AWS AI/ML-Services like AWS Comprehend, AWS Translate, and AWS Personalize. The authentication via Amazon Authentication is required.

MESSAGING SYSTEMS

Connects to a Kafka cluster.

Consumes Kafka cluster messages on real time data feeds from sensor devices and stores them in a table.

Resources

E-Books: KNIME Advanced Luck covers advanced features & more. Practicing Data Science is a collection of data science case studies from past projects. Both available at knime.com/knime-press

KNIME Blog: Engaging topics, challenges, industry news, & knowledge nuggets at knime.com/blog

E-Learning Courses: Take our free online self-paced courses to learn about the different steps in a data science project (with exercises & solutions to test your knowledge) at www.knime.com /knime-self-paced-courses

KNIME Hub: Browse and share workflows, nodes, and components. Add ratings, or comments to other workflows at hub.knime.com

KNIME Forum: Join our global community & engage in conversations at forum.knime.com

KNIME Server: For team-based collaboration, automation, management, & deployment check out KNIME Server at www.knime.com/knime-server

Note: Missing your favorite source? This list is just an extract of the whole set of the connector nodes currently available within KNIME Analytics Platform. Besides, new connector nodes are being created as we speak.

Cheat Sheet: Machine Learning with KNIME Analytics Platform

Open for Innovation
KNIME

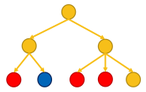
SUPERVISED LEARNING

Supervised Learning: A set of machine learning algorithms to predict the value of a target class or variable. They produce a mapping function (model) from the input features to the target class/variable. To estimate the model parameters during the training phase, labeled example data are needed in the training set. Generalization to unseen data is evaluated on the test set data via scoring metrics.

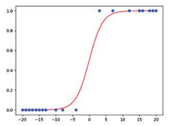
CLASSIFICATION

Classification: A type of supervised learning where the target is a class. The model learns to produce a class score and to assign each vector of input features to the class with the highest score. A cost can be introduced to penalize one of the classes during class assignment.

Decision Tree: Follows the C4.5 decision tree algorithm. These algorithms generate a tree-like structure, creating data subsets, aka tree nodes. At each node, the data are split based on one of the input features, generating two or more branches as output. Further splits are made in subsequent nodes until a node is generated where all or almost all of the data belong to the same class.



Logistic Regression: A statistical algorithm that models the relationship between the input features and the categorical output classes by maximizing a likelihood function. Originally developed for binary problems, it has been extended to problems with more than two classes (multinomial logistic regression).



Naive Bayes Learner

Naive Bayes: Based on Bayes' theorem and assuming statistical independence between input features (thus "naive"), this algorithm estimates the conditional probability of each output class given the vector of input features. The class with the highest conditional probability is assigned to the input data.

SVM Learner

Support Vector Machine (SVM): A supervised algorithm constructing a set of discriminative hyperplanes in high-dimensional space. In addition to linear classification, SVMs can perform non-linear classification by implicitly mapping their inputs into high-dimensional feature spaces, where the two classes are linearly separable.

K Nearest Neighbor

k-Nearest Neighbor (kNN): A non-parametric method that assigns the class of the k most similar points in the training data, based on a pre-defined distance measure. Class attribution can be weighted by the distance to the k-th point and/or by the class probability.

NUMERIC PREDICTION & CLASSIFICATION

RProp MLP Learner

Artificial Neural Networks (ANN, NN): Inspired by biological nervous systems, Artificial Neural Networks are based on architectures of interconnected units called artificial neurons. Artificial neurons' parameters and connections are trained via dedicated algorithms, the most popular being the Back-Propagation algorithm.

Keras Dense Layer

Deep Learning: Deep learning extends the family of ANNs with deeper architectures and additional paradigms, e.g. Recurrent Neural Networks (RNN). The training of such networks, has been enabled by recent advances in hardware performance as well as parallel execution.

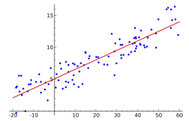
H2O Generalized Linear Model Learner

Generalized Linear Model (GLM): A statistics-based flexible generalization of ordinary linear regression, valid also for non-normal distributions of the target variable. GLM uses the linear combination of the input features to model an arbitrary function of the target variable (the link function) rather than the target variable itself.

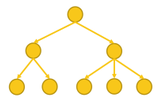
NUMERIC PREDICTION

Numeric Prediction: A type of supervised learning for numeric target variables. The model learns to associate one or more numbers with the vector of input features. Note that numeric prediction models can also be trained to predict class scores and therefore can be used for classification problems too.

Linear/Polynomial Regression: Linear Regression is a statistical algorithm to model a multivariate linear relationship between the numeric target variable and the input features. Polynomial Regression extends this concept to fitting a polynomial function of a pre-defined degree.



Regression Tree: Builds a decision tree to predict numeric values through a recursive, top-down, greedy approach known as recursive binary splitting. At each step, the algorithm splits the subsets represented by each node into two or more new branches using a greedy search for the best split. The average value of the points in a leaf produces the numerical prediction.



TIME SERIES ANALYSIS

Time Series Analysis: A set of numeric prediction methods to analyze/predict time series data. Time series are time ordered sequences of numeric values. In particular, time series forecasting aims at predicting future values based on previously observed values.

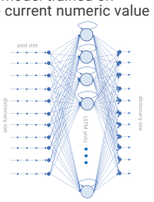
SARIMA Learner

Seasonal Auto-Regressive Integrated Moving Average (SARIMA): A linear Seasonal (S) Auto-Regressive (AR) model is constructed on a specified number p of past (seasonal) values; data are prepared by a degree of differencing d to correct non-stationarity; while a linear combination - named Moving Average (MA) - models the q past (seasonal) residual errors. All SARIMA model parameters are estimated concurrently by various algorithms, mostly following the Box–Jenkins approach.

Lag Column → Linear Regression Learner

ML-based TSA: A numeric prediction model trained on vectors of past values can predict the current numeric value of the time series.

Long Short Term Memory (LSTM) Units: LSTM units produce a hidden state by processing m x n tensors of input values, where m is the size of the input vector at any time and n the number of past vectors. The hidden state can then be transformed into the current vector of numeric values. LSTM units are suited for time series prediction as values from past vectors can be remembered or forgotten through a series of gates.



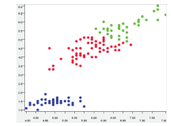
UNSUPERVISED LEARNING

Unsupervised Learning: A set of machine learning algorithms to discover patterns in the data. A labeled dataset is not required, since data are ultimately organized and/or transformed based on similarity or statistical measures.

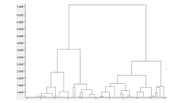
CLUSTERING

Clustering: A branch of unsupervised learning algorithms that groups data together based on similarity measures, without the help of labels, classes, or categories.

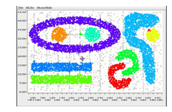
k-Means: The n data points in the dataset are clustered into k clusters based on the shortest distance from the cluster prototypes. The cluster prototype is taken as the average data point in the cluster.



Hierarchical Clustering: Builds a hierarchy of clusters by either collecting the most similar (agglomerative approach) or separating the most dissimilar (divisive a pproach) data points and clusters, according to a selected distance measure. The result is a dendrogram clustering the data together bottom-up (agglomerative) or separating the data in different clusters top-down (divisive).



DBSCAN: A density-based non-parametric clustering algorithm. Data points are classified as core, density-reachable, and outlier points. Core and density-reachable points in high density regions are clustered together, while points with no close neighbors in low-density regions are labeled as outliers.



SOTA Learner

Self-Organizing Tree Algorithm (SOTA): A special Self-Organizing Map (SOM) neural network. Its cell structure is grown using a binary tree topology.

Fuzzy c-Means

Fuzzy c-Means: One of the most widely used fuzzy clustering algorithms. It works similarly to the k-Means algorithm, but it allows for data points to belong to more than one cluster, with different degrees of membership.

RECOMMENDATION ENGINES

Recommendation Engines: A set of algorithms that use known information about user preferences to predict items of interest.

Association Rule Learner

Association Rules: The node reveals regularities in co-occurrences of multiple products in large-scale transaction data recorded at points-of-sale. Based on the a-priori algorithm, the most frequent itemsets in the dataset are used to generate recommendation rules.

Spark Collaborative Filtering Learner (MLlib)

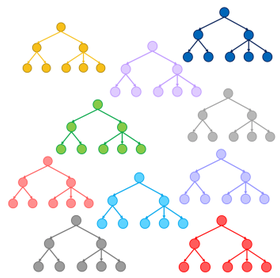
Collaborative Filtering: Based on the Alternating Least Squares (ALS) technique, it produces recommendations (filtering) about the interests of a user by comparing their current preferences with those of multiple users (collaborating).

ENSEMBLE LEARNING

Ensemble Learning: A combination of multiple models from supervised learning algorithms to obtain a more stable and accurate overall model. Most commonly used ensemble techniques are Bagging and Boosting.

BAGGING

Bagging: A method for training multiple classification/regression models on different randomly drawn subsets of the training data. The final prediction is based on the predictions provided by all the models, thus reducing the chance of overfitting.



Tree Ensemble of Decision/Regression Trees: Ensemble model of multiple decision/regression trees trained on different subsets of data. Data subsets with less or equal rows and less or equal columns are bootstrapped from the original training set. Final prediction is based on a hard vote (majority rule) or soft-vote (averaging all probabilities or numeric predictions) on all involved trees.

Random Forest of Decision/Regression Trees: Ensemble model of multiple decision/regression trees trained on different subsets of data. Data subsets with the same number of rows are bootstrapped from the original training set. At each node, the split is performed on a subset of sqrt(x) features from the original x input features. Final prediction is based on a hard vote (majority rule) or soft-vote (averaging all probabilities or numerical predictions) on all involved trees.

Prediction Fusion

Custom Ensemble Model: Combining different supervised models to form a custom ensemble model. The final prediction can be based on majority vote as well as on the average or other functions of the output results.

XGBoost Tree Ensemble Learner

XGBoost: An optimized distributed library for machine learning models in the gradient boosting framework, designed to be highly efficient, flexible, and portable. It features regularization parameters to penalize complex models, effective handling of sparse data for better performance, parallel computation, and more efficient memory usage.

BOOSTING

Boosting: A method for training a set of classification/regression models iteratively. At each step, a new model is trained on the prediction errors and added to the ensemble to improve the results from the previous model state, leading to higher accuracy after each iteration.

Gradient Boosted Regression Trees: Ensemble model combining multiple sequential simple regression trees into a stronger model. The algorithm builds the model stagewise. At each iteration, a simple regression tree is fitted to predict the residuals of the current model, following the gradient of the loss function. This leads to an increasingly accurate and complex overall model. The same regression trees can also be used for classification.

EVALUATION

Evaluation: Various scoring metrics for assessing model quality - in particular, a model's predictive ability or propensity to error.

Scorer

Confusion Matrix: A representation of a classification task's success through the count of matches and mismatches between the actual and predicted classes, aka true positives, false negatives, false positives, and true negatives. One class is arbitrarily selected as the positive class.

Accuracy Measures: Evaluation metrics for a classification model calculated from the values in the confusion matrix, such as sensitivity and specificity, precision and recall, or overall accuracy.

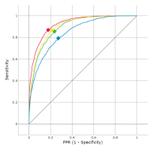
Cross Validation

Cross-Validation: A model validation technique for assessing how the results of a machine learning model will generalize to an independent dataset. A model is trained and validated N times on different pairs of training set and test set, both extracted from the original dataset. Some basic statistics on the resulting N error or accuracy measures gives insights on overfitting and generalization.

Numeric Scorer

Numeric Error Measures: Evaluation metrics for numeric prediction models quantifying the error size and direction. Common metrics include RMSE, MAE, or R^2. Most of these metrics depend on the range of the target variable.

ROC Curve: A graphical representation of the performance of a binary classification model with false positive rates on the x-axis and true positive rates on the y axis. Multiple points for the curve are obtained for different classification thresholds. The area under the curve is the metric value.



Resources

- **E-Books:** Learn even more with the KNIME books. From basic concepts in "KNIME Beginner's Luck", to advanced concepts in "KNIME Advanced Luck", through to examples of real-world case studies in "Practicing Data Science". Available for purchase at knime.com/knimepress
- **KNIME Blog:** Engaging topics, challenges, industry news, and knowledge nuggets at knime.com/blog
- **KNIME Hub:** Search, share, and collaborate on KNIME workflows, nodes, and components with the entire KNIME community at hub.knime.com
- **KNIME Forum:** Join our global community and engage in conversations at forum.knime.com
- **KNIME Server:** The enterprise software for team-based collaboration, automation, management, and deployment of data science workflows as analytical applications and services. Visit www.knime.com/knime-server for more information.