Comparing Classification Results for KNIME and Spark

You may have noticed that the classification results from KNIME and Spark MLlib decision trees are not exactly the same. This document describes the differences in the setup that can lead to this disparity.

Random Seed for Partitioning

In the step to partition data into training and test sets, random sampling is used in both KNIME and Spark MLlib. However, note that different random seed values are used. A value of 12345 is used for KNIME, and 13234 is used for Spark. Different seed values will generate different random number sequences. Since the selection of which samples go into the training vs. the test partition is based on these random number sequences, the contents of the training and test datasets will be different with different seed values. Consequently, different training and test sets will change the performance results of the classifier. Cross validation is a common approach to address this variability in performance with a single partitioning of the data.

Partitioning into Training and Test Sets

You also may have noticed that the sizes of the training and test sets are different between KNIME and Spark. This is due to the different sampling methods that they use to partition data.

Spark uses a sampling method that does not generate a fixed sample size. So if we specify that the test size should be 20% of the available data, the resulting test size is only approximately 20%, not necessarily exactly.

KNIME uses a different way to randomly select samples to be placed in either training and test set. Due to the different sampling techniques, the contents of the training and test sets are different for KNIME and Spark MLlib.

Decision Tree

KNIME and Spark MLlib use different methods to constrain the complexity of the decision tree model. In both, the minimum number of samples in a node can be specified as a stopping criterion for growing the tree. If the number of samples reaches this threshold, the node is not split.

In Spark, you can also specify the maximum depth of the tree as another stopping criterion (the maxDepth parameter). KNIME does not have this option in its Decision Tree Learner node.

KNIME does have an option for pruning the tree, however. The default setting in the Decision Tree Learner node uses ‘Reduced error pruning’. This prunes the tree by replacing a node with the class of the majority of the samples in that node, if doing so does not decrease the accuracy of the classifier. Spark’s DecisionTreeClassifier method does not have an option to prune the tree.

Performance Results

As we can see, there are several differences in the setup for a decision tree classifier in KNIME and Spark MLlib. These differences result in different trained models, and consequently, different classification performance numbers.