Tutorial: Building and Deploying Predictive Analytics Models Using the PMML Standard
Tutorial Overview

- Introduction: What is PMML? 3
  - Brief History and Main Components of PMML 5
- Building Models in R and exporting PMML 14
- Creating PMML in IBM SPSS Statistics 16
  - Scoring PMML in IBM SPSS Statistics 28
- Creating PMML in IBM SPSS Modeler 35
  - Scoring PMML in IBM SPSS Modeler 42
- Using PMML in Simulation 44
- Q&A
Introduction: What is PMML?

- An Industry Standard for XML Representation of Statistical and Data Mining Models
- Allows easy deployment of models and product interoperability
- Supported by over 20 vendors and organizations: commercial, government, and open source
- It is an open standard: issues in Mantis, code in GitHub.
- Developed by the Data Mining Group (DMG): dmg.org
- The Data Mining Group is a consortium managed by the Center for Computational Science Research, Inc., which is an Illinois based 501(c)(3) not-for-profit corporation
PMML 4.3 - General Structure

PMML 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. A PMML document is an XML document with a root element of type PMML. The general structure of a PMML document is:

```xml
<?xml version="1.0"?>
<PMML version="4.3"
xmlns="http://www.dmg.org/PMML-4_3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

Brief History of PMML

- Conceived by Dr. Robert Grossman, then the director of the National Center for Data Mining at the UIC
- Release 0.7 came out in 1997

- Release 1.1 (August 2000) had only 6 models:
  - `<!ENTITY % A-PMML-MODEL '([^TreeModel|NeuralNetwork|ClusteringModel|RegressionModel|GeneralRegressionModel|AssociationModel]+)'>`
  - `<!ELEMENT PMML (Header, DataDictionary, (%A-PMML-MODEL;)+, Extension*)>`

- Release 2.0 (August 2001) added TransformationDictionary, Naïve Bayes and Sequence models
Brief History of PMML, cont’d

- Releases up to 2.0 used a DTD, 2.1 was the first with XML Schema
- Release 3.0 (October 2004) introduced Functions, date/time data types, Model Composition (only composing trees and Regression models), Model Verification, Output, Targets, LocalTransformations, SVM, Ruleset, some new features in existing elements
- Release 4.0 (June 2009) had MultipleModels replace ModelComposition (to allow model ensembles, segmented models, as well as supporting old model composition), added Cox Regression into General Regression, added ModelExplanation and new built-in functions including “if-then-else” construct, added Time Series Model with only Exponential Smoothing.
- Release 4.1 (December 2011) added Baseline model, KNN, ScoreCard
- Last released version is 4.3, in August 2016. Added GaussianProcess and BayesianNetwork models, new built-in functions.
Main Components of PMML

- Header
- Data Dictionary
- Transformation Dictionary
- Model(s)
Contents of a PMML Model

- **Mining Schema**: target and predictors, importance, missing value treatment, invalid value treatment, outlier treatment
- **Output**: what to report, post-processing
- **Model Stats**: description of input data
- **Model Explanation**: model diagnostics
- **Targets**: target category info and prior probabilities
- **Local Transformations**: predictor transformations local to the model
- …*<Specific model contents>*…
- **Model Verification**: expected results for some cases
Specific Contents of Some PMML Models

✓ **Association Model:** Item, ItemSet, AssociationRule

✓ **Clustering Model:** ComparisonMeasure, ClusteringField, Cluster

✓ **General Regression:** ParameterList, FactorList, CovariateList, PPMatrix, PCovMatrix, ParamMatrix

✓ **Naïve Bayes:** BayesInputs, BayesOutput, TargetValueCounts, TargetValueStats

✓ **NearestNeighborModel:** ComparisonMeasure, KNNInputs

✓ **NeuralNetwork:** NeuralLayer, NeuralInput, Neuron, NeuralOutput

✓ **Regression:** RegressionTable, NumericPredictor, CategoricalPredictor, PredictorTerm

✓ **Sequence Model:** Item, ItemSet, Sequence, SequenceRule

✓ **TreeModel:** Node, predicates, ScoreDistribution

✓ **MiningModel:** composition or ensemble (or both) of models
PMML Transformations

- **NormContinuous**: piece-wise linear transform
- **NormDiscrete**: map a categorical field to a set of dummy fields
- **Discretize**: convert a continuous field into an ordinal one (e.g. by binning)
- **MapValues**: map one or more categorical fields into another categorical one, by explicitly specifying all mappings
- **Functions**: built-in and user-defined
- **TextIndex**: converting text into structured data
- **Lag**: use data from previous times
- **Aggregations**: for use in Association or Sequence models
Benefits of PMML

- Transparency: human and machine-readable
- Fosters best practices in model building and deployment
- Transforms models into dynamic assets by fast deployment
PMML Powered

- From http://dmg.org/pmml/products.html:
  - Alpine Data
  - Angoss
  - BigML
  - Equifax
  - Experian
  - FICO
  - Fiserv
  - Frontline Solvers
  - GDS Link
  - IBM (Includes SPSS)
  - JPMML
  - KNIME
  - KXEN
  - Liga Data
  - Microsoft
  - MicroStrategy
  - NG Data
  - Open Data
  - Opera
  - Pega
  - Pervasive Data Rush
  - Predixion Software
  - Rapid I
  - R
  - Salford Systems (Minitab)
  - SAND
  - SAS
  - Software AG
  - Spark
  - Sparkling Logic
  - Teradata
  - TIBCO
  - WEKA
  - Zementis
PMML in R

- R is an open source programming language and software environment for statistical computing and graphics that is supported by the R Foundation for Statistical Computing.

- [https://cran.r-project.org/web/packages/pmml/](https://cran.r-project.org/web/packages/pmml/)
- Depends on XML package
- Supports a number of R models
- Maintained by Tridivesh Jena from Zementis (now Software AG)

- `library(XML);`
- `library(pmml);`
Create PMML in R

> data(iris);
> iris[1,]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
  1      5.1      3.5      1.4      0.2  setosa

Build and save a linear regression model:
> irisLR<-lm(Sepal.Length~.,iris)
> pmml(irisLR)
> saveXML( pmml(irisLR), "IrisLR.xml" )

Build and save a decision tree (C&RT) model:
> irisTree <- rpart( Species~., iris )
> pmml( irisTree )
> saveXML( pmml( irisTree ), "IrisTree.xml" )
IBM SPSS Statistics


Employee Data (fake data)

Start IBM SPSS Statistics, Close the splash screen, File→Open→Data
On Windows (when installed in default location):
C:\Program Files\IBM\SPSS\Statistics\25\Samples\English\Employee Data.sav
On MAC:
Applications/IBM/SPSS/Statistics/25/Samples/English/Employee Data.sav
Examine the data and save it in CSV format

To save in CSV: File→Save As… Select “Comma delimited”, uncheck box for saving variable names, pick a path for your file.
Analyze the Data in IBM SPSS Statistics
Decision Tree Model in IBM SPSS Statistics
Define Saved Variables (scores) and PMML export
Paste the Syntax.
Convenient for saving and editing the jobs.
Build the Model and Examine Results

In the output viewer double-click the tree diagram to open the “Tree Editor”
Bayesian Regression Model

Bayesian inference about Linear Regression is a statistical method that is broadly used in quantitative modeling. Bayesian linear regression is an approach to Linear Regression where the statistical analysis is undertaken within the context of Bayesian inference.

Start at Analyze
Bayesian Regression cont’d

Select the dependent and predictors, pick “Use Both Methods”, then go to **Save** to create PMML and to get some scores.
Bayesian Regression cont’d

![Bayesian Linear Regression Models: Save dialog box](image)

<table>
<thead>
<tr>
<th>Save</th>
<th>Item to Save</th>
<th>Custom Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑</td>
<td>Means</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variances</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modes</td>
<td></td>
</tr>
<tr>
<td>☑</td>
<td>Credible interval lower limit</td>
<td></td>
</tr>
<tr>
<td>☑</td>
<td>Credible interval upper limit</td>
<td></td>
</tr>
</tbody>
</table>

Export model information to XML file:

```xml
C:\Projects\AdvancedManufacturing\BayesRegrEmployee.xml
```

[Buttons: Continue, Cancel, Help]
Estimate Bayes Regression Model and get results

- Parameter: (Intercept) 24582.037
- Parameter: Gender = Female -2603.909
- Parameter: Gender = Male 
- Parameter: Employment Category = Clerical -13763.277
- Parameter: Employment Category = Custodial
- Parameter: Employment Category = Manager

Bayesian Estimates of Coefficients

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mode</th>
<th>Posterior Mean</th>
<th>Variance</th>
<th>95% Credible Interval Lower Bound</th>
<th>95% Credible Interval Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>24582.037</td>
<td>24582.037</td>
<td>5888052.369</td>
<td>19823.997</td>
<td>29340.078</td>
</tr>
<tr>
<td>Gender = Female</td>
<td>-2603.909</td>
<td>-2603.909</td>
<td>643545.687</td>
<td>-4176.921</td>
<td>-1030.897</td>
</tr>
<tr>
<td>Gender = Male</td>
<td></td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>Employment Category =</td>
<td></td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>Clerical</td>
<td>-13763.277</td>
<td>-13763.277</td>
<td>2070653.047</td>
<td>-16584.880</td>
<td>-10941.673</td>
</tr>
<tr>
<td>Employment Category =</td>
<td></td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>Custodial</td>
<td></td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>Employment Category =</td>
<td></td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>Manager</td>
<td></td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
</tbody>
</table>

- Beginning Salary
- Log Likelihood Function
- Prior Distribution
- Posterior Distribution
Score PMML in IBM SPSS Statistics

Utilities->Scoring Wizard
Select Scoring Results to Save

Model Name: Employee_Chaid
Model Type: TREE_PMML

Select Scoring Functions:
Each selected function will create a new field in the dataset.

<table>
<thead>
<tr>
<th>Function</th>
<th>Field Names</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability of Predicted Category</td>
<td>PredictedProbability</td>
<td></td>
</tr>
<tr>
<td>Probability of Selected Category</td>
<td>SelectedProbability</td>
<td>3</td>
</tr>
<tr>
<td>Predicted Value</td>
<td>PredictedValueFromPMML</td>
<td></td>
</tr>
<tr>
<td>Node Number</td>
<td>NodeNumber</td>
<td></td>
</tr>
<tr>
<td>Confidence</td>
<td>Confidence</td>
<td></td>
</tr>
</tbody>
</table>

© 2017 IBM Corporation
Paste the Scoring Syntax
Examine Scoring Results
Score the Iris Example from R

Get Iris.csv from http://dmg.org/pmml/pmml_examples/index.html#Iris

Notice slight changes in field names from the R version. Our ability to match fields helps here.
Establish the Correspondence of Predictors
Scores
IBM SPSS Modeler

First released in 1994 as Clementine by Integral Systems Limited (ISL) of UK

ISL acquired by SPSS in 1998

Graphical interface for data mining/predictive analytics
Analyze the Data in IBM SPSS Modeler

- Create a Stream: source node, associate it to a file with data.
Analyze the Data in IBM SPSS Modeler

- Add a type node (specifying the target), then a model
- There is Auto-Modeling and many individual models
Add a Model to the Stream and Run It
Change Stopping Rules under Build Options for Consistency
Run the Stream, Get a Model Nugget
Export PMML by Right-Clicking on the Model Nugget in the Upper Right Corner
Score PMML in IBM SPSS Modeler

- File ➔ Models ➔ Import PMML
Drag the Model Nugget to the Canvas, Connect to the Data and to Output Table
Motivation for Monte-Carlo Simulation

- Predictive models require a set of inputs to predict an outcome
- Often the inputs are not known with certainty
- Monte-Carlo simulation produces a distribution of model outcomes that can be used to answer questions that are probabilistic in nature.
- Simulation works by:
  - randomly generating values for some inputs based on probability distributions and considering correlations, holding other inputs fixed
  - scoring the model
  - collecting statistics on predicted values and their dependence on the inputs
- PMML supports a wide range of statistical and data mining models and a good set of transformations.
- IBM SPSS Statistics 21 introduced a Simulation module based on PMML representation of models
Example 1: Diabetes treatment costs model in PMML

```xml
<DataDictionary numberOfFields="4">
  <DataField name="age" displayName="Age in years"
    otype="continuous" dataType="double" />
  <DataField name="income" displayName="Household income"
    otype="continuous" dataType="double" />
  <DataField name="glucose" displayName="Glycated hemoglobin level"
    otype="continuous" dataType="double" />
  <DataField name="cost" displayName="Treatment costs"
    otype="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>
</GeneralRegressionModel>
```
Example 1: Diabetes treatment costs model in PMML cont’d

```xml
<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>
```
Example 1: Selecting probability distributions based on data

<table>
<thead>
<tr>
<th>Input</th>
<th>Measuremen...</th>
<th>Fit to...</th>
<th>Distribution</th>
<th>Parameters</th>
<th>Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>Continuous</td>
<td>age</td>
<td>Triangular</td>
<td>max: 65, min: 13, mode: 53.99</td>
<td></td>
</tr>
<tr>
<td>glucose</td>
<td>Continuous</td>
<td>glucose</td>
<td>Lognormal</td>
<td>a: 7.55, b: 0.19</td>
<td></td>
</tr>
<tr>
<td>income</td>
<td>Continuous</td>
<td>income</td>
<td>Lognormal</td>
<td>a: 42430.23, b: 0.34</td>
<td></td>
</tr>
</tbody>
</table>
Example 1: Correlations tornado chart
Example 1: Probability density chart for target values computed based on the simulation
Example 2: Neural network diagram for bank default model

Hidden layer activation function: Hyperbolic tangent
Output layer activation function: Softmax
Example 2: Modeling bank loan defaults with a neural network

```xml
<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 (\times 100)" 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>
```
Example 2: Modeling bank loan defaults with a neural network

```xml
<TransformationDictionary>
  <DerivedField optype="categorical" dataType="double" name="defaultValue0">
    <NormDiscrete field="defaultValue" value="0" />
  </DerivedField>
  <DerivedField optype="categorical" dataType="double" name="defaultValue1">
    <NormDiscrete field="defaultValue" 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="20.55131" norm="8.67979989515938" />
    </NormContinuous>
  </DerivedField>
</TransformationDictionary>
```
Example 2: PMML for neural network model cont’d

```xml
<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 otype="continuous" dataType="double">
        <FieldRef field="debtincNorm" />
      </DerivedField>
    </NeuralInput>
    <NeuralInput id="1">
      <DerivedField otype="continuous" dataType="double">
        <FieldRef field="addressNorm" />
      </DerivedField>
    </NeuralInput>
    <NeuralInput id="2">
      <DerivedField otype="continuous" dataType="double">
        <FieldRef field="employNorm" />
      </DerivedField>
    </NeuralInput>
    <NeuralInput id="3">
      <!-- Additional Neural Inputs can be defined here -->
    </NeuralInput>
  </NeuralInputs>
</NeuralNetwork>
```
Example 2: PMML for neural network model cont’d

```xml
<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>
```
Example 2: PMML for neural network model cont’d

```xml
<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 otype="categorical" dtype="double">
            <FieldRef field="defaultValue0" />
        </DerivedField>
    </NeuralOutput>
    <NeuralOutput outputNeuron="6">
        <DerivedField otype="categorical" dtype="double">
            <FieldRef field="defaultValue1" />
        </DerivedField>
    </NeuralOutput>
</NeuralOutputs>
</NeuralNetwork>
```
Example 2: Bar chart for predicted target categories

Predicted Category

Percent

Predicted to Default

No

Yes
Example 2: Tornado chart
Example 3: Statistics 22 Automatic Linear Model for sales

- Using PMML 4.1
- Includes transformations for inputs:
  - Handling outliers using user-defined functions
  - Merging categories using MapValues element
- Model has two factors and three covariates
- Training data has only 60 cases, simulation is used to estimate overall sales expectations
Example 3: PMML for automatic linear model

<DataDictionary>
  <DataField dataType="double" display=None="sales" name="sales" optype="continuous"/>
  <DataField dataType="double" display=None="Case Count" name="caseCount" optype="continuous"/>
  <DataField dataType="double" display=None="Satisfaction level from customer survey" name="satisfaction_survey" optype="categorical">
    <Value displayValue="Strongly Negative" property="valid" value="1"/>
    <Value displayValue="Somewhat Negative" property="valid" value="2"/>
    <Value displayValue="Neutral" property="valid" value="3"/>
    <Value displayValue="Somewhat Positive" property="valid" value="4"/>
    <Value displayValue="Strongly Positive" property="valid" value="5"/>
  </DataField>
  <DataField dataType="double" display=None="Satisfaction level from social media data" name="satisfaction_social_media" optype="categorical">
    <Value displayValue="Strongly Negative" property="valid" value="1"/>
    <Value displayValue="Somewhat Negative" property="valid" value="2"/>
    <Value displayValue="Neutral" property="valid" value="3"/>
    <Value displayValue="Somewhat Positive" property="valid" value="4"/>
    <Value displayValue="Strongly Positive" property="valid" value="5"/>
  </DataField>
  <DataField dataType="double" display=None="advertising" name="advert" optype="continuous">
    <Interval closure="closedClosed" leftMargin="20686" rightMargin="82557"/>
  </DataField>
  <DataField dataType="double" display=None="cci" name="cci" optype="continuous">
    <Interval closure="closedClosed" leftMargin="26" rightMargin="116"/>
  </DataField>
  <DataField dataType="double" display=None="agents" name="agents" optype="continuous">
    <Interval closure="closedClosed" leftMargin="67" rightMargin="113"/>
  </DataField>
</DataDictionary>
Example 3: PMML for automatic linear model cont’d

```
<TransformationDictionary>
  <DefineFunction dataType="double" name="computeBounds" optype="continuous">
    <ParameterField name="RobustMean"/>
    <ParameterField name="RobustSD"/>
    <ParameterField name="CutOff"/>
    <ParameterField name="AnalysisWeight"/>
    <ParameterField name="Coeff"/>
    <Apply function="+">
      <FieldRef field="RobustMean"/>
      <Apply function="*"/>
      <FieldRef field="Coeff"/>
      <Apply function="*"/>
      <FieldRef field="CutOff"/>
      <Apply function="/"/>
      <FieldRef field="RobustSD"/>
      <Apply function="sqrt"/>
      <FieldRef field="AnalysisWeight"/>
    </Apply>
  </Apply>
</DefineFunction>
```

```
<DefineFunction dataType="double" name="treatOutlier" optype="continuous">
  <ParameterField name="Extreme"/>
  <ParameterField name="OUTLIER-TREATMENT-METHOD"/>
  <Apply function="if">
    <Apply function="equal">
      <FieldRef field="OUTLIER-TREATMENT-METHOD"/>
      <Constant>asExtremeValues</Constant>
    </Apply>
    <FieldRef field="Extreme"/>
  </Apply>
</DefineFunction>
```
Example 3: PMML for automatic linear model cont’d

```xml
<DefineFunction dataType="double" name="outlierHandle" optype="continuous">
  <ParameterField name="Var"/>
  <ParameterField name="OUTLIER-TREATMENT-METHOD"/>
  <ParameterField name="RobustMean"/>
  <ParameterField name="RobustSD"/>
  <ParameterField name="CutOff"/>
  <ParameterField name="AnalysisWeight"/>
  <Apply function="if">
    <Apply function="lessThan">
      <FieldRef field="Var"/>
      <Apply function="computeBounds">
        <FieldRef field="RobustMean"/>
        <FieldRef field="RobustSD"/>
        <FieldRef field="CutOff"/>
        <FieldRef field="AnalysisWeight"/>
        <Constant>-1</Constant>
      </Apply>
    </Apply>
  </Apply>
</DefineFunction>

<DerivedField dataType="double" display="advert" name="advert_transformed" optype="continuous">
  <Apply function="outlierHandle">
    <FieldRef field="advert"/>
    <Constant data="string">as Extreme Values</Constant>
    <Constant data="double">51231.7</Constant>
    <Constant data="double">17008.3588176869</Constant>
    <Constant data="double">3</Constant>
    <Constant data="double">1</Constant>
  </Apply>
</DerivedField>
```
Example 3: PMML for automatic linear model cont’d

```xml
<DerivedField dataType="double" displayName="Satisfaction level from social media data"
  name="satisfaction_social_media_transformed" optype="categorical">
  <MapValues outputColumn="output">
    <FieldColumnPair column="input" field="satisfaction_social_media"/>
  </MapValues>
  <InlineTable>
    <row>
      <input>1</input>
      <output>0</output>
    </row>
    <row>
      <input>2</input>
      <output>2</output>
    </row>
    <row>
      <input>3</input>
      <output>2</output>
    </row>
    <row>
      <input>4</input>
      <output>1</output>
    </row>
    <row>
      <input>5</input>
      <output>1</output>
    </row>
  </InlineTable>
</DerivedField>
</TransformationDictionary>
```
Example 3: PMML for automatic linear model cont’d

```xml
<ModelStats>
  <UnivariateStats field="sales">
    <Anova>
      <AnovaRow degreesOfFreedom="7" fValue="15.625917792178" meanOfSquares="152635084105.13" pValue="0.0000000000785363" sumOfSquares="1068445587392.9" type="Model"/>
      <AnovaRow degreesOfFreedom="52" meanOfSquares="97680716189.3675" sumOfSquares="5079397241847.11" type="Error"/>
      <AnovaRow degreesOfFreedom="59" sumOfSquares="15763853129240" type="Total"/>
    </Anova>
  </UnivariateStats>
  <UnivariateStats field="advert_transformed">
    <Counts totalFreq="60"/>
    <NumericInfo maximum="82557" mean="51231.7" minimum="20686"/>
  </UnivariateStats>
  <UnivariateStats field="cci_transformed">
    <Counts totalFreq="60"/>
    <NumericInfo maximum="116" mean="77.9666666666667" minimum="26"/>
  </UnivariateStats>
  <UnivariateStats field="agents_transformed">
    <Counts totalFreq="60"/>
    <NumericInfo maximum="113" mean="88.5833333333333" minimum="67"/>
  </UnivariateStats>
</ModelStats>
```

```xml
<ModelExplanation>
  <PredictiveModelQuality AIC="1525.71214930464" AICc="1528.5356787164" adj-r-squared="0.634406498794371" fStatistic="15.625917792178" numOfPredictors="5" numOfRecords="60" numOfRecordsWeighted="60" r-squared="0.67778199909655" sumSquaredError="5079397241847.11" sumSquaredRegression="15763853129240" targetField="sales">
    <LiftData/>
    <ModelLiftGraph/>
    <LiftGraph/>
  </PredictiveModelQuality>
</ModelExplanation>
```
Example 3: PMML for automatic linear model cont’d

- <ParameterList>
  - <Parameter label="Intercept" name="P0000001"/>
  - <Parameter label="[satisfaction_survey_transformed=0]" name="P0000002"/>
  - <Parameter label="[satisfaction_survey_transformed=1]" name="P0000003"/>
  - <Parameter label="[satisfaction_survey_transformed=2]" name="P0000004"/>
  - <Parameter label="[satisfaction_social_media_transformed=0]" name="P0000005"/>
  - <Parameter label="[satisfaction_social_media_transformed=1]" name="P0000006"/>
  - <Parameter label="[satisfaction_social_media_transformed=2]" name="P0000007"/>
  - <Parameter label="advert_transformed" name="P0000008"/>
  - <Parameter label="cci_transformed" name="P0000009"/>
  - <Parameter label="agents_transformed" name="P0000010"/>
</ParameterList>

- <FactorList>
  - <Predictor name="satisfaction_survey_transformed"/>
  - <Predictor name="satisfaction_social_media_transformed"/>
</FactorList>

- <CovariateList>
  - <Predictor name="advert_transformed"/>
  - <Predictor name="cci_transformed"/>
  - <Predictor name="agents_transformed"/>
</CovariateList>

- <PPMMatrix>
  - <PPCell parameterName="P0000002" predictorName="satisfaction_survey_transformed" value="0"/>
  - <PPCell parameterName="P0000003" predictorName="satisfaction_survey_transformed" value="1"/>
  - <PPCell parameterName="P0000004" predictorName="satisfaction_survey_transformed" value="2"/>
  - <PPCell parameterName="P0000005" predictorName="satisfaction_social_media_transformed" value="0"/>
  - <PPCell parameterName="P0000006" predictorName="satisfaction_social_media_transformed" value="1"/>
  - <PPCell parameterName="P0000007" predictorName="satisfaction_social_media_transformed" value="2"/>
  - <PPCell parameterName="P0000008" predictorName="advert_transformed" value="1"/>
  - <PPCell parameterName="P0000009" predictorName="cci_transformed" value="1"/>
  - <PPCell parameterName="P0000010" predictorName="agents_transformed" value="1"/>
</PPMMatrix>

- <ParamMatrix>
  - <PC beta="5393668 12533583" df="1" parameterName="P0000001"/>
  - <PC beta="589080 659345796" df="1" parameterName="P0000002"/>
Example 3: Probability density chart

<table>
<thead>
<tr>
<th>Sales Range</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; $6,358,626</td>
<td>5%</td>
</tr>
<tr>
<td>$6,358,626 - $7,734,351</td>
<td>90%</td>
</tr>
<tr>
<td>&gt; $7,734,351</td>
<td>5%</td>
</tr>
</tbody>
</table>
Example 3: Scatterplot of sales vs. satisfaction\_social\_media
Example 3: Scatterplot of sales vs. satisfaction_survey
Conclusions

- PMML can describe a wide range of statistical and data mining models and data transformations, enabling seamless portability of models between products.

- PMML is supported by a wide range of commercial products and open-source packages.

- In addition to model deployment, PMML can also be used for model evaluation and for simulation analysis.

- The use of PMML frees the simulation user from model details, allowing them to focus on the details of the simulation. PMML facilitates simulation analyses using different models built on the same data.
Questions?

Svetlana Levitan, PhD
Advisory Software Engineer
IBM SPSS Analytic Components
IBM Hybrid Cloud
IBM Representative in the Data Mining Group
PMML Release Manager for 4.3 and 4.4
slevitan@us.ibm.com