Module 8
8.3 Supervised learning: regression and classification methods in KNIME

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

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

1. INTRODUCTION

This capsule focuses on how to implement the different supervised learning algorithms introduced in modules 4 and 5 (*Supervised learning: regression techniques* and *classification techniques*, respectively) in *KNIME*. We will implement data flows representative of the data science lifecycle to solve linear (Module 4, Capsule 2, *Standard regression methods*) and regression tree (Module 4, Capsule 3, *Machine learning methods for regression*), classification problems with basic methods such as *k*-nearest neighbors (KNN) and decision trees (Module 5, Capsule 2, *Standard classification methods*), as well as advanced methods such as Random Forest (Module 5, Capsule 3, *Advanced classification methods*). Model validation processes will also be used to give statistical support to the results obtained.

1.1. REGRESSION

In this section we will create data flows to solve a regression problem with a linear regressor (Module 4, Capsule 2, *Standard regression methods*) and a regression tree (Module 4, Capsule 3, *Machine learning methods for regression*). We will use the HOMA dataset and illustrate different elements that can be used in *KNIME* to visualize which variables are most promising for regression and will draw regression plots and calculate different quality metrics for regression models.

1.1.1. How do we solve linear regression problems?

In this data flow, a “Linear Regression Learner” node and a “Regression Predictor” node will be used to perform linear regression (figure 1).

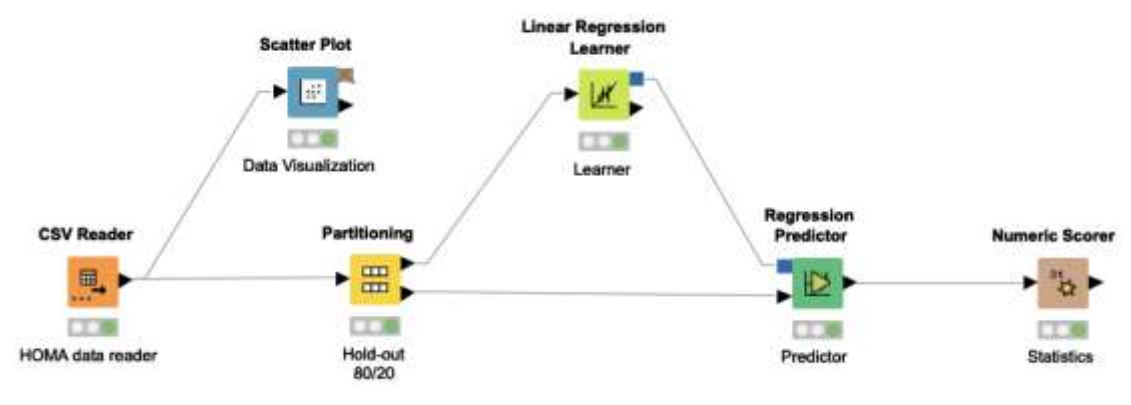


Figure 1. A linear regression data flow.

The HOMA dataset is initially read with a "CSV Reader" node. Once the data have been read, it is often useful to first visually analyze the variables and observe the correlations between them. The "Scatter Plot" node allows us to do this is by generating an interactive 2D graph to visualize the data associated with two given variables. For example, we can visualize different variables in the dataset with respect to HOMA. To achieve this, we click on the menu icon at the top right of the graph and select the variables we want to represent on the X and Y axis. Figures 2 and 3 graphically show the dispersion of the variables "Sex" and "SBP" with respect to HOMA, respectively.

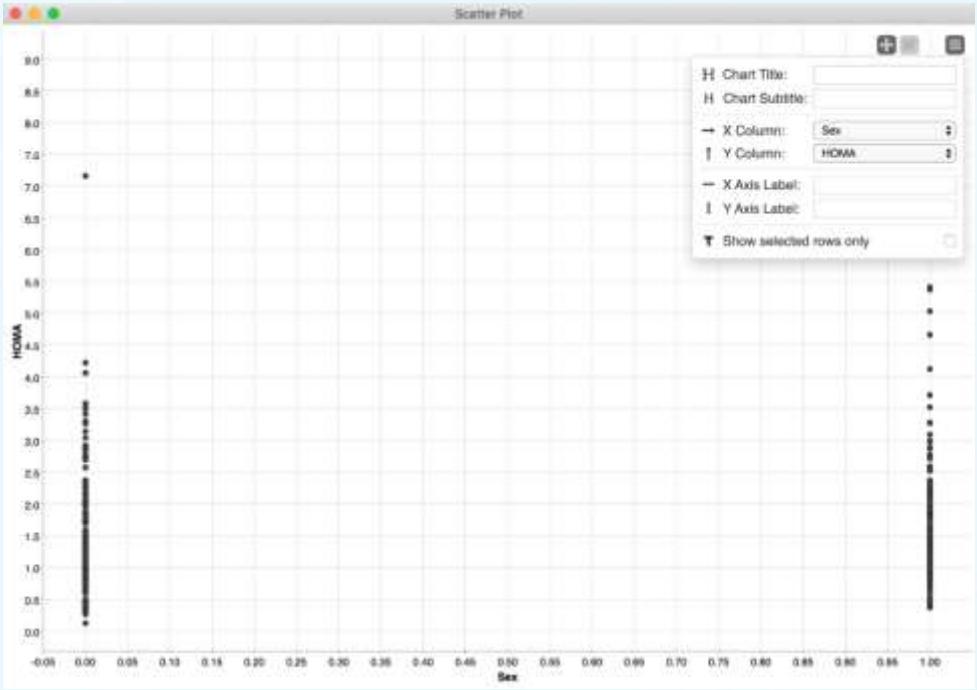


Figure 2. A linear regression data flow: visualization of the Sex variable with respect to HOMA.

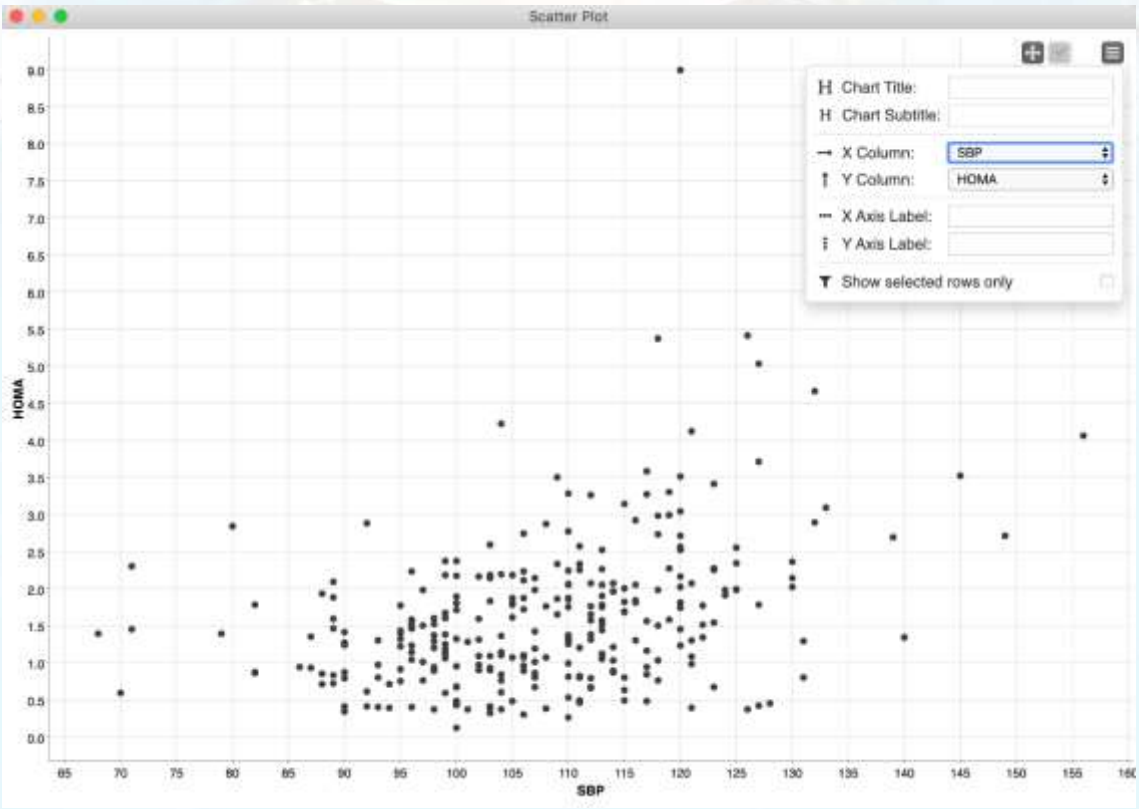


Figure 3. A linear regression data flow: visualization of the SBP variable with respect to HOMA.

Once the data has been read, it is used as an input for a "Partitioning" node to obtain a set of test and training data for the model by applying the hold-out procedure. Figure 4 shows the configuration and parameter dialog box for this node; the percentage of the first partition can be set absolutely or relatively and the type of partitioning ("From the top", "Linear", or "Random" sampling) can also be defined. In addition, a seed can be set so that the random partitioning follows the same pattern in different examples.

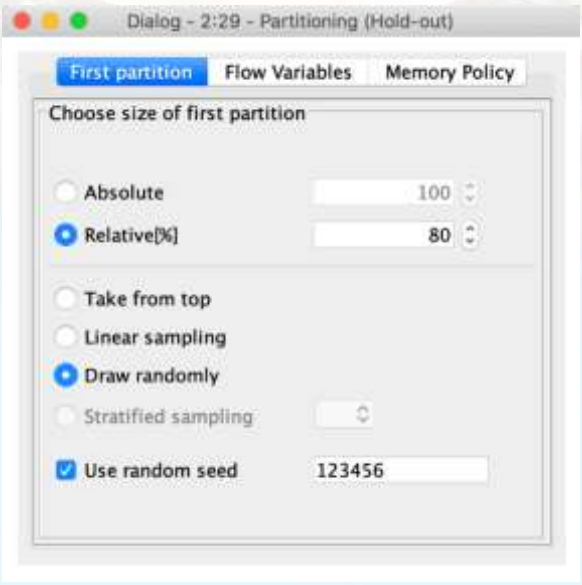


Figure 4. A linear regression data flow: partitioning options and parameters.

The "Linear Regression Learner" node uses the training data to create the model. Figure 5 shows the configuration options of this node, including the ability to select the target variable (in this example the HOMA variable), as well as the different variables to be included in the regression model learning process. The treatment we want to apply to the missing values can also be indicated.

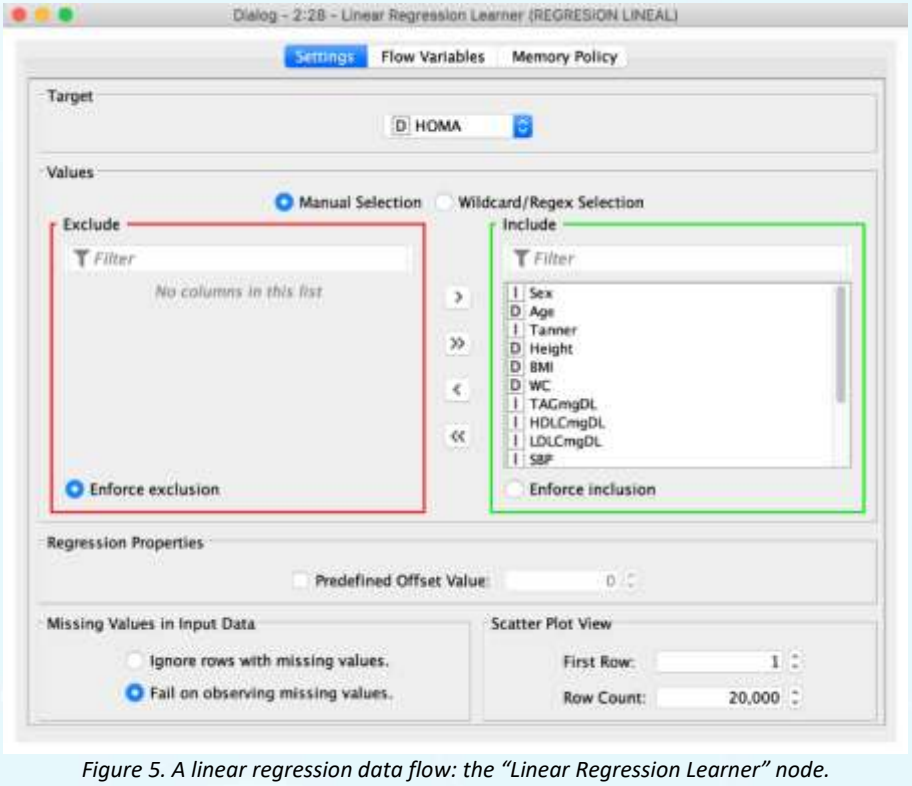
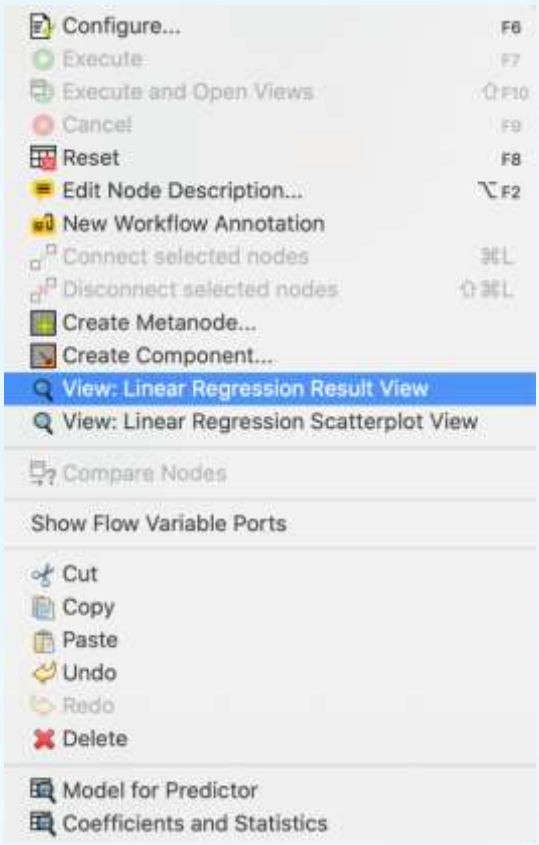


Figure 5. A linear regression data flow: the "Linear Regression Learner" node.

The output of the "Linear Regression Learner" node is two ports, one corresponding to the regression model and the other with the coefficients and statistical data about the regression model. Once the node is executed (green traffic light), the coefficients and statistical data can be viewed by right-clicking on the node and clicking on the "Linear Regression Results View" drop-down menu option. This displays the statistics of the model, including the coefficients, standard error, *t*-value, and $P>|t|$ for each of the variables in the dataset (figure 6).



Linear Regression Result View - 2:28 - L...

File

Statistics on Linear Regression

Variable	Coeff.	Std. Err.	t-value	P>|t|
Sex	0.3102	0.1133	2.7372	0.0067
Age	0.0039	0.0529	0.074	0.9411
Tanner	0.1989	0.1562	1.2738	0.2041
Height	2.3707	0.8868	2.6734	0.0081
BMI	0.1373	0.0264	5.1999	4.60E-7
WC	-0.0219	0.01	-2.1879	0.0297
TAGmgDL	0.0069	0.0019	3.6933	0.0003
HDLCmgDL	0.0097	0.0046	2.0945	0.0374
LDLCmgDL	-0.0029	0.0021	-1.3704	0.172
SBP	0.0047	0.0048	0.965	0.3356
DBP	0.009	0.0063	1.4189	0.1574
Sedentary	0.001	0.0007	1.4902	0.1376
Light	0.0017	0.0013	1.2703	0.2054
Moderate	0.0045	0.0062	0.7232	0.4703
Vigorous	-0.0056	0.0072	-0.7803	0.4361
Intercept	-6.134	1.0655	-5.7567	2.90E-8

Multiple R-Squared: 0.4991
Adjusted R-Squared: 0.4645

Figure 6. A linear regression data flow: regression model results.

We can also graphically display the values estimated by the linear model versus the actual values through the "Linear Regression Scatterplot View" drop-down menu. For example, figure 6 graphically shows the estimated values (straight line) of the "Height" variable versus the actual values.

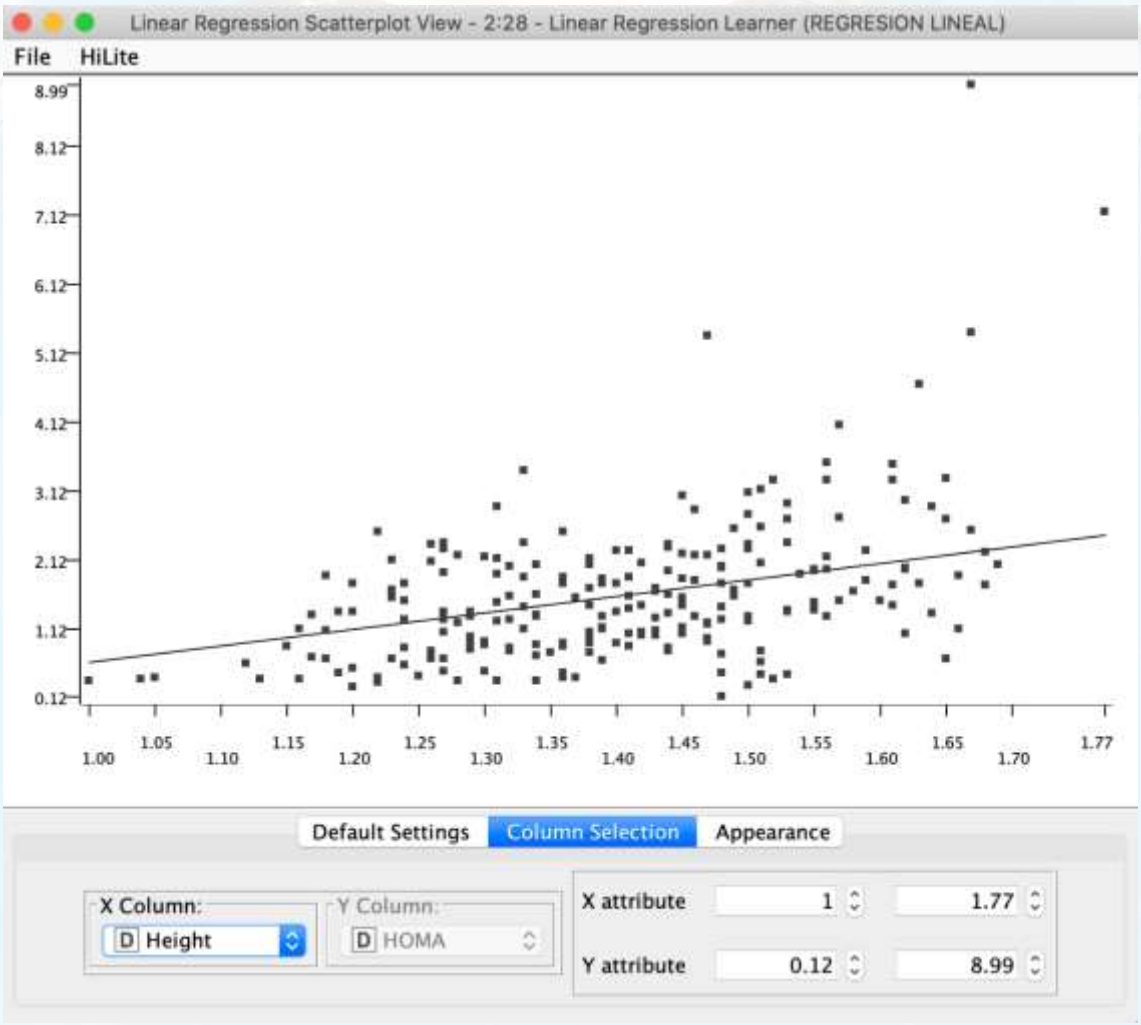


Figure 7. A linear regression data flow: estimated versus actual values of the "Height" variable.

Finally, the "Regression Predictor" node receives the regression model and the test data to evaluate the prediction through a port whose output is the data set to which the prediction is added. Statistics can be obtained from this data as metrics of the quality of the prediction by using the "Numeric Scorer" node. Figure 8 shows some of these metrics, including the R^2, mean absolute error, and mean square error, among others.

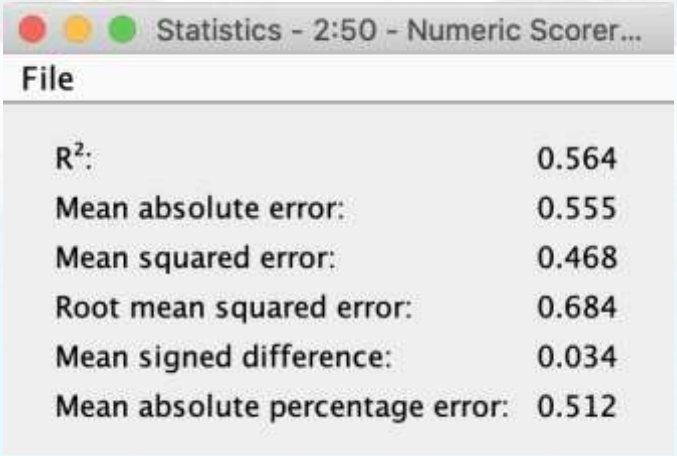


Figure 8. A linear regression data flow: statistical results.

1.1.2. How can we solve a regression problem with a regression tree?

In this data flow, we will use a "Simple Regression Tree Learner" node and a "Simple Regression Tree Predictor" node to conduct a regression (figure 9). Note that to include the node that implements the M5 regression algorithm used in Module 4, Capsule 3 (*Machine learning methods for regression*), we must install an extension in *KNIME*. The aim of this section is to illustrate how to solve a regression problem by using the "Simple Regression Tree Learner" node to represent of the concept of tree regression through the widely known CART algorithm.

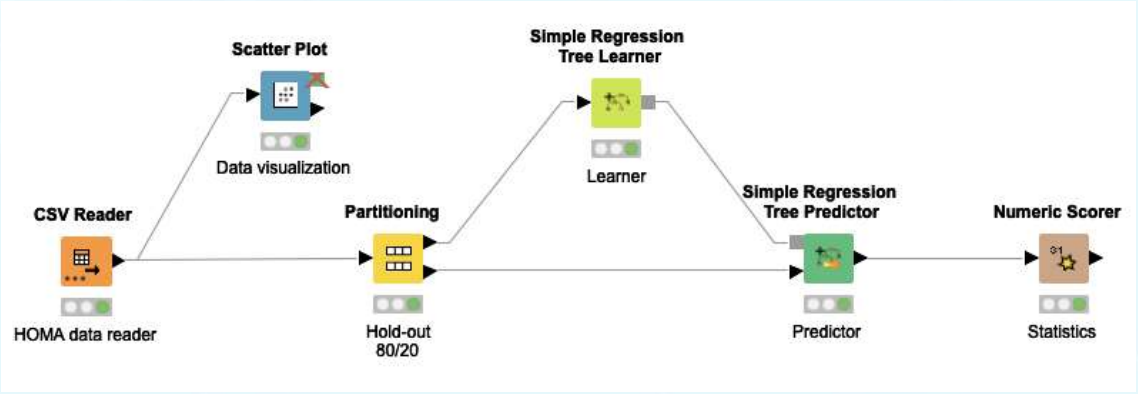


Figure 9. A regression tree data flow.

As in the previous example, data is read from the HOMA dataset and is partitioned with a hold-out of 80% for training and 20% for testing. This data will be used to teach the model and thus, as input to the "Simple Regression Tree Learner" node and for testing as input to the "Simple Regression Tree Predictor" node, respectively. Figure 10 shows the options and configuration parameters of the "Simple Regression Tree Learner" node.

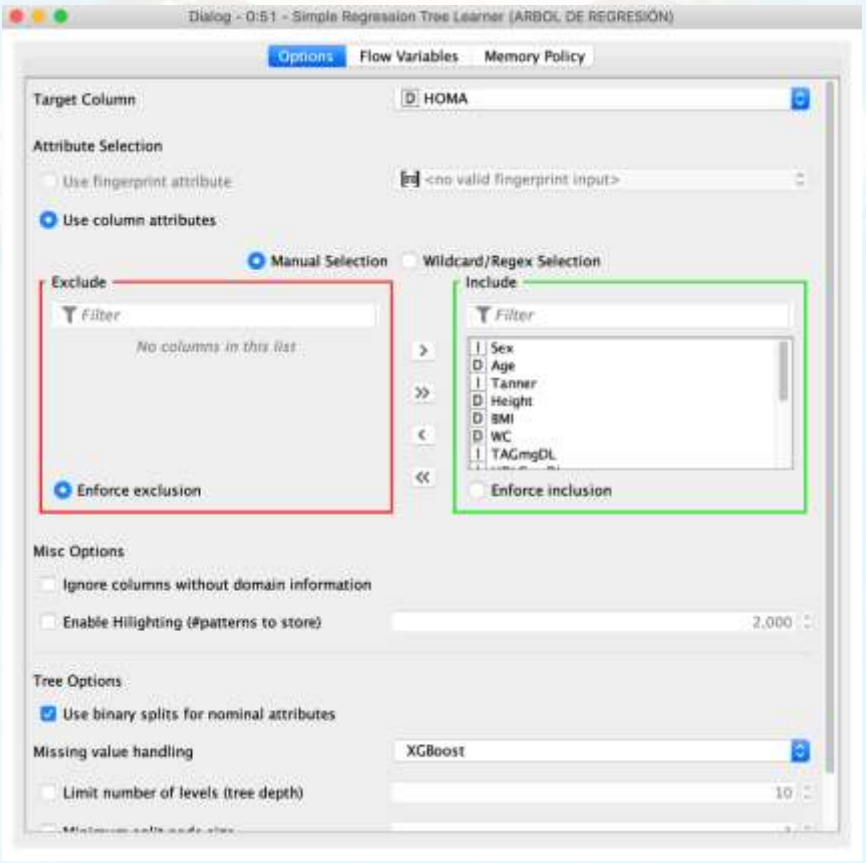


Figure 10. Regression tree options and configuration parameters.

Once the "Simple Regression Tree Learner" node is executed, it learns a tree model which can be visualized through the "View Regression Tree View" menu option (figure 11).

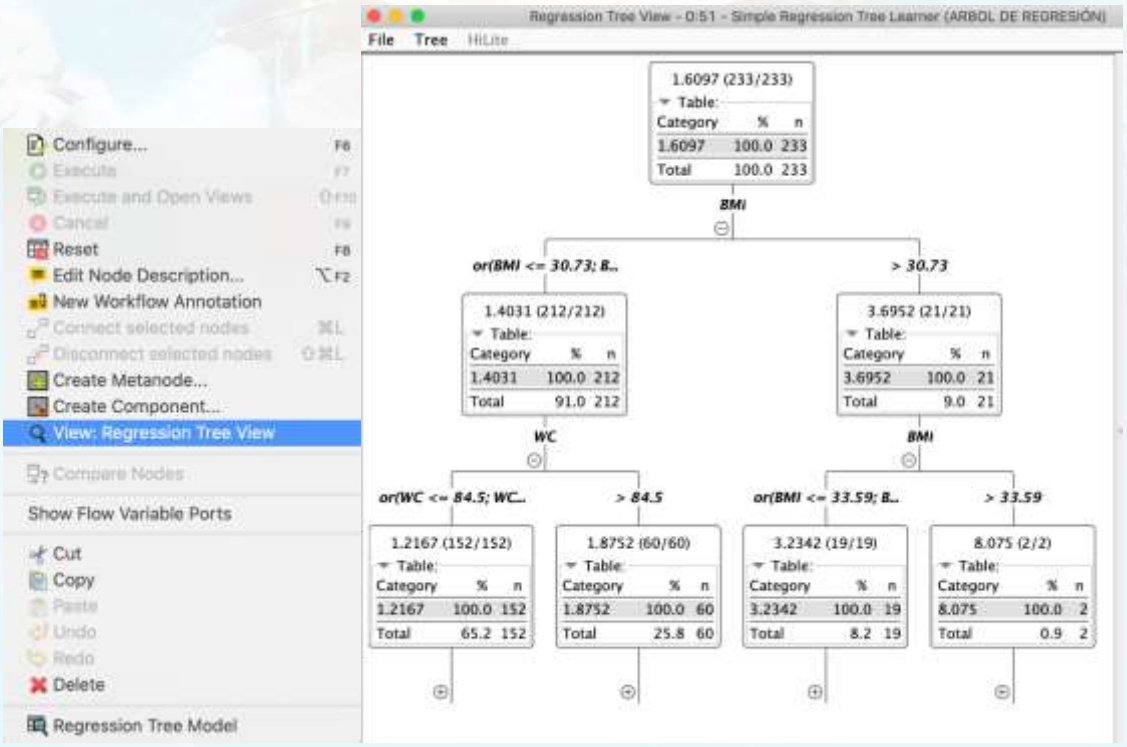


Figure 11. A regression tree visualization.

Finally, as in the previous example, the model taught with the "Simple Regression Tree Learner" node is used to predict with the "Simple Regression Tree Predictor" node. It is connected through the input port (the regression model) and test data are used to evaluate the prediction. Statistics such as the R^2, mean absolute error, mean square error, among others, can be obtained from this data as metrics for the quality of the prediction by applying the "Numeric Scorer" node, as shown in figure 12.

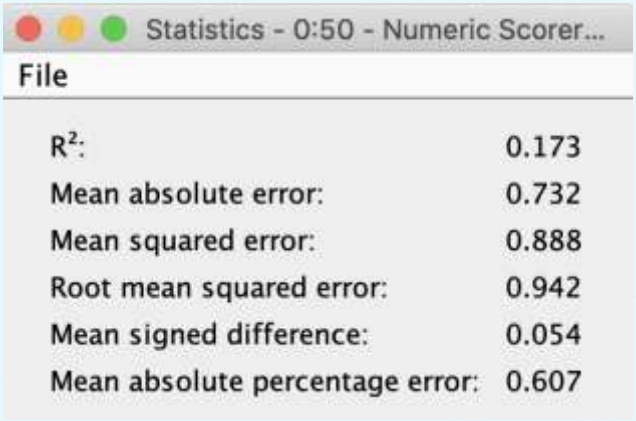


Figure 12. A regression tree data flow: statistical results.

1.2. CLASIFICATION

In this section, we will create data flows to solve a classification problem by employing the KNN (section 3.2.1), decision tree (section 3.2.2), or Random Forest (section 3.2.3) algorithms, considering the gene expression dataset for the skin melanoma problem originally mentioned in Module 5 (*Supervised learning: classification techniques*). Finally, we will also present an example of how to compare different algorithms through a ROC curve (section 3.2.4). Different elements of *KNIME* will be integrated into each data stream to illustrate the usefulness and versatility of *KNIME* in solving data science problems.

1.2.1. How can we solve a classification problem with the KNN algorithm?

The KNN algorithm will be used in this data flow (figure 13). As in all the previous examples, the data will first be read from the immune dataset with "CSV Reader" nodes. Next, we will use the "Column Appender" node to merge the variable data (immune X) with the data associated with the class (immune Y) into a single data table. This data will then be used as the input to a "Partitioning" node, to obtain a set of test and training data for the model. Afterwards, the "*K*-Nearest Neighbor" node uses the training and test data to create the model and display the classification results. We used a "Scatter Plot" node to visualize these results and measure the quality of the model obtained for classification, and where the confusion matrix and statistics (accuracy, *f*-metric, etc.) can be observed, a "Scorer" node was used.

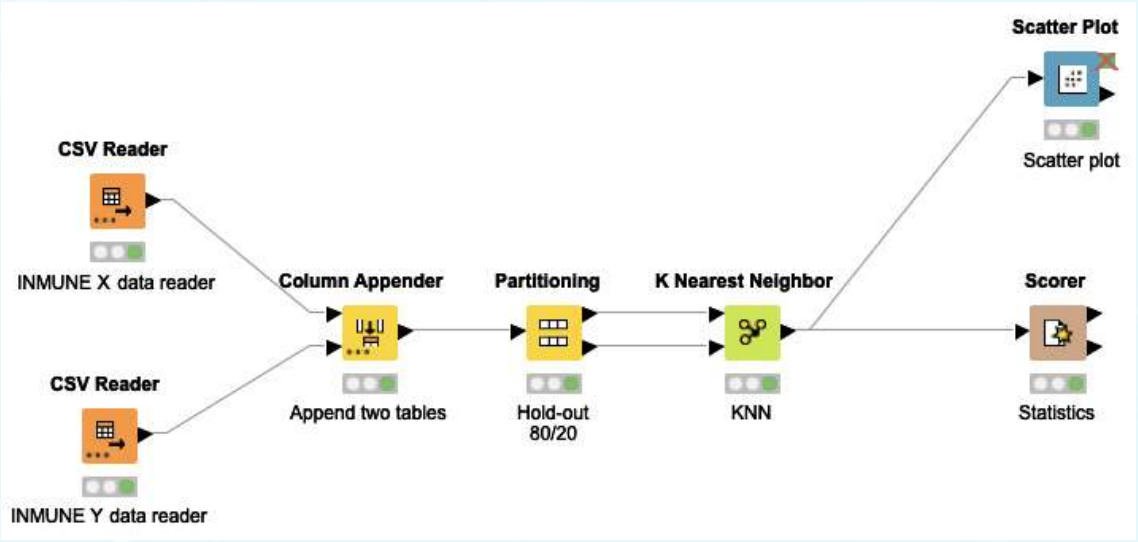


Figure 13. A K-Nearest neighbors algorithm data flow.

The machine learning model in this example is the "*K*-Nearest Neighbor" node with the column of the data table containing the class to be classified, the number of neighbors to be considered, etc. (figure 14).

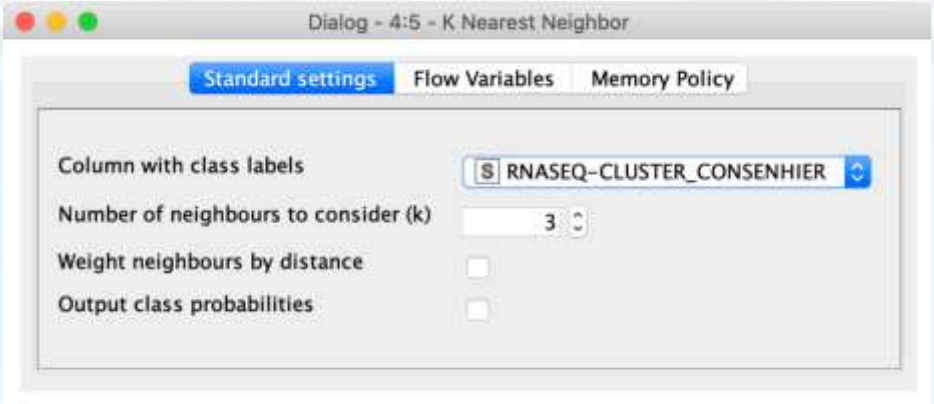


Figure 14. "K-Nearest neighbor" node figuration options and parameters.

We also used a "Scatter Plot" node to allow interactive visualization of the data in two dimensions, configuring the variables to be plotted on the X and Y axis. Figure 15 illustrates an example of this type of visualization. Finally, the "Scorer node" can be used to obtain the confusion matrix (figure 16) as well as information on different quality indicators of the model such as accuracy, precision, recall, and false positives, among others (figure 17).

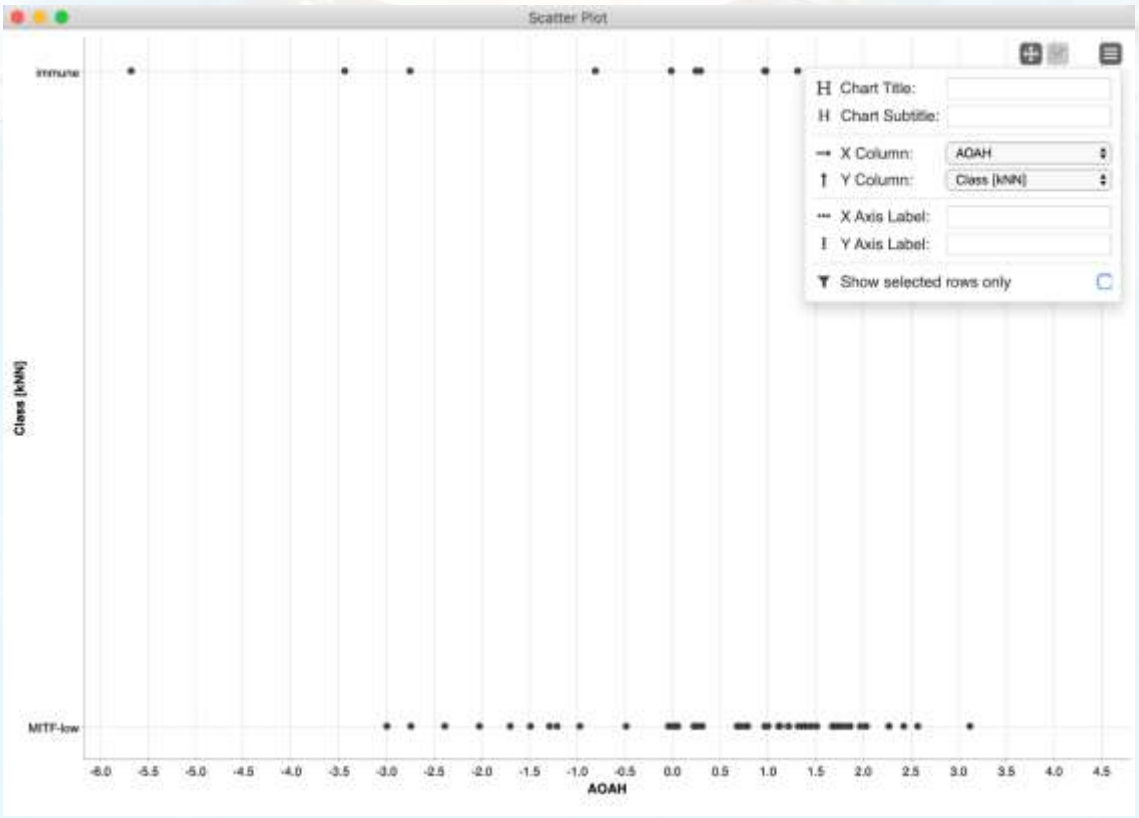


Figure 15. A K-nearest neighbor data flow: an example of a scatter plot.

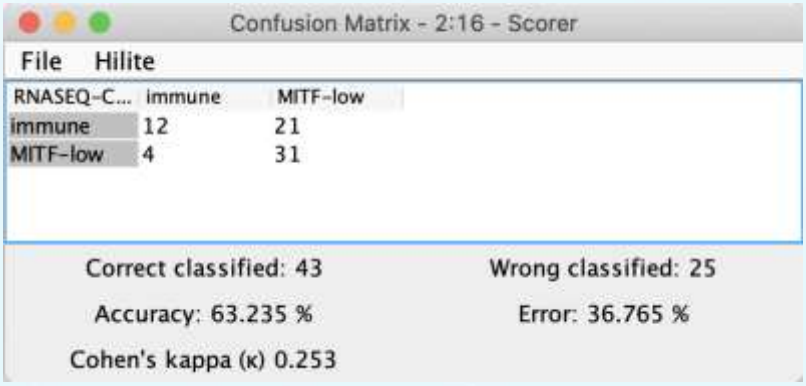


Figure 16. A K-nearest neighbor data flow: an example of a confusion matrix.

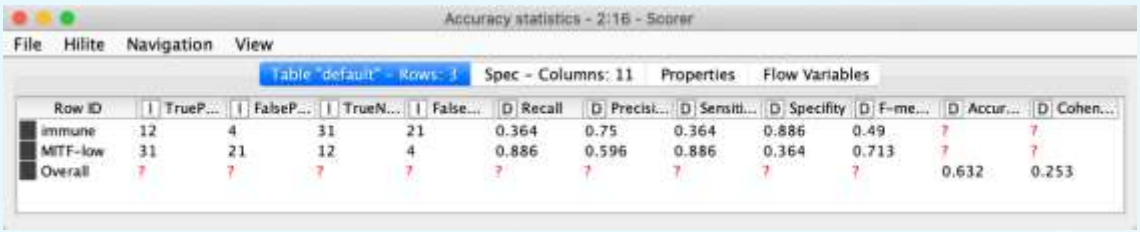


Figure 17. A K-nearest neighbor data flow: an example of a table with statistical results.

1.2.2. How can we solve a classification problem with a decision tree algorithm?

A decision tree will be used in this data flow (figure 18). As in the previous example, the data is first read from the immune dataset with the "CSV Reader" nodes, it is then grouped into a single table and split with the "Column Appender" and "Partitioning" nodes, respectively. In this case, the training data is used as input data for the "Decision Tree Learner" node which learns a model that is then used by the "Decision Tree Predictor" node together with the test data to obtain the classification results. In the specific flow example given here, a "Decision Tree View" node was also used to visually view the decision tree finally obtained. As in previous examples, a "Scorer" node is used to measure the quality of the model obtained for classification.

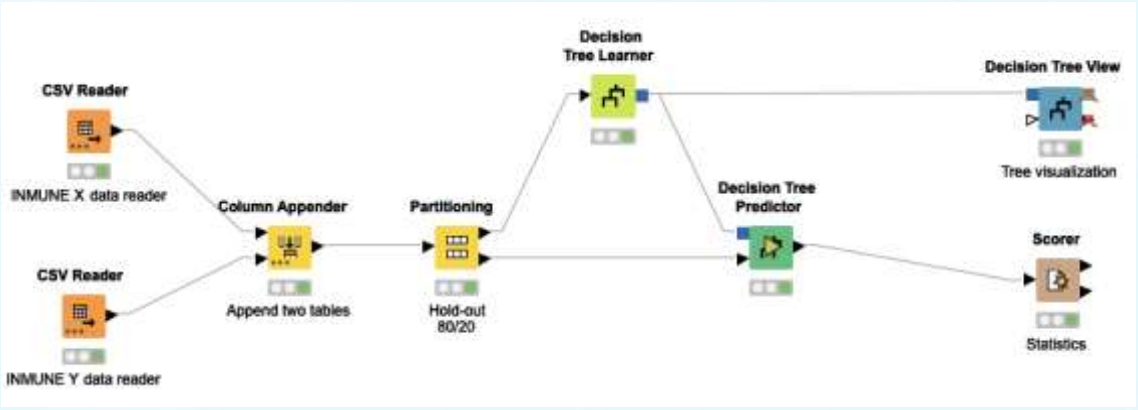


Figure 18. A decision tree data flow.

The "Decision Tree Learner" node has several parameters, as detailed in module 5 (*Supervised learning: classification techniques*) such as the class the data should be classified to, quality metrics, and pruning method, etc. Figure 19 shows the default parameters for this node.

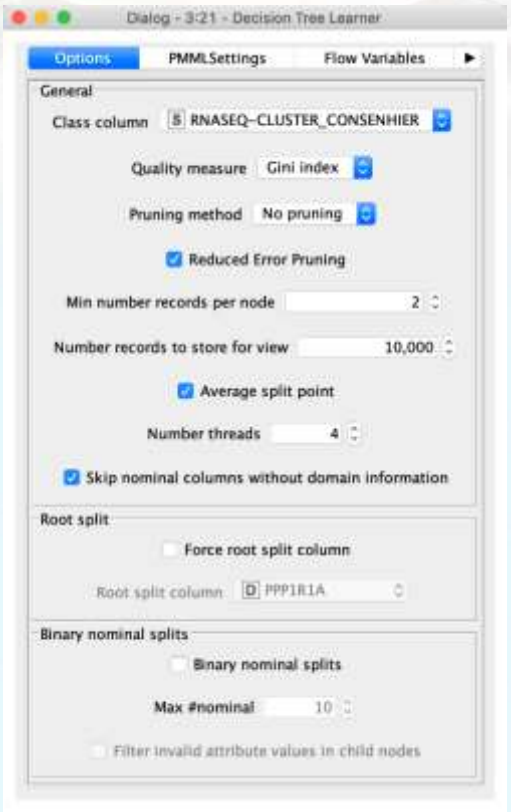


Figure 19.The options and configuration parameters for the Decision Tree Learner" node.

The "Decision Tree View" node in *KNIME* is particularly interesting because it allows the decision tree to be viewed in an interactive way. Figure 20 shows an example of the taught tree.

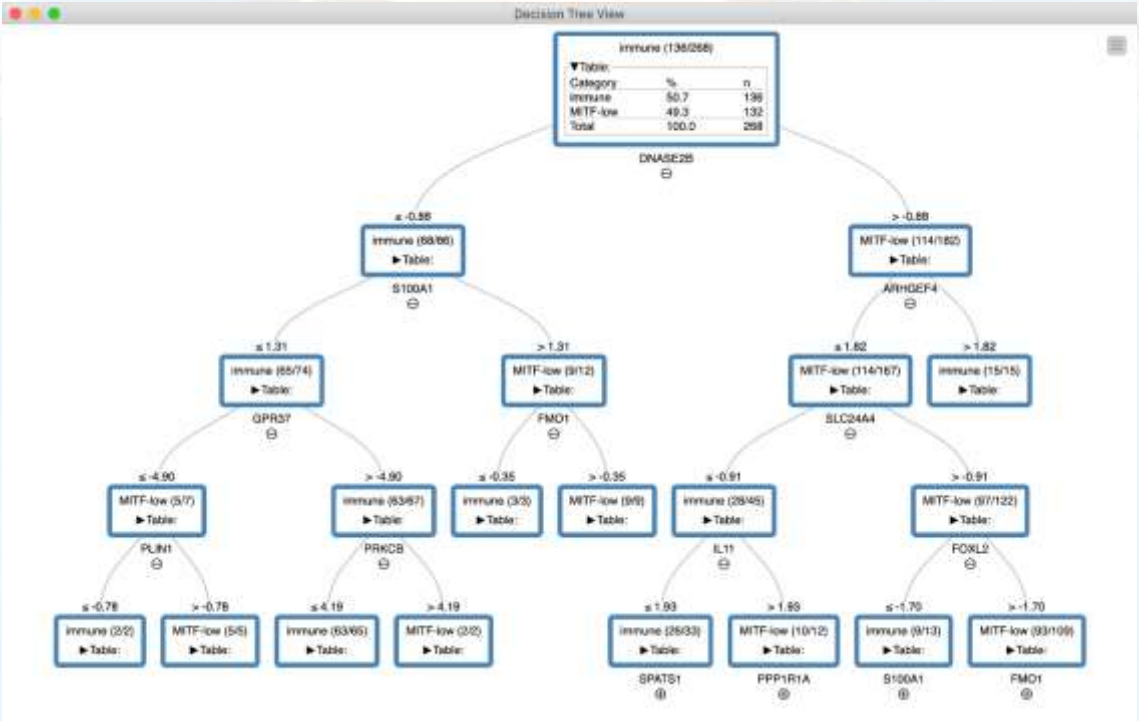


Figure 20. A decision tree data flow: decision tree visualization.

3.2.3 How can we solve a classification problem with Random Forest?

In this data flow, we will use the Random Forest algorithm (figure 21). Of note, in contrast to the previous data flows, this one uses a meta-node element (as detailed in Capsule 1 of this module, *How to use* KNIME*: workflows*) to perform the cross validation. The reading and manipulation of data is the same as in the previous examples, but the "Cross Validation" meta-node will allow us to include the Random Forest algorithm and use cross validation.

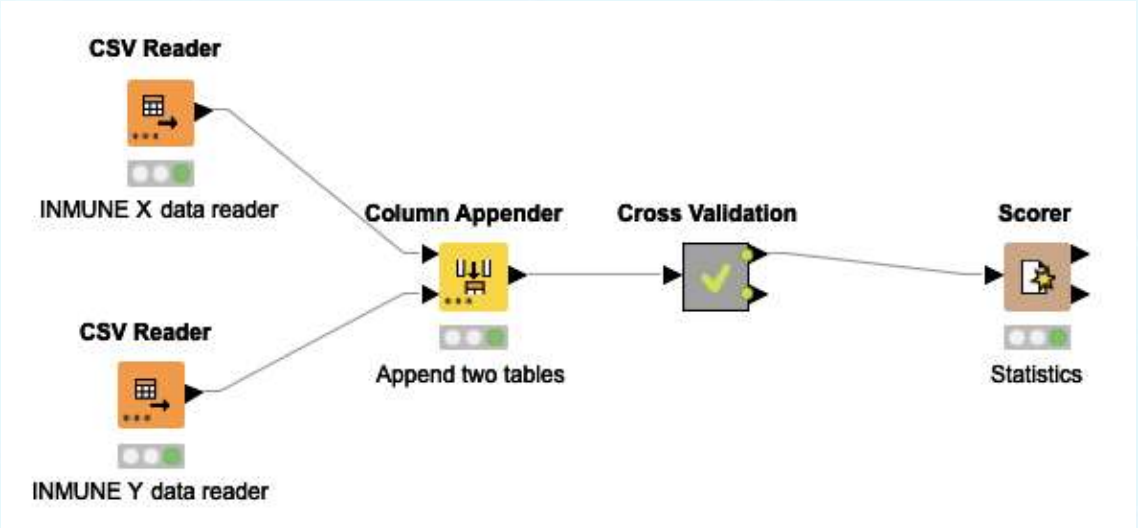


Figure 21. A Random Forest data flow.

Figure 22 shows an example of the "Cross-validation" meta-node.

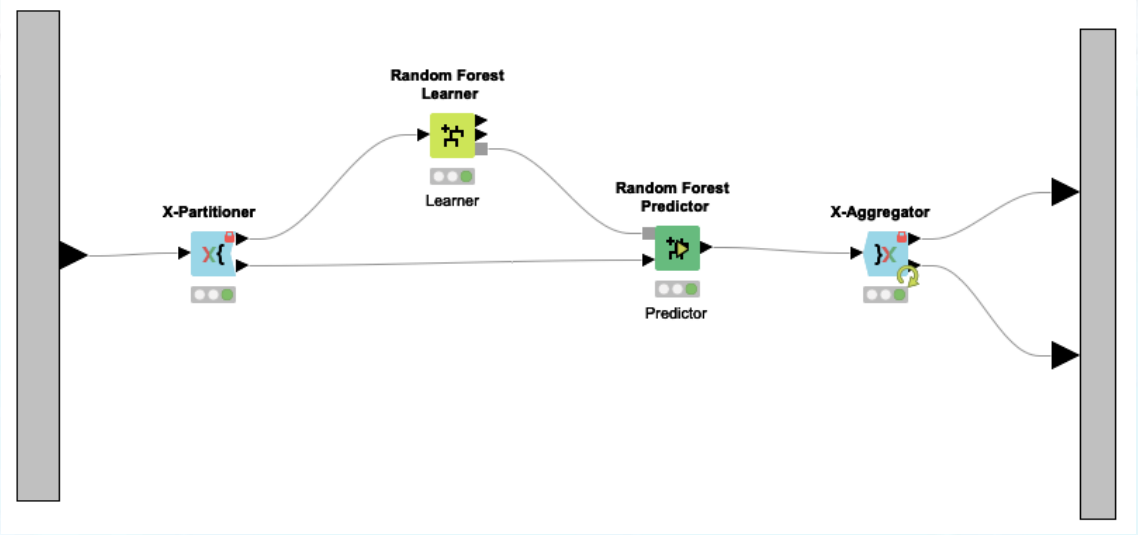


Figure 22. A Random Forest data flow: meta-node cross validation.

In this example, the "Random Forest Learner" and "Random Forest Predictor" nodes are integrated with the "X-Partitioner" and "X-Aggregator" nodes. The first two nodes create the model and make predictions according to the model, respectively, while the "X-Partitioner" and "X-Aggregator" nodes split the data and aggregate the results as many times as the number of validations set in the parameters (i.e., it performs as many iterations as the number of validations we have decided upon). Figure 23 shows the different options and parameters presented by the "Random Forest Learner" node, including the target class, attributes for inclusion in the learning process, and default "Information Gain Ratio" splitting mechanism ("Split Criterion").

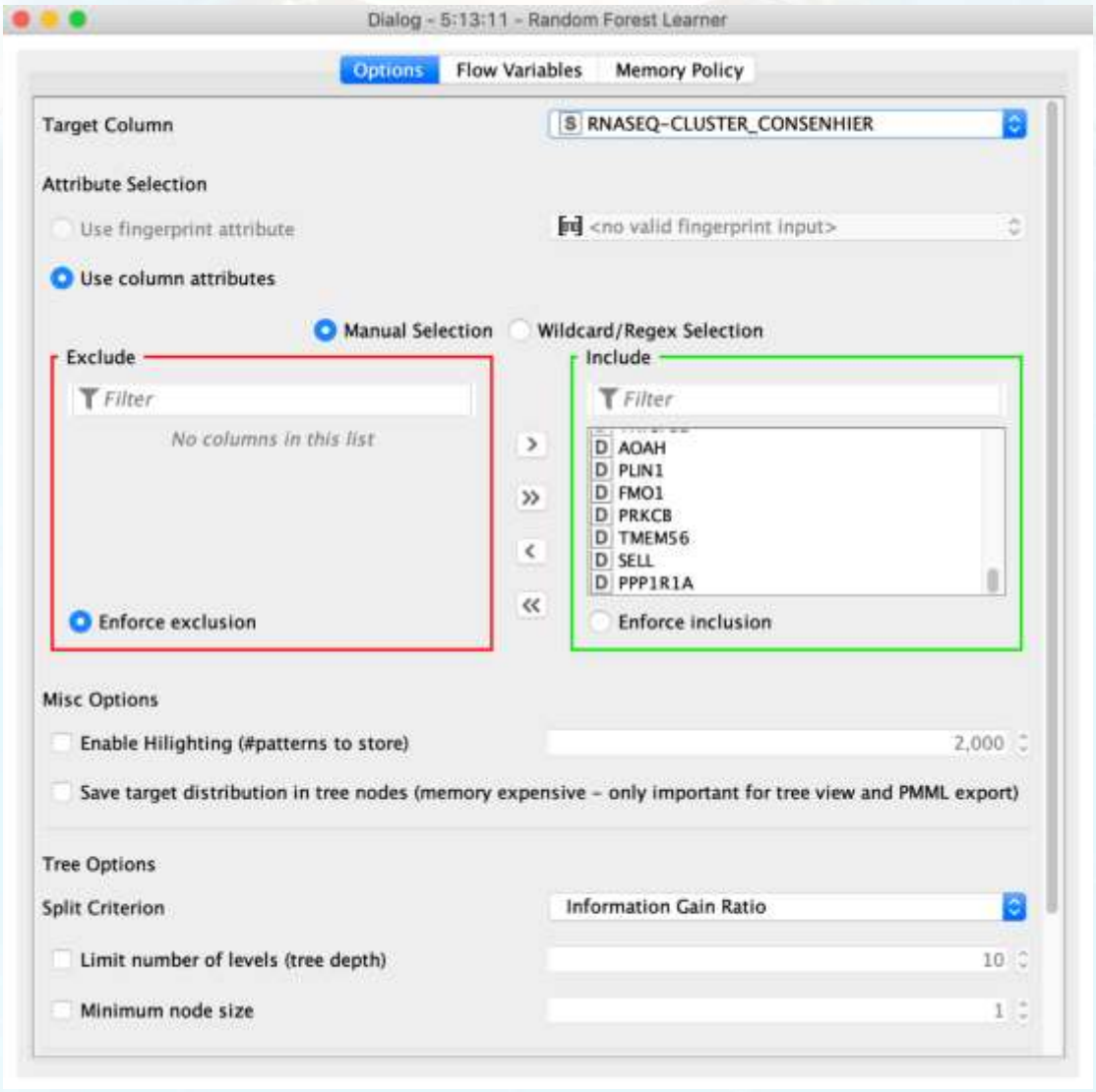


Figure 23. "Random Forest Learner" node options and parameters.

The options and parameters presented by the "X-Partitioner" node are the "Number of validations (folds)", option of "Linear", "Random", or "Stratified" sampling, and target "Class column" (figure 24) while the "X-Aggregator" node only has as parameters for the "Target" and "Prediction" columns (figure 25).

IMPORTANT: If you want to perform a comparative experiment on the same machine with several algorithms, it is advisable to check the "Random Seed" option and set a seed so that the same data are always used in the validation process.

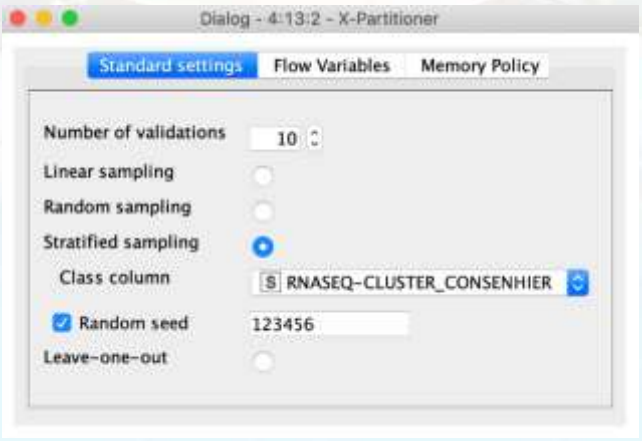


Figure 24. Cross validation of the "X-Partitioner" node options and parameters.

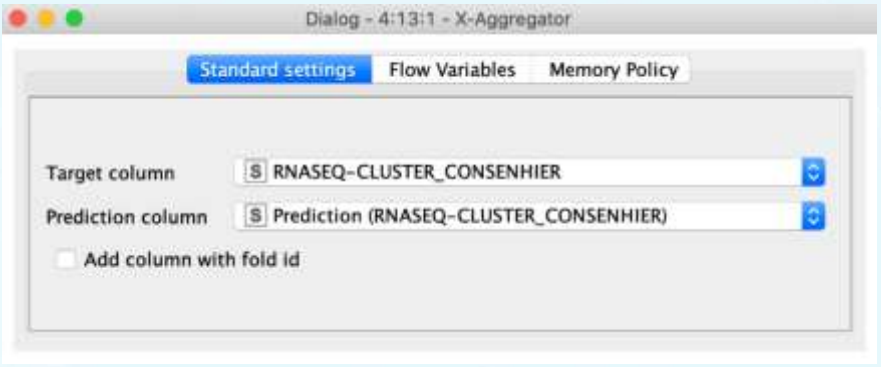


Figure 25. Cross validation of the "X-Aggregator" node options and parameters.

3.2.4 How can we compare different algorithms?

In this section we will detail a data flow to compare different algorithms when using the same data set, in an analogous way to comparisons of regression and classification problems. To avoid extending the material, we will perform a cross-validation with all the algorithms (KNN, decision tree, SVM, and Random Forest), representing them all in a ROC curve (as described in detail in Module 3, *Data science and machine learning*). Figure 26 shows the data flow for this example.

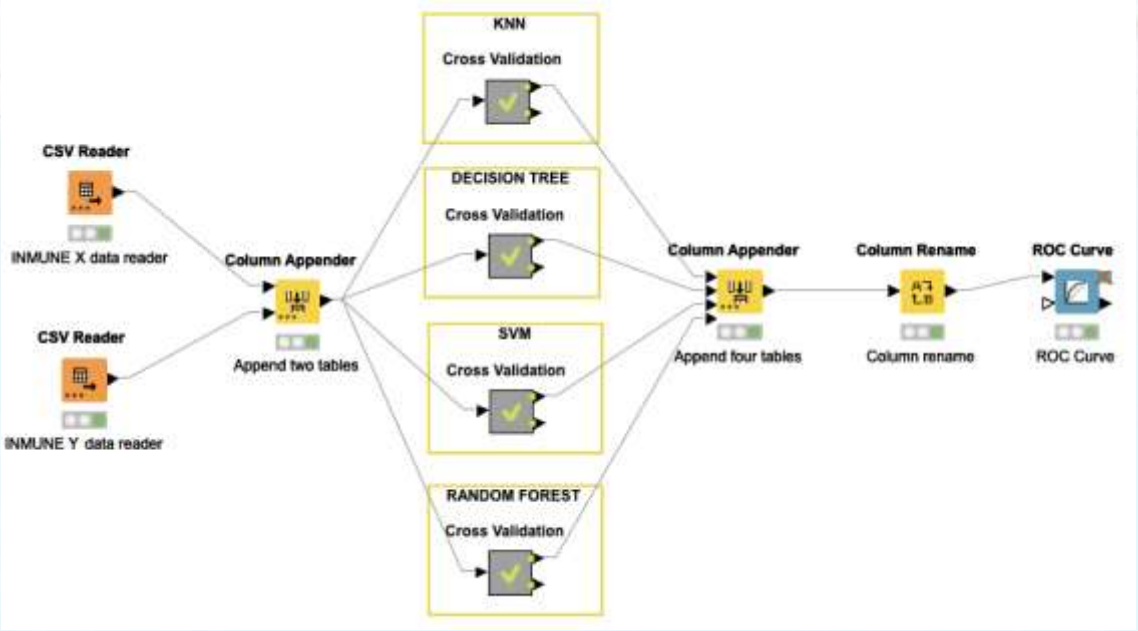


Figure 26. Algorithm comparison data flow

Following on from the previous examples, we will now use the immune dataset. First, we will read the data as in the previous examples and use the "Cross Validation" meta-node to cross validate each algorithm. The output of these nodes, which corresponds to the prediction table results from each of these algorithms, will then be merged into a single table by applying the "Column Appender" node. Next, the column corresponding to the probabilities of the positive class (in this case the immune class) will be renamed with the name of each algorithm so that it is interpretable in the ROC curve (figure 27). Finally, the "ROC Curve" node is used to create the ROC curve where the different areas under the curve for each of the algorithms can be seen on the same graph (figure 28).

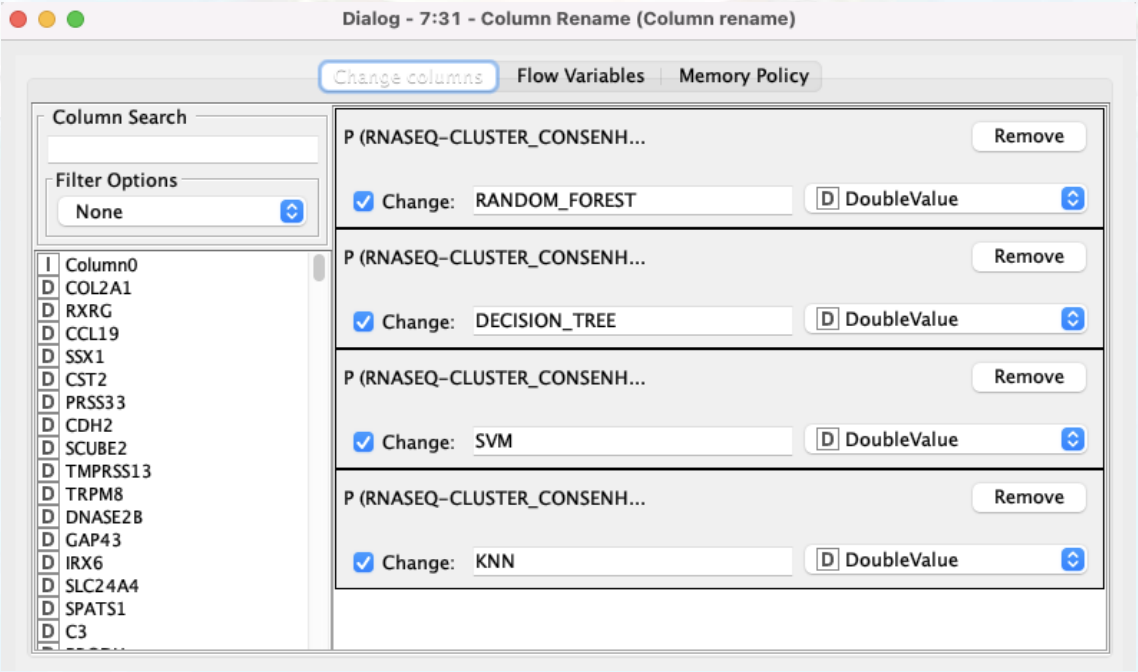


Figure 27. The "Column Rename" node options and parameters.

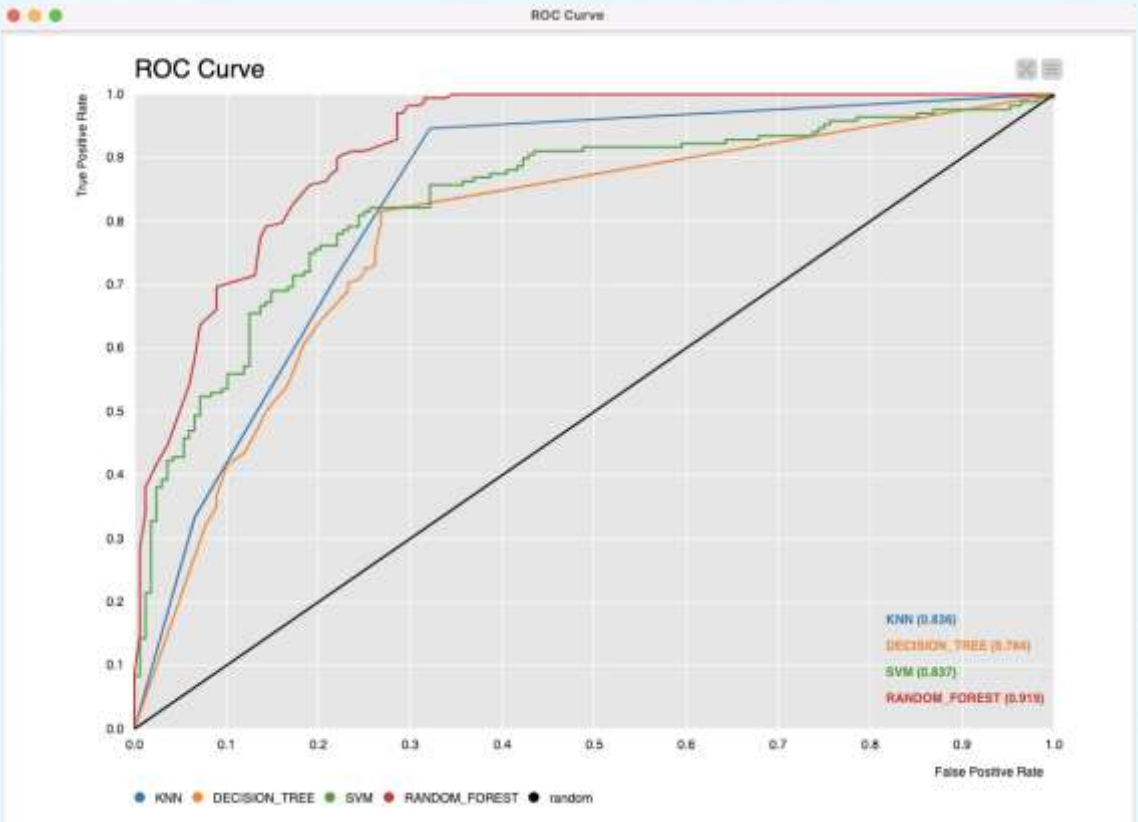


Figure 28. Algorithm comparison data flow: ROC curve.

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.