



**UNIVERSITE PARIS 13
INSTITUT GALILEE**



The PMML Interpreter

INTERNSHIP REPORT

**Sujeevan ASEERVATHAM,
DESS Exploration Informatique des Données.**

**Under the direction of:
Erik MARCADÉ, KXEN
Younes BENNANI, Université Paris 13**

April 2002 - September 2002

ACKNOWLEDGMENTS

I wish to express my gratitude to Erik Marcadé, the KXEN's CTO, for his support during the training period. Special thanks go to Bertrand Lamy who assisted me in the project's development and from who I learned much on computer engineering. I am also particularly thankful to Benoît Rognier who always was willing to help, and to explain things, on whatever subject (especially on Linux).

I also thank Fatima Adouhane, Alain Charroux, Redouane Mahrach and Victor Coustenoble for their helps and their kindnesses.

I also greet former, current and future trainees, especially Youssef Laghzali and Marie Berlioz.

ABSTRACT

This report contains information about the traineeship of 6 months that I did at KXEN during the last year of my computer engineering studies. The new emerging standard to define data mining models is described in this report as well as some background information on data mining. Moreover, a full description on the project I have had to develop is given.

The objective of this training was to study the new standard for the definition of data mining models. This standard, called The Predictive Model Markup Language (PMML) V2.0, is intended to allow the models' exportation from one application to another, opening then a new era: the era of cooperation between data mining applications such as IBM's Intelligent Miner, KXEN Analytic Framework, SAS, Clementine, SPSS and much others.

The study was to allow the development of a program able to validate and execute the PMML models. The program was intended to be given to the data mining community as an open source program in response to miscellaneous needs.

SUMMARY

The present report deals with my training period at KXEN from April the 1st, 2002 to September the 30th, 2002. An overview of the data mining is given as well as information about the Predictive Model Markup Language.

KXEN is an American company of about 35 people, with the research and development department located in France, near Paris. The company is specialized in the development of data mining software components to be embedded in applications (usually in statistical applications). KXEN is well known for the quality of the components. Indeed, KXEN is partner of companies such as BusinessObjects, IBM, SPSS...

Data Mining is a science that has emerged further to the request of companies wanting to obtain quickly knowledge from a large volume of data. Data mining is used to find hidden patterns in a dataset by discovering relationships among the data. Those patterns can then be used to help in decision-making, usually in Customer Relationship Management.

The data mining community is led by the Data Mining Group (DMG) that is composed of the most active companies in this science. One of the DMG's main objectives is to develop the cooperation between data mining applications. That means, in a first step, to make the data mining models interchangeable, one application can then read and execute models generated by another application. Further to this idea, the DMG has developed the Predictive Model Markup Language (PMML) that is an XML based language, which allows data mining application to export most of their models to other applications. Currently the latest version: PMML V2.0 defines 8 kinds of models.

KXEN, strongly believing that PMML can be beneficial to the community, has decided to implement the PMML V2.0 standard and to provide a free tool, the "PMML Interpreter", that will be able not only to read and validate PMML 2.0 models but also to execute such models on datasets. The PMML Interpreter must be able to check the models' compliancy to ensure that applications will release valid models in order to keep the standard free of any corruptions. Moreover, it must be able to execute models in order to convince the data mining application's editors to turn their models into the PMML 2.0 standard. Indeed, software editors can support the PMML at no cost because KXEN provides a free PMML interpreter that can be embedded into any application.

In the R&D department of KXEN, I was in charge of the PMML Interpreter's project. I mainly worked on the translation of the PMML 2.0 specification into C++ program.



TABLE OF CONTENTS

- INTRODUCTION 6**
- I. The presentations..... 7**
 - I.1. The KXEN company.....7**
 - 1.1. Company overview 7
 - 1.2. The team..... 7
 - 1.2.1. The management team 7
 - 1.2.2. The Research and Development team 7
 - 1.2.3. The Administration and Sales team 8
 - 1.2.4. The Scientific team..... 8
 - 1.3. KXEN Analytic Framework..... 9
 - 1.4. The Business Model..... 10
 - 1.5. Examples of applications of the KXEN components..... 10
 - I.2. Data Mining..... 10**
 - 2.1. What Is Data Mining? 10
 - 2.2. The applications of Data Mining..... 11
 - 2.3. The branches of industry of Data Mining. 12
 - I.3. The Training..... 12**
 - 3.1. The subject 12
 - 3.2. The Predictive Model Markup Language (PMML) 12
 - 3.3. The aim..... 14
 - 3.4. The work environment 15
- II. PMML V2.0 specification 16**
 - II.1. The header 16**
 - II.2. The Data Dictionary..... 16**
 - II.3. The Transformation Dictionary..... 16**
 - 3.1. Constant..... 17
 - 3.2. FieldRef..... 17
 - 3.3. NormContinuous 17
 - 3.4. NormDiscrete 18
 - 3.5. Discretize..... 18
 - 3.6. MapValues 18
 - 3.7. Aggregate 18
 - II.4. The Mining schema..... 19**
 - II.5. The Statistics..... 19**
 - 5.1. Counts..... 19
 - 5.2. NumericInfo 19
 - 5.3. DiscrStats 19
 - 5.4. ContStats 19
 - 5.5. Partition 20
 - II.6. The Regression model..... 20**
 - II.7. The Clustering model..... 21**



7.1. The Center-Based clustering	21
7.2. The Distribution-Based clustering	23
II.8. The Neural Network model	28
III. The PMML Interpreter.....	30
III.1. The architecture.....	30
1.1. The Basis library	30
1.2. The XMLEventParser library	30
1.3. The StateMachine library	31
1.4. The PmmlTree library	32
1.5. The IOConnector library	33
1.6. The TreeBuilder library.....	34
1.7. The Data flows	34
III.2. The encountered problems	35
2.1. The Data types in PMML.....	35
2.2. The Missing, invalid and outlier values	35
2.3. The Distribution based clustering.	36
2.4. The PMML V2.0 stability.	38
III.3. The results	39
CONCLUSION.....	41
REFERENCES	42
APPENDICES.....	43
Appendix 1: An example of PMML V1.1 File.	44
Appendix 2: An example of PMML V2.0 File.	45
Appendix 3: First Progress report, April the 8 th , 2002.	46
Appendix 4: Nominal to Numeric encoding proposal.....	51
GLOSSARY	54
INDEX.....	56

INTRODUCTION

This is the report of my training period at the KXEN's Research and Development department, France. A training period is an obligated part of the graduation program of the computer science department at the Université de Paris 13, France. Currently, I'm a fifth year student and from now I'll be working on my traineeship, which is about data mining and software development. The training period is from April the 1st, 2002 to September the 30th, 2002.

I've joined the R&D department on April the 2nd, 2002 (01/04 is public holiday in France) and after a week of presentation, I was given the PMML Interpreter's project. The project was to develop a program that will be able to read, validate and execute data mining models under the Predictive Model Markup Language (PMML) format. PMML was developed by the Data Mining Group to describe models generated by data mining applications.

Thus, my mission was to propose a specification for the project and once validated by my supervisor, Erik Marcadé, I was in charge of the project's achievement.

This document deals with the work I realized during the traineeship. In the first step, we will make a presentation of the company and the mission that I was given, for background information, then, in the next step, we'll describe the PMML 2.0 specification to then present the work realized on the project and finally we'll finish by a short conclusion about the training.

I. The presentations

I.1. The KXEN company.

1.1. Company overview

KXEN (Knowledge Extraction Engines) is a global analytic software company that provides advanced analytics to be embedded in existing enterprise applications and business processes. KXEN makes cutting-edge data mining technology available to both business decision makers and data mining professionals. KXEN provides them with more accurate, timely and actionable information. Customers will have the ability to understand, predict, manage and influence through a better knowledge of the information contained in their corporate data.

The KXEN Analytic Framework is distributed through partnerships with leading system integrators, application vendors, and OEMs.

Founded in 1998 by Roger Haddad, Erik Marcadé and Michel Bera, KXEN is a privately held company headquartered in California with research and development facilities in France. Offices are located across the USA and Europe.

After Receiving an initial round of seed financing of \$500,000 in March 1999, KXEN secured a Preferred Series B round of financing of \$5.5 million in June 2000 and lately extended that round for an additional \$2million from a core group of well-known investors including Sofinnova France, Sofinnova US, and Innovacom.

At the date of August 31, 2002, KXEN has 35 employees worldwide, excluding contractors. Its offices are located across the USA (San Francisco, Chicago, New York and Charlotte), as well as in Europe (Paris, France; Peterborough, England; and Zurich, Switzerland).

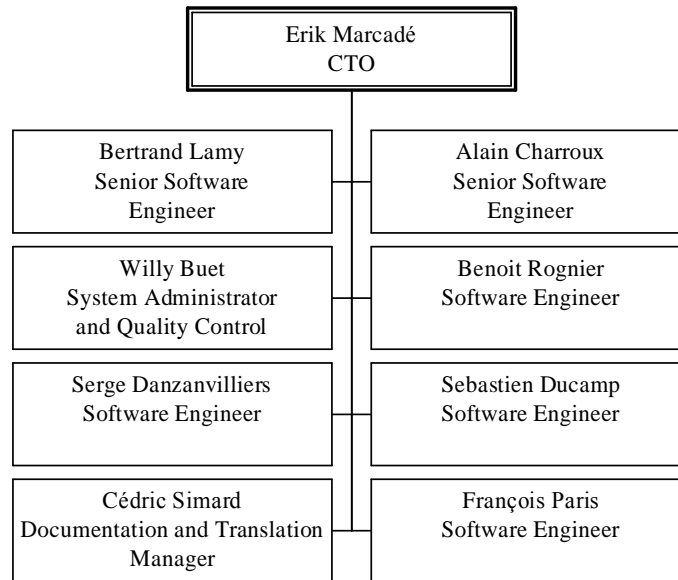
1.2. The team

1.2.1. The management team

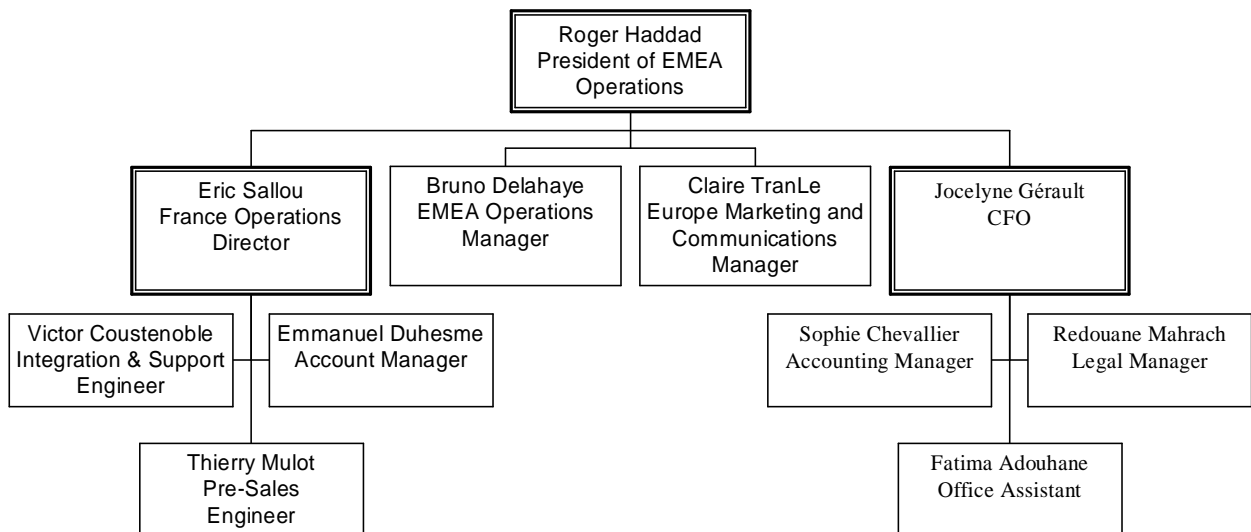
The management team is composed of the three company's founders:

- **Roger Haddad**: Chief Executive Officer
- **Erik Marcadé**: Chief Technical Officer
- **Michel Bera**: Chief Scientific Officer

1.2.2. The Research and Development team



1.2.3. The Administration and Sales team



1.2.4. The Scientific team

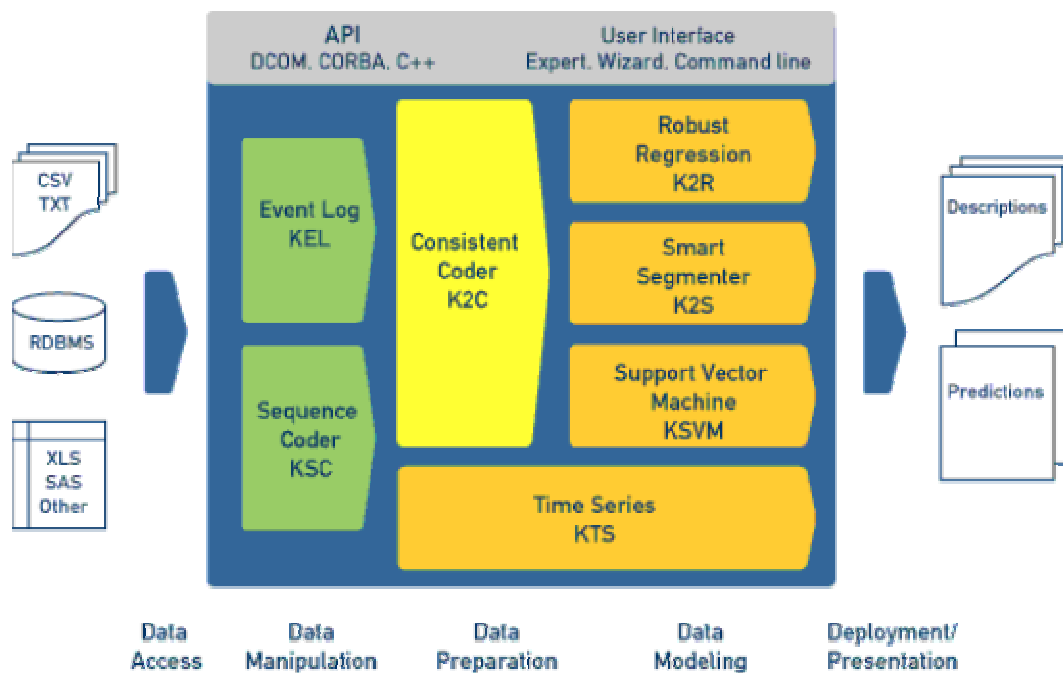
Chaired by Michel Bera, the scientific comity groups several experts in France and in the US:

- Leon Bottou
- Olivier Chapelle
- Lee Giles
- Yann LeCun
- Philippe Lelong
- Gregory Piatetsky-Shapiro
- Gilbert Saporta
- Emmanuel Viennet

1.3. KXEN Analytic Framework.

KXEN Analytic Framework is composed of eight components that can be separated into three groups:

- The first group is composed of components used to transform and to encode data before use. The automation of this treatment is one of the main interests of KXEN. It allows to save a lot of time and to treat the same way very different files (historic data, log files, time series...).
- The second one contains the modelling tools: the objective of such components is to build the best model: the one that is the most suitable for data and that gives the best score. There still, the user has several possibilities: robust regression, support vector machine, segmentation...
- The last one contains only one component, which allows the user to apply the model. It creates C code that is able to give the same results than the initial model.



Component	Function
Data preparation	
<i>K2C</i> (KXEN Consistent Coder)	Prepares data: encodes nominal and ordinal variables, automatically fills in missing values and detects out of range data
<i>KEL</i> (KXEN Event Log)	Aggregates events into periods of time: allows integrating transactional data with demographic customer data
<i>KSC</i> (KXEN Sequence Coder)	Aggregates events into a series of transitions (for example a customer click-stream from a Web site can be transformed into a series of data for each session)

<i>KTS</i> (KXEN Time Series)	Predicts meaningful patterns and trends in your data over time
<i>Modelling</i>	
<i>K2R</i> (KXEN Robust Regression)	Uses a proprietary regression algorithm to build predictive and descriptive models
<i>KSVM</i> (KXEN Support Vector Machine)	Is a binary classification component, particularly well suited for analysing data sets with a small number of observations but with a high number of variables
<i>K2S</i> (KXEN Smart Segmenter)	Discovers natural groupings or clusters in a set of data
<i>Generation of C code</i>	
<i>KCG</i> (KXEN Code Generator)	Generates C or XML code corresponding to the model built with the KXEN Analytic Framework

1.4. The Business Model.

The business model of KXEN is based on a policy of indirect sale via partners. Taking into account the specificity of its offer (software components), KXEN is distributed through partnerships with leading system integrators, application vendors and original equipment manufacturers (OEM's).

These various partners associate their trade competence with the new prospects brought by the KXEN components.

1.5. Examples of applications of the KXEN components.

- Predict which customers will buy a given product, and the reasons for it.
- Improve the risk analyse for credits, loans or mortgages.
- Find unexploited market segments and possible sources of profit.
- Discover influent factors on profit and productivity.
- Find customers about to leave.
- Predict the road traffics the TV audience.
- Predict energy consummation in order to optimise production.

I.2. Data Mining.

During the few last years, the quantity of data to work with has highly increased. Databases have now become an essential solution to store such quantities of data. However, as the volume increase, it becomes more and more harder to analyse and interpret the data. Moreover, Companies do not need data but information. It is then become a need to get information from large quantities of data.

2.1. What Is Data Mining?

The data mining (KDD: "Knowledge Data Discovery") is the solution to get, quickly, the essential information from data. One definition could be the search for valuable information in larges volumes of data.

The data mining is at the crossroad of databases, statistics, symbolic learning and artificial intelligence. The goal is to build models that describe the relation between

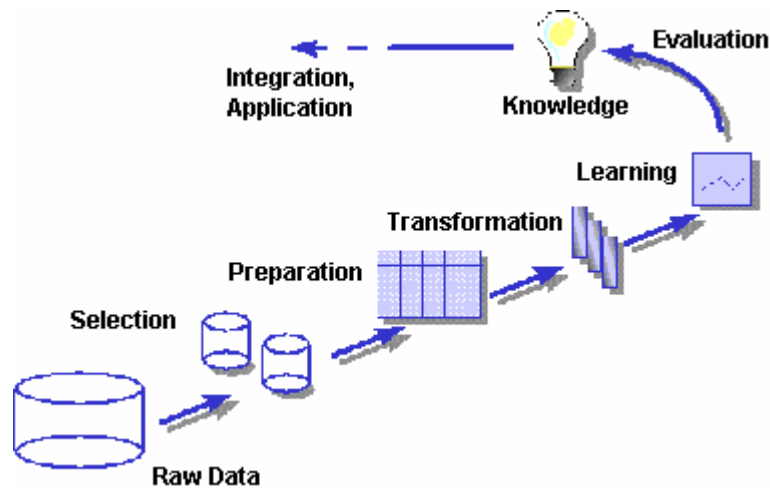
I. The presentations



descriptive variables (variables that describe or explain a situation) and target variables (variable that are the results of the situation). Such models can be used:

- To describe the most important variables that are responsible of the results, we can then for example classify data.
- To predict the result according to a situation.

The schema below describes the usual way the data mining is used:



With Data Mining, it is now possible to answer to questions like “which product are likely to be bought by people according to their sex and address?” from observation data like:

- “Males who live in Paris, buy the product B”
- “Females who live in Paris, buy the product A”
- ...

In this example descriptive variables are *sex*, *address* and the target variable is *product*.

2.2. The applications of Data Mining.

The data mining can be used for several purposes:

- To better target commercial efforts.
- To improve quality of the services.
- To detect fraudulent behaviors.
- To analyze technical data.
- ...

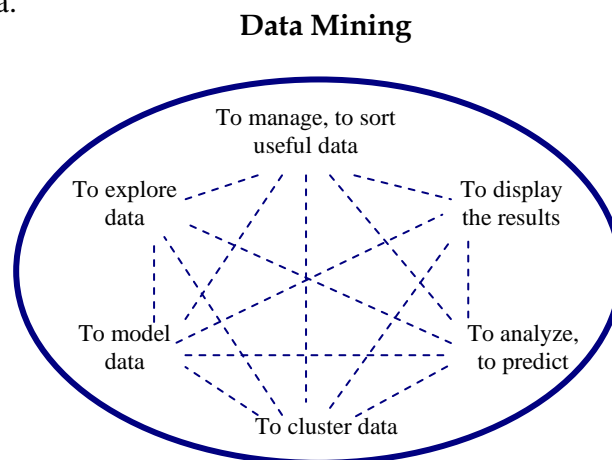
DataBases :

- Observation
- ...



Required Results:

- To improve quality
- To target customers
- ...



2.3. The branches of industry of Data Mining.

The use of data mining is numerous:

- Medicine and genetics.
- Astronomy.
- Industrial processes.
- Agriculture.
- Customer Relationship Management (CRM).
- ...

Data mining is especially used in activities that need a better knowledge of the customers to improve the sells.

I.3. The Training

The training at KXEN proceeds in the Research and Development department from April the first, 2002 to October the first, 2002. It is intended to provide a good experience of software engineering in a data mining state of the art company.

3.1. The subject

The subject of this training is to develop a program that must be able to read and interpret the **Predictive Model Markup Language (PMML)**. The program must respect the following constraints:

- In order to achieve good performance, it must be written in C++.
- It should be able to be executed on a wide variety of platform, that means it must be able to be compiled by several compilers, even by those that can, nowadays, be considered as obsolete.
- It should easily be integrated in a software environment, as a small component.
- It should be able to validate PMML files by indicating if the file is PMML compliant or not. In case if the file is not compliant, the program must provide a human-readable error message, indicating what's wrong with the file; the error message must be clear enough to quickly locate and correct the error in the file.
- It is intended to be distributed as free software so it must be well commented to allow several people to deal with the software's maintenance and evolution.

3.2. The Predictive Model Markup Language (PMML)

The Predictive Model Markup Language is an XML (eXtensible Markup Language) based language to describe statistical and data mining models (Appendices 1&2).

XML is a set of conventions used to structure data under a text file form that is both human-readable and machine-readable. It just looks like HTML (Hyper-Text Markup Language).

PMML is nothing else than XML but with its own conventions. The conventions are defined by a DTD (Document Type Definition) and by an XML schema but they essentially rely on the specification provided in English.

I. The presentations

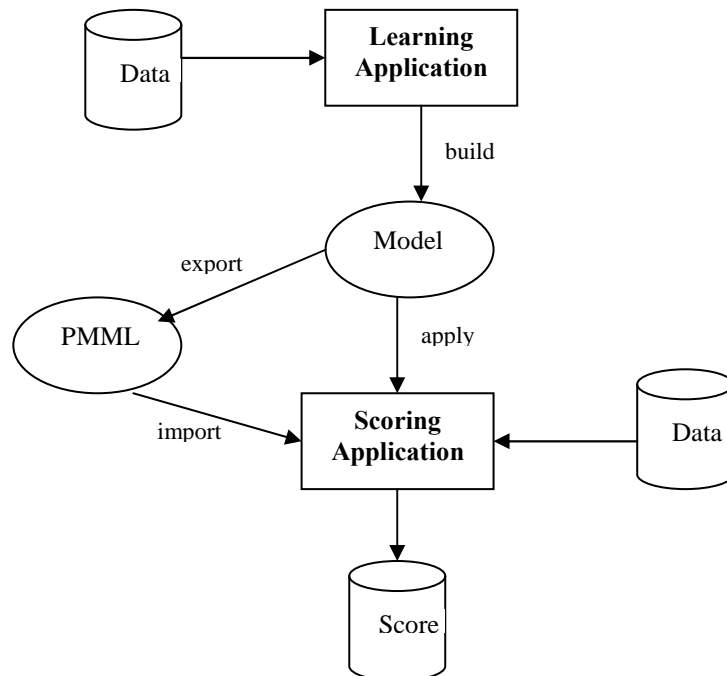


PMML was defined by the DMG (Data Mining Group) that includes the following companies:

- Angoss Software Corp.
- IBM Corp.
- KXEN.
- NCR Corp.
- Magnify Inc.
- Microsoft.
- MINEit Software Ltd.
- Oracle Corp.
- Quadstone.
- National Center for Data Mining.
- SAS.
- SPSS Inc.
- Xchange Inc.

Actually, Robert Grossman is the person in charge of PMML maintenance and evolution.

PMML provides a quick and easy way for companies to define and share predictive models. A PMML document is a definition of fully trained analytic models with sufficient information for an application to build and apply the models on a data set. With PMML, it is now possible to exchange Data Mining models between various applications, no matter about the application that generated the model.



PMML V1.1 was released in August 2000. It provides specification to define six kinds of models:

- Tree Models
- Neural Network models
- Clustering models.

I. The presentations



- Regression models.
- General Regression models.
- Association models.

Nowadays, the following softwares are known to be able to generate PMML V1.1 compliant models:

- IBM's Intelligent Miner for Data.
- KXEN components.
- Dialogis's D-Miner (formerly known as Kepler).

And the following softwares are able to interpret PMML V1.1 models:

- IBM's IM Scoring .
- Dialogis's D-Miner.

PMML V2.0 was released in August 2001. It introduces data transformations so PMML can do data preprocessing before applying models. The supported transformations are the following:

- Constant (the result is always a constant)
- Continuous Normalization by piecewise linear interpolation.
- Discrete Normalization of nominal values to discrete numeric values.
- Discretization of continuous values to nominal value.
- Values mapping.
- Aggregate function like sum, average... of a variable.

It also supports two more kinds of models:

- Naïve Bayes models.
- Sequence models.

Nowadays, only KXEN Analytic Frameworks is known to generate PMML V2.0 models.

3.3. The aim

The goal of PMML is to allow the definition of predictive models while ensuring that they will be independent of the application that has produced them. In fact, as PMML introduces the ability of defining vendor specific conventions (tags), sharing models is not as simple as it seems. Indeed, applications abusively use such conventions to describe their models. As a result, the information contained in the PMML core specification is not enough to build consistent models from such files. Then it often happens that a client cannot read a PMML file generated by one producer, however they both pretend to be PMML compliant. To face such problems, KXEN decided to provide, to the data mining community, a program that will help not only the producers to generate PMML compliant models but also the applications to read and interpret PMML models.

The primary aim is to allow the PMML producers to be sure that their files are PMML 2.0 compliant. In other words, if our program validates the file then it is 100% PMML 2.0 compliant, that means that any PMML clients that cannot read such files should be considered as PMML 2.0 non-compliant.

Moreover, it will give also the DMG a tool to validate that proposals of PMML extension can correctly be executed.

I. The presentations



The usage of this program can be beneficial to the community. Indeed, the guarantee that all the files available on the web are compliant, will allow giving a unique direction to the PMML's evolution.

The secondary aim is also to provide an open source interpreter that can apply PMML models on various kinds of data sources (files, databases...).

3.4. The work environment

The development team for this project is composed of only one member: the trainee supervised by Erik Marcadé, the KXEN CTO.

The materials available for the development are:

- A PC computer with Window 2000.
- Emacs for writing the program.
- "cl", the visual C++ compiler.
- Cygwin bash shell for the command line compilation.
- Doxygen for generating the project's documentation from the program's comment.
- CVS to share the program source code.

II. PMML V2.0 specification

A well-structured document regarding the XML's specification doesn't mean that the document is PMML compliant, the document must also be conform to the specification provided by the Data Mining Group.

This specification is essentially provided in English because XML's DTD or schema can only express the way that data should be structured to provide a valid PMML document.

So the biggest difficulty in this project is to understand and translate the specification in C++ code. Moreover, the task is made harder because of the fact that there are, at this time, no PMML V2.0 files since the norm is not yet supported.

In this section we will only discuss about the specification that has been studied and implemented. Indeed, PMML defines too many data-mining model to be implemented within the deadline.

II.1. The header

The header of a PMML document contains only background information about the application that generated the document. It contains a field to describe the copyright for the model, the name and the version of the application that produced the model, the date of creation and some additional annotations about the model.

Well, the header is not important but it may be interesting to store the information in case a user wants to have some background information.

II.2. The Data Dictionary

The data dictionary is a field where, at least, all the variables required to apply the model must be defined. That means the name of the variable must be given together with the values that the variable can take.

PMML defines three kinds of variables:

- Continuous variable: for those variables, either an interval of valid values or an enumeration of (valid or invalid) values must be defined.
- Ordinal variable: in PMML an ordinal variable can take any values in an ordered set of nominal values, this set of values must be defined.
- Categorical variable: those variables can take any defined nominal values.

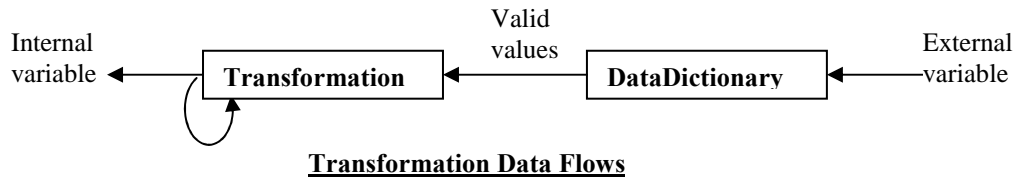
The data dictionary is used to check if a value taken by a variable, coming from the external environment, is valid or invalid.

II.3. The Transformation Dictionary

The transformation dictionary allows the definition of internal variables that are constructed by the mathematical transformations of external variables (those defined in the data dictionary). There are seven kinds of transformation (detailed below).

This field was introduced in PMML2.0, allowing then capturing some of the data pre-processing required to apply a model.

It is important to note that the output of a transformation can also be the input of another transformation.



3.1. Constant

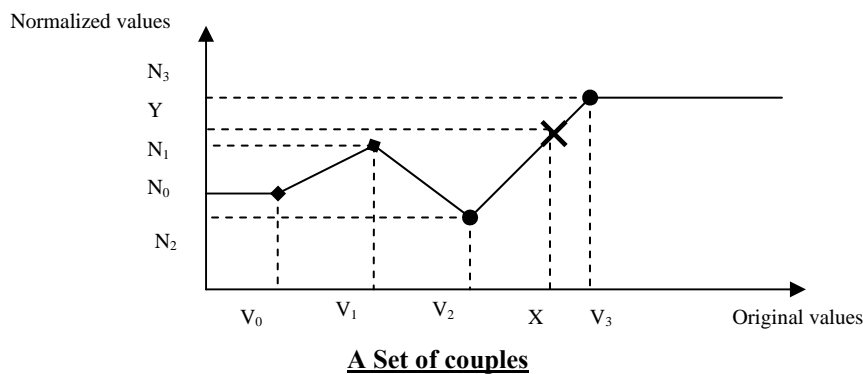
The constant transformation is the simplest one, which defines variables that always take the same defined value: a constant.

3.2. FieldRef

The “FieldRef” transformation is just used to rename a variable. The new variable is only a reference to the input variable. It was introduced to allow inline transformations (transformations that are defined outside the transformation dictionary, usually inside a model, emanating from PMML 1.1) to use a transformation already defined in the transformation dictionary. Moreover it is also used in models that can only be applied on transformed data (such as center-based clustering); a reference to a data dictionary defined variable is then used.

3.3. NormContinuous

NormContinuous defines how to normalize continuous variable. It uses the piecewise linear interpolation. A set of couples of value must be defined; each couple is defined by an original value and his normalized value.



To normalize a value, we just have to look for the value right inferior and the value right superior, for example if we want to normalize X, we just have to look for the two couples $(V_2; N_2)$ and $(V_3; N_3)$ such as $V_2 \leq X < V_3$, the normalize value Y of X is given by the formula:

$$Y = \frac{N_3 - N_2}{V_3 - V_2} \times (X - V_2) + N_2$$

3.4. NormDiscrete

The discrete normalization allows the encoding of nominal values into discrete numeric values that can be either 0 or 1. To do such a transformation, there are two methods but in both cases a comparison value V must be defined:

- The first method is called the indicator method; it operates on input variables that can be categorical or ordinal. It sets the output variable to 1 if the input variable is equal to V , otherwise it sets the output variable to 0.
- The second method is called the thermometer method; it operates only on input variables that are defined on an ordered set (such as ordinal variables). It sets the output variable to 1 if the input is greater or equal to V otherwise the output is set to 0. The thermometer method is not yet supported but future version of the PMML standard could.

3.5. Discretize

The discretize transformations is used to encode numeric values into nominal values. It is defined by a set of couples of (interval; nominal value). If the input variable belongs to an interval, then the output is set to the interval's associated nominal value.

3.6. MapValues

This transformation allows any input values to be mapped to any values. This technique uses a look-up table that is composed of, at least, two columns: one for the input values and the other one for the output values. To map a value, we just need to look for the value in the table's input column, the output is then defined at the same row but in the output column.

The look-up table can either be declared inline (inside the PMML document) or somewhere else. In the last case, a table locator must be given (for example: a database connection parameters).

At this time, the specification for table definition is not completely defined, so the table locator is not supported, and changes for the inline table's structure can occur.

3.7. Aggregate

The aggregate transformation is completely different from the others. Usually, transformations operate on a single input value and return a single output value, but aggregate needs to know all the values that the input variable will take. Thus, it can compute the sum, the minimum, the maximum and the count of the input variable. It can also compute a set of values taken by a given variable according to the input value; in this case the result of the transformation is not a single value but a set of values.

Well, this transformation cannot yet be supported in our project for two main reasons:

- At this time no information is provided on how to deal with a set of values. Indeed, the input is a single value but the result is several values whereas a single value is expected.
- This technique requires the reading of all the values taken by the input variable whereas our input-connector architecture does not allow multi-pass through data sources (see paragraph Architecture-IOConnector library).

II.4. The Mining schema

The mining schema is specific to each model defined in the PMML document whereas the Data and Transformation dictionaries refer to all defined models in the document.

The mining schema enumerates the variables used in the data-mining model; it also indicates if a variable is used as input to the model (active variable), as output (predicted variable) or as a non-used variable (supplementary field). It is important to note that the variables listed in the mining schema must be defined in the data dictionary.

The schema also defines the strategy to use in case of an outlier (if the value is outside a defined interval) and in case of missing values.

For outliers, there are three strategies:

- `asIs`: the outlier is considered as a valid value.
- `asMissingValues`: the outlier is considered as a missing value.
- `asExtremeValue`: in this case, a minimum and a maximum values must be defined; if the outlier is lower than the minimum, then it is set to the minimum value, otherwise if the outlier is greater than the maximum, then it is set to the maximum value.

For missing values, there are two strategies: the first one is to do nothing and the other one is to replace the missing value by a replacement value that must be defined.

II.5. The Statistics

For each model, statistic information can be defined for each of the variables used in the model. The statistics provide information on the variables used to build the model at learn time.

5.1. Counts

“Counts” defines not only the total frequency of values but also the frequencies of missing values and invalid values.

5.2. NumericInfo

This field defines statistics information about continuous variables. It indicates the minimum and the maximum values of the variable. It also provides the standard deviation, the median and the inter-quartile-range.

5.3. DiscrStats

`DiscrStats` enumerates a list of values (taken by the variable) with their frequencies. The most frequent value can also be defined.

5.4. ContStats

`ContStats` provides information for continuous variables. It defines the frequency, the sum of values and the sum of squared values for each interval.

5.5. Partition

This field is used to define partition composed of one or several variables, and then to provide statistical information about the variables that belong to the partition. It is then possible, for example, to define populations for clustering models.

II.6. The Regression model

Regressions techniques, in Data-Mining, are used to discover a relationship between a set of variables (independent variable) and a target variable (dependant variable). Once the relationship is discovered, the model is then build and it is possible to predict the value of the target variable according to the independent variables.

There are three kinds of regression:

- **Linear Regression:** This is a method of finding the linear equation that comes closest to fitting the set of independent variables. The regression formula is then:

$$Y = f(X_1, X_2, \dots, X_n) = w_0 + w_1 X_1 + w_2 X_2 + \dots + w_n X_n$$

X_1, \dots, X_n are the independent variables.

Y is the dependent (target) variable.

- **Stepwise Polynomial Regression:** This is the same method as the linear regression, except that we do not look for a line that fits the set of independent variables, but for a curve. The formula is then:

$$Y = f(X_1, X_2, \dots, X_n) = w_0 + w_1 X_1^{V_1} + w_2 X_2^{V_2} + \dots + w_n X_n^{V_n}$$

X_1, \dots, X_n are the independent variables.

Y is the dependent (target) variable.

- **Logistic Regression:** Linear and stepwise polynomial regressions are used to predict continuous variables whereas logistic is used to predict categorical variables. For each category of the target variable, there is one regression equation (polynomial or linear). Thus to predict the value, for each category, we need to convert the result of the regression equation into probability (confidence), to do this we can use the softmax function, the simplemax function, or use the result of the regression as the probability value, once done, the result is the category that has the greater probability (confidence value).

The formula of the logistic regression is:

X_1, \dots, X_n are the independent variables.

Y is the dependent (target) variable.

Y_j is the value of regression equation for category j .

P_j is the confidence value for category j

$$Y_j = f(X_1, X_2, \dots, X_n) = w_0 + w_1 X_1^{V_1} + w_2 X_2^{V_2} + \dots + w_n X_n^{V_n}$$

$$P_j = Y_j \quad \text{if no function is used}$$

$$P_j = \frac{Y_j}{\sum_j Y_j} \quad \text{if the simplemax function is used}$$

$$P_j = \frac{e^{Y_j}}{\sum_j e^{Y_j}} \quad \text{if the softmax function is used}$$

$$Y = \max_j P_j$$

II.7. The Clustering model

The goal of this technique is to discover structures in the data as a whole. Contrary to the regression techniques, there are no dependant or independent variables but just variables. That means that clustering cannot be used to predict but only to classify data (records that are a set of variables, usually a table's row). A clustering model is composed of one or several cluster (groups) that satisfy two criteria:

- The clusters are homogeneous: records that belong to the same group are similar to each other.
- All clusters are different, i.e. records that belong to a group are different of other group's records.

PMML allows defining two kinds of clustering models:

- The center-based clustering.
- The distribution-based clustering.

7.1. The Center-Based clustering

In center based clustering, clusters are defined by a centroid, which is an artificial point of the space of records. A centroid can also be seen as a record that represents an average location of the cluster.

To classify a record, we just need, for each cluster, to calculate the distance of the record from the cluster's centroid, then the record belongs to the cluster that has the lowest distance. It is also possible to classify the record according to his similarity with the clusters; in this case the record will belong to the cluster whose centroid has the greatest similarity with the record.

The calculation of the distance or similarity between a record and a centroid is equivalent to the calculation of the distance/similarity between two records, because a centroid is in the same space.

As records may contain nominal values, it is important to encode such values. So, transformations must be defined. Usually, numeric values are normalized using the NormContinuous transformation and nominal values are encoded using the NormDiscrete transformation but any transformations that provide numeric values are allowed. Once the transformations done, the record is equivalent to a vector of N coordinates (N is the number of variables that compose a record).

Now, the calculation is done by a combination of an inner function and an outer function. The inner function operates on two single field values (on the same variable of the two records), whereas the outer function computes an aggregation over all fields.

PMML supports the following inner functions:

- **absDiff**: Computes the absolute difference of the two values.

$$c(x, y) = |x - y|$$

- **gaussSim**: Computes the gaussian similarity.

$$c(x, y) = e^{-\ln(2) \times \left(\frac{x-y}{s}\right)^2}$$

s is the similarity scale

- **delta**:

$$c(x, y) = \begin{cases} 0 & \text{if } x=y \\ 1 & \text{if } x \neq y \end{cases}$$

- **equal**:

$$c(x, y) = \begin{cases} 1 & \text{if } x=y \\ 0 & \text{if } x \neq y \end{cases}$$

- **table**: This function uses a look-up matrix into which the distance/similarity value must be defined for each possible value.

$$c(x, y) = M(x, y)$$

M is the distance/similarity matrix.

It is important to note that the table inner function is not implemented (in the project) because PMML does not define how the matrix should be build for encoded values. Nevertheless it is supported in the distribution based clustering (see below).

The following outer functions (aggregation functions) are supported:

N is the number of variables that compose a record.

w_i is the weight of the variable i.

x_i and y_i are the values of the variable i (the ith coordinate of the two vectors).

c_i is the inner function for the variable i.

- **euclidean**: This function can only be used for distance calculation.

$$D = \sqrt{\sum_{i=1}^N w_i \times (c_i(x_i, y_i))^2}$$

- **squaredEuclidean**: Only used for distance calculation.

$$D = \sum_{i=1}^N w_i \times (c_i(x_i, y_i))^2$$

- **chebychev**: Only used for distance calculation.

$$D = \max_{i=1}^N (w_i \times c_i(x_i, y_i))$$

- **cityBlock**:

$$D = \sum_{i=1}^N w_i \times c_i(x_i, y_i)$$

- **minkovski**:

$$D = \sqrt[p]{\sum_{i=1}^N w_i \times |x_i - y_i|^p}$$

p is the minkowski parameter, $p > 0$

The following outer functions are only used for similarity calculations:

We set:

a_{11} = number of time $x_i=1$ and $y_i=1$

a_{10} = number of time $x_i=1$ and $y_i=0$

Idem for a_{01} and a_{00}

- o **simpleMatching**:

$$D = \frac{a_{11} + a_{00}}{a_{11} + a_{10} + a_{01} + a_{00}}$$

- o **jaccard**:

$$D = \frac{a_{11}}{a_{11} + a_{10} + a_{01}}$$

- o **tanimoto**:

$$D = \frac{a_{11} + a_{00}}{a_{11} + 2 \times (a_{10} + a_{01}) + a_{00}}$$

- o **binarySimilarity**:

$$D = \frac{c_{11} \times a_{11} + c_{10} \times a_{10} + c_{01} \times a_{01} + c_{00} \times a_{00}}{d_{11} \times a_{11} + d_{10} \times a_{10} + d_{01} \times a_{01} + d_{00} \times a_{00}}$$

$c_{11}, c_{10}, c_{01}, c_{00}, d_{11}, d_{10}, d_{01}, d_{00}$ are non null positive parameters

So, to classify a record, we just need to compute D (distance/similarity) for each cluster, and then the record will belong to the cluster that has the lowest D value for distance, or the greatest D value for the similarity.

7.2. The Distribution-Based clustering

The distribution-based clustering is also known as the demographic clustering. In this technique, clusters are not defined by a centroid but by a distribution of values. For each variable of a record, a cluster defines the probability that the variable belongs to the cluster according to his value. Usually, for each variable and for each value of the variable, the

probability is nothing else than the number of records that were classified (at learn time) in a cluster according to the variable and his value, divided by the size of the cluster (total number of records that belong to the cluster) (See in the references: “Techniques of Cluster Algorithms in Data Mining”).

Let’s take an example: for records composed of two variables: “sex” and “age”, we can defined two clusters like the one below:

We will assume the following:

- The variable “sex” is composed of two categories (male and female).
- The variable “age” is divided into 2 intervals ([0; 35] and [35; 80]); learning was done on 100 records (49 sex=male, 51 sex=female, 30 age=[0; 35] and 70 age=[35; 80]).

cluster	Cluster’s size	sex		age	
		<u>male</u>	<u>female</u>	<u>[0; 35]</u>	<u>[35; 80]</u>
0	49	49	0	19	30
1	51	0	51	11	40

To classify a record, for each cluster we need to calculate the probability that the record belongs to the cluster then the cluster that has the greatest probability must be selected.

It is important to note that the distribution based clustering does not need the nominal values to be encoded.

To calculate the probability, we need an inner function that operates on a single variable and an outer function that computes an aggregation over all the fields.

The algorithm is then:

L is the number of clusters
 R is a record composed of n variables (V_1, \dots, V_n)
 x_i is the value of V_i

$$R(V_1, \dots, V_n) \in \text{cluster}(k) \left| P_k(R) = \max_{j=1}^L P_j(R) \right.$$

$$P_j(R) = \text{Aggregate}_{i=1}^n P_{(j,V_i)}(x_i)$$

If V_i is a continuous variable defined on N intervals: $[A_1; B_1], \dots, [A_N; B_N]$

$$P_{(j,V)}(x) = \sum_{i=1}^N \frac{\text{freq}_j([A_i; B_i])}{\text{size}_j} \times \int_{A_i}^{B_i} c(x, y) dy$$

If V_i is a discrete (such as nominal) variable, with N categories y_1, \dots, y_N

$$P_{(j,V)}(x) = \sum_{i=1}^N \frac{\text{freq}_j(y_i)}{\text{size}_j} \times c(x, y_i)$$

It is important to know that the algorithm above was obtained by reverse engineering. Indeed, the PMML specification does not provide enough information to build an engine able to apply such models on data. Moreover, the demographic clustering's algorithm is the property of IBM, and then there is not a lots of information on the web.

To obtain the algorithm, we have produced demographics clustering PMML files with IBM's *Intelligent Miner for data*; after having modified some parameters we have applied these models on a dataset with IBM's *IM Scoring* (which is able to interpret PMML V1.1 files) and then we have compared the results. Finally, with some information found on the net (see the references), the hypothetical algorithm above was found.

The supported inner functions for the distribution based clustering are:

- **absDiff**: Computes the absolute difference of the two values. The variables must be numeric.

$$c(x, y) = |x - y|$$

- if $x \leq a \leq b$

$$\int_a^b |x - y| dy = \int_a^b (y - x) dy = \left[\frac{1}{2} y^2 - xy \right]_a^b = \frac{1}{2} (b^2 - a^2) - x(b - a)$$

- if $a \leq b \leq x$

$$\int_a^b |x - y| dy = \int_a^b (x - y) dy = \left[-\frac{1}{2} y^2 + xy \right]_a^b = \frac{1}{2} (a^2 - b^2) + x(b - a)$$

- if $a \leq x \leq b$

$$\begin{aligned} \int_a^b |x - y| dy &= \int_a^x (x - y) dy + \int_x^b (y - x) dy = \frac{1}{2} (a^2 - x^2 + b^2 - x^2) + x(x - a + x - b) \\ &= \frac{1}{2} (a^2 + b^2) + x(x - a - b) \end{aligned}$$

- **gaussSim**: Computes the gaussian similarity. The variables must be numeric.

$$c(x, y) = e^{-\ln(2) \cdot \left(\frac{x-y}{s}\right)^2}$$

s is the similarity scale

To calculate $\int_a^b e^{-\ln(2) \cdot \left(\frac{x-y}{s}\right)^2} dy$, we'll use the error function ERF(x) :

$$\text{ERF}(x) = \int_0^x \frac{2}{\sqrt{\pi}} e^{-y^2} dy$$

we set :

$$t = \sqrt{\ln(2)} \times \frac{x-y}{s}$$

$$\frac{dt}{dy} = -\frac{\sqrt{\ln(2)}}{s} \Leftrightarrow dy = -\frac{s}{\sqrt{\ln(2)}} dt$$

$$\int_a^b e^{-\ln(2) \cdot \left(\frac{x-y}{s}\right)^2} dy = \int_{\frac{\sqrt{\ln 2} \cdot \frac{x-a}{s}}{\sqrt{\ln 2} \cdot \frac{x-b}{s}}}^{\frac{\sqrt{\ln 2} \cdot \frac{x-b}{s}}{\sqrt{\ln 2} \cdot \frac{x-a}{s}}} -\frac{s}{\sqrt{\ln 2}} e^{-t^2} dt$$

$$= -\frac{s}{\sqrt{\ln(2)}} \left[\int_0^{\frac{\sqrt{\ln 2} \cdot \frac{x-b}{s}}{\sqrt{\ln 2} \cdot \frac{x-a}{s}}} e^{-t^2} dt - \int_0^{\frac{\sqrt{\ln 2} \cdot \frac{x-a}{s}}{\sqrt{\ln 2} \cdot \frac{x-b}{s}}} e^{-t^2} dt \right]$$

$$= -\frac{s}{\sqrt{\ln 2}} \cdot \frac{\sqrt{\pi}}{2} \left[\text{ERF}\left(\sqrt{\ln 2} \cdot \frac{x-b}{s}\right) - \text{ERF}\left(\sqrt{\ln 2} \cdot \frac{x-a}{s}\right) \right]$$

To calculate the integral, the error function is used, because some C libraries (math.h) have this function, nevertheless we'll use the one described in the Numerical Recipes book (see the references).

- **delta:**

$$c(x, y) = \begin{cases} 0 & \text{if } x=y \\ 1 & \text{if } x \neq y \end{cases}$$

$$\int_a^b c(x, y) dy = \int_a^b (1 - \delta(y-x)) dy$$

$\delta(x)$ is the Dirac Delta Function defined by :

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases}$$

$$\int_a^b (1 - \delta(y-x)) dy = \int_a^b dy - \int_a^b \delta(y-x) dy$$

$$= (b-a) - \int_a^b \delta(y-x) dy = \begin{cases} b-a-1 & \text{if } a \leq x \leq b \\ b-a & \text{otherwise} \end{cases}$$

- **equal:**

$$c(x, y) = \begin{cases} 1 & \text{if } x=y \\ 0 & \text{if } x \neq y \end{cases}$$

$$\int_a^b c(x, y) dy = \int_a^b \delta(y - x) dy = \begin{cases} 1 & \text{if } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

$\delta(x)$ is the Dirac Delta Function (see above)

- **table:** This function uses a look-up matrix into which the distance/similarity value must be defined for each possible value.

$$c(x, y) = M(x, y)$$

M is the distance/similarity matrix.

$M(x,y)$ must be constant $\forall y \in [a; b]$

$$\int_a^b M(x,y) dt = M(x, y) \int_a^b dt = \underline{M(x, y) \times (b - a)}$$

For the outer functions, all the outer functions listed (in the center based clustering, see above) are supported, except simpleMatching, jaccard, tanimoto and binarySimilarity that cannot be computed in the distribution based clustering.

For the minkovski function, the formula below is used:

$$D = \sqrt[p]{\sum_{i=1}^N w_i \times c(x_i, y_i)}$$

$$c(x, y) = |x - y|^p$$

p is the minkowski parameter, $p > 0$

- if $x \leq a \leq b$

$$\int_a^b (y - x)^p dy = \left[\frac{1}{p+1} (y - x)^{p+1} \right]_a^b = \frac{1}{p+1} [(b - x)^{p+1} - (a - x)^{p+1}]$$

- if $a \leq b \leq x$

$$\int_a^b (x - y)^p dy = \left[-\frac{1}{p+1} (x - y)^{p+1} \right]_a^b = \frac{1}{p+1} [(x - a)^{p+1} - (x - b)^{p+1}]$$

- if $a \leq x \leq b$

$$\int_a^b |x - y|^p dy = \int_a^x (x - y)^p dy + \int_x^b (y - x)^p dy = \frac{1}{p+1} [(x - a)^{p+1} + (b - x)^{p+1}]$$

II.8. The Neural Network model

An artificial neural network is a system composed of many simple processing elements, called neurons, operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements.

In data mining, neural networks are, usually, used for data classification and data estimation.

PMML V2.0 allows representing most of the neural network models such as back-propagation network and recurrent network.

A network is defined by:

- **An input layer:** This layer is composed of neurons that receive data from the external environment. The outputs of those neurons are the received data that can have been transformed (usually, continuous and discrete normalization).
- **Hidden layers:** Hidden layers are composed of neurons. Neurons are connected to themselves depending on the network’s architecture, and each connection is defined by a weight. Moreover, each neuron computes his output using his activation function that takes as input the sum of outputs of the output connected neurons.

The formula to compute the output of a neuron is:

Assume :

This neuron takes input from N neurons.

W_i is the weight of the connection from neuron i to this neuron.

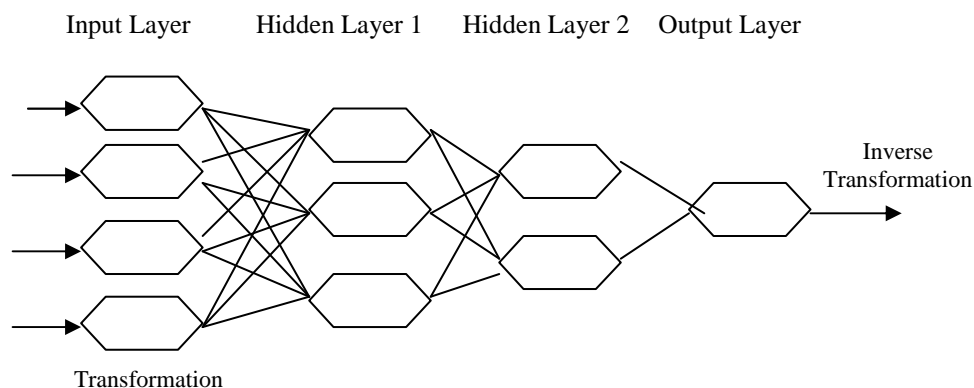
bias is the sum of weights of connections from bias neurons (neurons that have always 1 as output).

$$Z = \sum_{i=1}^N W_i \times output_i + bias$$

$$output = activation(Z)$$

- **An output layer:** The output layer contains neurons that can be connected to a transformation unit. The transformation unit is used to “denormalize” the predicted value; usually it is just the inverse of the transformation used in the input layer. The output layer contains one neuron if the network is used to estimate values and it may contain more than one neuron when the network is used to classify data. If the output layer contains more than one neuron, the neuron that has the maximal activation must be selected.

The schema below is an example of network:



An example of network

The supported activation functions are:

- **threshold:**

$$f(Z) = \begin{cases} 1 & \text{if } Z > t \\ 0 & \text{otherwise} \end{cases} \quad t \text{ is the value of the threshold}$$

- **logistic:**

$$f(Z) = \frac{1}{1 + e^{-Z}}$$

- **tanh:**

$$f(Z) = \frac{1 - e^{-2Z}}{1 + e^{-2Z}}$$

- **identity :**

$$f(Z) = Z$$

- **softmax :** This activation method can only be used by neurons that belong to the output layer.

Assume N output neurons with Z_i as the output of the network for neuron i :

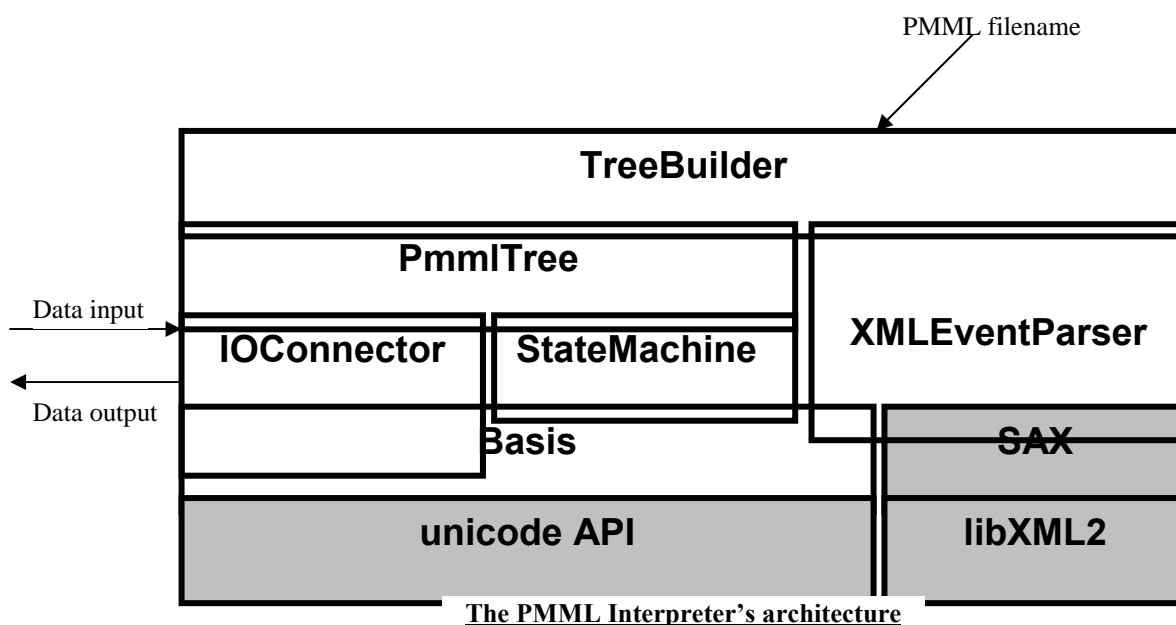
$$f(Z) = \frac{e^Z}{\sum_i^N e^{Z_i}}$$

III. The PMML Interpreter

At the beginning of a project, the most important thing is the design. The design determines if the project will be successful or not, but we'll have also to take into account that we have a deadline which is six months. So we cannot spend too much time in making a good conception; a compromise is then required. The software's architecture that we have proposed is not the best nor the worst, but one that seems acceptable according to the time spent.

III.1. The architecture

The program can be cut into several pieces. Indeed, it must read PMML files, it must then validate them and finally it must build and apply models on data sets. These observations have led us to propose the following architecture:



Each box represents a library that can be static or dynamic.
 Top-levels libraries are dependant of the libraries below.
 Dark box represents external library that are available on the web.

1.1. The Basis library

The basis library contains useful basic functions, like functions that operate on characters strings. All other libraries require these functions.

It is also possible to the program to handle all the alphabets in the world, thanks to Unicode; it is just enough to connect the Basis library to a Unicode API, using the Unicode API in the string class does the connection.

1.2. The XMLEventParser library

This library defines interfaces and methods to receive SAX-like events notification. It provides an abstract parser class that should be derived to allow connections with XML parser libraries.

By default, this library is provided with two parser connectors, one that connects this library to the libXML2 library, and the other that connects to the SaxInCpp library that connects itself to libXML2 library.

The libXML2 parser was selected for the following reasons:

- It is one of the most lightweight parser, moreover we do not ask the parser to do an extensive job, just to notify us about the data since our program is able to validate the file.
- It reads XML documents pretty fast.
- Since it has been written in C, libXML2 can be compiled on almost all platforms.

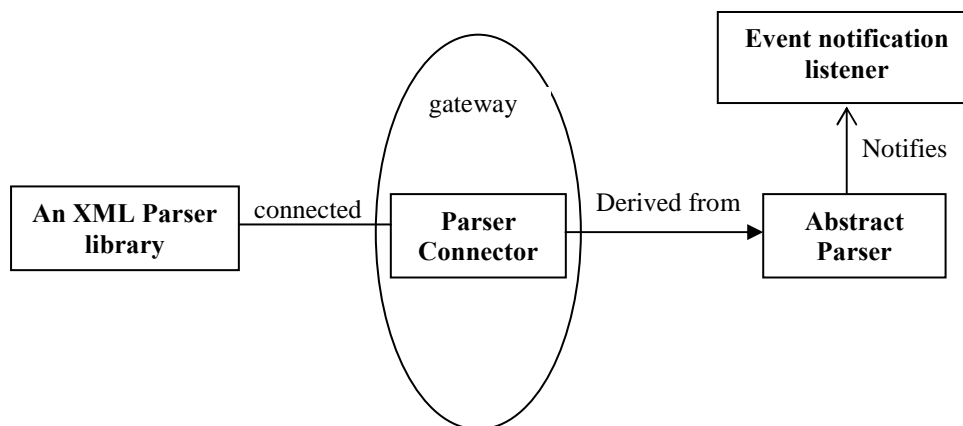
The SAX interface SaxInCpp was selected for the following reasons:

- By now, it is one of the best SAX implementation in C++.
- It provides connectors for the most used XML parsers (Xerces, libXML2, Expat, MSXML2...).

One of the biggest problems with SaxInCpp is that it makes an extensive use of templates and STL, which makes it impossible to compile on some platforms. That is the reason why we have decided to provide a default connector that does not use SaxInCpp but directly connects the XMLEventParser library to libXML2.

Of course, this library allows the use of any other XML parser, all there is to do is to provide a connector that is a class derived from the provided abstract parser class.

The following schema summarizes what has been said above:



Connection with an XML parser

1.3. The StateMachine library

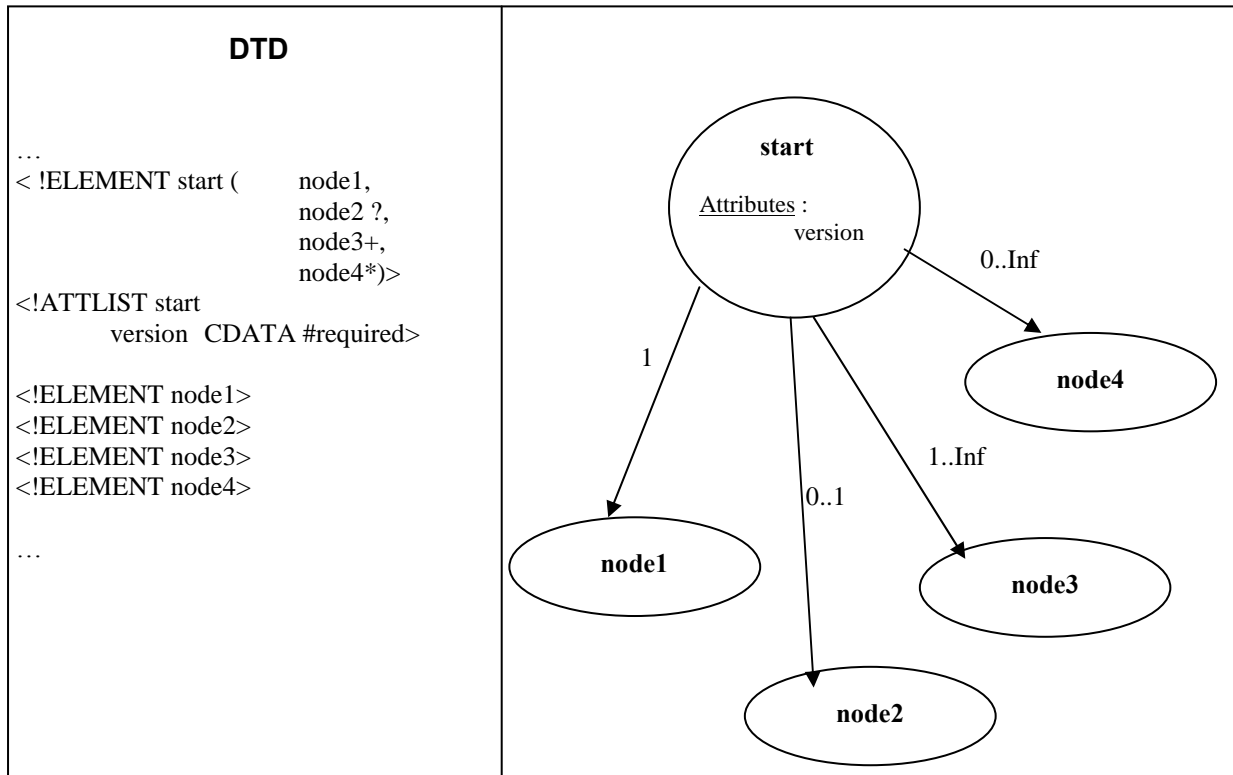
This library provides functions that build a state machine from a given structure. The structure defines states, transitions from a state to the other states, a list of attributes that can be defined for a state and a list of actions to perform in case a transition is made or an attribute is defined. The build state machine is useful to validate an XML document and in case of errors to locate them in the document and then to generate an explicit error message to indicate what was wrong.

A state is defined by a name (the XML's tag name), a list of transitions to other states and by a list of attributes that can be defined for the state.

A transition is defined by a destination state and by a cardinality that indicates the number of times the transition is valid.

An attribute is a kind of a state property, it is defined by a name and a cardinality that indicates if the attribute must be defined or can be defined.

Here is a schema to illustrate:



Relation between DTD and state machines

In order to check the validity of a document, several checks will be performed on each state:

- At the entry of a state: all non-optional attributes must be defined.
- When a transition must be made: the destination state and the transition's cardinality must be valid.
- At the end of a state (when a closing tag is encountered): the state must be consistent.

1.4. The PmmITree library

This library has been specifically designed for PMML V2.0. It provides the structure definition to build a state machine (thanks to the StateMachine library) to handle PMML V2.0 documents.

Moreover, it provides specific classes for each state of the defined state machine; each class derives from a base class that defines a node (to build a tree). It is then possible to define actions that will be triggered by the state machine according to the PMML document and that will operate on the specific node.

These operations will lead to the building of a valid tree that will be a representation of the PMML document. Moreover as each node is a specific class (according to the state=tag),

it is possible to define additional properties; that is what is done by adding, for example, for the nodes that are supposed to represent a data-mining model, the possibility to apply the model on a given data set.

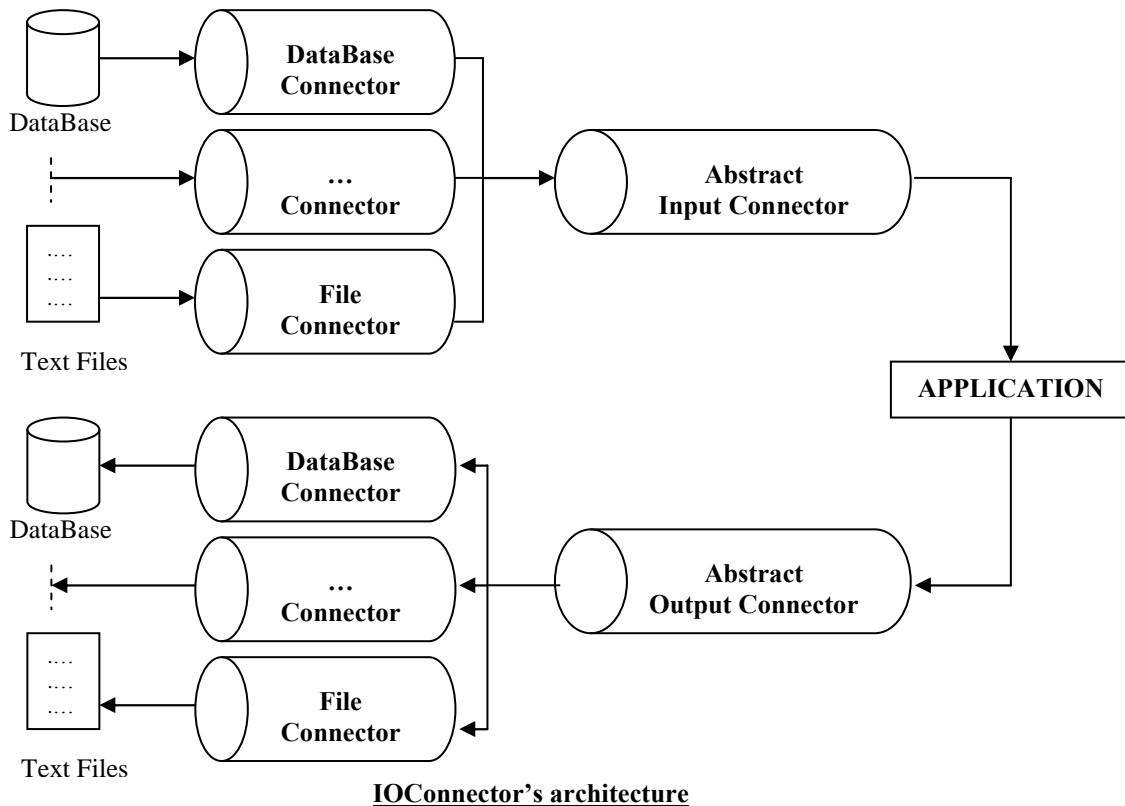
It is also important to note: additionally to the state consistency checks done by the state machine, PmmlTree defines another check that is specific to tree nodes, which consists in checking the consistency of a node according to his children. Indeed, once a node has been completely build, it may be important to check the consistency of this node according to the nodes it depends on. This check allows validating the PMML document according to the specification provided in English, as the XML cannot express them.

1.5. The IOConnector library

This library defines the connectors to the external environment. It defines an abstract input connector and an abstract output connector. To apply data from a source, on a PMML model, an input class derived from the abstract input connector must be provided, the same is true for the output.

An input and output connectors are provided for text files. That enables the reading and writing from text files like CSV files (semi-colon separated files like those generated by Excel).

The schema below describes this library’s architecture:

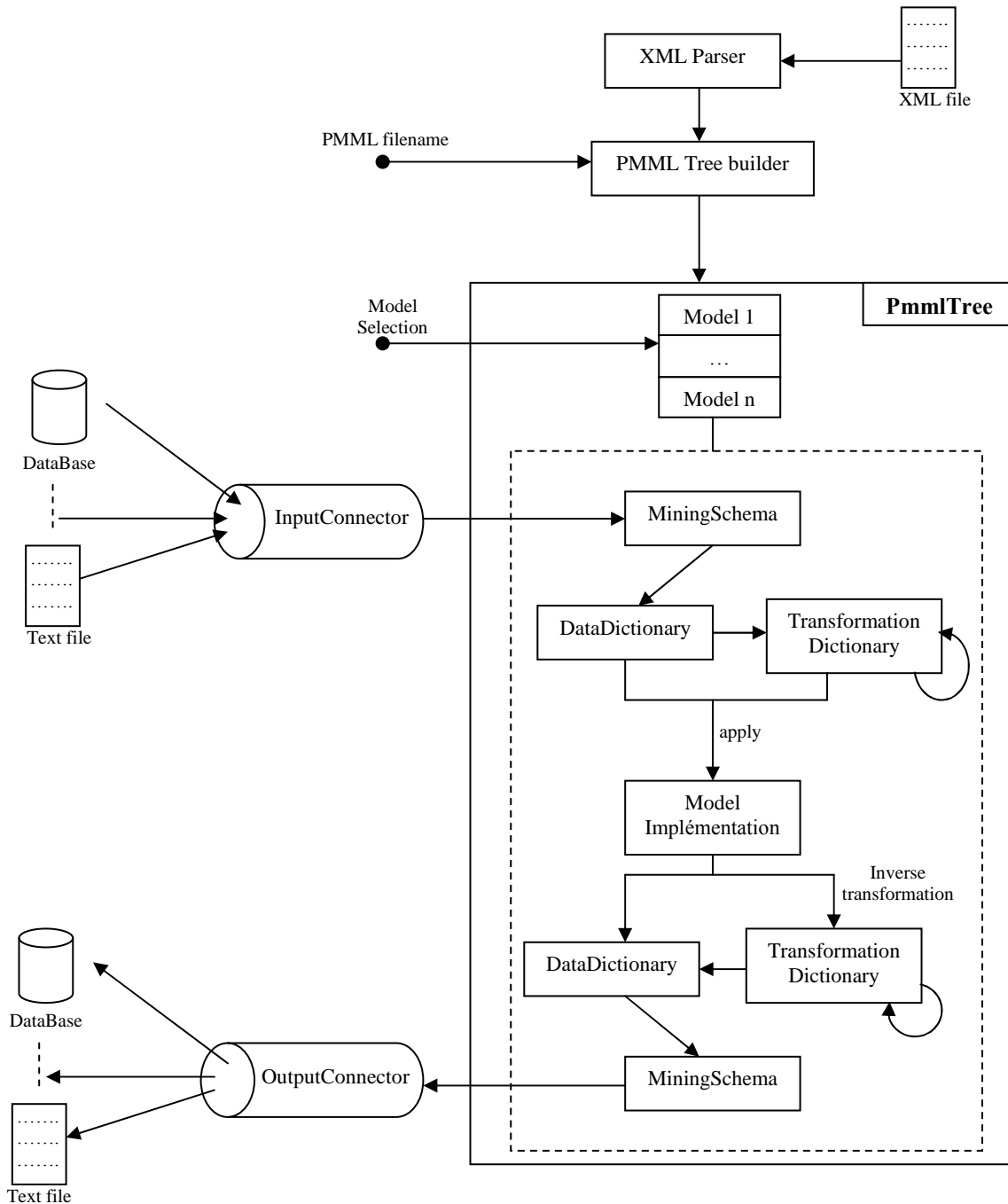


1.6. The TreeBuilder library

This library is provided for convenience. It contains classes used to connect the XMLEventParser library to the PmmlTree library. Thus it is easily possible to read an XML file and then construct the PMML tree.

1.7. The Data flows

The schema below describes the data's flow in the PMML Interpreter:



The data flows in the PMML Interpreter

III.2. The encountered problems

During the development of this project, several problems have been met. This section will detail some of the problems and the adopted solutions.

2.1. The Data types in PMML

PMML does not define data types for variables. It means only continuous variables can be numeric, other variables like categorical and ordinal variables must be strings of characters.

Several applications define ordinal (and even categorical) variables to be numeric variables. Moreover, in data mining, there are not only numeric and string type but also date, integer, real and much more. That means models that use other data types that the two defined in PMML, cannot be correctly exported to PMML V2.0 format.

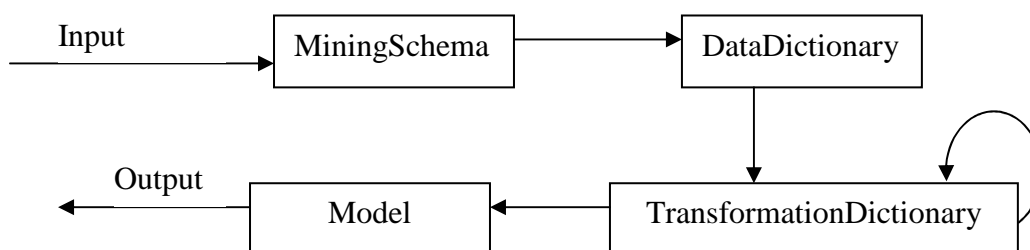
After some discussions with the Data Mining Group (especially with Robert Grossman and Gregor Meyer), future versions of PMML will support, at least, the data types defined by the w3c XML schema.

At this time, the PMML Interpreter only supports the data types defined by PMML V2.0 (numeric and nominal).

2.2. The Missing, invalid and outlier values

In several stages from reading input values to outputting predicted values, strategies are used in case of missing, invalid or outlier values. The PMML norm defines how to locally deal with such values but there are also places where the interpreter must use its own strategy. Using non-standard strategies means that predicted values might be different from one application to another. That's why we'll describe the strategy used by the PMML Interpreter, what is one among so much of others.

The schema below describes the main stages:



The main stages for prediction

- **MiningSchema** :

- If a low Value and/or a high value are defined, the mining schema is used to check if the input value is within the range, if not the user-defined strategy is used:
 - AsExtremeValues: the input value is set to the according extreme value.
 - AsIs: the input value is used as is, that means that the value is valid.
 - AsMissingValues: the input value is declared missing.
- If a missing replacement value is defined, any missing input values are replaced by the default value.

- **DataDictionary:**

The data dictionary is used to validate or invalidate values. The dictionary is only used to check if an input value is within defined intervals or valid values: if so the input value is declared valid, otherwise the value is invalid.

No strategy can be defined to deal with invalid values.

- **TransformationDictionary:**

Only values that are not invalid or missing can be transformed. If a value is invalid or missing, the output value will have the same property (invalid or missing).

- The NormContinuous transformation can declare a value outlier if the value is out of the defined range. As no strategy is defined, the PMML Interpreter will set a flag indicating that the value is out of range, and the value will be replaced by the according extreme value (so calculation on this value can normally be done).
- The Discretize can set a value missing if the value belongs to a non-defined discretize interval.
- The MapValues can declare a value missing if no mapping value is found and no default value was defined. It can also declare a value invalid: if the mapping value is not of an expected type.

No strategy is defined for missing and invalid transformed values.

- **RegressionModel:**

- For Categorical values:
 - The contribution of missing and invalid values is forced to zero.
- For Numerical values:
 - The contribution of invalid values is forced to zero.
 - Missing values are replaced by the mean value if it has been defined otherwise the contribution of such values is forced to zero.
 - Outliers are processed as if they were valid.

The result of the prediction can be checked using the mining schema if a mining field has been defined for the target variable.

- **ClusteringModel:**

- The contribution of missing and invalid values is forced to zero ($c(x,y)=0$).
Outlier values are processed as if they were valid.

- **NeuralNetwork:**

- The contribution of missing and invalid values is forced to zero.
Outlier values are processed as if they were valid.

2.3. The Distribution based clustering.

The PMML standard provides no information about how to apply distribution based clustering models on data, the algorithm used in the PMML Interpreter is the result of about 2 weeks of reverse engineering (see II.7.2). However, there are a few differences between data classified by the PMML Interpreter and those classified by IBM's Intelligent Miner (IMiner). Much time has been spent to try to find out the problem and then to adjust the algorithm, in

order to obtain acceptable result, but without success. We think that the algorithm is correct but that it misses a coefficient in inner functions.

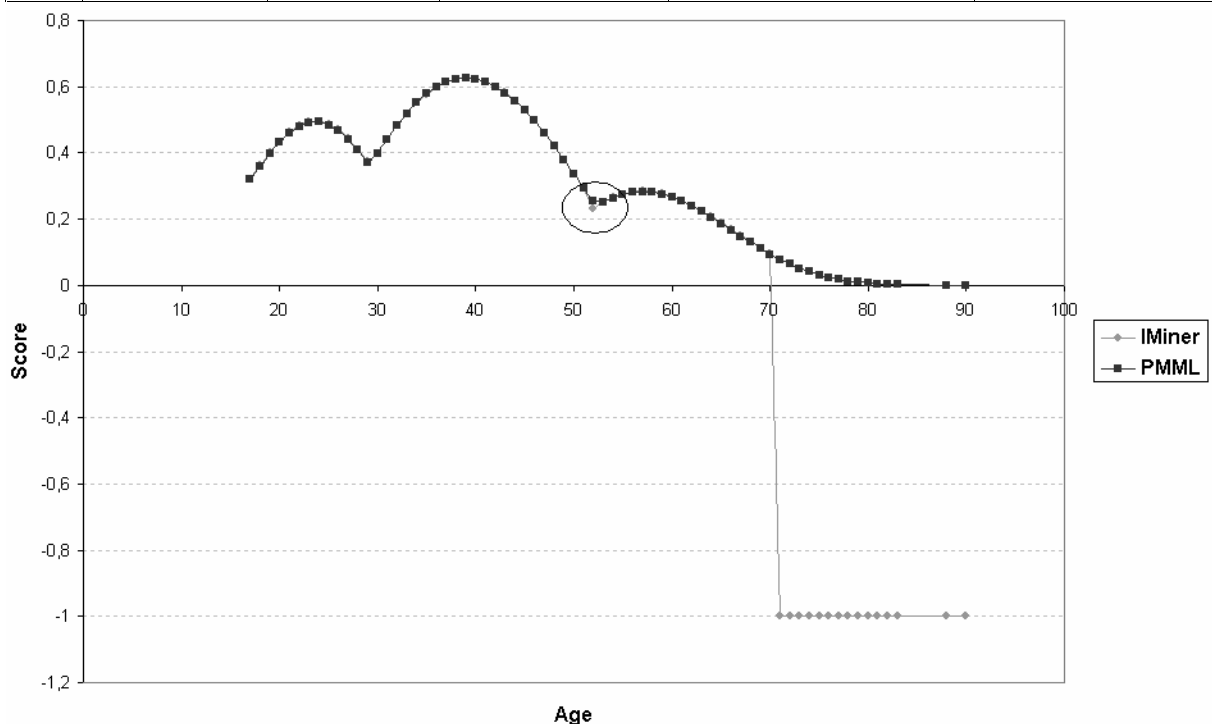
To illustrate the problem with have generated a model with IMiner and then we have applied the model on a data set:

- Only one variable was used: the variable age that is a continuous variable defined on the interval [15; 90], IMiner has divided the intervals into the sub-intervals below:

Intervals	Freq for cluster 0	Freq for cluster 1
[15;15[0	0
[15;20]	0	247
[20;25[0	598
[25;30[124	511
[30;35[675	0
[35;40[623	0
[40;45[617	0
[45;50[535	0
[50;55[85	315
[55;60[0	273
[60;65[0	197
[65;70[0	104
Total freq for clusters	2659	2255

- The inner function used is the gaussian similarity (gaussSim) with the similarity scale $s=6.79692703481335$, and the outer function is the cityBlock (the weight of the variable is $w=1$) (see II.7).
- The result of the classification is nearly the same except for one value of age:

Age	IMiner cluster	IMiner score	PMML cluster	PMML score for clust 0	PMML score for clust1
52	1	0.231788	0	0.253357813	0.231038686

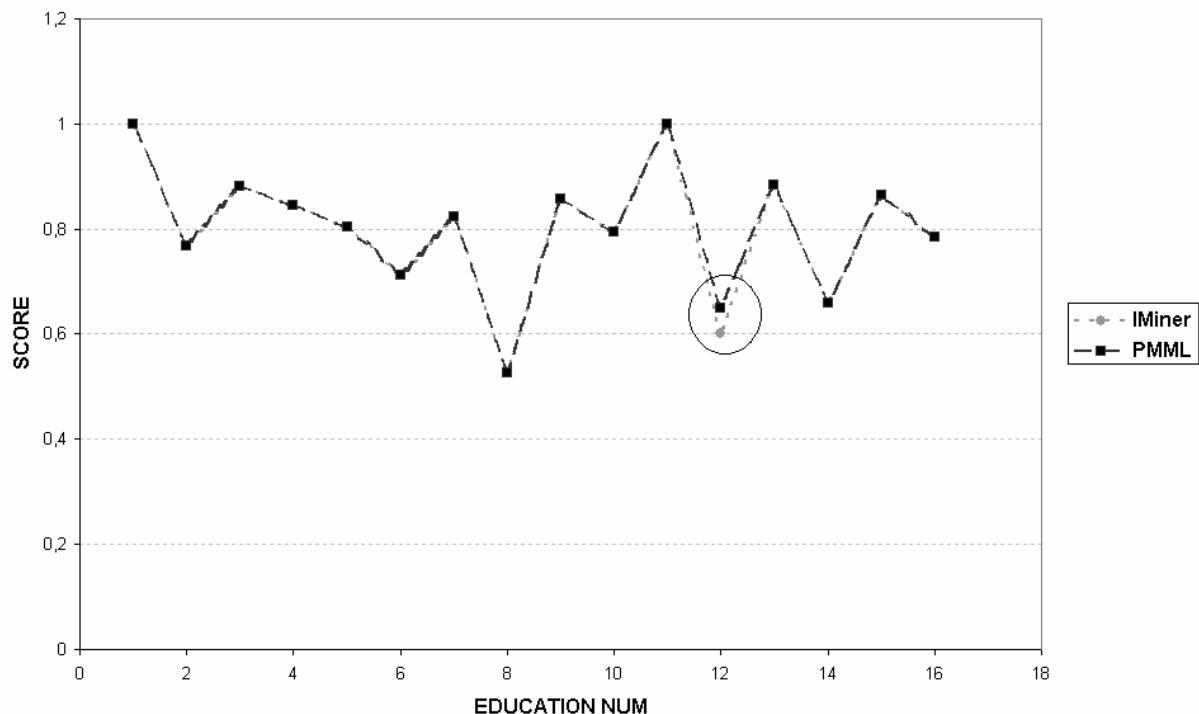


Note that IBM does not classify the values that are superior to 70 (score=-1) as no intervals are defined for such values whereas the PMML Interpreter softly continues the classification.

The same test was done for a discrete numeric variable:

- The ordinal variable is the education-number that can takes the following numeric values: {1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15; 16}.
- The inner function used is the gaussian similarity (gaussSim) with the similarity scale $s=1.26780370513461$ and the outer function is the cityBlock (the weight of the variable is $w=1$) (see II.7).
- The result of the classification is the same except for one value:

Educ	IMiner_cluster	IMiner_score	PMML_cluster	PMML_score for clust 4	PMML_score for clust0
12	0	0.600523	4	0.649701421243555	0.60052288560819



We think that only IBM (or the Data Mining Group) can provide the information to adjust the algorithm, that's one of the reasons (the other is the lack of time) why the algorithm was left as it is.

2.4. The PMML V2.0 stability.

Since the beginning of the project (April, 2002), the Data Mining Group has made several changes to the PMML V2.0 norm. Even if most of these changes were minor changes, they were sufficient to make an old V2.0 compliant file, non-compliant. Moreover, as no revision history is provided the updating of the PMML 2.0 Interpreter is made harder.

After some requests, the Data Mining Group has decided to review the modified version and to move changes to the PMML V2.0.2 and PMML V2.1 norms (standards that are still at definition state).

III.3. The results

By now, August 28, 2002, the PMML Interpreter is able to read and validate PMML V2.0 and PMML V1.1 files. Moreover, it can apply regression, clustering and neural network models on text file data sets.

The program was tested to be compiled on PC with *visual C++* and *GNU g++*.

The initial planning proposed for this project was the one below:

	April	May	June	July	August	September
Documentation	■					
XMLEventListener	■					
State Machine		■				
Basis, TreeBuilder		■				
Dictionaries		■	■			
Statistics			■	■		
Regression				■		
General Regression				■		
Neural Networks				■		
Trees				■	■	
Naïve Bayes					■	■
Association rules					■	■
Cluster						■
Sequences						■

During the project, some other tasks have been added to the project, those tasks and the difficulties encountered have completely modified the initial planning. The final planning is the one below:

III. The PMML Interpreter



	April	May	June	July	August	September
Documentation	■					
XMLEventListener	■					
State Machine		■				
Basis, TreeBuilder		■				
Dictionaries		■ ■ ■				
Statistics			■ ■			
Regression				■		
Clustering				■ ■		
Documentations				■		
Tests				■ ■		
Neural Networks					■ ■	
Code cleaning						■
Documentations						■ ■
Code finalization						■ ■ ■ ■

CONCLUSION

During my training period, I learned a lot especially on how to lead a project to its complete achievement. However, I'm just a little disappointed not to have implemented all the kinds of models. I personally think that I did not envisage all the problems that could arise when working on an unfixed specification (the PMML 2.0 specification). This error has then led to other problems that I did not successfully manage, especially to the time spent to try to solve those problems that finally have, dramatically, slowed down the progress. Problems encountered during a project are also part of the game; we must face them and sometimes it is not possible to solve them within a time limit, when such a condition is met, we must leave the problems on a side, and continue.

As a conclusion, this training has given me a good experience on project's development in a professional environment and has allowed me to know my limits and even how to push them.

REFERENCES

Numerical Recipes in C - The Art of Scientific Computing – 2nd ed.

William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery.
Cambridge University Press, 1988, 1992.

Techniques of Cluster Algorithms in Data Mining - version 2.0.

Johannes Grabmeier, Andreas Rudolph.

IBM Informationssysteme GmbH, 1998.

<http://www-3.ibm.com/software/data/iminer/fordata/clusttechn.pdf>

The Predictive Model Markup Language specification.

The Data Mining Group.

<http://www.dmg.org>

Extensible Markup Language (XML) 1.0 (second edition)

W3C, 2000.

<http://www.w3.org/TR/REC-xml>

XML Schema Part 2: Datatypes

W3C, 2001.

<http://www.w3.org/TR/xmlschema-2>



APPENDICES

Appendix 1: An example of PMML V1.1 File.

```

<?xml version="1.0" encoding="UTF-8"?>
<PMML version="1.1">
  <Header copyright="All the rights reserved to...">
    <Application version="2.0" name="KxEn Components" />
    <Annotation>Model name :</Annotation>
    <Timestamp />
  </Header>

  <DataDictionary numberOfFields="3">
    <DataField optype="continuous" name="age">
      <Interval rightMargin="90" leftMargin="17" closure="closedClosed" />
    </DataField>
    <DataField optype="categorical" name="sex">
      <Value value="Female" />
      <Value value="Male" />
    </DataField>
    <DataField optype="continuous" name="rr_class">
      <Interval rightMargin="" leftMargin="" closure="closedClosed" />
    </DataField>
  </DataDictionary>

  <RegressionModel targetVariableName="rr_class"
modelType="stepwisePolynomialRegression" modelName="">
    <RegressionTable intercept="-0.06902460729401255">
      <NumericPredictor mean="37.8" coefficient="0.0016879" exponent="1" name="age" />
      <CategoricalPredictor value="Female" coefficient="-0.0232244004234" name="sex" />
      <CategoricalPredictor value="Male" coefficient="0.011438891194030" name="sex" />
    </RegressionTable>
  </RegressionModel>
</PMML>

```

Appendix 2: An example of PMML V2.0 File.

```

<?xml version="1.0" encoding="UTF-8"?>
<PMML version="2.0">
  <Header copyright="All the rights reserved to...">
    <Application version="2.0" name="KxEn Components" />
    <Annotation>Model name :</Annotation>
    <Timestamp />
  </Header>

  <DataDictionary numberOfFields="3">
    <DataField optype="continuous" name="age">
      <Interval rightMargin="90" leftMargin="17" closure="closedClosed" />
    </DataField>
    <DataField optype="categorical" name="sex">
      <Value value="Female" />
      <Value value="Male" />
    </DataField>
    <DataField optype="continuous" name="rr_class">
      <Interval rightMargin="" leftMargin="" closure="closedClosed" />
    </DataField>
  </DataDictionary>

  <TransformationDictionary>
    <DerivedField name="c_age">
      <NormContinuous field="age">
        <LinearNorm norm="-0.238280" orig="19.267530" />
        <LinearNorm norm="-0.238280" orig="21.000000" />
        <LinearNorm norm="-0.223173" orig="22.103117" />
        <LinearNorm norm="-0.204592" orig="23.000000" />
        <LinearNorm norm="0.112818" orig="57.000000" />
        <LinearNorm norm="0.086877" orig="60.168179" />
        <LinearNorm norm="-0.011926" orig="61.000000" />
        <LinearNorm norm="-0.011926" orig="68.140244" />
      </NormContinuous>
    </DerivedField>
  </TransformationDictionary>

  <RegressionModel functionName="regression" targetFieldName="rr_class"
modelType="stepwisePolynomialRegression" modelName="">
    <RegressionTable intercept="-0.39307406850485277">
      <NumericPredictor mean="38.72" coefficient="-0.00021" exponent="1" name="age" />
      <NumericPredictor mean="0.0022" coefficient="0.288" exponent="1" name="c_age" />
      <CategoricalPredictor value="Female" coefficient="-0.014239137891" name="sex" />
      <CategoricalPredictor value="Male" coefficient="0.007114067256927" name="sex" />
    </RegressionTable>
  </RegressionModel>
</PMML>

```

Appendix 3: First Progress report, April the 8th, 2002.

PROGRESS REPORT OF THE PMML PROJECT

The goal of this project is to provide a validating PMML interpreter to the data mining community. The interpreter should be able, in a first time to detect non-conform PMML file, in a second time to build a predictive model described by the file and finally the interpreter must be able to apply data to the built model and to output the result.

I. Study of the PMML format

1. Encountered problems

The PMML format, presented on the www.dmg.org web site, raises more problems than expected. Indeed, the content of the web site seems to be incoherent, at several points there are contradictions between the dtd (displayed on the web site under the html format) and the explanations of the different parts of the dtd.

Here is an example that illustrates such incoherence:

- PMML's header taken at www.dmg.org/pmmlspecs_v2/dtd_v2_0.html:
`<!ELEMENT PMML (Header, DataDictionary, (%A-PMML-MODEL;)+, Extension*)>`
- The same header taken this time at www.dmg.org/pmmlspecs_v2/GeneralStructure.html:
`<!ELEMENT PMML (Header, Settings?, DataDictionary, TransformationDictionary, (%A-PMML-MODEL;)+, Extension*)>`

Note that both are parts of the PMML V2.0 format. However the two models are incompatible, i.e. for a PMML file that uses the first part of the dtd, the second part of the dtd would not be able to validate the file.

Another problem arises, the dmg (Data Mining Group) web site presents several xml structures that can be included in the dtd, but at which level? Indeed, the whole dtd does not refer to such structures (Example: the derivedValues structure).

The last encountered problem is that of structures that are not yet defined, for example the InlineTable structure (defined in the Taxonomy structure).

2. Suggested solutions

The PMML format is under development, one could think that the dmg counts on the various actors of Data Mining to give a relevant direction to the evolution of PMML. On the basis of this idea, we can think to use the basis and the various structures, already defined in PMML, to propose a format (with a valid dtd) that will allow realizing the predictive models that will use the following techniques: TreeModel, NeuralNetwork, ClusteringModel, RegressionModel, GeneralRegressionModel, AssociationModel.

Of course, it is not question to redefine a new format but just to refine the existing one.

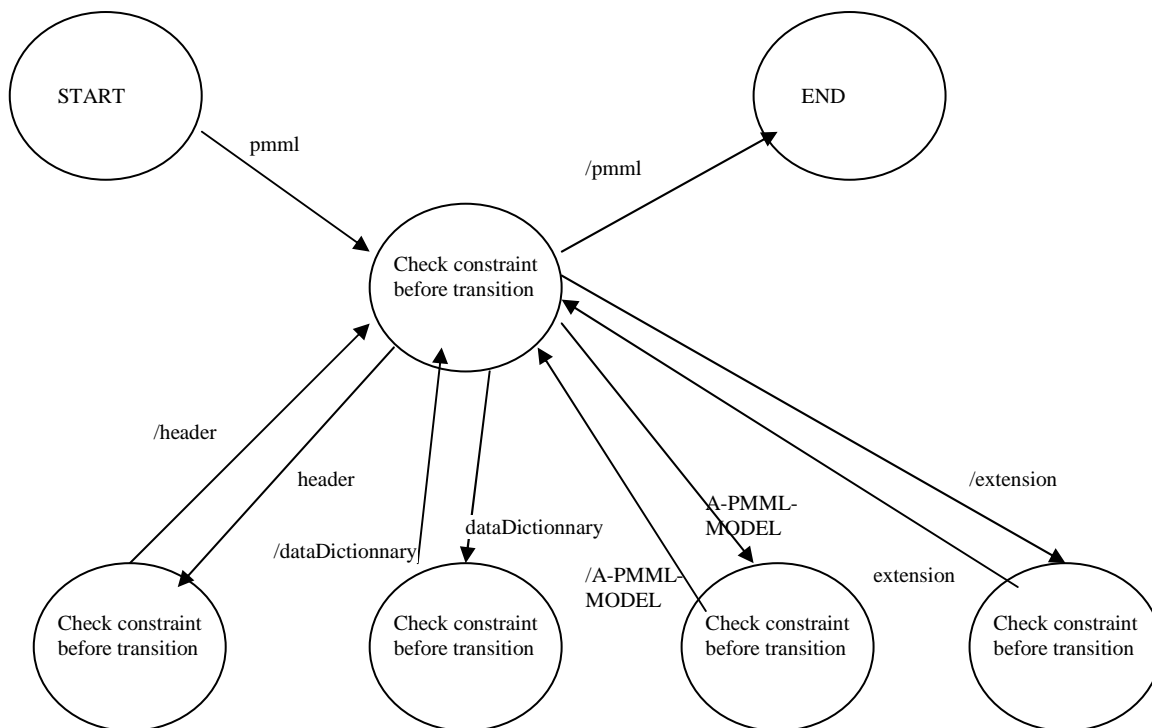
Another important point is that third-party can extend the dtd by defining extension structures. So the interpreter should not be able to use a fixed dtd but the one provided by the

file provider. The question is how would the provided dtd look like? The answer can only be obtained at run-time. Another question is how to be sure that the provided dtd is an extension of the PMML dtd? We can answer to this question by first parsing the file and then checking each structure, to know if it is in agreement with the fixed (PMML) dtd. Finally, what is the utility of the dtd? The provided dtd will be used to validate the xml file, i.e. to check if the file is in conformity with the xml standard. The fixed (PMML) dtd defines the structure that the interpreter should implement, in order to read a correct PMML (extended or not) file.

As we have seen above, we'll have to refine PMML V2.0, in order to get the needed information to build predictive models, but the interpreter must also be flexible to read extended files.

One solution is to build a structure, image of the PMML dtd, defined in C++ language. The structure must be easily modifiable, so we can be able to read and validate extended dtd or just to update the interpreter. Approximately, the structure will represent a determinist and state-finished machine, where each state will have constraints (attributes, number of transitions...).

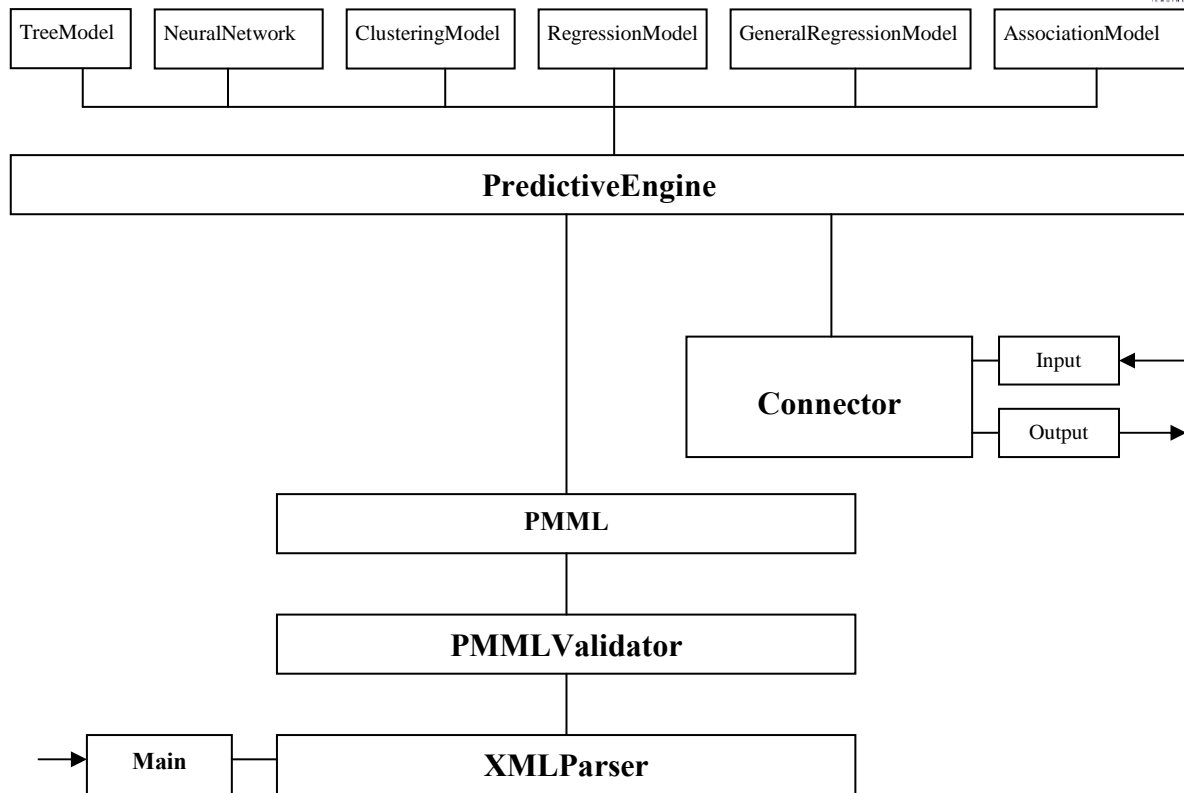
Here is a simple illustration:



This is just an overview of the idea to build a flexible PMML validator, the structure will be defined and deepened in the next reports.

II. Structure of the interpreter

This diagram illustrates the global structure of the interpreter:



1. Main

This is the entry point of the interpreter: an xml filename should be given.

2. XMLParser

This part is used to read an xml file. A SAX-like event parser will be used.

Each time an xml tag will be read, all the information about this tag (attributes, ...) will be send to the PMMLValidator in order to construct and check the PMML structure.

3. PMMLValidator

This part will be used to check and validate the PMML file. It will also construct a PMML structure that can be used to build the predictive engine.

4. PMML

This part contains all the structures needed to construct a memory representation of the PMML file.

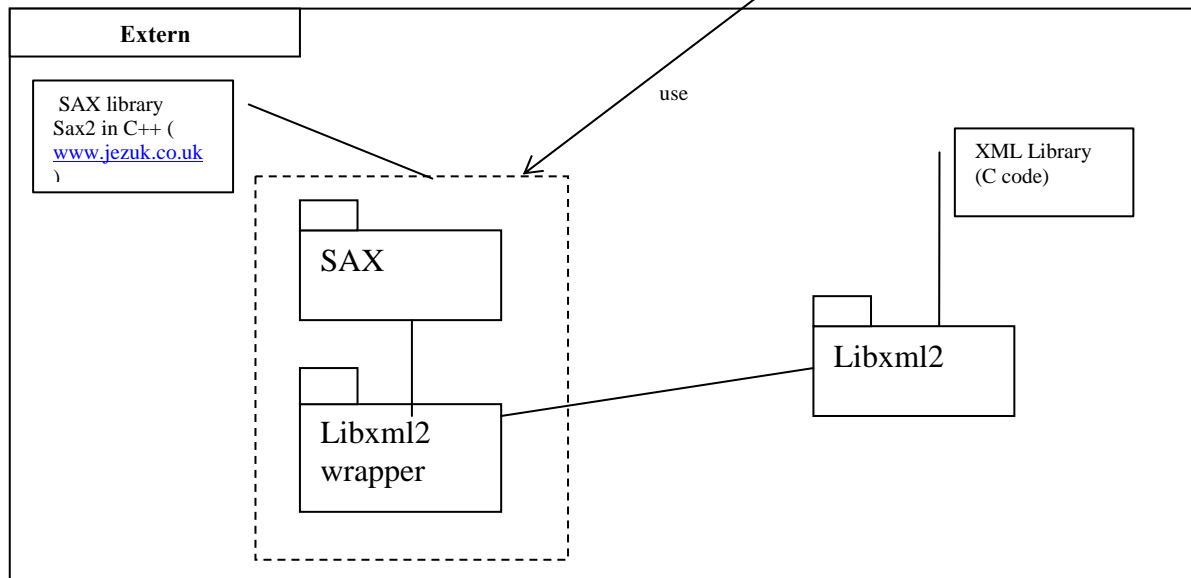
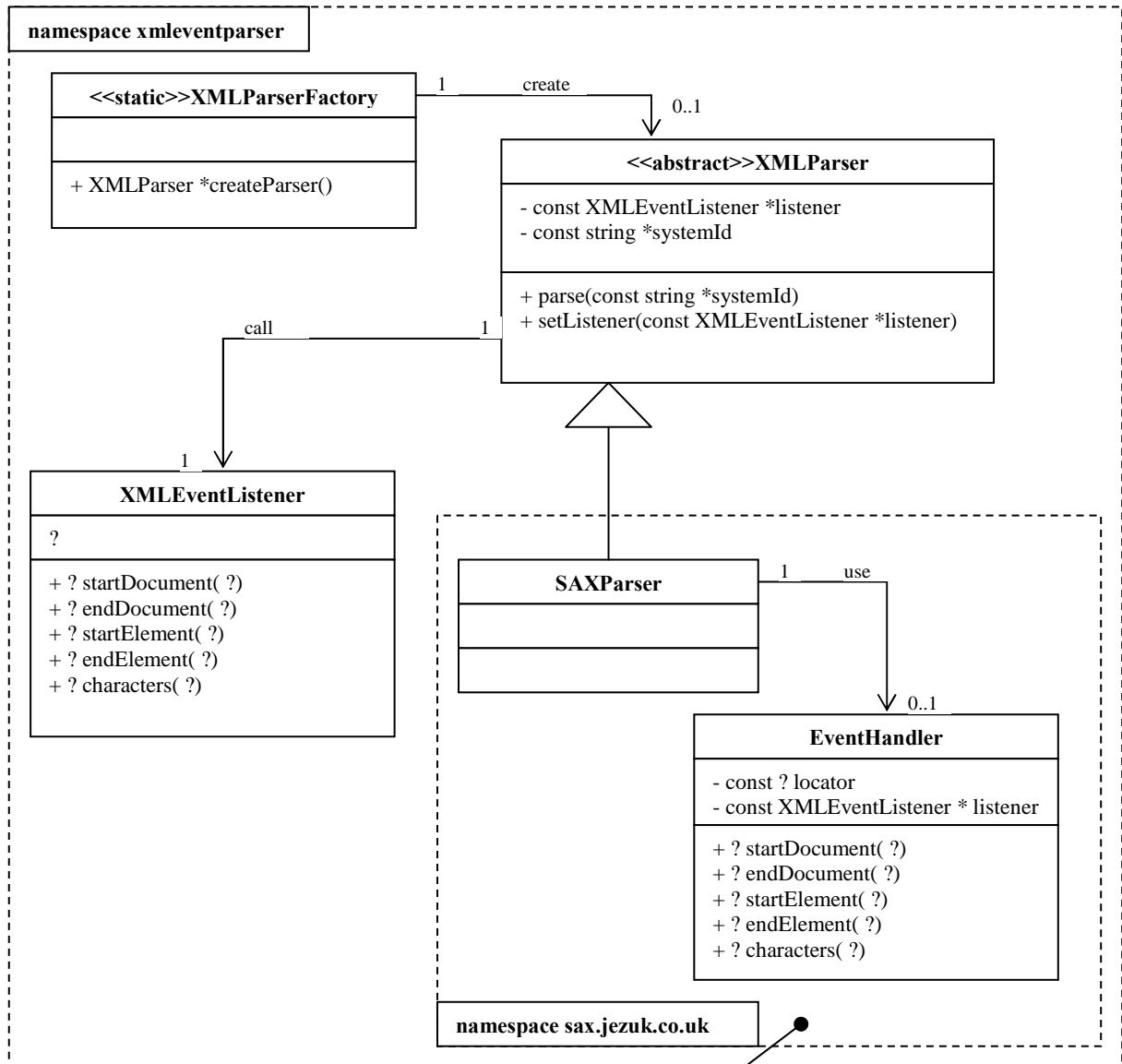
5. Predictive engine

This part of the interpreter will construct and run a model according to the PMML part (see above).

6. Connector

The connector provides data to apply on the built model. It also manages the output result of the prediction.

Structure of the xml file parser.



- **XMLParserFactory:**
This class is used to create a parser (an object that implements the XMLParser class). We just have to define the name of the class that implements the XMLParser.
- **XMLParser:**
This is an abstract class that defines methods that a parser must implement in order to be created and used by XMLParserFactory.
- **XMLEventListener:**
This class is the listener that should be informed each time a tag, or characters are read from the xml file. It will also inform the PMMLValidator that a tag has been read.
- **SAXParser:**
This class implements the methods defined by the XMLParser class. This class will use SAX in C++ and libxml2 as the xml parser.
- **EventListener:**
This class is notified by SAX each time a tag is read, it will forward the event to the event listener (an object of type XMLEventListener).

III. Various problems to resolve

- Define a PMML V2.0 dtd (structure) with enough information to build all the defined models.
- Define the correct structure to build the flexible validator.
- Check for the use of UTF chars instead of standard ASCII characters. This problem may be resolved if the interpreter only works with UTF chars even if the XML file is in standard ASCII, we just have to ask the xml parser (libxml for example) to output UTF chars. Check out if it is possible.

Appendix 4: Nominal to Numeric encoding proposal.

Can we encode nominal values to numeric values? (July the 5th, 2002)

- What is the Data Dictionary?

The data dictionary is a set, containing the definitions of external inputs to the PMML models. That means all the variables (fields) required to apply all the models (defined in the PMML file) must be defined in the data dictionary. A field definition consists of declaring the type of the field (numeric, ordinal, categorical), and a list of valid/invalid values or a list of valid intervals (for numeric field).

Additional external fields (that are not required in the models) can be declared but at least all external fields required to apply all the models must be declared.

- What is the Transformation Dictionary?

The transformation dictionary is a set of internal fields. An internal field is nothing less than a field result of some transformations applied on an external field. Contrary to the data dictionary, it is not possible to indicate neither the type of the result field nor the valid/invalid values. Nevertheless it is possible according to the type of the input field and the type of the transformation to guess the type of the transformed field.

A model can request fields as well from the data dictionary as from the transformation dictionary. The way to guess from which dictionaries the field should be taken is just to check in which dictionary the field (identified by a unique name) is declared. That means a field name is unique in the super set: data + transformation dictionaries.

- Is there a transformation able to encode a nominal to a numeric?

The only transformation well defined is the NormDiscrete, which takes a nominal, check if the value is equal to a given constant and attrib 1.0 if so or 0.0 otherwise. This is not what we want.

Another way is to use the MapValues, which uses a look up table; the function will look for a given value in the table's input column and attrib the new value defined at the same line but in the output columns. That just means it uses a pair of values <V1,V2>, defined in a table that can be declared inline (in the PMML file itself) or somewhere else (a database for example) defined by a table locator; if the input value equal V1 then it will return V2.

Theoretically, it is possible to encode fields of any type to another field of any type.

Well here is an illustration:

First here is the dtd of the transformation:

```
<!ELEMENT MapValues (FieldColumnPair+, Table) >
<!ATTLIST MapValues
  outputColumn CDATA #REQUIRED
  defaultValue CDATA #IMPLIED>

<!ELEMENT FieldColumnPair () >
<!ATTLIST FieldColumnPair
  field %FIELD-NAME; #REQUIRED
  column CDATA #REQUIRED
>
```

Now here is an example:

```
<DerivedField name="newField">
  <MapValues outputColumn="numForm">
    <FieldColumnPair field="gender" column="shortForm">
    </FieldColumnPair>
    <Table>
      <row>
        <shortForm> m</shortForm>
        <numForm>100</numForm>
      </row>
      <row>
        <shortForm>f</shortForm>
        <numForm>200</numForm>
      </row>
    </Table>
  </MapValues>
</DerivedField>
```

Now imagine that a model requires as an input: values from field *newField*, *newField* is an internal field (i.e a transformed input field), so we look for the external input field (the one that is defined in the data dictionary) of *newField*, which is *gender*, we know the type of *gender* thanks to the data dictionary, now we look for the mapped output field, which is *numForm*, we can now get the type of *numForm* thanks to the data dictionary, and finally the type of *newField* is the same than of *numForm*.

We can note that the pair of values is *<gender, numForm>* (*gender* is the key and *numForm* is the result). Moreover it is important to note that *numForm*, *gender* and *shortForm* are input fields (external fields) so they must be defined in the data dictionary: and even if *numForm* is part of the table and not of the standard input, however the field *shortForm* does not really need to be defined in the data dictionary since *shortForm* is only an alias of *gender*.

input = *gender*(i)

if *gender*(i) = *shortForm*(v) then *newField*(i)=*numForm*(v)

i and v can be seen as index or line number...

typeof(*shortForm*) = typeof(*gender*) as defined in the data dictionary.

typeof(*newField*) = typeof(*numForm*) as defined in the data dictionary.

Actually, what was said above are not completely defined by the PMML V2.0 norm since MapValues use tables and table are not yet defined:

“The elements **TableLocator** and **InlineTable** are not yet completely defined because other standardization groups are working on these issues. A preliminary definition of these elements for PMML 2.0 could look like the following example.”
(http://www.dmg.org/pmmlspecs_v2/Taxonomy.html).

But what we'll have to do to allow the above encoding is to force the dmg to publish the following on their website:

```
<!ELEMENT MapValues (FieldColumnPair+, Table) >
<!ATTLIST MapValues
  outputColumn    %FIELD-NAME; #REQUIRED
  defaultValue    CDATA #IMPLIED>
```

The attribute *outputColumn* must be defined in the data dictionary.

GLOSSARY

A

API – Application Programming Interface

A kind of libraries providing specific tools (functions).

Attribute

A qualifier on an XML tag that provides additional information. For example, in the tag <slide title="My Slide">, title is an attribute, and My Slide is its value.

C

cl

The Microsoft C/C++ compiler provided with Visual C++.

Clustering

Data mining techniques that group records together based on their locality and connectivity within the n-dimensional space.

CRM – Customer Relationship Management

The process by which companies manage their interactions with customers.

csv

A text file format that uses semi-colons to separate fields. Applications like Microsoft Excel are able to generate csv files.

CVS – Concurrent Versions System

An open source program that allows to share source code and to keep trace of source modification.

D

Data Mining

An information extraction activity whose goal is to discover hidden patterns contained in data. Data Mining uses a combination of machine learning, statistical analysis, modelling techniques and database technology.

DMG – Data Mining Group

The organisation that is in charge of PMML maintenance and evolution.

DOM – Document Object Model

An API definition for XML Model-Parser.

DTD – Document Type Definition

A document that defines a set of rules (tags) for an XML file.

E

Emacs

A free text editor available under the GNU licence.

H

HTML – Hyper-Text Markup Language

The language of the Web. A system of documents where documents can be linked one to another.

K

KDD – Knowledge Data Discovery

A term often used interchangeably with data mining. *See Data Mining*

L

Libxml2

An open source XML parser written in C, available under the MIT licence.

M**Model**

In data mining, a model is a description that adequately explains and predicts relevant data but is generally much smaller than the data itself.

Model-Parser

An XML parser that reads an entire XML document once and creates representation in memory.

N**Neural Network**

A computing model based on the architecture of the brain. A neural network consists of multiple simple processing units (neurons) connected by adaptive weights.

P**Parser**

See XML Parser.

Pattern

A pattern can be a relationship between two or more variables.

Pull-Parser

An XML parser that reads an XML document in small pieces. It relies on the application to request more data.

Push-Parser

An XML parser that reads an XML document and notifies the application about the events.

PMML – Predictive Model Markup Language

An XML based language to describe statistical and data mining models.

S**SAX – Simple Api for Xml**

An API definition for XML Push-Parsers.

T**Tag**

A generic term for a language element descriptor. It is often referred to as markup.

U**Unicode**

A character-encoding scheme that was developed to store all the alphabets in the world.

X**XML – eXtensible Markup Language**

A set of rules (tags) used to structure data under a text format.

XML Parser

A program that reads and XML document and determines the structure and properties of the data. There are three kinds of XML parsers: push-parsers, pull-parsers and model-parsers.

XML schema

The W3C specification for an XML document.

INDEX

A

activation function.....28, 29
 association 14

B

back-propagation network.....28

C

center based clustering21, 27
 centroïd.....21, 23
 clustering13, 17, 20, 21, 39
 component9, 10, 12, 14
 CRM12
 Customer Relationship Management12

D

Data Mining6, 7, 10, 11, 12, 13, 14, 16, 19,
 20, 28, 33, 35
 Data Mining Group6, 13, 16, 35, 38
 demographic clustering23, 25
 distribution based clustering.21, 22, 23, 24,
 25, 27, 36
 DMG.....13, 14
 Document Type Definition.....12
 DTD.....12, 16

E

eXtensible Markup Language12

G

gaussian22, 25, 37, 38
 general regression..... 14

H

Hidden layer28

I

identity 29
 IM Scoring 14, 25
 IMiner 36, 37
 input layer 28
 Intelligent Miner 14, 25, 36

K

KDD..... 10
 Knowledge Data Discovery 10
 KXEN Analytic Framework..... 7, 14

L

libXML2 31
 linear regression..... 20
 logistic..... 29
 logistic regression 20

N

naïve bayes..... 14
 neural network 13, 28, 39
 neuron 28, 29

O

output layer 28

P

PMML 6, 12, 13, 14, 15, 16, 17, 18, 19, 21,
 22, 25, 28, 30, 32, 33, 34, 35, 36, 38, 39,
 41
 PMML Interpreter..... 6, 34, 35, 36, 38, 39
 Predictive Model Markup Language .. 6, 12

R

recurrent network 28
 regression 9, 14, 20, 21, 36, 39



S

SAX.....30, 31, 48
 sequence14
 simplemax20
 softmax20, 29
 state machine31, 32, 33
 stepwise polynomial regression20

T

tanh29
 threshold.....29

transformation14, 16, 17, 18, 19, 22, 28, 36
 tree 13, 32, 33, 34

U

Unicode..... 30

X

XML 12, 16, 31, 32, 33, 34
 XML parser..... 30, 31
 XML schema 12, 35



