PMML: An Open Standard for Sharing Models

by Alex Guazzelli, Michael Zeller, Wen-Ching Lin and Graham Williams

Introduction

The **PMML** package exports a variety of predictive and descriptive models from R to the Predictive Model Markup Language (Data Mining Group, 2008). PMML is an XML-based language and has become the de-facto standard to represent not only predictive and descriptive models, but also data pre- and post-processing. In so doing, it allows for the interchange of models among different tools and environments, mostly avoiding proprietary issues and incompatibilities.

The **PMML** package itself (Williams et al., 2009) was conceived at first as part of Togaware's data mining toolkit Rattle, the R Analytical Tool To Learn Easily (Williams, 2009). Although it can easily be accessed through Rattle's GUI, it has been separated from Rattle so that it can also be accessed directly in R.

In the next section, we describe PMML and its overall structure. This is followed by a description of the functionality supported by the **PMML** package and how this can be used in R. We then discuss the importance of working with a valid PMML file and finish by highlighting some of the debate surrounding the adoption of PMML by the data mining community at large.

A PMML primer

Developed by the Data Mining Group, an independent, vendor-led committee (`http://www.dmg.org`), PMML provides an open standard for representing data mining models. In this way, models can easily be shared between different applications, avoiding proprietary issues and incompatibilities. Not only can PMML represent a wide range of statistical techniques, but it can also be used to represent their input data as well as the data transformations necessary to transform these into meaningful features.

PMML has established itself as the lingua franca for the sharing of predictive analytics solutions between applications. This enables data mining scientists to use different statistical packages, including R, to build, visualize, and execute their solutions.

PMML is an XML-based language. Its current 3.2 version was released in 2007. Version 4.0, the next version, which is to be released in 2009, will expand the list of available techniques covered by the standard. For example, it will offer support for time series, as well as different ways to explain models, including statistical and graphical measures. Version 4.0 also expands on existing PMML elements and their capabilities to further represent the diverse world of predictive analytics.

PMML Structure

PMML follows a very intuitive structure to describe a data mining model. As depicted in Figure 1, it is composed of many elements which encapsulate different functionality as it relates to the input data, model, and outputs.

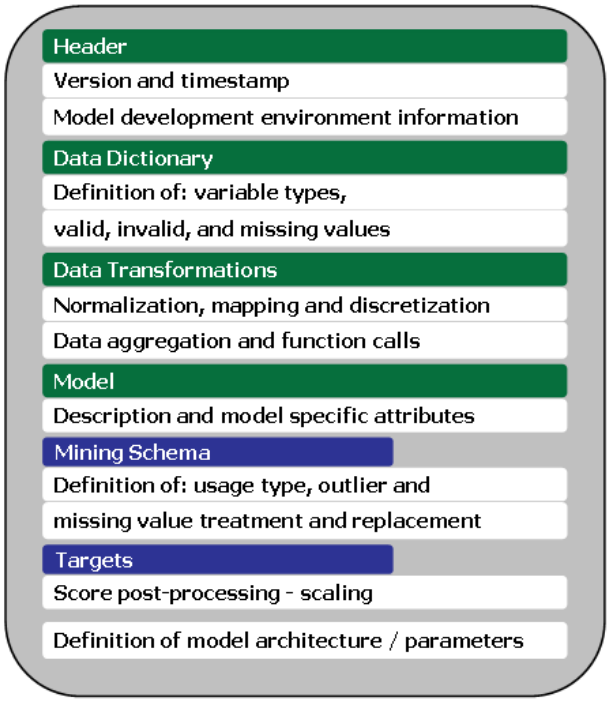

Figure 1: PMML overall structure described sequentially from top to bottom.

Sequentially, PMML can be described by the following elements:

Header

The header element contains general information about the PMML document, such as copyright information for the model, its description, and information about the application used to generate the model such as name and version. It also contains an attribute for a timestamp which can be used to specify the date of model creation.

Data dictionary

The data dictionary element contains definitions for all the possible fields used by the model. It is in the data dictionary that a field is defined as continuous, categorical, or ordinal. Depending on this definition, the appropriate value ranges are then defined as well as the data type (such as string or double). The data dictionary is also used for describing the list of valid, invalid, and missing values relating to every single input data.

Data transformations

Transformations allow for the mapping of user data into a more desirable form to be used by the mining model. PMML defines several kinds of data transformation:

- *Normalization:* Maps values to numbers, the input can be continuous or discrete.
- *Discretization:* Maps continuous values to discrete values.
- *Value mapping:* Maps discrete values to discrete values.
- *Functions:* Derive a value by applying a function to one or more parameters.
- *Aggregation:* Summarizes or collects groups of values.

The ability to represent data transformations (as well as outlier and missing value treatment methods) in conjunction with the parameters that define the models themselves is a key concept of PMML.

Model

The model element contains the definition of the data mining model. Models usually have a model name, function name (classification or regression) and technique-specific attributes.

The model representation begins with a mining schema and then continues with the actual representation of the model:

- *Mining Schema:* The mining schema (which is embedded in the model element) lists all fields used in the model. This can be a subset of the fields defined in the data dictionary. It contains specific information about each field, such as name and usage type. Usage type defines the way a field is to be used in the model. Typical values are: active, predicted, and supplementary. Predicted fields are those whose values are predicted by the model. It is also in the mining schema that special values are treated. These involve:
  - *Outlier Treatment:* Defines the outlier treatment to be used. In PMML, outliers can be treated as missing values, as extreme values (based on the definition of high and low values for a particular field), or as is.
  - *Missing Value Replacement Policy:* If this attribute is specified then a missing value is automatically replaced by the given values.
  - *Missing Value Treatment:* Indicates how the missing value replacement was derived (e.g. as value, mean or median).
- *Targets:* The targets element allows for the scaling of predicted variables.
- *Model Specifics:* Once we have the data schema in place we can specify the details of the actual model.

  A multi-layered feed-forward neural network, for example, is represented in PMML by its activation function and number of layers, followed by the description of all the network components, such as connectivity and connection weights.

  A decision tree is represented in PMML, recursively, by the nodes in the tree. For each node, a test on one variable is recorded and the sub-nodes then correspond to the result of the test. Each sub-node contains another test, or the final decision.

  Besides neural networks and decision trees, PMML allows for the representation of many other data mining models, including linear regression, generalised linear models, random forests and other ensembles, association rules, cluster models, näive Bayes models, support vector machines, and more.

PMML also offers many other elements such as built-in functions, statistics, model composition and verification, etc.

Exporting PMML from R

The **PMML** package in R provides a generic `pmml` function to generate PMML 3.2 for an object. Using an S3 generic function, the appropriate method for the class of the supplied object is dispatched.

An example

A simple example illustrates the steps involved in generating PMML. We will build a decision tree, using **rpart**, to illustrate. With a standard installation of R with the **PMML** package installed, the following should be easily repeatable.

First, we load the appropriate packages and dataset. We will use the well known Iris dataset that records sepal and petal characteristics for different varieties of iris.

```
> library(pmml)
> library(rpart)
> data(iris)
```

We can build a simple decision tree to classify examples into iris varieties. We see the resulting structure of the tree below. The root node test is on the variable *Petal.Length* against a value of 2.45. The "left" branch tests for *Petal.Length* < 2.45 and delivers a decision of *setosa*. The "right" branch splits on a further variable, *Petal.Width*, to distinguish between *versicolor* and *virginica*.

```
> my.rpart <- rpart(Species ~ ., data=iris)
> my.rpart

n= 150

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 150 100 setosa ...
  2) Petal.Length< 2.45 50 0 setosa (0.33 ...
  3) Petal.Length>=2.45 100 50 versicolor ...
    6) Petal.Width< 1.75 54 5 versicolor ...
    7) Petal.Width>=1.75 46 1 virginica ...
```

To convert this model to PMML we simply call the `pmml` function. Below we see some of the output.

The exported PMML begins with a header that contains information about the model. As indicated above, this will contain general information about the model, including copyright, the application used to build the model, and a timestamp. Other information might also be included.

```
> pmml(my.rpart)

<PMML version="3.2" ...
 <Header
  copyright="Copyright (c) 2009 Togaware"
  description="RPart Decision Tree">
  <Extension name="timestamp"
    value="2009-02-15 06:51:50"
    extender="Rattle"/>
  <Extension name="description"
    value="iris tree"
    extender="Rattle"/>
  <Application name="Rattle/PMML"
    version="1.2.7"/>
 </Header>
```

Next, the data dictionary records information about the data fields from which the model was built. Here we see the definition of a categoric and a numeric data field.

```
 <DataDictionary numberOfFields="5">
  <DataField name="Species" ...
   <Value value="setosa"/>
   <Value value="versicolor"/>
   <Value value="virginica"/>
  <DataField name="Sepal.Length"
    optype="continuous"
    dataType="double"/>
  </DataField>
 ...
```

The header and the data dictionary are common to all PMML, irrespective of the model.

Next we record the details of the model itself. In this case we have a tree model, and so a 'TreeModel' element is defined. Within the tree model we begin with the mining schema.

```
 <TreeModel modelName="RPart_Model"
            functionName="classification"
            algorithmName="rpart"
            ...>
  <MiningSchema>
   <MiningField name="Species"
                usageType="predicted"/>
   <MiningField name="Sepal.Length"
                usageType="active"/>
 ...
```

This is followed by the actual nodes of the tree. We can see that the first test involves a test on the *Petal.Length*. Once again, it is testing the value against 2.45.

```
  <Node id="1" score="setosa"
        recordCount="150" defaultChild="3">
   <True/>
   <ScoreDistribution value="setosa"
     recordCount="50" confidence="0.33"/>
   <ScoreDistribution value="versicolor"
     recordCount="50" confidence="0.33"/>
   <ScoreDistribution value="virginica"
     recordCount="50" confidence="0.33"/>
   <Node id="2" score="setosa"
         recordCount="50">
    <CompoundPredicate
       booleanOperator="surrogate">
     <SimplePredicate field="Petal.Length"
         operator="lessThan" value="2.45"/>
...
  </Node>
 </TreeModel>
</PMML>
```

We also note that more information is captured here than displayed with R's print method for the **rpart** object. The **rpart** object in fact includes information about surrogate splits which is also captured in the PMML representation.

Usually, we will want to save the PMML to a file, and this can easily be done using the `saveXML` function from the **XML** package (Temple Lang , 2009).

```
> saveXML(pmml(my.rpart),
+         file="my_rpart.xml")
```

Supported Models

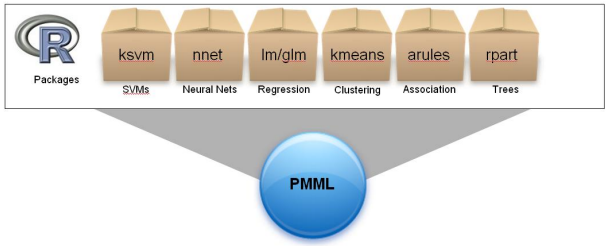

Figure 2: PMML Export functionality is available for several predictive algorithms in R.

Although coverage is constantly being expanded, PMML exporter functionality is currently available for the following data mining algorithms:

1. Support Vector Machines — **kernlab** (Karatzoglou et al., 2008): PMML export for SVMs implementing multi-class and binary classification as well as regression for objects of class `ksvm` (package **kernlab**). It also exports transformations for the input variables by following the scaling scheme used by **ksvm** for numerical variables, as well as transformations to create dummy variables for categorical variables.

   For multi-class classification, **ksvm** uses the *one-against-one* approach, in which the appropriate class is found by a voting scheme. Given that PMML 3.2 does not support this approach, the export functionality comes with an extension piece to handle such cases. The next version of PMML to be released in early 2009 will be equipped to handle different classification methods for SVMs, including one-against-one.

   The example below shows how to train a support vector machine to perform binary classification using the audit dataset — see Williams (2009).

   ```
   > library(kernlab)
   > audit <- read.csv(file(
   +"http://rattle.togaware.com/audit.csv")
   + )
   > myksvm <- ksvm(as.factor(Adjusted) ~ .,
   +             data=audit[,c(2:10,13)],
   +             kernel = "rbfdot",
   +             prob.model=TRUE)
   > pmml(myksvm, data=audit)
   ```

   Note that the **PMML** package is being invoked in the example above with two parameters: 1) the `ksvm` object which contains the model representation; and 2) the data object used to train the model. The **PMML** package uses the data object to be able to retrieve the values for categorical variables which are used internally by `ksvm` to create dummy variables, but are not part of the resulting `ksvm` object.

2. Neural Networks — **nnet**: PMML export for neural networks implementing multi-class or binary classification as well as regression models built with the **nnet** package, available through the **VR** bundle (Venables and Ripley, 2002). Some details that are worth mentioning are:

   - Scaling of input variables: Since **nnet** does not automatically implement scaling of numerical inputs, it needs to be added to the generated PMML file by hand if one is planning to use the model to compute scores/results from raw, unscaled data.
   - The PMML exporter uses transformations to create dummy variables for categorical inputs. These are expressed in the 'NeuralInputs' element of the resulting PMML file.
   - PMML 3.2 does not support the censored variant of softmax.
   - Given that **nnet** uses a single output node to represent binary classification, the resulting PMML file contains a discretizer with a threshold set to 0.5.

3. Classification and Regression Trees — **rpart** (Therneau and Atkinson. R port by B. Ripley, 2008): PMML export functionality for decision trees built with the **rpart** package is able to export classification as well as regression trees. It also exports surrogate predicate information and missing value strategy (default child strategy).

4. Regression Models — `lm` and `glm` from **stats**: PMML export for linear regression models for objects of class `"lm"` and binary logistic regression models for objects of class `"glm"` built with the binomial family. Note that this function currently does not support multinomial logistic regression models or any other regression models built using the **VGAM** package.

5. Clustering Models — `hclust` and `kmeans` from **stats**: PMML export functionality for clustering models for objects of class `"hclust"` and `"kmeans"`.

6. Association Rules — **arules** (Hahsler et al., 2008): PMML export for association rules built with the **arules** package.

7. Random Forest (and randomSurvivalForest) — **randomForest** (Breiman and Cutler. R port by A. Liaw and M. Wiener, 2009) and **randomSurvivalForest** (Ishwaran and Kogalur , 2009): PMML export of a randomSurvivalForest `"rsf"` object. This function gives the user the ability to export PMML containing the geometry of a forest.

PMML validation

Since PMML is an XML-based standard, the specification comes in the form of an XML Schema. Zementis (`http://www.zementis.com`) has built a tool that can be used to validate any PMML file against the current and previous PMML schemas. The tool can also be used to convert older versions of PMML (2.1, 3.0, 3.1) to version 3.2 (the current version). The PMML Converter will validate any PMML file and is currently able to convert the following modeling elements:

1. Association Rules
2. Neural Networks
3. Decision Trees
4. Regression Models
5. Support Vector Machines
6. Cluster Models

The PMML converter is free to use and can be accessed directly from the Zementis website or installed as a gadget in an iGoogle console.

It is very important to validate a PMML file. Many vendors claim to support PMML but such support is often not very complete. Also, exported files do not always conform to the PMML specification. Therefore, interoperability between tools can be a challenge at times.

More importantly, strict adherence to the schema is necessary for the standard to flourish. If this is not enforced, PMML becomes more of a blueprint than a standard, which then defeats its purpose as an open standard supporting interoperability. By validating a PMML file against the schema, one can make sure that it can be moved around successfully. The PMML Converter is able to pinpoint schema violations, empowering users and giving commercial tools, as well as the **PMML** package available for R, an easy way to validate their export functionality. The example below shows part of the data dictionary element for the iris classification tree discussed previously. This time, however, the PMML is missing the required dataType attribute for variable *Sepal.Length*. The PMML Converter flags the problem by embedding an XML comment into the PMML file itself.

```
<DataDictionary>
  <!--PMML Validation Error:
  Expected attribute: dataType in
  element DataField -->
  <DataField name="Sepal.Length"
    optype="continuous">
  </DataField>
  <DataField name="Sepal.Width"
    dataType="double"
    optype="continuous">
</DataField>
...
```

Discussion

Although PMML is an open standard for representing predictive models, the lack of awareness has made its adoption slow. Its usefulness was also limited because until recently predictive models were in general built and deployed using the same data mining tool. But this is now quickly changing with models built in R, for example, now able to be deployed in other model engines, including data warehouses that support PMML.

For those interested in joining an on-going discussion on PMML, we have created a PMML discussion group under the AnalyticBridge community (`http://www.analyticbridge.com/group/pmml`). In one of the discussion forums, the issue of the lack of support for the export of data transformations into PMML is discussed. Unless the model is built directly from raw data (usually not the case), the PMML file that most tools will export is incomplete. This also extends to R. As we saw in the previous section, the extent to which data transformations are exported to PMML in R depends on the amount of pre-processing carried out by the package/class used to build the model. However, if any data massaging is done previous to model building, this needs to be manually converted to PMML if one wants that to be part of the resulting file. PMML offers coverage for many commonly used data transformations, including mapping and normalization as well as several built-in functions for string, date and time manipulation. Built-in functions also offer several mathematical operations as depicted in the PMML example below, which implements: *maximum(round(inputVar/1.3))*.

```
<Apply function="max">
  <Apply function="round">
    <Apply function="/">
      <FieldRef field="inputVar"/>
      <Constant>1.3</Constant>
    </Apply>
  </Apply>
</Apply>
```

Integrated tools that export PMML should allow not only for the exporting of models, but also data transformations. One question we pose in our discussion forum is how to implement this in R.

More recently, KNIME has implemented extensive support for PMML. KNIME is an open-source framework that allows users to visually create data flows (http://www.knime.org). It implements plugins that allow R-scripts to be run inside its Eclipse-based interface. KNIME can import and export PMML for some of its predictive modules. In doing so, it allows for models built in R to be loaded in KNIME for data flow visualization. Given that it exports PMML as well, it could potentially be used to integrate data pre- and post-processing into a single PMML file which would then contain the entire data and model processing steps.

PMML offers a way for models to be exported out of R and deployed in a production environment, quickly and effectively. Along these lines, an interesting development we are involved in is the ADAPA scoring engine (Guazzelli et al., 2009), produced by Zementis. This PMML consumer is able to upload PMML models over the Internet and then execute/score them with any size dataset in batch-mode or real-time. This is implemented as a service through the Amazon Compute Cloud. Thus, models developed in R can be put to work in matter of minutes, via PMML, and accessed in real-time through web-service calls from anywhere in the world while leveraging a highly scalable and cost-effective cloud computing infrastructure.

Bibliography

L. Breiman and A. Cutler. R port by A. Liaw and M. Wiener. *randomForest: Breiman and Cutler's Random Forests for Classification and Regression*, 2009. URL http://cran.r-project.org/package=randomForest. R package version 4.5-30.

Data Mining Group. PMML version 3.2. WWW, 2008. URL http://www.dmg.org/pmml-v3-2.html.

A. Guazzelli, K. Stathatos, and M. Zeller. Efficient deployment of predictive analytics through open standards and cloud computing. *To appear in ACM SIGKDD Explorations*, June 2009. URL http://www.sigkdd.org/explorations.

M. Hahsler, C. Buchta, B. Gruen, and K. Hornik. *arules: Mining Association Rules and Frequent Itemsets*, 2008. URL http://cran.r-project.org/package=arules. R package version 0.6-8.

H. Ishwaran and U. Kogalur. *randomSurvivalForest: Ishwaran and Kogalur's Random Surival Forest*, 2009. URL http://cran.r-project.org/package=randomSurvivalForest. R package version 3.5.1.

A. Karatzoglou, A. Smola, and K. Hornik. *The kernlab package*, 2008. URL http://cran.R-project.org/package=kernlab. R package version 0.9-8.

D. Temple Lang. *XML: Tools for parsing and generating XML within R and S-Plus*, 2009. URL http://cran.R-project.org/package=xml. R package version 2.3-0.

T. M. Therneau and B. Atkinson. R port by B. Ripley. *rpart: Recursive Partitioning*, 2008. URL http://cran.r-project.org/package=rpart. R package version 3.1-42.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Statistics and Computing. Springer, New York, 4th edition, 2002. URL http://cran.r-project.org/package=VR. R package version 7.2-45.

G. Williams. *rattle: A graphical user interface for data mining in R*, 2009. URL http://cran.r-project.org/package=rattle. R package version 2.4.31.

G. Williams, M. Hahsler, A. Guazzelli, M. Zeller, W. Lin, H. Ishwaran, U. B. Kogalur, and R. Guha. *pmml: Generate PMML for various models*, 2009. URL http://cran.r-project.org/package=pmml. R package version 1.2.7.

Alex Guazzelli, Michael Zeller, Wen-Ching Lin
Zementis Inc
info@zementis.com

Graham Williams
Togaware Pty Ltd
Graham.Williams@togaware.com