Module 8
8.2 How to solve a problem with KNIME

By **María Martínez Rojas**
Associate Professor Doctor in EAE al University of Málaga

By **José Manuel Soto Hidalgo**
Associate Professor at ATC, University de Granada

2. INTRODUCTION

Previous modules have shown examples of how to solve a data science problem from the perspective of the two main analysis branches: supervised and unsupervised learning (Modules 4, 5, and 6). In this capsule we will design a data flow that represents the general function of the data science life cycle (see Module 3, *Data science and machine learning*): (1) data reading, (2) data manipulation, (3) data exploration, (4) data analysis, (5) quality metrics, and (6) exporting and reporting. The main objective of this exercise is to show the capacity of the *KNIME* tool to solve data science problems.

The data flow consists of five distinct parts that mark the main stages of the life cycle. It starts with nodes that allow the data to be read and its preparation or manipulation. Nodes for visual exploration of the data are subsequently displayed to perform prior data analysis, while the final analysis is performed based on the generation of models and their validation. Finally, conclusions are drawn by generating reports or exporting the results.

Figure 1 shows the data flow we will develop in this capsule. Specifically, we will build a decision tree with an Iris dataset, read the data from a .csv file, manipulate and visualize the data, and validate the models generated with the decision tree.

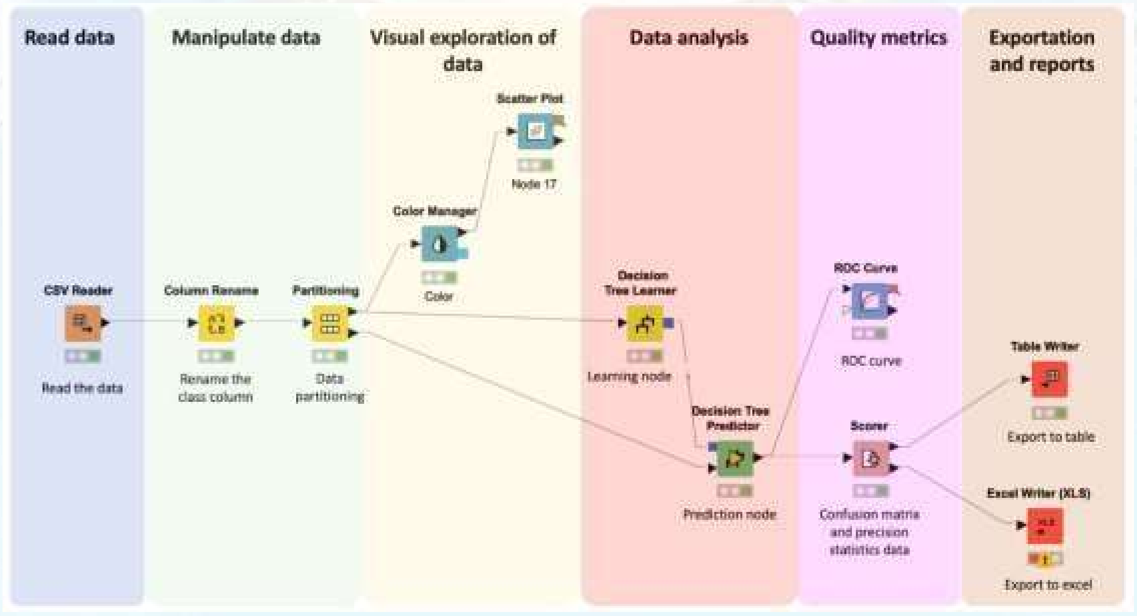


Figure 1. Data flow when using the Iris array.

2.1. DESIGNING THE DATA FLOW

To begin with we must create a new workspace. This can be done in two ways (as shown in figure 2.):

- Click on 'New' in the toolbar panel at the top of *KNIME*.
- Right-click on a folder in your local workspace in the *KNIME Explorer*.

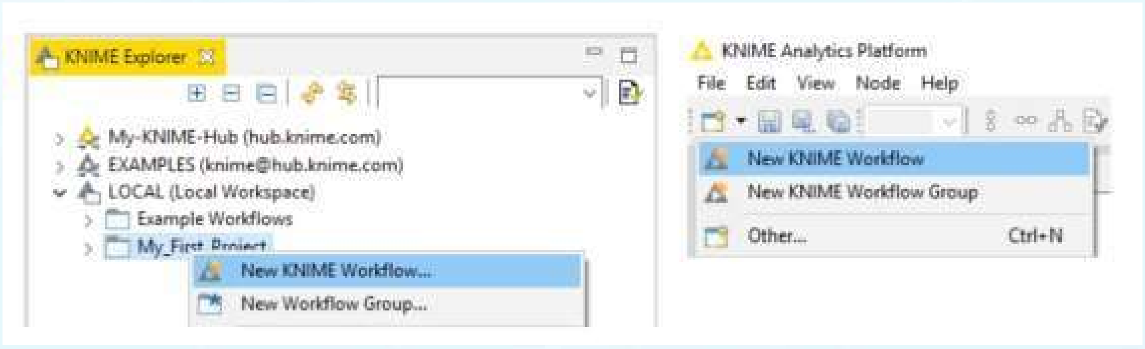


Figure 2. Creating a new workspace in KNIME.

As discussed above, the data flow will be illustrated with the Iris data set which consists of 50 samples of each of three species of Iris (*Iris setosa*, *I. virginica*, and *I. versicolor*). Four characteristics were measured for each sample: the length and width of the sepals and petals, in centimeters. The set is represented in .csv format and so the first node required is one that allows us to read this type of file. To do this, we must go to the node repository in the "IO → Read" section. We can also directly write the node name in the search engine available at the top.

As shown in figure 3, *KNIME* offers several nodes for reading files of different types which include but are not limited to ARFF, Table, PMML, and Excel formats. In our case, we selected the "CSV Reader" node that allows.csv files to be read. To use the node in the workflow you can either drag it directly from the repository and drop it into the workspace or double-click on the node in the repository and it will automatically appear in the workflow editor.

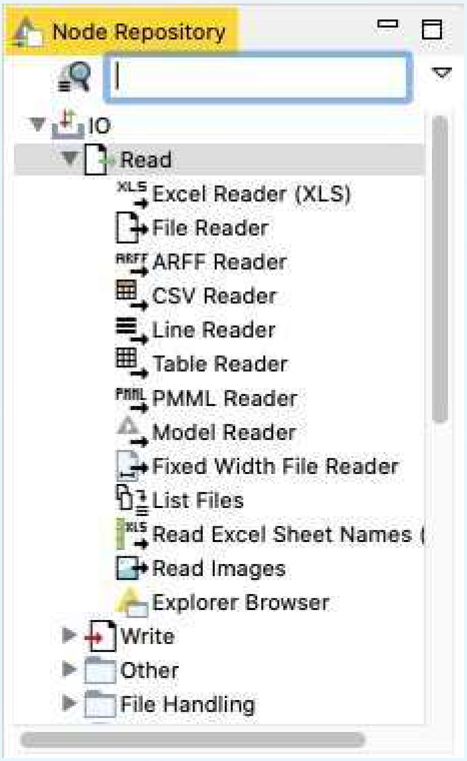


Figure 3. Nodes for reading different data formats.

Next, we must configure the node using one of three options: press the F6 key, double click on the node, or right click and select "Configure..." as shown in figure 4. Besides configuring the node, you can also perform other tasks from this menu, including running the node, viewing its outputs, editing the node, and showing the port data, etc.

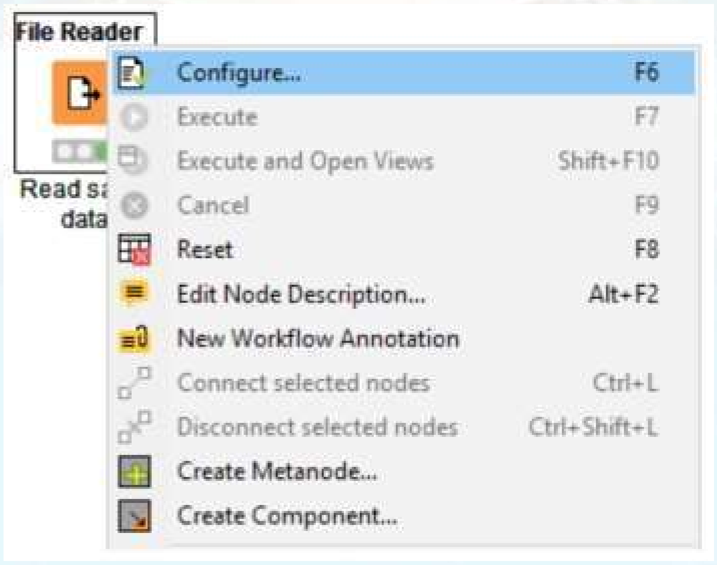
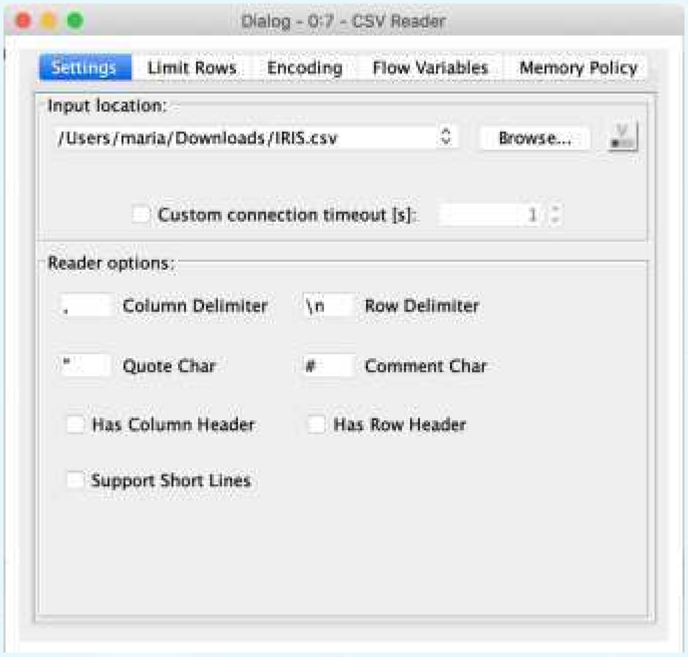

File Reader
Read sa
data
Configure... F6
Execute F7
Execute and Open Views Shift+F10
Cancel F9
Reset F8
Edit Node Description... Alt+F2
New Workflow Annotation
Connect selected nodes Ctrl+L
Disconnect selected nodes Ctrl+Shift+L
Create Metanode...
Create Component...

Figure 4. Configuring a node.

Once the configuration option is selected, a configuration dialog box appears (figure 5) in which we define the path to the file where the data are located by clicking on the "Browse" button. It is also possible to configure other available options and to preview the data in this window. For example, in our case, we deselected the "Has column header" and "Has row header" options because our dataset does not have these features. Once configured, click on the "ok" button and the node will appear in yellow (configured, but not executed). After this, you can run the node by pressing the F7 key, by right-clicking and selecting the "Run" option, or directly from the corresponding button in the menu bar at the top.


Dialog - 0:7 - CSV Reader

Settings | Limit Rows | Encoding | Flow Variables | Memory Policy

Input location: /Users/maria/Downloads/IRIS.csv Browse...

Custom connection timeout [s]: 1

Reader options:

, Column Delimiter   \n Row Delimiter

" Quote Char   # Comment Char

Has Column Header   Has Row Header

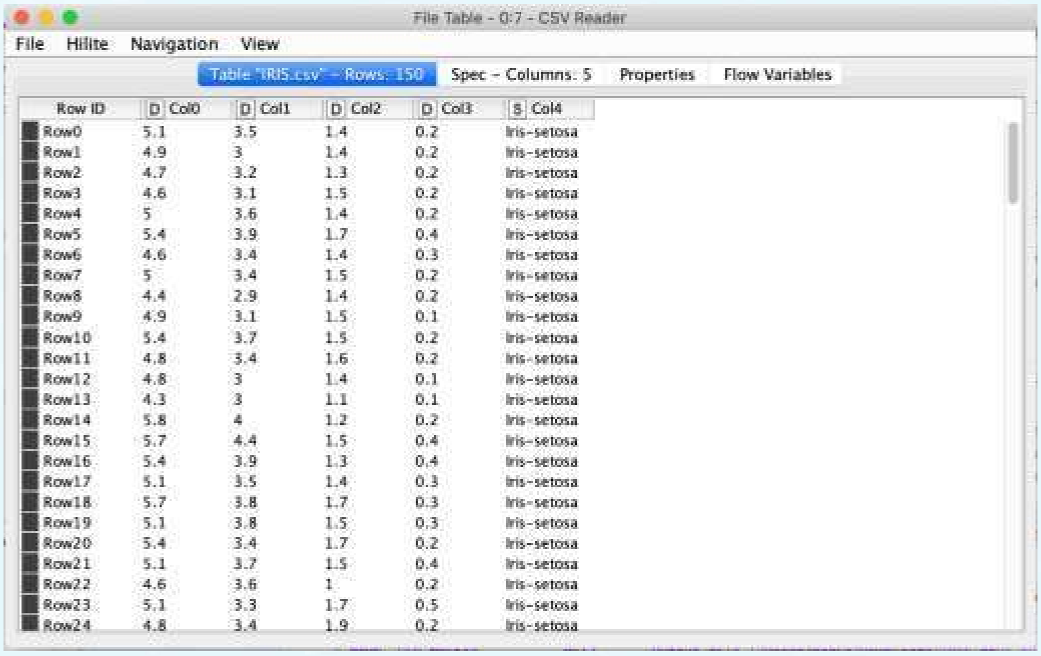
Support Short Lines

Figure 5. Input node configuration.

Once executed correctly (indicated when the node's traffic light turns green), we can visualize the data the node has read. The data will be in a table format and so to read them, right click and select the "File Table" option or select the button in the upper menu bar. Figure 6 shows the table containing the Iris data set data. As shown at the top, the set contains 150 rows (corresponding to 150 samples) and 5 columns (corresponding to the 4 variables and the class):

- RowID: row identification
- Col0: Sepal length
- Col1: Sepal width
- Col2: Petal length
- Col3: Petal width
- Col4: Species

As shown in figure 6, the name of each column defines the type of data contained in each column. For example, columns 0, 1, 2, and 3 contain decimal values (D=double) while column 4 contains a text string (S=string).



Row ID	D Col0	D Col1	D Col2	D Col3	S Col4
Row0	5.1	3.5	1.4	0.2	Iris-setosa
Row1	4.9	3	1.4	0.2	Iris-setosa
Row2	4.7	3.2	1.3	0.2	Iris-setosa
Row3	4.6	3.1	1.5	0.2	Iris-setosa
Row4	5	3.6	1.4	0.2	Iris-setosa
Row5	5.4	3.9	1.7	0.4	Iris-setosa
Row6	4.6	3.4	1.4	0.3	Iris-setosa
Row7	5	3.4	1.5	0.2	Iris-setosa
Row8	4.4	2.9	1.4	0.2	Iris-setosa
Row9	4.9	3.1	1.5	0.1	Iris-setosa
Row10	5.4	3.7	1.5	0.2	Iris-setosa
Row11	4.8	3.4	1.6	0.2	Iris-setosa
Row12	4.8	3	1.4	0.1	Iris-setosa
Row13	4.3	3	1.1	0.1	Iris-setosa
Row14	5.8	4	1.2	0.2	Iris-setosa
Row15	5.7	4.4	1.5	0.4	Iris-setosa
Row16	5.4	3.9	1.3	0.4	Iris-setosa
Row17	5.1	3.5	1.4	0.3	Iris-setosa
Row18	5.7	3.8	1.7	0.3	Iris-setosa
Row19	5.1	3.8	1.5	0.3	Iris-setosa
Row20	5.4	3.4	1.7	0.2	Iris-setosa
Row21	5.1	3.7	1.5	0.4	Iris-setosa
Row22	4.6	3.6	1	0.2	Iris-setosa
Row23	5.1	3.3	1.7	0.5	Iris-setosa
Row24	4.8	3.4	1.9	0.2	Iris-setosa

Figure 6. A data table read with the "CSV reader" node.

The name used to identify these columns is assigned by default by *KNIME* but can be changed with the "column rename" node found in the manipulation node repository. For example, to show the potential and versatility of *KNIME* in terms of data manipulation, we can add it to our data flow and join the output port of the read node with the input port of the node used to rename the column (figure 7). As shown, the node then appears with the yellow light that tells us that it must be configured before running it.

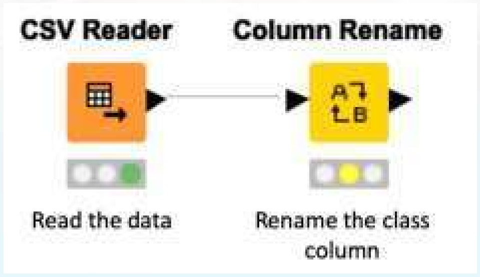
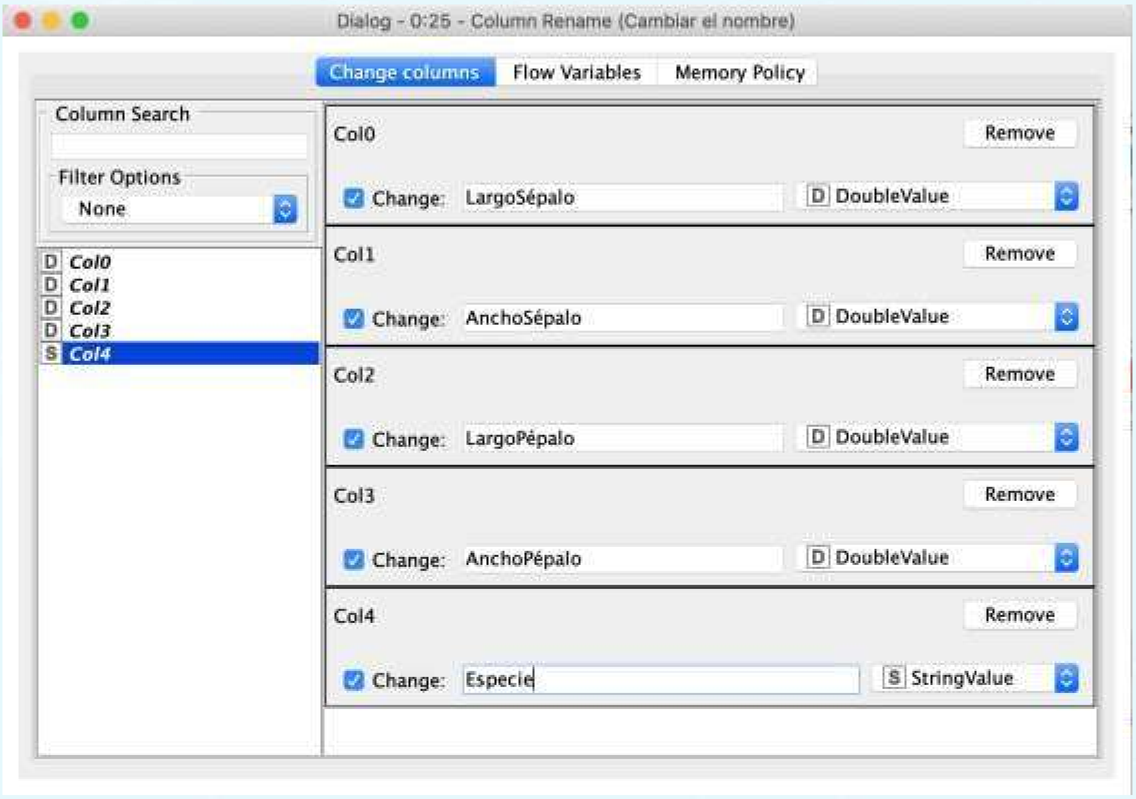


Figure 7: Linking the first two nodes of the example data stream.

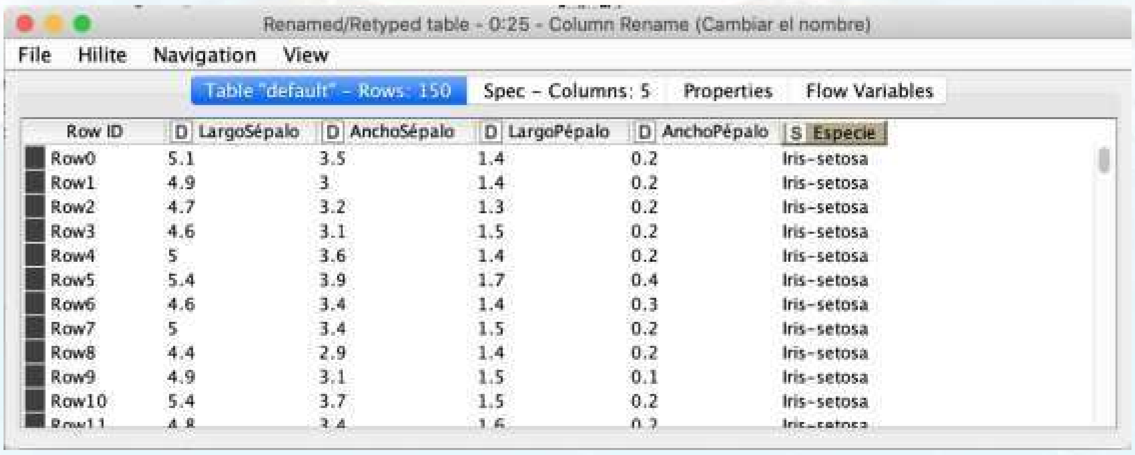
To do this, as mentioned above, double-click on the node, or select the option from the menu that opens after right-clicking. Figure 8 shows the configuration menu of this node; the columns we should rename can be selected on the left and the new name and data type contained in the column can be indicated on the right side.



Col0	☑ Change:	LargoSépalo	D DoubleValue
Col1	☑ Change:	AnchoSépalo	D DoubleValue
Col2	☑ Change:	LargoPépalo	D DoubleValue
Col3	☑ Change:	AnchoPépalo	D DoubleValue
Col4	☑ Change:	Especie	S StringValue

Figure 8. Node configuration used to rename the columns.

The result obtained by executing this node is illustrated in figure 9 which shows how the columns were renamed.


Renamed/Retyped table - 0:25 - Column Rename (Cambiar el nombre)
File Hilite Navigation View
Table "default" - Rows: 150
Spec - Columns: 5
Properties
Flow Variables

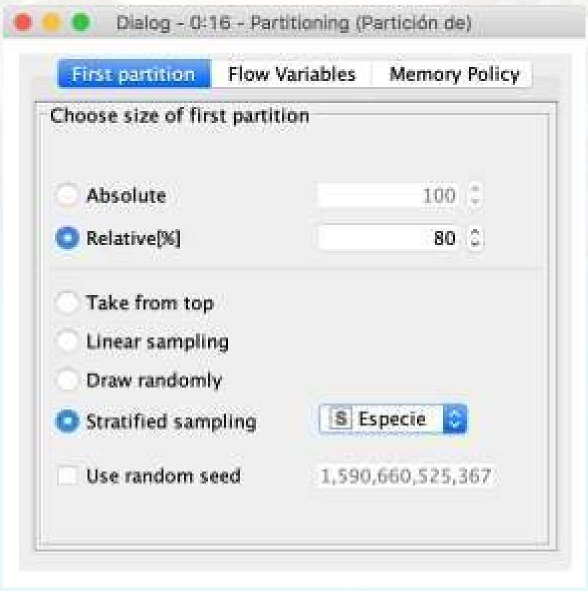
Row ID	D LargoSépalo	D AnchoSépalo	D LargoPépalo	D AnchoPépalo	S Especie
Row0	5.1	3.5	1.4	0.2	Iris-setosa
Row1	4.9	3	1.4	0.2	Iris-setosa
Row2	4.7	3.2	1.3	0.2	Iris-setosa
Row3	4.6	3.1	1.5	0.2	Iris-setosa
Row4	5	3.6	1.4	0.2	Iris-setosa
Row5	5.4	3.9	1.7	0.4	Iris-setosa
Row6	4.6	3.4	1.4	0.3	Iris-setosa
Row7	5	3.4	1.5	0.2	Iris-setosa
Row8	4.4	2.9	1.4	0.2	Iris-setosa
Row9	4.9	3.1	1.5	0.1	Iris-setosa
Row10	5.4	3.7	1.5	0.2	Iris-setosa
Row11	4.8	3.4	1.6	0.2	Iris-setosa

Figure 9. Output of the node which allows the columns to be renamed.

Finally, to finish the data preprocessing, we should divide the data set obtained into two different sets. This will create one set of data from which a model can be taught (training set) and another with which to validate the model obtained (test set). Remember that the objective of this current example is to teach a model that will allow us to obtain the class of a flower according to four of its characteristics.

To partition the data set we will use the methodology detailed in Module 3 (Capsule 2, *Supervised learning*). To achieve this, we use the “Partitioning” node that divides the input table into two partitions that can be queried through the two output ports. Figure 10 shows the configuration menu of this node. First, we must specify whether the partitioning must be performed ‘absolutely’ (as an absolute number of rows in the partition) or ‘relatively’ (as a percentage of the number of rows in the input table present in the first partition).

In this example, we perform a relative partitioning of 80%. For example, the “Take from top” option sorts the top rows in the first output table and the rest in the second table while “Stratified sampling” performs a sampling in which the distribution of values in the selected column is (approximately) maintained in the output tables. Remember that details of the different node options can be found in the node description area of the initial *KNIME* screen.



Dialog - 0:16 - Partitioning (Partición de)

First partition | Flow Variables | Memory Policy

Choose size of first partition

Absolute 100

[✓ Relative[%]] 80

Take from top

Linear sampling

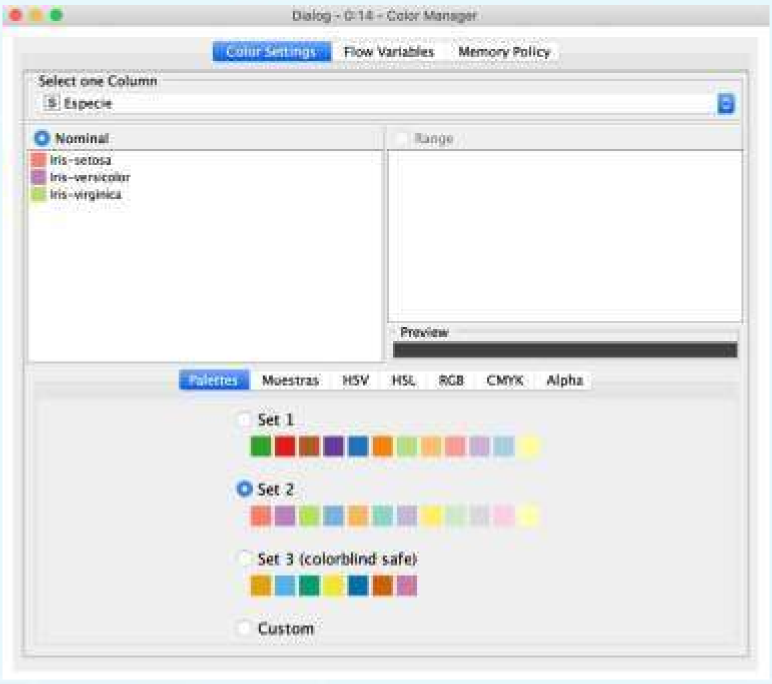
Draw randomly

[✓ Stratified sampling] S Especie

Use random seed 1,590,660,525,367

Figure 10. Node configuration for dataset partitioning.

Next, we will use a couple of nodes from the "Views" repository that allow us to explore the data visually, a very interesting aspect in data analysis. First, we are going to use the "Color manager" node to visually distinguish the species (classes) by selecting the option to configure the node by setting the column where the classes are located at the top (figure 11).



Dialog - 0:14 - Color Manager

Color Settings | Flow Variables | Memory Policy

Select one Column

S Especie

[✓ Nominal]

Iris-setosa

Iris-versicolor

Iris-virginica

Range

Preview

Palettes | Muestras | HSV | HSL | RGB | CMYK | Alpha

Set 1

[✓ Set 2]

Set 3 (colorblind safe)

Custom

Figure 11. The "Color manager" node configuration window.

Once configured, the node is executed and a color is assigned to each of the classes (figure 12). In this specific case, the *I. setosa* class was assigned red and the *I. versicolor* class was assigned purple.

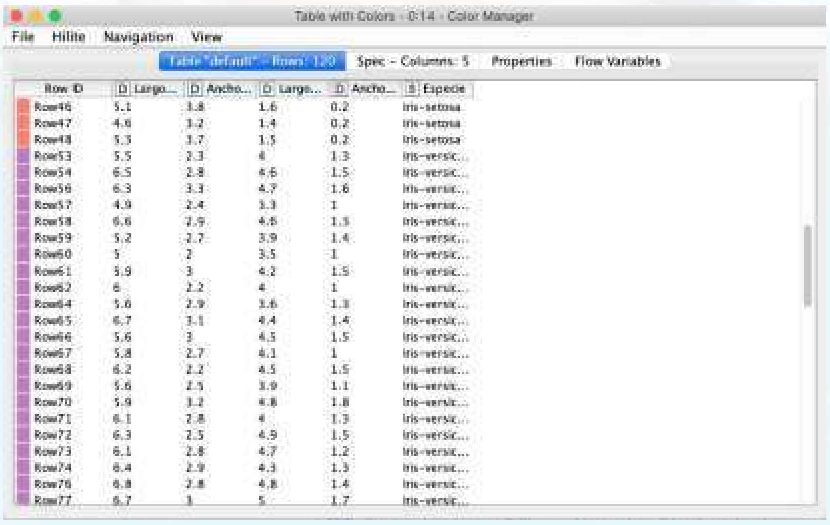


Table with Colors - 0:14 - Color Manager
File
Hilite
Navigation
View
Table "default" - Rows: 120
Spec - Columns: 5
Properties
Flow Variables

	Row ID	D Largo...	D Ancho...	D Largo...	D Ancho...	S Especie
🟥	Row46	5.1	3.8	1.6	0.2	Iris-setosa
🟥	Row47	4.6	3.2	1.4	0.2	Iris-setosa
🟥	Row48	5.3	3.7	1.5	0.2	Iris-setosa
🟪	Row53	5.5	2.3	4	1.3	Iris-versic...
🟪	Row54	6.5	2.8	4.6	1.5	Iris-versic...
🟪	Row56	6.3	3.3	4.7	1.6	Iris-versic...
🟪	Row57	4.9	2.4	3.3	1	Iris-versic...
🟪	Row58	6.6	2.9	4.6	1.3	Iris-versic...
🟪	Row59	5.2	2.7	3.9	1.4	Iris-versic...
🟪	Row60	5	2	3.5	1	Iris-versic...
🟪	Row61	5.9	3	4.2	1.5	Iris-versic...
🟪	Row62	6	2.2	4	1	Iris-versic...
🟪	Row64	5.6	2.9	3.6	1.3	Iris-versic...
🟪	Row65	6.7	3.1	4.4	1.4	Iris-versic...
🟪	Row66	5.6	3	4.5	1.5	Iris-versic...
🟪	Row67	5.8	2.7	4.1	1	Iris-versic...
🟪	Row68	6.2	2.2	4.5	1.5	Iris-versic...
🟪	Row69	5.6	2.5	3.9	1.1	Iris-versic...
🟪	Row70	5.9	3.2	4.8	1.8	Iris-versic...
🟪	Row71	6.1	2.8	4	1.3	Iris-versic...
🟪	Row72	6.3	2.5	4.9	1.5	Iris-versic...
🟪	Row73	6.1	2.8	4.7	1.2	Iris-versic...
🟪	Row74	6.4	2.9	4.3	1.3	Iris-versic...
🟪	Row76	6.8	2.8	4.8	1.4	Iris-versic...
🟪	Row77	6.7	3	5	1.7	Iris-versic...

Figure 12. Output of the "Color manager" node.

Another common node is "Scatter plot" which presents a scatter plot in which different columns can be chosen for x and y. The two variables for analysis are selected in the configuration menu and after executing the node, an image like the one shown in figure 13 is obtained. This figure shows the different cases according to the colors assigned to each class in the previous node.

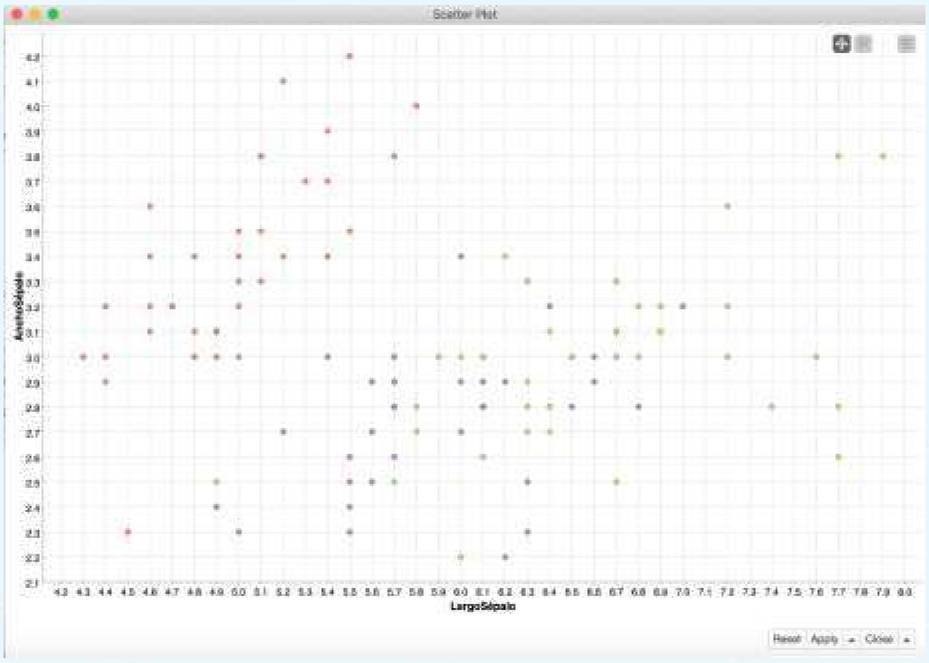
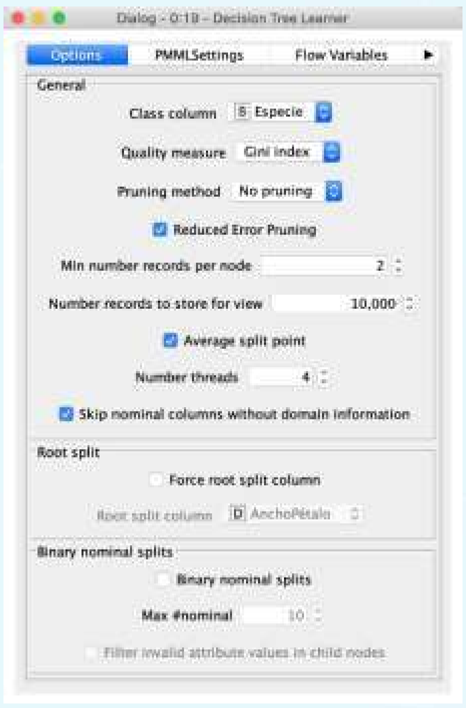


Figure 13. Output of the "Scatter plot" node.

We will start with the nodes that allow us to develop the model in the data analysis phase. In this example we will use the decision tree, which was explained in Module 5 (Capsule 2, *Standard classification methods*), as a representation technique for the classification of the species, although, as already mentioned, any other machine learning algorithm could be used. First, we must search the repository for the two required nodes: the learning and prediction nodes (“Decision Tree Learner” and “Decision Tree Predictor”, respectively) and drag them into the workflow.

Next we connect the ports to the previous node which, in this example, was the one we used to partition the initial dataset. Remember that the output of this node was two data sets that will be used to teach and test our model. Therefore, the upper output port of the ‘partitioning’ node must be connected to the input port of the “Decision Tree Learner” node and the lower output port to the “Decision Tree Predictor” node. These two nodes must be connected to each other through the blue port that allows transportation of the model generated by the “Decision Tree Learner” node.

Before executing these nodes we must configure them with the values and/or parameters considered appropriate for the data set. As shown in the learning node menu in figure 14, first we indicate which dataset column contains the species or class to be classified.



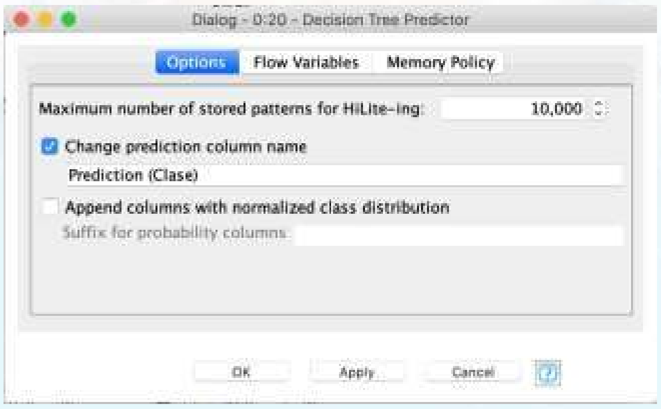
General
Class column: S Especie
Quality measure: Gini index
Pruning method: No pruning
[✓] Reduced Error Pruning
Min number records per node: 2
Number records to store for view: 10,000
[✓] Average split point
Number threads: 4
[✓] Skip nominal columns without domain information

Root split
[ ] Force root split column
Root split column: D AnchoPétalo

Binary nominal splits
[ ] Binary nominal splits
Max #nominal: 10
[ ] Filter invalid attribute values in child nodes

Figure 14. Configuration of the “Decision Tree Learner” node.

In addition, the “quality measure”, “pruning method”, and “minimum number of records per node”, etc. can also be configured. In the configuration menu of the node used to predict the species, we must indicate the name of the new column with the species classification result (figure 15). Finally, the two nodes can then be executed.



Maximum number of stored patterns for HiLite-ing: 10,000

[✓] Change prediction column name

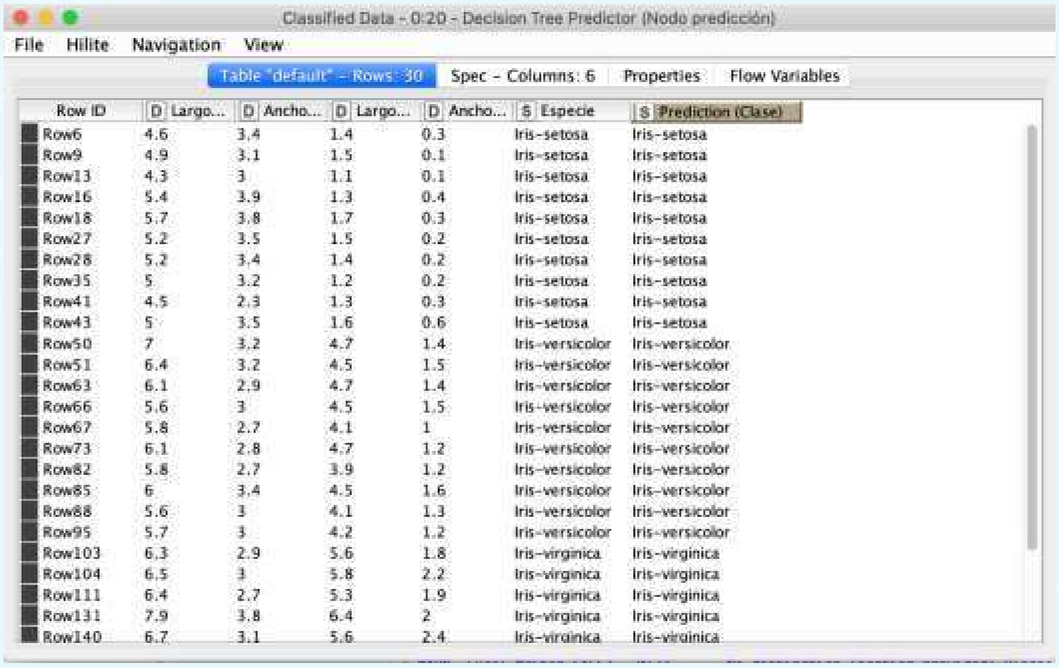
Prediction (Clase)

[ ] Append columns with normalized class distribution

Suffix for probability columns:

Figure 15. Configuration of the "Decision Tree Predictor" node.

The result of this node is shown in figure 16. To access this table, select the option from the drop-down menu by right-clicking or selecting it from the menu bar. An important aspect of the data science life cycle is evaluation of the quality of the taught models to measure how well they can predict new examples. To do this, we will include two nodes ("Scorer" and "ROC curve") that allow us to analyze the quality of the learned model in detail, i.e., how well the model is ranking.

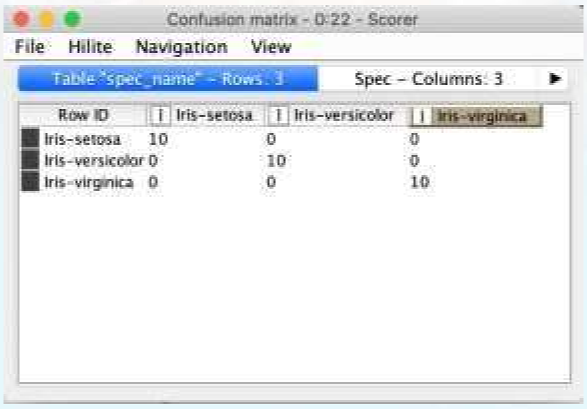

Classified Data - 0:20 - Decision Tree Predictor (Nodo predicción)
File
Hilite
Navigation
View
Table "default" - Rows: 30
Spec - Columns: 6
Properties
Flow Variables

Row ID	D Largo...	D Ancho...	D Largo...	D Ancho...	S Especie	S Prediction (Clase)
Row6	4.6	3.4	1.4	0.3	Iris-setosa	Iris-setosa
Row9	4.9	3.1	1.5	0.1	Iris-setosa	Iris-setosa
Row13	4.3	3	1.1	0.1	Iris-setosa	Iris-setosa
Row16	5.4	3.9	1.3	0.4	Iris-setosa	Iris-setosa
Row18	5.7	3.8	1.7	0.3	Iris-setosa	Iris-setosa
Row27	5.2	3.5	1.5	0.2	Iris-setosa	Iris-setosa
Row28	5.2	3.4	1.4	0.2	Iris-setosa	Iris-setosa
Row35	5	3.2	1.2	0.2	Iris-setosa	Iris-setosa
Row41	4.5	2.3	1.3	0.3	Iris-setosa	Iris-setosa
Row43	5	3.5	1.6	0.6	Iris-setosa	Iris-setosa
Row50	7	3.2	4.7	1.4	Iris-versicolor	Iris-versicolor
Row51	6.4	3.2	4.5	1.5	Iris-versicolor	Iris-versicolor
Row63	6.1	2.9	4.7	1.4	Iris-versicolor	Iris-versicolor
Row66	5.6	3	4.5	1.5	Iris-versicolor	Iris-versicolor
Row67	5.8	2.7	4.1	1	Iris-versicolor	Iris-versicolor
Row73	6.1	2.8	4.7	1.2	Iris-versicolor	Iris-versicolor
Row82	5.8	2.7	3.9	1.2	Iris-versicolor	Iris-versicolor
Row85	6	3.4	4.5	1.6	Iris-versicolor	Iris-versicolor
Row88	5.6	3	4.1	1.3	Iris-versicolor	Iris-versicolor
Row95	5.7	3	4.2	1.2	Iris-versicolor	Iris-versicolor
Row103	6.3	2.9	5.6	1.8	Iris-virginica	Iris-virginica
Row104	6.5	3	5.8	2.2	Iris-virginica	Iris-virginica
Row111	6.4	2.7	5.3	1.9	Iris-virginica	Iris-virginica
Row131	7.9	3.8	6.4	2	Iris-virginica	Iris-virginica
Row140	6.7	3.1	5.6	2.4	Iris-virginica	Iris-virginica

Figure 16. Output of the "Decision Tree Predictor" node with sorted data.

The "scorer" node provides the confusion matrix (Module 5, Capsule 1, *Classification: what, why, and how?*) which gives a count of the hits and misses for each of the species

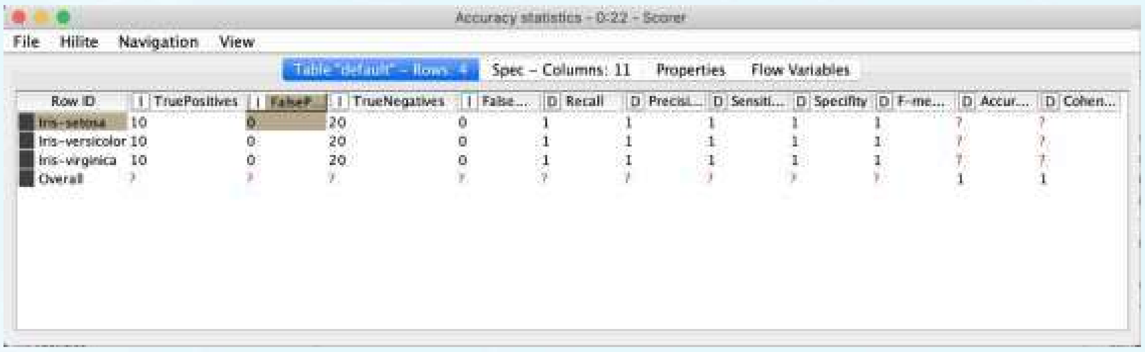
(*I. setosa*, *I. virginica*, and *I. versicolor*) allowing us to check the extent to which our model is confusing classes (figure 17).



Row ID	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	10	0	0
Iris-versicolor	0	10	0
Iris-virginica	0	0	10

Figure 17. Confusion matrix provided by the "Scorer" node.

Secondly, it provides a table with statistical data: true positives, true negatives, false positives, false negatives, recall, precision, *F*-metrics, etc. (figure 18).



Row ID	TruePositives	FalseP...	TrueNegatives	False...	Recall	Precisi...	Sensiti...	Specifity	F-me...	Accur...	Cohen...
Iris-setosa	10	0	20	0	1	1	1	1	1	?	?
Iris-versicolor	10	0	20	0	1	1	1	1	1	?	?
Iris-virginica	10	0	20	0	1	1	1	1	1	?	?
Overall	?	?	?	?	?	?	?	?	?	1	1

Figure 18. Table containing accuracy statistics data.

The "ROC curve" node allows us to obtain ROC curves for two-class classification problems, as detailed in module 5 (Capsule 1, *Classification: what, why, and how?*). The input table should contain one column with the actual class values (including all the class values as possible values) and a second column containing the probabilities that an element is classified as belonging to the selected class. In the example given here we have configured this node with the following data:

- Class column: (class preference)
- Positive class value: *Iris setosa*

The result is shown in figure 19.

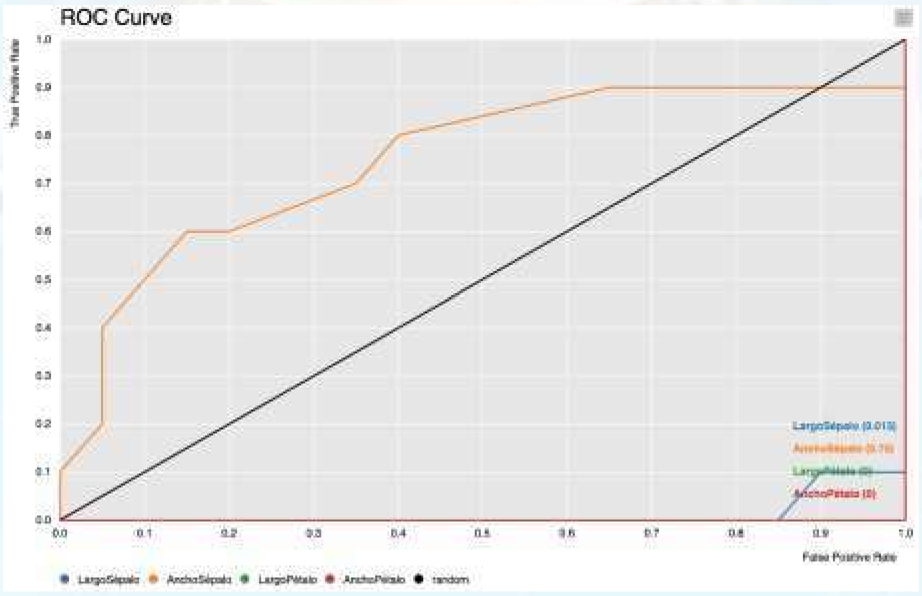
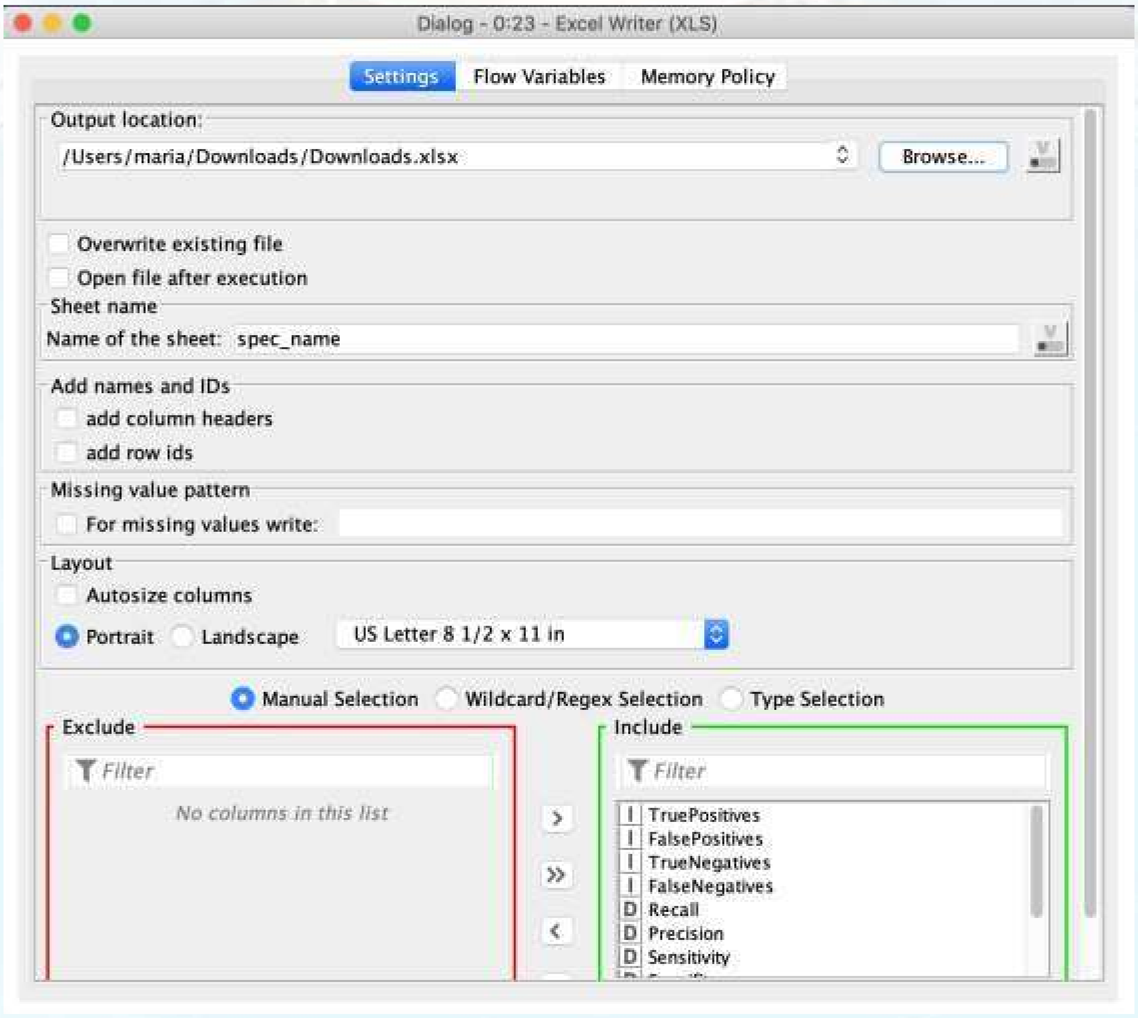


Figure 19. ROC curve.

Finally, the last nodes of the workflow allow us to write the results obtained. In this example, the “Table Writer” and “Excel Writer” nodes were included so that we can export the confusion table and statistical data, respectively. In the reading nodes we must indicate in the configuration menu where the file will be saved, as well as the data to be exported. For example, figure 20 shows the menu of the node that allows the data to be exported in Excel format. The location of the output file is indicated in the upper part, some formatting options are shown in the middle part, and the data to be exported can be defined in the lower part.


Dialog - 0:23 - Excel Writer (XLS)

Settings | Flow Variables | Memory Policy

Output location: /Users/maria/Downloads/Downloads.xlsx

Overwrite existing file [ ]

Open file after execution [ ]

Sheet name

Name of the sheet: spec_name

Add names and IDs

add column headers [ ]

add row ids [ ]

Missing value pattern

For missing values write: [ ]

Layout

Autosize columns [ ]

[✓ Portrait] Landscape — US Letter 8 1/2 x 11 in

[✓ Manual Selection] Wildcard/Regex Selection Type Selection

Exclude: No columns in this list

Include: TruePositives, FalsePositives, TrueNegatives, FalseNegatives, Recall, Precision, Sensitivity

Figure 20. The "Excel Writer" node configuration menu.

REFERENCES

- Bakos, G. (2013). KNIME essentials. Packt Publishing Ltd.

- Blokdyk, G. (2019). KNIME a Complete Guide - 2019 Edition. Emereo Pty Limited, 2019.

- McCormick, K. (2019). Introduction to Machine Learning with KNIME. linkedin.com.

- Silipo, R. (2016). Introduction to Data Analytics with KNIME: A Data Science Approach to Analytics. O'Reilly.

- Silipo, R., & Mazanetz, M. P. (2012). The KNIME cookbook. KNIME Press, Zürich, Switzerland.

- Strickland, J. (2016). Data Analytics Using Open-Source Tools. Lulu. com.