Optimization of Web Mining Applications through PMML

IOAN POP
Department of Computer Science
Lucian Blaga University
5-7 Ioan Ratiu Str, Sibiu
ROMANIA

IOSIF MIRCEA NEAMȚU
Department of Computer Science
Lucian Blaga University
5-7 Ioan Ratiu Str, Sibiu
ROMANIA

Abstract: In this paper we deal with some ways to use the data mining tools in Web Mining applications. It is well-known that Predictive Model Markup Language (PMML) language offers good capabilities to create applications for predictive models achievement. PMML gives applications a tool to define statistical and data mining models and to share them with other PMML-compliant apps. Our examples of integration and optimization of application are developed with the PMML standard. These examples are the meta-models in order to assure the processing of the information through pre-processing PMML language by making shared models. In our example we deal with PMML models for Clustering like center-based cluster models.

Key-Words: Data Mining, Web Mining, PMML Language, Web Applications

1 Introduction

Daily, multiples transactions of data are accumulated in data bases, but in same time a growing quantity of data is stored in the warehouse Web. This reality is assessing the development of new technologies. The Web technology XML allows developers of programs to create their own personalized tags, which supply the possibility of transmission, validation and interpretation of the data between applications and organizations. [7]

PMML (acronym for Predictive Model Markup Language) is an XML-based language which provides a tool to define statistical and data mining models and to share models between PMML compliant applications. Predictive modelling is the process by which a model is created or chosen to try to best predict the probability of an outcome. In many cases the model is chosen on the basis of detection theory to try to guess the probability of a signal given a set amount of input data. PMML provides applications a vendor-independent method of defining models so that proprietary issues and incompatibilities are no longer a barrier to the exchange of models between applications [8].

PMML describes the inputs to data mining models, the transformations used prior to prepare data for data mining, and the parameters which define the models themselves. It allows users to develop models within one vendor's application, and use other vendors' applications to visualize, analyze, evaluate or otherwise use the models. The PMML language is a formal specification of the associated gramar of a defined language in XML Schema.[1]. PMML describes the inputs for the data mining models, the previous transformations used to the data preparation for the data mining processings, as well as the parameters that define the models itself. [2]-[5] This language is used for a large variety of the applications, including the applications from finance, e-business, direct marketing, production, and defence. The achieved products by many developers with PMML include both active componets of the group – Data Mining Group – that develops the standard and another elements from the format of the XML interchange for the statistical and data mining models. [8]

The design naturally supports functional separation of analysis and application environments while maintaining logical integrity. Models can be created in several analytical applications, based on the same data that is available to the application environment. These models are stored as PMML in one or more database tables. Any application that accesses data can call a single database function to apply any chosen model on this data. A typical example of such a design is shown in Fig.1. [6]

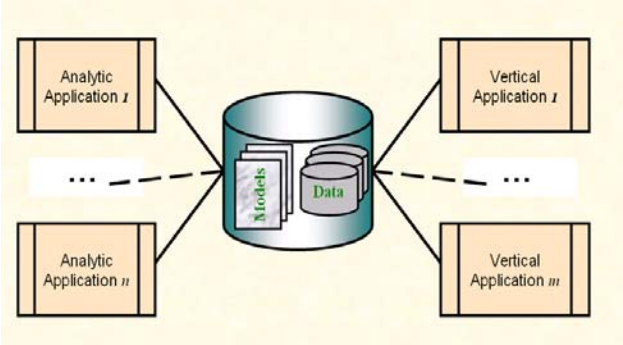

Figure 1: Model Building and Application Environments

2 General Structure of a PMML Document

The PMML standard uses XML to represent mining models. The structure of the models is described by an XML Schema. One or more mining models can be contained in a PMML document. The general structure of a PMML document is showed in Fig. 2.

```
<?xml version="1.0"?>
 <PMML version="3.1"
   xmlns="http://www.dmg.org/PMML-3_1"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
      <DataDictionary> ... </DataDictionary>
   ... a model ...
 </PMML>
```

Figure 2. The general structure of a PMML document.

Note that because of the namespace declaration in its current form, PMML cannot be mixed with content of a different namespace.

A PMML document must be valid with respect to the PMML XSD. XSD is the acronym for **X**ML **S**chema **D**efinition, a way to describe and validate data in an XML environment. A document must not require a validating parser, which would load external entities. The root element of a PMML document must have type PMML (see Fig. 3). [8]

```
<xs:element name="PMML">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Header"/>
      <xs:element ref="MiningBuildTask" minOccurs="0"/>
      <xs:element ref="DataDictionary"/>
      <xs:element ref="TransformationDictionary" minOccurs="0"/>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
       <xs:choice>
          <xs:element ref="AssociationModel"/>
          <xs:element ref="ClusteringModel"/>
           ... other element ...
        </xs:choice>
      </xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="version" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

Figure 3. The root element of a PMML document.

A PMML document can contain more than one model. If the application system provides a means of selecting models by name and if the PMML consumer specifies a model name, then that model is used; otherwise the first model is used. A PMML compliant system is not required to provide model selection by name. The list of mining models in a PMML document may even be empty. The document can be used to carry the initial metadata before an actual model is computed. A PMML document containing no model is not meant to be useful for a PMML consumer.

The element **MiningBuildTask** can contain any XML value describing the configuration of the training run that produced the model instance. The fields in the DataDictionary and in the TransformationDictionary together are identified by unique names. Other elements in the models can refer to these fields by name. Multiple models on one PMML document can share the same fields in the TransformationDictionary. Nevertheless, a model can also define its 'own' derived fields in the element Local Transformations. Furthermore, various models use DerivedField elements directly in the definition of the model. For example, DerivedFields appear inline in the input layer of neural networks.

Certain types of PMML models such as neural networks or logistic regression can be used for different purposes. That is, some instances implement prediction of numeric values, while others can be used for classification. Therefore, PMML defines five different mining functions.

```
<xs:simpleType name="MINING-FUNCTION">
  <xs:restriction base="xs:string">
    <xs:enumeration value="associationRules"/>
    <xs:enumeration value="sequences"/>
    <xs:enumeration value="classification"/>
    <xs:enumeration value="regression"/>
    <xs:enumeration value="clustering"/>
  </xs:restriction>
</xs:simpleType>
```

Figure 4. The **Mining** functions from a PMML document.

Each model has an attribute **functionName** which specifies the mining function.

```
<xs:element name="ExampleModel">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="MiningSchema"/>
      <xs:element ref="Output" minOccurs="0"/>
      <xs:element ref="ModelStats" minOccurs="0"/>
      <xs:element ref="Targets" minOccurs="0"/>
      <xs:element ref="LocalTransformations" minOccurs="0" />
      ...
      <xs:element ref="ModelVerification" minOccurs="0"/>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="modelName" type="xs:string" use="optional"/>
    <xs:attribute name="functionName" type="MINING-FUNCTION" use="required"/>
    <xs:attribute name="algorithmName" type="xs:string" use="optional"/>
  </xs:complexType>
</xs:element>
```

Figure 5. The **Model** element from a PMML document.

The non-empty list of mining fields, define a mining schema. The output element gives a list of result values and internal results such as confidences or probabilities that can be computed by the model. The univariate statistics contain global statistics on (a subset of the) mining fields. The targets section holds more information on the target values and accompanying information like prior probabilities, optypes and the like. LocalTransformations holds derived fields that are local to the model. Other model specific elements follow after that, in the content of ExampleModel. Finally, the ModelVerification part gives sample data and results of the model so consumers can instantly validate. **modelName**: the value in modelName identifies the model with a unique name in the context of the PMML file. This attribute is not required. Consumers of PMML models are free to manage the names of the models at their

discretion. **functionName** and **algorithmName** describe the kind of mining model, e.g., whether it is intended to be used for clustering or for classification. The algorithm name is free-type and can be any description for the specific algorithm that produced the model. This attribute is for information only.

3 Naming Conventions and Extension Mechanism in PMML

The naming conventions for PMML are: **ElementNames** are in mixed case, first uppercase; **attributeNames** are in mixed case, first lowercase; **constants** in enumerations are in mixed case, first lowercase; **SIMPLETYPES** are all uppercase.

The PMML schema contains a mechanism for extending the content of a model. **Extension** elements should be present as the first child in all elements and groups defined in PMML. This way it is possible to place information in the `Extension` elements which affects how the remaining entries are treated. The main element in each model should have `Extension` elements as the first and the last child for maximum flexibility.

```
<xs:element name="Extension">
 <xs:complexType>
  <xs:complexContent mixed="true">
   <xs:restriction base="xs:anyType">
    <xs:sequence>
     <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="extender" type="xs:string" use="optional"/>
    <xs:attribute name="name" type="xs:string" use="optional"/>
    <xs:attribute name="value" type="xs:string" use="optional"/>
   </xs:restriction>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

Figure 6. The `Extension` element from a PMML document.

These extension elements have a content model of `ANY`, where vendor specific extension elements can be included. However, element types must start with **X-**. This convention helps to avoid conflicts with possible future extensions to standard PMML.`Extension` also features the attributes **name** and **value** to specify single extension attributes, where `name` will specify the name of the extension attribute and `value` the respective value. If a document uses local namespaces, then the name of the namespace should not start with `PMML` or `DMG` or any variant of these names with lowercase characters. They are reserved for future use in PMML.

Extension attributes could be added to all elements in PMML if the prefix **x-** was used. This mechanism is deprecated; extension elements should be used instead. PMML documents with extension attributes using the old convention are still considered to be valid PMML. However, note that PMML documents containing old-style `x-` extension attributes will not validate in XML schema, but one can use XSL transformation to remove all `x-` extension attributes and receive an XML document that will validate.

Exemple: An extension attribute `format` and extension element `DataFieldSource` can be added to a `DataField` respectively to a `DataField` in the PCDATA section like this from Fig.7.

```
<DataField name="foo" optype="continuous" >
  <Extension name="format" value="%9.2f"/>
</DataField >
          respectively
<DataField name="foo" optype="continuous">
  <Extension>
    <DataFieldSource sourceKnown="yes">
      <Source>derivedFromInput</Source>
    </DataFieldSource>
  </Extension>
</DataField >
```

Figure7.The **format** extension from a PMML document.

4 Clustering Models

A cluster model basically consists of a set of clusters. For each cluster a center vector can be given. In center-based models a cluster is defined by a vector of center coordinates. Some distance measure is used to determine the nearest center, which is the nearest cluster for a given input record. The center vectors then only approximate the clusters. The model must contain information on the distance or similarity measure used for clustering. It may also contain information on overall data distribution, such as covariance matrix, or other statistics.

```
<xs:element name="ClusteringModel">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="MiningSchema"/>
      <xs:element ref="Output" minOccurs="0" />
      <xs:element ref="ModelStats" minOccurs="0"/>
      <xs:element ref="LocalTransformations" minOccurs="0" />
      <xs:element ref="ComparisonMeasure"/>
      <xs:element ref="ClusteringField" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="CenterFields" minOccurs="0"/>
      <xs:element ref="MissingValueWeights" minOccurs="0"/>
      <xs:element ref="Cluster" maxOccurs="unbounded"/>
      <xs:element ref="ModelVerification" minOccurs="0"/>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="modelName" type="xs:string" use="optional"/>
    <xs:attribute name="functionName" type="MINING-FUNCTION" use="required" />
    <xs:attribute name="algorithmName" type="xs:string" use="optional"/>
    <xs:attribute name="modelClass" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="centerBased"/>
          <xs:enumeration value="distributionBased"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="numberOfClusters" type="INT-NUMBER" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="CenterFields">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="DerivedField" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Figure 8. The Clustering Model element.

Names of coordinates in **CenterFields**,

ClusteringFields and statistics must be consistent with the names of the fields in the data dictionary and in the transformation dictionary. [9]

The attribute **modelClass** specifies whether the clusters are defined by center-vectors or whether they are defined by the statistics. The fields which are used in the center vectors are normalized. In particular this allows the mapping of categorical input fields to numeric values in center vectors. See the DerivedField and the section on normalization. MiningField information (in MiningSchema) must be present for each active variable. For numeric variables it specifies the treatment of outliers. Note that there may be supplementary mining fields. The statistics for these fields are part of the model but they are not required to apply the model. For each active MiningField, an element of type UnivariateStats (see ModelStats) holds information about the overall (background) population. This includes (required) DiscrStats or ContStats, which include possible field values and interval boundaries. Optionally, statistical information is included for the background data.

A cluster is defined by its center vector or by statistics. A center vector is implemented by a NUM-ARRAY. Each Partition corresponds to a cluster and holds field statistics to describe it. The definition of a cluster may contain a center vector as well as statistics. The attribute modelClass in the ClusteringModel defines which one is used to actually define the cluster. [9]

```
<xs:element name="MissingValueWeights">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      <xs:group ref="NUM-ARRAY"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Cluster">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="KohonenMap" minOccurs="0"/>
      <xs:group ref="NUM-ARRAY" minOccurs="0"/>
      <xs:element ref="Partition" minOccurs="0"/>
      <xs:element ref="Covariances" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="optional"/>
    <xs:attribute name="size" type="xs:nonNegativeInteger" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="KohonenMap">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="coord1" type="xs:float" use="optional"/>
    <xs:attribute name="coord2" type="xs:float" use="optional"/>
    <xs:attribute name="coord3" type="xs:float" use="optional"/>
    </xs:complexType>
</xs:element>
```

Figure 9. Three elements: MissingValueWeights, Cluster, and KohonenMap

If a clustering model is center-based then each cluster has a NUM-ARRAY containing the center coordinates for the cluster. The correspondence between input fields and their coordinates is defined by CenterFields, see above. Note that categorical fields can have more than one coordinate, depending on the normalization method.

If some normalization is defined for input fields, then the center coordinates are defined using the normalized values. For numeric fields we could also use the values in the original domain. For categorical values, however, the center vector is not required to contain the indicator values 0.0 or 1.0. There can also be any values between 0.0 and 1.0. These indicate a distribution of categorical values, defining a kind of virtual center point.

MissingValueWeights is used to adjust distance or similarity measures for missing data. See the formulas for ComparisonMeasure below.

The element KohonenMap is appropriate for clustering models that were produced by a Kohonen map algorithm. The attributes coord*i* describe the position of the current cluster in a map with up to three dimensions. This element is not relevant to the scoring function.

If the cluster model contains statistics with mean values, then the center coordinates are not necessarily identical to the corresponding mean values. There may be differences depending on the kind of normalization of input values.

A covariance matrix stores coordinate-by-coordinate variances (diagonal cells) and covariances (non-diagonal cells). [9]

```
<xs:element name="Covariances">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded" />
   <xs:element ref="Matrix"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Figure 10. The Covariances element.

The covariance matrix must be symmetric so only half of the non-diagonal covariance cells need to be stored. Missing covariance cells are reconstructed by symmetry. The sequence of rows/columns corresponds to the sequence in MiningSchema. Note that Covariances does not contain information about internal fields that are defined in the TransformationDictionary. [9]

```
<xs:element name="ClusteringField">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="Comparisons" minOccurs="0"/>
  </xs:sequence>
   <xs:attribute name="field" type="FIELD-NAME" use="required"/>
   <xs:attribute name="fieldWeight" type="REAL-NUMBER" default="1"/>
   <xs:attribute name="similarityScale" type="REAL-NUMBER" use="optional"/>
   <xs:attribute name="compareFunction" type="COMPARE-FUNCTION" use="optional"/>
 </xs:complexType>
</xs:element>
```

Figure 11. The ClusteringField element.

field refers (by name) to a MiningField or to a DerivedField.

fieldWeight is the importance factor for the field. This field weight is used in the comparison functions in order to compute the comparison measure. The value must be a number greater than 0. The default value is 1.0.

similarityScale is the distance such that similarity becomes 0.5. [9]

```
<xs:element name="Comparisons">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="Matrix"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Figure 12. The Comparisons element.

compareFunction is a function of taking two field values and a similarityScale to define similarity/distance. It can override the general specification of compareFunction in ComparisonMeasure.

For the computation of distances and similarities see fig. 12.

Comparison is a matrix which contains the similarity values or distance values, depending on the attribute modelClass in ClusteringModel. The order of the rows and columns corresponds to the order of discrete values or intervals in that field.

Distance or Similarity Measure

When two records are compared then either the distance or the similarity is of interest. In both cases the measures can be computed by a combination of an 'inner' function and an 'outer' function. The inner function compares two single field values and the outer function computes an aggregation over all fields.

Each field has a comparison function, this is either defined as a default in **ClusteringModel** or it can be defined per **ClusteringField**. Given two field values x and y, the inner function can be one of:

absDiff: absolute difference $c(x,y) = |x-y|$ (1)

gaussSim: gaussian similarity

$$c(x,y) = \exp(-\ln(2)*z*z/(s*s)) \quad (2)$$

where z=x-y, and s is the value of attribute **similarityScale** (required in this case) in the **ClusteringField**

delta: $c(x,y) = 0$ *if* $x=y$, *1 else* (3)

equal: $c(x,y) = 1$ *if* $x=y$, *0 else* (4)

table: $c(x,y)$ = *lookup in similarity matrix* (5)

```
<xs:simpleType name="COMPARE-FUNCTION">
  <xs:restriction base="xs:string">
    <xs:enumeration value="absDiff" />
    <xs:enumeration value="gaussSim" />
    <xs:enumeration value="delta" />
    <xs:enumeration value="equal" />
    <xs:enumeration value="table" />
  </xs:restriction>
</xs:simpleType>
```

Figure 13. The COMPARE-FUNCTION element.

ComparisonMeasure

Per **ClusteringModel** there is one aggregation function: depending on the attribute kind in **ComparisonMeasure** the aggregated value is optimal if it is 0 (for distance measure) or greater values indicate optimal fit (for similarity measure).

A model can have MissingValueWeights in order to adjust distance measures for missing data.

Given:

- A data vector Xi, i=1,...,n, where some of the values may be missing
- A vector of cluster seeds Yi, i=1,...,n, all non missing
- A vector of field weight values, Wi, i=1,...,n. Wi is the value of **fieldWeight** in the **ClusteringField** corresponding to the i-th field. If the attribute is missing Wi is assumed to be 1.0.
- A vector of adjustment values, Qi, i=1,...,n, all nonmissing Qi is the i-th value in the element MissingValueWeights. If the model does not have MissingValueWeights, Qi is assumed to be 1.0.
- A function nonmissing() that returns 1 if the argument is nonmissing, or 0 otherwise
- A summation SumNM(expr_i) that returns the sum of expr_i where the index i ranges from 1 to N where Xi is not missing.

The adjustment values are used to compute an adjustment factor

$$AdjustM = \frac{sum[Q_i]}{sum[nonmissing(X_i)*Q_i]} \quad (6)$$

Note that AdjustM = 1.0 if there are no missing values.

The following aggregation functions are defined by PMML. (see Fig.14). [9]

```
<xs:element name="ComparisonMeasure">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
   <xs:choice>
    <xs:element ref="euclidean"/>
      ... other clustering element ...
   </xs:choice>
  </xs:sequence>
  <xs:attribute name="kind" use="required">
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:enumeration value="distance"/>
     <xs:enumeration value="similarity"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
   <xs:attribute name="compareFunction"
                 type="COMPARE-FUNCTION" default="absDiff"/>
   <xs:attribute name="minimum" type="NUMBER" use="optional"/>
   <xs:attribute name="maximum" type="NUMBER" use="optional"/>
 </xs:complexType>
</xs:element>
<xs:element name="euclidean">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
                                  ... etc. ...
```

Figure 14. The tree structures for aggregation functions.

euclidean: kind=distance

$$D = (sumNM(\ Wi * c(Xi,Yi)^2\ ) * AdjustM\ )^{(1/2)} \quad (7)$$

squaredEuclidean, aka squared: kind=distance

$$D = sumNM(\ Wi * c(Xi,Yi)^2\ ) * AdjustM \quad (8)$$

euclidean and squaredEuclidean are essentially equivalent because only the minimum or maximum of values is needed in order to assign a cluster.

chebychev, aka maximum: kind=distance

$D = max\ (Wi*c(Xi,Yi))\ *AdjustM\ over\ all\ i$ (9)

cityBlock, aka sum:

$D = sumNM\ (Wi*c(Xi,Yi))\ *AdjustM$ (10)

minkowski: p-parameter>0

$D = (sumNM(c(Xi,Yi)^p)\ *AdjustM)^{(1/p)}$ (11)

For binary or categorical data, let two individuals X and Y compare their values for each attribute, and

a11 = number of times where Xi=1 and Yi=1;
a10 = number of times where Xi=1 and Yi=0 ;
a01 = number of times where Xi=0 and Yi=1;
a00 = number of times where Xi=0 and Yi=0

simpleMatching: kind=similarity, min=0, max=1

$D = (a11 + a00) / (a11 + a10 + a01 + a00)$ (12)

jaccard: kind=similarity

$D = (a11) / (a11 + a10 + a01)$ (13)

tanimoto: kind=similarity, min=0, max=1

$D = (a11 + a00)/(a11 + 2*(a10+a01) + a00)$. (14)

binarySimilarity: kind=similarity, min=0, max=1, c.. and d.. are parameters > 0.

$$D = \frac{c11*a11 + c10*a10 + c01*a01 + c00*a00}{d11*a11 + d10*a10 + d01*a01 + d00*a00} \quad (15)$$

The attributes `minimum` and `maximum` in `ComparisonMeasure` are not used in the scoring function. [9]

Example for a center-based clustering model

```
<?xml version="1.0" ?> <PMML version="3.1" xmlns=http://www.dmg.org/PMML-3_1
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Header copyright="dmg.org"/>
 <DataDictionary numberOfFields="3">
  <DataField name="marital status" optype="categorical" dataType="string">
    <Value value="s"/> <Value value="d"/> <Value value="m"/> </DataField>
  <DataField name="age" optype="continuous" dataType="double"/>
  <DataField name="salary" optype="continuous" dataType="double"/>
 </DataDictionary>
  <ClusteringModel modelName="Mini Clustering" functionName="clustering"
   modelClass="centerBased" numberOfClusters="2">
   <MiningSchema> <MiningField name="marital status"/>
    <MiningField name="age"/> <MiningField name="salary"/> </MiningSchema>
   <ComparMeasure kind="distance"><squaredEuclidean/></ComparMeasure>
    <ClusteringField field="marital status" compareFunction="absDiff"/>
    <ClusteringField field="age" compareFunction="absDiff"/>
    <ClusteringField field="salary" compareFunction="absDiff"/>
    <CenterFields>
     <DerivedField name="c1" optype="continuous" dataType="double">
      <NormCont field="age"> <LinearN orig="45" norm="0"/>
       <LinearN orig="82" norm="0.5"/><LinearN orig="105" norm="1"/>
      </NormCont> </DerivedField>
     <DerivedField name="c2" optype="continuous" dataType="double">
      <NormCont field="salary"> <LinearN orig="39000" norm="0"/>
       <LinearN orig="39800" norm="0.5"/><LinearN orig="41000" norm="1"/>
      </NormCont> </DerivedField>
     <DerivedField name="c3" optype="continuous" dataType="double">
       <NormDiscr field="marital status" value="m"/> </DerivedField>
     <DerivedField name="c4" optype="continuous" dataType="double">
       <NormDiscr field="marital status" value="d"/> </DerivedField>
     <DerivedField name="c5" optype="continuous" dataType="double">
       <NormDiscr field="marital status" value="s"/> </DerivedField>
    </CenterFields>
    <MissingValueWeights> <Array n="5" type="real">1 1 1 1 1</Array>
    </MissingValueWeights>
    <Cluster name="marital status is d or s">
     <Array n="5" type="real"> 0.524561 0.486321 0.128427 0.459188 0.412384</Array>
    </Cluster>
    <Cluster name="marital status is m">
      <Array n="5" type="real"> 0.69946 0.419037 0.591226 0.173521 0.235253</Array>
    </Cluster>
  </ClusteringModel>
</PMML>
```

Figure 15. Example for a center-based clustering model.

Conformance of Center-based clustering:

The comparison function 'absDiff' (that is, the value of attribute 'compareFunction' in 'CompaMeasure') is in core, other comparison functions are not in core. The aggregation function 'squaredEuclidean' is in core other aggregation functions are not in core.

4 Conclusions

The language PMML provides a flexible mechanism for defining schema for predictive models and supports model selection and model averaging involving multiple predictive models. It has proved useful for applications requiring ensemble learning, partitioned learning, and distributed learning. In addition, it facilitates moving predictive models across applications and systems.

The future of PMML apps may be cover the following topics: the need for PMML to be responsive to the needs of vendors so users can be more successful in deploying models that use sophisticated features of data mining tools; including supplement info that would be useful for tracking model behaviour and understand conditions when the model was created (for example, date of training data collect matrices, and economic data like the inflation rate, energy costs, interest rate; support for transformations that are beyond PMML's transformations and built-in functions (perhaps using inline SQL or C code).

Our contribution is that we create a general framework for applications from web data mining through PMML standard.

References:

[1]E.van der Vlist, *XML Schema*, ed. Amazon, 2004.

[2]R. Pechter, Data mining standards, services and platforms 2005 workshop report, *ACM SIGKDD Explorations Newsletter*, Volume 7 Issue 2, pp: 137-138 , 2005.

[3]E. R. Harold, W. S. Means, *XML in a Nutshell*, Third Edition, ed. Amazon, 2004.

[4]E.R.Harold, *XML 1.1 Bible*, ed. Amazon, 2004.

[5]C. Swann, G. Caines, *XML in FLASH*, ed. Teora, 2002.

[6]R. van Lent, D. Zwietering, *Technical Integration of Data Mining*, http://www.dmreview.com/editorial/ newsletters.cfm, 2002.

[7]XML schema data types, XML schema structures, XML schema primer at http://www.w3c.org/TR/

[8]PMML 3.1, http://www.dmg.org/pmml-v3-1.html

[9]http://www.dmg.org/v31/ClusteringModel.html