Comprehensive PMML Preprocessing in KNIME

Dominik Morent
KNIME.com GmbH
Technoparkstrasse 1
Zürich, Switzerland
dm@knime.com

Kostantinos Stathatos
Zementis Inc
San Diego, CA, United States
kostas.stathatos@zementis.com

Wen-Ching Lin
Zementis Inc
San Diego, CA, United States
wenching.lin@zementis.com

Michael R. Berthold
Dept. of CIS
University of Konstanz
Germany
Michael.Berthold@Uni-Konstanz.de

ABSTRACT
This paper describes PMML extensions for the modular open source data analytics platform KNIME adding preprocessing support and the ability to edit existing PMML code. It is also shown how the PMML model representation in KNIME can be used within meta learning schemes such as boosting and bagging.

Keywords
PMML, KNIME, Preprocessing, Meta-Learning

1. INTRODUCTION
KNIME is a modular, open[1] platform for data integration, processing, analysis, and exploration [1]. The visual representation of the analysis steps enables the entire knowledge discovery process to be intuitively modeled and documented in a user-friendly and comprehensive fashion. From the beginning KNIME has supported open standards for exchanging data and models. Early on, support for the Predictive Model Markup Language (PMML) [2] was added and most of the KNIME mining modules natively support PMML, including association analysis, clustering, regressions, neural network, and tree models. With the latest KNIME release, PMML support was enhanced to cover PMML 4.0.

Many existing tools enable predictive models to be exported as PMML, however preprocessing steps cannot usually be included in the export. This, of course, limits applicability of the resulting PMML code substantially. To advance the understanding of PMML and its preprocessing capabilities, Guazzelli et al. [4] recently made a tool available to the data mining community called the "Transformations Generator". This interactive tool allows PMML code to be generated for a sequence of preprocessing steps. The caveat here is that whenever PMML code is made available, it needs to be manually copied to an existing PMML file. The newest version of KNIME, which now includes support for a number of preprocessing nodes, takes this several steps further since it allows for the automatic placement of preprocessing PMML code inside a model file. As a result, not only the predictive models but also the steps required to clean up and transform the data can be visually modeled as a KNIME workflow and exported into (or imported from) one coherent PMML file.

In addition, KNIME nodes can be used to replace parts of an existing PMML file with the result that KNIME workflows can be used to modify PMML code. Finally, the expanded PMML support now means that built-in meta-learning schemes, such as boosting and bagging can be used to generate, manage, and evaluate ensembles of PMML documents in KNIME.

[1]KNIME is downloadable from http://www.knime.org

2. GENERATION OF PMML MODELS
Most of the KNIME mining modules natively support PMML. Several PMML models can be produced as well as consumed, such as TreeModel, NeuralNetwork, ClusteringModel, RegressionModel, GeneralRegressionModel, SupportVectorMachineModel. Besides being used as an internal storage format, PMML provides the flexibility to export those models for usage in all PMML enabled scoring engines like the Zementis ADAPA Decision Engine [3]. In the opposite direction it allows KNIME to consume models produced by other tools and use them for scoring.

A typical workflow for learning and applying a decision tree model is illustrated in Figure 1. After reading in the data it is partitioned into two data sets, one for training the decision tree model in the "Decision Tree Learner" and an independent test data set to validate the trained model using the "Decision Tree Predictor". Finally the model and the prediction are evaluated in the "Scorer" which calculates a confusion matrix and accuracy statistics and the results can be interactively explored with the "Interactive Table". The exchange of the PMML model in the workflow is symbolized by the two connected small blue squares attached to

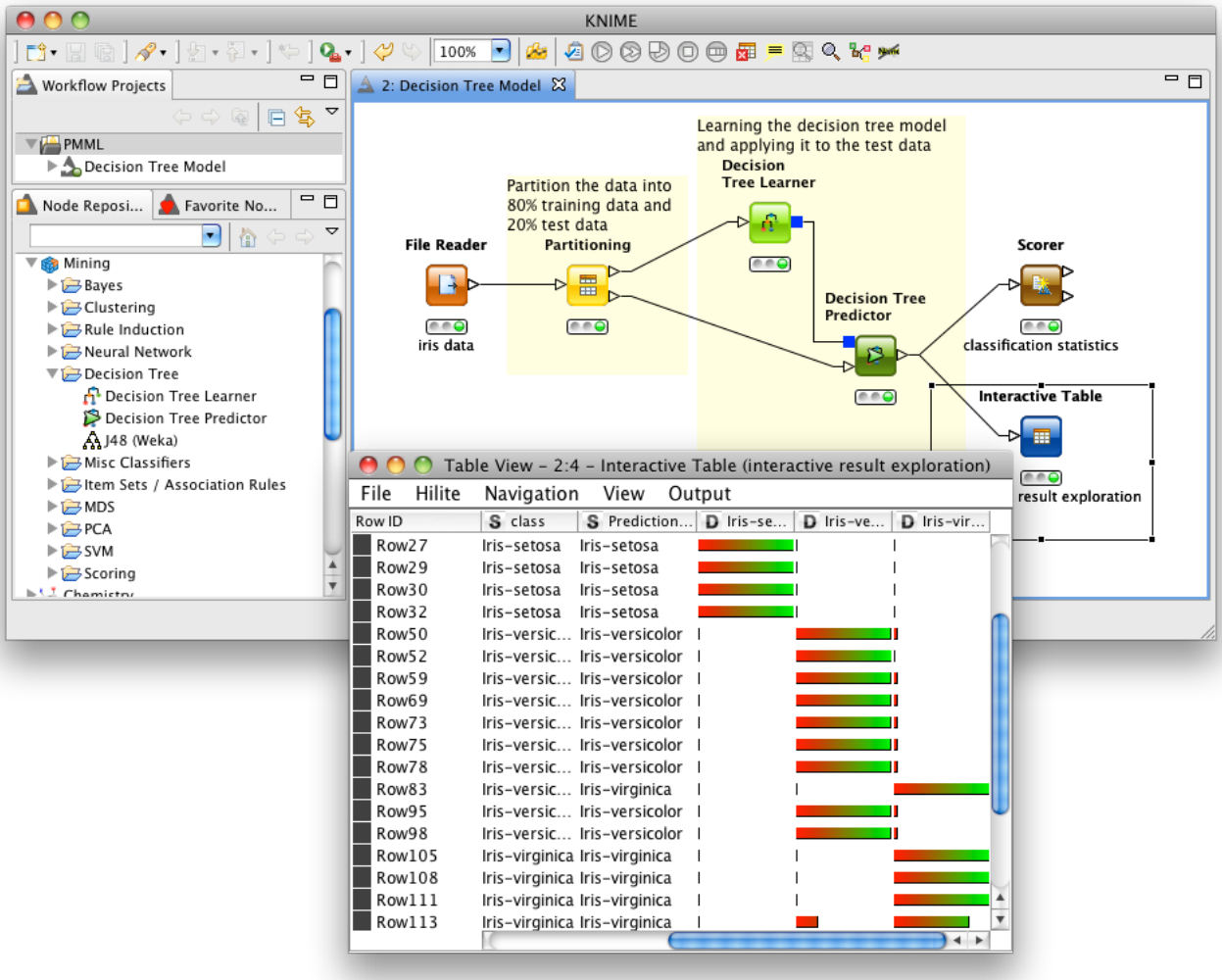

Figure 1: Workflow creating a PMML decision tree model and applying it to test data.

the learner and predictor node.

The "PMML Reader" and "PMML Writer" provide import and export functionality facilitating the exchange of PMML with other tools. Figure 2 shows a typical application of those two nodes. Instead of providing the learned model di-

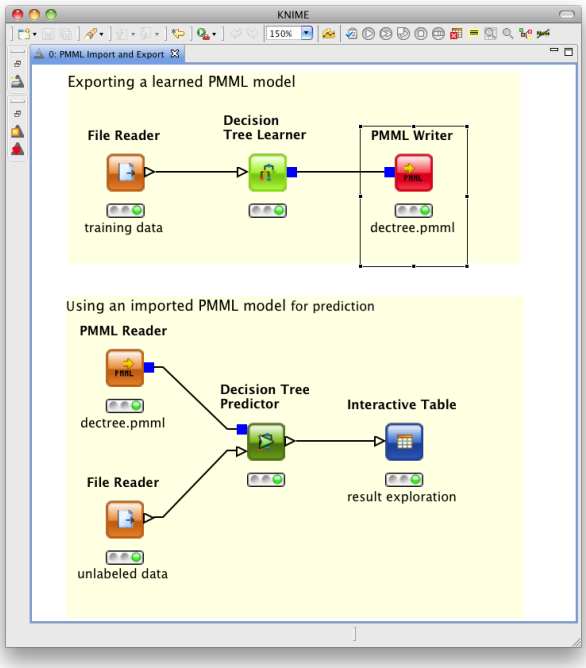

Figure 2: Import and export of PMML models.

rectly to the predictor as shown in the previous example it is passed to the "PMML Writer" which produces a PMML file that is written to disk. Its counterpart the "PMML Reader" allows externally produced PMML models to be imported and applied to data that is read into KNIME whereupon the results can be explored.

3. ADDING PMML PREPROCESSING

The recent KNIME release (version 2.4) comes with additional PMML capabilities. Extending the functionality described in the previous section, many preprocessing nodes offer PMML support and can be included in the generated PMML document. This functionality permits entire data processing flows to be visually modeled in KNIME and exported to PMML. The workflow in Figure 3 shows how multiple preprocessing steps are added to a learned PMML clustering model. The general idea behind the PMML preprocessing support in KNIME is that all preprocessing nodes that are capable of providing or interpreting PMML are given additional "PMML Ports" which are represented by the small blue rectangles[2]. This enables an existing PMML (fragment) to be fed into a node, which subsequently adds more information to the incoming PMML code. All nodes which have connected PMML Ports are part of the PMML

[2]The light blue incoming ports are optional while the dark blue ports must be connected.

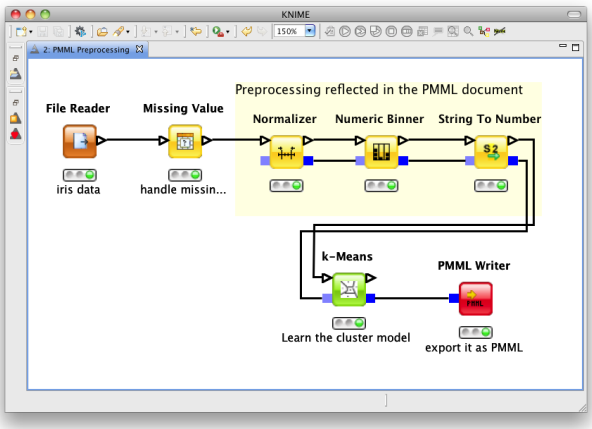

Figure 3: Creating a PMML model including preprocessing steps.

flow in KNIME and are reflected in the final PMML document. Other nodes like "Missing Value" in the example are not visible in the produced PMML. This approach facilitates determination of precisely which parts are added to the PMML and visualizes this selection.

4. VISUAL EDITING OF PMML

Besides exporting PMML-enabled parts of KNIME workflows as PMML, the content of imported PMML models created by other PMML producers can be modified within KNIME. Figure 4 shows a workflow that adds a number of preprocessing operations to an imported PMML model. First a normalization followed by a binning operation and a string conversion is performed. Those operations are then integrated into the PMML model imported by the PMML Reader and exported to a PMML document in the PMML Writer.

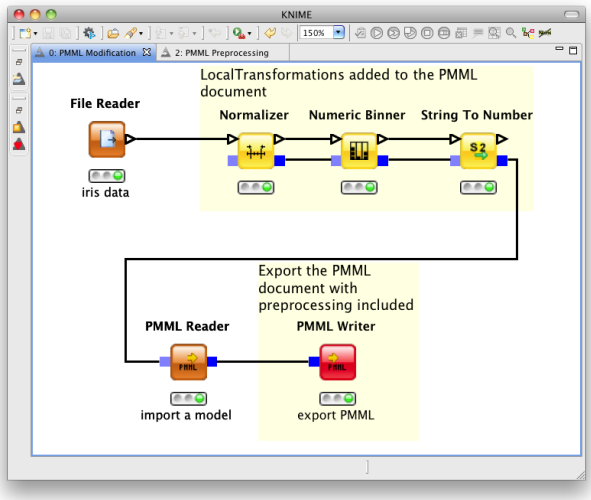

Figure 4: Modifying an existing PMML model to include preprocessing steps.

It is also possible to integrate additional PMML models into an existing PMML document, replace transformations and models or to remove them. This permits existing PMML documents to be edited in a visually well-documented workflow without touching the underlying XML structure and worrying about the validity of the PMML XML-schema.

5. META-LEARNING WITH PMML

The recently introduced PMML data type enables multiple PMML models to be collected in data tables, together with additional information such as model weights. This makes it possible to model meta-learning schemes, such as boosting [5] and bagging [6] as simple KNIME workflows and ultimately also allows the generation of ensembles of PMML models[3]. Figure 5 shows a workflow for a boosting approach. The general idea behind boosting is to combine

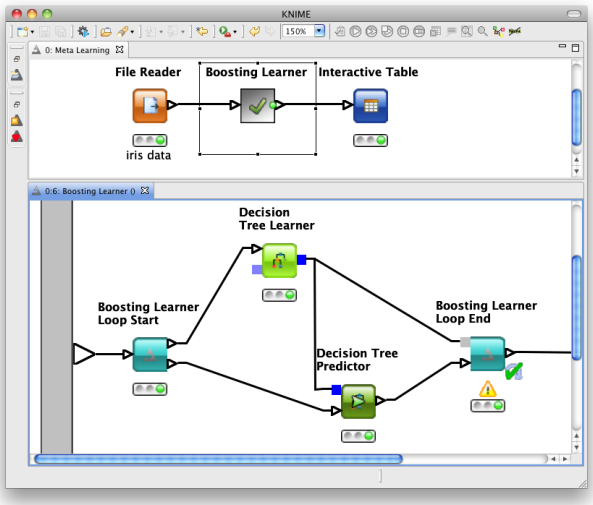

Figure 5: Collecting multiple PMML models for meta-learning (boosting).

multiple weak classifiers weighted by their accuracy into one strong learner (for more details see [5]). The upper part of figure 5 contains the workflow that reads the training data and feeds it into the "Boosting Learner‘" meta-node which is shown in the lower part. It contains a loop that trains a configurable number of weak learners and weights them by their scoring accuracy. The result is a weighted ensemble of PMML models that can be used for prediction with a "Boosting Predictor".

6. CONCLUSIONS

In this short paper we have demonstrated the new PMML preprocessing support introduced in the KNIME 2.4 release. In addition to adding support for PMML-preprocessing it is now also possible to modify (parts of) existing PMML codes modeled by KNIME workflows. We concluded by showing how meta learning schemes such as boosting can also be modeled in KNIME and how this can be extended to create PMML code for ensembles of models.

[3]In the current version a collection of PMML documents is created that contains one model each. Future versions will incorporate support for model ensembles within one PMML document.

7. ACKNOWLEDGMENTS

The authors would like to thank Iris Adae from the University of Konstanz for her valuable feedback and support on enhancing the PMML support in KNIME.

8. REFERENCES

[1] M.R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel. KNIME: The Konstanz Information Miner. In *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*. Springer, 2007.

[2] A. Guazzelli, W. Lin, T. Jena. PMML in Action: Unleashing the Power of Open Standards for Data Mining and Predictive Analytics. CreateSpace, 2010.

[3] A. Guazzelli, K. Stathatos, M. Zeller (2009). Efficient Deployment of Predictive Analytics through Open Standards and Cloud Computing. The ACM SIGKDD Explorations Newsletter, Volume 11/1, July 2009.

[4] A. Guazzelli, W. Lin, T. Jena, and M. Zeller. The PMML Path towards True Interoperability in Data Mining. In the Proceedings of 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, San Diego. To be published.

[5] R.E. Schapire and Y. Singer. Improved Boosting Algorithms Using Confidence-Rated Predictors. In *Machine Learning* 37(3):297-336. Springer, New York, NY, USA 1999.

[6] L. Breiman. Bagging Predictors. In *Machine Learning* 24(2):123-140. Springer, New York, NY, USA 1996.