Engineering Solid Mechanics 11 (2023) 175-190

Contents lists available at GrowingScience

Engineering Solid Mechanics

homepage: www.GrowingScience.com/esm

Comparison of different supervised machine learning algorithms for bead geometry prediction in GMAW process

Teerapun Saeheaw[a*]

[a]*Department of Teacher Training in Mechanical Engineering, King Mongkut's University of Technology North Bangkok, Bangkok 10800, Thailand*

ARTICLE INFO

Article history:
Received 22 September 2022
Accepted 23 December 2022
Available online
23 December 2022

Keywords:
GMAW Process
Bead Geometry Prediction
Machine Learning
Regression Learner App
Fine Tree
Artificial Neural Network

ABSTRACT

Gas Metal Arc Welding (GMAW) is an extensively implemented arc welding process through the control of input process parameters and the metal from the filler wire. Despite its popular use in various industries, the complex interrelationship between the actual bead and the varying welding parameters makes it challenging to predict appropriate bead geometries via mathematical modeling in a continually changing welding process. In this study, the Regression Learner App was used to compare the performance of supervised Machine Learning (ML) predictive models comprising the Linear Regression (LR), Regression Tree (RT), Support Vector Machine (SVM), Ensembles of Tree (ET), Gaussian Process Regression (GPR), and Artificial Neural Network (ANN) using GMAW dataset. The dataset was scaled and normalized at a range of -1 to +1 to facilitate the visualization of the variation effect. The wire feed speed, voltage, weld velocity, unmelted wire length, and melted wire volume were considered as the input parameters to predict the bead geometry. In addition, the five-fold cross-validation was employed to avoid overfitting and poor generalization. Finally, statistical indicators, namely the Coefficient of Determination (R^2), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE), were performed on all developed models to evaluate their performance. Thus, the fine tree and ANN models achieved the highest prediction accuracies of 88–91%, signifying their potential use in future research. In short, the present study demonstrated the performance of various supervised ML algorithms for bead geometry prediction, which would assist the selection of appropriately supervised ML algorithms in future studies.

1. Introduction

Gas Metal Arc Welding (GMAW) is implemented in various industrial processes, particularly for ferrous and non-ferrous materials, given its exceptional welding quality, high efficiency, straightforward automation in manufacturing, and low-cost production for the development of arc welding. Proper control of certain input process parameters and the metal from the filler wire in the GMAW process would ensure the metal from the electrode passes through the arc and is deposited in a uniform layer on the base metal called bead (Tham et al., 2012). Such principles can be applied by essentially controlling the power (voltage and current), where the voltage is governed by the arc length, which correspondingly relies on the diameter of the electrode. Moreover, the voltage and current are set according to the welding position and the size of the workpiece (Kamble & Rao, 2013).

In view of this, the control of the bead geometry through several process factors, such as the welding voltage, welding current, welding velocity, wire feed speed, and Contact Tip to Work Distance (CTWD), is crucial in welding processes since they influence the mechanical properties of the weldment. Despite the growing significance of bead geometry prediction to minimize the duration and material in unnecessary trials across numerous industrial applications, the complex interrelationship

* Corresponding author.
E-mail addresses: teerapun.s@fte.kmutnb.ac.th (T. Saeheaw)

ISSN 2291-8752 (Online) - ISSN 2291-8744 (Print)

doi: 10.5267/j.esm.2022.12.003

between the actual bead and different welding parameters makes it complicated to determine the most suitable mathematical model for the bead geometry prediction in a constantly changing welding process (Wu et al., 2017). Thus, the application of supervised Machine Learning (ML) algorithms has been investigated to address this problem.

Generally, ML is a subtheme of computer science that explores a limited set of input data to construct valuable algorithms capable of learning from and establishing data-driven predictions. Notably, supervised learning refers to the ML task of inferring a function from a set of labeled training data with a given target response (Hastie et al., 2009). Following the 'No Free Lunch' theorem (Wolpert and Macready, 1997) interpretation, there is no universal ML method that concisely provides the best results in terms of predictions based on all sets of potential data (Wolpert, 1996). In other words, it is not possible to select the proper algorithm in advance. Hence, the most appropriate method to determine the performance analysis of bead geometry prediction is by evaluating different variants of supervised ML algorithms.

Several conventional computational models have been developed to predict bead geometry, which include the Submerged Arc Welding (SAW) process modeling via Linear Regression (LR) (Yang et al., 1993) and the Metal Inert Gas (MIG) process modeling via the second-order regression analysis (Fauzi et al., 2018). However, a recent study (Dong et al., 2016) highlighted that the key drawback of these models was their poor precision due to the difficulty to establish a reliable formula that includes all the processing complexities. Interestingly, the advanced research of ML models in recent years has prompted the development of Artificial Neural Network (ANN) that exploits a vast amount of data to learn the input-output correlation and offers a better capability to predict non-linear processes. The prediction of weld features using ANN has been reported by several researchers. For instance, Dutta and Pratihar (2007) recorded superior bead geometry prediction using the ANN model obtained through the Tungsten Inert Gas (TIG) welding process compared to the prediction based on LR approaches. A similar investigation agreed that the ANN model outperformed the regression model (Tafarroj and Kolahan, 2018).

In addition to ANN, other ML approaches, such as Support Vector Machines (SVMs), have been evaluated in past studies to predict the weld properties. Previously, Dong et al. (2017) reported the effective use of an SVM to predict the backside weld bead shape for online weld quality control. In another study, Liang et al. (2019) presented an enhanced accuracy of the bead penetration prediction from the weld pool surface using the Support Vector Regression (SVR) compared to the ANN model. Based on the above-mentioned reports, ML models are effective measures to predict bead geometry in any welding process.

While the applicability and effectiveness of ML algorithms on predictive bead geometry modeling expands progressively, there is yet any research that attempts to thoroughly compare the performance of different ML algorithms for optimum bead geometry prediction. Therefore, this study was aimed to compare the performance of various ML models comprising Linear Regression (LR), Regression Tree (RT), Support Vector Machine (SVM), Ensembles of Trees (ET), Gaussian Process Regression (GPR), and Artificial Neural Network (ANN) to accurately predict the bead geometry based on varying parameters, including wire feed speed, voltage, weld velocity, unmelted wire length, and melted wire volume using the Regression Learning App in GMAW process. The performance indicators, namely Coefficient of Determination (R^2), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE), were used to validate the accuracy of each proposed model. The results of this study would facilitate a greater understanding of the bead geometry prediction using ML algorithms and establish precise research goals accordingly.

2. Materials and Methods

2.1 Experimental Details

This study employed the bead-on-plate approach to deposit weld metal on the surface of a 300 mm × 40 mm × 6.35 mm base metal, which is composed of a flat carbon steel sheet piece with the addition of a filler wire ER770S with a diameter of 1.2 mm. All materials and preparation were based on the AWS A5.18 specification for carbon steel electrodes and rods for the GMAW standard, as stipulated by the American Welding Society. A TIME 5000 Digital power source was used to supply the welding power, while a state machine implemented in the control interface was used to determine the operational parameters and measure the arc variables (welding voltage and welding current) and the welding power source status. Meanwhile, a flat welding table was developed in the Automation and Control Group laboratory of the University of Brasilia as a platform to place and secure the piece to be welded and allow the specimen to move linearly in one direction. The welding table was equipped with a stepper motor control circuit with control signals to modify the direction, speed, and travel time of the starting and end points of the weld beads. For each experiment, the working angle and CTWD were fixed at zero degrees (0°) and 15 cm, respectively, while the speed of the wire feed was set directly proportional to the welding current (Chandrasekaran et al., 2019). The shielding gas was composed of 96% Ar and 4% CO_2. The developed bead geometry was then revealed by polishing and etching the welded samples with 2.5% Nital solution. Subsequently, macrographic evaluation was employed in the same longitudinal direction as the torch movement to measure the developed bead geometry.

2.2 Dataset Description

A data-driven model was built in this study using a GMAW process dataset from reference (Martinez et al., 2021) that contains 2,170 records with eight attributes from the experimental results and was 10 times higher than the quantity of the input variables (Hastie et al., 2009). As shown in Table 1, the statistical features of the gathered datasets were divided into five input parameters to construct different prediction models, which include the direct input parameters (wire feed speed (Ws), voltage (V), and weld velocity (Wv)) and indirect input parameters (unmelted wire length (Ul) and droplet or melted wire volume (Mv)). These parameters were modified during the process and influenced three output parameters of the bead geometry (bead width (Width), penetration depth (Depth), and reinforcement height (Height)). The column Mean represents the average values, while the columns Min and Max refer to the minimum and maximum values of each variable. The final column Std presents the standard deviation over the requested variable. The respective inputs and outputs were used to develop the predictive bead geometry model.

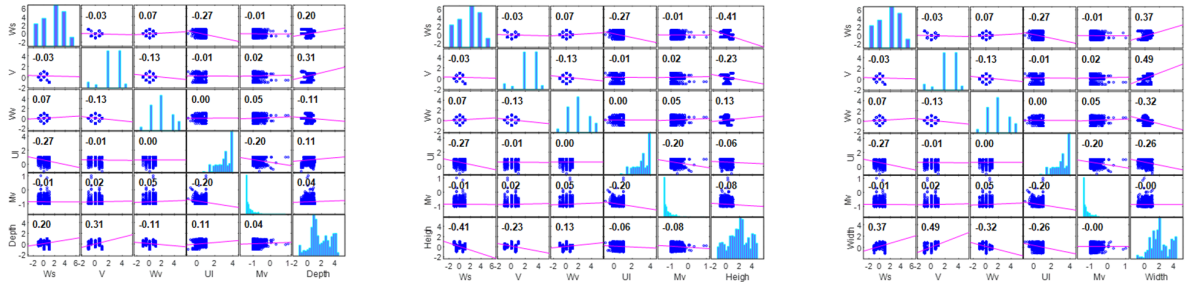
Table 1. Statistical features of the gathered datasets

	Parameter	Notation	Mean	Min	Max	Std
Input	Wire feed speed (m/min)	Ws	6.38	4.80	8.20	1.06
	Voltage (volt)	V	26.09	17.00	32.00	3.63
	Welding velocity (mm/s)	Wv	9.99	6.20	13.40	1.90
	Unmelted wire length (mm)	Ul	13.78	9.00	15.00	1.12
	Melted wire volume (mm^3)	Mv	0.74	5.6×10^{-5}	10.54	0.98
Output	Penetration depth (mm)	Depth	1.95	0.00	3.62	0.89
	Reinforcement height (mm)	Height	1.77	0.00	3.33	0.86
	Bead width (mm)	Width	5.94	2.67	8.21	1.28

2.3 Exploratory Data Analysis

This section explores the effect of input variables on bead geometry through the Pearson correlation calculations. Fig. 1 illustrates the correlation matrix that relates the correlated values and correlations between the depth, height, and width features. The correlated values indicate the correlation strength, where a correlation value lower than zero indicates a negative correlation while a zero value indicates no correlation. Based on Fig. 1(a), the Depth was inversely proportional to Wv and Ul but directly proportional to Ws and V. The decrease in-depth with the increase in Wv was due to the shorter period for the arc force to penetrate the surface of the base metal. In addition, since the Ul regulates the degree of heating resistance of the wire before it melts in the weld pool, a shorter Ul leads to a smaller heating effect and deeper penetration, and vice versa. Considering the Mv value, which promotes the fluid motion in the weld pool, the penetration depth increases as the Mv increases (Kim and Na, 1995). Besides, the increase in V and Ws caused more molten metal to be deposited in the groove, hence, increasing the Depth (Kamble, and Rao, 2013).

In contrast, Fig. 1(b) shows that the Height decreased with the increase in Ws, V, Ul, and Mv. However, a smaller Wv increased the metal deposited per unit length, resulting in larger Height. Additionally, an increase in V caused more melting of the base metal, resulting in a slight Height increase. Conversely, the increase in Ws led to less deposition of molten metal on top of the base metal, consequently decreasing the Height (Kamble & Rao, 2013). Furthermore, Fig. 1(c) depicts that the increase in Width correlates directly with the increase in Ws and V but inversely decreases with the increase in Wv and Ul. The increase in V formed wider, flatter, and less penetrated weld beads. Moreover, the increase in Ws corresponded to the higher amount of deposited metal on the base metal and increasing the Width. A higher Ws also decreased both the volume of deposited metal and the heat input per unit length (Kamble & Rao, 2013).



(a) (b) (c)

Fig. 1 Correlation matrix among all the features with: (a) Depth, (b) Height, and (c) Width

2.4 Data Preprocessing

The collected data consists of a broad range of values and is unfit to be trained by the ML models in their original form. Therefore, each dataset was normalized within the range of -1 to +1 (Singh and Singh, 2020). The plot matrix of the normalized data is shown in Fig. 2:

$$x'_{i,n} = \left[2 \times \frac{x_{i,n} - \min(x_i)}{\max(x_i) - \min(x_i)} \right] - 1 \quad (1)$$

where $x'_{i,n}$ represents the final form of the normalized data, $x_{i,n}$ refers to the value to be normalized, and $\max(x_i)$ $\min(x_i)$ are the maximum and minimum values of i^{th} attribute, respectively.

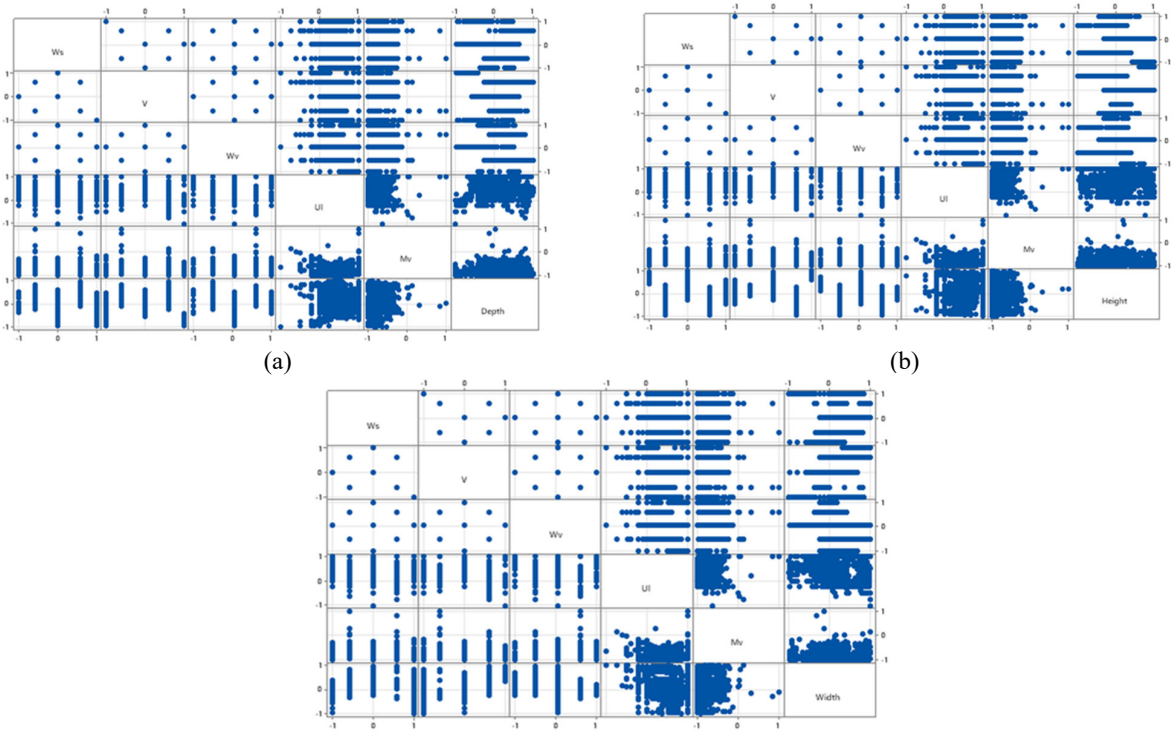


Fig. 2 Plot matrix of normalized dataset with: (a) Depth, (b) Height, and (c) Width

Following the normalization, the dataset was randomly divided into two groups, namely the training set and the test set. Accordingly, the training set contains 70% dataset and was used to develop the models via supervised learning. The remaining 30% dataset in the test set was utilized to evaluate the network performances and generalization capabilities.

2.5 ML Models

The Regression Learner App was employed for the ML analysis, and the comparison was performed for three datasets, including training, testing, and overall, using a desktop Intel Core i7-4770 CPU @3.4 GHz, 16 GB RAM. The parameters in the proposed models were updated during the training to obtain the most accurate predictions. The identified parameters were then saved into a file and are referred to as the model. The model is fixed once it is saved unless additional training was performed and the file was overwritten using the new parameters. Table A.1 presents the parameter settings of the proposed models.

2.5.1 Linear Regression (LR)

LR is one of the easiest supervised learning algorithms and is applied to analyze the relationship between a dependent variable and a single independent variable, referred to as the Simple Linear Regression (SLR), or more than one independent variable, known as Multiple Linear Regression (MLR). The LR relationship is modeled to predict the future state of the dependent variable based on linear equations, which contrasts with non-linear regression models that are developed using

non-linear equations (Tripepi et al., 2008). The regression line is employed to acquire the relevant information, which is derived using Eq. (2):

$$y = \beta_0 + \beta_1 x + \varepsilon \tag{2}$$

where y is the predicted dependent variable value, β_0, β_1 are the intercept and slope, respectively, x is a given independent variable value, and ε is a random component.

MLR can be used to interpret the relationship direction between the dependent variable and the independent variables. The independent factors explain the whole variation in the dependent variable. Eq. (3) describes the mathematical model of MLR for a total of *n* number of independent variables:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \varepsilon \tag{3}$$

where $\beta_1, \beta_2, ..., \beta_n$ are the regression coefficients of $x_1, x_2, ..., x_n$, respectively. As other independent variables are fixed in the MLR model, the slope β_i corresponds to the increased dependent variable (y) as a result of the unit increment in the respective independent variable (x_i) (Montgomery et al., 2021).

2.5.2 Regression Tree (RT)

RT has received considerable interest ever since it was first proposed by Leo Breiman in 1984 (Loh, 2011). Basically, a case is denoted as (x, y), where x and *y* represent the attribute vector and target, respectively, and a regression function is utilized to evaluate the impact of *x* on *y* when the relationship between x and y is altered. The construction of an RT involves three consecutive steps: (1) applying a learning dataset for tree growth; (2) utilizing a test dataset or cross-validation for tree pruning; and (3) selection of the best-pruned tree.

RT makes use of the Decision Tree (DT), which is a powerful decision support tool in ML. DT follows the concepts of training, validation, and test sets, as well as the concern of overfitting versus under-fitting. While a tree is, graph-theoretically-speaking, the fundamental model in DT learning, it is crucial to recognize a stylized control flow that overlays the tree structure. Primarily, a decision-type question is inquired at each inner node of the tree, which also contains the root. The next child node is determined according to the given response. Finally, the leaf node represents the classification of the dataset since each leaf node exhibits its class level. DT learning has the advantage of solving more complex decision boundaries compared to other learning methods, such as logistic regression. Moreover, DT learning is beneficial for non-linearly isolated samples due to the absence of a hyperplane that segregates samples into two different classes. However, the capacity of DTs to describe complex decision boundaries can be a trap in itself as a result of the potential occurrence of overfitting unless other methods are used, such as the "pruning the tree" technique. The intended output is evaluated by assigning a value to each final region. As given in Eq. (4), the tree is referred to as a function defined by h with R_j referring to the disjoint areas allocated to each leaf of the DT:

$$h(x) = \sum_{j=1}^{j} b_j 1_{\{x \in R_j\}} \tag{4}$$

All types of RTs, namely fine tree, medium tree and coarse tree, were employed in this study.

2.5.3 Support Vector Machine (SVM)

SVMs, which were developed by Vladimir Vapnik in the 1990s in accordance with the statistical learning theory, is widely used in classification applications. Generally, an SVM measures the extreme boundaries and draws the edges, usually termed hyperplanes, which separate the dataset into two classes. The data points on either side of the hyperplane that is closest to the hyperplane are called Support Vectors (SVs) which are used to plot the boundary line. Given that non-ideal decision limits would misclassify new data points, extreme data assists in identifying the limitations called support vectors and they are inclined to dismiss the training data points (Mur et al., 2020). Unlike classical regression analysis, which refers to the identification of the function $f(x)$ with minimal difference between the empirically experimental responses and predictions for all training datasets, one of the key elements of SVM is the persistence to achieve optimum generalized performance with the smallest generalized error limit compared to the smallest observed training error. The generalized error limit describes the combined arrangement term, which minimizes the complex collection of functions as well as the training error.

A training set in the regression process is expressed as the following:

$$Z = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\} \quad (5)$$

where x_m and y_m are the independent and dependent variables of the regression model, respectively.

$$y_i = \sum_{i=1}^{n} (\alpha_i - \beta_i) \times (x, x_i) + b \quad (6)$$

where b defines the biasness and both α_i and β_i represent Lagrange multipliers. In terms of non-linear cases, the added Kernel function in the modified equation is expressed as:

$$y_i = \sum_{i=1}^{n} (\alpha_i - \beta_i) \times k(x, x_i) + b \quad (7)$$

where $k(x, x_i)$ is the Kernel function. Note that this study utilized all the Kernel functions, including linear, quadratic, cubic, and Gaussian terms.

2.5.4 Ensembles of Tree (ET)

Apart from a relatively swift training process and the ability to tolerate massive amounts of data, DTs are prevalently applied since they often provide a discreet visual representation of the categorization of ML systems. DTs are also frequently exploited in ensemble learning approaches, such as boosted trees and random forests. Meanwhile, overfitting is specifically well-served through bagging methods. The bagging is a major class of ML algorithms that contains random forests, where multiple DTs are learned in random forests before they are combined to form a single graph-theoretic forest. Numerous DTs in the forest categories are characterized by the new feature vectors. The final classification is performed by integrating these individual categories.

DTs exhibit a restricted generalization although it is one of the most effective and understandable classification methods. Consequently, the samples in DTs are exposed to a low biasness but a substantial variance. Instead of applying a single DT, multiple DTs are joined together to provide enhanced prediction results, which forms the basis of ET. The working principle of ET is to generate a strong learner by uniting a number of weak learners (Mendes-Moreira et al., 2012). Among the most typical ET training approaches are bagging and boosting. Although both approaches have different strategies to generate the selected trees, all data for each new tree-building was randomly chosen.

Combining an RT and boosting forms a hybrid boosted tree, which is repeatedly fitted with various DTs, including the random forest model, to enhance the accuracy of the model. In contrast, the baggage approach is applied in the random forest model, which records a similar probability for the selection of successive samples for each occurrence. The input data are weighted in the trees and boosted tree is utilized as a boosting strategy. Since the weights are imprecisely estimated for this model, the prior tree is chosen as the new tree. In short, the primary tree fitted to the model would explain the inaccuracy and be designated as a new tree. Hence, a more powerful model with enhanced accuracy is achieved by taking the old tree against a new tree.

Furthermore, the variance of a DT can be reduced by employing a bagging tree. Theoretically, numerous data subsets from the training sample are formed, which are randomly selected through replacement. Each data subset is then applied to train the corresponding DT model, resulting in numerous new models. The average predicted value from several trees is finally employed, which is more accurate and powerful compared to the use of a single DT.

2.5.5 Gaussian Process Regression (GPR)

A GPR is made up of an infinite group of random variables with a fixed joint Gaussian distribution in any of its finite subsets. The model is characterized by a mean function and a covariance function (Richardson et al., 2017). Since the GPR is a linear combination of random variables with a normal distribution, the mean function is commonly regarded to be zero. GPR is also considered a non-linear function distribution and is expressed as:

$$f(x) \sim GP(\mu(x), k(x, x')) \quad (8)$$

where $\mu(x)$ is the mean and $k(x, x')$ refers to the positive-semi definite Kernel function that defines the covariance between any two realizations of $f(x)$ and $f(x')$:

$$k(x, x') = \mathrm{cov}(f(x), f(x')) \quad (9)$$

Both the means and Kernel parameters are often assumed to be zero ($\mu x = 0$) and θ ($k\left(x, x' \middle| \theta\right)$), respectively. The $f\left(X\right) = \left(f\left(x_1\right), f\left(x_2\right), ..., f\left(x_n\right)\right)$ has a joint multivariate Gaussian distribution for any infinite input collection of $X = \left(x_1, x_2, ..., x_n\right)$:

$$f\left(x\right) \sim N\left(0, K_{XX}\left(\theta\right)\right) \tag{10}$$

where the Kernel function defines the elements of the N-by-N covariance matrix as:

$$\left[K_{XX}\left(\theta\right)\right]_{i,j} = k\left(x_i, x_j \middle| \theta\right) \tag{11}$$

The covariance function assists in determining the specific properties of the model, including periodicity, stationarity, and smoothness. Eq. (12) shows that the basic and broadly used GPR is composed of a simple zero mean and a squared exponential covariance function, while the value of r is expressed in Eq. (13):

$$k\left(x, x'\right) = \sigma_f^2 \exp\left[\frac{-r}{2}\right] \tag{12}$$

$$r = \frac{\left|x - x'\right|^2}{l^2} \tag{13}$$

where l refers to the length scale and σ_f is the model noise. Both σ_f and l are hyper-parameters and affect the performance of the Gaussian Process. In the present study, all the Kernel functions were used, including the rational quadratic, Matern 5/2, squared exponential, and exponential function. The formulation and information of these functions were based on those in past literature (Asante-Okyere et al., 2018).

2.5.6 Artificial Neural Network (ANN)

ANN is a sophisticated data processing system, which is composed of basic elements known as neurons based on the human brain system. Neurons of one layer are built by interacting and combining neurons with each other via connection links at specific weights and each neuron receives a weighted input sum. Occasionally, the activation function refers to the unique transfer function of every neuron, and the output signal is developed when the weighted input sum surpasses a specified threshold. This activity is described as feed-forward given the forward direction of the information flow. In addition, the backpropagation and gradient descent is usually applied to minimize the error.

The major aspect of ANN is its capacity to learn and predict bead geometry. A collection of input/output data can be utilized by ANN to construct non-linear structures (Abbasi and El Hanandeh, 2016). For this study, the proposed ANN model was composed of two hidden layers. The model was trained using the Levenberg-Marquardt backpropagation algorithm. Each neuron received weighted inputs from the uppermost layer before the total weighted input was calculated and an activation function was applied to generate the outputs for the subsequent layer. The input sets of $X = \left(x_1, x_2, ..., x_n\right)$ were multiplied by the vector weight $W_j = \left(w_{j1}, w_{j2}, ..., w_{jn}\right)$. Finally, the biasness was added to the expression, as follows:

$$Y_j = \sum_{i=1}^{n} w_{ji} x_i + b_j \tag{14}$$

where Y_j is the weighted output sum.

2.6 Cross-Validation

The concern regarding the random separation of the dataset into the training set and testing set is that certain representative samples may be missing during the training process. Therefore, cross-validation is an important evaluation to ensure that the developed model is robust and that every data point is represented in the training process. In this study, the five-fold cross-validation was carried out by randomly dividing the dataset into five equal-sized sets (or five-folds) to prevent overfitting in the developed model. Four of the pairs were used for training, while the other one was used for testing. The validation was performed five times with each time excluding one fold out of the training and used for testing. Eventually, the average iteration was used to estimate the accuracy of the model (Apaydın, 2004).

2.7 Performance Indices

Three statistical metrics were used to examine the prediction model performances, namely the Coefficient of Determination (R^2), Root Means Square Error (RMSE), and Mean Absolute Error (MAE), following Eqs. (15)–(17), respectively. The R^2 value represents the correlation between the prediction model (y^p) and the corresponding target (y^t). The R^2 value ranges from zero to 1 with an R^2 equal to 1 indicating a perfect correlation, while R^2 is negative when a trained model is worse than the controlled sample.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left[y_i^p - y_i^t\right]^2}{\sum_{i=1}^{n}\left[y_i^p - y^m\right]^2}, \qquad y^m = \frac{1}{n}\sum_{i=1}^{n} y_i^p \tag{15}$$

where n is the number of experimental data. Additionally, the RMSE and MAE are exceptional metrics to verify the accuracy of the prediction models, being close to 0. The RMSE is always positive and is equal to that of the response. Similar to the RMSE, MAE is also always positive but is less sensitive to outliers. The RMSE was calculated to measure the standard deviation of the residuals, while MAE was computed to determine the average absolute difference between y^t and y^p values in the dataset.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i^t - y_i^p\right)^2} \tag{16}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i^t - y_i^p\right| \tag{17}$$

Following the construction of a predictive approach, the performance of the model was evaluated in terms of R^2, RMSE, and MAE for bead depth, height, and width, respectively. The best model should achieve the highest R^2 and the lowest RMSE and MAE.

3. Results and Discussion

Prior to the performance evaluation of each model, the respective model was applied to make predictions using the training set before the predictions were compared to the target values based on the various metric calculations. The model assessment provides valuable information regarding the generalized model's prediction to that of the new data. Table A.2 shows the R^2, RMSE, and MAE metrics according to the respective developed models in order to quantitatively analyze their performance.

LR models are commonly used as the first test model since they possess linear predictors as part of the model parameters, which makes them easier to interpret and achieve a faster prediction-making process. Nevertheless, LR models often exhibit lower predictive accuracy due to the highly constrained form of these models. In this study, a wide range of LR models was tested, including standard linear, robust, interactions, and stepwise. The interaction linear method provides a greater understanding of the association between the model's variables by supplementing interaction terms to a regression model to assist more hypothesis testing. On the contrary, the robust linear is an alternative approach to the least-squares regression with lower sensitivity to outliers compared to the standard LR. This approach is able to identify influential observations, where the weight value of each observation in the robust regression is regulated by a special influence function. Meanwhile, the stepwise linear approach iteratively evaluates the statistical significance of each independent variable. The final variables to be used in the final LR model are selected by adding or removing the independent variable in each iteration. Based on the results, the best-fitted LR model recorded an interaction linear with an off-robust option and a stepwise linear. The results showed that the overall R^2 values of the best-fitted LR models for the depth, width, and height of bead geometry prediction were 0.41, 0.61, and 0.52, respectively.

The RT approach is also simple to interpret, offers a rapid fitting and predicting process, and occupies a low memory usage. Basically, growing smaller trees with fewer larger leaves could inhibit overfitting, while the leaf size can be adjusted using the minimum leaf size setting. A prediction is made beginning at the top node before the predictor values are examined at each node to determine the next branch to follow. Once a branch reaches a leaf node, the responding value is set to that of the corresponding node value. A variety of RT methods were employed in this study, including the fine tree, medium tree, and coarse tree approach. For comparison, a fine tree with plenty of small leaves usually exhibits a high accuracy on the training data but may not detect comparable accuracy using an independent test set. While a very leafy tree is inclined to overfit, the accuracy of the validation is regularly much lower compared to its training accuracy. Conversely, a medium tree with medium-sized leaves is appropriate for a less flexible response function. On the contrary, a coarse tree with fewer large leaves exhibits a low training accuracy. However, it can achieve higher robustness, where its training accuracy can be close

to that of a representative test set. According to the findings, the best fitted RT model was the fine tree with a minimum leaf size of four, and the overall R^2 values of the bead geometry prediction were 0.91, 0.88, and 0.90 for the depth, width, and height, respectively.

Although linear SVMs are easily interpreted, the method could suffer low predictive accuracy. Oppositely, non-linear SVMs can be more accurate but are more complex to interpret. The support vectors highly influence the function of the SVM to predict new values. Prior to the training via SVM, numerous Kernel functions were applied to the data to determine the non-linear transformation, quadratic, cubic, and Gaussian. Generally, the box constraint regulates the penalty imposed on observations with huge residuals. A more flexible model is achieved with a larger box constraint, while a rigid model with less sensitivity to overfitting is achieved with a smaller box constraint. Furthermore, smaller prediction errors than the epsilon value are dismissed and considered equal to zero. A smaller epsilon value also forms a more flexible model. The SVs are data points with errors larger than the epsilon value. The function the SVM uses to predict new values depends only on the SVs. The epsilon value controls the width of the epsilon insensitive zone, which is used to fit the training data. The epsilon value also affects the number of SVs used to construct the regression function. A larger epsilon value implies that fewer SVs are selected. However, a larger epsilon value results in more flat estimates. Moreover, the epsilon value determines the level of accuracy of the approximated function. It relies entirely on the target values in the training set. A poor result would be expected if the epsilon value is larger than the range of the target values. Contrarily, a zero value epsilon may lead to overfitting. The epsilon must therefore be chosen to reflect the data in a certain manner. Choosing the epsilon to be a certain accuracy would only guarantee the accuracy on the training set, which oftentimes achieves a certain overall accuracy. Therefore, it is necessary to choose a slightly smaller epsilon value. Besides, the Kernel scale, which varies substantially, controls the size of the predictors. A smaller Kernel scale generates a model with greater flexibility. The results indicate that the fine Gaussian SVM achieved the best-fitted model with automatic box constant mode with an overall R^2 value for the depth, width, and height of the bead geometry prediction at 0.87, 0.82, and 0.87, respectively.

Meanwhile, numerous weak learners were combined into a single high-quality ET model. While both approaches involved RT learners, bagging or bootstrap aggregating were used to produce bagged trees, and the least-squares boosting was applied to form boosted trees. In addition, the learning shrinkage rate was specified for boosted trees, which is usually set to an initial value of 0.1. The ET model would require more learning iterations if the learning rate is set to lower than 1 but commonly achieves higher accuracy. Based on the findings, the best ET model was the bagged trees with a minimum leaf size of eight and an overall R^2 value of 0.85, 0.81, and 0.84 for the depth, width, and height of the bead geometry prediction, respectively.

In terms of GPR models, they are difficult to interpret but provide highly accurate results. A probability distribution is used to model the response over a space of functions. The flexibility of the preset models is automatically selected to simultaneously limit the training error and protect against overfitting. For this study, the association of the response as a function of the distance between the predictor values was identified using different Kernel functions, including the rational quadratic, Matern 5/2, squared exponential, and exponential. As such, the scale of the correlation length is similar for all predictors using an isotropic Kernel. In contrast, each predictor has a unique correlation length scale using a non-isotropic Kernel, which can enhance the accuracy of the model but make the model fitting relatively slower. Overall, the exponential GPR recorded the best-fitted model with R^2 values of 0.91, 0.85, and 0.89 for the depth, width, and height of the bead geometry prediction, respectively.

While the ANN model has the capability to distinguish complicated non-linear relationships between independent and dependent variables, the ‘black box’ characteristic of the ANN model prohibits users from accessing the precise decision-making process. Nevertheless, the flexibility of the model is enhanced in response to the size and number of fully connected layers in the neural network. Additionally, each model is feedforward and fully connected neural network for regression. Based on the results, ANN showed similar properties with a fine tree, which demonstrated a greater predictive ability compared to other models, particularly in predicting the complex relationship between the bead geometry and its process parameters. The overall R^2 values of the ANN model were 0.91, 0.88, and 0.90 for the depth, width, and height of the bead geometry prediction, respectively.

Figs. 3–5 presents the overall performance comparison between the LR, RT, SVM, ET, GPR, and ANN models in terms of R^2, RMSE, and MAE for the bead depth, height, and width, respectively. According to the bead depth prediction in Fig. 3, the fine tree and ANN models achieved the highest R^2 and the lowest RMSE and MAE. The R^2, RMSE, and MAE of the fine tree were 0.91, 0.1502, and 0.1032 for the training data and 0.90, 0.1595, and 0.1092 for the test data, respectively. Comparatively, the test data recorded a lower R^2 value (1.10%) and a higher RMSE and MAE values (6.19% and 5.81%) than the training data, respectively. Conversely, the R^2, RMSE, and MAE of the ANN model were 0.91, 0.1503, and 0.1034 for the training data and 0.90, 0.1596, and 0.1094 for the test data, respectively. The test data demonstrated a lower R^2 value (1.10%) and a higher RMSE and MAE (6.19% and 5.80%) compared to the training data, respectively. As previously expressed, a model is considered highly predictive when it possesses a lower RMSE and MAE value.

Fig. 3 Performance indices of the bead depth prediction for each model in terms of (a) R^2, (b) RMSE, and (c) MAE

Fig. 4 illustrates the R^2, RMSE, and MAE of the bead height prediction of each model. Similar to the results in Fig. 3, the fine tree and ANN demonstrated superior performance in all three indices compared to other ML models. The R^2, RMSE, and MAE of the fine tree model were 0.90, 0.1657, and 0.1238 for the training data and 0.89, 0.1672, and 0.1254 for the test data, respectively. The test data recorded a lower R^2 value (1.11%) and a higher RMSE and MAE (0.90% and 1.29%) compared to the training data, respectively. Additionally, the R^2, RMSE, and MAE of the ANN model were 0.90, 0.1656, and 0.1236 for the training data and 0.89, 0.1670, and 0.1253 for the test data, respectively. The test data recorded a lower R^2 value (1.10%) and a higher RMSE and MAE (0.84% and 1.38%) compared to the training data, respectively.

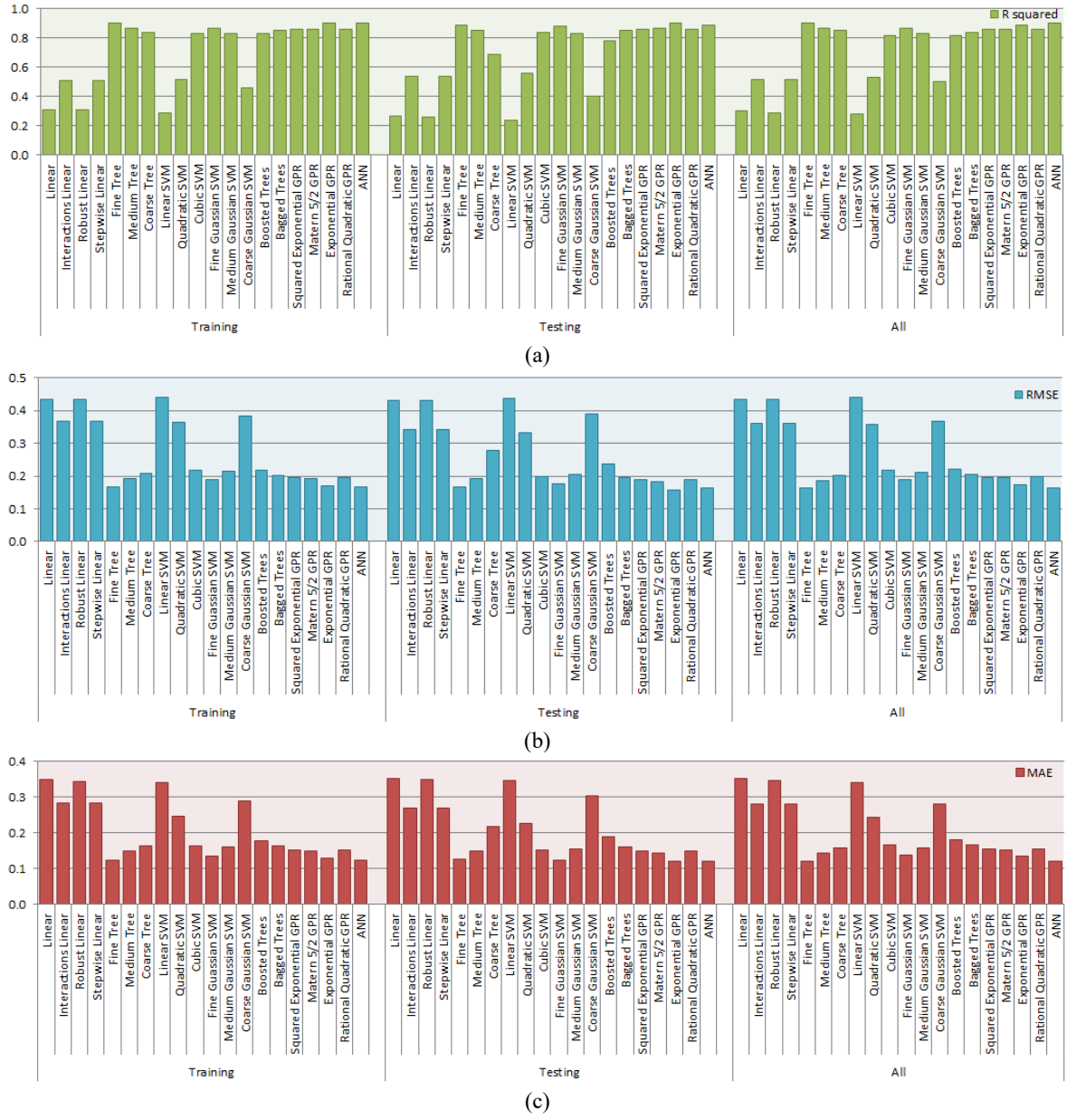


Fig. 4 Performance indices of the bead height prediction for each model in terms of (a) R^2, (b) RMSE, and (c) MAE

According to the R^2, RMSE, and MAE of the bead width prediction of each model in Fig. 5, the fine tree and ANN showed exceptional accuracies compared to other ML models. The R^2, RMSE, and MAE of the fine tree model were 0.88, 0.1578, and 0.1016 for the training data and 0.87, 0.1749, and 0.1147 for the test data, respectively. The test data achieved a lower R^2 value (1.14%) and a higher RMSE and MAE (10.84% and 12.89%) compared to the training data, respectively. Meanwhile, the R^2, RMSE, and MAE of the ANN model were 0.88, 0.1579, and 0.1018 for the training data and 0.87, 0.1751, and 0.1149 for the test data, respectively. In addition, the test data recorded a lower R^2 value (1.14%) and a higher RMSE and MAE (10.89% and 12.87%) compared to the training data, respectively.

(a)

(b)

(c)

Fig. 5 Performance indices of the bead width prediction for each model in terms of (a) R^2, (b) RMSE, and (c) MAE

Based on the three performance indices, the study revealed that the fine tree and ANN were the best-fitted ML models. In terms of the precise analytical approximations, the trend of the R^2, RMSE, and MAE assessment standards can be arranged in a decreasing order as follows: Fine Tree = ANN > Exponential GPR > Fine Gaussian SVM = Medium Tree > Rational Quadratic GPR > Matern 5/2 GPR = Squared Exponential GPR > Bagged Trees > Medium Gaussian SVM > Boosted Trees > Coarse Tree > Cubic SVM > Quadratic SVM > Interactions Linear = Stepwise linear > Coarse Gaussian SVM > Linear > Robust Linear > Linear SVM. Regardless of the slight performance variations, the majority of the developed models demonstrated the ability to learn the patterns and provided excellent prediction performance in terms of the R^2, RMSE, and MAE.

4. Conclusion

This study investigated the comparison of various ML models on the bead geometry in a GMAW process. The findings demonstrated the superior accuracy of the fine tree and ANN models in all three indices. The R^2, RMSE, and MAE values showed that both models accurately predicted the bead geometry, including the bead depth for training (0.91, 0.15, and 0.10), testing (0.90, 0.16, and 0.11), and all dataset (0.91, 0.15, and 0.10), the bead width for training (0.88, 0.16, and 0.10), testing (0.87, 0.18, and 0.12), and all dataset (0.88, 0.16, and 0.10), and bead height for training (0.90, 0.17, and 0.12), testing (0.89, 0.17, and 0.13), and all dataset (0.90, 0.16, and 0.12), respectively. The Regression Learner App models produced a MATLAB

software code that precisely predicted the bead geometry with respect to the measurements. Overall, the findings in this study can be potentially utilized to predict the optimum bead geometry in a cost-effective and time-efficient manner.

Acknowledgement

This research received no external funding.

References

Abbasi, M., & El Hanandeh, A. (2016). Forecasting municipal solid waste generation using artificial intelligence modelling approaches. *Waste Management, 56*, 13-22.

Apaydın, E. (2004). Introduction to Machine Learning (Adaptive Computation and Machine Learning).

Asante-Okyere, S., Shen, C., Yevenyo Ziggah, Y., Moses Rulegeya, M., & Zhu, X. (2018). Investigating the predictive performance of Gaussian process regression in evaluating reservoir porosity and permeability. *Energies, 11*(12), 3261.

Chandrasekaran, R. R., Benoit, M. J., Barrett, J. M., & Gerlich, A. P. (2019). Multi-variable statistical models for predicting bead geometry in gas metal arc welding. *The International Journal of Advanced Manufacturing Technology, 105*(1), 1573-1584.

Dong, H., Cong, M., Liu, Y., Zhang, Y., & Chen, H. (2016, June). Predicting characteristic performance for arc welding process. *In 2016 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER) (pp. 7-12). IEEE*.

Dong, H., Huff, S. A., Cong, M., & Zhang, Y. (2017, July). Backside weld bead shape modeling using support vector machine. *In 2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER) (pp. 277-282). IEEE*.

Dutta, P., & Pratihar, D. K. (2007). Modeling of TIG welding process using conventional regression analysis and neural network-based approaches. *Journal of Materials Processing Technology, 184*(1-3), 56-68.

Fauzi, E. I., Samad, Z., Jamil, M. C., Nor, N. M., & Boon, G. P. (2018). Parametric modeling of metal inert gas (MIG) welding process using second-order regression model analysis. *Journal of Advanced Manufacturing Technology (JAMT), 12*(1 (2)), 367-382.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *Overview of supervised learning. In The elements of statistical learning (pp. 9-41)*. Springer, New York, NY.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2, pp. 1-758). New York: springer.

Kamble, A. G., & Rao, R. V. (2013). Experimental investigation on the effects of process parameters of GMAW and transient thermal analysis of AISI321 steel. *Advances in Manufacturing, 1*(4), 362-377.

Kim, J. W., & Na, S. J. (1995). A study on the effect of contract tube-to-workpiece distance on weld pool shape in gas metal arc welding. *Welding Journal, 74*(5).

Liang, R., Yu, R., Luo, Y., & Zhang, Y. (2019). Machine learning of weld joint penetration from weld pool surface using support vector regression. *Journal of Manufacturing Processes, 41*, 23-28.

Loh, W. Y. (2011). Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery, 1*(1), 14-23.

Martinez, R. T., Bestard, G. A., & Alfaro, S. C. A. (2021). Two gas metal arc welding process dataset of arc parameters and input parameters. *Data in Brief, 35*, 106790.

Mendes-Moreira, J., Soares, C., Jorge, A. M., & Sousa, J. F. D. (2012). Ensemble approaches for regression: A survey. *Acm computing surveys (csur), 45*(1), 1-40.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2021). *Introduction to linear regression analysis*. John Wiley & Sons.

Mur, R., Díaz, I., & Rodríguez, M. (2020). Comparative Study of Surrogate Modelling Techniques Applied to Three Different Chemical Processes. *In Computer Aided Chemical Engineering* (Vol. 48, pp. 145-150). Elsevier.

Richardson, R. R., Osborne, M. A., & Howey, D. A. (2017). Gaussian process regression for forecasting battery state of health. *Journal of Power Sources, 357*, 209-219.

Singh, D., & Singh, B. (2020). Investigating the impact of data normalization on classification performance. *Applied Soft Computing, 97*, 105524.

Tafarroj, M. M., & Kolahan, F. (2018). A comparative study on the performance of artificial neural networks and regression models in modeling the heat source model parameters in GTA welding. *Fusion Engineering and Design, 131*, 111-118.

Tham, G., Yaakub, M. Y., Abas, S. K., Manurung, Y. H., & Jalil, B. A. (2012). Predicting the gmaw 3f t- fillet geometry and its welding parameter. *Procedia Engineering, 41*, 1794-1799.

Tripepi, G., Jager, K. J., Dekker, F. W., & Zoccali, C. (2008). Linear and logistic regression analysis. *Kidney international, 73*(7), 806-810.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural computation, 8*(7), 1341-1390.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation, 1*(1), 67-82.

Wu, S., Gao, H., Zhang, W., & Zhang, Y. M. (2017). Analytic weld pool model calibrated by measurements part 1: Principles. *Weld. J, 96*(6), 193s-202s.

Yang, L. J., Bibby, M. J., & Chandel, R. S. (1993). Linear regression equations for modeling the submerged-arc welding process. *Journal of Materials Processing Technology, 39*(1-2), 33-42.

Appendix A

Table A.1 Parameters settings of ML models

Model	Parameters settings
Linear	Terms : Linear Robust option : Off
Interactions Linear	Terms : Interactions Robust option : Off
Robust Linear	Terms : Linear Robust option : On
Stepwise Linear	Terms : Linear Upper bound on terms : Interactions Maximum number of steps : 1000
Fine Tree	Minimum leaf size : 4 Surrogate decision splits : Off
Medium Tree	Minimum leaf size : 12 Surrogate decision splits : Off
Coarse Tree	Minimum leaf size : 36 Surrogate decision splits : Off
Linear SVM	Kernel function : Linear Kernel scale : Automatic Box constraint : Automatic Epsilon : Automatic Standardize data : true
Quadratic SVM	Kernel function : Quadratic Kernel scale : Automatic Box constraint : Automatic Epsilon : Automatic Standardize data : true
Cubic SVM	Kernel function : Cubic Kernel scale : Automatic Box constraint : Automatic Epsilon : Automatic Standardize data : true
Fine Gaussian SVM	Kernel function : Gaussian Kernel scale : 0.56 Box constraint : Automatic Epsilon : Automatic Standardize data : true
Medium Gaussian SVM	Kernel function : Gaussian Kernel scale : 2.2 Box constraint : Automatic Epsilon : Automatic Standardize data : true
Coarse Gaussian SVM	Kernel function : Gaussian Kernel scale : 8.9 Box constraint : Automatic Epsilon : Automatic Standardize data : true

Table A.1 *(continued)*

Model	Parameters settings
Boosted Trees	Minimum leaf size : 8 Number of learners : 30 Learning rate : 0.1
Bagged Trees	Minimum leaf size : 8 Number of learners : 30
Squared Exponential GPR	Kernel function : Squared Exponential Use isotropic kernel : true Kernel scale : Automatic Kernel sigma : Automatic Sigma : Automatic Standardize : true Optimize numeric parameters : true
Matern 5/2 GPR	Kernel function : Matern 5/2 Use isotropic kernel : true Kernel scale : Automatic Kernel sigma : Automatic Sigma : Automatic Standardize : true Optimize numeric parameters : true
Exponential GPR	Kernel function : Exponential Use isotropic kernel : true Kernel scale : Automatic Kernel sigma : Automatic Sigma : Automatic Standardize : true Optimize numeric parameters : true
Rational Quadratic GPR	Kernel function : Rational Quadratic Use isotropic kernel : true Kernel scale : Automatic Kernel sigma : Automatic Sigma : Automatic Standardize : true Optimize numeric parameters : true
Artificial Neural Network	Network type : Feed-forward backprop Training functions : Trainlm Adaption learning function : Learngdm Number of layers : 2 Number of neurons : 10 Transfer function : Tansig

Table A.2 Comparison of performance indices of different models

Model	Dataset	Depth			Width			Height		
		RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE
Linear	Training	0.4440	0.17	0.3633	0.3209	0.50	0.2580	0.4342	0.31	0.3489
	Testing	0.4365	0.22	0.3552	0.3326	0.52	0.2675	0.4306	0.27	0.3531
	All	0.4421	0.19	0.3624	0.3271	0.50	0.2634	0.4337	0.30	0.3509
Interactions Linear	Training	0.3693	0.43	0.3007	0.2839	0.61	0.2274	0.3659	0.51	0.2843
	Testing	0.3808	0.40	0.3135	0.2905	0.63	0.2349	0.3408	0.54	0.2683
	All	0.3749	0.41	0.3054	0.2888	0.61	0.2329	0.3604	0.52	0.2810
Robust Linear	Training	0.4479	0.16	0.3559	0.3237	0.49	0.2534	0.4353	0.31	0.3441
	Testing	0.4422	0.19	0.3466	0.3370	0.50	0.2649	0.4315	0.26	0.3492
	All	0.4464	0.17	0.3536	0.3290	0.49	0.2601	0.4348	0.29	0.3461
Stepwise Linear	Training	0.3697	0.43	0.3009	0.2839	0.61	0.2274	0.3662	0.51	0.2849
	Testing	0.3821	0.40	0.3132	0.2911	0.63	0.2350	0.3421	0.54	0.2697
	All	0.3748	0.41	0.3061	0.2888	0.61	0.2329	0.3605	0.52	0.2812
Fine Tree	Training	0.1502	0.91	0.1032	0.1578	0.88	0.1016	0.1657	0.90	0.1238
	Testing	0.1595	0.90	0.1092	0.1749	0.87	0.1147	0.1672	0.89	0.1254
	All	0.1510	0.91	0.1017	0.1623	0.88	0.1042	0.1626	0.90	0.1199
Medium Tree	Training	0.1780	0.87	0.1263	0.1962	0.81	0.1336	0.1919	0.87	0.1481
	Testing	0.1997	0.84	0.1471	0.2200	0.79	0.1522	0.1936	0.85	0.1504
	All	0.1765	0.87	0.1233	0.1920	0.82	0.1296	0.1869	0.87	0.1432
Coarse Tree	Training	0.2191	0.80	0.1637	0.2308	0.74	0.1636	0.2081	0.84	0.1641
	Testing	0.2347	0.77	0.1743	0.2672	0.69	0.1887	0.2796	0.69	0.2193
	All	0.2022	0.83	0.1481	0.2140	0.75	0.1496	0.2007	0.85	0.1577
Linear SVM	Training	0.4681	0.08	0.3492	0.3305	0.47	0.2505	0.4407	0.29	0.3401
	Testing	0.4622	0.12	0.3421	0.3455	0.48	0.2623	0.4381	0.24	0.3455
	All	0.4656	0.10	0.3479	0.3371	0.47	0.2574	0.4406	0.28	0.3423
Quadratic SVM	Training	0.3272	0.55	0.2223	0.2644	0.66	0.1850	0.3637	0.52	0.2480
	Testing	0.3686	0.44	0.2366	0.2651	0.69	0.1826	0.3325	0.56	0.2268
	All	0.3418	0.51	0.2295	0.2660	0.67	0.1883	0.3562	0.53	0.2433
Cubic SVM	Training	0.2153	0.81	0.1442	0.2287	0.75	0.1524	0.2174	0.83	0.1634
	Testing	0.2218	0.80	0.1478	0.2344	0.76	0.1590	0.1990	0.84	0.1527
	All	0.2596	0.80	0.1632	0.2350	0.74	0.1643	0.2188	0.82	0.1667
Fine Gaussian SVM	Training	0.1687	0.88	0.1096	0.1846	0.83	0.1139	0.1886	0.87	0.1353
	Testing	0.1585	0.90	0.1086	0.1855	0.85	0.1191	0.1751	0.88	0.1226
	All	0.1744	0.87	0.1132	0.1968	0.82	0.1222	0.1902	0.87	0.1381
Medium Gaussian SVM	Training	0.1984	0.84	0.1357	0.2126	0.78	0.1365	0.2159	0.83	0.1605
	Testing	0.2134	0.81	0.1478	0.2154	0.80	0.1491	0.2048	0.83	0.1544
	All	0.1982	0.84	0.1373	0.2110	0.79	0.1422	0.2111	0.83	0.1582
Coarse Gaussian SVM	Training	0.3984	0.34	0.2812	0.2924	0.58	0.2055	0.3844	0.46	0.2890
	Testing	0.3962	0.35	0.2822	0.3043	0.60	0.2257	0.3905	0.40	0.3044
	All	0.3888	0.35	0.2694	0.2832	0.62	0.2033	0.3675	0.50	0.2795
Boosted Trees	Training	0.1949	0.84	0.1515	0.2015	0.80	0.1508	0.2173	0.83	0.1793
	Testing	0.1905	0.85	0.1501	0.2116	0.80	0.1585	0.2355	0.78	0.1903
	All	0.1957	0.84	0.1513	0.2092	0.79	0.1560	0.2201	0.82	0.1818
Bagged Trees	Training	0.1867	0.85	0.1423	0.1996	0.81	0.1484	0.2011	0.85	0.1625
	Testing	0.1977	0.84	0.1544	0.2131	0.80	0.1565	0.1963	0.85	0.1603
	All	0.1896	0.85	0.1447	0.1998	0.81	0.1465	0.2048	0.84	0.1666
Squared Exponential GPR	Training	0.1760	0.87	0.1277	0.1955	0.81	0.1392	0.1962	0.86	0.1536
	Testing	0.1713	0.88	0.1296	0.2074	0.81	0.1514	0.1886	0.86	0.1486
	All	0.1793	0.87	0.1308	0.2024	0.81	0.1439	0.1965	0.86	0.1542
Matern 5/2 GPR	Training	0.1725	0.88	0.1247	0.1891	0.83	0.1333	0.1929	0.86	0.1508
	Testing	0.1655	0.89	0.1248	0.2040	0.82	0.1474	0.1834	0.87	0.1440
	All	0.1750	0.87	0.1271	0.1987	0.81	0.1400	0.1943	0.86	0.1522
Exponential GPR	Training	0.1456	0.91	0.1040	0.1668	0.86	0.1165	0.1684	0.90	0.1307
	Testing	0.1263	0.93	0.0936	0.1802	0.86	0.1261	0.1554	0.90	0.1202
	All	0.1508	0.91	0.1084	0.1787	0.85	0.1245	0.1734	0.89	0.1342
Rational Quadratic GPR	Training	0.1732	0.87	0.1253	0.1845	0.83	0.1294	0.1962	0.86	0.1536
	Testing	0.1707	0.88	0.1292	0.2043	0.82	0.1476	0.1886	0.86	0.1486
	All	0.1769	0.87	0.1287	0.1971	0.82	0.1385	0.1970	0.86	0.1546
Artificial Neural Network	Training	0.1503	0.91	0.1034	0.1579	0.88	0.1018	0.1656	0.90	0.1236
	Testing	0.1596	0.90	0.1094	0.1751	0.87	0.1149	0.1670	0.89	0.1253
	All	0.1510	0.91	0.1017	0.1623	0.88	0.1042	0.1626	0.90	0.1197