Metals and Materials International (2021) 27:298–305
https://doi.org/10.1007/s12540-020-00713-w

Exploration of Machine Learning to Predict Hot Ductility of Cast Steel from Chemical Composition and Thermal Conditions

Daegeun Hong[1] · Sanghum Kwon[2] · Changhee Yim[1]

Received: 11 February 2020 / Accepted: 23 March 2020 / Published online: 3 June 2020

Abstract
This study explores the use of machine learning (ML) as a data-driven approach to estimate hot ductility of cast steel from literature data. Four ML algorithms were used to predict hot ductility by considering elemental composition and thermal conditions. Experimentally-measured reduction of area (RA) values were converted to a low-temperature limit, center-temperature, and high-temperature limit, which were represented as Gaussian curves. The prediction accuracy of the four ML models was evaluated using RMSE for these three output variables. In a case study of three steels that had different contents of alloying elements, only the Neural-net model predicted the RA trough more accurately in all cases. These results demonstrate the utility of ML models to predict hot ductility of cast steels.

Keywords Prediction model · Data-driven inverse model · Random forest · Gaussian process · Support vector machine · Neural network · Gaussian fitting

1 Introduction

It is difficult to measure hot ductility considering various chemical composition ranges and continuous casting conditions of all commercially produced steels [1–3], so a more efficient hot ductility prediction method is required. As an alternative to existing physics-based and simple regression method, data-driven approach is significant interest to steel fields [4–10] by providing correlations among chemical composition, operation conditions, and mechanical properties. It helps to find out the properties of the material without performing any experiments. To analyze hot ductility, traditional forward simulation models [3, 11–13] including dynamic simulation models and state-space models simulate the physical behavior of steel by equations based on heat transfer/material transfer. These forward models require a lot of input variables and expertise to build a model. Uncertainty is also accompanied by assumptions and simplifications in the model construction.

On the other hand, data-based inverse models have been tried to analyze the existing ductility behavior [14, 15]. The inverse model implements a model using the observed input and output data. Data-driven inverse models have received a lot of attention in the field of model predictive control [16] because they are easy to build. The advantage of this approach is that the model can be flexibly constructed only by the input and output variables selected by the simulation performer. The inverse model is largely divided into a regression model and a machine learning (ML) models [16]. In particular, the ML method to search for the optimal model based on the learning algorithm is used like pattern recognition and data mining in various fields including steel and material, and can approximate nonlinear behavior.

Machine learning (ML) models use learning algorithms like pattern recognition to identify the relationship between the dependent and independent variables [16, 17]. ML models can describe systems that are have strong nonlinearities [18] and can extract knowledge from large datasets [19, 20]. Supervised learning is a method of ML in which the algorithm is provided input data (e.g., casting conditions), and a clear statement of the output (e.g., 'cracked' or not), so it has a clear objective when it derives the model and predictions future response. The

✉ Changhee Yim
chyim@postech.ac.kr

1 Graduate Institute of Ferrous Technology, Pohang University of Science and Technology, Pohang 37673, Republic of Korea

2 Steel Making Research Group, POSCO Research Institute, Pohang 37859, Republic of Korea

properties of a steel depend on its chemical composition and the manner in which it is processed. These parameters are numerous and complex.

Various ML models have been used in attempts to predict the mechanical properties from these parameters. Examples include a random forest model (RF) [4] to predict the physical features of reduced activation ferritic/martensitic (RAFM) steels [21], a Gaussian Process (GP) [5] meta-model to predict the tensile yield strength of Ferrium® PH48S [5], a Support Vector Machine (SVM) [6, 7] to predict the tensile properties and yield strength of RAFM steels [6], and a Neural Network (NN) [8–10] to model the effects of elemental composition and thermal conditions concerning tensile strength of low carbon steels, and to assess the strength and particle size of hot strip mill products [8].

However, few studies have been conducted to predict reduction of area (RA). Examples include regression model [22, 23] and a back-propagation NN [14, 15]. These methods have the limitation that the prediction and judgment were made using one algorithm under a narrow range of conditions. It is difficult to generalize that the proposed methods are the best or optimal. Therefore, this study builds four ML models (RF, GP, SVR, NN) that analyze literature data related to hot ductility, and compares their prediction accuracy, then explains the characteristics and accuracy of each model and selects the optimal ML model for hot ductility analysis.

2 Materials and Methods

2.1 Dataset

Data sets on hot ductility had been collected using a web-based academic database. The terms such as hot ductility and high temperature ductility had been employed to find out from online scientific literature databases like ScienceDirect, Scopus, and Springer. Hits were pruned to leave only data from steel production. Then data that were incomplete, were duplicates, or were of unknown origin were deleted. This process left 4050 measurements of ductility according to temperature with steel composition and experimental variables. A total of 4420 experimental data were collected by adding 370 data obtained with the tensile test. The hot ductility-prediction models, the dependent variables are the ductility values according to temperature at $600 \leq T \leq 1200$ °C; independent parameters such as 16 compositions and five process conditions (Table 1).

2.2 Data Preprocessing

The ranges of variables were standardized using the MinMaxScaler [24]; this process avoids parameter distortion that may occur in modeling due to different scales of variables. The MinMaxScaler changes the data so that all the attributes range between 0 and 1. In an n-dimensional dataset, all data are contained within a hyper-rectangle on a Cartesian hyperplane, between the origin and 1 on all axes. First, the raw data in vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ are normalized as

Table 1 Parameters considered in gathered data

Input variable	Minimum	Maximum	Input variable	Minimum	Maximum
C	0.001	0.795	Heating temperature	1100	1600 °C
Si	0.006	3.200	Heat holding time	60	720 s
Mn	0.010	23.100	Cooling rate	0.08	40.0 °C/s
P	0.000	0.360	Cooling holding time	0.0	900.0 s
S	0.000	0.305	Strain rate	0.0001	0.5 1/s
Cu	0.000	1.420			
Nb	0.000	0.200			
Ni	0.000	36.220			
Al	0.000	8.230			
Mo	0.000	3.410			
N	0.000	0.184			
Cr	0.000	22.330			
V	0.000	0.350			
Ti	0.000	0.099			
B	0.000	0.029			
Sn	0.000	0.192			

$$x_i' = \frac{x_i - Min(x_i)}{Max(x_i) - Min(x_i)} \quad (1)$$

where x_i' represents the normalized value of x_i, $Max(x_i)$ is its highest value, and $Min(x_i)$ is its lowest value.

The goal of this study is to confirm thermal conditions that do not put cast steels in the bending/unbending area of the casting machine when their ductility has a low value. For the most part of steel grades, RA is a U-shaped or V-shaped style. Thus, even if the hot ductility follows a Gaussian curve, the low-ductility region can be predicted easily. However, even under the same thermal conditions, RA at each temperature cannot be predicted using literature data because they include variations depending on the researcher and the experimental apparatus. However, the overall trend of RA is the same even if the accrual values different, Gaussian fitting can be used to predict low-ductility regions. Therefore, to convert the observed RA values into a Gaussian curve, Gaussian fitting [25] is applied to convert them to low temperature limit (LTL), center temperature (CT) and high temperature limit (HTL). HTL represents the median of the maximum RA values, and LTL is the median of the minimum values, and CT indicates the mean of the Gaussian distribution. The ranges of LTL, CT, and HTL calculated in this study were 510.7–918.9 °C, 677.2–977.1 °C, 740.3–1149.3 °C respectively.

3 Data-Driven Models

3.1 RF

RF is an ensemble learning model that combines multiple models, and combines multiple random decision trees [26]. The RF selects n data sets in a bootstrap manner that allows random redundancy in a given learning data, then selects d variables without allowing duplicates in the selected data sample. d is usually the square root of the total number of variables in the given data [27]. The process of generating a decision tree from the selected data samples is repeated k times. Then the mean or multiple predictions are selected of the results from the decision trees created in this way; this is called an ensemble technique. Bootstrap Aggregating uses an ensemble technique that combines multiple bootstrap samples into one classifier; this method is called 'bagging'.

Each decision tree node t is divided into a left child node t_L and a right child node t_R. The optimal node segmentation criterion S that derives the prediction value is calculated as

$$S = \arg max \Delta\hat{i}(s,t) = \hat{i}(t) - \left[\hat{p}(t_L)\hat{i}(t_L) + \hat{p}(t_R)\hat{i}(t_R)\right], \quad (2)$$

where $\hat{p}(t)$ is the conditional probability at node, n_t is the data size at node t, and $\overline{z_t}$ is the average of the output values at node t. Then the impurity function $\hat{i}(t)$ at node t is determined as Mean Squared Error (MSE)

$$\hat{i}(t) = \frac{1}{n_t}\sum_{x_i \in t}\left(y_i - \overline{z_t}\right)^2. \quad (3)$$

The important hyper-parameters in the RF model are the number of decision trees used for bagging, and the depth of decision trees. In general, the learning model's accuracy increases as the increase in the number and depth of decision trees [26]. RF works well without much parameter tuning and is not sensitive to the scale of the data. RF can calculate the importance of independent variables [28]. RF can obtain the type of independent variables used in all nodes of the RF model and the increase in the amount of information obtained from those nodes. The relative importance of each independent variables can be obtained by comparing the average increase in the amount of information that each of them contributes [16].

3.2 GP

Gaussian process assumes that the observed values appear with a unique normal distribution at all-time series points. [29]. GP studies interrelation with mean and covariance for input and output variables. GP represents the set $f(X) = \{(x_i), \ldots, (x_n)\}$ of the output function $f(x_i)$ for the set X of the input values x_i. In this case, each $f(x_i)$ follows a Gaussian distribution. GP is expressed as

$$f(X) \sim GP(m(X), k(X,X)), \quad (4)$$

where $m(X)$ is the mean function of the set of output functions, $k(X,X)$ is the covariance function of X and X, The average function of the output function set is

$$m(X) = E[f(X)], \quad (5)$$

and the covariance function is

$$k(X,X) = E[(f(X) - m(X))(f(X) - m(X))^T], \quad (6)$$

which can also be obtained by the covariance matrix

$$k(X,X) = \begin{pmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \cdots & k(x_2,x_n) \\ \vdots & \vdots & & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \cdots & k(x_n,x_n) \end{pmatrix}, \quad (7)$$

where $k(x_i,x_k)$ is the covariance kernel function of $x_j(j = 1 : n)$ and $x_k(k = 1 : n)$, or the Matern kernel

$$k_{Matern}(x_i,x_k) = \frac{2^{1-v}}{\Gamma(v)}\left(\frac{\sqrt{2v}|x_i - x_k|}{l}\right)^v K_v\left(\frac{\sqrt{2v}|x_i - x_k|}{l}\right) \quad (8)$$

Hyper-parameters r and λ determine the amplitude and twist of the model, respectively. In this study, the maximum likelihood estimation (MLE) method is applied to estimate r and λ. The GP model has the advantage of being able to express the probabilistic attributes of the model by the covariance function. However, as the number of training data increases, computation time and computer memory usage increase exponentially [29].

3.3 SVM

SVM is a classification algorithm [30] that finds an optimal hyperplane that maximizes the size of the margin between categories and that includes as much data as possible within the distance between support vectors [31]. This study uses least-squares SVM [32]. The objective function was

$$\arg min(\vartheta, e) = \frac{1}{2}||\vartheta||^2 + \frac{1}{2}\gamma \sum_{i=1}^{n} e_i^2$$
$$\text{subject to } y_i = \vartheta^T \emptyset\left(x_i\right) + b + e_i, \quad i = 1, \dots, n, \tag{9}$$

where ϑ is a weight vector that determines the size of the margin, γ is a normalization constant, e is the error between the measured value and the predicted value, $\emptyset(x)$ is a nonlinear function that links the input space of the input variable x into a high-dimensional feature space, and b is a bias value.

Kernel SVM is a form of SVM that uses kernels such as Polynomial, Sigmoid, and RBF to compute in a high-dimensional feature space without needing to calculate the coordinates of the data in that space, but rather by easily operating the inner product spaces [16]. Important parameters in the kernel SVM are C and gamma. C determines the number of data samples that can be placed in different classes. The RBF kernel has one γ parameter, which is the inverse of the Gaussian kernel width. Both γ and C adjust the complexity of the model, and must be adjusted together. SVM can simulate high-dimensional nonlinear systems by using a small number of data. However, the process of mapping data to a high-dimensional plane requires a large amount of computation time and memory [32]. SVM has the disadvantage of requiring careful preprocessing of and setting of parameters.

3.4 NN

NN is a structure that computers are adopting to solve problems in a way similar to how humans handle problems through the brain. In this study, a basic multilayer perceptron model [33, 34] is used. It comprises an input layer that accepts stimuli, an output layer that expresses response, and a hidden layer that processes input to yield output. Each layer consists of several nodes and is joined by weighted connections to nodes in other layers [35, 36]. Input to the input layer passes through an activation function to the hidden layer, which processes the input and sends it to the output layer. NN model has two main processes. The first is a feed-forward process that uses a series of input variables, the hidden layer's variables, the relationships (Connectivity, Weight) between each pair of variables, and Transfer Functions. The second process is a backpropagation process, which can increase the accuracy of calculations by responding to the difference between the calculated value and the true value.

The objective function of the NN is to reduce the difference between the output values N(ω, x_i) and the measured value y_i of NN by adjusting the weight ω between nodes:

$$\text{argmin } E(\omega) = \frac{1}{2}\sum_{i=1}^{n}\left(N\left(\omega, x_i\right) - y_i\right)^2, \tag{10}$$

where n is the number of data and x is the input value. In this study, the backpropagation learning algorithm was applied to find the optimal ω [37].

The main hyper-parameters of NN model are the type of output function, the learning rate, the number of hidden-layers, and the number of hidden-layer nodes [38, 39]. The advantage of NN is that they can capture information embedded in large amounts of data and create very complex models [40, 41]. However, data preprocessing is important because all features work relatively well on homogeneous data that have the same meaning. The parameters of NN must be tuned carefully to avoid local optimization and overfitting [42].

4 Evaluation

In this paper, the accuracy of each model is calculated by Root Mean-Squared Error (RMSE) [19, 43–45]

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(z_i - y_i\right)^2}{N}} \tag{11}$$

where z_i is the ith real RA, y_i is the i-tth predicted RA, and N is the number of test data. The RMSE is applied to estimates of LTL, CT, and HTL for all four algorithms.

The evaluation results of ML models lead to the hyper-parameter settings that control its complexity. Therefore, the hyper-parameter setting should be optimized for each model before they are compared. In this study, k-fold cross validation [19, 46] is used to optimize each model. This method randomly divides the training data into k equal subsets, then uses k-1 of them as training data and to predict the rest of the data. The final average accuracy is calculated by averaging the evaluation results obtained by repeating this process for all k subsets.

This study used $k = 10$ for 80% of the training data. The cross-validation was performed with varying hyper-parameters for each ML model, then they were applied to quantify their ability to predict the remaining 20% of the data. For each ML method, the model with the highest average accuracy was identified as the model with the optimal hyper-parameter setting for each ML method. Finally, optimized model was evaluated by calculating its RMSE in predictions of test data. All analyzes in this study used Scikit-learn, Python's ML library.

5 Result

5.1 Model Training

5.1.1 RF

The important hyper-parameters in an RF model are the number of decision trees and the highest depth of each decision tree that is used for bagging. Typically, increasing the number of trees and the depth of the tree improves model accuracy. In this study, the RMSE error of tenfold cross-validation decreased as the depth of decision tree was increased; the average RMSE reached the saturation point when the number of decision trees was 204 and the highest depth was 24. Therefore, subsequent tests used 204 decision trees and a model with a maximum depth of 24. Changes in RMSE were also observed while changing parameters such as the minimum number of samples needed to separate an inner node and the lowest number of samples that should be in the leaf node, the changes were not significant, so but the default values 2 and 1 were used respectively.

5.1.2 GP

The Gaussian-Process-Regressor API supports several kernel combinations, each of which describes several attributes of the data. This paper tested all combinations of kernels supported by the API, and a white kernel that describes noise in the data. The Matern kernel provided the best accuracy. The optimal hyper-parameter of each kernel was found in the same way, and the noise level of the white kernel was optimized to 0.0001. The Matern kernel showed the best accuracy when the length scale of the kernel was 40 and the parameter ν that controls the smoothness of the learned function was 2.5.

5.1.3 SVM

In this study, the RBF kernel was used for the SVR. Polynomial and sigmoid kernels were tested, but the RBF kernel showed the best accuracy (Table 2). To optimize the accuracy of the SVR model, tenfold cross-validation was performed while adjusting the hyper-parameter values C and γ. The highest accuracy was achieved with $C=2.0$ and γ with a "scale" option. We tested other hyper-parameters in kernels, but all showed the best accuracy at the default values. Another hyper-parameter used in the RBF kernel is epsilon, with a default of 0.1. It associates the epsilon tube with no penalty in the training loss function with the estimated point within the epsilon distance from the real value.

Table 2 Evaluation result of four machine learning models in terms of accuracy

Model	RMSE (standard deviation) [°C]		
	LTL	CT	HTL
RF	32.7 (3.4)	24.8 (3.4)	44.8 (3.2)
GP	46.6 (3.7)	43.7 (5.1)	80.1 (17.5)
SVM	44.1 (3.9)	39.8 (2.6)	68.3 (6.0)
NN	20.6 (1.4)	15.2 (1.8)	27.6 (2.9)

5.1.4 NN

NN model used in this study is a multi-layered perceptron containing an input layer, hidden layers, and an output layer. The main hyper-parameters of this model are the number of hidden-layers, the number of hidden-layer nodes, the activation function, the initializer, the learning rate, the dropout probability, and the batch size. The number of hidden layers was increased from 2, and the average accuracy was calculated by tenfold cross-validation while rising the node number in each layer. Accuracy was highest with four hidden layers and 64 nodes. Higher numbers of hidden layers or nodes resulted in overfitting. The activation function used leaky relu ($\alpha=0.2$) because it showed better accuracy than other activation functions. The He Initializer was selected, because it is suitable for leaky-Relu. Optimized learning rate was 0.0001 and optimal batch size was 32. For regularization, early stopping using Dropout (keep probability = 0.75) and Test Set Accuracy were used.

5.2 Accuracy of the Models

This study compared the accuracy of RF, GR, SVM, and NN four ML algorithms to predict LTL, CT, and HTL of steels (Table 2). The evaluation results of the three indicators in the NN model indicated the best accuracy. RMSE of NN were: LTL = 20.6 °C, CT = 15.2 °C, and HTL = 27.6 °C. The model of RF was the next most-accurate; it has a lower accuracy according to the correlation between input variables than other models, which is considered to be influenced by the characteristics of randomly selecting input variables. RMSE of RF were: LTL = 32.7 °C, CT = 24.8 °C, and HTL = 44.8 °C. The SVM and GP model had lower evaluation result. RMSE of SVM were: LTL = 44.1 °C, CT = 39.8 °C, and HTL = 68.3 °C. RMSE of GP were:

LTL = 46.6 °C, CT = 43.7 °C, and HTL = 80.1 °C. NN seems best to predict the hot ductility value, but all were evaluated in a case study.

6 Case Study

6.1 Experimental Procedure

To test the accuracy of the four RA-prediction models, additional high-temperature tensile tests were performed on three steels not included in the previous training and testing (Table 3). The test specimen was fabricated using steel obtained by continuous casting (Fig. 1a). Tensile tests were conducted by a Caster and Thermo-mechanical simulator (40,334, Fuji Electronic Industrial, Saitama). Temperature was elevated to 1673 K (1400 °C) at 10 K/s, kept for 300 s at 1673 K, then reduced to 873–1273 K at 0.5 K/s (Fig. 1b). They were kept at the temperature for 60 s, then subjected to tensile strain of 5×10^{-4} until they broke. RA was gauged for each fractured specimen.

6.2 Analysis of Four Models

The models showed different accuracies in prediction of the ductility troughs of three steels. Steel 1 is an unalloyed steel that typically exhibits a trough shape with the lowest RA at 1073 K (800 °C) (Fig. 2a). For this steel, all four models predict new experimental values well. Steel 2 has a small amount of added alloying elements, and has a wider ductility trough (Fig. 2b) than Steel 1; all models predicted this tendency, although their predictions differ slightly. Steel 3 contains a large amount of alloy metals, and a wide ductility trough (Fig. 2c); only the NN predicted new data adequately.

These results suggest that the NN model is more accurate than the other models for prediction of complex cases that can occur when a large amount of alloy is added. A current trend in steelmaking is to increase the content and diversity of alloying elements to meet the changing requirements of steels. The NN model seems able to anticipate the ductility behavior of these novel steels.

7 Conclusion

This paper has presented development, optimization, and evaluation of four ML models (RF, GP, SVM, and NN) to estimate hot ductility of steel grades from its content and thermal history. Each model was optimized by tuning its hyper-parameters by preprocessing the data. First, all variables in the data were scaled to the range [0, 1], then the experimentally-measured RA values were changed to LTL, CT and HTL by applying Gaussian distribution. Then the most accurate version of each type of ML model was obtained by optimizing their hyper-parameters.

All four models predicted test data well after tenfold cross-validation, although the NN model showed better accuracy than RF, GP, and SVM in all three indicators of

Table 3 Elemental composition (wt%) of steel

Steel	Component									
	C	Si	Mn	P	S	Nb	Ti	Cu	Ni	Cr
Steel 1	0.1	0.01	0.43	0.01	0.01	–	–	–	–	–
Steel 2	0.05	0.01	1.9	0.01	0.002	0.01	0.01	0.28	0.9	0.02
Steel 3	0.05	0.24	0.6	0.003	0.001	0.004	0.0026	0.012	9.2	0.01

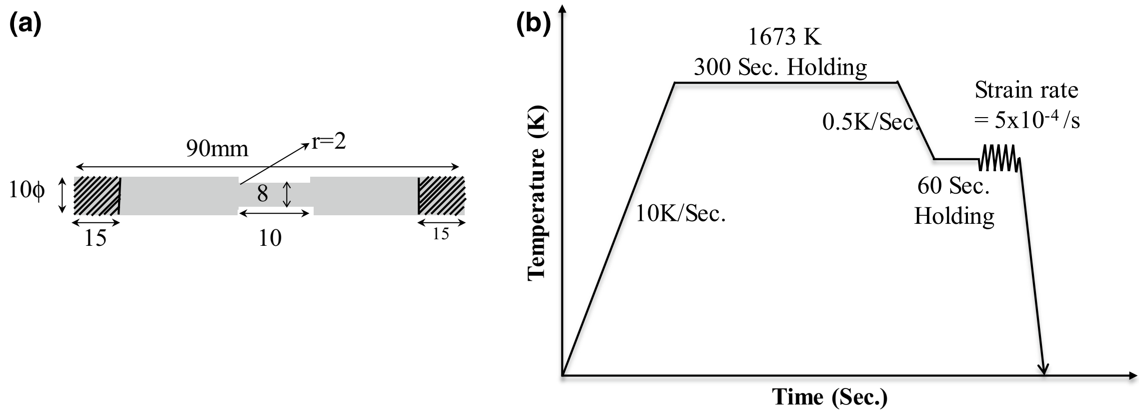

Fig. 1 **a** Specifications of test specimen, and **b** heat-treatment conditions of tensile test

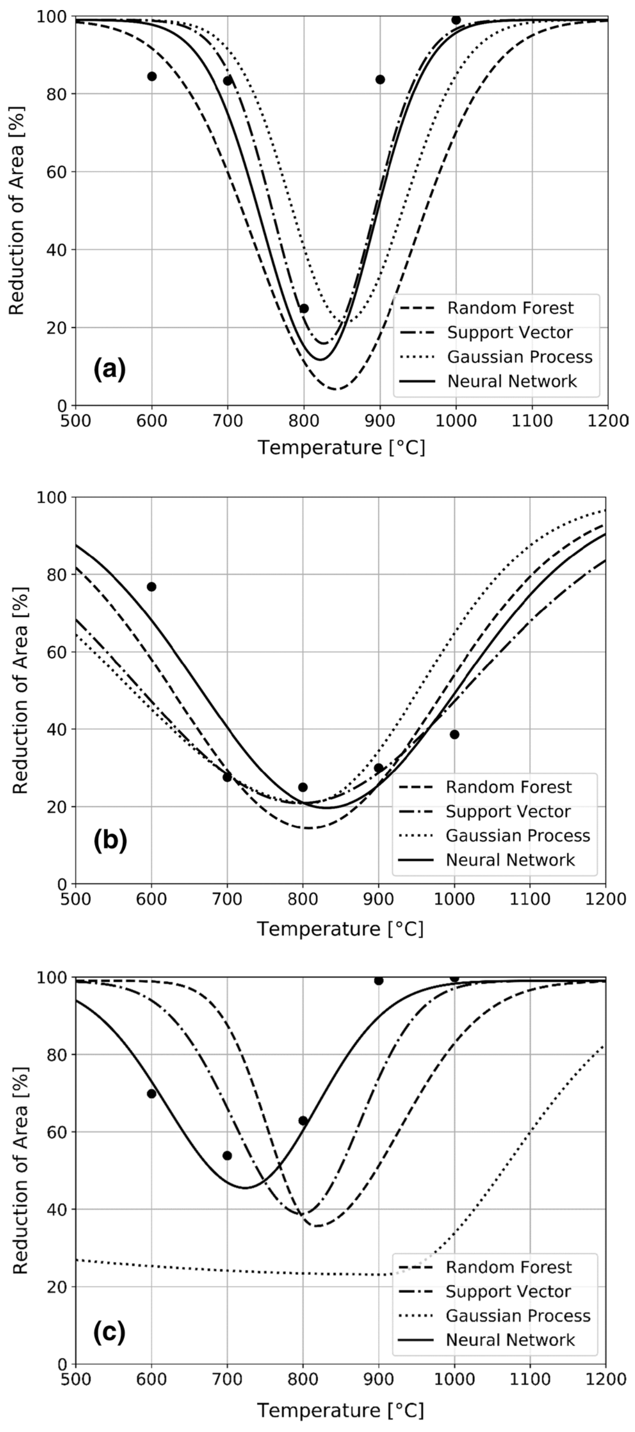

Fig. 2 Comparison of experimental (points) and modeling results (lines) of **a** Steel 1, **b** Steel 2 and **c** Steel 3

LTL, CT and HTL. However, in tests on three steels that had not been used for training and testing, only the NN model predicted all three the ductility troughs well. The difference in results can occur due to the non-linearity of the data and of hot ductility. We infer that the results are influenced by the high-diversity of the data sets, and by the large number of characteristics compared to the number of data. Secondly, RA varies depending on the steel composition and operating conditions, so the correlation between the input and output parameters is very complex. Therefore, the model of NN that best expresses complex nonlinearity is most suitable for predicting hot ductility from steel composition and thermal history. The proposed ML model in this study is more accurate than the physics-based models. This result shows that the ML model is a viable method to predict hot ductility, and to guide optimal control of secondary cooling conditions during continuous casting.

References

1. Z. Li, P. La, J. Sheng, Met. Mater. Int. (2020). https://doi.org/10.1007/s12540-020-00662-4
2. S.C. Seo, K.S. Son, S.K. Lee, Met. Mater. Int. **14**, 559 (2008)
3. B. Kim, S. Jeong, S. Park, Met. Mater. Int. **25**, 201 (2019)
4. C. Wang, C. Shen, X. Huo, C. Zhang, W. Xu, Nucl. Eng. Technol. (2020). https://doi.org/10.1016/j.net.2019.10.014
5. F. Yan, Y.C. Chan, A. Saboo, J. Shah, G.B. Olson, W. Chen, CMES. **117**(3), 343–366 (2018)
6. S.F. Long, M. Zhao, X.F. He, Comput. Mater. Continua (CMC) **58**, 727–760 (2019)
7. L. Wang, Z. Mu, H. Guo, J. Univ. Sci. Technol. Beijing Miner. Metall. Mater. **13**(6), 512–515 (2006)
8. M.R. Toroghinejad, M.B. Esfahani, *Rijeka: Artificial Neural Networks: Industrial and Control Engineering Applications* (Intech, London, 2011)
9. P.Y. Chou, J.T. Tsai, J.H. Chou, IEEE Access. **4**, 585–593 (2016)
10. T. Thankachan, K. Sooryaprakash, Arab. J. Sci. Eng. **43**(3), 1335–1343 (2018)
11. S.I. Hong, Met. Mater. **6**, 275–279 (2000)
12. X.P. Li, J.K. Park, J. Choi, Met. Mater. **5**, 25–32 (1999)
13. S.C. Seo, H.J. Kim, B.H. Park, Met. Mater. Int. **12**, 273 (2006)
14. Z. Sterjovski, D. Nolan, K.R. Carpenter, J. Mater. Process. Technol. **170**, 536–544 (2005)
15. S.H. Kwon, D.G. Hong, C.H. Yim, Ironmak. Steelmak. **22**, 1–2 (2019). https://doi.org/10.1080/03019233.2019.1699358
16. A. Muller, S. Guido, *Sebastopol* (O'Reilly Media, California, 2016)
17. K. Ažman, J. Kocijan, ISA Trans. **46**(4), 443–457 (2007)
18. D. Michie, D.J. Spiegelhalter, C.C. Taylor, *Machine Learning, Neural and Statistical Classification* (Ellis Horwood, New York, 1994)
19. A. Agrawal, P.D. Deshpande, A. Cecen, G.P. Basavarsu, A.N. Choudhary, S.R. Kalidindi, Integr. Mater. Manuf. Innov. **3**, 8 (2014)
20. A. Muller, S. Guido, *Sebastopol* (O'Reilly Media, California, 2016)
21. Q. Liu, X. Zhang, B. Wang, in *Science and Technology of Casting Processes*, ed. by M. Srinivasan (InTech, London, 2012)
22. C. Spradbery, B. Mintz, Ironmak. Steelmak. **32**, 319–324 (2005)
23. Q. Liu, X. Zhang, B. Wang, in *Science and Technology of Casting Processes*, ed. by M. Srinivasan (InTech, London, 2012)
24. Z.W. Xu, X.M. Liu, K. Zhang, IEEE Access. **7**, 47068–47078 (2019)
25. L. Lawrence, J. Mach. Learn. Res. **6**, 1783–1816 (2005)
26. N. Brieman, Mach. Learn. **45**(1), 5–32 (2001)
27. L. Breiman, Mach. Learn. **24**(20), 123–140 (1996)

28. G. Louppe, Doctoral dissertation, University of Liège Liège, Belgium, 2014
29. C.E. Rasmussen, C.K.I. Williams, *Gaussian Processes for Machine Learning* (MIT Press, Cambridge, 2006)
30. E. Ceperic, V. Ceperic, A. Baric, IEEE Trans. Power Syst. **28**(4), 4356–4364 (2013)
31. H. Wang, D. Hu, in *Proceedings of the International Conference on Neural Networks and Brain*, vol. 1 (2005)
32. L. Bottou, C.J. Lin, *Large Scale Kernel Machines* (MIT press, Cambridge, 2007)
33. N. Buduma, *Sebastopol* (O'Reilly Media, California, 2017)
34. E. Maleki, O. Unal, Met. Mater. Int. (2019). https://doi.org/10.1007/s12540-019-00448-3
35. C.H. Park, D. Cha, M. Kim, Met. Mater. Int. **25**, 768–778 (2019)
36. P.L. Narayana, C. Li, J. Hong, Met. Mater. Int. **25**, 1063–1071 (2019)
37. S. Singh, H. Bhadeshia, D. MacKay, H. Carey, I. Martin, Ironmak. Steelmak. **25**, 355–365 (1998)
38. F.D. Foresee, M.T. Hagan, in *Proceedings of the International Conference on Neural Networks* (1997)
39. M. Mesbah, A. Fattahi, A.R. Bushroa, Met. Mater. Int. (2019). https://doi.org/10.1007/s12540-019-00495-w
40. Y.C. Lin, H. Yang, D. Chen, Met. Mater. Int. (2019). https://doi.org/10.1007/s12540-019-00435-8
41. P.J. Angeline, G.M. Saunders, J.B. Pollack, IEEE Trans. Neural Netw. **5**(1), 54–65 (1994)
42. X. Yao, Proc. IEEE **87**(9), 1423–1447 (1999)
43. N. Sandhya, V. Sowmya, C.R. Bandaru, G.R. Babu, Int. J. Recent Technol. Eng. **8**(3), 235–241 (2019)
44. I. Santos, J. Nieves, Y.K. Penya, P.G. Bringas, in *ICCAS-SICE 2009: ICROS-SICE International Joint Conference 2009* (2009)
45. S. Guoa, J. Yub, X. Liuc, C. Wang, Q. Jiang, Comput. Mater. Sci. **160**, 1–8 (2019)
46. J. Schmidt, M.R.G. Marques, S. Botti, M.A.L. Marques, NPJ Comput. Mater. **5**, 1–36 (2019)