Predictive Model Markup Language (PMML) Representation of Bayesian Networks: An Application in Manufacturing

Saideep Nannapaneni[1], Anantha Narayanan[2], Ronay Ak[3], David Lechevalier[4], Thurston Sexton[3], Sankaran Mahadevan[5], and Yung-Tsun Tina Lee[3]

[1]Department of Industrial, Systems, and Manufacturing Engineering, Wichita State University, Wichita, KS, 67260, USA

[2]Department of Mechanical Engineering, University of Maryland, College Park, MD, 20742, USA

[3]Systems Integration Division, Engineering Laboratory, National Institute of Standards and Technology (NIST), Gaithersburg, MD, 20899, USA

[4]Le2i, Université de Bourgogne, BP 47870, 21078 Dijon, France

[5]Department of Civil and Environmental Engineering, Vanderbilt University, Nashville, TN, 37235, USA

Abstract

Bayesian networks (BNs) represent a promising approach for the aggregation of multiple uncertainty sources in manufacturing networks and other engineering systems for the purposes of uncertainty quantification, risk analysis, and quality control. A standardized representation for BN models will aid in their communication and exchange across the web. This paper presents an extension to the Predictive Model Markup Language (PMML) standard, for the representation of a BN, which may consist of discrete variables, continuous variables, or their combination. The PMML standard is based on Extensible Markup Language (XML) and used for the representation of analytical models. The BN PMML representation is available in PMML v4.3 released by the Data Mining Group. We demonstrate the conversion of analytical models into the BN PMML representation, and the PMML representation of such models into analytical models, through a Python parser. The BNs obtained after parsing PMML representation can then be used to perform Bayesian inference. Finally, we illustrate the developed BN PMML schema for a welding process.

Keywords: PMML; Bayesian networks; Uncertainty; XML; Analytics; Standard; Manufacturing

1. Introduction

Recent technological developments in sensors and data collection have resulted in the availability of large amounts of data from the operation of engineering systems such as manufacturing, aerospace, and civil infrastructure systems [1]. However, in the era of digital transformation, data collection alone is not enough to make smart decisions. To create a smart manufacturing environment where all available information—from within the plant floor and from along the supply chain—is captured in real-time, data analytics techniques are used to derive actionable insights to improve the efficiency of the process and product quality. Such data-driven decision-making can provide manufacturing sector with several advantages [2]. One of the key aspects within data mining is the construction of appropriate analytical (predictive) models from data to enable the required analyses such as surface roughness prediction [3], process monitoring [4], health management, prognostics and diagnostics [5–7], optimization [8], and uncertainty quantification [9,10]. Several types of analytical models to represent the data have been developed and used, such as support vector machines (SVMs), neural networks (NNs), Gaussian process regression (GPR), decision trees, and Bayesian networks (BNs). In addition to building such analytical models, it is also necessary to be able to exchange these models across different data mining environments. This model exchange capability enables collaborative development of analytical models and rapid model deployment without allocating the additional resources for rebuilding the models. For universal communication of models, standards and protocols [11] are necessary that result in standardized representations of the analytical models.

There are two prominent standards that are being used for the communication of analytical models between two data mining environments: (1) Predictive Model Markup Language (PMML) [12], and (2) Portable Format for Analytics (PFA) [13]. PMML represents the de facto standard for the representation and communication of predictive analytic models [12,14]. PFA represents an emerging JSON-based standard for the representation of statistical models and their associated model and data transformations. Similar to PFA, PMML documents are also intermediate text files, in the Extensible Markup Language (XML) file format [14], produced by data analysis tools and later consumed by a scoring engine in the production environment. Both these standards are being developed and maintained by the Data Mining Group (DMG), an independent consortium that develops data mining standards [15]. A primary difference between the two formats is that PFA enables analytics along with model representation and PMML is primarily used for model representation. Between these two standards, PMML has been widely used and supported by several statistical programming tools such as R, Python and SAS, therefore, this paper primarily focusses on PMML. PMML defines a set of rules for the representation of analytical models (called the PMML schema) such that they can easily be exchanged. The entire model development and deployment can be carried out in the following steps: (1) An analytical model such as a BN is created using available data; (2) The analytical model is converted to a PMML representation (using a parser) and communicated to an appropriate practitioner; and (3) The practitioner imports the PMML representation into an analytical model and uses it for his/her needs accordingly. It should be noted that PMML representation is only a textual representation of a model and has no predictive capabilities. The PMML representation needs to be first converted to an analytical model in a programming language, such as R or Python, for prediction.

As mentioned above, several types of manufacturing process analytics are carried out, one of which is uncertainty quantification, which refers to the quantification of several uncertainty sources, and the aggregation and effect on the quantities of interest (QoIs). The QoIs in the manufacturing domain typically refer to the Key Performance Indicators (KPIs) such as energy consumption per part, cycle time, total process time for manufacturing/machining a part, throughput, cost per part, and overall equipment effectiveness (OEE). In complex multistage and multi-level manufacturing processes, the evaluation of KPIs and the uncertainty associated with them becomes increasingly difficult as the uncertainty aggregates from multiple individual processes that affect the overall KPIs. The different uncertainty sources in manufacturing analytics can be related to the inputs (i.e., variability in the input material properties such as density), process variability, models (i.e., uncertainty in the modeling of individual manufacturing processes), and sensors (associated with measurement errors). In our earlier work [9,16], we have used a BN as a mathematical framework for the aggregation of uncertainty from multiple sources to quantify the uncertainty in the KPIs. BNs facilitate two types of analyses: (1) Forward uncertainty propagation, where the uncertainty in the outputs (such as KPIs) can be estimated, given the uncertainty about various inputs, model parameters, and model errors, and (2) Bayesian inference, where unmeasured inputs or model parameters can be estimated given observations of the outputs.

BNs have been used for various applications in manufacturing such as predicting surface roughness [17], tool wear [18,19], maintenance [20], fault diagnosis [21], job-shop scheduling [22], supply chain diagnostics [23], and life cycle analysis [24]. Their ability to incorporate various uncertainty sources and facilitate uncertainty quantification analysis makes them particularly useful for the manufacturing domain. To ease the adoption of BNs in manufacturing through a variety of commercial tools, we believe that it is beneficial to have a standardized representation of such models. To this end, this paper develops a standardized representation of a Bayesian network using PMML, which as mentioned above is the de facto standard for the representation and communication of predictive analytical models.

The previous version of PMML, v4.2.1 [25], did not have a PMML schema for BN representation. However, a schema for a Naïve Bayes model was available. A Naïve Bayes model is a probabilistic classification model and can be considered as a special case of a two-level BN, where the input features are in the first level and the classification class is in the second level [26]. Since Naïve Bayes is a classification model, it cannot be used for the probabilistic prediction of KPIs, which needs to be performed using a regression model constructed using available data. On the other hand, a BN can accommodate dependence between several parameters/features, can facilitate multi-level modeling for the purposes of classification or regression, as opposed to a Naïve Bayes model, which is a two-level classification model that assumes independence between features when conditioned on the classification class. Therefore, we have developed a generalized schema for a BN, and it is available in the latest version of PMML, v4.3 [27]. In this paper, we explain the BN PMML schema and demonstrate its usage using a real-world case study of a welding process. It is worth mentioning that PMML v4.3 also has a schema for Gaussian process regression (GPR) [28], which provides probabilistic prediction. There are a few differences between a BN and a GPR. As mentioned earlier, a BN can be used for both forward uncertainty propagation and Bayesian inference whereas a GPR is typically used for prediction at new input values. A BN is used to aggregate uncertainty from multiple processes in a manufacturing network and estimate the overall uncertainty in the output QoIs resulting from several individual manufacturing processes [29]. A primary disadvantage of a BN compared to GPR is that the prediction or inference can be computationally expensive as it is mostly sampling-based, although there are special cases of a BN where analytical estimation is possible; these cases are later mentioned in Section 2.2.

The overall contributions made through this paper are: (1) Standardized representation of a BN using PMML; (2) Development of a Python-based parser for conversion of PMML representations into analytical models and vice-versa; and (3) Demonstration of proposed methods using a welding process case study.

The rest of the paper is organized as follows. Section 2 provides a brief background of BNs, their construction, and inference techniques. Section 3 discusses the BN PMML representation, and Section 4 presents a BN PMML parser for conversion of a PMML descriptive model to an analytical model, which can be used for forward uncertainty propagation and Bayesian inference. Section 5 illustrates the proposed methodology using the case study of a welding process followed by concluding remarks and future work in Section 6.

2. Theory of Bayesian Networks

A BN represents the joint probability distribution of a set of random variables through a directed acyclic graphical model, consisting of nodes and directed arcs where nodes represent random variables and arcs represent the dependence between the nodes. Using the directional information from the graphical model, the joint probability of the random variables is represented as a product of conditional and marginal probability distributions. The joint probability of n random variables, $\boldsymbol{X} = \{X_1, X_2 \dots X_n\}$ can be represented as

$$f(\boldsymbol{X}) = \prod_{i=1}^{n} f(X_i | Pa_{X_i}) \qquad (1)$$

where Pa_{X_i} represents the set of parent nodes of X_i, i.e., nodes from which the arcs direct to X_i and $f(X_i | Pa_{X_i})$ represents the conditional probability distribution of X_i conditioned on its parent nodes. If X_i has no parent nodes (also referred to as root nodes), then $f(X_i | Pa_{X_i})$ represents the marginal distribution of X_i. It should be noted that the joint probability will be equal to the product of marginal probabilities when all the nodes are independent to each other. For illustration, consider a 5-node BN shown in Figure 1. Using the dependence information from Figure 1, the joint probability distribution of X_1, X_2, X_3, X_4, and X_5 can be described as

$$f(X_1, X_2, X_3, X_4, X_5) = f(X_1) f(X_2) f(X_3 | X_1, X_2) f(X_4 | X_3, X_1) P(X_5 | X_2) \qquad (2)$$

Techniques for the construction of BNs are discussed in Section 2.1. BNs are primarily used to update our knowledge on a subset of random variables when another disjoint subset of random variables is observed. Depending upon the network complexity, several analytical and approximate techniques were developed in the literature to perform Bayesian inference; these techniques are discussed in Section 2.2.

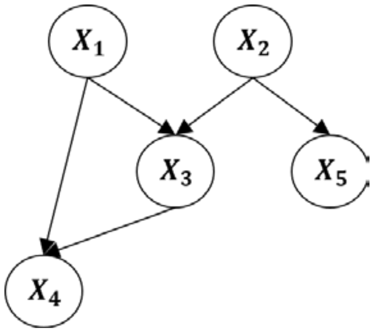


Figure 1: An illustrative Bayesian network.

2.1. Bayesian network construction

Construction of a Bayesian network involves estimation of dependence relationships between variables, both qualitatively and quantitatively through marginal and conditional probability distributions. The techniques for the construction of BNs can be categorized into three types: (1) Using available mathematical models [16]; (2) Data-driven approaches [30]; and (3) Hybrid approaches [29,31]. The available mathematical models can be derived from the physics of the system, or empirical models constructed from a combination of expert knowledge and experiments regarding the system. When mathematical models are unavailable but data regarding system operation is available, the BN can be constructed using BN learning algorithms, which are discussed later in this section. In some cases, it may be possible that the available mathematical models explain some aspects of the system, and data is available regarding the unknown aspects of the system. In such scenarios, a hybrid approach for learning is implemented where a BN is built in two steps. In the first step, a partial BN is constructed using available mathematical models or expert domain knowledge. In the second step, the partial BN constructed in the first step is used as a starting point (as opposed to a completely unconnected graph in the case of data-driven approach) to learn the remaining dependencies between the variables. We discuss below the learning algorithms for BN construction from data. Constructing a BN involves learning the dependence between variables, both qualitatively and quantitatively. Qualitative learning is typically referred to as structure learning whereas quantitative learning involves the estimation of parameters of the conditional and marginal probability distributions associated with several nodes in the Bayesian network. The techniques for learning a BN structure is discussed below.

Given an amount of data on a set of random variables, we first identify the BN structure (network topology) that best explains the dependence between the variables. The structure learning algorithms can be classified into three categories: (1) Constraint-based; (2) Score-based; and (3) Hybrid, i.e., a combination of constraint-based and score-based methods [30]. Constraint-based methods perform conditional independence tests between the random variables, and then identify the BN structure that satisfies the independence tests. Commonly used independence tests for discrete variables include Mutual Information Test (MIT), G-test and Chi-squared test, and those for continuous variables include MIT and conditional correlations [30,32]. The expressions of MIT for discrete and continuous variables are given in Eqs. 3 and 4, respectively. Note that MIT can be used between discrete-discrete and continuous-continuous variable pairs but not for discrete-continuous variable pairs.

$$I_{X,Y} = \sum_{Y}\sum_{X} f(x,y)\, log\left(\frac{f(x,y)}{f(x)f(y)}\right) \tag{3}$$

$$I_{X,Y} = \int_{Y}\int_{X} f(x,y)\, log\left(\frac{f(x,y)}{f(x)f(y)}\right) \tag{4}$$

where $f(x,y)$ represents the joint probability distribution of X and Y, and $f(x)$ and $f(y)$ represent the marginal distributions of X and Y, respectively. The conditional and marginal distributions of the considered variables in Eqs. 3 and 4 are derived from the available data in the learning process. Some constraint-based algorithms include the PC algorithm [33], Grow-Shrink [34], and Incremental Association Markov Blanket (IAMB) [35] along with its variants such as Fast Incremental Association and Interleaved Incremental Association [35,36] .

Score-based methods assign a score for every possible BN; this score is calculated based on the goodness-of-fit of the BN in fitting the available data. Using the scoring measure, a set of heuristic optimization techniques are used to learn the BN structure that optimizes the defined score. Commonly used scoring measures include Likelihood, Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), Bayesian Dirichlet Equivalence (BDe), and Minimum Description Length (MDL) [30]. Some commonly used measures (BIC and BDe) are provided in Eqs. 5 and 6. The BDe score refers to the posterior probability of a BN given data.

$$BIC = k \times \ln(n) - 2 \times \ln(L) \tag{5}$$

$$f(G|D) \propto f(D|G)\, f(G) = f(G) \int f(D|G,\Theta)\, f(\Theta|G)\, d\Theta \tag{6}$$

In Eq. 5, $L, k,$ and n represent the likelihood of observing the available data given a BN, the number of free parameters that are estimated, and the number of available data samples, respectively. In Eq. 6, G and D refer to a BN structure and available data, respectively. Some commonly used optimization techniques for obtaining an optimal BN include greedy search algorithms such as Hill-Climbing and Tabu search [30]. A key difference between the constraint-based algorithms and score-based algorithms is that the former may result in a partially directed graph whereas the latter always results in a directed graph. The inability of the constraint-based methods to obtain a directed graph is because the directions cannot be deduced from the available data. The directionality of the graph is particularly useful to decompose the joint probability distribution into a set of conditional and marginal probability distributions; this decomposition is useful to carry out Bayesian inference.

Hybrid algorithms employ a combination of constraint-based and score-based techniques to obtain an optimal BN structure. The optimal BN through hybrid algorithms is obtained in two stages. In the first stage, constraint-based methods are used to obtain a partially directed graph. In the second stage, the score-based methods are used to obtain the orientation of undirected edges in the partially directed graph that best explains the available data. Some examples of hybrid learning algorithms include Max-Min Hill Climbing and 2-phase Restricted Maximization [37]. The next step after obtaining an optimal BN structure is the estimation of the parameters in the conditional dependence relationships between the random variables. Typically, the principle of Maximum Likelihood is used for parameter learning.

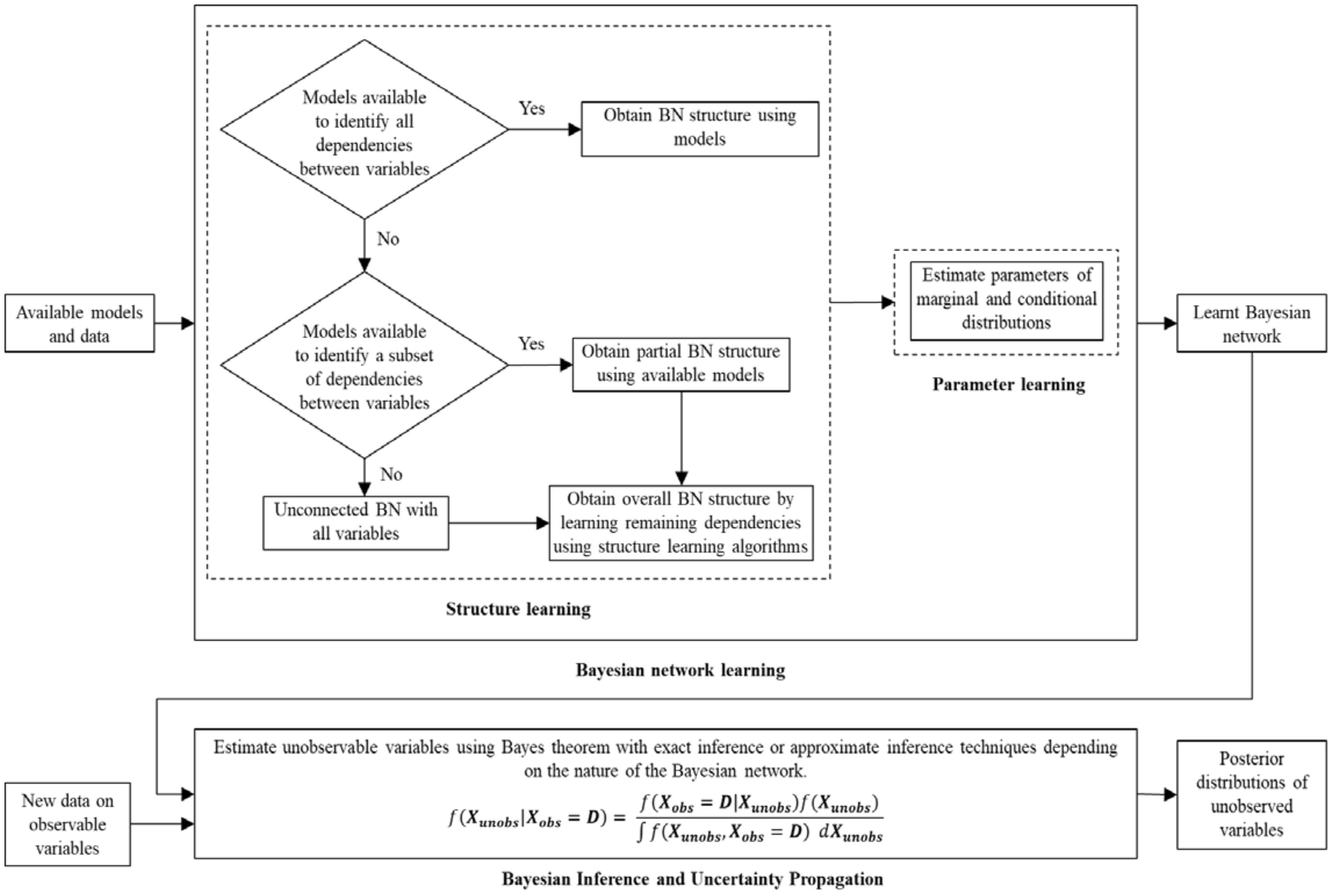
Incorrect Bayesian network learning can lead to incorrect model predictions, which can result in incorrect decision-making when the model is used for further analytics such as probabilistic design optimization and system health diagnosis and prognosis. When the Bayesian network is learnt from validated mathematical models, its structure (dependence relationships) may be known precisely. Conversely, Bayesian network learning from data often depends on the amount of available data, and the complexity of model (e.g., maximum number of parent nodes for a child node) [38]. In addition, different learning methods (constraint-based, score-based or hybrid) and different learning metrics within each method (e.g., different scoring metrics in score-based methods) may result in different Bayesian networks. Therefore, the constructed Bayesian network models need to be validated using additional data (collected independently of training data) before using them for further analytics and decision-making. The readers are referred to [39] for a review of validation techniques.

2.2. Uncertainty propagation and Bayesian inference

The BN constructed using the techniques in Section 2.1 can now be used for inferring the posterior distributions of unobserved variables (denoted as $\boldsymbol{X_{unobs}}$) using any data ($\boldsymbol{D}$) on the observable variables ($\boldsymbol{X_{obs}}$) via Bayes' theorem as

$$f(\boldsymbol{X_{unobs}}|\boldsymbol{X_{obs}} = \boldsymbol{D}) = \frac{f(\boldsymbol{X_{obs}} = \boldsymbol{D}|\boldsymbol{X_{unobs}})f(\boldsymbol{X_{unobs}})}{\int f(\boldsymbol{X_{unobs}}, \boldsymbol{X_{obs}} = \boldsymbol{D})\ d\boldsymbol{X_{unobs}}} \tag{7}$$

In Eq. 7, the terms $f(\boldsymbol{X_{unobs}}|\boldsymbol{X_{obs}} = \boldsymbol{D})$, $f(\boldsymbol{X_{unobs}})$, and $f(\boldsymbol{X_{obs}} = \boldsymbol{D}|\boldsymbol{X_{unobs}})$ represent the posterior distributions of unobserved variables, their prior distributions, and the likelihood function of observed variables. The denominator term, $\int f(\boldsymbol{X_{unobs}}, \boldsymbol{X_{obs}} = \boldsymbol{D})\ d\boldsymbol{X_{unobs}}$, refers to the probability of observable variables to be equal to the data; this is a deterministic value that can be computed by marginalizing over the unobservable variables. Updating in a generic BN can be computationally intractable due to high-dimensional integration, i.e., exact inference is prohibitively expensive. However, exact algorithms are available for updating in special classes of BNs such as discrete BNs, conditional linear Gaussian networks, and networks where conditional dependence relationships are modeled using mixtures of truncated exponentials or truncated polynomials [40]. When exact updating techniques are not available, sampling-based techniques are used to obtain the posterior distributions. In the case of forward uncertainty propagation when the observed variables are the inputs, the distributions of downstream variables can be obtained using Monte Carlo sampling over the conditional dependence relationships. For Bayesian inference, techniques such as Markov Chain Monte Carlo (MCMC) methods [41], variational methods [42], bootstrap filters [43], approximate Bayesian computation (ABC) [44], and unscented transforms [45] can be used to obtain the posterior distributions of the upstream nodes given observations of the downstream nodes. The above BN learning procedure and Bayesian inference are summarized in the flowchart shown in Figure 2.



Models available to identify all dependencies between variables
Yes
Obtain BN structure using models
No
Available models and data
Models available to identify a subset of dependencies between variables
Yes
Obtain partial BN structure using available models
No
Unconnected BN with all variables
Obtain overall BN structure by learning remaining dependencies using structure learning algorithms
Structure learning
Estimate parameters of marginal and conditional distributions
Parameter learning
Learnt Bayesian network
Bayesian network learning
Estimate unobservable variables using Bayes theorem with exact inference or approximate inference techniques depending on the nature of the Bayesian network.
$f(\boldsymbol{X_{unobs}}|\boldsymbol{X_{obs}} = \boldsymbol{D}) = \frac{f(\boldsymbol{X_{obs}} = \boldsymbol{D}|\boldsymbol{X_{unobs}})f(\boldsymbol{X_{unobs}})}{\int f(\boldsymbol{X_{unobs}}, \boldsymbol{X_{obs}} = \boldsymbol{D})\ d\boldsymbol{X_{unobs}}}$
New data on observable variables
Posterior distributions of unobserved variables
Bayesian Inference and Uncertainty Propagation

Figure 2. Flowchart summarizing the Bayesian network learning and inference procedures.

3. Bayesian Network – PMML Schema

PMML is an XML-based standard that defines schemas for representing different machine learning models. The schemas are defined using the XML Schema Definition Language (XSD) standard [46]. In this section, we describe the schema that we developed to represent BNs. While we use some XSD terminology in describing this schema, the conceptual model may be applied in any notation or environment. The figures in this section were generated by a software tool for viewing and editing XSDs. The XSD schema itself is tool agnostic, and may be used with any XML processing tool. An XML document contains a sequence of ‘elements’ and their ‘attributes’, and elements may contain other elements in a hierarchical tree-like structure. The schema defines the types of these elements and attributes, and their structure. In addition, the schema for BN uses other terms already defined in the PMML standard, and we refer the reader to the PMML standard for more details on these terms [12].

A BN typically consists of two types of nodes: root nodes and non-root nodes. Root nodes are variables that can be described without dependence on any other nodes. Similarly, non-root nodes are variables that are not independent but are described based on their parent nodes. The root nodes and non-root nodes can be either discrete or continuous in nature. The following information is needed for a complete description of any BN.

- Marginal probability tables for discrete root nodes, i.e., a set of all possible values that can be taken by a discrete variable and their probabilities.
- Marginal probability distributions of continuous root nodes, which include a distribution type (e.g., Gaussian, Uniform, etc.) and its corresponding distribution parameters.
- Conditional probability distribution tables of discrete non-root (child) nodes defined on a set of parent nodes, i.e., for a given realization of a discrete parent node or a given range of continuous parent node, a conditional probability table (possible values and their probabilities) is defined.
- Conditional probability distributions of continuous non-root nodes defined over a set of parent nodes; this conditional distribution consists of a distribution type and its parameters dependent on the parent nodes.

We describe here the PMML schema to represent the above information. Figure 3 provides a high-level view of the BN PMML schema. We first define an element type called *BayesianNetworkmodel*, which will be the root element of our BN model. Figure 3 shows the main constituents of a *BayesianNetworkModel* in PMML. The attributes of this element are shown in the box at the top, along with the attribute type. Its child elements are shown in the rounded rectangles on the bottom branch of the figure. The generic BN PMML representation provides a set of rules based on which instance BN models can be defined. *modelName* represents the name of the instance BN model such as “Welding Process BN”. *functionName* represents if the model is of classification or regression type, and its type *MINING-FUNCTION* is a special type defined elsewhere in the PMML standard. *algorithmName* refers to the name of the algorithm (e.g., MCMC) that is used for prediction process. *isScorable* identifies if the model can be used to perform predictions. Most of the elements shown in Figure 3 are generic and apply to all predictive models in PMML, except the element *BayesianNetworkNodes*, which was specifically defined for describing BNs. The description regarding the standard elements can be found in [15] and [47], and in this paper, we focus on the *BayesianNetworkNodes* element. Figure 4 describes the *BayesianNetworkNodes* schema. The nodes in a BN can be either discrete or continuous, which is specified in Figure 4.

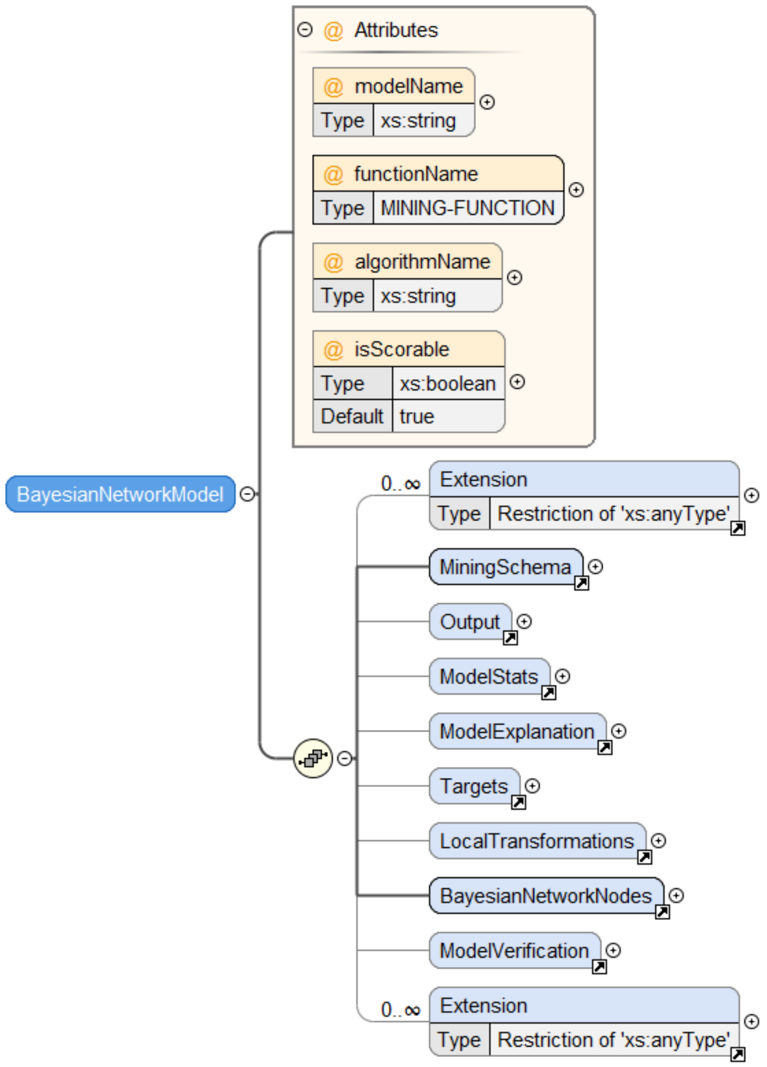


Figure 3. Schema of a *BayesianNetworkModel* element.

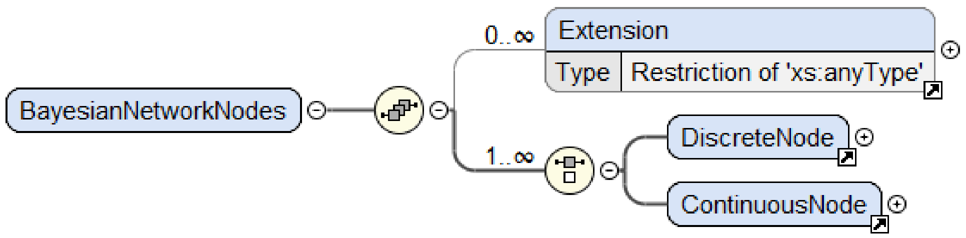


Figure 4. Schema of a *BayesianNetworkNodes* element.

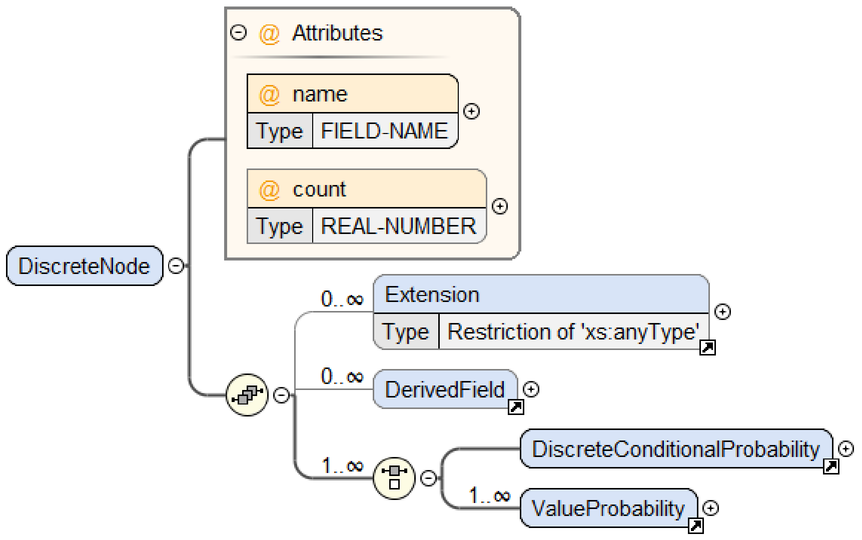


Figure 5. Schema of a *DiscreteNode* element.

3.1. DiscreteNode element

The schema for the *DiscreteNode* is provided in Figures 5 and 6. A *DiscreteNode* element has two attributes: *name* and *count*. *name* refers to the variable name. *count* is an optional attribute that refers to the total number of cases (frequency) in the data used to compute the probability values, and is useful for updating the probability values when new data is available. *DiscreteNode* is a complex element, i.e., it is dependent on other elements such as *DerivedField*, *DiscreteConditionalProbability*, and *ValueProbability*. A discrete variable can be either a root node or a non-root node. If a discrete node is root node, then the possible values and their probabilities are defined using the *ValueProbability* element, as shown in Figure 7. *ValueProbability* is a basic element, i.e., this does not depend on any other elements, as opposed to *DiscreteNode*. The possible values are provided using the *value* attribute, and the probabilities are provided using the *probability* attribute. The probability attribute is of type *PROB-NUMBER*, which is a special type defined in the PMML standard, and represents real numbers between 0.0 and 1.0. As opposed to a discrete root node, a discrete non-root node is defined using the *DiscreteConditionalProbability* element. As discussed above, a conditional probability table of a discrete non-root node is described based on the values of the parent nodes. In the *DiscreteConditionalProbability* element, the parent node values are provided using the *ParentValue* element and the probabilities conditional on these parent node values are provided using the *ValueProbability* element. The *ParentValue*, as shown in Figure 8, is also a basic element where the parent nodes and their values are given using the *parent* and *value* attributes. The parent attribute is of type *FIELD-NAME*, which specifies that the value of parent must be another element in the model (i.e. the parent node must exist in the BN model). In some cases, when a discrete non-root node is dependent on a continuous node, the continuous node is discretized and the conditional probability table of the discrete node is defined in each discretized range of the continuous variable.

This discretization transformation is described using the *DerivedField* element, a standard PMML element that is specified once but used in multiple model elements within the PMML document for reducing the size of PMML documents that contain multiple model elements (refer to PMML v4.3 [27] for more details). Thus, schema elements are defined to describe discrete root and non-root nodes.

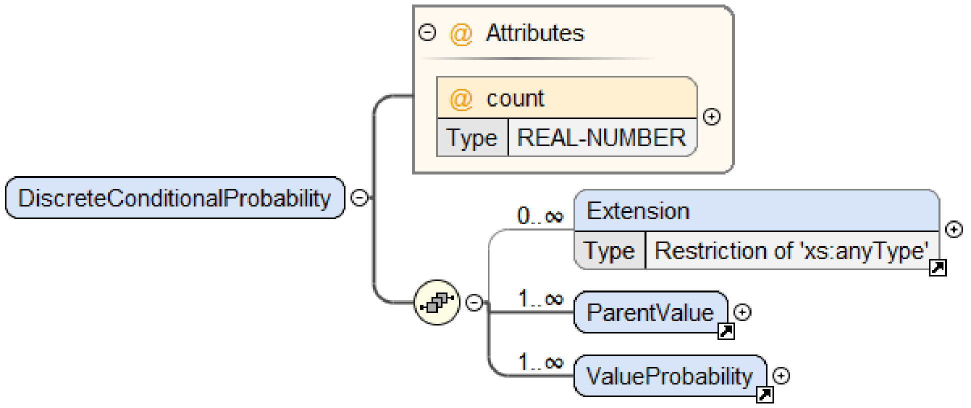


Figure 6. Schema of a *DiscreteConditionalProbability* element.

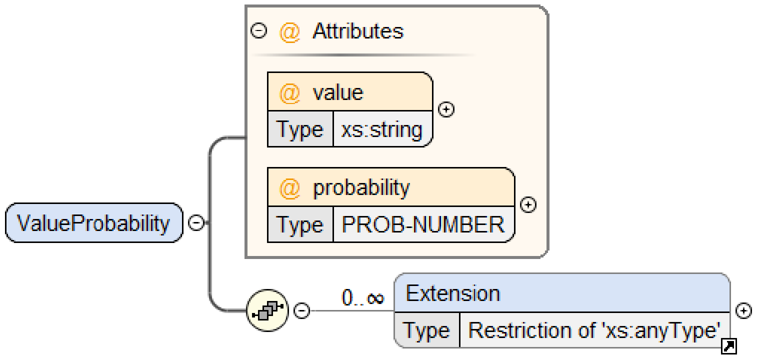


Figure 7. Schema of a *ValueProbability* element.

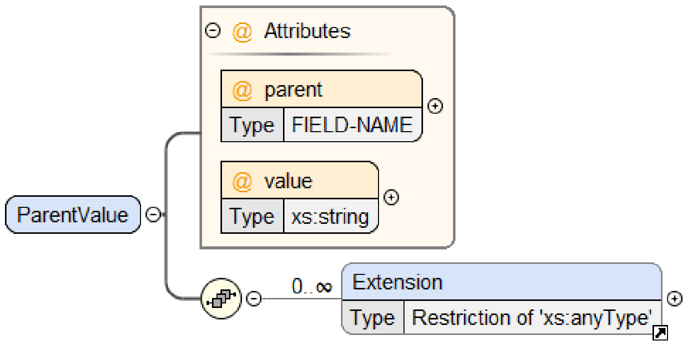


Figure 8. Schema of a *ParentValue* element.

Consider a discrete non-root node K_2 with two possible values, $K_2 = 0, 1$ dependent on two nodes (parent nodes): a discrete root node, K_1, which has two possible values, $K_1 = 0, 1$; and a continuous root node, J_1, which is defined as a uniform distribution between -1 and 1. A sample conditional probability table for K_2 can be defined as shown in Table 1. The discretization of the continuous node (see Table 1), can be represented using *DerivedField* element. In Table 1, $Pr(.)$ represents the probability function, which outputs the probability of an event. In the following subsection, we consider continuous nodes.

TABLE 1. Conditional probability table of a discrete variable with discrete and continuous parent variables.

	$K_1 = 0$ $-1 \le J_1 < 0$	$K_1 = 0$ $0 \le J_1 \le 1$	$K_1 = 1$ $-1 \le J_1 < 0$	$K_1 = 1$ $0 \le J_1 \le 1$
$Pr(K_2 = 0 \mid K_1, J_1)$	0.2	0.4	0.5	0.8
$Pr(K_2 = 1 \mid K_1, J_1)$	0.8	0.6	0.5	0.2

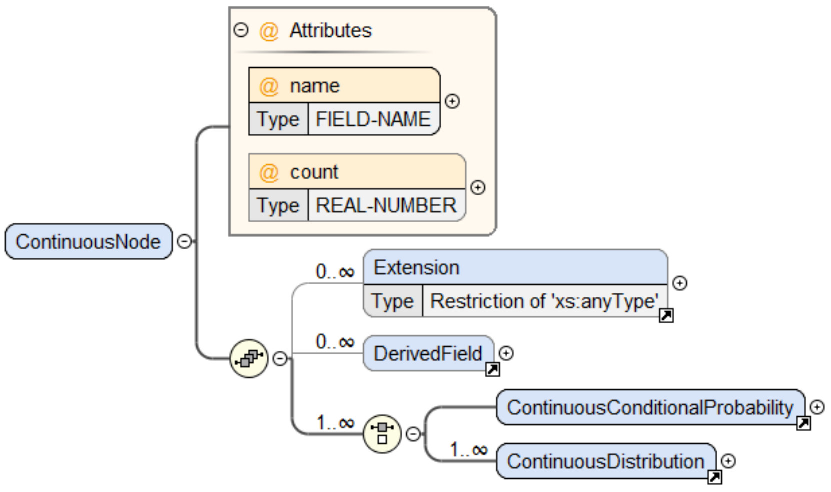


Figure 9. Schema of a *ContinuousNode* element.

3.2. ContinuousNode element

Similar to a discrete node, a continuous node is defined using the *ContinuousNode* element, as illustrated in Figure 9. The *ContinuousNode* element has the same two attributes as the *DiscreteNode* element, i.e., *name* and *count*. The name of the continuous node and the number of entries used to define the probability distribution of the continuous node are described using the *name* and *count* attributes. The probability distributions of continuous root and non-root nodes are described using *ContinuousDistribution* and *ContinuousConditionalProbability* elements, shown in Figures 10 and 11 respectively. In this paper, we provide four possible options for a continuous distribution: Normal (Gaussian), Lognormal, Uniform, and Triangular. Each distribution type has its associated distribution parameters. For example, the parameters of a Normal distribution are the mean and standard deviation whereas a Triangular distribution has three parameters: lower and upper bounds, and the mode value. For illustration, the schema elements of a Normal distribution are provided in Figures 12-14. Similarly, schema elements for the other three distribution types are defined but are not shown in this paper. Please refer to [27] for more details. The *ContinuousConditionalProbability* element is used to describe a continuous non-root node, which can have both discrete and continuous parent nodes. The *ContinuousConditionalProbability* element is defined over two other elements: *ParentValue* and *ContinuousDistribution*. The discrete parent nodes and their values are defined using the *ParentValue* element whereas the *ContinuousDistribution* element is used to define the conditional probability distributions dependent on the continuous parent nodes.

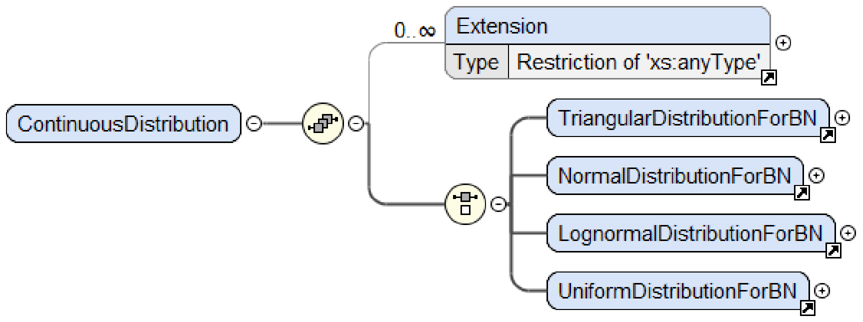


Figure 10. Schema of a *ContinuousDistribution* element.

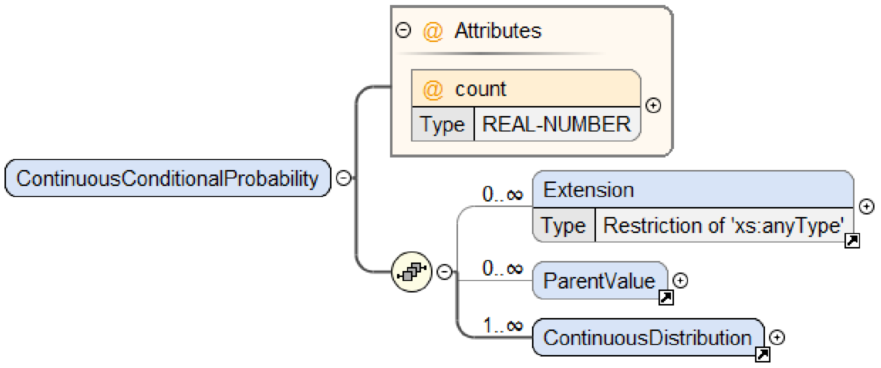


Figure 11. Schema of a *ContinuousConditionalProbability* element.

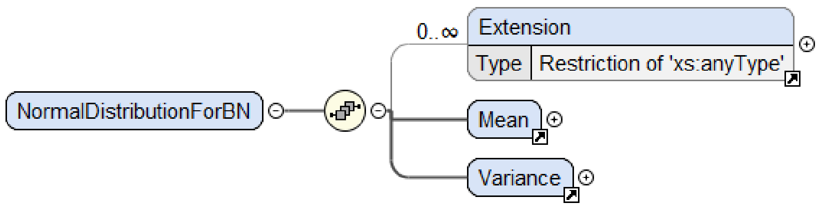


Figure 12. Schema of a *NormalDistributionForBN* element.

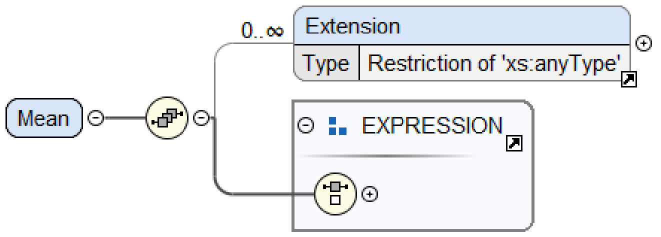


Figure 13. Schema of a *Mean* element.

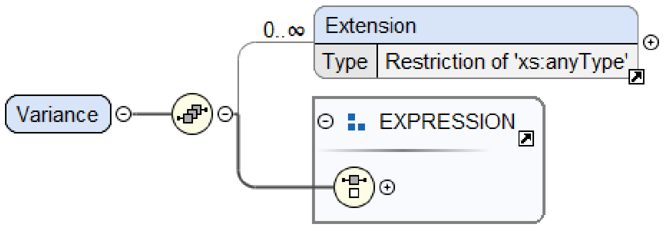


Figure 14. Schema of a *Variance* element.

Consider a continuous non-root node J_2 whose parent nodes consist of a discrete node K_1 with two possible values $K_1 = 0, 1$; and a continuous node J_1. A sample conditional distribution of J_2 can be defined as follows: $f(J_2|J_1, K_1 = 0) \sim N(3 \times J_1, 1)$ and $f(J_2|J_1, K_1 = 1) \sim N(5 \times J_1, 1)$. Here, $f(.)$ represents a probability distribution and $N(.)$ represents a Gaussian/Normal distribution. For every value of the discrete parent node ($K_1 = 0, 1$), a continuous variable is defined through a probability distribution whose parameters are dependent on the continuous parent nodes (J_1).

3.3. Representation of functional relationships between variables

As discussed in Section 2.1, one of the procedures for the construction of a BN is by using available physics-based models. Physics-based models often represent functional relationships between the inputs and the output, i.e., for given values of the inputs, the output is a deterministic quantity. Since physics-based models are affected by several uncertainty sources (such as model parameters, model form assumptions, numerical approximations), the output can be stochastic if model uncertainty is considered [48]. Such stochastic models can be represented using the techniques presented in Sections 3.1 and 3.2. In this section, we discuss the representation of such functional nodes. In the BN terminology, the inputs can be considered as parent nodes of an output (non-root node). Let X represents a non-root node, and Π_X represents its parent nodes with $X = g(\Pi_X)$, where $g(.)$ represents a functional relationship between the inputs (Π_X) and output (X).

We discuss two approaches here regarding the representation of functional nodes. The first approach approximates the functional node as a continuous variable with a Gaussian conditional probability distribution (CPD), given as $N(g(\Pi_X), \epsilon)$, i.e., the mean of the Gaussian CPD is the functional relationship between the inputs (Π_X) and output (X), and the standard deviation is equal to ϵ, a small number close to zero (e.g., the machine precision, which is in the order of 10^{-15}). A Gaussian CPD is used here; however, any continuous distribution such as lognormal or triangular can also be used by choosing their parameters appropriately. The second approach to represent a functional node is to model it as a discrete node with two states whose conditional probability table $(X|\Pi_X = \Pi_X^*)$ is defined as follows.

TABLE 2. Conditional probability table of a functional variable.

	$\Pi_X = \Pi_X^*$
$\Pr(X = g(\Pi_X^*)\|\Pi_X = \Pi_X^*)$	1
$\Pr(X \neq g(\Pi_X^*)\|\Pi_X = \Pi_X^*)$	0

Here, Π_X^* is realization of Π_X on which the conditional probability table of X is defined. The two states of X are $X = g(\Pi_X^*)$ and $X \neq g(\Pi_X^*)$. As X has a functional relationship, the probability of X being equal to $g(\Pi_X^*)$ is always equal to 1, and 0 otherwise. A primary drawback with the second approach is that the states of the output variable (X) change with the values of its parent nodes ($\Pi_X = \Pi_X^*$). For example, if $\Pi_X = \Pi_X^*$, then $g(\Pi_X^*)$ is one of the states; however, if $\Pi_X = \Pi_X^+$, then $g(\Pi_X^*)$ may not be one of the states but $g(\Pi_X^+)$ is. Such changing states cannot be captured using the schema for discrete variables discussed in Section 3.2, where the states are known and their corresponding probabilities are learnt from physics-based models, expert knowledge, or data. So, we adopt the first approach in this paper to represent a functional node.

4. BN PMML Parser

The subsequent step after the development of the BN PMML schema is the development of a parser that can translate an analytical BN model into its corresponding PMML representation following the above schema. Here, analytical BN model is defined as a BN model on which predictions can be performed as opposed to a "descriptive" BN model such as its PMML representation, on which predictions cannot be performed directly. A "descriptive" BN model only illustrates the variables present in a BN and the dependence relationships between them. One popular Python library for probabilistic programming is PyMC3, which is primarily concerned with building and sampling the posterior distributions of Bayesian models [49]. It is possible to define a BN using PyMC3, and subsequently perform sampling or include observations that will change the posterior distributions. This makes the model generated using PyMC3 an example of an analytical model. PMML functions as a language-agnostic "descriptive model", to which one would need a parser that can translate to and from analytic libraries like PyMC3. To illustrate an implementation of such a parser, a tool was developed in the Python programming language*.This example tool utilizes pyMCNet, an extension to PyMC3 that allows for the definition of BNs using graphs via the NetworkX package [50], which enables fast prototyping and visualization of BN

* https://github.com/usnistgov/pmml_pymcBN

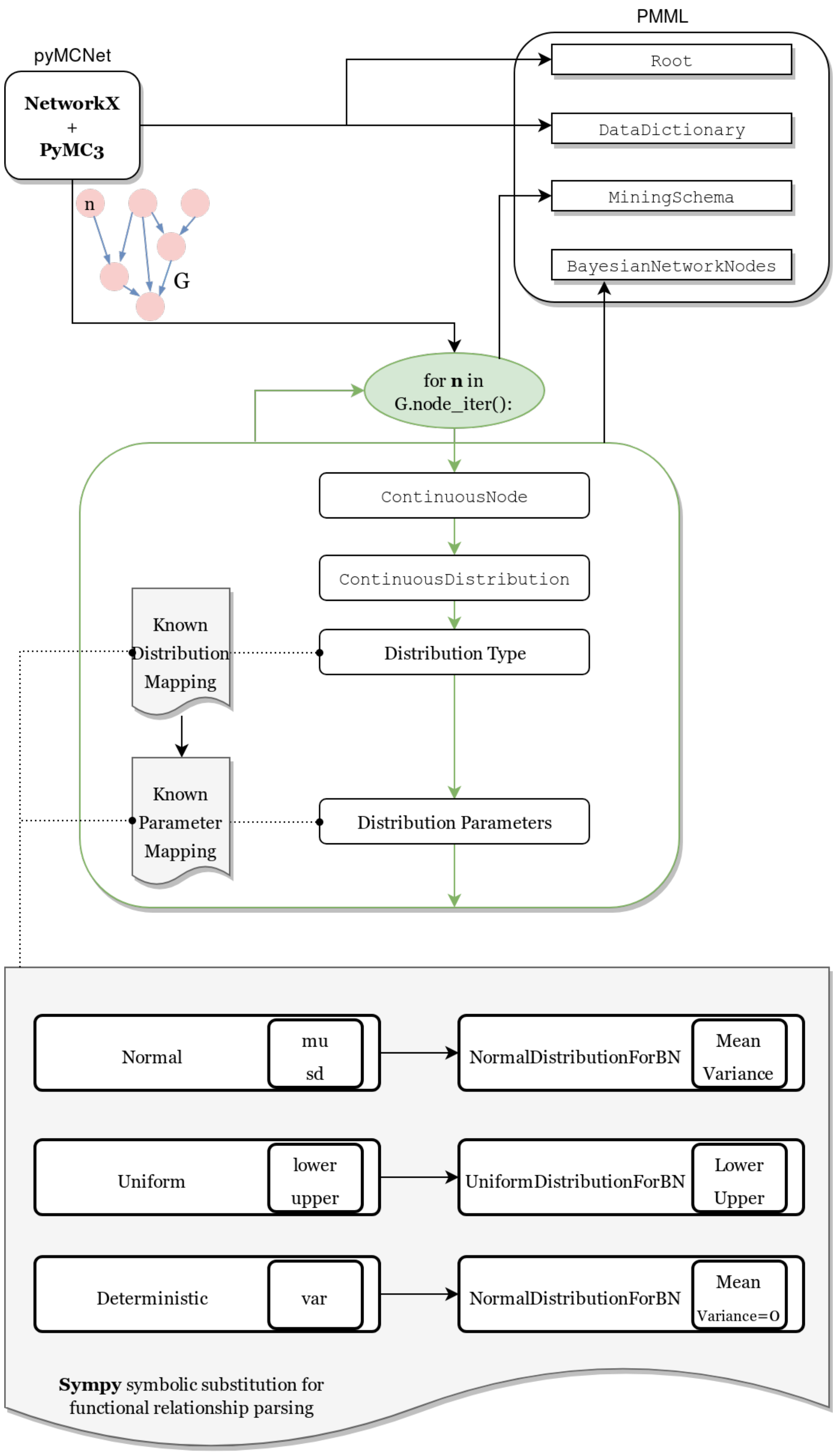


Figure 15. Flowchart illustrating the functionality of Python to PMML parser

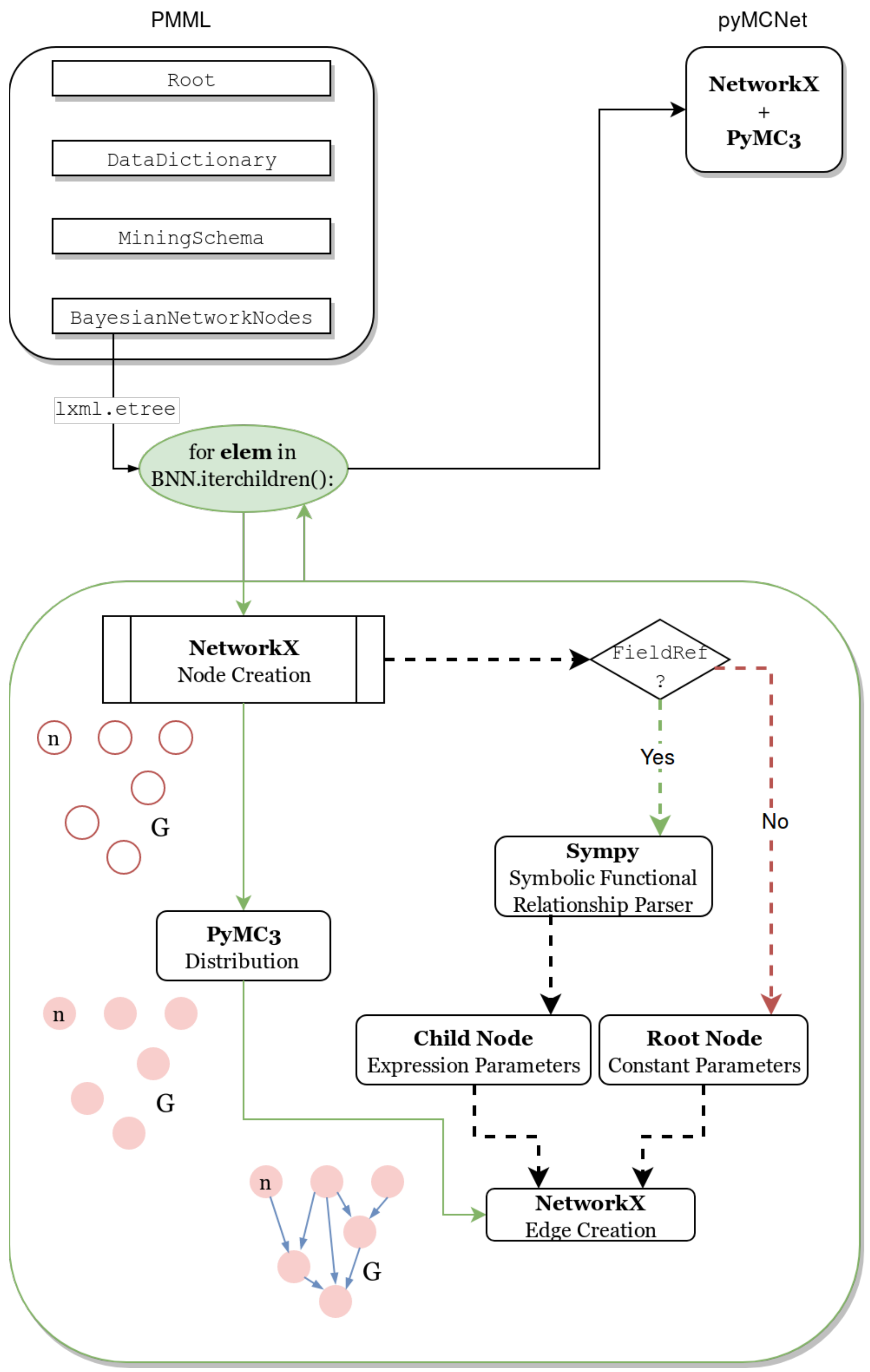


Figure 16. Flowchart illustrating the functionality of PMML to Python parser

models. Currently, only a few continuous node types are supported in this tool, though in theory any distribution allowed by PyMC3 could be added. Because the PMML schema allows BN nodes to be defined using expressions, some parsing is done using the symbolic-math library SymPy [51], to allow node expression definitions using other node values. Figures 15 and 16 provide an overview of the parser. Figure 15 illustrates the flow of transformation from Python to PMML format while Figure 16 illustrates the flow of transformation from PMML to Python. In particular, the parser can be used to convert a PyMC3 model in Python into its corresponding PMML representation and vice versa. The Python-to-PMML direction maps the names of XML node definitions to known PyMC3 random variables, using Sympy to parse functional node relationships, which then populates a pyMCNet graph. The PMML-to-Python direction maps nodes from PMML to PyMC3 distributions and their parameters, from which the parameter functional relationships (FieldRef in PMML [27]) are used to add edges in the pyMCNet graph.

5. Case study: Welding process

In this section, we demonstrate the BN PMML representation for the energy prediction of a welding process and its parsing into an analytical model. Welding represents a fabrication process of joining two metal parts to form a single part and is one of the fundamental processes in manufacturing. There are different types of welding processes, i.e., different ways to join two metal parts such as electric arc welding and torch welding [52]. Electric arc welding using an electric arc while torch welding uses an oxyacetylene torch to melt and join the metal parts. Energy efficiency represents a key metric for sustainability evaluation of a manufacturing process [53]. An approach for energy efficiency evaluation is the comparison of theoretical energy consumption and the real-world energy consumption of the welded products. The theoretical energy computation is greatly affected by several uncertainty sources such as the parameters of the welding process; therefore, their identification and quantification are necessary for a comprehensive sustainability evaluation. In the case study below, we describe the arc welding process and equations for computing the theoretically minimum energy consumption, identify the uncertain parameters, and connect them to the overall energy using a BN. We construct the associated BN and represent it using the BN PMML standard or schema. In the presence of any process data, the process parameters are updated using Bayes theorem; these updated parameters are used to update the minimum energy value, which can later be used for sustainability evaluation in the presence of experimental energy values.

5.1. Process description

Consider the cross-section of the weld between two metal pieces as shown in Figure 17 [16]. If L represents the length of the weld, l, h, g, t and e represent the weld parameters, as shown in Figure 16, then the overall volume of the weld, V, can be calculated as shown in Eq. 8.

$$V = L \times (0.75 \times l \times h + g \times t + 0.5 \times (l - g) \times (t - e)) \tag{8}$$

Assuming that the metal and the filler are of the same material, the minimum theoretical energy required for the welding process is given as

$$E = \rho \times V \times (C_p \times (T_f - T_i) + H) \tag{9}$$

where ρ, C_p, T_f, T_i, and H represent the density of the welding material, heat capacity, and final and initial temperatures of the weld and the latent heat, respectively. Using the above equations, a BN showing the dependence relationships for the energy prediction of a welding process is shown in Figure 18.

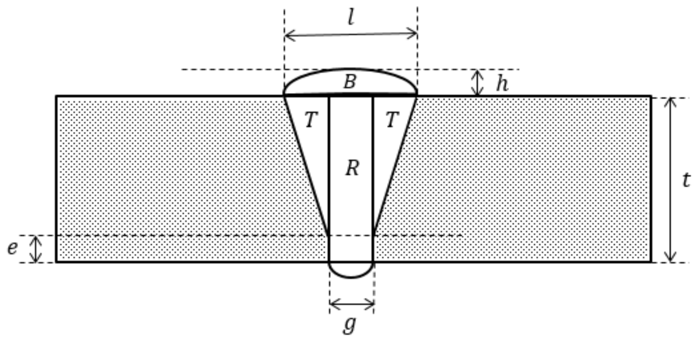


Figure 17. Cross-section of the weld showing the welding parameters [16].

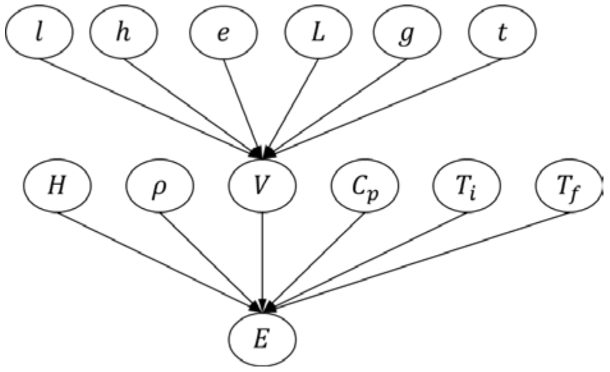


Figure 18. BN for energy prediction of welding process using physics-based models [16].

For illustration purposes, this example considered a physics-based deterministic model for the description of energy consumption in the welding process. In some cases, a physics-based model may not accurately quantify the energy consumption, due to several uncertainty sources such as model form assumptions, unknown model parameters, and numerical approximations. We refer the reader to the work of Nannapaneni and Mahadevan [48] for more details regarding the representation and quantification of model uncertainty sources. Section 2 discussed three ways to construct a BN – physics models, data-driven, or their combination. This example illustrates the BN construction using available physics-based models. The parameters (L, t, e, H, C_p, T_i and T_f) are assumed to be known while the parameters (l, g and h) are assumed to be uncertain and quantified using Gaussian distributions whose parameters are unknown. However, prior knowledge regarding the distribution parameters are assumed available. In addition, the density of the weld material is not known precisely but prior knowledge is assumed available. The values of all the known and unknown parameters are given in Table 3. The BN of the welding process, after removing all the known variables in Figure 18 and adding nodes for the distribution parameters of $l, g,$ and h, is given in Figure 19.

In Figure 19, green circular, yellow circular and blue squared represent uniform, normal, and deterministic nodes respectively. Nodes are either observed (grayed-out) or unobserved. The edge variable indicated the type of node relationship, either a deterministic/functional relationship ("x"), or a stochastic (in this case, location parameter "μ" or scale parameter "σ"). Details of the BN construction and inference regarding the welding process were discussed in [16]. To model sensor uncertainty, the Energy node is split into a deterministic node E_d (described as E above in Figure 19), and a stochastic node E_L with uncertainty that models the likelihood of our sensor data. It should be noted that the volume (V) and Energy (E) are functional nodes, i.e., given the values of the parent nodes, the values of V and E are known deterministically due to the presence of functional relationships as shown in Eqs. 8 and 9. Following the procedure for the representation of functional nodes in Section 3.3, V and E are represented as discrete nodes, each with two states dependent on the values of their parent nodes.

TABLE 3. Welding process parameters.

Parameter	Value
$L(mm)$	500
$t(mm)$	15
$e(mm)$	11
$H(kJ/kg)$	270
$C_p(kJ/kg.K)$	0.5
$T_i(K)$	300
$T_f(K)$	1600
l	$N(\mu_l, \sigma_l)$
μ_l	$U(8.3,8.6)$
σ_l	$U(0.2,0.7)$
g	$N(\mu_g, \sigma_g)$
μ_g	$U(1.6,2.2)$
σ_g	$U(0.05,0.2)$
h	$N(\mu_h, \sigma_h)$
μ_h	$U(2.5,2.8)$
σ_h	$U(0.3,0.6)$
$\rho(kg/m^3)$	$N(8250,10)$

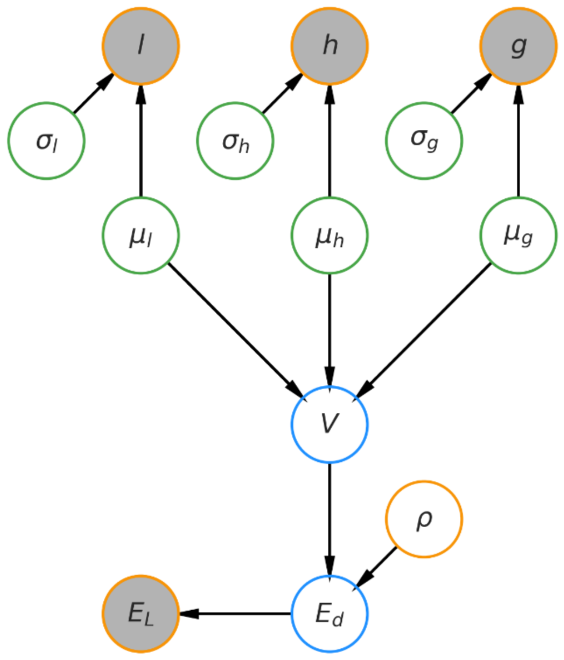


Figure 19. BN for energy prediction of a welding process [16].

5.2. PMML Representation Parsing and Prediction

The PMML representation of the BN for welding process using the BN PMML schema described in Section 3 is given in Figure 20, where scheme elements such as MiningSchema, and DataFields are not shown, and indicated with a (…), for brevity. Note the static node mu_l, the deterministic node “E_d”, and the stochastic node “E_L” having a functional relationship with “E_d”. The PMML representation of the BN for the welding process described using the BN PMML schema can now be shared with whomever necessary for their computational requirements without rebuilding the BN. After transferring the BN PMML file of the welding process, the Python parser described in Section 4 is used to convert the PMML file into an analytical model in Python using the PyMC3 package [49]. For ease in notation, we term the environment where the BN PMML representation of the welding process is created as the training environment and the environment where this PMML file is used for prediction as testing environment following the notation in [28].

Let 100 data points be available in the testing environment on the parameters $l, h, g,$ and E_L collected through sensors. The sensor uncertainty in the measurement of l, h and g is assumed to be quantified using a Gaussian distribution with zero mean and standard deviation of 0.1 mm. Similarly, the sensor uncertainty in the energy measurement is quantified using a Gaussian distribution with zero mean and a standard deviation of 1 kJ. A Gaussian distribution with zero mean is used to model the sensor uncertainty as both the positive and negative values occur with equal probability due to the symmetric nature of a Gaussian distribution. This allows E_L itself to be modeled as a Gaussian, namely, $E_L \sim N(E_d, 1)$.

```
<PMML version="4.3" xmlns="http://www.dmg.org/PMML-4_3">

  <Header copyright="DMG.org" description="Bayesian Network Model"/>
  <DataDictionary numberOfFields="13">
    <DataField dataType="double" name="mu_l" optype="continuous"/>
       ...
    <DataField dataType="double" name="E_L" optype="continuous"/>
  </DataDictionary>
  <BayesianNetworkModel modelName="Bayesian Network Model" functionName="regression">
    <MiningSchema>
      <MiningField name="l" usageType="active"/>
       ...
      <MiningField name="E_d" usageType="target"/>
    </MiningSchema>
    <BayesianNetworkNodes>
      <ContinuousNode name="mu_l">
        <ContinuousDistribution>
          <UniformDistributionForBN>
            <Lower>
              <Constant dataType="double">0.0083</Constant>
            </Lower>
            <Upper>
              <Constant dataType="double">0.0086</Constant>
            </Upper>
          </UniformDistributionForBN>
        </ContinuousDistribution>
      </ContinuousNode>
      ...
      <ContinuousNode name="E_d">
        <ContinuousDistribution>
          <NormalDistributionForBN>
            <Variance>
              <Constant dataType="double">0.0</Constant>
            </Variance>
            <Mean>
              <Apply function="*">
                <Constant dataType="double">920.000000000000</Constant>
                <FieldRef field="V"/>
                <FieldRef field="rho"/>
              </Apply>
            </Mean>
          </NormalDistributionForBN>
        </ContinuousDistribution>
      </ContinuousNode>
      <ContinuousNode name="E_L">
        <ContinuousDistribution>
          <NormalDistributionForBN>
            <Mean>
              <FieldRef field="E_d"/>
            </Mean>
            <Variance>
              <Constant dataType="double">1.00000000000000</Constant>
            </Variance>
          </NormalDistributionForBN>
        </ContinuousDistribution>
      </ContinuousNode>
    </BayesianNetworkNodes>
  </BayesianNetworkModel>
</PMML>
```

Figure 20. PMML instance of the described welding BN.

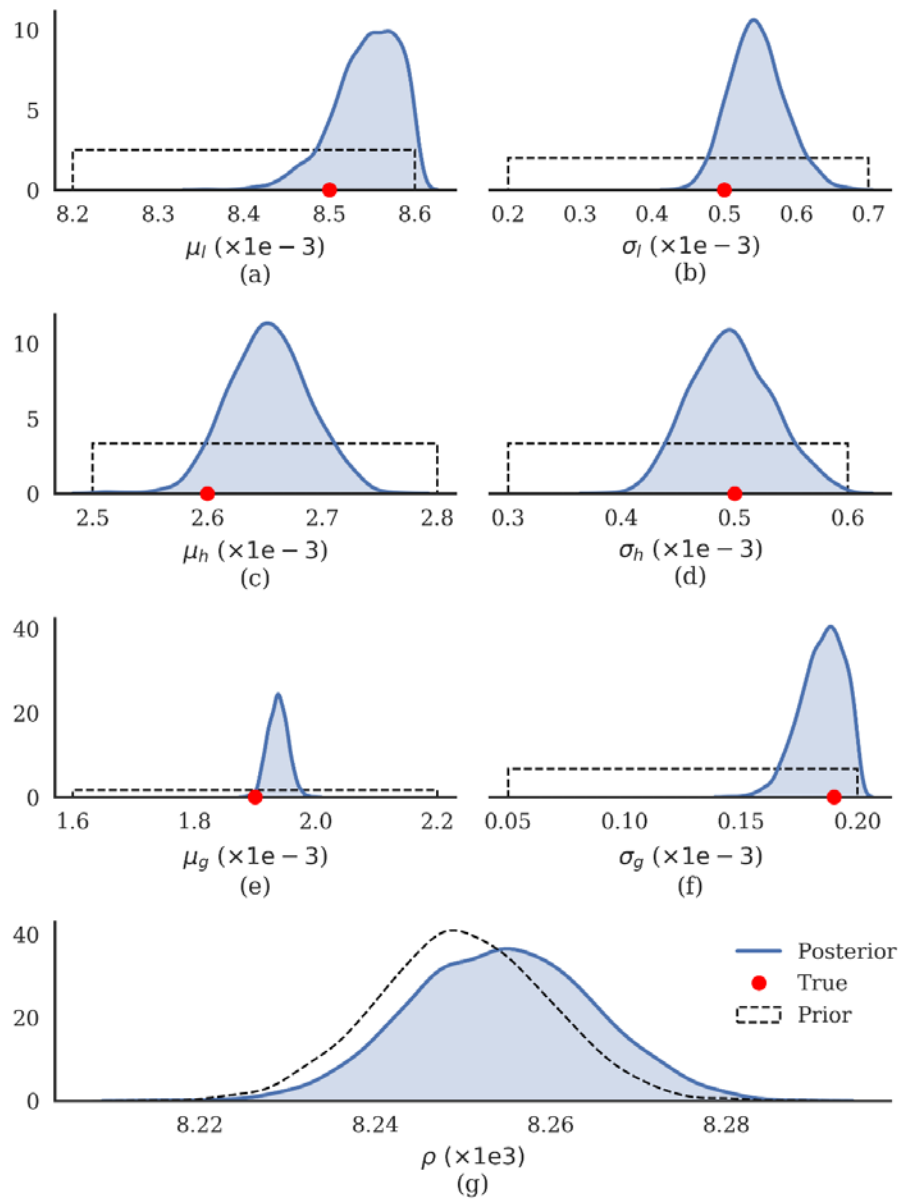


Figure 21. Comparison of the prior and posterior distributions for the inference parameters (a) mean of parameter 'l', (b) standard deviation of 'l', (c) mean of parameter 'h', (d) standard deviation of parameter 'h', (e) mean of parameter 'e', (f) standard deviation of parameter 'e', and (g) density.

The objective is to estimate the distribution parameters μ, σ (mean and standard deviation) for dimension variables $l. h$ and g, along with our belief of the weld filler material density (ρ); this is performed using the Bayes' theorem as

$$f\left(\mu_l, \sigma_l, \mu_h, \sigma_l, \mu_g, \sigma_g, \rho \middle| l_{obs}, h_{obs}, g_{obs}, E_{obs}\right) \\ \propto f\left(l_{obs}, h_{obs}, g_{obs}, E_{obs} \middle| \mu_l, \sigma_l, \mu_h, \sigma_l, \mu_g, \sigma_g, \rho\right) f\left(\mu_l, \sigma_l, \mu_h, \sigma_l, \mu_g, \sigma_g, \rho\right) \tag{9}$$

The No-U-Turn Sampling algorithm (NUTS), which uses adaptive Hamiltonian Monte Carlo for efficient sampling [54], is used for inference; the prior and posterior distributions along with the true value used to generate synthetic data, are shown in Figure 21. A total of 80,000 samples were generated using the NUTS algorithm and the last 40,000 are used for constructing the posterior distributions. The first 40,000 samples are ignored (burn-in) to account for the convergence of Markov chain of samples [41]. It can be observed that the variance of posterior distributions of the dimensional parameters (l, g, h) have reduced due to the observation data. The variance reduction in the density (ρ) is not significant; this can be attributed to the insensitivity of the observed parameters to the energy consumption as illustrated in our previous work using variance-based global sensitivity analysis [16].

6. Conclusion and Future Work

This paper presents the Predictive Model Markup Language (PMML) representation of a generic Bayesian network (BN), which may contain discrete (categorical) variables, continuous variables, or their combination. BNs are probabilistic acyclic graphical models, which have been studied for a variety of applications such as uncertainty quantification, design optimization under uncertainty, risk analysis, and quality control. BNs represent a joint probability distribution over a set of random variables through a combination of marginal and conditional distributions. The BN PMML schema accommodates both discrete variables and continuous variables described by Normal (Gaussian), Lognormal, Uniform, and Triangular distributions. The availability of such a PMML schema helps the exchange of BN models across the PMML compliant software platforms; this enables industry practitioners to use the models directly without rebuilding their own, and hence saving valuable time and computational resources. PMML schema-based models are descriptive in nature; i.e., they cannot directly be used for prediction or inference. Therefore, a parser was developed that allows converting a descriptive BN PMML model into an analytical model in Python using the PyMC3 and NetworkX packages. The analytical model is then used to perform predictions using Markov Chain Monte Carlo methods. In addition, the developed parser also enables the conversion of an analytical model in Python into its corresponding PMML representation. Thus, the parser enables conversion in both directions: from descriptive PMML platform into an analytical model in Python and vice versa. In this paper, we demonstrate the developed methods for PMML representation and parsing for a BN related to a welding process.

Future work should accommodate other parametric distribution types, such as Beta, Multinomial, Dirichlet, and Exponential, and non-parametric distributions, such as kernel density estimations to represent both discrete and continuous variables. In this work, a functional node is approximated using a Gaussian conditional probability distribution with the mean equal to the functional relationship and a small standard deviation, which is in the order of the machine precision. This approach creates errors in the computation; therefore, future work should consider effective representation of functional (deterministic) nodes in the BN PMML schema. In addition, PMML schema for variants of BNs such as dynamic BNs and hierarchical BNs need to be considered. Future work will also be aimed at adding support for generic BNs in the Portable Format for Analytics (PFA) standard. Since PFA offers standard ways to procedurally specify data manipulation algorithms, it offers the possibility of describing advanced inference algorithms within the standard model in a tool independent way, which is currently not possible with PMML.

Acknowledgements

The research reported in this paper is supported by funds from the National Institute of Standards and Technology (NIST) under cooperative agreements with Vanderbilt University (Grant No. 70NANB16H297) and University of Maryland (Grant No. 70NANB14H250), and by the NIST Foreign Guest Researcher Program. The support is gratefully acknowledged.

Disclaimer

Certain commercial systems are identified in this paper. Such identification does not imply recommendation or endorsement by NIST; nor does it imply that the products identified are necessarily the best available for the purpose. Further, any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NIST or any other supporting U.S. government or corporate organizations.

References

[1] Dubey, R., Gunasekaran, A., Childe, S. J., Wamba, S. F., and Papadopoulos, T., 2016, “The Impact of Big Data on World-Class Sustainable Manufacturing,” Int. J. Adv. Manuf. Technol., **84**(1–4), pp. 631–645.

[2] Kang, H. S., Lee, J. Y., Choi, S., Kim, H., Park, J. H., Son, J. Y., Kim, B. H., and Noh, S. Do, 2016, “Smart Manufacturing: Past Research, Present Findings, and Future Directions,” Int. J. Precis. Eng. Manuf. - Green Technol., **3**(1), pp. 111–128.

[3] Asiltürk, İ., and Çunkaş, M., 2011, “Modeling and Prediction of Surface Roughness in Turning Operations Using Artificial Neural Network and Multiple Regression Method,” Expert Syst. Appl., **38**(5), pp. 5826–5832.

[4] Ak, R., Helu, M. M., and Rachuri, S., 2015, “Ensemble Neural Network Model for Predicting the Energy Consumption of a Milling Machine,” *20th Design for Manufacturing and the Life Cycle Conference*, Boston, MA.

[5] Lee, J., Wu, F., Zhao, W., Ghaffari, M., Liao, L., and Siegel, D., 2014, “Prognostics and Health Management Design for Rotary Machinery Systems - Reviews, Methodology and Applications,” Mech. Syst. Signal Process., **42**(1–2), pp. 314–334.

[6] Zhao, R., Wang, D., Yan, R., Mao, K., Shen, F., and Wang, J., 2017, “Machine Health Monitoring Using Local Feature-Based Gated Recurrent Unit Networks,” IEEE Trans. Ind. Electron., **65**(2), pp. 1539–1548.

[7] Jardine, A. K. S., Lin, D., and Banjevic, D., 2006, “A Review on Machinery Diagnostics

and Prognostics Implementing Condition-Based Maintenance," Mech. Syst. Signal Process., **20**(7), pp. 1483–1510.

[8] Jayal, A. D., Badurdeen, F., Dillon, O. W., and Jawahir, I. S., 2010, "Sustainable Manufacturing: Modeling and Optimization Challenges at the Product, Process and System Levels," CIRP J. Manuf. Sci. Technol., **2**(3), pp. 144–152.

[9] Nannapaneni, S., and Mahadevan, S., 2016, "Manufacturing Process Evaluation Under Uncertainty: A Hierarchical Bayesian Network Approach," *Proceedings of the ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, p. V01BT02A026-V01BT02A026.

[10] Bhinge, R., Biswas, N., Dornfeld, D., Park, J., Law, K. H., Helu, M., and Rachuri, S., 2014, "An Intelligent Machine Monitoring System for Energy Prediction Using a Gaussian Process Regression," *2014 IEEE International Conference on Big Data (Big Data)*, pp. 978–986.

[11] Guazzelli, A., Stathatos, K., and Zeller, M., 2009, "Efficient Deployment of Predictive Analytics through Open Standards and Cloud Computing," ACM SIGKDD Explor. Newsl., **11**(1), p. 32.

[12] Guazzelli, A., Zeller, M., Lin, W., and Williams, G., 2009, "PMML: An Open Standard for Sharing Models," R J., **1**(1), pp. 60–65.

[13] Pivarski, J., Bennett, C., and Grossman, R. L., 2016, "Deploying Analytics with the Portable Format for Analytics (PFA)," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pp. 579–588.

[14] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F., 1997, "Extensible Markup Language (XML)," World Wide Web J., **2**(4), pp. 27–66.

[15] "The Data Mining Group (DMG)" [Online]. Available: http://dmg.org. [Accessed: 10-Jul-2018].

[16] Nannapaneni, S., Mahadevan, S., and Rachuri, S., 2016, "Performance Evaluation of a Manufacturing Process under Uncertainty Using Bayesian Networks," J. Clean. Prod., **113**, pp. 947–959.

[17] Correa, M., Bielza, C., Ramirez, M. D. J., and Alique, J. R., 2008, "A Bayesian Network Model for Surface Roughness Prediction in the Machining Process," Int. J. Syst. Sci., **39**(12), pp. 1181–1192.

[18] Tobon-Mejia, D. A., Medjaher, K., and Zerhouni, N., 2012, "CNC Machine Tool's Wear Diagnostic and Prognostic by Using Dynamic Bayesian Networks," Mech. Syst. Signal Process., **28**, pp. 167–182.

[19] Nannapaneni, S., Mahadevan, S., and Dubey, A., 2018, "Real-Time Control of a Cyber-Physical Manufacturing Process under Uncertainty," *ASME 2018 International Manufacturing Science and Engineering Conference*, College Station, Texas, USA.

[20] Kurz, D., Kaspar, J., and Pilz, J., 2011, "Dynamic Maintenance in Semiconductor Manufacturing Using Bayesian Networks," *Automation Science and Engineering (CASE),*

2011 IEEE Conference on, pp. 238–243.

[21] Chan, A., and McNaught, K. R., 2008, “Using Bayesian Networks to Improve Fault Diagnosis during Manufacturing Tests of Mobile Telephone Infrastructure,” *Journal of the Operational Research Society*, pp. 423–430.

[22] Masruroh, N. A., and Poh, K. L., 2007, “A Bayesian Network Approach to Job-Shop Rescheduling,” *IEEM 2007: 2007 IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 1098–1102.

[23] Kao, H.-Y., Huang, C.-H., and Li, H.-L., 2005, “Supply Chain Diagnostics with Dynamic Bayesian Networks,” Comput. Ind. Eng., **49**(2), pp. 339–347.

[24] Zhu, J. Y., and Deshmukh, A., 2003, “Application of Bayesian Decision Networks to Life Cycle Engineering in Green Design and Manufacturing,” Eng. Appl. Artif. Intell., **16**(2 SPEC.), pp. 91–103.

[25] “Predictive Model Markup Language (PMML) v4.2.1” [Online]. Available: http://dmg.org/pmml/v4-2-1/GeneralStructure.html. [Accessed: 10-Jul-2018].

[26] Friedman, N., Geiger, D., and Goldszmidt, M., 1997, “Bayesian Network Classifiers,” Mach. Learn., **29**(2–3), pp. 131–163.

[27] “PMML 4.3 - General Structure” [Online]. Available: http://dmg.org/pmml/v4-3/GeneralStructure.html. [Accessed: 10-Jul-2018].

[28] Park, J., Lechevalier, D., Ak, R., Ferguson, M., Law, K., Lee, Y., and Rachuri, S., 2017, “Gaussian Process Regression (GPR) Representation in Predictive Model Markup Language (PMML),” Smart Sustain. Manuf. Syst., **1**(1), pp. 121–141.

[29] Nannapaneni, S., Mahadevan, S., Dubey, A., Lechevalier, D., Narayanan, A., and Rachuri, S., 2017, “Automated Uncertainty Quantification through Information Fusion in Manufacturing Processes,” Smart Sustain. Manuf. Syst., **1**(1), pp. 153–177.

[30] Scutari, M., 2010, “Learning Bayesian Networks with the Bnlearn R Package,” J. Stat. Softw., **35**(3), pp. 1–22.

[31] Bartram, G., and Mahadevan, S., 2014, “Integration of Heterogeneous Information in SHM Models,” Struct. Control Heal. Monit., **21**(3), pp. 403–422.

[32] Neapolitan, R. E., 2004, *Learning Bayesian Networks*, Pearson Prentice Hall.

[33] Spirtes, P., Glymour, C., and Scheines, R., 1993, *Causation, Prediction, and Search*, Springer-Verlag, New York.

[34] Margaritis, D., 2003, “Learning Bayesian Network Model Structure from Data,” Carnegie Mellon University.

[35] Tsamardinos, I., Aliferis, C. F., and Statnikov, A., 2003, “Algorithms for Large Scale Markov Blanket Discovery,” *The 16th International Florida Artificial Intelligence Research Society Conference*, American Associ ation for Artificial Intelligence, St. Augustine, Florida, pp. 376–380.

[36] Yaramakala, S., and Margaritis, D., 2005, “Speculative Markov Blanket Discovery for

Optimal Feature Selection," *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 809–812.

[37] Tsamardinos, I., Brown, L. E., and Aliferis, C. F., 2006, "The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm," Mach. Learn., **65**(1), pp. 31–78.

[38] Campos, C. P. de, Tong, Y., and Ji, Q., 2008, "Constrained Maximum Likelihood Learning of Bayesian Networks for Facial Action Recognition," *Computer Vision – ECCV 2008*, Springer, Berlin, Heidelberg, pp. 168–181.

[39] Ling, Y., and Mahadevan, S., 2013, "Quantitative Model Validation Techniques: New Insights," Reliab. Eng. Syst. Saf., **111**, pp. 217–231.

[40] Simonsson, I., and Mostad, P., 2016, "Exact Inference on Conditional Linear Γ-Gaussian Bayesian Networks," *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*, pp. 474–486.

[41] Hastings, W. K., 1970, "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," Biometrika, **57**(1), pp. 97–109.

[42] Fox, C. W., and Roberts, S. J., 2012, "A Tutorial on Variational Bayesian Inference," Artif. Intell. Rev., **38**(2), pp. 85–95.

[43] Smith, A. F. M., and Gelfand, A. E., 1992, "Bayesian Statistics without Tears: A Sampling-Resampling Perspective," Am. Stat., **46**(2), pp. 84–88.

[44] Csillery, K., Blum, M., Gaggiotti, O., and Francois, O., 2010, "Approximate Bayesian Computation (ABC) in Practice," Trends Ecol. Evol., **25**(7), pp. 410–418.

[45] Li, C., and Sankaran, M., 2018, "Efficient Approximate Inference in Bayesian Networks with Continuous Variables," Reliab. Eng. Syst. Saf., **169**, pp. 269–280.

[46] "W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures" [Online]. Available: https://www.w3.org/TR/xmlschema11-1/. [Accessed: 10-Jul-2018].

[47] Pechter, R., 2009, "What's PMML and What's New in PMML 4.0?," ACM SIGKDD Explor. Newsl., **11**(1), p. 19.

[48] Nannapaneni, S., and Mahadevan, S., 2016, "Reliability Analysis under Epistemic Uncertainty," Reliab. Eng. Syst. Saf., **155**, pp. 9–20.

[49] Salvatier, J., Wiecki, T. V., and Fonnesbeck, C., 2016, "Probabilistic Programming in Python Using PyMC3," PeerJ Comput. Sci., **2**, p. e55.

[50] Hagberg, A. A., Schult, D. A., and Swart, P. J., 2008, "Exploring Network Structure, Dynamics, and Function Using NetworkX," *Proceedings of the 7th Python in Science Conference*, pp. 11–16.

[51] Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, Am., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, Š., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., and Scopatz, A., 2017, "SymPy: Symbolic Computing in Python," PeerJ Comput. Sci., **3**, p. e103.

[52] Weman, K., 2012, *Welding Processes Handbook*, Woodhead Publishing.

[53] Mani, M., Madan, J., Lee, J. H., Lyons, K. W., and Gupta, S. K., 2014, “Sustainability Characterisation for Manufacturing Processes,” Int. J. Prod. Res., **52**(20), pp. 5895–5912.

[54] Hoffman, M. D., and Gelman, A., “The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo,” J. Mach. Learn. Res., **15**, pp. 1593–1623.