

A Simple Strategy for Composing Data Mining Operations

Robert L. Grossman and Gregor Meyer

Abstract

An important element in data preparation is composing data mining operations. In this note, we discuss some of the issues involved when composing data mining operations. We also describe the support in PMML Version 3.0 for two of the most common type of compositions: using the output of one model as the input to another model (model sequencing) and using one model to select one or more other models (model selection or averaging).

1. Introduction and Background

It is now standard in data mining to view the data mining process as consisting of several steps, some of the most important of which are: data preparation, data modeling, and scoring or deployment.

Today, there are well defined architectures and standards for scoring. In particular, the Predictive Model Markup Language or PMML [DMG] provides a clean interface between producers of models, such as a statistical or data mining system, and consumers of models, such as a scoring system or application that employs embedded analytics.

On the other hand, the architectures, processes and standards are much less mature for data preparation. Although it can be much more complicated, it is sometimes helpful to think of data preparation as the process that takes one or more tables of data and produces a single table of feature vectors, which are an input to a statistical or data mining model. See Figure 1.

For example, PMML Version 2.1 includes the following four common types of data preparation operations:

- Normalization: Normalization transforms continuous or discrete values or ranges to numbers.
- Discretization: Discretization transforms continuous values to discrete values.
- Value mapping: Value mapping transforms discrete values to discrete values.
- Aggregation: Aggregation summarizes or collects groups of values, for example by computing averages.

It is an interesting exercise to try to see in how many practical cases the data preparation can be reduced to the compositions of the four operations above. It turns out to be quite large.

More generally, one could include certain built-in functions in data preparation or user-defined functions:

- Functions: Data preparation functions derive a value by applying a function to one or more parameters.

Notice that each of the itemized operations above can be thought of as a function. The question arises:

1. What is an appropriate architecture and what are appropriate standards so that the functions that arise in data preparation can be composed?
2. More generally, what is an appropriate architecture and what are appropriate standards so that the data mining models can be composed?

Here are two simple, motivating examples, which are quite common in practice, related to the second question of how data mining models can be composed:

Example 1. A classification tree may be used to select two or more logistic regression models. In this paper, we refer to this as model selection.

Example 2. A logistic regression model may be used as the input to a classification tree. In this paper, we refer to this as model sequencing.

Note that composing models is difficult for several reasons:

- The composition of data mining operations is only partially defined. In general, it is not well defined to take the output of one model and use it as the input to another model.
- Model composition covers several different use cases in practice.
- Just as a clean separation between model producers and model consumers led to the development of a wide variety of scoring engines and embedded data mining applications, the hope is that a simple mechanism for defining composition can lead to standard architectures for data preparation. The challenge is to balance the generality of defining a very general composition with the complexity required of model consumers with constructs sufficiently powerful to implement common use cases.

In this article, we introduce the composition mechanism developed by the PMML Working Group, which covers the composition of normalization, discretization, value mapping, and aggregation, as well as the two motivating examples.

In Section 2, we describe some of the infrastructure of PMML. Section 3 outlines the basic idea. In Section 4, we describe how model selection is done in PMML. In Section 5, we describe how model sequencing is done in PMML. Section 6 indicates the current status of PMML and Section 7 contains a summary and conclusion.

Acknowledgements. The work described in this paper was done by the Data Mining Group's (DMG) Predictive Model Markup Language (PMML) Working Group. A more complete description is available at www.dmg.org.

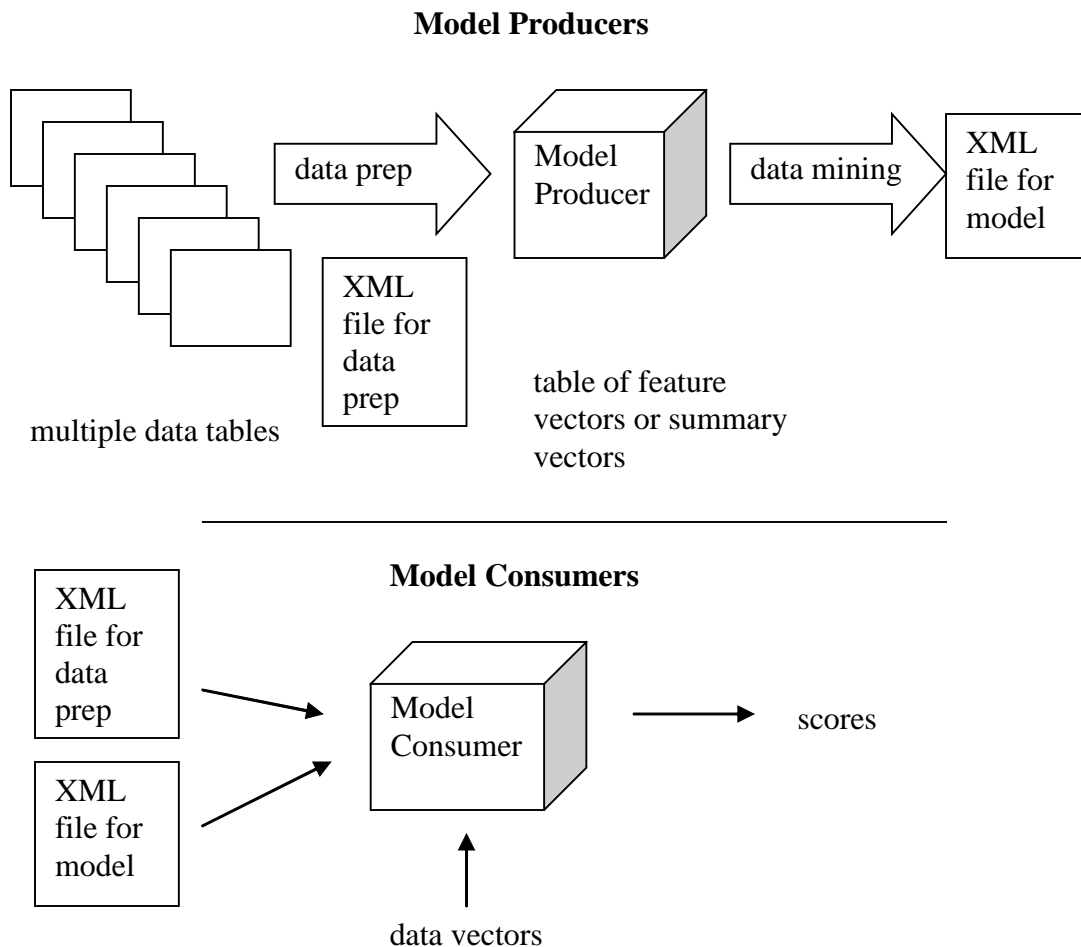


Figure 1. How XML descriptions of data preparation and models can be used to help create standard architectures for model producers and model consumers.

2. Data Attributes, Mining Attributes, and Derived Attributes

In this section, we briefly review how data attributes, mining attributes and derived attributes are used in PMML.

Features vectors, which are the inputs to models, are defined in PMML in the following way.

1. A data dictionary defines a set of data attributes.
2. A mining schema can identify one or more data attributes as mining attributes. A mining attribute also includes additional information; for example, such as how a data attribute is

used by a model (as an independent attribute, a predicted attribute, or excluded). Some mining elements are directly used as inputs to a model.

3. Other mining attributes can be used as inputs to the Transformation Dictionary to define derived attributes to the model.

See Figure 2.

In PMML Version 2.1, the Transformation Dictionary contains the required specifications for normalization, discretization, value mapping, and aggregation functions. See Example 1 for a simple example of a discretization.

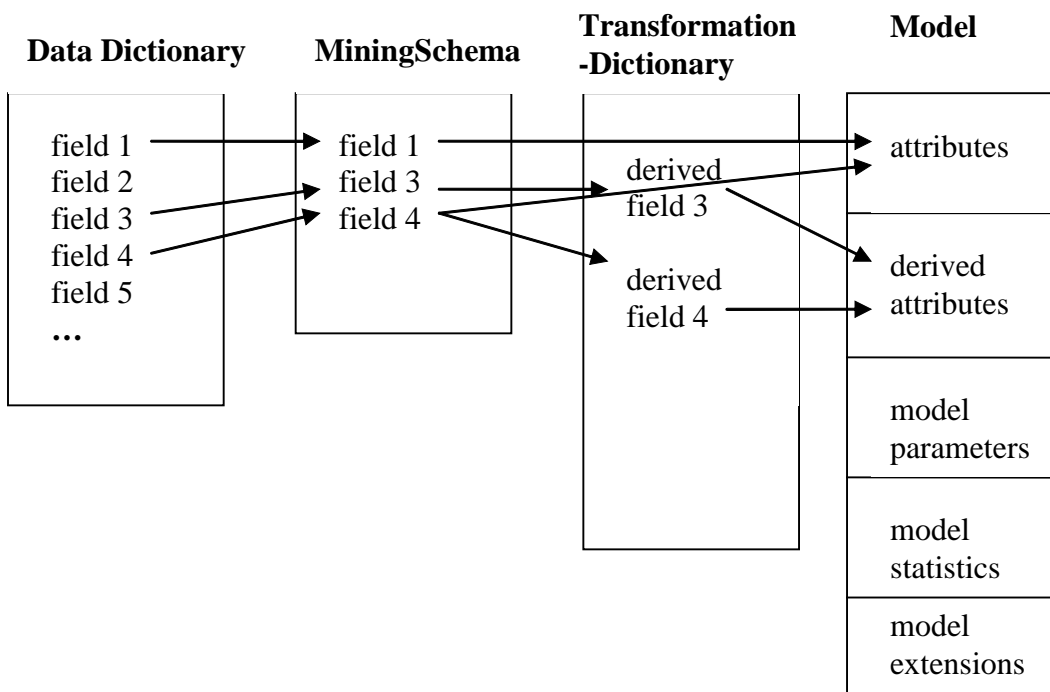


Figure 2. This figure illustrates how four inputs to a model can be defined: two of the attributes are taken directly from the MiningSchema and two of the attributes are derived and defined using the Transformation Dictionary.

3. Basic Ideas

There are three basic ideas at the core of how PMML Version 3.0 supports composition.

1. The first idea is that models can be embedded in other models. In this case, much of the model infrastructure of the embedded model (e.g., mining attributes, transformation dictionary, etc.) is not required. For this reason, PMML Version 3.0 supports a simplified version of a model called an embedded model. See Table 1. For example, a

stand-alone tree model is called `TreeModel`, an embedded tree model is called `DecisionTree`.

2. The second idea is that model sequencing can be supported by using essentially the same approach as used to defined `DerivedFields` with a `TransformationDictionary`. More specifically, an embedded model can be used to define what is called a `ResultField`, which can be used as the input to a model in essentially the same way as a `DerivedField`.
3. The third idea is to provide a container for a collection of models called `MiningModel`. Model selection can then be supported by using a `DecisionTree` within a `MiningModel`. Voting in ensembles can be supported by simply using a `RegressionModel` within a `MiningModel`.

We note that the philosophy in PMML has been to employ the simplest mechanism that will support the desired outcome. For example, models are explicitly defined by specifying parameters, instead of the supporting arbitrary code to define models. In the same way, the three ideas above are powerful enough to support a wide variety of different types of model compositions that occur in practice. We do not need to define a more general mechanism for composing data mining operations.

The reason for this philosophy is that it simplifies the design of PMML Consumers.

Stand alone model	Model used in selection or sequencing	Main content
<code>RegressionModel</code>	<code>Regression</code>	<code>RegressionTable</code>
<code>TreeModel</code>	<code>DecisionTree</code>	<code>Nodes</code>
...

Table 1. This table describes how the same model may be used in a stand-alone fashion or as an embedded model.

4. Model Selection

The basic idea used for model selection in PMML is to use the embedded version of the model or models and then use a `DecisionTree` to select the appropriate model.

Model selection in PMML Version 3.0 uses the `MiningModel` with a regression function as a container and a decision tree as the selection logic. With this approach, model selection is sufficiently powerful to support some common types of ensemble operations, such as voting or averaging.

See Example 2.

5. Model Sequencing

As mentioned above, a motivating example for model sequences is using an attribute defined by a regression model as an attribute of a decision tree.

The basic idea used in PMML to describe a sequence of models is simple. As Example 2 illustrates, PMML Version 3.0 defines a new element called `Regression`, which contains enough information to specify a new attribute, which PMML calls a `ResultField`, and to define this result field using a regression model. The parameters for the regression model are specified using an element called `RegressionTable`. Once a `ResultField` is defined in this way, it can be used as an input to the surrounding model, which in this example is a `TreeModel`. In other words, the `ResultField` called “term” in this example can be used as an input to any node in a `TreeModel`.

See Example 3.

6. Status

PMML Version 3.0 has not yet been released. The approach used for model selection and model sequencing is still subject to change.

7. Conclusion

In this article, we have described why the composition of data transformations and common statistical and data mining models is important for data preparation. We have introduced the approach used in PMML Version 3.0 for selecting models, that is, for choosing one or models from a container of models. We have also introduced the approach used in PMML Version 3.0 for using the output of one model as the input to another model or what is called model sequencing.

8. References

[DMG] Data Mining Group, www.dmg.org

```
<Discretize field="Profit">
  <DiscretizeBin binValue="negative">
    <Interval closure="openOpen" rightMargin="0" />
    <!-- left margin is -infinity by default -->
  </DiscretizeBin>
  <DiscretizeBin binValue="positive">
    <Interval closure="closedOpen" leftMargin="0" />
    <!-- right margin is +infinity by default -->
  </DiscretizeBin>
</Discretize>
```

Example 1. This example shows how PMML Version 2.1 defines a data preparation operation for discretization.

```

<PMML>
...
<MiningModel function="regression">
<MiningSchema>
  as usual
</MiningSchema>

... derived fields as usual ...

<DecisionTree> <!-- wrapper for content as in TreeModel -->
  <Node><True/> <!-- root node in a DecisionTree is always True -->
  <Node> <!-- 1st sub-tree, is a leaf node -->
    <Predicate age<=50 .../>
    <Regression> <!-- embedded regression equation -->
      <RegressionTable intercept="2.34">
        ... predictors: 0.03*income + 1.23*age ...
      </RegressionTable>
    </Regression>
  </Node>
  <Node> <!-- 2nd sub-tree, is a leaf node -->
    <Predicate age>50 .../>
    <Regression> <!-- embedded regression equation -->
      <RegressionTable intercept="2.22">
        ... predictors: 0.01*income -0.11*age*mc ...
      </RegressionTable>
    </Regression>
  </Node>
  </Node> <!-- end of root node -->
</DecisionTree>

</MiningModel>
</PMML>

```

DecisionTree contains nodes that can contain embedded models

regression model

regression model

Example 2. This example illustrates how PMML Version 3.0 supports model selections. In this case, two or more embedded regression models are selected using a decision tree.

```

<PMML>
...
<TreeModel function='regression'>
  <MiningSchema>
    <!-- declare fields "age", "income", "married", ... as usual -->
    <MiningSchema>

    <!-- encode a categorical input field using a PMML 2.1 transformation -->
    <DerivedField name="mc" optype="continuous">
      <MapValues ... >

      <!-- map "yes" to 1.0
      map "no" to -1.0 -->

      </MapValues>
    </DerivedField>

    <!-- derive a new input term -->
    <!-- use an embedded regression model as a transformation -->
    <Regression>
      <ResultField name="term" feature="predicted">
        <!-- The ResultField selects the predicted value from
        this regression element and binds
        it to the name "term" -->

        <!-- RegressionTable as defined in RegressionModel -->
        <RegressionTable>
          <!--with intercept="2.34"
          and predictors: 0.03*income + 1.23*age*mc -->
          <RegressionTable>
        </RegressionTable>
      </ResultField>
    </Regression>

    <Node> <!-- this is the root node of the TreeModel -->
    <!-- A decision tree as usual,
    it can refer to fields "age", "income", "married"
    as well as to the derived attribute "mc"
    and regression attribute "term" -->
    ...

    </Node>
  </TreeModel>
</PMML>

```

This code defines a derived field called "mc".

This code defines a result field called "term" using a regression model.

This code defines a node that can use the derived field "mc" and the result field "term".

Example 3. This example illustrates how a TreeModel can use attributes that are defined using a regression model.