Erschienen in: PMML Workshop Proceedings : KDD 2013. - New York, NY : ACM, 2013. - ISBN 978-1-4503-2573-8

Ensembles and PMML in KNIME

Alexander Fillbrunn[1], Iris Adä[1], Thomas R. Gabriel[2] and Michael R. Berthold[1,2]

[1]Department of Computer and Information Science
Universität Konstanz
Konstanz, Germany
First.Last@Uni-Konstanz.De

[2]KNIME.com AG
Technoparkstrasse 1
Zurich, Switzerland
First.Last@KNIME.com

ABSTRACT

In this paper we describe how ensembles can be trained, modified and applied in the open source data analysis platform, KNIME. We focus on recent extensions that also allow ensembles, represented in PMML, to be processed. This way ensembles generated in KNIME can be deployed to PMML scoring engines. In addition ensembles created by other tools and represented as PMML can be applied or further processed (modified or filtered) using intuitive KNIME workflows.

Categories and Subject Descriptors

D.2.6 [**Software Engineering**]: Programming Environments—*Graphical Environments, Interactive Environments*; D.2.12 [**Software Engineering**]: Interoperability; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Graphical User Interfaces*; I.5.5 [**Pattern Recognition**]: Implementation—*Interactive Systems*

General Terms

Ensemble, PMML, KNIME

Keywords

PMML, Ensemble Models, KNIME

1. INTRODUCTION

KNIME is a user-friendly and comprehensive open source data integration, processing, analysis and exploration platform [1]. The tool provides a graphical user interface for modeling and documenting complex knowledge discovery processes from data extraction to applying predictive models. KNIME already uses the PMML (Predictive Model Markup Language) standard in many of its nodes. A previous publication [9] introduced KNIME's ability to include preprocessing steps in the PMML model. Various data mining models including for example decision trees, association analysis, clustering, neural network and regression models can also be generated as PMML in KNIME.

PMML is an XML-based exchange format for predictive data mining models [7] and often used to ensure compatibility among different data mining systems. A data mining task defined in one tool can easily be carried out with another if the PMML description of the task is shared; because it is based on XML, it can also easily be edited outside of these tools. 2009 saw the release of PMML 4.0, which includes support for multiple models in a single PMML document. PMML 4.1, released in 2011, further simplifies the representation of such models. The *MiningModel* can contain multiple other models. These models can be of different types and have an optional weight and a predicate, which is an expression indicating whether the model should be used or not.

The most prominent application of such a model collection is ensemble learning and prediction. Ensembles are built using models from multiple, often *weak* learners. The individual weak learner might not be very powerful, however combining many of them frequently increases the quality of the final prediction substantially. Two well known ensemble learning methods are bagging [2] and boosting [5]. We show how these can be realized using a number of KNIME nodes. In bagging, models are trained in parallel on different, randomly chosen samples of the data. During the prediction phase all models are used for prediction and the individual predictions are aggregated into a single result. The final prediction is in most cases even more accurate than that of a single, strong predictor. In boosting, models are trained iteratively, this means that the result of the previously built weak learner is included in the next iteration. The KNIME nodes realize the AdaBoost [6] algorithm. Records misclassified in a previous iteration receive a higher weight and are more likely to be chosen in the next iteration. A consequent weak learner will hence put more focus on these records.

Starting with version 2.8, KNIME allows ensemble models to be generated and modified in the PMML format. Several new nodes enable models to be collected from multiple learners and inserted into a single ensemble model to be used by a variety of existing data processing nodes. There is now a predictor node for PMML ensemble models in KNIME.

In the remainder of this work, first the PMML support in KNIME is outlined. This is followed by an explanation of the ensemble generation process in KNIME as well as details of the new integration of PMML mining models. In the last section we illustrate how externally created mining models can be used and edited.

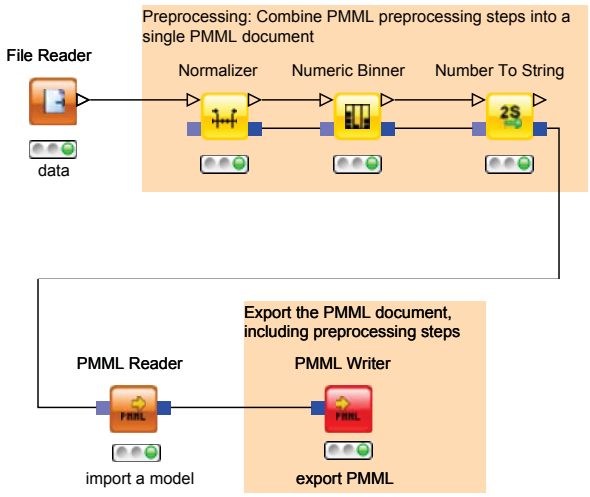

Figure 1: An example of how KNIME includes the preprocessing steps in an externally created PMML document.

2. PMML IN KNIME

In KNIME, data manipulation steps are modeled as connected nodes. The data flows through a workflow following connections between nodes. Each node can have one or more input and output ports. A port can either represent a data table, flow variables or a model. Flow variables and models are used to pass meta information between nodes. The KNIME platform natively supports PMML documents, as it uses them to document preprocessing steps and pass predictive models between nodes. The PMML Writer and Reader nodes in KNIME not only allow models to be stored for later use but also enable models to be exported for further processing with other data mining products such as the ZEMENTIS ADAPA Decision Engine [8].

Starting with KNIME 2.6 the data preprocessing nodes provide an additional optional input and output port, denoted by blue rectangles. In this *PMML port* a PMML document can be passed to the node. The passed PMML document is enriched with the preproccessing steps as generated from the node. All preprocessing steps that take place in the node are documented in the passed document. If no document is passed, the node generates an empty document and includes only information from the node. Figure 1 shows an example preprocessing workflow. Here the data set is read and then preprocessed. During preprocessing normalization, binning and transformation from a numerical to a categorical value is applied. The output PMML of the *Number to String* node contains the information of these three steps. Next an externally generated model is read using the *PMML Reader* and combined to the previously generated preprocessing PMML. In the last step the *PMML Writer* is used to export the new PMML document.

3. GENERATION OF ENSEMBLES

In this section we will show how ensembles can be generated in KNIME. First of all, we outline generation, without the new PMML, support for bagging, boosting and delegating. Afterwards the PMML support is explained using bagging as an exemplary ensemble learning situation.

3.1 Bagging

Bagging is an ensemble learning technique that creates multiple weak predictors and combines them to a single aggregated predictor that may lead to better predictions than each of the original ones. Each predictor is created by training a model on a different, representative part of the data. Even though each of the predictors might make a bad prediction on its own, as a collective they are often able to model the data's underlying concept better. To create a strong predictor, the weak predictions are aggregated using one of many different methods such as taking the mean of a numeric prediction or using the class that was predicted by the majority of predictors for a classification problem.

Ensemble generation in KNIME was already supported in versions earlier than 2.8. It involves splitting the data into chunks, collecting the models in a table, iterating the table in another loop and applying the model on the input data. Afterwards the prediction results need to be collected and a voting loop calculates the final prediction. Figure 2 shows a workflow that can be used for learning and prediction, which creates an ensemble this way. The data are shuffled and then split into chunks, which are processed in a loop. For each chunk a learner learns a model which is then passed to a *Model Loop End* node. This node collects the models from all iterations and outputs a table with all the models once the last iteration has finished. In the second loop, another specialized node, the *Model Loop Start*, iterates the table with models and passes each model into a predictor node, which also has access to the data to be predicted. A *Voting Loop End* node finally produces an output by collecting the predictions from the predictor node and, for each record, selects the prediction that was made by most of the models.

3.2 Boosting

Another ensemble learning strategy that can be realized with KNIME nodes is boosting: the AdaBoost algorithm [6]. As briefly outlined in the introduction, the AdaBoost algorithm assigns a higher weight to records that were misclassified in the previous iteration. In the subsequent iteration the sample is chosen based on this weight: therefore records with a higher weight are more likely to be chosen. Consequently the next base learner focuses on patterns incorrectly classified by the previous base learner.

For boosting KNIME offers preconfigured meta nodes, which turn setting up a boosting workflow into fairly straight forward process. Two special meta nodes, namely the *Boosting Learner* and the *Boosting Predictor* node support such workflow. The content of a *Boosting Learner* for multilayer perceptrons is shown in Figure 3. The input to the *Boosting Learner Loop Start* is a data table containing the training data. In the loop, the *RProp MLP Learner* first trains a model on the data and then passes it to the *PMML Predictor*, which forms a prediction on the training data using the model. In the *Boosting Learner Loop End* the performance of the model is evaluated and another iteration is triggered until the model is considered good enough. Then a data table with multiple models is sent to the output port.

Figure 4 shows the content of the associated prediction meta node. Here the models in the table that were generated by the learner are used to make a prediction for the incoming data. The *Boosting Predictor Loop End* then selects the best prediction. The *Scorer* node at the end can be used for debugging purposes, for example with a confusion matrix.

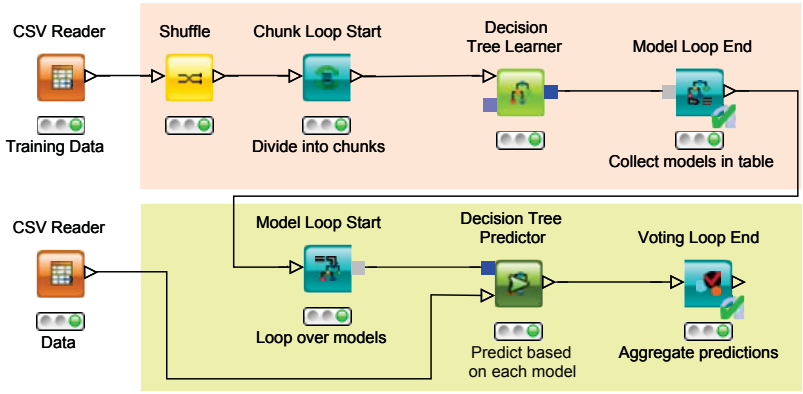

Figure 2: Bagging in KNIME. As described in Section 3.1 the basic bagging methodology is shown here. First multiple decision trees are learned on subsets of the data and collected in one table using the *Model Loop End* node. This table is used for predicting the test data, as read from the *CSV Reader*. The prediction and final aggregation of the multiple predicted values is performed in the second loop.

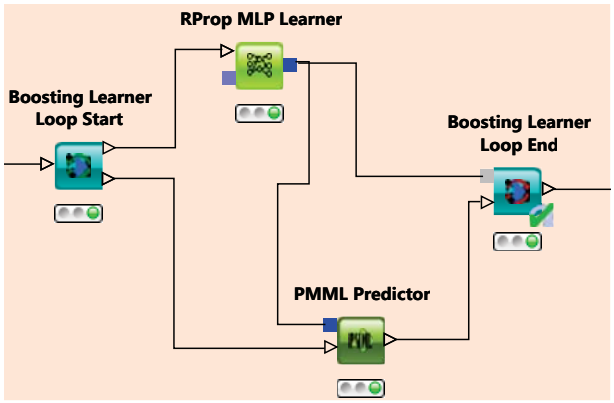

Figure 3: The boosting learner meta node in KNIME. Here a multi layer perceptron is trained in each loop iteration. The final algorithm is performed with a loop construct.

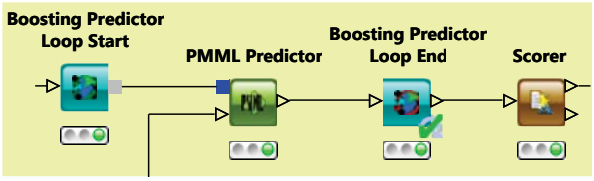

Figure 4: The boosting predictor meta node in KNIME. The general PMML predictor here means that no additional configuration is necessary. For every boosting table the predictor will produce the prediction based on the results of the base learners.

3.3 Delegating

The last ensemble technique implemented in KNIME is a simple version of delegating [4]. For delegating, patterns wrongly predicted from the precedent base learner are forwarded to the next one. Delegating is also implemented in a loop. The *Delegating Loop Start* outputs all the patterns that were either incorrect classified or that had received a classification with a low probability in a previous iteration. In the content of the loop a new model is learned and passed to the first port of the *Delegating Loop End*. The second port receives the insufficient classified data points from this run and sends them back to the *Delegating Loop Start* node. The loop finishes when either no more data points are classified as incorrect or after a fixed number of iterations. In Figure 5 a delegating regression in KNIME is shown.

3.4 PMML MiningModels

The PMML standard has a special model type for ensemble models, which is called a *MiningModel*. It serves as a container for multiple data mining models that are part of an ensemble and also defines the aggregation method that should be used when making a prediction. The PMML *MiningModel* consists of multiple segments. Each segment can be weighted optionally. Each segment contains a model and a predicate which tells the consuming node when it should use the model and when to ignore it. The model types that can be included in such an ensemble include *TreeModel*, *NeuralNetwork*, *ClusteringModel*, *RegressionModel*, *GeneralRegressionModel* and *SupportVectorMachineModel*. While it is possible to have models of different types in one *MiningModel*, these models must all fit to the *MiningModel*'s mining function, which can either be regression, classification or clustering. Another important attribute of a *MiningModel* is the aggregation method that is used to create a single prediction based on the results of the individual models. The PMML standard currently defines ten different aggregation methods of which nine are currently supported by KNIME. These methods include majority vote, sum, average and weighted average. Model chaining is not supported since it is currently not possible to generate nested *MiningModels* from flat tables.

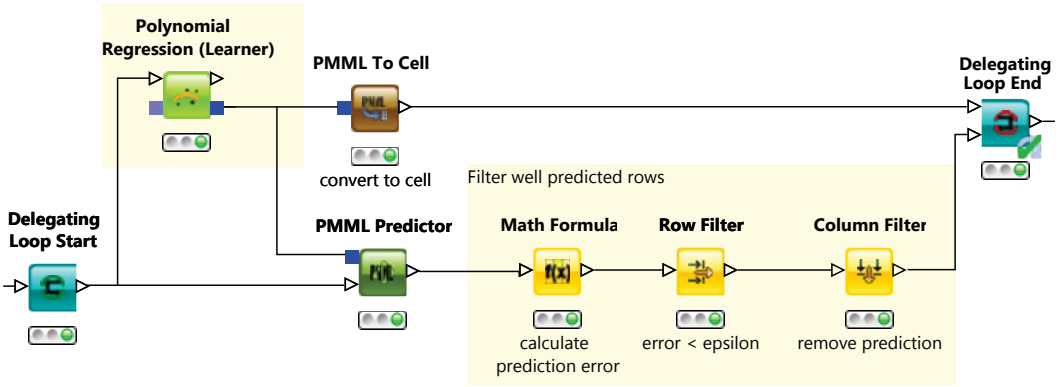

Figure 5: A delegating regression is shown in this Figure. Well predicted data points are filtered, by calculating the error to the original value. Data points with a error higher than a threshold a filtered with the *Row Filter* and the rest is fed back to the start of the loop.

3.5 Creating PMML MiningModels

Where possible (that is: supported in PMML), KNIME's mining nodes use PMML as the format for their models. Using special ports learner nodes output their models as PMML. A corresponding predictor consumes the PMML model together with the dataset to be predicted. As of version 2.7 KNIME also contains nodes that natively support consuming or producing PMML *MiningModels*. The newly introduced nodes allow the integration of several models from one or many learners into a single ensemble model, a so-called *MiningModel* as defined by [3].

There are two different approaches in KNIME to create a PMML *MiningModel*. An ensemble model can be created from a data table using the new *Table to PMML Ensemble* node. In order to generate the model the table must contain one column with PMMLCells and can optionally include a second numerical column used to assign a weight to each model in the PMML column. E.g. the *Boosting Learner Loop End* in Figure 3 and the Model Loop End in Figure 2 produce such a data table. The second approach feeds the models into the new *PMML Ensemble Loop End* node directly, where they are collected and output to the PMML *MiningModel* in the final iteration of the loop. Model weights can be assigned to the Loop End node by using a flow variable. These variables are passed from node to node together with the main data but have an extra port, denoted by a red circle as the input port.

3.6 Bagging realization with PMML

For bagging with PMML ensemble models, the data first has to be split into chunks using the *Chunk Loop Start* node. The number of chunks the data is divided into is equal to the number of models the ensemble will contain. Shuffling the data beforehand is recommended in order to give each individual learner a representative chunk of data. In the loop initiated by the *Chunk Loop Start* the learner is applied to each chunk of data and passes the trained model to a *PMML Ensemble Loop End* node, which collects the models and outputs the ensemble once the loop terminates. The aggregation method for the ensemble can be set in the configuration of the *PMML Ensemble Loop End*. Additionally it is possible to assign each model a weight by passing a flow variable. This is a specialty only available with the new *MiningModel* based realization. Although the *Model Loop End* collects models as well it does not take weight into account. When a flow variable for weight is given, the Loop End checks the variables value in each iteration and assigns it as the currently trained models weight. While a minimal bagging workflow previously needed two loops and seven nodes, it can now be constructed with only one loop and five nodes. Figure 6 shows a bagging workflow that uses the new *PMML Ensemble Loop End* node. In Figure 7 the weight is read from a table and assigned via the optional flow variable port. The dialog contains configuration possibilities for the weight and the aggregation method.

4. WORKING WITH PMML ENSEMBLES

In this section we show how ensembles, which are built using the *PMML Ensemble Loop End* or the *Table to PMML Ensemble* nodes can be modified and how ensemble prediction in KNIME works.

4.1 Modifying PMML MiningModels

Apart from creating models, KNIME also supports the modification of ensembles and the retrieval of a subset of models from an ensemble. This makes it possible for example to load an externally generated mining model, add and remove models or modify a model's weight. To do so, an ensemble can be transformed into a data table with a PMML column for the models and a double column containing the weights of the models whereupon all of the data manipulation nodes available in KNIME can be used. The new node that is available for this task is the *PMML Ensemble to Table* node. KNIME provides several nodes to modify the resulting table, such as the *Row Filter*, *Math Expression* or *Cell to PMML* node. One could even imagine using the XML nodes to directly modify the PMML of each model.

Figure 8 shows a workflow that loads a PMML Mining model from a file. It then splits the ensemble into a table structure, which contains the individual PMML models and the weights. The table is sorted by the model weight and only the top five models are retained. Finally a new ensemble mining model using only those five models is created and stored.

Due to the fact that PMML is a XML dialect, KNIME's XML processing capabilities can be drawn upon when mod-

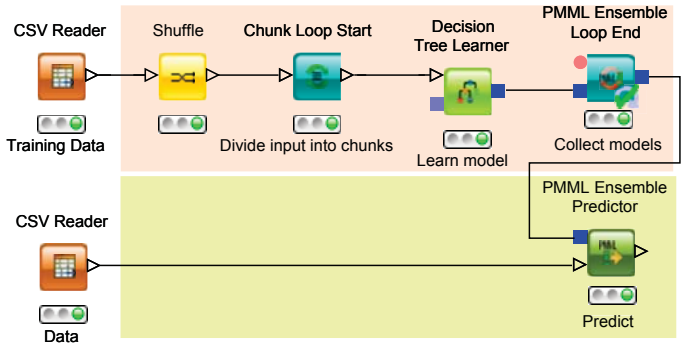

Figure 6: Bagging using the PMML *MiningModels*. The prediction and aggregation of the predicted values is performed by the *PMML Ensemble Predictor*.

ifying various parts of PMML Mining Models, with the exception of from adding and removing models. XML nodes can be used to modify data dictionaries, predicates or model parameters.

4.2 Prediction with PMML MiningModels

The generated ensemble models can also, of course, be used for prediction in KNIME. To make a prediction, each of the models in the ensemble is applied to the data to produce an individual prediction and these are then combined for the final output. We can, of course, do this within KNIME by splitting the ensemble and then applying individual predictors. However, in addition to to the new PMML ensemble creation nodes, there is also a new KNIME node, which can utilize these models to make direct predictions. The *PMML Ensemble Predictor*(as used in Figure 6 or 7) node has two inputs: a PMML Mining model input and a table input for the data to be predicted.

For the prediction the node internally creates a predictor for each model in the ensemble and executes it on the input data. The result is a number of predictions from multiple models. These results are aggregated into a single value. Depending on the mining function of the ensemble, different aggregations, such as Majority Vote or Weighted Average for example, can be applied. For classification, only majority vote, weighted majority vote, select first and select all are applicable since the results from the predictions are not numeric. Results from applying regression models can only be aggregated using average, weighted average, media, max, sum, select first and select all. Clustering models can be used with (weighted) majority vote, select first and select all. The mining function of an ensemble is determined by an XML attribute in the PMML document and the predictor always checks if the selected aggregation is valid and displays an error otherwise.

5. CONCLUSIONS

The new nodes in KNIME provide native support for PMML Mining Models which are either generated internally from KNIME learners or by using the already existing means of loading PMML into KNIME from other sources. Compared to the prior style of storing multiple models in data tables this new approach is easier to use and allows the sharing of other ensemble based mining models with other tools. Ensemble models that are created by the *Table to PMML Ensemble* or the *PMML Ensemble Loop End* nodes are fully compatible with existing KNIME nodes that process PMML. They can be read and written into the file system and stored in table cells.

6. 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] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[3] DMG. Pmml 4.0 - multiple models: Model composition, ensembles, and segmentation. `http://www.dmg.org/v4-0-1/MultipleModels.html`. visited on May 6th 2013.

[4] C. Ferri, P. Flach, and J. Hernández-Orallo. Delegating classifiers. In *Proceedings of the twenty-first international conference on Machine learning*, page 37. ACM, 2004.

[5] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[6] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[7] A. Guazzelli, W.-C. Lin, and T. Jena. *PMML in Action: unleashing the power of open standards for data mining and predictive analytics*. CreateSpace, 2010.

[8] A. Guazzelli, K. Stathatos, and M. Zeller. Efficient deployment of predictive analytics through open standards and cloud computing. *SIGKDD Explor. Newsl.*, 11(1):32–38, Nov. 2009.

[9] D. Morent, K. Stathatos, W.-C. Lin, and M. R. Berthold. Comprehensive pmml preprocessing in knime. In *Proceedings of the PMML Workshop*. KDD 2011, 2011.

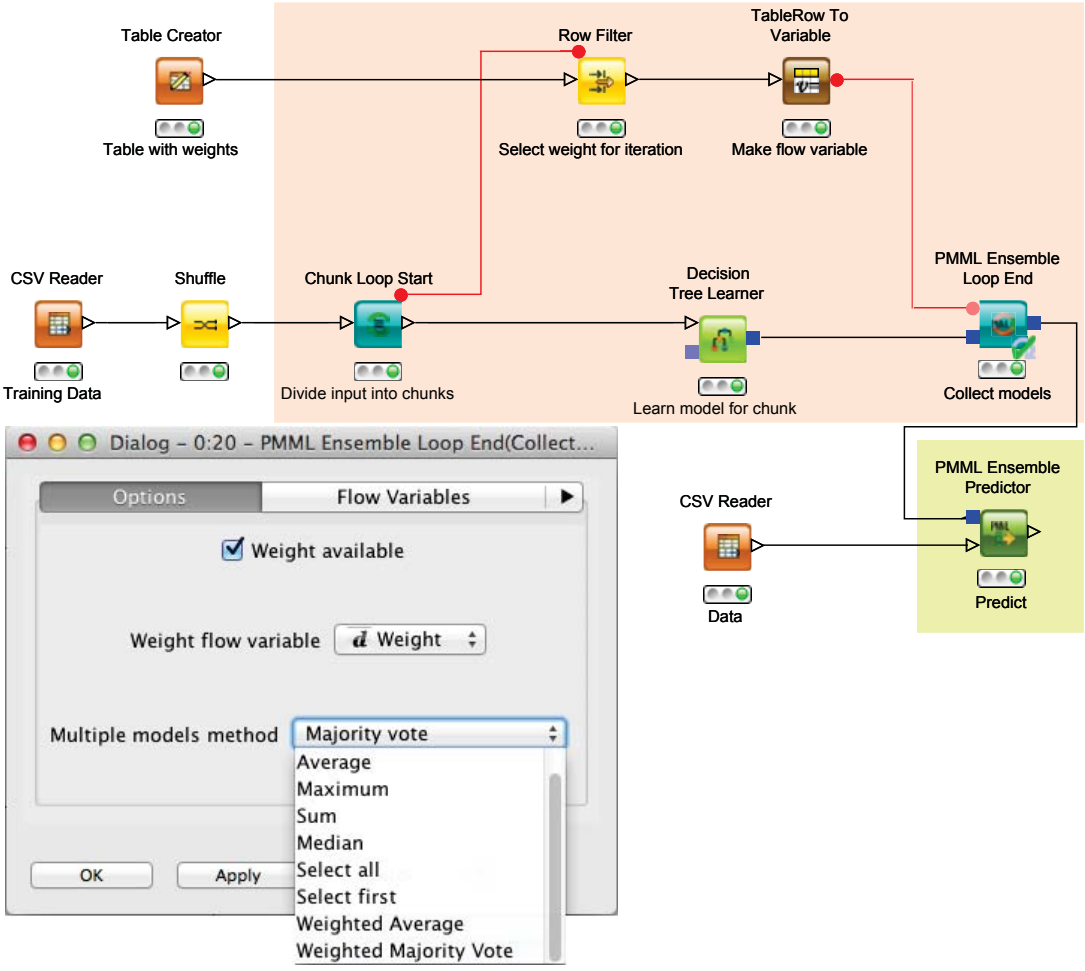

Figure 7: Bagging using flow variables to assign weights to individual models in the ensemble. If no flow variable is selected, the weight is set to 1 for all models.

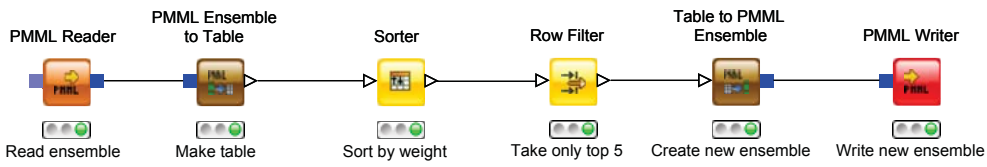

Figure 8: Loading an ensemble model from a file, keeping only the top 5 models and writing it back to disk.