GAUSSIAN PROCESS REGRESSION (GPR) REPRESENTATION IN PREDICTIVE MODEL MARKUP LANGUAGE (PMML)

Jinkyoo Park[1], *David Lechevalier*[2], *Ronay Ak*[3], *Max Ferguson*[4], *Kincho Law*[4], *Yung-Tsun Tina Lee*[3], *and Sudarsan Rachuri*[5]

Abstract: This paper describes Gaussian Process Regression (GPR) models presented in Predictive Model Markup Language (PMML). PMML is an Extensible Markup Language (XML)-based standard language used to represent data mining and predictive analytic models, as well as pre- and post-processed data. The previous PMML version, PMML 4.2, did not provide capabilities for representing probabilistic (stochastic) machine learning algorithms that are widely used for constructing predictive models taking the associated uncertainties into consideration. The newly released PMML version 4.3, which includes GPR model, provides new features - confidence bounds and distribution for the predictive estimations. Both features are needed to establish the foundation for uncertainty quantification analysis. Among various probabilistic machine learning algorithms, GPR has been widely used for approximating a target function due to its capability of representing a complex input and output relationships without predefining a set of basis functions, and predicting a target output with uncertainty quantification. GPR is being employed to various manufacturing data analytics applications, which necessitates representing this model in a standardized form for easy and rapid employment. In this paper, we present a GPR model and its representation in PMML. Furthermore, we demonstrate a prototype using a real data set in the manufacturing domain.

Keywords: PMML, Gaussian Process Regression, Predictive Analytics, Data Mining, Standards, XML.

[1]Korea Advanced Institute of Science and Technology (KAIST), Dept. of Industrial and Systems Engineering, Daejeon 34141, Republic of Korea
[2]Université de Bourgogne, Laboratoire d'Electronique, Informatique et Image (Le2i), Dijon 21000, France
[3]National Institute of Standards and Technology (NIST), Engineering Lab, Gaithersburg, MD 20899
[4]Stanford University, Department of Civil and Environmental Engineering, Stanford, CA 94305-4020
[5]Department of Energy (DOE), Advanced Manufacturing Office, Washington, DC 20585

1. Introduction

The last decade has seen an explosion in the application and research of machine learning [1] across different industries. The number of data sources, computing environments, and machine learning tools continues to increase. Continuous improvements in sensor technologies, data acquisition systems, and data-mining and big data-analytics techniques allow the industries to effectively and efficiently collect large, rapid, and diverse volumes of data and get valuable insights from this data. The manufacturing industry, which is vital to all economies, is one of the domains experiencing a never seen increase in available data collected from each stage of the product life cycle (design, production, and post-production). The availability of large volume of data has given strong impetus to data-driven decision making.

Given the specific nature of manufacturing systems being dynamic, uncertain, and complex, to overcome some of today's major challenges of complex manufacturing systems, valid candidates are machine learning techniques [2]. These data-driven approaches are able to find highly complex and non-linear patterns in data of different types and sources and transform raw data to features spaces, so-called models, which are then applied for prediction [3], detection and classification [4, 5], or forecasting [6] of a quantity of interest (QoI) for a specific problem of the manufacturing process or system. A prediction model must be capable of providing a quantification of the uncertainty associated with the prediction for informed decision-making. However, majority of the existing statistical techniques provide point predictions [3-6] without considering that predictions can be affected by uncertainties in the model parameters and input data. Bhinge et al. [7] and Park et al. [8] used the Gaussian Process (GP) to build a nonlinear regression model that predicts the energy usage of a milling machine. The GP models calculate the complex relationship between the input machining parameters and output energy consumption, and construct a prediction function for the energy consumption with confidence bounds [7, 8]. Using the similar data set collected at the UC Berkeley Mechanical Engineering Machine Shop [9], Ak et al. [10] built an empirical prediction model using an ensemble Neural Networks (NNs) approach to estimate the energy consumption for a computer numerical control (CNC) milling machine tool. In addition to pure data-driven techniques, Nannapaneni et al. [11] proposed a hybrid framework using Bayesian Networks to aggregate the uncertainty from multiple sources for the purpose of uncertainty quantification (UQ) in the prediction of performance of a manufacturing process. The mathematical equations are used to calculate the conditional probability relationships between parent and child nodes in the network. The work focuses on both data and model uncertainties. The proposed uncertainty-evaluation methodology is demonstrated using two manufacturing processes: welding and injection molding.

For the manufacturing industry, in addition to the need for continuous development of the advanced analytics techniques, open standards, communication protocols, or best practices are also needed to effectively and consistently deploy methodologies and technologies in order to

assess process performance [12]. In this context, the rapid growth of data analytics and the ever-increasing complexity of machine learning models has made the development and adoption of predictive model standards a necessity. There are evolving data analytics standards that support computational modeling (e.g., data analytics and simulation) and lay foundations for modeling and integrating manufacturing systems and related services for continuous improvement within the enterprise [12]. To be widely adopted, these standards must support a wide range of machine learning models including non-parametric probabilistic models such as Gaussian Process Regression (GPR).

The Predictive Model Markup Language (PMML) is a de facto standard used to represent data mining and predictive analytic models [13-16]. It is developed by the Data Mining Group (DMG), a consortium of commercial and open-source data mining companies, and is supported by many statistical tools such as R [14], ADAPA [16], SAS [17] and Python [18]. PMML is an XML-based language; thus all PMML documents must conform to the XML standard and the structure of the PMML is defined by an XML Schema [13-15]. The Schema describes the general structure of PMML documents as well as model-specific elements. The PMML standard can be used to represent both predictive and descriptive models, as well as pre-processing and post-processing transformations. Users can easily share the analytic models represented in PMML between different PMML-compliant applications. The models can be exchanged freely between computing environments, promoting interoperability across programming languages, and physical devices. One can train a model in a local computing environment, express the trained model as a PMML document, and then deploy the PMML document to a scoring engine to perform predictions for a new unseen data set. In addition to PMML, there exists similar standards such as Portable Format for Analytics (PFA) [13], which is an emerging standard for statistical models and data transformation engines.

Although the previous PMML version, PMML 4.2, covers many data mining algorithms, it does not provide capabilities for representing probabilistic machine learning algorithms such as GPR, Bayesian (Belief) Networks, and Probabilistic Support Vector Machines that are widely used for constructing predictive models with uncertainty quantification capability. It is worth mentioning that PMML 4.2 [13] represents Naïve Bayes models that are considered as a special case of a two-level Bayesian Networks. Naïve Bayes are used as a classification model not as a regression model [19]. A major disadvantage of Naïve Bayes models is that it makes a strong assumption that the input variables are conditionally independent, given the output [19], and do not provide the uncertainty quantification capabilities that we mentioned previously. In this paper, we focus on GPR model and present the GPR PMML Schema. We also present a prototype to generate a GPR PMML representation using a real data set in the manufacturing domain.

The previous PMML 4.2 standard only supports parametric regression models, such as General regression, Neural Networks, Tree regression. The parametric regression models specify the type

of basis functions that will be used to approximate a target function and optimize the parameters that best fits the model to the data. As a result, such models' representability strongly depends on the selected basis function and provide only a point estimation for an unknown target input. As a nonparametric probabilistic regression, GPR, however, is a regression model based on nonparametric Bayesian statistics [20]. It predicts the target function value in the form of posterior distribution computed by combining a prior distribution on the target function and likelihood (noise) model. GPR uses a Gaussian Process (GP) as a prior to describe the distribution on the target function (i.e., the latent function values) and Gaussian likelihood to represent the noise distribution. When a GP prior is used with Gaussian likelihood, the posterior distribution on the unknown target function value can be analytically represented as a Gaussian distribution, which provides an estimated mean value and uncertainty in the predicted value. One special characteristic of a GPR model is that it does not use any predefined set of basis functions but uses training data points to represent the structure of the target function. Therefore, a GPR model can represent complex input and output relationships with a small number of hyper-parameters. Due to its ability to quantify uncertainty in the predictive model, GPR has received increased attention in the machine-learning community over the past decade. The GPR algorithm has been applied to many fields, including manufacturing, robotics, music rendering, and others [7, 8, 21, 22]. Note that GPR can also be used for classification by converting the repression output to class probability [23]. In this paper, we focus on regression application and its representation in PMML.

The purpose of this paper is to illustrate the use of the standard PMML representation of a GPR model for an advanced manufacturing application, in this case, for energy prediction on part machining. Furthermore, a MATLAB-based interpreter has been developed to import and export the GPR PMML models. The paper is organized as follows. Section 2 briefly introduces the basic concepts of GPR. In Section 3 the GPR PMML schema is introduced. An example using manufacturing data from a numerical control (NC) milling machine is demonstrated in Section 4. Finally, Section 5 concludes the paper with a brief summary and discussion for future work.

2. Theory of Gaussian Process Regression (GPR)

GPR approximates an unknown target function $y = f(\boldsymbol{x})$ in a probabilistic manner. Given a data set $\boldsymbol{D}^n = \{(\boldsymbol{x}^i, y^i) | i = 1, \dots, n\}$ with n samples, $\boldsymbol{x}^{1:n} = (\boldsymbol{x}^1, \dots, \boldsymbol{x}^n)$ and $\boldsymbol{y}^{1:n} = (y^1, \dots, y^n)$ denote the inputs and the corresponding (possibly noisy) output observations, respectively. The goal of GPR is to construct a posterior distribution $p(f^{new}|D^n)$ on the function value $f^{new} = f(\boldsymbol{x}^{new})$ corresponding to an unseen target input $\boldsymbol{x}^{new}$. The prediction is given as a probability distribution rather than a point estimate, quantifying uncertainty in the target value. A detailed description of GPR can be found in [20, 23]. The basic steps for training a GPR model and then performing prediction (or scoring) is summarized as illustrated in Figure 1, which also shows the organization of this section.

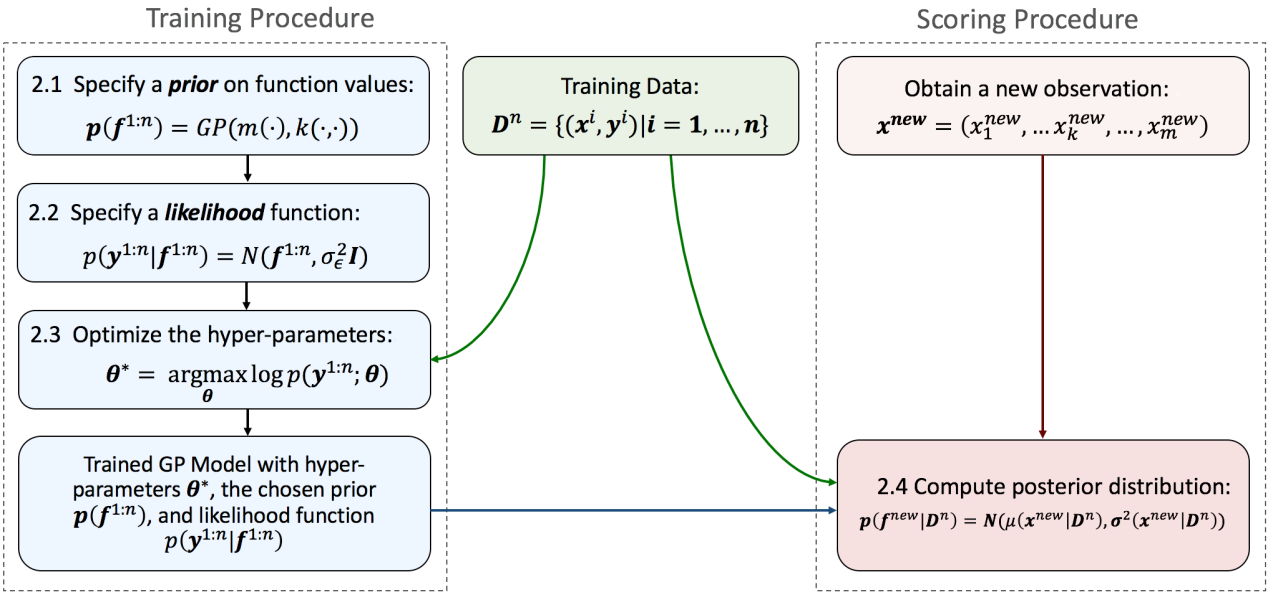

Figure 1. Flowchart showing GPR training and scoring procedure.

2.1 Training Procedure

GPR uses GP as a prior to describe the distribution on the target function $f(\boldsymbol{x})$. In GPR, the function values $\boldsymbol{f}^{1:n} = (f^1, \dots, f^n)$ corresponding to the input $\boldsymbol{x}^{1:n} = (\boldsymbol{x}^1, \dots, \boldsymbol{x}^n)$ are treated as random variables, where $f^i \coloneqq f(\boldsymbol{x}^i)$. GP is defined as a collection of random variables (stochastic process), any finite number of which is assumed to be jointly Gaussian distributed. GP can fully describe the distribution over an unknown function $f(\boldsymbol{x})$ by its mean function $m(\boldsymbol{x}) = \mathrm{E}[f(\boldsymbol{x})]$ and a kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ that approximates the covariance $\mathrm{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x}') - m(\boldsymbol{x}'))]$. The kernel (covariance) function represents a geometrical distance measure assuming that the more closely located inputs would be more correlated in terms of their function values. That is, the prior on the function values is represented as:

$$(\boldsymbol{f}^{1:n}) = GP\big(m(\cdot), k(\cdot,\cdot)\big) \tag{1}$$

where $m(\cdot)$ is a mean function capturing the overall trend in the target function value, and $k(\cdot,\cdot)$ is a kernel function used to approximate the covariance.

In GPR, the kernel (covariance) function describes the structure of the target function. Thus, the type of kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ used to build a GPR model and its parameters can strongly affect the overall representability of the GPR model and impact the accuracy of the prediction model. A wide variety of kernel functions can be used [23]; for example, the linear kernel, the polynomial kernel, the squared exponential kernel, which are included in the current PMML standard. As an

example, Eq. (2) shows the Automatic Relevance Determination (ARD) squared exponential covariance function [7, 8, 23], which is a widely used in many applications. However, the widely used Matern (covariance) kernel function is not included in the current PMML representation. The function requires the evaluation of gamma function to compute the covariance between two points [23]; it is expected to include the case in the near future.

$$k(\boldsymbol{x}^i, \boldsymbol{x}^j) = \gamma \exp\left(-\frac{1}{2}(\boldsymbol{x}^i - \boldsymbol{x}^j)^T \operatorname{diag}(\boldsymbol{\lambda})^{-2}(\boldsymbol{x}^i - \boldsymbol{x}^j)\right) \tag{2}$$

The ARD kernel function is described by the parameters, γ and $\boldsymbol{\lambda}$. The term γ is referred to as the signal variance that quantifies the overall magnitude of the covariance value. The parameter vector $\boldsymbol{\lambda} = (\lambda_1, \dots \lambda_i, \dots, \lambda_m)$ is referred to as the characteristic length scales that quantify the relevancy of the input features in $\boldsymbol{x^i} = (x_1^i, \dots x_k^i, \dots, x_m^i)$, where m defines the number of input variables, for predicting the response y. A large length scale λ_i indicates weak relevance for the corresponding input feature $\boldsymbol{x}^i$and vice versa.

2.2 Likelihood of Observations

The latent random variables (or a random vector) $\boldsymbol{f}^{1:n}$ needs to be inferred from the observation $\boldsymbol{y}^{1:n} = (y^1, \dots, y^n)$. Each observed value is assumed to contain some random noise ϵ^i, such that $y^i = f(\boldsymbol{x}^i) + \epsilon^i$. Different likelihood functions can be used to model the random noise term. It is common to assume that the noise term ϵ^i is independent and identically distributed Gaussian, in which case the likelihood function becomes:

$$p(\boldsymbol{y}^{1:n}|\boldsymbol{f}^{1:n}) = N(\boldsymbol{f}^{1:n}, \sigma_\epsilon^2 \mathbf{I}) \tag{3}$$

where the noise variance, σ_ϵ^2, quantifies the level of noise that exists in the response $y^i = f(\boldsymbol{x}^i) + \epsilon^i$. The Gaussian likelihood function is used because it guarantees that the posterior can also be expressed as a Gaussian distribution. Including the noise model, the covariance function is parameterized (i.e., defined) by the hyper-parameters jointly denoted by $\boldsymbol{\theta} = (\sigma_\epsilon, \gamma, \boldsymbol{\lambda})$ for the case of the squared exponential covariance function as shown in Eq. (2).

2.3 Hyper-parameter Optimization

The marginal distribution of the observations can be computed as:

$$p(\boldsymbol{y}^{1:n}; \boldsymbol{\theta}) = \int p(\boldsymbol{y}^{1:n}|\boldsymbol{f}^{1:n}; \boldsymbol{\theta})\, p(\boldsymbol{f}^{1:n}; \boldsymbol{\theta})\, d\boldsymbol{f}^{1:n} \tag{4}$$

When the GP prior and Gaussian likelihood function are used, the marginal distribution is also Gaussian. One attractive property of the GP model is that it provides an analytical closed form

expression for the marginal likelihood of the data (with the unknown latent function being "marginalized" out). The marginal log-likelihood for the training data set $\boldsymbol{D}^n = \{(\boldsymbol{x}^i, y^i)|i = 1, \dots, n\}$ can be expressed as [20, 23]:

$$\log p\,(\boldsymbol{y}^{1:n}; \boldsymbol{\theta}\,) = -\frac{1}{2}(\boldsymbol{y}^{1:n})^T(\mathbf{K} + \sigma_\epsilon^2\mathbf{I})^{-1}\boldsymbol{y}^{1:n} - \frac{1}{2}\log|\mathbf{K} + \sigma_\epsilon^2\mathbf{I}| - \frac{n}{2}\log 2\pi \tag{5}$$

where $\mathbf{K}$ is the covariance (kernel) matrix whose (i,j)th entry is $\mathbf{K}_{ij} = k(\boldsymbol{x}^i, \boldsymbol{x}^j)$. Then, the hyper-parameters $\boldsymbol{\theta} = (\sigma_\epsilon, \sigma_s, \boldsymbol{\lambda})$ for the noise model and the kernel function are determined as ones maximizing the marginal log-likelihood of the training data $\boldsymbol{D}^n = \{(\boldsymbol{x}^i, y^i)|i = 1, \dots, n\}$ as:

$$\begin{aligned} \boldsymbol{\theta}^* &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log p\,(\boldsymbol{y}^{1:n}; \boldsymbol{\theta}\,) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(-\frac{1}{2}(\boldsymbol{y}^{1:n})^T(\mathbf{K} + \sigma_\epsilon^2\mathbf{I})^{-1}\boldsymbol{y}^{1:n} - \frac{1}{2}\log|\mathbf{K} + \sigma_\epsilon^2\mathbf{I}| - \frac{n}{2}\log 2\pi\right) \end{aligned} \tag{6}$$

As long as the kernel function is differentiable with respect to its hyper-parameters $\boldsymbol{\theta}$, the marginal likelihood can be differentiated and optimized by various off-the-shelf mathematical programming tools.

2.4 Scoring Procedure – Computing the Posterior Distribution of the Target Function Value

After optimizing the model hyper-parameters, the GPR model is referred to as 'trained'. The model is fully characterized by the prior, the likelihood function, the hyper-parameters, and the training data set. Let's denote a newly observed input $\boldsymbol{x}^{new}$:

$$\boldsymbol{x}^{new} = (x_1^{new}, \dots x_k^{new}, \dots x_m^{new})$$

The (hidden) function value $f(\boldsymbol{x}^{new})$, denoted here as f^{new}, for the new input $\boldsymbol{x}^{new}$, and the observed outputs $\boldsymbol{y}^{1:n} = \{y^1, \dots, y^n\}$ follow a multivariate Gaussian distribution:

$$\begin{bmatrix} \boldsymbol{y}^{1:n} \\ f^{new} \end{bmatrix} \sim N\left(\mathbf{0}, \begin{bmatrix} (\mathbf{K} + \sigma_\epsilon^2\mathbf{I}) & \boldsymbol{k} \\ \boldsymbol{k}^T & k(\boldsymbol{x}^{new}, \boldsymbol{x}^{new}) \end{bmatrix}\right) \tag{7}$$

where $\boldsymbol{k}^T = \left(k(\boldsymbol{x}^1, \boldsymbol{x}^{new}), \dots, k(\boldsymbol{x}^n, \boldsymbol{x}^{new})\right)$. The posterior distribution on the response f^{new} for the newly observed (and previously unseen) input $\boldsymbol{x}^{new}$ given the historical data $\boldsymbol{D}^n = \{(\boldsymbol{x}^i, y^i)|i = 1, \dots, n\}$ can then be expressed as a 1-D Gaussian distribution $f^{new} \sim N\left(\mu(\boldsymbol{x}^{new}|\boldsymbol{D}^n), \sigma^2(\boldsymbol{x}^{new}|\boldsymbol{D}^n)\right)$ with the mean and variance functions expressed, respectively, as [20, 23]:

$$\mu(\boldsymbol{x}^{new}|\boldsymbol{D}^n) = \boldsymbol{k}^T(\mathbf{K} + \sigma_\epsilon^2\mathbf{I})^{-1}\boldsymbol{y}^{1:n} \tag{8}$$

$$\sigma^2(\boldsymbol{x}^{new}|\boldsymbol{D}^n) = k(\boldsymbol{x}^{new}, \boldsymbol{x}^{new}) - \boldsymbol{k}^T(\mathbf{K} + \sigma_\epsilon^2\mathbf{I})^{-1}\boldsymbol{k} \tag{9}$$

Here, $\mu(\boldsymbol{x}^{new}|\boldsymbol{D}^n)$ and $\sigma^2(\boldsymbol{x}^{new}|\boldsymbol{D}^n)$ can be used as the scoring functions for evaluating, respectively, the mean and variance of the hidden function output f^{new} corresponding to the input data $\boldsymbol{x}^{new}$.

As long as the kernel function is differentiable with respect to its hyper-parameters $\boldsymbol{\theta}$, the marginal likelihood can be differentiated and optimized by various off-the-shelf mathematical programming tools, such as GPML [24], PyGP [25], scikit-learn [26], and GPy [27].

3. GPR-PMML Schema

PMML provides an open standard for representing data mining and predictive models, thus the models can be transferred easily from one environment to another. Once a machine learning model has been trained in an environment like MATLAB, Python, or R, it can be saved as a PMML file. The PMML file can then be moved to a production environment such as an embedded system or a cloud server. The code in the production environment can import the PMML file, and use it to generate predictions for new unseen data points. It is important to note that PMML does not control the way that the model is trained, it is purely a standardized way to represent the trained model.

Figure 2 shows the general structure of a PMML document which includes four basic elements, namely header, data dictionary, data transformation, and the data mining or predictive model [13]. The *Header* element provides a general description of the PMML document, including name, version, timestamp, copyright, and other relevant information for the model development environment. The *DataDictionary* element contains the data fields and their types as well as the admissible values for the input data. The data transformation is performed via the *TransformationDictionary* or *LocalTransformations* element that describes the mapping of the data, if necessary, into a form usable by the mining or predictive model. PMML defines various data transformations such as normalization, discretization, etc. [13]. The model element contains the definition and description of the predictive model, which, for the GPR model, include the optimal hyper-parameters and the training data set. For detailed explanations of PMML structure and each element within the structure, we refer the reader to [13-16, 28].

Our discussion focuses on the description of the GPR model defined in PMML. As a non-parametric model, GPR requires a training data set in order to fully characterize the model [20, 23, 29]. The term non-parametric implies an infinite-dimensional parameter space that the

number and nature of the parameters are flexible and not fixed in advance [29, 30]. Therefore, all of the training data must be included in the PMML document along with the other model parameters. The list of information required to characterize the GPR model includes:

- The training inputs $\boldsymbol{x}^{1:n} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n)$ and the corresponding outputs $\boldsymbol{y}^{1:n} = (y^1, \ldots, y^n)$.
- The type of a kernel function $k(\cdot,\cdot)$ used to describe the underlying structure of the target function.
- The hyper-parameters for the specified kernel function and the noise variance representing the error magnitude in the output.

The PMML standard for the GPR model thus includes the description of the model, the kernel type and its hyper-parameters, and the training data set.

The current PMML only focuses on the case where the mean function in Eq. (1) is zero, i.e., $m(\boldsymbol{x}) = \boldsymbol{0}$, for two reasons. First, using zero mean function, $m(\boldsymbol{x}) = \boldsymbol{0}$, it simplifies the learning and prediction procedures without compromising its representability. Second, a mean function $m(\boldsymbol{x})$ can be represented by other parametric models, such as generalized linear regression, thus it can be expressed using the parametric models already included in the PMML standard. Integrating the regression models into current PMML GPR Schema would be desirable in the future.

3.1 *GaussianProcessModel* Element

A GPR model is represented by a *GaussianProcessModel* element defined in the XML schema, which contains all the necessary information to fully characterize the model. Figure 3 shows the attributes that can be added to the *GaussianProcessModel* element, and the elements which can be nested within the *GaussianProcessModel* element. Some of the attributes in the *GaussianProcessModel* are optional. The optional attributes are used to provide additional metadata about the model, such as the optimization algorithm used to optimize the hyper-parameters. The attributes defined are as follows:

- **modelName** specifies the name of the GPR model.
- **functionName** specifies whether the model type is ‘classification’ or ‘regression’.
- **optimizer** specifies the optimization algorithm used in training the Gaussian process model.
- **isScorable** specifies whether the model is complete, and can actually be used to generate predictions.

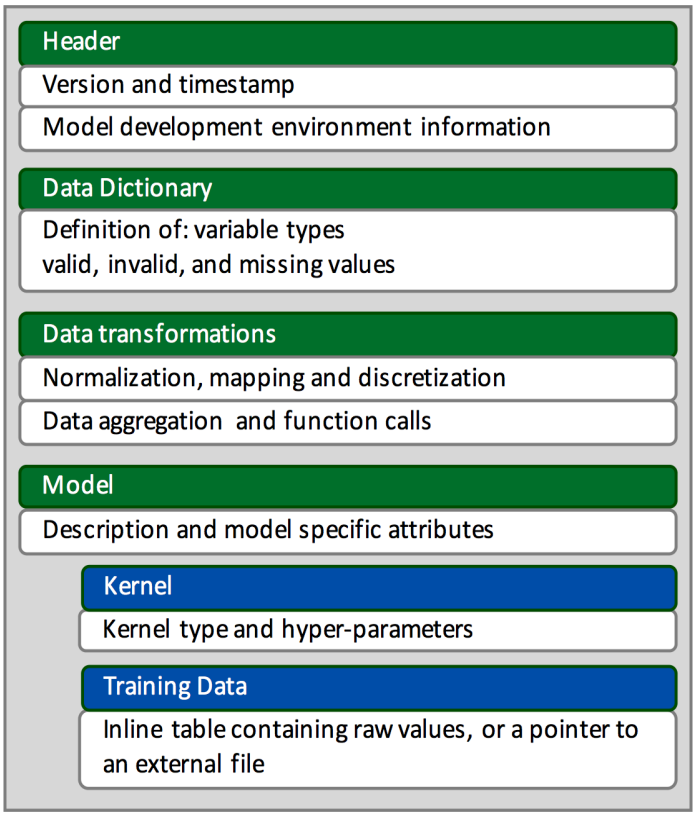

Figure 2. The structure and contents of a GPR PMML file adapted from [14].

The schema also defines the elements that can be nested within the *GaussianProcessModel element*. The *LocalTransformations* element is used to define any mathematical transformation applied to the input or output values of the model. The *GaussianProcessModel* element must contain one of the four possible kernel elements, *RadialBasisKernel*, *ARDSquaredExponentialKernel*, *AbsoluteExponentialKernel*, and *GeneralizedExponentialKernel*, as shown in the schema (see Figure 3). As an example, the *ARDSquaredExponentialKernel* element is described in Section 3.2. The *TrainingInstances* element contains the training data, and is described in Section 3.3.

3.2 *Kernel* Element

The *GaussianProcessModel* element must contain a single kernel element, which defines the type of kernel used in the model. For example, the ARD squared exponential kernel function defined in Eq. (2) is represented in the PMML schema as shown in Figure 4.

The ARD squared exponential kernel is characterized by the covariance magnitude, γ, the length-scale vector, $\boldsymbol{\lambda}$, and the noise variance, σ_ϵ^2. The covariance magnitude and the noise variance are stored as numerical attributes on the kernel element. The length-scale vector is represented as an

Array element, nested inside the kernel element. Figure 5 shows the schema for the nested length-scale element.

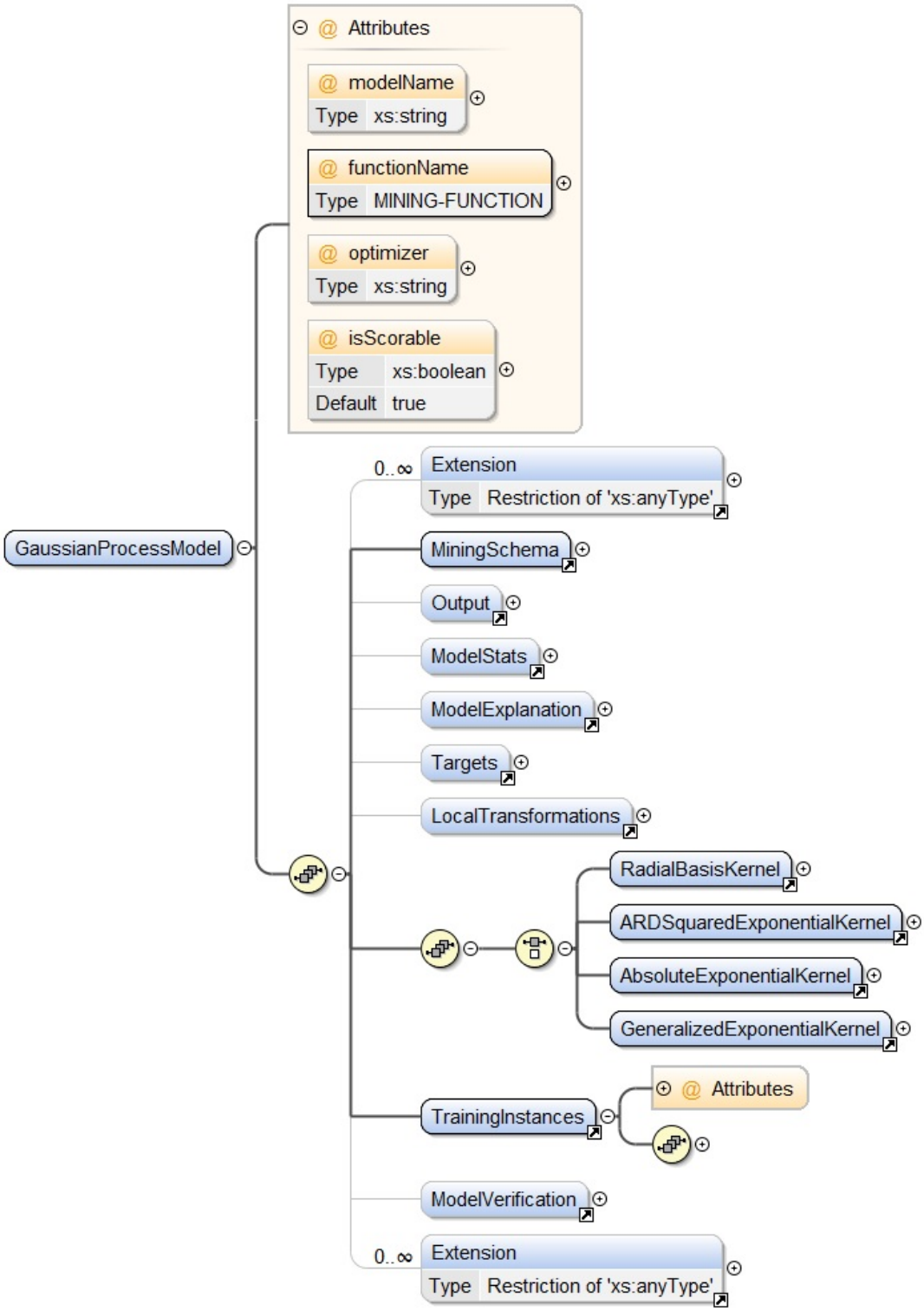

Figure 3. PMML schema of Gaussian process regression model.

3.3 *TrainingInstances* Element

For GPR, the training data is required to fully characterize the model. The training data is included in the *TrainingInstances* element. The schema for the *TrainingInstances* element is defined in the PMML 4.2 standard [13] and is repeated here for the sake of clarity as shown in Figure 6. The training data can be represented as either an *InlineTable* or *TableLocator* element**.** The *InlineTable* element allows training data to be included directly into the PMML document. The *InstanceFields* element is used to specify the fields that are included in the training data. Data contained in an external table can be referenced using a *TableLocator* element**.**

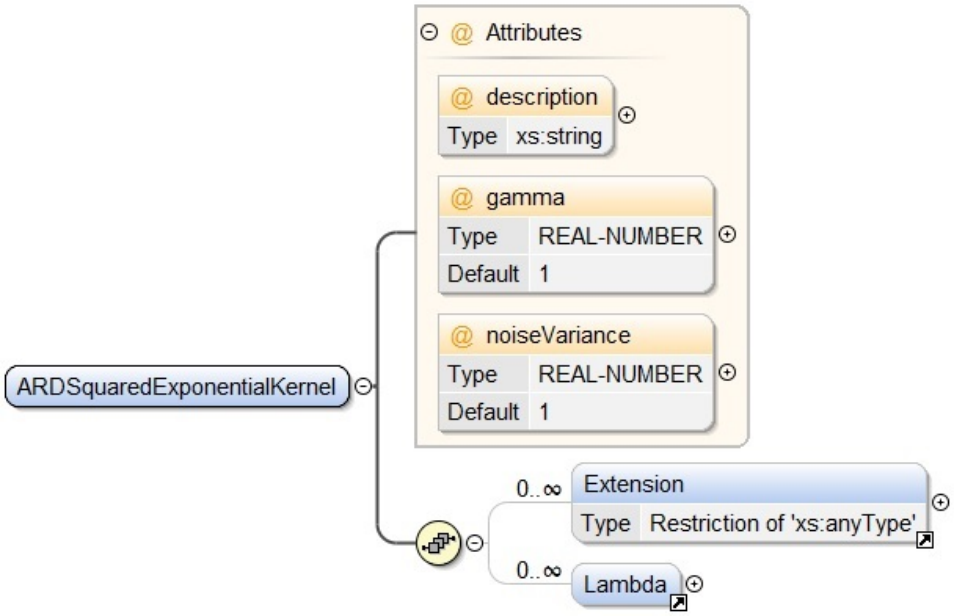

Figure 4. *ARDSquaredExponentialKernel* element schema.

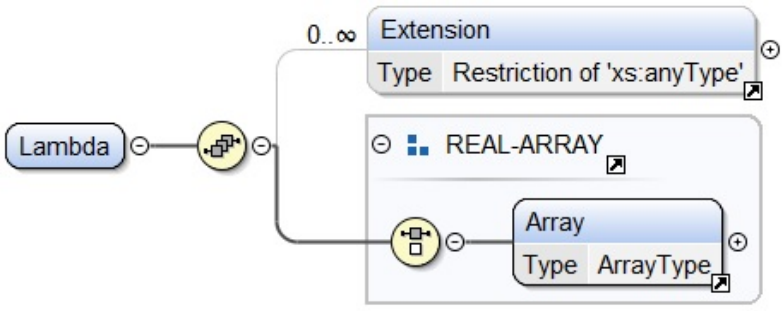

Figure 5. Length-scale (Lambda) element schema.

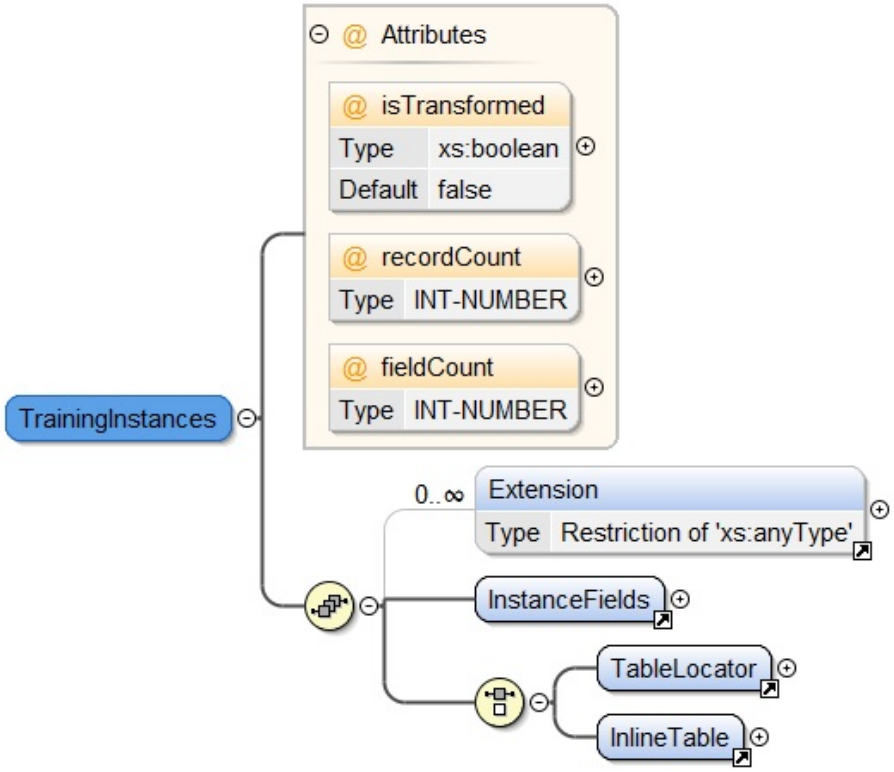

Figure 6. *TrainingInstances* element schema [13].

4. Application of GPR PMML to Predict Milling Machine Energy Consumption

This section discusses how an energy prediction model for a milling machine is stored according to the PMML standard for GPR, using a specifically developed MATLAB package [31]. A MATLAB package is developed to save generic GPR models to a file in the PMML format. Specifically, a GPR energy model is trained using real energy-consumption data obtained from a Mori Seiki NVD 1500DCG 3-axis milling machine at the UC Berkeley Mechanical Engineering Machine Shop [9], and the model is in the PMML format. The saved model can be used to predict energy consumption when machining a new part as well to determine the best tool path for machining a part with minimal energy [7, 8]. The scoring (or the prediction) procedure using the GPR PMML file is illustrated.

4.1 Energy Prediction Model

Machine learning techniques have been applied to various applications in manufacturing domain [2]; one of them is monitoring and optimizing the energy efficiency of the manufacturing processes, which has become a priority in the manufacturing industry. Advanced monitoring and operation strategies of machine tools will improve energy usage in manufacturing. The first step

towards developing such strategies is to understand the energy consumption pattern of machine tools and manufacturing operations. These strategies can enable the development of energy prediction models to determine how different operational strategies influence the energy consumption pattern of a machine tool and to derive an optimal strategy to select efficient operations for machining a part.

Herein, a GPR model is developed to predict the energy consumption of the milling machine as a function of the operating (machine) parameters. While this case study used face milling as a demonstrative example, the same technique has shown to be applicable to other milling operations [7, 8]. The experimental design, setup, and data processing techniques used for generating the data sets for this study have been previously described by Bhinge et al. [7]. A total of 196 face milling experiments for machining parts using the milling machine were conducted. The data sets were obtained using different operating parameters including feed rate, spindle speed, and depth of cut. Each operation within the data sets refers to a single NC block. The energy consumption, $E \in \mathbb{R}$, for each NC block was obtained by numerically integrating the power time series recorded over the duration of the block.

In this study, the output parameter, y, is defined as the energy consumption per unit length. Furthermore, the input features employed are defined as follows:

- $x_1 \in \mathbb{R}$ Feed rate: the velocity at which the tool is fed
- $x_2 \in \mathbb{R}$ Spindle speed: rotational speed of the tool
- $x_3 \in \mathbb{R}$ Depth of cut: depth of material that the tool is removing
- $x_4 \in \{1, 2, 3, 4\}$ Active cutting direction: (1 is for x-axis, 2 for y-axis, 3 for z-axis, and 4 for x-y axes)
- $x_5 \in \{1, 2, 3\}$ Cutting strategy: the method for removing material (1 is for conventional, 2 for climbing, and 3 for both)

For this example, we choose to use all of the measured input features, and subsequently define the input feature vector, $\boldsymbol{x} = \{x_1, \dots, x_5\}$. We assume that the output $y = f(\boldsymbol{x}) + \epsilon$ is measured with noise $\epsilon \sim N(0, \sigma_\epsilon^2)$. The ARD squared exponential kernel is used as the covariance function. Furthermore, we assume the mean function from the prior (see Eq. (1)) to be a zero function.

The GPR model is used to generate energy density predictions $\hat{y}^i$ for a new set of operating conditions $\boldsymbol{x}^i$. Let $\boldsymbol{D^{train}}$ denote the training data set containing the input feature data and the output parameter data. Each new prediction $\hat{y}^i$ is represented by a mean energy density function $\mu(\boldsymbol{x}^i|\boldsymbol{D^{train}})$ and associated standard deviation function $\sigma(\boldsymbol{x}^i|\boldsymbol{D^{train}})$. We can then estimate the energy consumption $\hat{E}^i$ and the corresponding standard deviation S^i on the estimated energy consumption value as:

$$\hat{E}^i = \mu\left(\boldsymbol{x}^i \middle| \boldsymbol{D}^{\boldsymbol{train}}\right) \times l_i \tag{10}$$

$$S^i = \sigma\left(\boldsymbol{x}^i \middle| \boldsymbol{D}^{\boldsymbol{train}}\right) \times l_i \tag{11}$$

Details on developing the GPR model have been reported in [7, 8, 20, 23].

4.2 Representing the Model using PMML

In a real application, it is likely that the same GPR model will be used to generate predictions in multiple different computing environments. Using the PMML schema described in this work, it is possible to communicate the GPR model in a standard way to different PMML-compliant scoring engines. Thus, one can train the model in one computing environment, and then communicate the model to multiple other environments using the GPR PMML format.

A MATLAB package [31] is developed to consume and produce GPR PMML documents. The package is made up of four main modules: a PMML parser, PMML generator, GPR core, and GPR interface. The PMML parser is designed to extract important information from a PMML file, such as the kernel type and hyper-parameters. The PMML generator provides a way to generate a PMML representation of a GPR model. The GPR core provides the mathematical logic for training GPR models and scoring new observations, relying on the scoring and optimization algorithms provided by the Gaussian Process for Machine Learning (GPML) toolbox [24]. The GPR interface provides a clean object-orientated interface to the PMML generator, PMML parser and GPR core.

The package provides a simple way to save trained GPR models to PMML. Given the kernel type, likelihood function, optimized hyper-parameters, and training data, the package generates a valid PMML representation of the GPR model. Internally, the package uses the JAXP extensible markup language parser [32] to generate the PMML file, according to the schema described in this document. The package also provides a way to load a GPR model from a PMML file and then generate predictions for a new data point, $\boldsymbol{x}^{new}$. The JAXP parser is used to extract the kernel type, likelihood function, hyper-parameters, and training data from the PMML file. This information is then used to calculate $\mu(\boldsymbol{x}^{new}|\boldsymbol{D}^n)$ and $\sigma(\boldsymbol{x}^{new}|\boldsymbol{D}^n)$ as described in Section 2.4.

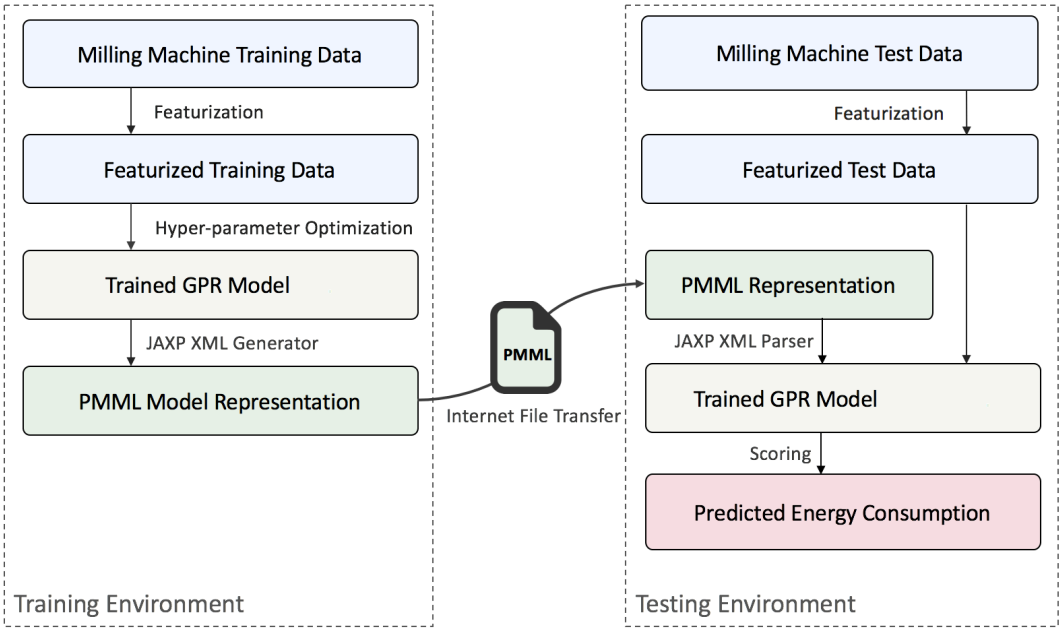

Figure 7. Example of training and testing flow using PMML to persist GPR Model.

For our study, the MATLAB package [31] is installed on two separate computers, both of which are connected to the Internet. Hereafter, we will refer to the first computer as the training environment and the second computer as the testing environment. The milling machine data set is randomly divided into a training set $\boldsymbol{D^{train}}$ and testing set $\boldsymbol{D^{test}}$ with the ratio of training to testing points set to 7:3. A GPR model is trained in the training environment using $\boldsymbol{D^{train}}$, and saved in the PMML format using the previously described MATLAB package. As illustrated in Figure 7, the PMML file is transferred to the testing environment via an internet connection. The trained GPR model parameters are loaded from the PMML file using the MATLAB package. The PMML file contains the optimized hyper-parameters, kernel function type, training data, variable names, and PMML version number. The optional copyright attribute is included in the PMML header element. A new set of energy consumption predictions, $\boldsymbol{\widehat{E}^{test}}$, is generated from the PMML file in the testing environment. An abbreviated version of the PMML file corresponding to the case study, i.e., energy prediction model, is shown for reference in Figure 8. Note that most of the training data has been omitted for brevity.

```
<?xml version="1.0" encoding="utf-8"?>
<PMML xmlns="http://www.dmg.org/PMML-4_3" version="4.3">
   <Header copyright="Copyright (c) Stanford University"/>
   <DataDictionary numberOfFields="6">
      <DataField dataType="double" name="x1" optype="continuous"/>
      <DataField dataType="double" name="x2" optype="continuous"/>
      <DataField dataType="double" name="x3" optype="continuous"/>
      <DataField dataType="double" name="x4" optype="continuous"/>
      <DataField dataType="double" name="x5" optype="continuous"/>
      <DataField dataType="double" name="y1" optype="continuous"/>
   </DataDictionary>
   <GaussianProcessModel functionName="regression" modelName="Gaussian Process Model">
      <MiningSchema>
         <MiningField name="x1" usageType="active"/>...
         <MiningField name="y1" usageType="predicted"/>
      </MiningSchema>
      <Output>
         <OutputField dataType="double" feature="predictedValue"
            name="MeanValue" optype="continuous"/>
         <OutputField dataType="double" feature="predictedValue"
            name="StandardDeviation" optype="continuous"/>
      </Output>
      <ARDSquaredExponentialKernel gamma="20.6643" noiseVariance="0.48149">
         <Lambda>
            <array n="5" type="real">
               190.602107 1605.475542 2.968979 0.959402 1.017964
            </array>
         </Lambda>
      </ARDSquaredExponentialKernel>
      <TrainingInstances fieldCount="6" isTransformed="false" recordCount="1466">
         <InstanceFields>
            <InstanceField column="x1" field="x1"/>
            <InstanceField column="x2" field="x2"/>
            <InstanceField column="x3" field="x3"/>
            <InstanceField column="x4" field="x4"/>
            <InstanceField column="x5" field="x5"/>
            <InstanceField column="y1" field="y1"/>
         </InstanceFields>
         <InlineTable>
            <row>
               <x1>75.9855</x1>
               <x2>1502.1048</x2>
               <x3>1</x3>
               <x4>2</x4>
               <x5>2</x5>
               <y1>7.2812</y1>
            </row>
            ...
         </InlineTable>
      </TrainingInstances>
   </GaussianProcessModel>
</PMML>
```

Figure 8. PMML file representing the energy consumption model.

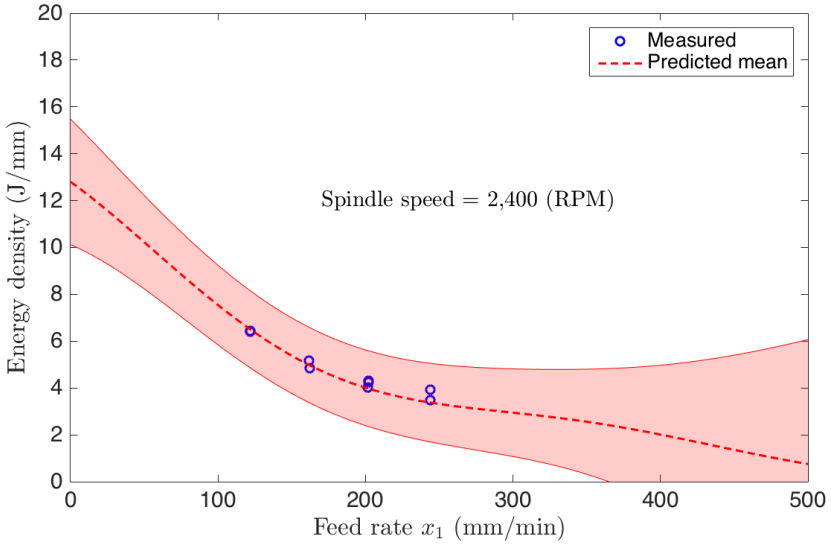

Figure 9. Predicted energy consumption density for generic test parts machined using face milling, y-direction cut, 2,400 RPM spindle speed, conventional cutting strategy, and 1mm cut depth. The shaded area shows one-standard deviation bounds for the prediction.

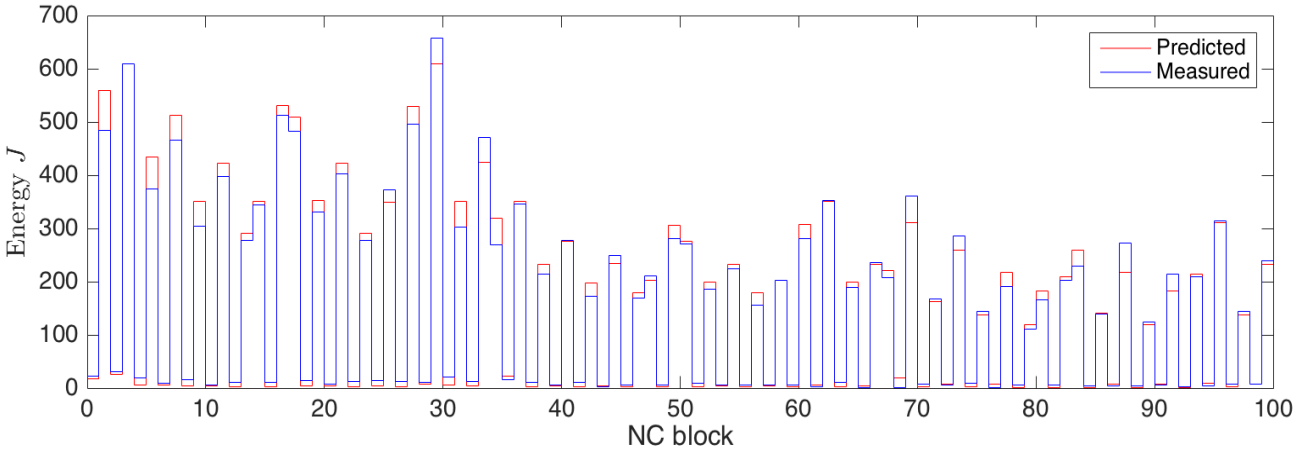

Figure 10. Predicted mean energy consumption for face milling operations using the GPR, compared to the actual energy consumption of the machine.

Figure 9 shows the energy consumption predicted by the model stored in the PMML file. As shown in Figure 10 the GPR model provides a good estimation of the energy consumption of the milling machine on the test data set. Due to the standardized nature of PMML, the predicted energy consumption is independent from the computing platform/tool.

5. Discussion and Conclusion

This study presents PMML representation of Gaussian Process Regression (GPR) model and a prototype to generate GPR PMML representation using a real data set. GP is a stochastic process that is capable of modeling the complex relationship between the input and output variables, and constructing a prediction function with confidence bounds. Estimation of confidence bounds for future values provides more information about the predictions and thus, plays an important role in risk-informed decision making. The newly released PMML version 4.3, which includes the Gaussian Process model, provides this capability.

In the PMML v4.3, the *GaussianProcessRegression* element is defined to represent a trained GPR Model. An XML schema of the GPR model is provided to assist in parsing and validating of GPR models. We have demonstrated that a GPR model for the energy consumption of a milling machine can be structured in the PMML file format using a purposely-built MATLAB software package [31]. The development of an abstraction layer on the GPR PMML file format makes it trivial to convert existing GPR models to valid PMML. The standardized PMML file format ensures that transferring the model from one environment to another is reliable and straightforward.

For the case of GP regression, a prediction on a target input is represented by a posterior distribution that can be expressed in a closed form using the parameters optimized and the training data. For classification, however, the prediction cannot be represented using analytical expression, because the output of the regression needs to be converted to a probability using a squashing function, i.e., sigmoid function. To predict the probability of output class, it is required to approximate the inference procedure and this complicates the scoring procedure of classification tasks. In addition, the current PMML GPR model can be extended so that it can represent multi-output GP regression models, which is similar to representing a parametric regression model for a vector output. To represent multi-output regression model, the PMML will include a more generalized kernel function that can represent the covariance in the same type of output as well as the covariance in the different output values.

The current PMML GPR standard assumes that the size of training data is moderate. One of the drawbacks of the GP is that it is not sparse and uses the whole set of samples or features information to perform the prediction. Thus, the GPR PMML standard requires that all of the training data points are stored in the PMML file along with the model parameters. This can cause excessively large PMML file size and slow load time. When the size of training data is large, approximated methods for training and prediction are required [33]. There have been recent rapid developments in efficient approximation techniques for Gaussian processes. To reduce the computational and storage requirements, approximated GP regression approaches, such as stochastic variational inference for Gaussian process models [33], Gaussian process mixture model

[34], local GP [19, 35], and sparse GP [35-39] have been proposed. In most cases, these methods reduce the computational demand when training and scoring from GPR models, making them very favorable. Future work could investigate the representation of local or sparse GPR models in the PMML format.

One promising application of the PMML format is its ability to train models in a powerful computing environment like the Google Compute Engine [29], and then transfer those models to a personal computer or embedded device for scoring. The development of a GPR PMML scoring machine in a low-level language such as C would allow GPR PMML models to be evaluated on embedded devices. This would provide a simple standardized framework to develop smart sensors and devices.

ACKNOWLEDGMENTS AND DISCLAIMER

The work described in this paper was funded in part by the National Institute of Standards and Technology (NIST) cooperative agreement with Stanford University, Grant No. 70NANB12H273, and was supported by National Institute of Standards and Technology's Foreign Guest Researcher Program. The authors would like to acknowledge the support of the late Prof. David Dornfeld and Mr. Raunak Bhinge of the Laboratory for Manufacturing and Sustainability at UC Berkeley, USA in collecting, preparing, and providing machine operation data. 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.

REFERENCES

1. Jordan, M. I. and Mitchell, T. M. "Machine Learning: Trends, perspectives and prospects," *Science*, Vol. 349, 2015, pp. 255-260.
2. Wuest, T., Weimer, D., Irgens, C. and Thoben, K-D., "Machine Learning in Manufacturing: Advantages, Challenges, and Applications," *Production & Manufacturing Research: An Open Access Journal*, Vol. 4, 2016, pp. 23-45.
3. Suresh, P. V. S., Venkateswara Rao, P. and Deshmukh, S. G., "A Genetic Algorithmic Approach for Optimization of Surface Roughness Prediction Model," *International Journal of Machine Tools and Manufacture*, Vol. 42, 2002, pp. 675-680.
4. Ghosh, N., Ravi, Y. B., Patra, A., Mukhopadhyay, S., Paul, S., Mohanty, A. R. and Chattopadhyay, A. B., "Estimation of Tool Wear During CNC Milling Using Neural Network-Based Sensor Fusion," *Mechanical Systems and Signal Processing*, Vol. 21, 2007, pp. 466-479.
5. Jihong, Y. and Lee J., "Degradation Assessment and Fault Modes Classification Using Logistic Regression", *Journal of Manufacturing Science and Engineering*, Vol. 127, 2005, pp. 912-914.
6. Carbonneau, R., Laframboise, K. and Vahidov, R., "Application of Machine Learning Techniques for Supply Chain Demand Forecasting," *European Journal of Operational Research*, Vol. 184, 2008, pp. 1140-1154.

7. Bhinge, R., Biswas, N., Dornfeld, D., Park, J., Law, K. H., Helu, M. and Rachuri, S., "An Intelligent Machine Monitoring System for Energy Prediction Using a Gaussian Process Regression," presented at the *IEEE International Conference on Big Data,* Bethesda, MD, Oct. 27-30, 2014, IEEE, pp. 978–986.
8. Park, J., Law, K. H., Bhinge, R., Biswas, N., Srinivasan, A., Dornfeld, D., Helu, M. and Rachuri, S., "A Generalized Data-Driven Energy Prediction Model with Uncertainty for a Milling Machine Tool Using Gaussian Process," presented at the *ASME 2015 International Manufacturing Science and Engineering Conference (MSEC2015),* July 8-12, 2015, Charlotte, NC, ASME, pp. V002T05A010.
9. LMAS Database, By Raunak Bhinge, http://lmas.berkeley.edu/raunak.html, viewed 1/6/2016.
10. Ak, R., Helu, M. and Rachuri, S., "Ensemble Neural Network Model for Predicting the Energy Consumption of a Milling Machine" presented at *the ASME 2015 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2015)*, Aug. 2-5, 2015, Boston, MA, ASME, pp. V004T05A056.
11. Nannapaneni, S., Sankaran M. and Rachuri, S., "Performance Evaluation of a Manufacturing Process Under Uncertainty Using Bayesian Networks," *Journal of Cleaner Production*, Vol. 113, 2016, pp. 947-959.
12. Lyons, K., Bock, C., Shao G. and Ak, R., "Standards Supporting Simulations of Smart Manufacturing Systems", presented at the *2016 Winter Simulation Conference*, Arlington, VA, Dec 11-14, 2016.
13. The predictive model markup language (PMML) 4.2, By The Data Mining Group (DMG), http://dmg.org/, viewed 01/06/2016.
14. Guazzelli, A., Zeller, M., Lin, W.-C. and Williams, G., "PMML: An open standard for sharing models," *The R Journal*, Vol. 1, 2009, pp. 60–65.
15. Guazzelli, A., "What is PMML? Explore the Power of Predictive Analytics and Open Standards*," IBM developerWorks*, 2010, pp. 1-10.
16. Guazzelli A., Stathatos K. and Zeller, M., "Efficient Deployment of Predictive Analytics Through Open Standards and Cloud Computing," *ACM SIGKDD Explorations Newsletter*, Vol. 11, 2009, pp. 32-38.
17. Getting Started with SAS® Enterprise Miner™ 7.1, By SAS, December 2011, https://support.sas.com/documentation/cdl/en/emgsj/64144/PDF/default/emgsj.pdf., viewed 1/6/2016.
18. Python, By Python Software Foundation, https://www.python.org/, viewed 12/22/2016.
19. Mitchell, T. M., 1997, "Bayesian Learning," Machine Learning, McGraw-Hill, New York, 1997, pp. 154–200.
20. Rasmussen, C. E., 2004, "Gaussian Processes in Machine Learning," Advanced Lectures on Machine Learning, Springer Berlin Heidelberg, New York, 2004, pp. 63–71.
21. Nguyen-Tuong, D., Peters, J. R. and Seeger, M., "Local Gaussian Process Regression for Real Time Online Model Learning," presented at the *22nd Annual Conference on Neural Information Processing Systems*, Dec 8-10, 2008, Vancouver, British Columbia, Advances in Neural Information Processing Systems 21, NIPS Foundation, Inc., pp. 1193–1200.
22. K. Teramura, O. Hideharu, T. Yuusaku, M. Shimpei, and M. Shin-ichi, "Gaussian Process Regression for Rendering Music Performance," in *International Conference on Music Perception and Cognition (ICMPC 10)*, Sapporo, Japan, 2008, pp. 167–172.
23. C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning,* the MIT Press, 2006.

24. C. E. Rasmussen and H. Nickisch, "Gaussian Processes for Machine Learning (GPML) Toolbox," *J. Mach. Learn. Res.*, vol. 11, no. Nov, pp. 3011–3015, 2010.
25. Scikit-learn, Machine Learning in Python, By Mathieu Blondel et al., http://scikit-learn.org/stable/, viewed 12/22/2016.
26. Python package for Gaussian process regression in python, By Oliver Stegle, Max Zwiessele, Nicolo Fusi, https://pypi.python.org/pypi/pygp, viewed 12/22/2016.
27. GPy, By Sheffield Machine Learning Group, https://sheffieldml.github.io/GPy/, viewed 12/22/2016.
28. A. Guazzelli, W.-C. Lin, and T. Jena, *PMML in Action: Unleashing the Power of Open Standards for Data Mining and Predictive Analytics*, 2nd ed. Paramount, CA: CreateSpace, 2012.
29. Orbanz, P. and Teh, Y. W., 2011, "Bayesian nonparametric models," Encyclopedia of Machine Learning, Springer, USA, 2011, pp. 81-89.
30. Murphy, K. P., *Machine Learning: A Probabilistic Perspective*, The MIT Press, Cambridge, MA, 2012, p. 16.
31. Gaussian Process Regression PMML Package for MATLAB, By Max Ferguson and Jinkyoo Park, https://github.com/maxkferg/matlab-pmml, viewed 06/30/2016.
32. Suttor, J., Walsh, N. and Kawaguchi, K., *JSR 206 Java API for XML Processing (JAXP) 1.3*, Technical Report, Sun Microsystems, 2004, pp. 1-252.
33. Hensman, James, Nicolo Fusi, and Neil D. Lawrence, Gaussian processes for big data, in Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence (UAI2013), 26 Sep 2013.
34. Shi, J. Q., Murray-Smith, R. and Titterington, D. M., "Hierarchical Gaussian Process Mixtures for Regression," *Statistics and Computing*, Vol. 15, 2005, pp. 31–41.
35. Snelson, E. and Zoubin, G., "Local and Global Sparse Gaussian Process Approximation," presented at the *11th International Conference on Artificial Intelligence and Statistics*, San Juan, Puerto Rico, March 21-24, 2007, AISTATS, vol. 11, pp. 524-531.
36. Snelson, E. and Ghahramani, Z., "Sparse Gaussian Processes using Pseudo-inputs," presented at the *Neural Information Processing Systems 2005 Conference*, Vancouver, British Columbia, Dec 5-8, 2005, Neural Information Processing Systems (NIPS), 2005, pp. 1257–1264.
37. Quiñonero-Candela, J. and Rasmussen C. E., "A Unifying View of Sparse Approximate Gaussian Process Regression," *Journal of Mach. Learning Research*, Vol. 6, 2005, pp. 1939–1959.
38. Titsias, M. K., "Variational Learning of Inducing Variables in Sparse Gaussian Processes," presented at the *12th International Conference on Artificial Intelligence and Statistics*, 2009, Vol. 12, pp. 567–574.
39. A. Ranganathan, M. H. Yang, and J. Ho, "Online Sparse Gaussian Process Regression and Its Applications," IEEE Trans. on Image Processing, vol. 20, no. 2, Feb. 2011, pp. 391–404.
40. M. Cohen, K. Hurley, P. Newson, Google Compute Engine, O'Reilly Media, 2015.
41. Ferguson, M., Law, K., Bhinge, R., Lee, Y-S. T. and Park, J., "Evaluation of a PMML-Based GPR Scoring Engine on a Cloud Platform and Microcomputer Board for Smart Manufacturing", presented at the *IEEE International Conference on Big Data*, Dec 5-8, 2016, Washington D.C., pp. 1-10.