PMML in Simulation

Svetlana Levitan, PhD
IBM Corp.
200 W. Madison St
Chicago, IL 60606
312-477-4570
slevitan@us.ibm.com

Richard Cohen, PhD
IBM Corp.
Indianapolis, IN
rjcohen@us.ibm.com

Vladimir Shklover, PhD
IBM Corp.
200 W. Madison St
Chicago, IL 60606
312-477-4617
vshklove@us.ibm.com

ABSTRACT
Predictive models use a set of inputs to predict an outcome. In cases where the input values are not known with certainty, Monte Carlo simulation can be used to generate input values based on specified probability distributions for the uncertain inputs. The simulated values are then used in the predictive model to generate an outcome. The process is repeated many times, resulting in an approximation to the distribution(s) of the target (or targets) of the model that can be used to answer questions about the likelihood of various outcomes.

Predictive Model Markup Language (PMML) is an industry standard for representing statistical and data mining models in XML. It is a very convenient way to represent model information for use in a Monte Carlo simulation. This paper describes how PMML enables a seamless transition from traditional modeling to modeling in the presence of uncertainty through Monte Carlo simulation.

Categories and Subject Descriptors
G.3 [Probability and Statistics]: Statistical Computing, Statistical Software; I.5.1 [Models]: Statistical Models, Neural Nets; D4.8 [Performance]: Modeling and Prediction, Simulation.

General Terms
Management, Performance, Standardization, Languages.

Keywords
Simulation, Open Standards, Predictive Analytics, Data Mining, PMML, Predictive Model Markup Language, Generalized Linear Model, Neural Networks, MLP.

1. INTRODUCTION
Predictive models, such as linear regression or decision tree, require a set of known inputs to predict an outcome. In many real-world applications, however, inputs are not known with certainty, and users are interested in accounting for that uncertainty in their models. For example, given a profit model that includes cost of materials as an input, users may want to account for uncertainty in that input and determine the likelihood that profit will fall below a target value. Uncertainty in the inputs to predictive models can be addressed with Monte Carlo simulation, which results in a distribution of model outcomes that can be used to answer questions that are probabilistic in nature.

Predictive Model Markup Language (PMML) [1][2] is an industry standard for representing statistical and data mining models in XML. PMML can contain a wide range of predictive models and comprehensive summary statistics for the predictors, making it a very convenient way to represent model information for use in a Monte Carlo simulation.

IBM SPSS Statistics 21, released in 2012, introduced support for performing Monte Carlo simulations based on predictive models represented in PMML. It contains Simulation Builder, which is an advanced interactive interface for users who are designing and running simulations. It provides the full set of capabilities for designing a simulation, saving the specifications to a simulation plan file, specifying output and running the simulation. From the creation of a simulation plan to running the simulation, PMML is used as the primary source of information about the predictive model on which the simulation is based. In that regard, the predicted values of the target variable(s) are computed by scoring the PMML model.

This paper uses hypothetical examples to describe how PMML enables a seamless transition from traditional modeling to modeling in the presence of uncertainty through Monte Carlo simulation.

2. THE PROCESS
Each input in a predictive model, that is to form the basis of a simulation, is specified as simulated or fixed. Values of simulated inputs are generated by drawing from a specified probability distribution (such as the triangular distribution), allowing you to account for uncertainty in the values of those inputs. Fixed inputs are those whose values are known and held fixed at the known values.

The predictive model is evaluated using the generated values of the simulated inputs and the specified values of the fixed inputs to calculate the target (or targets) of the model. The process is repeated many times (typically tens of thousands or hundreds of thousands of times), resulting in a distribution of target values. Each repetition of the process generates a separate data record that consists of the values of the inputs and the predicted target (or targets) of the model. The distribution of predicted target values can then be used to evaluate the likelihood of various outcomes.

PMML'13, August 11, 2013, Chicago, IL, USA.

http://dx.doi.org/10.1145/2501055.2501063

3. EXAMPLE 1: APPLYING SIMULATION TO MODELING DIABETES TREATMENT COSTS

In the healthcare field, simulation might be used to model treatment costs for a particular population of patients, such as patients with diabetes. For example, a healthcare insurance provider might want to know the likelihood--over the entire population of policy holders with diabetes--that annual per patient cost will exceed some threshold.

Statements based on the distribution of costs over a given population of policy holders require both a predictive model of costs and sufficient data to represent the underlying distribution. Perhaps there is sufficient data to build a predictive model of costs but there is concern that the data do not provide adequate coverage of the space of input values to use it alone as a model of the distribution of costs. Using simulation, you can simulate as much data as needed and apply the predictive model to the data to obtain the cost distribution.

In this hypothetical example, a generalized linear model for treatment cost was built with the GENLIN procedure in IBM SPSS Statistics 21 (it could also have been built in IBM SPSS Modeler 14). Figure 1 shows the PMML that describes the predictive model that will be used for the simulation.

```
<DataDictionary numberOfFields="4">
   <DataField name="age" displayName="Age in years"
    optype="continuous" dataType="double" />
   <DataField name="income" displayName="Household income"
    optype="continuous" dataType="double" />
   <DataField name="glucose" displayName="Glycated hemoglobin level"
    optype="continuous" dataType="double" />
   <DataField name="cost" displayName="Treatment costs"
    optype="continuous" dataType="double" />
</DataDictionary>
<GeneralRegressionModel modelType="generalizedLinear"
  modelName="diabetes_costs" algorithmName="GZLM"
  functionName="regression" linkFunction="log" distribution="normal">
 <MiningSchema>
  <MiningField name="age" />
  <MiningField name="income" />
  <MiningField name="glucose" />
  <MiningField name="cost" usageType="predicted" />
 </MiningSchema>
 <ParameterList>
  <Parameter name="P0000001" label="Intercept" />
  <Parameter name="P0000002" label="age" />
  <Parameter name="P0000003" label="income" />
  <Parameter name="P0000004" label="glucose" />
 </ParameterList>
 <CovariateList>
  <Predictor name="age" />
  <Predictor name="income" />
  <Predictor name="glucose" />
 </CovariateList>
 <PPMatrix>
  <PPCell value="1" predictorName="age" parameterName="P0000002" />
  <PPCell value="1" predictorName="income" parameterName="P0000003" />
  <PPCell value="1" predictorName="glucose" parameterName="P0000004" />
 </PPMatrix>
 <ParamMatrix>
  <PCell parameterName="P0000001" beta="6.09743161075828" df="1" />
  <PCell parameterName="P0000002" beta="0.0207898463088766" df="1" />
  <PCell parameterName="P0000003" beta="-1.13778193530211e-005" df="1" />
  <PCell parameterName="P0000004" beta="0.334202953548899" df="1" />
 </ParamMatrix>
</GeneralRegressionModel>
```

Figure 1. PMML 4.0 code fragment for generalized linear model.

The PMML contains a DataDictionary element [3] that describes the fields used in the model, all of which are continuous. The model element GeneralRegressionModel [4] specifies the model type as "generalizedLinear" and shows that the link function is "log" and that the target distribution is "normal". The "MiningSchema" [5] element shows that the variable *cost* is the target and that the other three variables are predictors. The "ParameterList" element indicates that an intercept is included in the model. The "PPMatrix" element shows the correspondence between the model parameters and the predictors. Finally, the "ParamMatrix" element contains the beta coefficients for the model.

To build a simulation of the cost distribution for this model, you load the PMML model file into the Simulation Builder in IBM SPSS Statistics. The Simulated Inputs grid (see Figure 2) lists all of the fields that are inputs in the predictive model. The model used in this example contains the following input fields: *age* is the age of the individual covered by the policy; *glucose* is the individual's average glycated hemoglobin level, which indicates the blood glucose level over prolonged periods; *income* is the household income of the individual.

To run a simulation, one must specify a distribution for each input in the model, or specify the input as fixed and provide the fixed value. In this example, all model inputs will be simulated, so a distribution must be specified for each of the three inputs. When the data used to build the predictive model are available, as in this example, one can automatically find the distribution that most closely fits the data for each of the inputs.

The results are shown in the Distribution column in Figure 2. The name of the distribution that most closely fits the data for each input is displayed, along with the distribution parameters. For example, the data for *glucose* are most closely fit by a lognormal distribution with parameters a=7.55 and b=0.19. The chart adjacent to the distribution parameters shows the distribution function superimposed on a histogram of the data that were used for fitting that input.

Users can examine all distributions that were considered when fitting a particular input, along with the goodness of fit statistics for each fitted distribution.

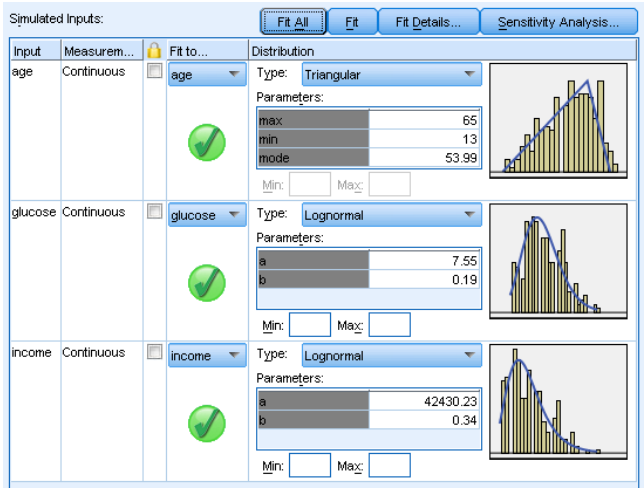

Figure 2. Portion of the Simulation Builder in SPSS Statistics showing best-fitting distributions for simulated inputs.

The range of simulated values for a given distribution can be adjusted to ensure that the simulated values make sense for the current application. For example, the lognormal distribution for *glucose* includes all positive values (the value 0 is excluded), but

glycated hemoglobin levels for people with diabetes are typically in the range from 5 to 14 so you might restrict the range of simulated values to lie between 5 and 14.

Specifications for a simulation are always written to a simulation plan file, which contains all of the information needed to run the simulation. It includes the PMML model file for the predictive model (when the simulation is based on a PMML model), probability distributions for the simulated inputs, values for any fixed inputs, correlations between simulated inputs, as well as other settings. Users can open a simulation plan file in SPSS Statistics, optionally make modifications and re-run the simulation without having to re-enter all of the specifications; and simulation plan files can be shared between users.

The output from running a simulation contains both tables and charts. For example, the Correlation tornado chart (see Figure 3) shows the Pearson correlation between a continuous target and its simulated inputs. In the current example of diabetes treatment costs, the chart shows the correlation between the target *cost* and the simulated inputs *age*, *income* and *glucose*.

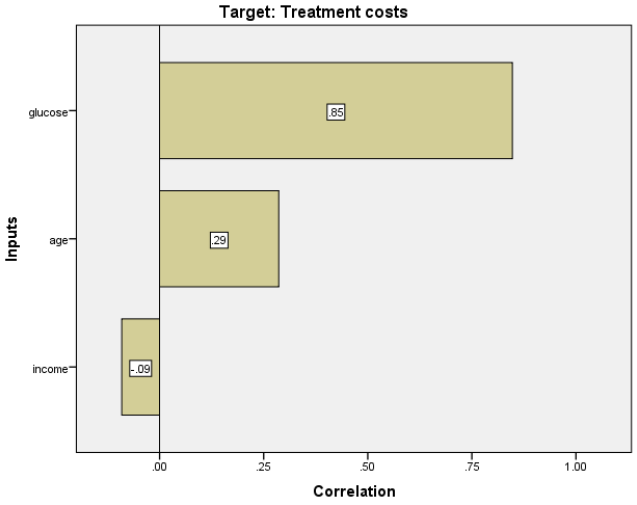

Figure 3. Correlation tornado chart showing Pearson correlations between a continuous target and the simulated inputs.

The Probability Density chart (see Figure 4) displays the distribution of the target of the predictive model that is generated by the simulation. It includes moveable reference lines that allow users to interactively probe the distribution. A table below the chart displays the probabilities in the regions bounded by the reference lines. For example, you can set the right-hand line at a given cost value and then easily read off the probability that costs exceed that value. In Figure 4, the right-hand line is set to $20000 and the table shows that there is about an 8% chance that annual per patient costs exceed that value.

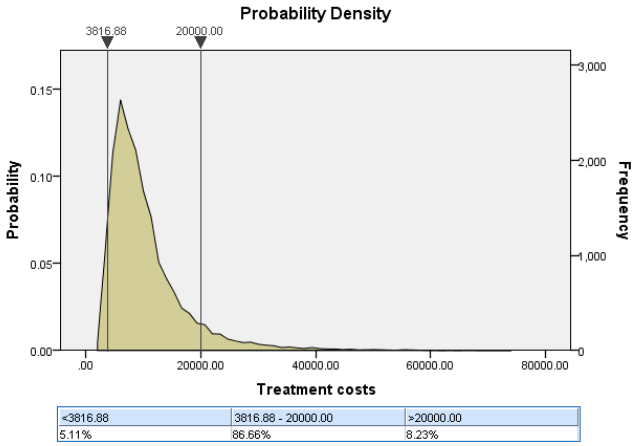

Figure 4. Probability density chart showing the distribution of target values from the simulation.

4. EXAMPLE 2: USING SIMULATION TO MODEL BANK LOAN DEFAULTS

In the banking industry, simulation might be used for estimating the percentage of loans that will go into default. A particular bank may have a model for predicting whether a given loan applicant will default on their loan. As in the previous example, there may be sufficient data to build a predictive model but there may be concern that the data do not provide adequate coverage of the input space to make statements about the underlying population of loan applicants. Using simulation, the bank can simulate as much data as needed and apply the predictive model to the data to obtain a better representation of the underlying distribution.

In this hypothetical example, a neural network model for predicting loan defaults was built with the multilayer perceptron algorithm (or MLP) in IBM SPSS Statistics. Figure 5 below shows the PMML that describes the predictive model that will be used for the simulation.

```
<DataDictionary numberOfFields="5">
 <DataField name="default" displayName="Previously defaulted"
 optype="categorical" dataType="double">
  <Value value="0" displayValue="No" />
  <Value value="1" displayValue="Yes" />
 </DataField>
 <DataField name="debtinc" displayName="Debt to income ratio (x100)"
 optype="continuous" dataType="double" />
 <DataField name="address" displayName="Years at current address"
 optype="continuous" dataType="double" />
 <DataField name="employ" displayName="Years with current employer"
 optype="continuous" dataType="double" />
 <DataField name="creddebt" displayName="Credit card debt in thousands"
 optype="continuous" dataType="double" />
</DataDictionary>
<TransformationDictionary>
 <DerivedField optype="categorical" dataType="double" name="defaultValue0">
   <NormDiscrete field="default" value="0" />
 </DerivedField>
 <DerivedField optype="categorical" dataType="double" name="defaultValue1">
  <NormDiscrete field="default" value="1" />
 </DerivedField>
 <DerivedField optype="continuous" dataType="double" name="debtincNorm">
    <NormContinuous field="debtinc">
      <LinearNorm orig="0.4" norm="1.42831440532609" />
      <LinearNorm orig="41.3" norm="4.48233880100496" />
    </NormContinuous>
 </DerivedField>
```

```
 <DerivedField optype="continuous" dataType="double" name="addressNorm">
   <NormContinuous field="address">
     <LinearNorm orig="0" norm="1.21921047706448" />
     <LinearNorm orig="34" norm="3.78141153367095" />
   </NormContinuous>
 </DerivedField>
 <DerivedField optype="continuous" dataType="double" name="employNorm">
   <NormContinuous field="employ">
     <LinearNorm orig="0" norm="1.25620371411656" />
     <LinearNorm orig="31" norm="3.39662499186859" />
   </NormContinuous>
 </DerivedField>
 <DerivedField optype="continuous" dataType="double" name="creddebtNorm">
   <NormContinuous field="creddebt">
     <LinearNorm orig="0.011696" norm="0.719703140552011" />
     <LinearNorm orig="20.56131" norm="8.67979989515938" />
   </NormContinuous>
 </DerivedField>
</TransformationDictionary>
<NeuralNetwork functionName="classification" activationFunction="tanh">
 <MiningSchema>
  <MiningField name="debtinc" />
  <MiningField name="address" />
  <MiningField name="employ" />
  <MiningField name="creddebt" />
  <MiningField name="default" usageType="predicted" />
 </MiningSchema>
 <NeuralInputs>
  <NeuralInput id="0">
   <DerivedField optype="continuous" dataType="double">
    <FieldRef field="debtincNorm" />
   </DerivedField>
  </NeuralInput>
  <NeuralInput id="1">
   <DerivedField optype="continuous" dataType="double">
    <FieldRef field="addressNorm" />
   </DerivedField>
  </NeuralInput>
  <NeuralInput id="2">
   <DerivedField optype="continuous" dataType="double">
    <FieldRef field="employNorm" />
   </DerivedField>
  </NeuralInput>
  <NeuralInput id="3">
   <DerivedField optype="continuous" dataType="double">
    <FieldRef field="creddebtNorm" />
   </DerivedField>
  </NeuralInput>
 </NeuralInputs>
 <NeuralLayer numberOfNeurons="1">
  <Neuron id="4" bias="0.44410958316748">
   <Con from="0" weight="0.261033430978141" />
   <Con from="1" weight="0.24171294814757" />
   <Con from="2" weight="0.736484824175965" />
   <Con from="3" weight="0.567379316580886" />
  </Neuron>
 </NeuralLayer>
 <NeuralLayer numberOfNeurons="2" activationFunction="identity"
  normalizationMethod="softmax">
  <Neuron id="5" bias="0.707248459004506">
   <Con from="4" weight="1.65089666902876" />
  </Neuron>
  <Neuron id="6" bias="0.00665819072845599">
   <Con from="4" weight="1.12916867843686" />
  </Neuron>
 </NeuralLayer>
 <NeuralOutputs>
  <NeuralOutput outputNeuron="5">
   <DerivedField optype="categorical" dataType="double">
    <FieldRef field="defaultValue0" />
   </DerivedField>
  </NeuralOutput>
  <NeuralOutput outputNeuron="6">
   <DerivedField optype="categorical" dataType="double">
    <FieldRef field="defaultValue1" />
   </DerivedField>
  </NeuralOutput>
 </NeuralOutputs>
</NeuralNetwork>
```

Figure 5. PMML 4.0 code fragment for neural network MLP model.

The DataDictionary element contains all five fields used in the model. The TransformationDictionary [6] element describes the transformations applied to the original inputs before they can be used in a neural network model.

Usually, continuous variables are standardized so that their values range between -1 and 1, and categorical variables (including the target) are transformed into groups of "dummy" variables, one for each category value. PMML provides a large set of tools for describing various transformations. Each new transformed variable is described by a DerivedField element, and all those elements are placed into a TransformationDictionary element. The standardization of continuous variables is described by the NormContinuous element that contains a piece-wise linear function. Each LinearNorm element has one point; two such points are needed to define a line. Dummy variables for categorical predictors or targets are created using the NormDiscrete element. This element contains a variable name and its category value. The derived field has the value of 1 when the value of the variable in a record matches the specified category value and zero otherwise. Some more advanced features have recently been added to this part of the standard, but they are not used in this example.

The neural network MLP model usually consists of an input layer, one or more hidden layers and an output layer. Each layer has one or more neurons. Each neuron is connected to the neurons in the previous layer, and the connections have weights. The weights are initialized with small random numbers and then modified during the network's training on the data until convergence is achieved. PMML describes all of this information very clearly using the NeuralNetwork element [7] that includes MiningSchema, NeuralInputs, NeuralLayer, NeuralOutputs, Neuron and Con (for the connection) elements. The diagram of the model expressed in the PMML is shown in Figure 6.

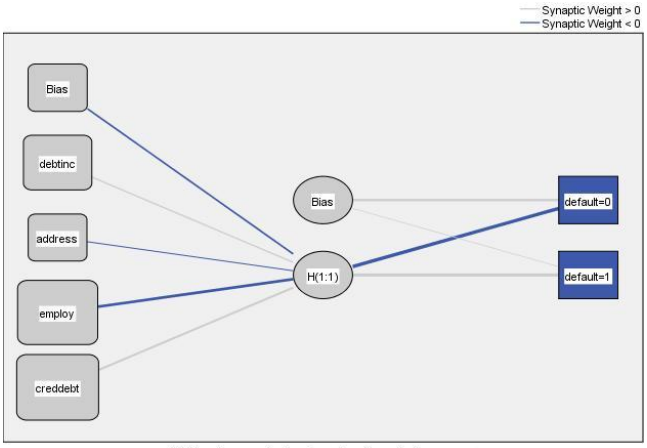

Figure 6. Network diagram for neural network MLP model.

To build a simulation based on this model, you load the PMML model file into the Simulation Builder in IBM SPSS Statistics (as done in the previous example) and determine the best-fitting distribution for each of the model inputs based on the available loan applicant data. In this hypothetical example, you find that the input variables can best be described by the following distributions: *address* and *employ* are best fit by an exponential distribution; *creddebt* is best fit by a lognormal distribution; *debtinc* is best fit by a gamma distribution. Saving these

distributions and the PMML model to a simulation plan, you would run the simulation. This involves generating the input values according to the specified distributions and scoring the PMML model with the simulated inputs, taking into account the data transformations specified in the PMML.

The model for the current example has a categorical target so the target distribution is represented as a bar chart (see Figure 7) with the two values "Yes" and "No", specifying whether applicants will or will not default on a loan.

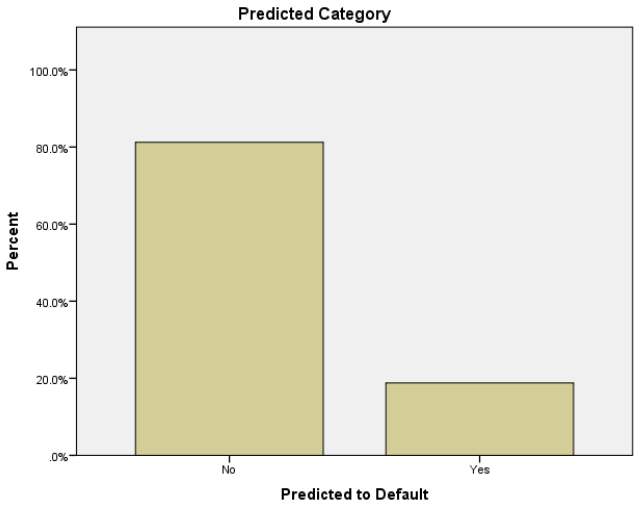

Figure 7. Bar chart showing the percentages for each of the target values.

5. CONCLUSION

PMML can describe a wide range of statistical and data mining models and data transformations and allow easy portability of these models between different products. The use of PMML in simulation frees the user from the need to worry about representing their model and allows them to concentrate on other aspects of simulation such as selecting best distributions. It also allows a fast and easy application of simulation analysis to a number of different models built on the same data.

6. ACKNOWLEDGEMENTS

The authors are grateful to IBM SPSS colleagues for their support and help with this project. We also thank other DMG members for their cooperation and hard work in advancing the PMML standard to its current state.

7. REFERENCES

[1] DMG website: www.dmg.org
[2] A. Guazzelli, W. Lin, T. Jena (2012). *PMML in Action (2nd edition): Unleashing the Power of Open Standards for Data Mining and Predictive Analytics.* CreateSpace.
[3] http://www.dmg.org/v4-1/DataDictionary.html
[4] http://www.dmg.org/v4-1/GeneralRegression.html
[5] http://www.dmg.org/v4-1/MiningSchema.html
[6] http://www.dmg.org/v4-1/Transformations.html
[7] http://www.dmg.org/v4-1/NeuralNetwork.html