Received May 20, 2021, accepted June 22, 2021, date of publication June 24, 2021, date of current version July 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3092221

Software to Predict the Process Parameters of Electron Beam Welding

VADIM S. TYNCHENKO[1,2], (Senior Member, IEEE), SERGEI O. KURASHKIN[2], (Graduate Student Member, IEEE), VALERIA V. TYNCHENKO[3,4], VLADIMIR V. BUKHTOYAROV[1,5], VLADISLAV V. KUKARTSEV[3,6], ROMAN B. SERGIENKO[7], SERGEI V. TYNCHENKO[4,8], AND KIRILL A. BASHMUR[1]

[1]Department of Technological Machines and Equipment of Oil and Gas Complex, School of Petroleum and Natural Gas Engineering, Siberian Federal University, 660041 Krasnoyarsk, Russia
[2]Information-Control Systems Department, Institute of Computer Science and Telecommunications, Reshetnev Siberian State University of Science and Technology, 660037 Krasnoyarsk, Russia
[3]Department of Computer Science, Institute of Space and Information Technologies, Siberian Federal University, 660041 Krasnoyarsk, Russia
[4]Department of Computer Science and Computer Engineering, Institute of Computer Science and Telecommunications, Reshetnev Siberian State University of Science and Technology, 660037 Krasnoyarsk, Russia
[5]Department of Information Technology Security, Institute of Computer Science and Telecommunications, Reshetnev Siberian State University of Science and Technology, 660037 Krasnoyarsk, Russia
[6]Department of Information Economic Systems, Engineering and Economics Institute, Reshetnev Siberian State University of Science and Technology, 660037 Krasnoyarsk, Russia
[7]Gini Gmbh, 80339 Munich, Germany
[8]Department of Digital Control Technologies, Institute of Business Processes Management, Siberian Federal University, 660041 Krasnoyarsk, Russia

Corresponding author: Vadim S. Tynchenko (vadimond@mail.ru)

This work was supported by the Ministry of Science and Higher Education of the Russian Federation "Development of the Theory of Self-Configuring Machine Learning Algorithms for Modeling and Predicting the Characteristics of Components of Complex Systems" under Contract FEFE-2020-0013.

ABSTRACT This paper discusses the problem of choosing the effective process parameters of electron beam welding (EBW). To that end, the research team has developed a mathematical model that applies machine learning to predict the effective process parameters. Since predicting process parameters requires a regression model, this research uses regression analysis algorithms such as the ridge regression and the random forest regressor. The paper analyzes whether these algorithms are applicable to the problem and tests the accuracy of their predictions. To generalize the approach and strengthen the justification of choosing the hyperparameters of the regression algorithms studied herein and considering the high variability of these hyperparameters, the multiobjective optimization technique applicable for this combinatorial problem - an (evolutionary) genetic algorithm - is proposed to determine effective sets of hyperparameters. All the models successfully addressed the task, achieving a forecasting accuracy of at least 89%. The article presents the final form of the ridge regression model describing the dependence of the weld's depth and width: for the weld depth, there is a 2nd degree polynomial dependence with a regularization of 10^{-5}, and for the weld width, there is a 3rd degree polynomial dependence with a regularization of 10^{-4}. An automated system based on this approach that accurately predicts the process parameters is proposed herein. In addition to performing basic modeling functions, the proposed system allows the visualization of the model-predicted data in the form of an interactive plot. This function could be useful for technologists by allowing them to determine the process parameters that ensure the required weld dimensions. Adopting the proposed EBW parameter prediction method in practice will provide decision support for cases when engineers need to test the EBW process or to start making new products.

INDEX TERMS Electron beam welding, choice of process parameters, software, decision support, prediction, ridge regression, random forest, genetic algorithm, algorithm ensembles, machine learning.

I. INTRODUCTION

Currently, the processes of making permanent joints in various aerospace products using electron beam welding, induction soldering, or diffusion bonding are based on reusing preparation and equipment operation parameters that have been well tested before.

Thus, an important part of the process consists of calibrating such parameters and testing them to see whether the results are reproducible and the final products are reliable. However, when one needs to make a different type of joint or the quality of joints needs improvement, then the engineering

The associate editor coordinating the review of this manuscript and approving it for publication was Imran Sarwar Bajwa.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

staff needs not only to calibrate and test a new process but also to conduct pilot studies to find the parameters that will make a product of suitable quality. These are costly steps to take as they take some equipment runtime and working hours on the part of process designers and process testing staff. In addition, each experiment will ultimately and knowingly result in destroying a specimen when testing the quality of permanent joints.

In particular, this paper analyzes the calibration and testing of the process parameters of electron beam welding (EBW).

If the process is a closed system that has various inputs and outputs, then we can model it mathematically and use the model as a tool to predict and optimize the process parameters. The goal here was to develop a mathematical model to predict the process parameters of electron beam welding to provide decision support and therefore help choose better process parameters in both local or global optimum seeking. This, in turn, will help calibrate the process faster and reduce the costs of pilot studies. Such an approach is expected to bring both cost and quality improvements.

In essence, this problem is a regression problem. Regression problems are solvable by the following methods:

- *Polynomial Regression With L2 Regularization:* A polynomial regression can be used, e.g., to run quick statistical testing of cooling systems [1] or to predict energy efficiency [2]. The *ridge* regression can be used to find the extreme value indices [3] and to solve other problems [4]–[8].
- *Random Forest:* This algorithm has been used successfully to estimate battery capacity [9] and for mapping [10], [11].

These are common prediction algorithms. The *ridge* regression has been proven to be an effective wind speed and power predictor [12], precancerous cervical lesion vs normal cervical tissue differentiator based on methylation microarray data [13], or a gas compression index predictor [14]. The lasso regression has been proven effective in prediction model testing [15], predicting Indian banks' failures [16], and predicting malignant pleural mesothelioma patients' survival [17]. In addition, *random forests* can effectively predict the results of a transplantation surgery [18], a building's energy consumption rate [19], sugar cane yields [20], or traffic [21]. Python, a programming language, [22]–[24] was used in this research since it had previously been proven to be great for machine learning [25] including regressions, and because of scikit-learn [26]–[28], a Python module that contains convenient machine learning libraries suitable for various ML tasks [29], [30].

In [31], to solve the problem of making clinical decisions, the authors used the extreme machine learning method. The effectiveness of this method was assessed according to the specified criteria. The approach proposed in this work achieved high efficiency. The authors of [32] propose an alternative learning scheme for the extreme machine learning method, which is based on the chaotic moth-flame optimization strategy. The approach developed by the authors can be used in medicine for the diagnosis of diseases. The study in [33] considers an approach for solving the prediction problem based on the kernel extreme learning machine, which uses the fruit fly optimization algorithm. The diagnostic approach proposed by the authors considered in [34] for detecting paraquat poisoning is based on the extreme learning machine, and the method includes two stages: gas chromatography-mass spectrometer provides raw data and chaos enhanced gray wolf optimization algorithm is adopted to search the optimal feature sets.

The proposed algorithms are common prediction algorithms. By applying different statistical data processing approaches and algorithms, one can build approximated dependencies to help engineering staff make better-educated choices regarding the ranges of variable parameters for testing, whether it is a new process or an existing process that needs to be improved. This paper analyzes how the above could be accomplished for products made by electron beam welding. In addition, the key concept behind algorithms, methods, and software is their versatility.

II. MATERIALS AND METHODS

The input data were collected from experiments conducted to improve the process of making an EBW product comprising two parts of nonhomogeneous materials.

The EBW unit involved in the experiments was designed to provide deep vacuum electron beam welding of assemblies made of stainless steel, titanium, aluminum, or special alloys. This EBW unit was capable of reproducing the process parameters using the available control system.

Welding was performed on simulation samples whose parameters matched those of the intended product.

To make the welding process less energy-intensive, the research team lowered the **welding current (IW)**, raised the **focus current (IF),** increased the **welding speed (VW)** and changed the **sample surface to electron optics distance (FP)**.

The parameters were optimized to minimize the weld dimensions**: depth** and **width**.

As part of this research, the team conducted 72 experiments. The accelerating voltage was constant at 19.8 to 20 kV. The collected data (welding parameters and weld dimensions in the cross-sections of all specimens) are hereinafter referred to as the **dataset**.

The purpose was to find a mathematical pattern [31] that could explain how the process parameters (IW, IF, VW, and FP) were correlated with the weld dimensions (depth and width).

Table 1 provides the **dataset** statistics.

Thus, we obtained the following:

- A dataset represented by:

$$L = \left\{x_{iw_i}, x_{if_i}, x_{vw_i}, x_{fp_i}, y_{depth_i}, y_{width_i}\right\}_{i=1}^{n},$$

where

n is the number of experiments,

x_{iw} is the welding current, mA,

TABLE 1. Dataset statistics.

	IW	IF	VW	FP	Depth	Width
count	72	72	72	72	72	72
mean	45.7	141.33	8.64	78.33	1.20	1.97
std	1.7	5.146	2.06	21.49	0.23	0.28
min	43	131	4.5	50	0.80	1.68
25%	44	139	8	60	1.08	1.76
50%	45	141	9	80	1.20	1.84
75%	47	146	10	80	1.29	2.05
max	49	150	12	125	1.76	2.60

x_{if} is the focus current, mA,
x_{vw} is the welding speed, rpm,
x_{fp} is the distance to the electron optics, mm,
y_{depth} is the weld depth,
y_{width} is the weld width, and
$x \epsilon Q^4$, $y \epsilon Q$, and Q are sets of positive rational numbers.

- The model $f(X)$, which is used to predict the values for each element, where x is the process parameters.

To score the quality of the model $f(X)$, the team used the following regression metrics:

- Mean standard error (MSE),
- Mean absolute error (MAE), and
- Coefficient of determination R^2 (R2).

MAE assessed by n_{samples} is written as:

$$\text{MAE}\left(y, \hat{y}\right) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \left|y_i - \hat{y}_i\right|,$$

where $\hat{y}$ is the predicted value of the *i-th* sample, y_i is the corresponding actual value.

MSE assessed by n_{samples} is written as:

$$\text{MSE}\left(y, \hat{y}\right) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

R^2 represents the fraction of variance (from y) is calculated as:

$$R^2\left(y, \hat{y}\right) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2},$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$,

$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} m_i^2$,

y_i is the corresponding actual value for the total n.

A. SOURCE DATA ANALYSIS

In the initial stage, a set of source data was analyzed; and a correlation matrix was built for the electron beam welding process parameters, which is shown in Figure 1.

Based on the values of the paired correlation coefficients, the following conclusions can be drawn:

- IW and IF have a high level of negative correlation (−0.86),
- IW and FP have a high level of positive correlation (0.84), and
- IF and FP have a very high level of negative correlation (−0.98).

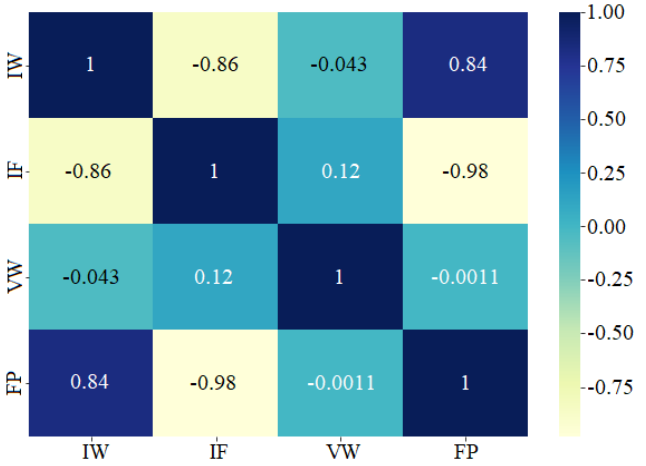


FIGURE 1. Correlation matrix of the EBW parameters.

For the remaining factor pairs, the correlation level is low or very low.

Thus, since some process parameters exhibit a high correlation level, the use of the least squares method (LSM) is complicated by the unstable estimates of the multivariate linear regression coefficients. Accordingly, in its original form, the LSM is poorly applicable for the set problem of building an electron beam welding process model. To improve the causality and reduce the variance in estimates, the LSM modified with *L2* regularization - the *ridge* regression (*ridge*) - was used.

Along with the *ridge*, the efficiency of another popular algorithm - the *random forest regressor* [32] - was studied herein.

B. RIDGE

The *ridge* model was used as implemented in scikit-learn 0.22.2 of Python 3.8.

Once the dataset was analyzed and a correlation matrix was constructed, we obtained the following cross-parametric correlations:

- IW and IF were highly correlated (−0.86),
- IW and FP were highly correlated (0.84), and
- IF and FP were very highly correlated (−0.98).

Thus, the process parameters exhibited high levels of correlation. Therefore, to improve the causality and reduce the variance in estimates, regularization had to be applied, which in this case was L2 regularization.

A simple linear model has the following coefficients:

$$w = \left(w_1, \ldots, w_p\right),$$

where w – coefficients in the linear model, and p – the number of coefficients.

The model can be constructed by minimizing the residual sum of squares between X and y:

$$\min_{w} \|Xw - y\|_2^2,$$

where X – the vector of input values, and y – the output.

Scikit-learn has a linear model with *L2 Ridge* regularization. It addresses the issues of the usual least squares by applying a coefficient penalty. The *ridge* coefficients minimize the residual penalty of the sum of squares:

$$\min_{w} \left(\|Xw - y\|_2^2 + \alpha \|w\|_2^2 \right),$$

where α – a complexity parameter.

The complexity parameter (**alpha**) $\alpha \geq 0$ controls the shrinkage: the greater α is, the greater the shrinkage, making the coefficients more collinearity resistant.

Next, the authors used the *polynomial features* function to create a polynomial regression. This function generates a new matrix of elements that comprises all the polynomial combinations of elements to a **degree** that is equal to or less than the specified degree. By expanding the hypothesis space to all the polynomials where p degree $= p$, the linear model is written as:

$$f(X) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_p x_p + w_{12} x_1^2 + w_{22} x_1 x_2 + \ldots + w_{p2} x_1 x_p + \ldots + w_{pp} x_p^2 + \ldots.$$

If degree $= 2$, the linear model for all the four parameters will be written as:

$$f(X) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_{11} x_1^2 + w_{12} x_1 x_2 + w_{13} x_1 x_3 + w_{14} x_1 x_4 + w_{22} x_2^2 + w_{23} x_2 x_3 + w_{24} x_2 x_4 + w_{33} x_3^2 + w_{34} x_3 x_4 + w_{44} x_4^2.$$

First, we standardized the dataset by applying the *standard scaler* function. Dataset standardization is a common requirement found across many machine learning estimators as the estimators might perform poorly if some parameters are significantly non-Gaussian.

The standard estimate for each variable h from the training set is calculated as:

$$z = \frac{(h - u)}{s},$$

where u is the mean of the training sets, s and is the standard deviation of the training sets.

The *ridge* hyperparameters that need to be fit to find the best solution are the following:

- the polynomial degree; and
- alpha, the strength of regularization.

C. RANDOM FOREST REGRESSOR

The *random forest* is a meta estimator that runs a series of tree classifiers on different subsamples of a dataset and then applies averaging for more accurate prediction and control alignment. The subsample size always matches the initial size of the input set [33].

Random forests reduce the variance of forest estimates. Individual decision trees tend to exhibit high variance and overfit. Injected randomness in forests results in decision trees that have multiple disparate prediction errors. Some errors can be eliminated by taking the mean of the predictions. *Random forests* reduce the variance by combining different trees and sometimes by slightly increasing the bias. In practice, the variance is often reduced substantially, producing a better general model.

Scikit-learn contains the *Random Forest Regression (RFR)* as an RF implementation. *RFR* contains an averaging algorithm based on randomized decision trees Extra-Trees. The algorithm is a perturbation and merger method devised specifically for trees [34].

RFR builds each tree of the ensemble from a subsample taken by bootstrapping from the training set. In addition, when splitting each node while constructing a tree, the best split is determined based on all input parameters.

Bootstrapping is a method that consists of the following. Let there be a set X sized N. Let us evenly retrieve N elements from the set and return them. That means we will retrieve a random element N times from the set assuming that the probability N to retrieve one element is equal for each element. Each time, we retrieve an element from all N existing elements. We denote a new sample as X_1. The process is repeated M times to generate M subsamples $X_1, \ldots, X_M$. Now that we have a sufficient number of subsamples, we can estimate different statistics of the initial distribution.

The algorithm that constructs a *random forest* of N (**n_estimators**) trees is as follows:

For each $n = 1, \ldots, N$, do the following:

1. Generate a sample X_n via bootstrapping.
2. Build a decision tree b_n on the sample X_n:
   a. Select the best feature according to the specified **criterion**, and then split the tree using it. Repeat this step until the subsample is depleted.
   b. Build the tree until each leaf has no more than n_{min} elements (**min_samples_leaf**) or until the tree has reached a certain depth (**max_depth**).
   c. For each split, first select m random features (**max_features**) from n original features. Further optimal subsampling is performed only over the selected features.

The final decision is the mean:

$$a(x) = \frac{1}{N} \sum_{i=1}^{N} b_i(x),$$

where N – number of trees in *random forest* algorithm, b_i – parameters of the polynomial (resolving tree), x – inputs; and $i = \overline{1, N}$.

The *RFR* hyperparameters that need to be fit to find the best solution are the following:

- n_estimators (NE): how many trees are in the forest,
- criterion (CR): the function that measures the quality of tree splitting or branching (MSE or MAE),

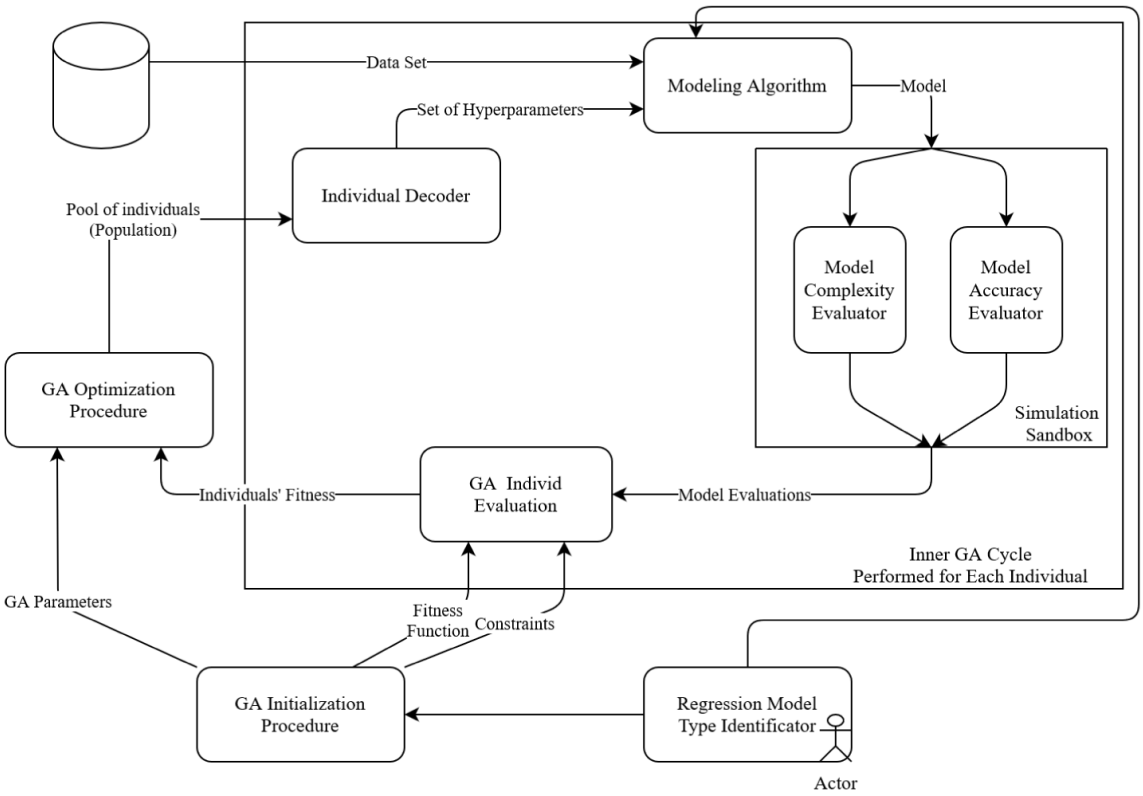


FIGURE 2. Generalized scheme of the procedure for configuring hyperparameters.

- max_depth (MD): maximum tree depth,
- max_features (MF): maximum number of features for splitting,
- min_samples_leaf (MSL): minimum number of elements per leaf, and
- min_samples_split (MSS): minimum number of elements for splitting an inner tree node.

D. OPTIMIZING MODEL HYPERPARAMETERS

To estimate the quality of the $f(X)$ model, the following metrics have been used in the study:

- Mean absolute error, and
- Coefficient of determination R2.

At the EBW process simulation stage, the optimal hyperparameters of the machine learning models have been chosen.

To universalize the approach and strengthen the justification of choosing the hyperparameters of the regression algorithms studied herein, as well as considering the high variability of these hyperparameters, the multiobjective optimization technique applicable for this combinatorial problem - an (evolutionary) genetic algorithm - has been proposed to determine the effective sets of hyperparameters. The genetic algorithm and its variants are widely used to solve complex optimization problems, including a multiobjective statement of problems, which for this case involves considering a pair of criteria - the complexity of the resulting regression model and the estimation of the model's accuracy according to the quality criterion chosen.

The generalized scheme of the procedure for tuning hyperparameters of regression model (Figure 2) involves the execution of procedures of a multicriteria genetic algorithm (GA) over a set of solutions, for which an estimate of the model's quality with given values of hyperparameters is obtained.

In the study, the results of which are described herein, the following estimation techniques were used to calculate the criteria for each set of the regression algorithm hyperparameters to unify the calculations and generalize the approach (it, in fact, allows considering the system as an open one capable of integrating other regression methods).

- To estimate the regression model complexity when implementing the genetic algorithm, a 10-fold calculation procedure was performed using the model built with the current (estimated) set of hyperparameters in 10% of the points available for such an estimation. The described procedure to ensure statistical and computational stability was implemented in special software and hardware 'sandbox' - a computing environment with exclusive computing power designed to calculate the regression model. Equal computing power was provided for all regression analysis methods and the models they generated. The model complexity estimate was the average (machine) time of performing the procedure described.

- The regression model accuracy was estimated using the chosen criterion - the mean error (square or absolute) calculation. To save computational resources for implementing the genetic optimization algorithm and the time to perform the appropriate experiments, the model values, calculated based on the results of the procedure for estimating the first (as described above) criterion, can be used.

The use of a genetic algorithm to determine an effective set of hyperparameters is obviously associated with multifold estimation regression models obtained for each set of hyperparameters of the algorithms considered. However, this approach seems to be justified since even considering the possible limitations introduced for some hyperparameters, a complete enumeration of their combinations, in fact, implies a complete factorial, the number of experiments in which will be enormously large. Moreover, each of these experiments will also require calculating the above-described criteria. The efficiency of the genetic algorithm as a heuristic optimization algorithm, which is expressed, among other things, in its ability to generate solutions when considering relatively small volumes of search domain subspaces, is its well-known advantage.

Considering the multiobjective statement of the problem, the genetic algorithm has generated a set of solutions approximating the Pareto frontier for the problem to be solved. According to the study objectives, a set of effective hyperparameters comprised those that ensured the maximum regression model accuracy (the 2nd criterion) while meeting the criterion limitation on the computation time (the 1st criterion), corresponding to the requirements imposed by the technological system for which the computation was performed.

When searching for hyperparameters of regression models, the following settings of the genetic algorithm is used:

- Method of decisions (individuals in the population) coding: bit string,
- Population size: 50 individuals,
- Maximum number of search steps: 200 steps,
- Selection: tournament with a tournament size of 5 individuals,
- Crossing type: single point,
- Proportion of offspring in the new population (reproduction - crossover fraction): 0.9,
- Mutation type: uniform with rate equal 0.01.

The rest of the optimization procedure settings are used in the ''Use default'' value of the MATLAB Optimtool environment.

Setting the genetic algorithm hyperparameters is beyond the scope of the study, the results of which are described herein, and is the subject of separate research.

III. EXPERIMENTAL OPTIMIZATION OF REGRESSION MODELS

The models were configured and trained separately for y_{depth} and y_{width} over the set of parameters X.

The success of training the model with optimal hyperparameters on the full dataset is measured as **train_score**.

Cv_score represents the estimation of the model prediction accuracy using cross-validation. Cross-validation comprises 5 blocks.

To improve the validation accuracy, the following algorithm is implemented:

1) Set $K = 1$.
2) Set $i = 1$.
3) Randomly shuffle the DS_i dataset.
4) Calculate the S_i estimate for the DS_i dataset using the cross_val_score function from the scikit-learn package.
5) If $i < K$, then set $i = i + 1$ and go to step 3. Otherwise, go to the next step.
6) Calculate the final estimate.

$$S_K^{cv\_score} = \frac{1}{K}\sum_{j=1}^{K} S_j.$$

7) Check the fulfillment of the inequality

$$0.9 \le \frac{S_K^{cv\_score}}{S_{K-1}^{cv\_score}} \le 1.1 :$$

   a. If the inequality is met, then the estimate is $S_K^{cv\_score}$. The computation is finished.
   b. If the inequality is not met, then set $K = K + 1$ and go to step 2.

The computing system that was used to conduct the experiments had the following characteristics:

- Central processing unit: Intel Core i9-9900.
- Memory: 16 GB of DDR4 2133 MHz.
- Video card: Sapphire Radeon RX 580.
- Operating system: Windows 10.

A. FINDING THE HYPERPARAMETERS OF THE RIDGE MODEL

1) RIDGE MODEL FOR Y_{DEPTH}

The hyperparameter degree was selected from the following: 1, 2, 3, 4, 5, and 6. The hyperparameter alpha was selected from the following: 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, and 10.

The optimal hyperparameters were found using *GridSearchCV,* where the MAE was the metric for scoring each test, and five-fold CV was conducted. Table 2 shows top ten results.

Table 2 uses the following notation: mean_test_score is the mean test score.

We then plotted the degree curves at alpha $= 10^{-5}$, as shown in Figure 3. The alpha curves were plotted at degree $= 2$, as shown in Figure 4.

The optimal hyperparameters were the following: degree $= 2$, and alpha $= 10^{-5}$. Table 3 presents the test results.

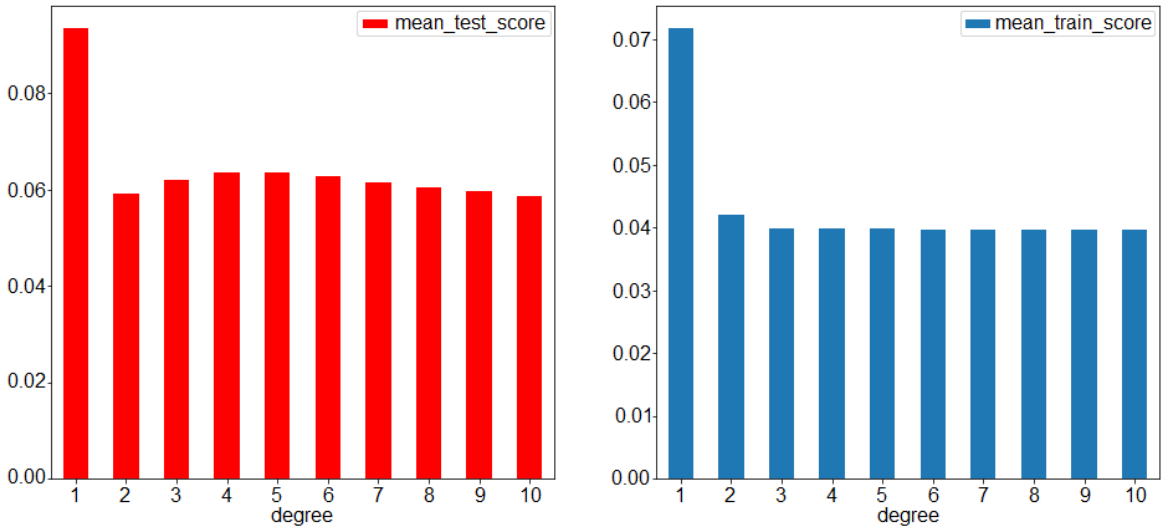


FIGURE 3. Degree curves.

TABLE 2. Best results of optimizing the *ridge* hyperparameters for y_{depth}.

degree	alpha	mean_test_score
2	10^{-5}	0.059237
3	10^{-4}	0.060137
4	10^{-3}	0.060498
2	10^{-4}	0.060570
6	10^{-2}	0.060892
5	10^{-3}	0.061015
6	10^{-3}	0.061220
2	10^{-8}	0.061906
3	10^{-5}	0.061918
2	10^{-6}	0.061995

TABLE 3. *Ridge* model scores for y_{depth}.

	R2	MAE
train_score	0.929407	0.044803
cv_score	0.894361	0.055461

2) RIDGE MODEL FOR Y_{WIDTH}

The hyperparameter degree was selected from the following: 1, 2, 3, 4, 5, and 6. The hyperparameter alpha was selected from the following: 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, and 10. The optimal hyperparameters were found using GridSearchCV, where the MAE was the metric for scoring each test, and five-fold CV was conducted.

Table 4 shows top ten results in descending order.

TABLE 4. Best results of optimizing the *ridge* hyperparameters for y_{width}.

degree	alpha	mean_test_score
5	10^{-4}	0.032053
5	10^{-5}	0.032089
5	10^{-6}	0.032093
5	10^{-7}	0.032094
5	10^{-8}	0.032094
4	10^{-4}	0.032128
4	10^{-5}	0.032176
4	10^{-6}	0.032188
4	10^{-7}	0.032190
3	10^{-4}	0.032835

Table 4 uses the following notation: mean_test_score is the mean test score. The degree curves were plotted at alpha $= 10^{-4}$, as shown Figure 5.

As the Figure shows, starting from degree $= 3$, the score virtually did not change. Thus, degree $= 3$ was used further on so as not to complicate the model.

The alpha curves were plotted at degree $= 3$, as shown in Figure 6.

The optimal hyperparameters were the following: degree $= 3$, and alpha $= 10^{-4}$. Table 5 presents the test results.

B. OPTIMIZING HYPERPARAMETERS FOR THE RANDOM FOREST REGRESSOR (RFR)

1) RFR MODEL FOR Y_{DEPTH}

The hyperparameters were selected from the following ranges:

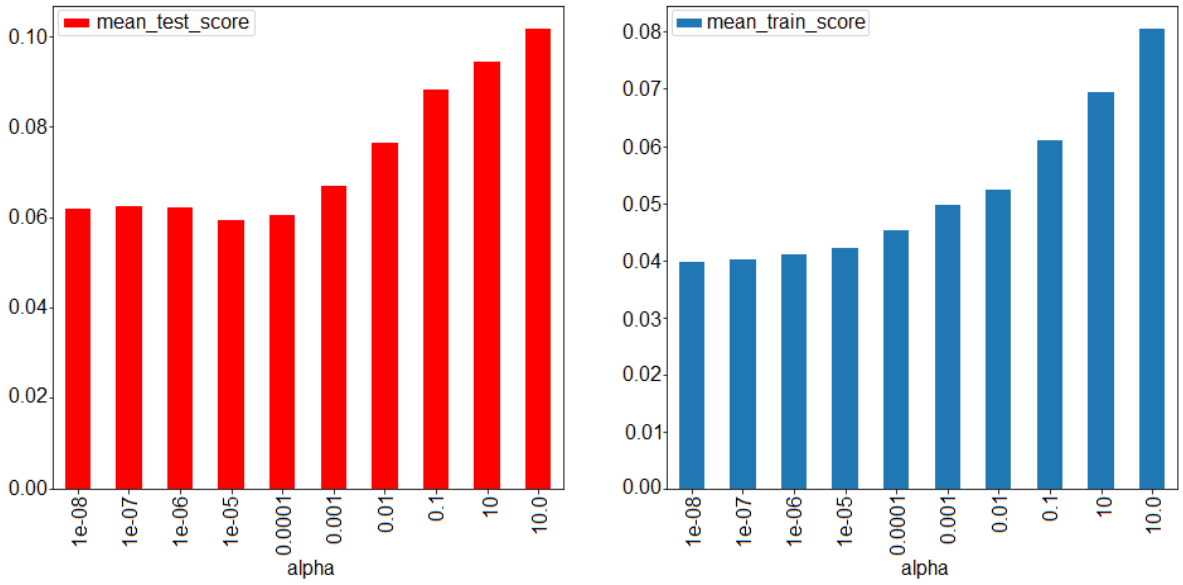


FIGURE 4. Alpha curves.

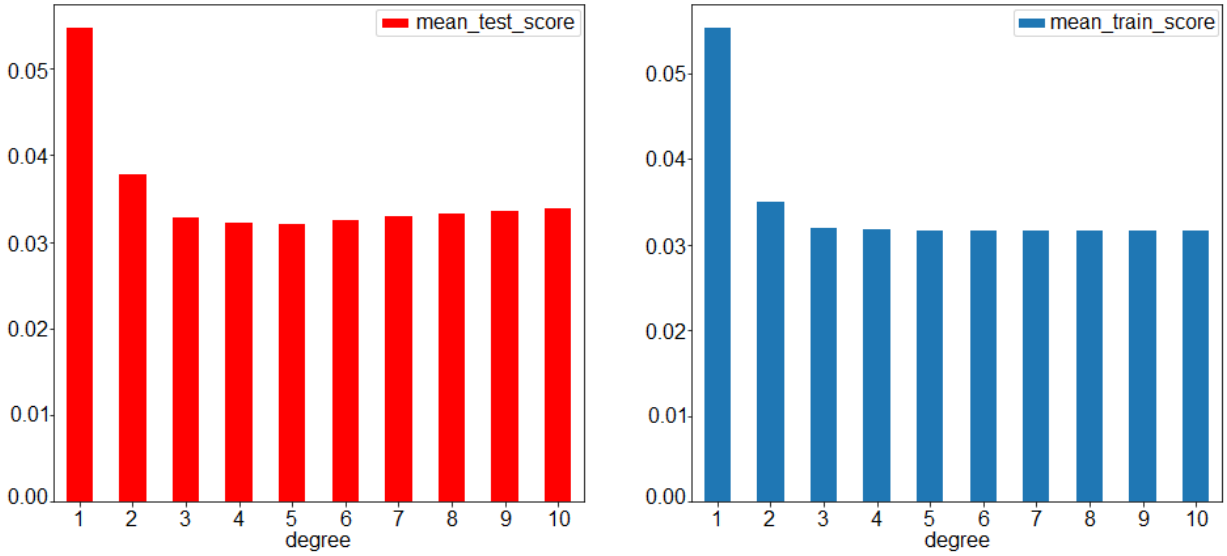


FIGURE 5. Degree curves.

- NE: from 10 to 100 with a step size of 10;
- CR: MSE or MAE;
- MD: 4, 6, 8, 9, 10, 11, or 12;
- MF: from 1 to 4 with a step size of 1;
- MSL: from 1 to 4 with a step size of 1; and
- MSS: from 2 to 5 with step size of 1.

At MD = 0, tree nodes expand to the point that all leaves are clear or until all leaves are less than the MSS.

The optimal hyperparameters were found using GridSearchCV, where the MAE was the metric for scoring each test, and five-fold CV was conducted. Table 6 shows the top ten results in descending order.

Table 6 uses the following notation: mean_test_score is the mean test score.

The NE curves were plotted using (CR = MSE, MD = 8, MF = 3, MSL = 1, and MSS = 2), as shown in Figure 7.

The MD curves were plotted using (NE = 80, CR = MSE, MF = 3, MSL = 1, and MSS = 2), as shown in Figure 8.

The MSS curves were plotted using (NE = 80, CR = MSE, MD = 8, MF = 3, and MSL = 1), as shown in Figure 9.

The optimal hyperparameters were the following: NE = 80, CR = MSE, MD = 8, MF = 3, MSL = 1, and MSS = 2.

The process parameters had the following significance: 4% for x_{iw}, 30% for x_{if}, 47% for x_{vw}, and 19% for x_{fp}.

Table 7 presents the test results.

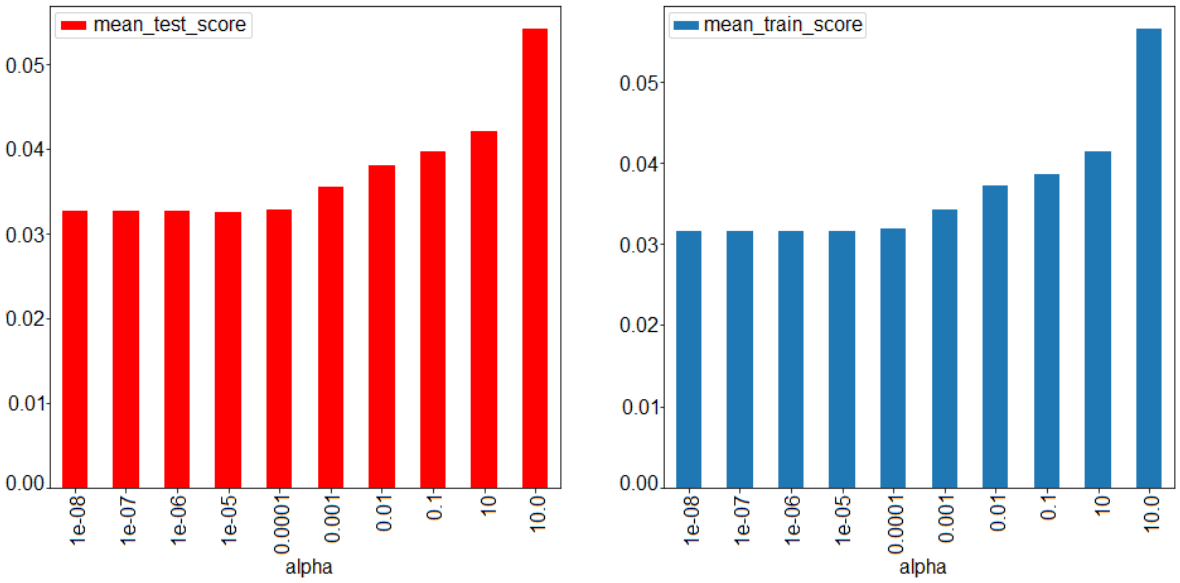


FIGURE 6. Alpha curves.

TABLE 5. *Ridge* model scores for y_{width}.

	R2	MAE
train_score	0.975834	0.030768
cv_score	0.973392	0.037040

TABLE 6. Best results of optimizing the RFR hyperparameters for y_{depth}.

NE	CR	MP	MF	MSL	MSS	mean_test_score
80	MSE	8.0	3	1	2	0.056
80	MSE	10.0	3	1	2	0.056
80	MSE	11.0	3	1	2	0.056
80	MSE	0	3	1	2	0.056
80	MSE	12.0	3	1	2	0.056
60	MSE	8.0	3	1	2	0.057
60	MSE	12.0	3	1	2	0.057
60	MSE	10.0	3	1	2	0.057
60	MSE	0	3	1	2	0.057

2) RFR MODEL FOR Y_{WIDTH}

The hyperparameters were selected from the following ranges:

- NE: from 10 to 100 with a step size of 10;
- CR: MSE or MAE;
- MD: 4, 6, 8, 9, 10, 11, or 12;
- MF: from 1 to 4 with a step size of 1;
- MSL: from 1 to 4 with a step size of 1; and
- MSS: from 2 to 5 with a step size of 1.

At MD = none, the tree nodes expand to the point that all leaves are clear or until all leaves contain less than the MSS.

The optimal hyperparameters were found using *GridSearchCV*, where the MAE was the metric for scoring each test, and five-fold CV was conducted.

Table 8 shows the top ten results in descending order.

TABLE 7. *RFR* model scores for y_{depth}.

	R2	MAE
train_score	0.932585	0.043007
cv_score	0.892730	0.054735

TABLE 8. Best results of optimizing the *RFR* hyperparameters for y_{width}.

NE	CR	MP	MF	MSL	MSS	mean_test_score
60	MAE	6	1	1	2	0.0316
50	MAE	6	1	1	2	0.0318
70	MAE	10	1	1	3	0.032
70	MAE	12	1	1	3	0.032
70	MAE	0	1	1	3	0.032
70	MAE	11	1	1	3	0.0321
70	MAE	8	1	1	3	0.0322
70	MAE	6	1	1	3	0.0322
60	MAE	8	1	1	2	0.0323

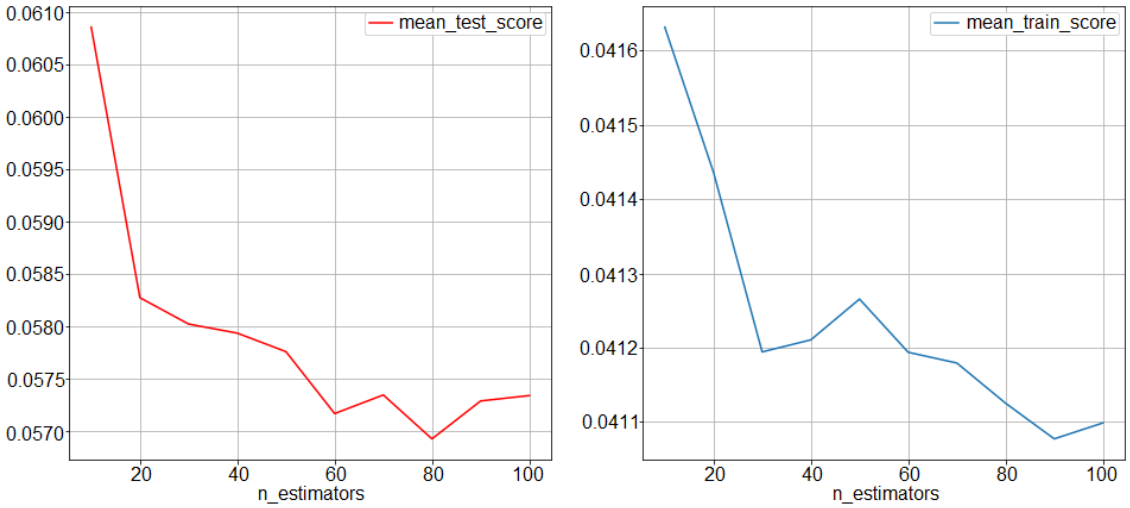


FIGURE 7. Curves of n_estimators.

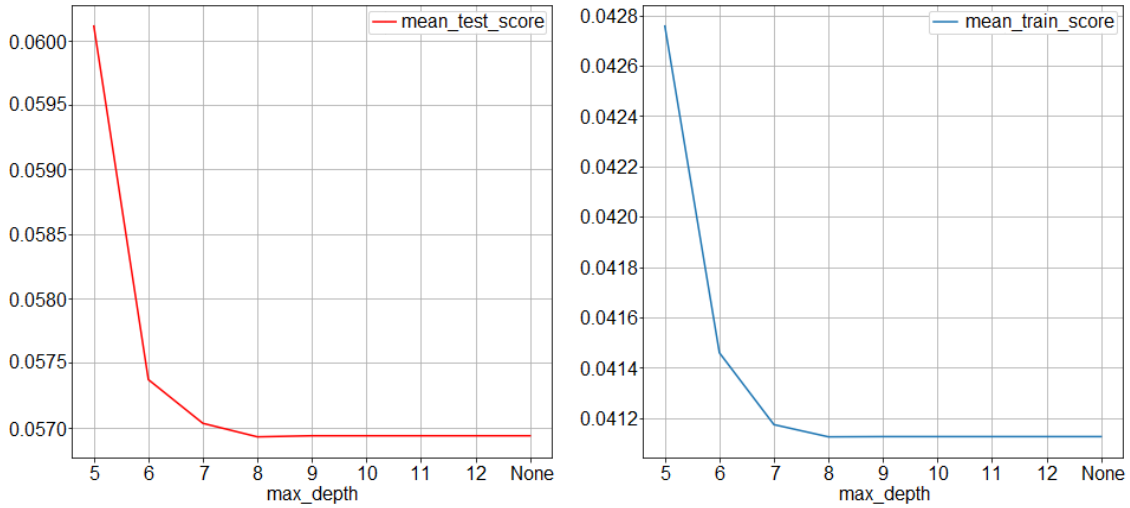


FIGURE 8. Curves of max_depth.

Table 8 uses the following notation: mean_test_score is the mean test score.

The NE curves were plotted using (CR = MAE, MD = 6, MF = 1, MSL = 1, and MSS = 2), as shown in Figure 10.

The MD curves were plotted using (NE = 60, CR = MAE, MF = 1, MSL = 1, and MSS = 2), as shown in Figure 11.

The MSS curves were plotted using (NE = 60, CR = MAE, MD = 6, MF = 1, and MSL = 1), as shown in Figure 12.

The optimal hyperparameters were the following: NE = 60, CR = MAE, MD = 6, MF = 1, MSL = 1, and MSS = 2.

The process parameters had the following significance: 15% for x_{iw}, 28% for x_{if}, 42% for x_{vw}, and 15% for x_{fp}.

Table 9 presents the test results.

TABLE 9. *RFR* model scores for y_{width}.

	R2	MAE
train_score	0.971372	0.033067
cv_score	0.962793	0.044064

C. DISCUSSION OF THE RESULTS

Thus, the research produced the following machine learning models to solve the regression problem:

1. *Ridge*, and
2. *Random forest regressor*.

The experimental research derived the following optimal hyperparameters for the *ridge regression*:

1. for y_{depth}: degree = 2, and alpha = 1e-5; and
2. for y_{width}: degree = 3, and alpha = 1e-4.

The experimental research derived the following optimal hyperparameters for the *RFR*:

1. for y_{depth}: NE = 80, CR = MSE, MD = 8, MF = 3, MSL = 1, and MSS = 2; and

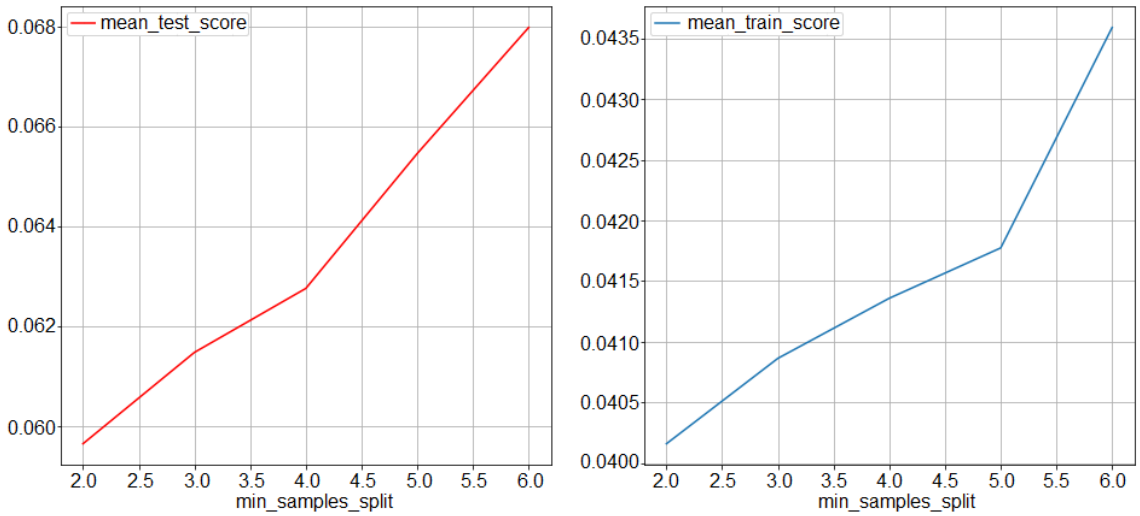


FIGURE 9. Curves of min_samples_split.

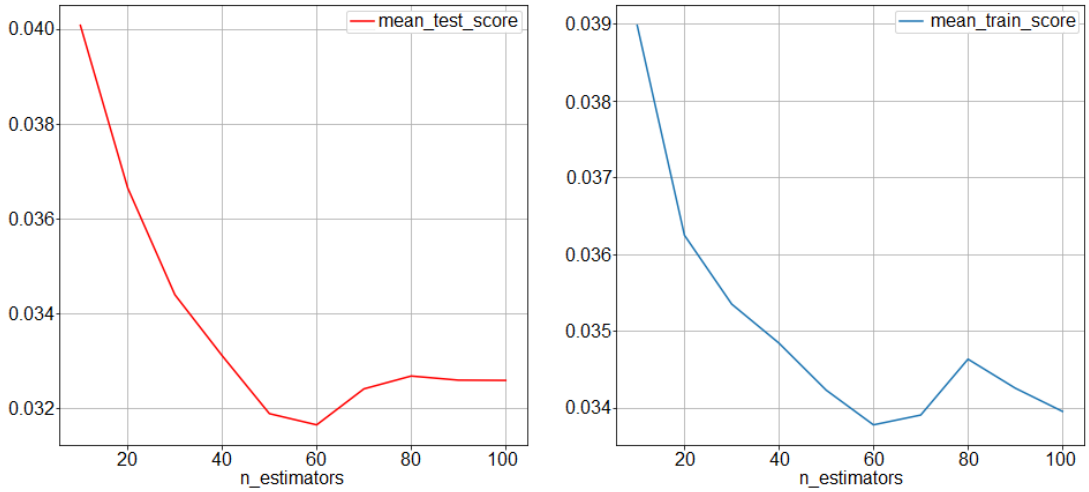


FIGURE 10. Curves of n_estimators.

TABLE 10. Model scores as tested by cv_score.

MODEL		R2	MAE
Ridge	Depth	0.894361	0.055461
	Width	0.973392	0.037040
RFR	Depth	0.892730	0.054735
	Width	0.962793	0.044064

2. for y_{width}: NE = 60, CR = MAE, MD = 6, MF = 1, MSL = 1, and MSS = 2.

Table 10 summarizes the scores of the final models.

As the table shows, the proposed methods have similar scores in terms of the coefficient of determination and the mean absolute error. The table also shows that the proposed method solves the regression problem fairly well.

All models have successfully handled the task, showing forecasting accuracy of at least 89%. With a comparable simulation quality, the *ridge* method is the least resource-intensive method since it requires less computational space. As a result, the *ridge* regression was chosen as mathematical support for a process decision-support tool for electron beam welding [35]–[37].

The experimental results have shown that the model obtained using the *ridge* regression based on the source data given in Section 1.2 to describe the dependence of the weld depth ($f_{depth}(X)$) on the input parameters has a polynomial degree of 2 and a degree of regularization of 10^{-5}:

$$\begin{aligned} f_{depth}(X) = {} & w_0 + w_1 x_{iw} + w_2 x_{if} + w_3 x_{vw} + w_4 x_{fp} \\ & + w_{11} x_{iw}^2 + w_{12} x_{iw} x_{if} + w_{13} x_{iw} x_{vw} + w_{14} x_{iw} x_{fp} \\ & + w_{22} x_{if}^2 + w_{23} x_{if} x_{vw} + w_{24} x_{if} x_{fp} + w_{33} x_{vw}^2 \\ & + w_{34} x_{vw} x_{fp} + w_{44} x_{fp}^2. \end{aligned}$$

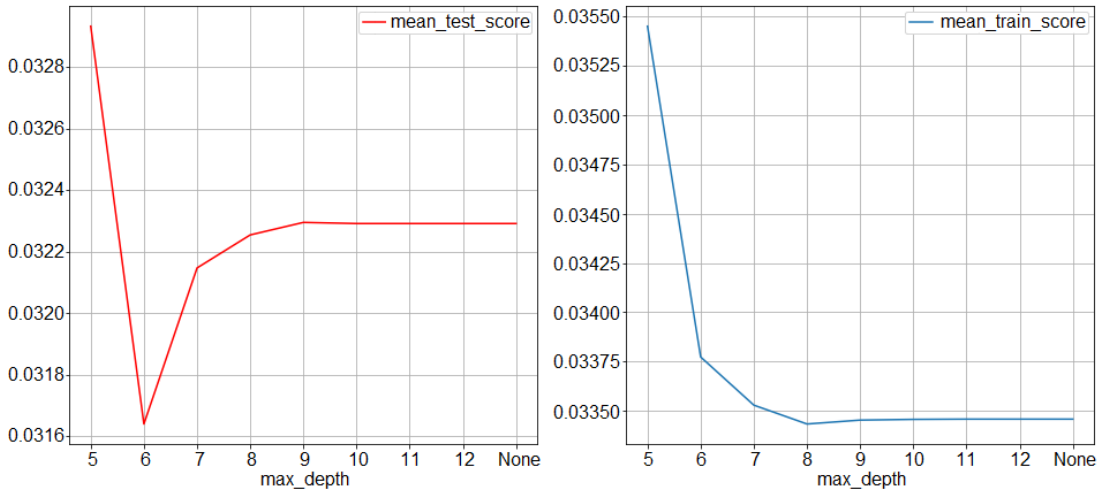


FIGURE 11. Curves of max_depth.

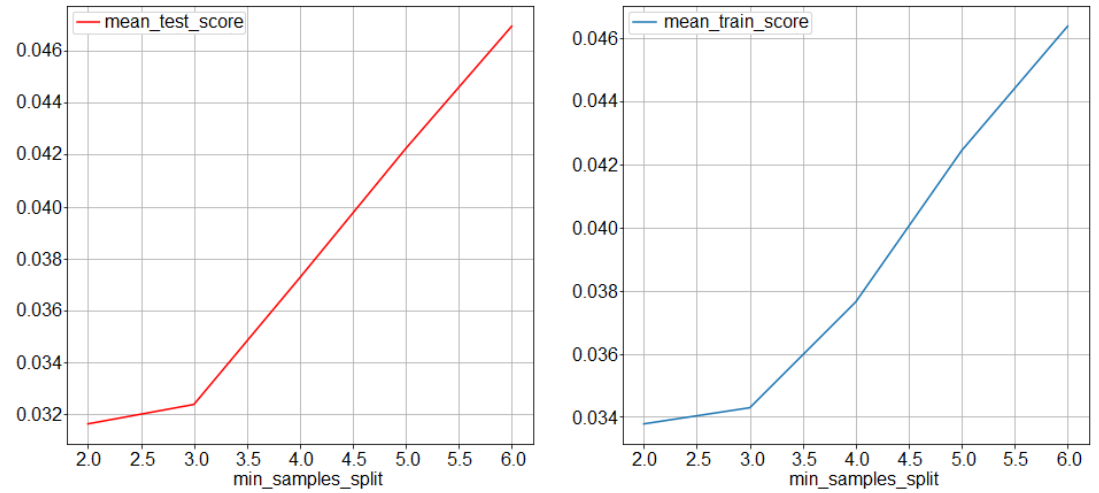


FIGURE 12. Curves of min_samples_split.

The 3rd-degree polynomial and a degree of regularization of 10^{-4} have shown the best results in simulating the weld width ($f_{width}(X)$):

$$
\begin{aligned}
f_{width}(X) &= w_0 + w_1 x_{iw} + w_2 x_{if} + w_3 x_{vw} + w_4 x_{fp} \\
&+ w_{111} x_{iw}^3 + w_{112} x_{iw}^2 x_{if} + w_{113} x_{iw}^2 x_{vw} + w_{114} x_{iw}^2 x_{fp} \\
&+ w_{122} x_{iw} x_{if}^2 + w_{123} x_{iw} x_{if} x_{vw} + w_{124} x_{iw} x_{if} x_{fp} + w_{133} x_{iw} x_{vw}^2 \\
&+ w_{134} x_{iw} x_{vw} x_{fp} + w_{144} x_{iw} x_{fp}^2 + w_{222} x_{if}^3 + w_{223} x_{if}^2 x_{vw} \\
&+ w_{224} x_{if}^2 x_{fp} + w_{233} x_{if} x_{vw}^2 + w_{234} x_{if} x_{vw} x_{fp} + w_{244} x_{if} x_{fp}^2 \\
&+ w_{333} x_{vw}^3 + w_{334} x_{vw}^2 x_{fp} + w_{344} x_{vw} x_{fp}^2 + w_{444} x_{fp}^3.
\end{aligned}
$$

IV. AUTOMATED DECISION SUPPORT SYSTEM FOR ELECTRON BEAM WELDING

A. SYSTEM DESIGN

Based on the experimental study of the efficiency of data analysis methods as a prediction model, in this program, *ridge* was used as it was the fastest model. The application has been written to support decisions in configuring electron beam welding.

The application communicates with the EBW experiment database and can add or remove experiments, train prediction models, plot data, and predict the results of applying different process parameters.

Figure 13 shows the possible scenarios.

The app has a single interface present on its home screen. Figure 14 shows the flowchart.

As shown in the flowchart, the app first checks whether the database needs to be revised. Then, the user revises the database by adding or removing experiments or just adds/removes an experiment straight away, if needed. Then, the app trains the predictive model. The following outputs are rendered in separate fields: quality scoring, data visualization, and EBW parameters (beam current, welding speed, accelerating voltage, focus current, and beam diameter).

These fields are needed for the next step of predicting the weld dimensions.

B. HOW THE AUTOMATED SYSTEM WORKS

The application is an automated workstation for an EBW engineer.

It has the following functions:

- View the experiment table,
- Add an experiment,
- Remove the last experiment,
- Configure and train a predictive model automatically,
- Visualize model-predicted data, and
- Predict the weld dimensions provided the EBW process parameters.

The application has the following requirements:

- Operating system: Windows 7 or newer;
- Python 3.6 or newer; and
- Python packages: Scikit-learn 0.22.2, Pandas 1.0.3, NumPy 1.18.2, and Matplotlib 3.2.1 or newer.

To start the program, run main.py. This will open the home screen. Figure 15 shows the main screen.

Figure 16 shows the Add/remove experiment window.

Click Train Model in the Prediction screen to cause the app to automatically configure and train a model on the full experiment database, and the prediction accuracy will be shown on the screen. Plot Data plots an interactive curve on model-predicted outputs. In Model Prediction, specify the desired EBW parameters and click Predict to see the weld dimensions predicted by the model for the given process parameters.

The main screen shows a summary of the experiments in the Information field. Data Table shows the complete list of all the EBW process experiments. Add Experiment will add or remove the last experiment.

When clicking the Train Model button in the Prediction section, the program automatically establishes and trains the model based on the overall experimental database available as of the moment the button is clicked, and the prediction accuracy is displayed.

The Data Plot section allows visualizing the model-predicted data in the form of an interactive plot (Figure 17).

The values of the parameter chosen in the form (Figure 5, Data Plot section) are plotted on the abscissa. The weld depth/width is plotted in millimeters on the ordinate. Other process parameters can be changed, and the changes in the weld dimensions (depth and width) can be observed on a real-time basis. When the model is generated, the technologist can determine the process parameters ensuring the required weld dimensions. E.g., at a weld depth prescribed by the technology, choose an electron beam welding mode at which the minimum weld width will be achieved, or vice versa.

In the Model Prediction section, the EBW process parameters of interest should be specified, after clicking the Predict button, the program shows the trained model-predicted weld dimensions for the parameters specified.

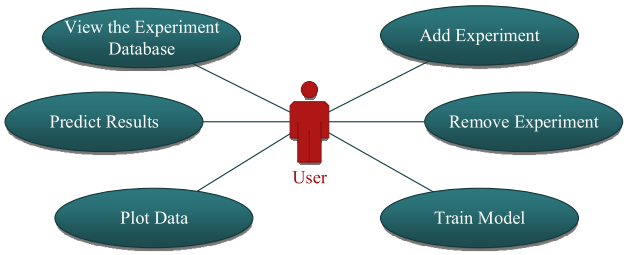


FIGURE 13. Diagram of the process decision-support tool use cases.

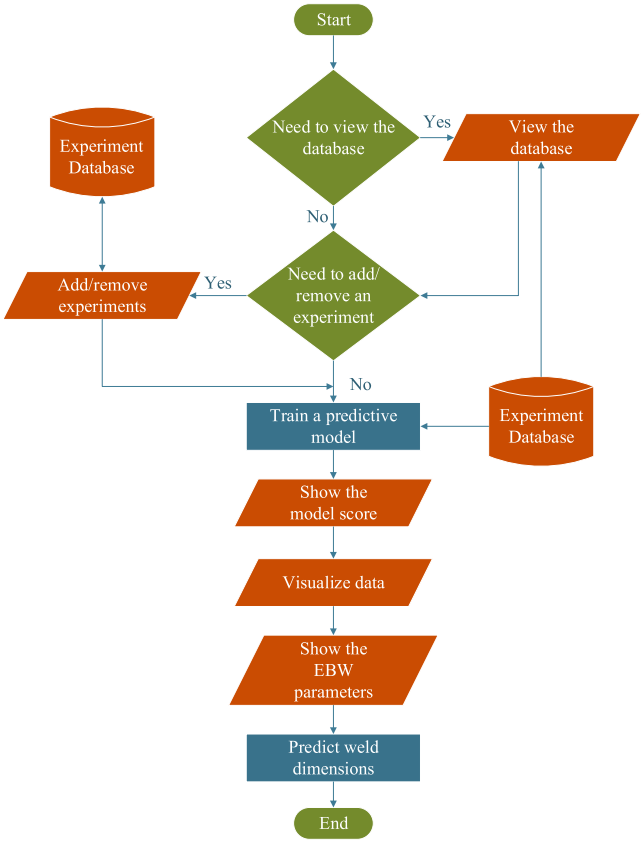


FIGURE 14. Flowchart of the application.

V. EXPERIMENTAL STUDY

To evaluate the efficiency of the approach proposed for solving the problem of predicting the weld properties in the normal operating mode, a study was performed based on the previously obtained sets of 10 process parameters that did not participate in building the models.

The model and experimental data comparison results are given in Table 11.

Computation of the determination coefficients gave the below values:

- the weld depth - 0.89, and
- the weld width - 0.91.

TABLE 11. Comparison of the model and experimental data.

Parameter	Model	Specimen 1	Specimen 2	Specimen 3	Specimen 4	Mean Square Simulation Error	Absolute Simulation Error
Weld Depth, mm	1.74	1.68	1.78	1.84	1.80	0.068	0.065
Weld Width, mm	2.48	2.40	2.44	2.51	2.52	0.051	0.048

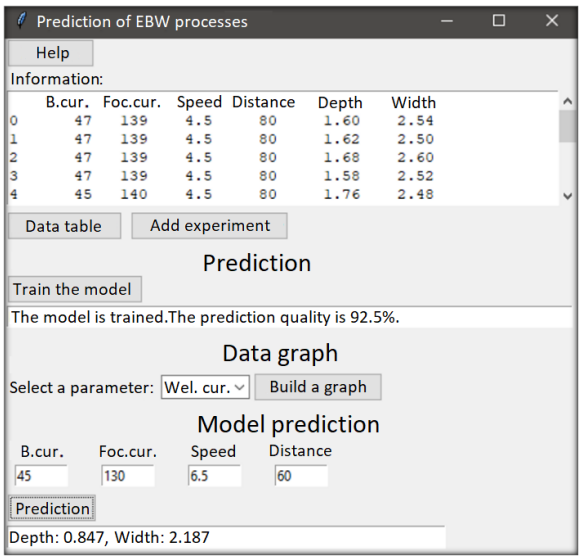


FIGURE 15. Main screen.

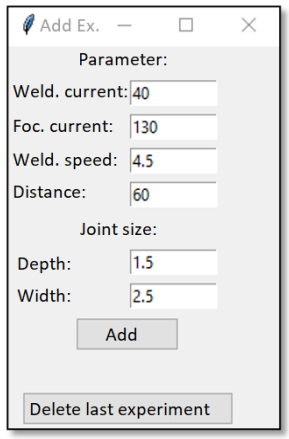


FIGURE 16. Add/remove experiment window.

Accordingly, we can conclude whether the model is good quality and on the possibility of its practical use to solve the problem of predicting the electron beam welding modes with parameters other than those used when training the model.

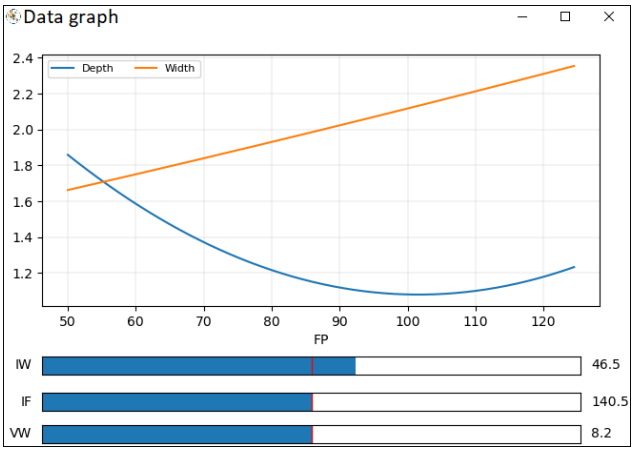


FIGURE 17. Data plot window.

Using the approach proposed, the process parameters were chosen for electron beam welding of a product made of titanium alloy VT-14 with a thickness of 1.2 mm, which differed from those represented in the training dataset: welding current - 45 mA, welding speed - 10 rpm, focus current - 141 mA, and sample surface to electron optics distance - 80 mm.

The experiments were performed on the following equipment:

- electron beam gun,
- 60, 30 kV accelerating voltage source,
- controlled high-precision positioning arm drive,
- vacuum chamber with a negative pressure system,
- MT-Turbo 65D/0/8 KF40M MTM turbomolecular pumping system,
- differential air pumping system, and
- ISO63 solenoid vacuum valve.

As a metallographic examination result, weld joint microslices have been obtained for four specimens, which are shown in Figure 18.

The simulated and experimentally obtained weld depth and width are given in Table 12.

The experimental results allow us to conclude that the model built using the approach proposed herein provides high reliability in solving the problem of predicting the electron beam welding process parameters.

TABLE 12. Experiment and simulation results.

Welding Mode				Weld Dimensions, mm				Error			
				Model		Experiment		Absolute, mm		Relative, %	
IW, mA	IF, mA	VW, rpm	FP, mm	Depth	Width	Depth	Width	Depth	Width	Depth	Width
47	139	4.5	80	1.58	2.5	1.68	2.6	0.1	0.1	6.33	4
45	140	4.5	80	1.76	2.48	1.64	2.56	0.12	0.08	6.82	3.23
45	140	8	80	1.16	1.96	1.2	2.0	0.04	0.04	3.45	2.04
46	141	10	80	1.2	1.78	1.09	1.86	0.11	0.08	9.17	4.49
47	141	12	80	1.08	1.68	1.16	1.74	0.08	0.06	7.41	3.57
48	131	10	125	0.8	2.16	0.88	2.0	0.08	0.16	10	7.41
46	146	10	60	1.36	1.76	1.24	1.8	0.12	0.04	8.82	2.27
43	150	9	50	1.08	1.82	1.08	1.8	0	0.02	0	1.10
44	146	9	60	1.2	1.76	1.2	1.88	0	0.12	0	6.82
45	146	9	60	1.36	1.76	1.28	1.76	0.08	0	5.88	0
Average Error				0.073		0.07		5.79		3.49	

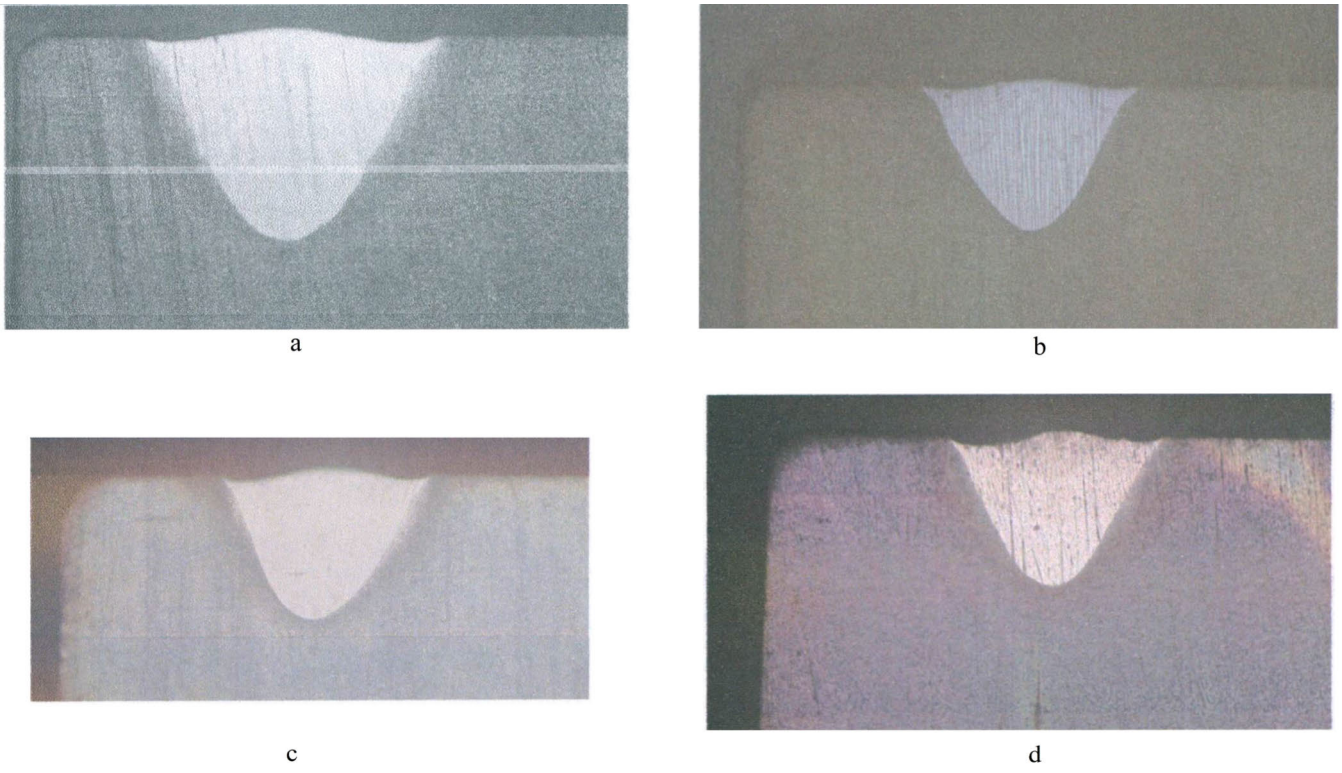


FIGURE 18. Metallographic examination results: a - specimen 1, b - specimen 2, c - specimen 3, and d - specimen 4.

VI. CONCLUSION

The research team developed and trained machine learning models to predict the process parameters of electron beam welding. *Ridge* and *RFR* models were used to predict these parameters.

The experimental research derived the following optimal hyperparameters for the *ridge* regression:

- for y_{depth}: degree = 2, and alpha = 1e-5; and
- for y_{width}: degree = 3, and alpha = 1e-4.

The experimental research derived the following optimal hyperparameters for *RFR*:

- for y_{depth}: NE = 80, CR = MSE, MD = 8, MF = 3, MSL = 1, and MSS = 2.
- for y_{width}: NE = 60, CR = MAE, MD = 6, MF = 1, MSL = 1, and MSS = 2.

The methods solve the regression problem fairly well. The *ridge* regression is far less computationally complex than the *RFR* while the prediction accuracy does not differ significantly. Therefore, the former is a more suitable solution for predicting the EBW process parameters.

Python 3.6 and its Scikit-learn 0.22.2, Pandas 1.0.3, NumPy 1.18.2, and Matplotlib 3.2.1 packages were then used

to implement an EBW decision support system to help engineers better configure the parameters when implementing a new process configuration or improving the quality under an existing one.

The main aim of this study is to show to specialists from the aerospace industry that the implementation of state-of-the-art models could help them to highly increase the efficiency of technological process parameter searches.

The future directions of this study are the implementation of other highly efficient methods for establishing regression models and the development of embedded systems to support decision-making of aerospace industry technologists. For example, with an increase of the experimental data volume, it is possible to use such a bio-inspired method for constructing regression models as artificial neural networks.

REFERENCES

[1] Y. G. Akhlaghi, X. Ma, X. Zhao, S. Shittu, and J. Li, "A statistical model for dew point air cooler based on the multiple polynomial regression approach," *Energy*, vol. 181, pp. 868–881, Aug. 2019.

[2] S. Qiu, S. Li, F. Wang, Y. Wen, Z. Li, Z. Li, and J. Guo, "An energy exchange efficiency prediction approach based on multivariate polynomial regression for membrane-based air-to-air energy recovery ventilator core," *Building Environ.*, vol. 149, pp. 490–500, Feb. 2019.

[3] S. Buitendag, J. Beirlant, and T. de Wet, "Ridge regression estimators for the extreme value index," *Extremes*, vol. 22, no. 2, pp. 271–292, Oct. 2018.

[4] R. Tandon, S. Si, P. Ravikumar, and I. Dhillon, "Kernel ridge regression via partitioning," 2016, *arXiv:1608.01976*. [Online]. Available: http://arxiv.org/abs/1608.01976

[5] A. K. M. E. Saleh, M. Arashi, and B. M. G. Kibria, *Theory of Ridge Regression Estimation With Applications*, vol. 285, 1th ed. Hoboken, NJ, USA: Wiley, 2019, pp. 109–303.

[6] A. Rokem and K. Kay, "Fractional ridge regression: A fast, interpretable reparameterization of ridge regression," 2020, *arXiv:2005.03220*. [Online]. Available: http://arxiv.org/abs/2005.03220

[7] S. Wang, B. Ji, J. Zhao, W. Liu, and T. Xu, "Predicting ship fuel consumption based on LASSO regression," *Transp. Res. D, Transp. Environ.*, vol. 65, pp. 817–824, Dec. 2018.

[8] X. Yang and W. Wen, "Ridge and lasso regression models for cross-version defect prediction," *IEEE Trans. Rel.*, vol. 67, no. 3, pp. 885–896, Sep. 2018, doi: 10.1109/TR.2018.2847353.

[9] Y. Li, C. Zou, M. Berecibar, E. Nanini-Maury, J. C.-W. Chan, P. van den Bossche, J. Van Mierlo, and N. Omar, "Random forest regression for online capacity estimation of lithium-ion batteries," *Appl. Energy*, vol. 232, pp. 197–210, Dec. 2018.

[10] W. Chen, X. Xie, J. Wang, B. Pradhan, H. Hong, D. T. Bui, Z. Duan, and J. Ma, "A comparative study of logistic model tree, random forest, and classification and regression tree models for spatial prediction of landslide susceptibility," *Catena*, vol. 151, pp. 147–160, Apr. 2017.

[11] J. W. Coulston, C. E. Blinn, V. A. Thomas, and R. H. Wynne, "Approximating prediction uncertainty for random forest regression models," *Photogramm. Eng. Remote Sens.*, vol. 82, no. 3, pp. 189–197, Mar. 2016, doi: 10.14358/PERS.82.3.189.

[12] J. Naik, R. Bisoi, and P. K. Dash, "Prediction interval forecasting of wind speed and wind power using modes decomposition based low rank multi-kernel ridge regression," *Renew. Energy*, vol. 129, pp. 357–383, Dec. 2018.

[13] M. A. van de Wiel, T. G. Lien, W. Verlaat, W. N. van Wieringen, and S. M. Wilting, "Better prediction by use of co-data: Adaptive group-regularized ridge regression," *Statist. Med.*, vol. 35, no. 3, pp. 368–381, Sep. 2015, doi: 10.1002/sim.6732.

[14] M. Maalouf, N. Khoury, D. Homouz, and K. Polychronopoulou, "Accurate prediction of gas compressibility factor using kernel ridge regression," in *Proc. 4th Int. Conf. Adv. Comput. Tools Eng. Appl. (ACTEA)*, Jul. 2019, pp. 1–4, doi: 10.1109/ACTEA.2019.8851106.

[15] J. Z. Musoro, A. H. Zwinderman, M. A. Puhan, G. ter Riet, and R. B. Geskus, "Validation of prediction models based on lasso regression with multiply imputed data," *BMC Med. Res. Methodol.*, vol. 14, no. 1, pp. 1–13, Dec. 2014, doi: 10.1186/1471-2288-14-116.

[16] S. Shrivastava, P. M. Jeyanthi, and S. Singh, "Failure prediction of Indian banks using SMOTE, lasso regression, bagging and boosting," *Cogent Econ. Finance*, vol. 8, no. 1, Feb. 2020, Art. no. 1729569, doi: 10.1080/23322039.2020.1729569.

[17] A. C. Kidd, M. McGettrick, S. Tsim, D. L. Halligan, M. Bylesjo, and K. G. Blyth, "Survival prediction in mesothelioma using a scalable lasso regression model: Instructions for use and initial performance using clinical predictors," *BMJ Open Respiratory Res.*, vol. 5, no. 1, Jan. 2018, Art. no. e000240, doi: 10.1136/bmjresp-2017-000240.

[18] T. Shaikhina, D. Lowe, S. Daga, D. Briggs, R. Higgins, and N. Khovanova, "Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation," *Biomed. Signal Process. Control*, vol. 52, pp. 456–462, Jul. 2019.

[19] Z. Wang, Y. Wang, R. Zeng, R. S. Srinivasan, and S. Ahrentzen, "Random forest based hourly building energy prediction," *Energy Buildings*, vol. 171, pp. 11–25, Jul. 2018.

[20] Y. Everingham, J. Sexton, D. Skocaj, and G. Inman-Bamber, "Accurate prediction of sugarcane yield using a random forest algorithm," *Agronomy Sustain. Develop.*, vol. 36, no. 2, Jun. 2016, doi: 10.1007/s13593-016-0364-z.

[21] Y. Liu and H. Wu, "Prediction of road traffic congestion based on random forest," in *Proc. 10th Int. Symp. Comput. Intell. Design (ISCID)*, vol. 2, Dec. 2017, pp. 361–364, doi: 10.1109/ISCID.2017.216.

[22] G. Van Rossum. (2007). *Python Programming Language*. [Online]. Available: https://thereaderwiki.com/en/Python_(programming_language)

[23] M. Lutz, *Learning Python: Powerful Object-Oriented Programming*. Newton, MA, USA: O'Reilly Media, 2013.

[24] M. Guzdial and B. Ericson, *Introduction to Computing and Programming in Python*. London, U.K.: Pearson, 2016.

[25] S. Raschka and V. Mirjalili, *Python Machine Learning*. Birmingham, U.K.: Packt Publishing, 2017.

[26] G. Hackeling, *Mastering Machine Learning With Scikit-Learn*. Birmingham, U.K.: Packt Publishing, 2017.

[27] A. Géron, *Hands-on Machine Learning With Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Newton, MA, USA: O'Reilly Media, 2019.

[28] K. Jolly, *Machine Learning With Scikit-Learn Quick Start Guide: Classification, Regression, and Clustering Techniques in Python*. Birmingham, U.K.: Packt Publishing, 2018.

[29] D. Paper and D. Paper, "Scikit-learn regression tuning," in *Hands-on Scikit-Learn for Machine Learning Applications: Data Science Fundamentals With Python*. Berkeley, CA, USA: Apress, Nov. 2019, pp. 189–213, doi: 10.1007/978-1-4842-5373-1_7.

[30] K. Polimis, A. Rokem, and B. Hazelton, "Confidence intervals for random forests in Python," *J. Open Source Softw.*, vol. 2, no. 19, p. 124, Nov. 2017, doi: 10.21105/joss.00124.

[31] J. Xia, H. Chen, Q. Li, M. Zhou, L. Chen, Z. Cai, Y. Fang, and H. Zhou, "Ultrasound-based differentiation of malignant and benign thyroid nodules: An extreme learning machine approach," *Comput. Methods Programs Biomed.*, vol. 147, pp. 37–49, Aug. 2017, doi: 10.1016/j.cmpb.2017.06.005.

[32] M. Wang, H. Chen, B. Yang, X. Zhao, L. Hu, Z. Cai, H. Huang, and C. Tong, "Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses," *Neurocomputing*, vol. 267, pp. 69–84, Dec. 2017, doi: 10.1016/j.neucom.2017.04.060.

[33] Y. Zhang, R. Liu, A. A. Heidari, X. Wang, Y. Chen, M. Wang, and H. Chen, "Towards augmented kernel extreme learning models for bankruptcy prediction: Algorithmic behavior and comprehensive analysis," *Neurocomputing*, vol. 430, pp. 185–212, Mar. 2021, doi: 10.1016/j.neucom.2020.10.038.

[34] X. Zhao, X. Zhang, Z. Cai, X. Tian, X. Wang, Y. Huang, H. Chen, and L. Hu, "Chaos enhanced grey wolf optimization wrapped ELM for diagnosis of paraquat-poisoned patients," *Comput. Biol. Chem.*, vol. 78, pp. 481–490, Feb. 2019, doi: 10.1016/j.compbiolchem.2018.11.017.

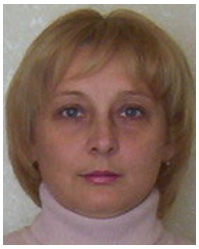
[35] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1, no. 2. Cambridge, MA, USA: MIT Press, 2016.

[36] A. Cutler, D. R. Cutler, and J. R. Stevens, "Random forests," in *Ensemble Machine Learning*. Boston, MA, USA: Springer, Jan. 2012, pp. 157–175, doi: 10.1007/978-1-4419-9326-7_5.

[37] X. Sun and H. Zhou, "An empirical comparison of two boosting algorithms on real data sets based on analysis of scientific materials," in *Advances in Computer Science, Intelligent System and Environment*, vol. 105. Berlin, Germany: Springer, 2011, pp. 327–331, doi: 10.1007/978-3-642-23756-0_53.

VADIM S. TYNCHENKO (Senior Member, IEEE) was born in Kyiv, in 1985. He received the B.S. and M.S. degrees in systems analysis and operations research, in 2006 and 2008, respectively, and the Ph.D. degree in engineering from the Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia, in 2008. He is currently an Associate Professor with the Information and Control Systems Department, Reshetnev Siberian State University of Science and Technology. He also works as an Associate Professor with the Department of Technological Machinery and Equipment for the Oil and Natural Gas Industry of Siberian Federal University, Krasnoyarsk. He is the author of more than 200 scientific articles and more than 20 inventions. His research interests include intelligent control systems, reliability improvement, and the management of technical objects and processes.

SERGEI O. KURASHKIN (Graduate Student Member, IEEE) was born in Krasnoyarsk, Russia, in 1995. He received the B.S. and M.S. degrees in computer science and computer engineering from the Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree. He is also a Senior Laboratory Assistant and a Lecturer Assistant with the Information and Control Systems Department, Reshetnev Siberian State University of Science and Technology. He is the author of 23 scientific articles. His research interests include electron beam welding and the automation of technological processes.



VALERIA V. TYNCHENKO was born in Melitopol, in 1963. She received the degree in engineer-mathematician in the direction of applied mathematics with Kyiv Polytechnic University, Kyiv, Ukraine, and the Ph.D. degree in engineering from the Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia, in 2008. She currently works as an Associate Professor with the Department of Informatics and Computer Engineering, Reshetnev Siberian State University of Science and Technology. She works as an Associate Professor with the Computer Science Department, Siberian Federal University, Krasnoyarsk. She is the author of more than 50 scientific articles. Her research interests include intelligent data processing algorithms and effective parallel algorithms.

VLADIMIR V. BUKHTOYAROV was born in Zelenogorsk, in 1986. He received the B.S. and M.S. degrees in systems analysis and operations research, in 2008 and 2010, respectively, and the Ph.D. degree in engineering from the Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia, in 2010. He is currently an Associate Professor with the Information Technology Security Department, Reshetnev Siberian State University of Science and Technology. He also works as an Associate Professor with the Department of Technological Machinery and Equipment for Oil and Natural Gas Industry of Siberian Federal University, Krasnoyarsk. He is the author of more than 150 scientific articles and more than 15 inventions. His research interests include computational intelligence, simulation techniques for machinery design, machinery dependability analysis, and reliability improvement.

VLADISLAV V. KUKARTSEV was born in Nazarovo, in 1980. He received the B.S. and M.S. degrees in systems analysis and operations research, in 2001 and 2003, respectively, and the Ph.D. degree in engineering from the Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia, in 2009. He is currently an Associate Professor with the Information Economic Systems Department, Reshetnev Siberian State University of Science and Technology. He also works as an Associate Professor with the Computer Science Department, Siberian Federal University, Krasnoyarsk. He is the author of more than 100 scientific articles and 15 inventions. His research interests include intellectual management systems, the reproduction of basic production assets, and the management of technical and socioeconomic systems.

ROMAN B. SERGIENKO was born in Krasnoyarsk, Russia, in 1987. He received the B.S. and M.S. degrees in systems analysis and operations research, in 2008 and 2010, respectively, and the Ph.D. degree in engineering from the Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, in 2010. He is currently a Machine Learning Engineer for NLP with Gini Gmbh, Munich, Germany. He is the author of more than 20 scientific articles and more than ten inventions. His research interests include machine learning and natural language processing.

SERGEI V. TYNCHENKO was born in Kyiv, Ukraine, in 1959. He received the degree in electronics engineering in the direction of automated control systems from the Kyiv Higher Radio Engineering School, Kyiv, and the Ph.D. degree in engineering from the Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia, in 2000. He currently works as an Associate Professor with the Department of Informatics and Computer Engineering, Reshetnev Siberian State University of Science and Technology, Krasnoyarsk. He works as an Associate Professor with the Department of Digital Control Technologies, Siberian Federal University, Krasnoyarsk. He is the author of more than 50 scientific articles. His research interests include automated control systems, radio electronics, and multiprocessor computing systems.

KIRILL A. BASHMUR was born in Krasnoyarsk, Russia, in 1989. He received the M.S. degree in oil and gas engineering from the Siberian Federal University, Krasnoyarsk, in 2012, where he is currently pursuing the Ph.D. degree. He is currently a Senior Lecturer with the Department of Technological Machinery and Equipment for Oil and Natural Gas Industry, Siberian Federal University. He is the author of three books, more than 100 articles, and more than ten inventions. His research interests include simulation techniques in machinery design, analysis of machinery dependability, and reliability improvement.