



Using PMML Models with InterSystems Products

Version 2019.4
2020-01-28

Using PMML Models with InterSystems Products

InterSystems IRIS Data Platform Version 2019.4 2020-01-28

Copyright © 2020 InterSystems Corporation

All rights reserved.

InterSystems, InterSystems IRIS, InterSystems Caché, InterSystems Ensemble, and InterSystems HealthShare are registered trademarks of InterSystems Corporation.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)

Tel: +1-617-621-0700

Tel: +44 (0) 844 854 2917

Email: support@InterSystems.com

Table of Contents

Using PMML Models with InterSystems Products	1
1 Background	1
2 InterSystems IRIS Support for PMML	1
2.1 Supported Models	1
2.2 The Iris Sample	2
3 Creating a Class to Contain PMML Models	2
3.1 Supported Data Dictionary Extensions	3
3.2 Generated Classes	3
4 Test Pages for Executing PMML Models	4
4.1 Sample Model Testing Page	4
4.2 Test Page for a Single Input Record	5
5 Executing PMML Models Programmatically	7
5.1 Executing a Model with a Single Input Record	7
5.2 Executing a Model with a Batch of Input Records	7
6 Options for Using PMML in InterSystems IRIS Business Intelligence	7
6.1 Calling the Model from a Pivot Table	8
6.2 Including PMML Predictions in an Analytics Listing	8
6.3 Exporting Batch Results to an Analytics Cube	9
7 Generating a Model Class from the Spark Connector	9

Using PMML Models with InterSystems Products

This article discusses how to use the InterSystems IRIS® data platform runtime support for PMML (Predictive Modelling Markup Language). It discusses the following topics:

- [Background on PMML](#)
- [How InterSystems IRIS supports PMML](#)
- [How to create a class that contains PMML models](#)
- [How to use the PMML test pages](#)
- [How to execute PMML models programmatically](#)
- [Options for using PMML within InterSystems IRIS Business Intelligence](#)
- [How to generate a PMML model class from the Spark Connector](#)

1 Background

PMML (Predictive Modelling Markup Language) is an XML-based standard that expresses analytics models. It provides a way for applications to define statistical and data mining models so that they can be easily reused and shared. The standard is particularly helpful because the analytics tools used to generate models (tools such as PMML:R, KNIME, SAS, and SPSS) are very different in architecture from the tools used in an InterSystems IRIS or production environment.

In a typical scenario, data scientists use an analytical tool to produce a data mining model based on large amounts of historical data, which is then exported to PMML. The model can then be deployed in a runtime environment and executed on incoming observations, predicting values for the model's target metrics.

For more information, see <http://www.dmg.org/>.

2 InterSystems IRIS Support for PMML

InterSystems IRIS provides runtime support for PMML [4.1](#) and [4.2](#) as follows:

- You can define a *PMML model class*, which is a subclass of `%DeepSee.PMML.Definition` that contains a PMML model. When that class is compiled, the system generates the code needed to execute the model or models described in it.
- InterSystems IRIS provides an API for executing the models, based on the data input that you provide.
- InterSystems IRIS provides a sample test page that uses the API.

2.1 Supported Models

InterSystems IRIS supports PMML [4.1](#) and [4.2](#) and the following PMML models:

- Naive Bayes models (<http://www.dmg.org/v4-1/NaiveBayes.html>)
- Regression models (<http://www.dmg.org/v4-1/Regression.html>)
- General regression models, except probit link function and contrast matrices (<http://www.dmg.org/v4-1/GeneralRegression.html>)
- Tree models, except nested models (<http://www.dmg.org/v4-1/TreeModel.html>)
- Text models, basic support (<http://www.dmg.org/v4-1/Text.html>)
- Support Vector Machine (SVM) models (<http://www.dmg.org/v4-1/SupportVectorMachine.html>)
- Clustering models (<http://www.dmg.org/v4-1/ClusteringModel.html>)
- RuleSet models (<http://www.dmg.org/v4-1/RuleSet.html>)
- NeuralNetwork models (<http://www.dmg.org/v4-1/NeuralNetwork.html>)

InterSystems IRIS also supports the `<MiningModel>` element, which provides “Model Segmentation” — the process of combining the output of multiple models for a more balanced prediction. See <http://www.dmg.org/v4-1/MultipleModels.html>. Note that InterSystems IRIS does not support the “Model Composition” approach, which is deprecated.

2.2 The Iris Sample

This article uses the Samples-Data-Mining sample (<https://github.com/intersystems/Samples-Data-Mining>). InterSystems recommends that you create a dedicated namespace called `SAMPLES` (for example) and load samples into that namespace. For the general process, see *Downloading Samples for Use with InterSystems IRIS*.

This sample includes a copy of the Iris data set, a well-known sample used in predictive analytics. The Iris data set provides measurements for the petal and sepal measurements for approximately 50 flowers in three different species of irises. These measurements are strongly predictive of the iris species.

Once you have set up the sample, you can use the PMML models in the `DataMining.PMML.Iris` class. This class contains a PMML definition that includes the following models:

- A tree model that predicts the iris species, based on petal and sepal measurements
- A general regression model that predicts the sepal length, based on the sepal width, petal measurements, and species

3 Creating a Class to Contain PMML Models

To create a class that contains PMML models:

1. Access the [PMML model testing page](#) (described later in this article).
2. Click **New**.
3. For **Class name**, type a fully qualified class name.
4. For **PMML file**, click **Browse** and select a PMML file..
5. Click **Import**.

Or use Atelier and create a subclass of `%DeepSee.PMML.Definition`. In this class, create an XData block named `PMML`, and paste a PMML definition into that XData block. For this XData block, set the `XMLNamespace` keyword as follows:

```
XMLNamespace = "http://www.intersystems.com/deepsee/pmml"
```

For an example, see the [sample class](#) `DataMining.PMML.Iris`. The following shows a partial extract:

```
Class DataMining.PMML.Iris Extends %DeepSee.PMML.Definition
{
XData PMML [ XMLNamespace = "http://www.intersystems.com/deepsee/pmml" ]
{
<PMML version="4.1">
<Header>
<Timestamp>03/07/2013 11:54:41</Timestamp>
</Header>
<DataDictionary numberOfFields="5">
<Extension name="isc:datasource">
<X-SQLDataSource name="Analysis dataset">
<X-FieldMap fieldName="PetalLength" spec="PetalLength" />
<X-FieldMap fieldName="PetalWidth" spec="PetalWidth" />
<X-FieldMap fieldName="SepalLength" spec="SepalLength" />
<X-FieldMap fieldName="SepalWidth" spec="SepalWidth" />
<X-FieldMap fieldName="Species" spec="Species" />
<X-SQL>SELECT PetalLength, PetalWidth, SepalLength, SepalWidth, UPPER(Species) Species
FROM DataMining.IrisDataset</X-SQL>
</X-SQLDataSource>
</Extension>
<DataField name="PetalLength" displayName="PetalLength" optype="continuous" dataType="double" />
<DataField name="PetalWidth" displayName="PetalWidth" optype="continuous" dataType="double" />
<DataField name="SepalLength" displayName="SepalLength" optype="continuous" dataType="double" />
<DataField name="SepalWidth" displayName="SepalWidth" optype="continuous" dataType="double" />
<DataField name="Species" displayName="Species" optype="categorical" dataType="string">
<Value value="IRIS-SETOSA" property="valid" />
<Value value="IRIS-VERSICOLOR" property="valid" />
<Value value="IRIS-VIRGINICA" property="valid" />
</DataField>
</DataDictionary>
...

```

For information on setting up and using this sample, see “[The Iris Sample](#),” earlier in this article.

Note that you should try to keep input field names below 50 characters, where possible. While longer field names are supported, providing values for long field names quickly gets confusing. Additionally, certain endpoints, such as the generated SQL stored procedure, only support arguments of limited lengths, and will truncate and add a unique integer to longer ones.

3.1 Supported Data Dictionary Extensions

InterSystems supports two kinds of `<Extension>` elements in the `<DataDictionary>` element:

- `<X-SQLDataSource>` defines a data source in terms of an SQL query. This element defines a mapping from the SQL fields to the data fields in the PMML definition.

For an example, see the [sample class](#) `DataMining.PMML.Iris`

- `<X-DeepSeeDataSource>` defines a mapping from the measures and dimensions of a given cube to the data fields in the PMML definition.

3.2 Generated Classes

When you compile your PMML class (`PackageName.ClassName`), the system generates following classes:

- `PackageName.ClassName.Data` represents the data needed to execute the models. This class corresponds to the `<DataDictionary>` element of the PMML definition.

- `PackageName.ClassName.ModelName` represents a model.

This class provides methods to execute the model.

- `PackageName.ClassName.ModelName.Input` represents the input for the given model.
- `PackageName.ClassName.ModelName.Output` represents the output for the given model.

InterSystems IRIS uses these classes to execute the model or models.

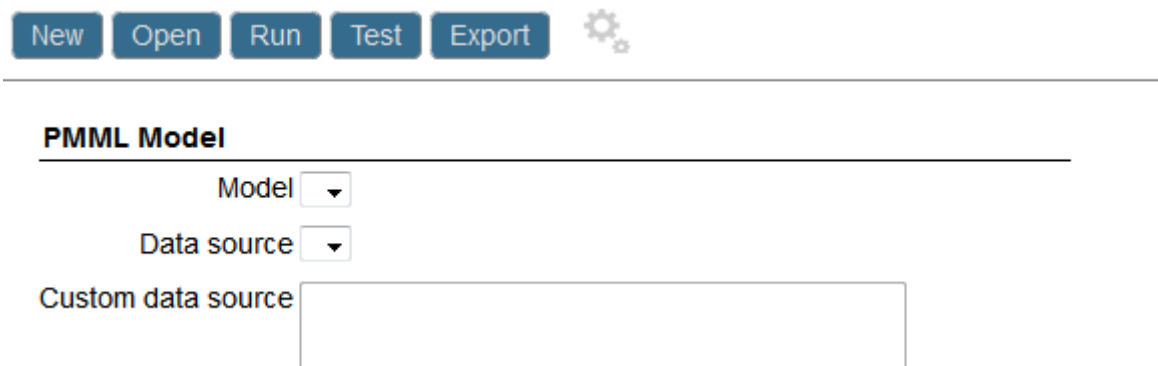
4 Test Pages for Executing PMML Models

InterSystems IRIS provides test pages that you can use to execute PMML models for [batches of records](#) or for [single input records](#). To access this pages:

1. Open the Management Portal.
2. Switch to the appropriate namespace.
3. Click **Analytics > Tools > PMML Model Tester**.

4.1 Sample Model Testing Page

The system then displays a page like the following (partially shown):



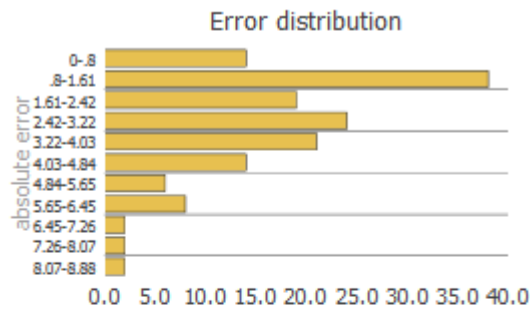
The screenshot shows a user interface for testing PMML models. At the top, there is a horizontal bar with five buttons: "New", "Open", "Run", "Test", and "Export", followed by a gear icon. Below this bar, the title "PMML Model" is displayed. Underneath the title, there are two dropdown menus: "Model" and "Data source". Below these dropdowns is a text input field labeled "Custom data source".

To use this page:

1. Click **Open**, select a model class, and then click **OK**.
2. Click a model from the **Model** drop-down list.
3. Click an option from the **Data source** drop-down list. Options include:
 - **Analysis dataset** — Selects the source data on which the model was developed.
 - **Cube mapping** — Selects the `<X-DeepSeeDataSource>` element in the model, if any. See “[Supported Data Dictionary Extensions](#),” earlier in this article.
 - **Custom data source (SQL)** — Enables you to enter an SQL query that provides a set of records.
4. If you selected **Custom data source (SQL)**, type an SQL SELECT query into **Custom data source**.
5. Click **Run**.

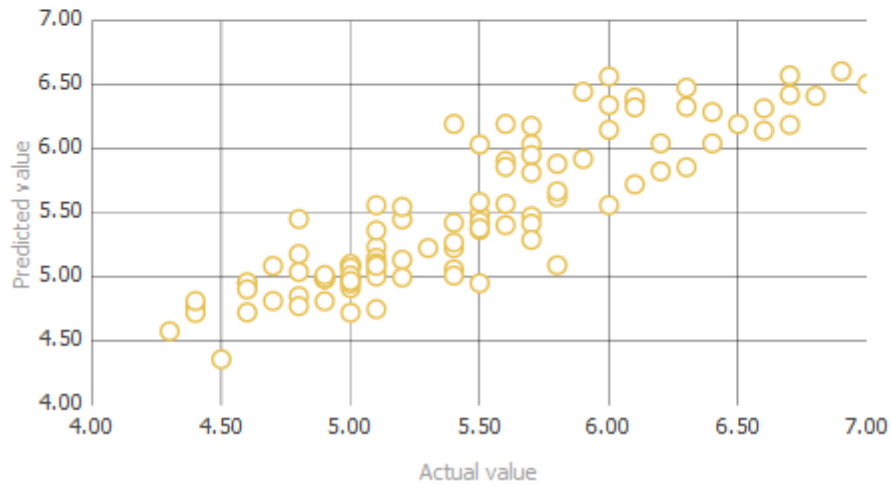
InterSystems IRIS iterates through the records and then displays a summary of the results. The details depend upon the model. The following shows an example:

Test results



Record count: 150
 Sigma: 0.30
 RSE: 0.1353
 R²: 0.8647

Value distribution



4.2 Test Page for a Single Input Record

You can also test the model with a single input record. To do so, press **Test**, which displays a dialog box like the following:

PMML Model test page

Data object

Species

Petal Length

Petal Width

Sepal Length

Sepal Width

Model

Predicted value:

Trace info:

The fields listed in **Data object** correspond to the data fields in your PMML definition.

To use this page, select a model from the **Model** drop-down list. The model determines which fields are input fields and which are output fields. Then enter values into the input fields. When you have entered all the input values, the page displays the predicted value for the output field for the given model. For example:

PMML Model test page

Data object

Species

Petal Length

Petal Width

Sepal Length

Sepal Width

Model

Predicted value:

IRIS-VERSICOLOR

predicted value: IRIS-VERSICOLOR
probability: 1

Trace info:

Rank	Info
1	Tree node 1: (SepalWidth greaterThan "3.1450000000000004618")
2	Tree node 3: (PetalWidth greaterThan "1.3349816849816849018")
3	Tree node 7: (SepalLength lessThan "6.5753846153846158273")
4	Tree node 13: (PetalLength lessThan "5.0525000000000002131")

5 Executing PMML Models Programmatically

InterSystems IRIS also provides an API that you can use to execute PMML models.

5.1 Executing a Model with a Single Input Record

To run a predictive model for a single record:

1. Create an instance of the generated class *PackageName . ClassName . ModelName*. This class defines methods you can use to execute the model.
2. Create an instance of the generated class *PackageName . ClassName . Data* and set its properties. The purpose of this instance is to contain the input values.
3. Invoke the **%ExecuteModel()** method of the model instance.

```
Method %ExecuteModel(ByRef pData As %DeepSee.PMML.Data,
                    Output pOutput As %DeepSee.PMML.ModelOutput) as %Status
```

For *pData*, use the data object that you created in step 2.

This method returns, as an output argument, an instance of `%DeepSee.PMML.ModelOutput` that contains the output of the model. Specifically, this is an instance of the generated class *PackageName . ClassName . ModelName . Output* for the given model.

4. To see the details for the output, use `ZWRITE`. The *pOutput* object includes one property for each `<OutputField>` in the `<Output>` element of the model definition. If there is no `<Output>` element, *pOutput* includes a single field named after the predicted `<MiningField>` element.

Or, if you specified `<X-DeepSeeDataSource>` in your PMML definition, use **%ExecuteModelDeepSee()**. See the class reference.

5.2 Executing a Model with a Batch of Input Records

To run a predictive model with a batch of input records, use the **%RunModel()**, **%RunModelFromResultSet**, or **%RunModelFromSQL** methods in `%DeepSee.PMML.Utils`. These methods store the resulting predictions in the `%DeepSee_PMML_Utils.TempResult` table.

6 Options for Using PMML in InterSystems IRIS Business Intelligence

This section discusses options for using PMML with InterSystems IRIS Business Intelligence:

- [Calling the model from an Analytics pivot table](#)
- [Including predictions in a detail listing](#)
- [Exporting the results to a cube](#)

For background information, see *Defining Models for InterSystems Business Intelligence*.

6.1 Calling the Model from a Pivot Table

You can invoke a PMML model from within a pivot table. To do so, define a calculated member that uses the %KPI function to invoke the %DeepSee.PlugIn.PMML plug-in. Use the following syntax:

```
%KPI( "%DeepSee.PMML", fieldName, series, "PMML", modelName, parmName1, parmValue1, parmName2, parmValue2, "%CONTEXT" )
```

Where:

- *fieldName* is the quoted name of an output field of an InterSystems IRIS PMML model class.
- *series* is the optional number of a series (row) in the plug-in. Specify 1 or omit this argument.
- *modelName* is the quoted name of an InterSystems IRIS PMML model class.
- *parmName1*, *parmName2*, and so on are optional quoted names of parameters of the %DeepSee.PlugIn.PMML plug-in. Note that parameter names are case-sensitive. The %DeepSee.PlugIn.PMML plug-in provides the following parameters:
 - "%cube" — Specifies the cube on which this KPI is executed.
 - "modelName" — Specifies the name of the model to execute. If specified, this must be a model in the given model class. If left blank, the first model in the class will be executed.
 - "aggregate" — Specifies how to aggregate the predicted values over different facts. The available values are "average" (the default), "sum", "max", "min", "maxFreqValue", "maxProbValue", "minFreqValue" and "minProbValue".

Note that not all aggregations might make sense for each output field.

 - "NullValues" — Specifies whether or not to include null predictions when aggregating results. Available values are "ignore" (the default) and "count".

The order in which you list the parameters does not affect the results.

You can specify up to 16 parameters and their values.

- *parmValue1*, *parmValue2*, and so on are the corresponding values of the named filters.

The special %CONTEXT parameter to cause the plug-in to consider the context of query, which is otherwise ignored. For details, see the reference for the %KPI function in the *InterSystems MDX Reference*.

For example, use the following syntax to get the average value for the output field *MyField* for a PMML model class named *Test.MyModel*, which contains only a single model:

```
%KPI( "%DeepSee.PMML", "MyField", , "PMML", "Test.MyModel", "aggregate", "average", "%CONTEXT" )
```

6.2 Including PMML Predictions in an Analytics Listing

To include record-level predictions in a detail listing, you can use the \$\$\$PMML token in the listing query. This token takes the PMML definition class name and the model name as its primary parameters. As an optional third argument, you can pass the name of the predicted feature you wish to include in the query (this argument defaults to "predictedValue").

The following shows the definition of a listing query that uses this token:

```
UserID, TotalWagered, PercentLost "Lost %" , $$$PMML[MyPMML.Poker,PercentLost] "Predicted Loss %"
```

6.3 Exporting Batch Results to an Analytics Cube

After you run a predictive model with a batch of input records, you can export the results to a cube. This option enables you to visualize the results in a different way. The cube contains two levels: `ActualValue` and `PredictedValue`.

To export the results to a cube, use the [PMML test page](#) and click **Export**. InterSystems IRIS prompts you for the following information:

- **Export** — Select **data only** or **classes and data**
- **Result class name** — Specify the name of the persistent class to which the results are written. This is used as the source class for the cube.
- **Link to source class** — Specify the class that contains the source records. The result class includes a property named `Record` that points to this class.
- **Drop data** — Select this if you want to empty the result class (**Result class name**) before performing the export. Or clear this if you want to append the newly exported data to the end of the result class table.
- **Cube name** — Specify the logical name of the cube.
- **Cube class name** — Specify the class name of the cube.
- **Overwrite classes** — Select this if you have performed this export earlier and now want to overwrite the classes with new data and definitions.

The system then displays the **Build Cube** dialog box, where you can build the given cube. Click either **Build** or **Cancel**. You can also later access this cube via the Architect and build it there.

After you build the cube, use the Analyzer to examine it. The following shows an example. The `ActualValue` level is used as rows and the `PredictedValue` levels is used as columns:

Actual Value	IRIS-SETOSA	IRIS-VERSICOLOR	IRIS-VIRGINICA
IRIS-SETOSA	50		
IRIS-VERSICOLOR	3	45	2
IRIS-VIRGINICA		6	44

7 Generating a Model Class from the Spark Connector

Also, you can also generate a PMML model class from a Spark pipeline model (`org.apache.spark.ml.PipelineModel`). To do this, call the `iscSave()` extension method provided by the InterSystems Spark Connector.

To do this

1. Make sure to add the following dependency when starting your Spark shell/master:

```
Spark 2.1.x: org.jpmmml:jpmmml-sparkml:1.2.7
Spark 2.2.x: org.jpmmml:jpmmml-sparkml:1.3.3
```

Note that Spark ships with an older version of two PMML JAR files than JPMML needs, so you must remove these JAR files from the `$SPARK_HOME/jars/` directory to avoid conflicts. For details, see <https://github.com/jpmmml/jpmmml-sparkml#library>.

2. Run the following commands in the Spark shell:

```
import org.apache.spark.ml.{Pipeline, PipelineModel}
import org.apache.spark.ml.feature.RFormula
import org.apache.spark.ml.classification.LogisticRegression
import org.apache.spark.ml.linalg.Vector
import org.apache.spark.sql.Row

val training = spark.createDataFrame(Seq((1.0, 0.0, 1.1, 0.1),(0.0, 2.0, 1.0, -1.0),(0.0, 2.0, 1.3,
  1.0),(1.0, 0.0, 1.2, -0.5))).toDF("label","feature1", "feature2", "feature3")
val formula = new RFormula().setFormula("label ~ feature1 + feature2 + feature3")
val lr = new LogisticRegression().setMaxIter(10).setRegParam(0.001)
val pipeline = new Pipeline().setStages(Array(formula, lr))
val model = pipeline.fit(training)

import com.intersystems.spark._
import com.intersystems.spark.ml._
model.iscSave("abc.TestMe", training.schema)
```

Note that only the last three lines are specific to generating a PMML model class.

3. Now you will see a PMML model class named `abc.TestMe` class in the IRIS master namespace. You can use this model as described earlier in this topic.

For more information on the InterSystems Spark Connector, see *Using the InterSystems Spark Connector*.