Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 120 (2023) 1077–1082

www.elsevier.com/locate/procedia

56th CIRP Conference on Manufacturing Systems, CIRP CMS '23, South Africa

Machine learning for predicting dimensions of extrusion blow molded parts: A comparison of three algorithms

Christian D. Øien[a,*], Torbjørn L. Leirmo[b]

[a]*Department of Mechanical and Industrial Engineering, Norwegian University of Science and Technology, 7491 Trondheim, Norway*
[b]*SINTEF Manufacturing, 2830 Raufoss, Norway*

Abstract

In the perspective of feed-forward control through a manufacturing process chain, production data can be used downstream to correct subsequent processes and improve product quality. In an extrusion blow-moulding (EBM) use case, supervised learning (SL) has been applied to predict geometrical dimensions based on process data, as a possible basis for feed-forward control. The labeled tabular dataset showed significant time dependency which complicates the learning task. In this paper we present a time-series regression approach and compare three common SL algorithms – Random Forest, Gradient Boosting and XGBoost. The results show that only part of the variations in product geometry could be learned from process data and that future work will be necessary in order to increase the models' performance.

Keywords: Manufacturing; machine learning; supervised learning; feed-forward control

1. Introduction

Recent developments in information and communication technologies, computer science, and data science have brought along a whole new paradigm of manufacturing. Popularly known as smart manufacturing or industry 4.0, this evolution of digital transformation in the manufacturing industry has brought along many new possibilities for increased efficiency and productivity through self-adapting and self-optimizing systems [3]. While the necessary tools and concepts were introduced a long time ago, the infrastructure and computational power have now reached a level where artificial intelligence and self-optimizing systems are feasible in production systems [14]. Consequently, the industry is currently undergoing widespread adoption of related technologies, at the same time as the uniqueness of every manufacturing system and each manufacturing process gives rise to challenges in specific implementations.

Development and application of intelligent systems in manufacturing using machine learning (ML) has been ongoing for decades [15]. Various ML techniques have been applied to problems in machining [9], forming [16], and additive manufacturing [1] to predict product quality and optimize process parameters.

For extrusion blow molding (EBM), specifically, artificial neural networks (ANN) were used to predict the dimensions of a blow molded part already in 1993 [5]. ANNs have also been used for predicting dimensions and material distributions of the parison (see section 2.1) [12], which eventually also affect the final dimensions of the product. Other methods applied to the EBM process include gradient-based optimization [8] and genetic algorithms [20] for process optimization. Other efforts have employed neural networks to establish a foundation for process optimization with a genetic algorithm [11, 19]. However, the total applications of ML to the EBM process are limited, and there seems to be a knowledge gap in mapping product properties to high-dimensional production process data collected over time.

Generally, in discrete manufacturing applications data mining and feature extraction [21] can be used to link process data with product characteristics. In view of ML-based approaches, this will work as the base for applying supervised learning (SL) on the resulting tabular data, in the form of either classification or regression. The focus in this paper is regression, where the trained model is a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ predicting the nu-

[*] Corresponding author. Tel.: +47-97736238; *E-mail address:* christian.d.oien@ntnu.no

merical value of a certain characteristic of a Work-in-Progress (WIP) part based on *n* input features describing the corresponding production processes. Specifically, we evaluate the time-dependency of such a dataset acquired over time from an EBM process and investigate how various SL algorithms tackle this time dependency and are able to generalize the function f so that predictions can be made on future WIP parts. In the case of a manufacturing process chain based on EBM, it is of interest to evaluate if feed-forward control of proceeding processes, i.e. on-line adjustments to those processes to counteract unwanted variations in EBM, could be implemented based on ML predictions.

Manufacturing execution systems (MES) may collect data from hundreds or thousands of sensors, along with a comparable amount of varying parameter values used in process control. The motivation for applying ML to estimate product dimensions based on process data is that complex transient dependencies can be learned from such training data. The governing research question of the current work has therefore been defined as:

In the case of EBM, to which extent can features extracted from continuously collected process data be used to learn the complex dependencies that affect variations in product dimensions?

The remainder of the paper is structured as follows: First, a thorough description of the methods is given in section 2 before the results are presented and briefly analyzed in section 3. A discussion of the findings and their implications is made in section 4 before conclusions are made and future work is outlined in section 5.

2. Methods

2.1. Manufacturing use case

The data was logged from an EBM process, which is used to produce hollow plastic parts. Simply put it works by extrusion-feeding molten plastic around a mandrel to form a so-called parison. This loose tube of hot plastic, at a suitable length pulled by its own weight, is then enclosed by a tool and expanded by pressurized air towards the inner tool surface. This is illustrated in Fig. 2.1. By water cooling of the tool, the part hardens in a few seconds. After the tool opens, the part is cut loose and a new parison can form the next part in a cyclic manner.

The EBM process is automated and regulated in order for the product to have correct weight, correct vertical distribution of material, correct dimensions, etc. In our use case the system controlling the process and automatically regulating some of the process parameters is digitalized and stores about 170 feature values for each product. These features, describing the conditions and regulation of the EBM process, have been the base for our work and consist of several temperatures, forces, pressures, durations, positions, velocities, and flow rates. They describe the feeding and extrusion of plastic material, parameters regulated in connection with forming the parison, the state of the plastic material at various process stages, water and air cooling, the air pressurization during forming and durations of specific parts of the process.

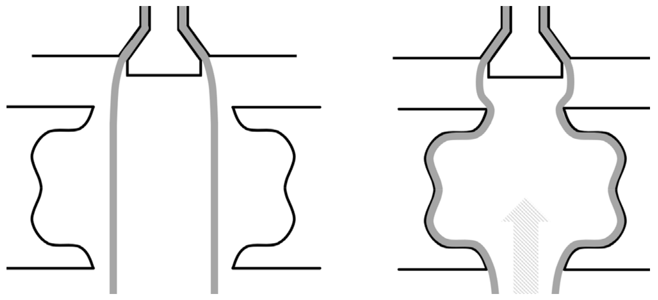
Fig. 1. Schematic representation of the extrusion blow molding process

2.2. Data collection

In the studied EBM manufacturing set-up, the manufacturing execution system (MES) was programmed to store data as features, i.e. as representative values for each produced part, as opposed to continuous time series. The data relating to one specific product variant produced were exported from a database as a comma separated values (CSV) file, covering the year 2021. Additionally, time series data regarding the temperature and pressure of the cooling water reservoir was collected from a separate system, re-sampled, and merged time-wise with the feature table.

The products were regularly measured in a measurement fixture at a rate of about 1 per 1000, based on a defined routine. Four outer dimensions and two wall thickness measurements from each product were taken as labels that should be predicted by regression. The production feature table and the table of labels were merged by an inner join on the product ID to ensure coherence.

2.3. Data preparation

Substantial work was done to go through each process feature with process engineers, building on their process knowledge to exclude features whose origin is questionable either in accuracy, variance, or relevance to the response variables. Several of the excluded features were binary machine signals, but also some air pressures, calculated air volumes, and sub-process durations. Next, the dataset was methodically cleaned by removing invalid entries and parameters of constant value. Feature data formats were checked to ensure conformance between the different data sources. The data preparation was completed using Python 3 in a Jupyter Notebook environment and the Pandas [13] package. Six synthetic features were also created, based on existing process features (measurements), to assist the ML algorithms. This included calculations of energy transfer and heat dissipation during the blow molding process. The final dataset contains 1238 rows with 100 input features named $x_1, x_2, ..., x_{100}$ and 6 output variables as labels named $y_1, y_2, ..., y_6$. The first four output variables are outer product dimensions while the last two are thicknesses. A standardized version of this dataset, i.e. where features are shifted by their mean and divided by their standard deviation as shown in Table 1. This dataset, denoted D0, is the basis for the presented research and is shared and available on GitHub [17].

Table 1. Excerpt of the D0 dataset. Numbers are rounded for brevity.

timestamp	x1	⋯	x100	y1	⋯	y6
2021-01-10 15:52:00	-0.068	⋯	0.424	-0.172	⋯	-2.110
2021-01-10 15:52:00	-0.068	⋯	0.424	-0.172	⋯	-2.110
2021-01-10 15:53:00	0.015	⋯	1.483	0.432	⋯	-1.117
⋮	⋮	⋮	⋮	⋮	⋮	⋮
2021-12-17 06:08:00	1.602	⋯	0.754	-0.172	⋯	-0.521
2021-12-17 06:08:00	1.185	⋯	-0.154	0.432	⋯	-1.017
2021-12-17 06:08:00	1.185	⋯	-0.154	0.432	⋯	-1.017

2.4. *Correlations and dimensionality reduction*

In this D0 dataset there are significant correlations between certain groups of input variables. In order to deal with this effectively it was chosen to apply Principal Component Analysis (PCA) [21]. Based on a PCA transformation of the input variables, the resulting eigenvalues of each eigenvector can be used to define a suitable number of principal components to use, effectively reducing the dimensionality of the dataset without significantly losing information.

Based on the 100 input variables of D0 it was chosen to keep 50 principal components, retaining 99% of the variance while removing correlations. See Fig. 2. The resulting *PCA-transformed* dataset of total size 56×1238 is referred to as D1.

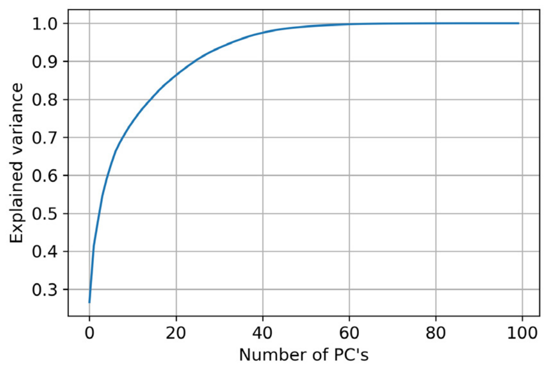

Fig. 2. The ratio of explained variance in the D0 dataset by a subset of (the first) n principal components, as a function of n.

2.5. *Train/test-splitting of time-dependent datasets*

Splitting of data into sets for training, validation and testing is paramount in any SL application. Thus, separate subsets of data are used to learn the unknown function, validate model performance during training and tune hyperparameters, (ref. section 2.8) and evaluate final model performance, respectively. If a static function is expected to map inputs to outputs, data is often randomly shuffled before setting aside certain portions for training, validation, and testing. For time-series data, however, where the input-output dependencies are expected to change over time, it is necessary to define these subsets without shuffling.

During the current work, it was noted that an excellent test set regression accuracy could be achieved on a test set of D0 resulting from a shuffled split, while the same model would yield wildly misleading predictions on a chronologically held out second test set. This shows that the collected data is strongly time-dependent, and that over-fitting may occur if the test set is directly or indirectly represented in the training set. In the following work, all test set splits were done sequentially, while a subsequent shuffled split of training and validation data has been regarded as legitimate.

2.6. *Lag features*

Generally, the characteristics of a given product may be assumed to correlate with both prior process data (x) and the characteristics of previous products (y). In order to facilitate learning of such dynamic patterns one can introduce *lag features*. To facilitate this an extended dataset with lag features was constructed based on D1, where 5 preceding sets of input and output values (both x and y) were added to each row, resulting in 330 input features. This dataset of size 336×1238 is referred to as D2 in the following.

2.7. *Random forests, Gradient Boosting and XGBoost*

Random Forest (RF) is a well-known class of SL algorithms based on the decision tree. Even though there are several variants, it is generally an ensemble learning method that works by constructing a large number of decision trees from sub-samples of the training dataset [10, 2]. It can be seen as a parallelization of several decision trees that together minimize generalization error. A decision tree that estimates a certain (set of) continuous values at each node is sometimes called a regression tree.

Gradient Boosting [7] (GB) works by building an additive expansion of the unknown function (that we seek in order to map inputs to outputs) by superposing ”base learners” as functions of the inputs. In the GradientBoostingRegressor class of the Scikit-learn library [18] the base regressors are regression trees.

Extreme Gradient Boosting (XGB) [4] is a type of Gradient Boosting algorithm with emphasis on a specific type of regularization. Simply put, in addition to minimizing a differentiable convex loss function, it penalizes model complexity in a way that naturally prevents over-fitting as well as being well suited for parallelized computation.

All of the three aforementioned algorithm types can be found in *classification* (predicting one or more discrete values) and *regression* (predicting one or more continuous values) variants.

In the case of multidimensional production data, tree-based algorithms are a natural starting point since they often work well on tabular datasets, where the task is, quite explicitly, to learn a function of several variables. In addition, they are some-what easier to train (and design) than deep learning approaches with ANN that depend on more extensive hyperparameter optimization.

2.8. *Hyperparameter optimization*

Most machine learning algorithms have a vast range of adjustment possibilities, such as the type of objective function to use, the evaluation metric used during model validation, the number of iterations or components, and several coefficients depending on the algorithm type. Such *hyperparameters* need to be optimized over a given parameter space, and the optimum combination will depend on the dataset used. A complicating factor is that the effect of the hyperparameters on the model generalization accuracy are inter-dependent. To carry out this optimisation an exhaustive grid search was implemented based on the GridSearchCV class of scikit-learn, with a 3-fold cross-validation of each hyperparameter combination. The searched parameter spaces for each algorithm, along with the optimisation results, are presented in Tables 2, 3, and 4. As a simplification, the combination of output variable y_3 and the D2 dataset was used for the optimisation of each model type. The resulting optimized parameter sets were then used across output variables and for both the D1 and D2 based analysis. Please refer to the aforementioned GitHub repository [17] for further details and the code used.

Table 2. Grid search set up and results for Random Forest

Hyperparameter	Levels	Optimum
min_samples_split	2, 3, 5, 10	3
max_depth	5, 10, 20, 30	30
criterion	squared_error, friedman_mse	friedman_mse
max_samples	0.7, 0.8, 0.9	0.9

Table 3. Grid search set up and results for Gradient Boosting

Hyperparameter	Levels	Optimum
min_samples_split	2, 5, 10	5
loss	squared_error, absolute_error, huber	huber
max_depth	3, 4, 5	3
max_features	sqrt, 30, 40	40
criterion	friedman_mse, squared_error	squared_error
subsample	0.6, 0.7, 0.8	0.8

Table 4. Grid search set up and results for XGB

Hyperparameter	Levels	Optimum
min_child_weight	2, 5, 10	5
gamma	0.01, 0.02, 0.05, 0.1	0.01
subsample	0.5, 0.6, 0.7	0.7
colsample_bytree	0.6, 0.7, 0.8	0.8
max_depth	3, 4, 5	5
objective	reg:(squarederror, pseudohubererror)	reg:squarederror

2.9. *Regression standard error*

Standard error has been chosen as a metric suitable to compare the accuracy of regression models. Given a vector of true (observed) values y_i and corresponding regression estimates $\hat{y}_i$ of output variable i, it is defined as $S_{e,i} = \sigma(y_i - \hat{y}_i)/\sigma(y_i)$ where σ denotes the sample standard deviation.

3. Results

Regression accuracy has been compared between RF, GB and XGB models that have undergone hyperparameter optimisation as described in section 2.8 and trained on the D1 and D2 datasets, respectively, showing the effect of including lag features and the algorithms' ability to learn a generalized function to estimate the six product dimensions $y_1, y_2, \cdots, y_6$. Each algorithm was trained 20 times, whereas each model was used to predict the output variables of the chronologically split hold-out test set. To compare predicted and true values the regression standard error were calculated for each output variable and for each combination of algorithm and dataset. The 20 repetitions were then used to estimate the 95% confidence interval of each achieved accuracy. These results are shown in Table 5.

From the results we can see that four of the output variables were predicted more accurately by models trained on the D2 dataset (with lag features), while the two other by models trained on the D1 dataset (without lag features). Furthermore, it can be observed that the random accuracy variation is highest for the GB models, lower for the XGB models, and clearly lowest for the RF models.

To visualize the performance of the over-all best model (RF trained on D2), the predictions of the most and least accurately predicted output variables by that model, i.e. y_4 and y_1, are plotted in Fig. 3. Additionally, a scatter plot of true vs. predicted values of y_4 using the same model is shown in Fig. 4.

In Fig. 3 one should note a discrepancy in the first approx. 70 data points, where y_4 seems to be predicted in an opposite manner. For the rest of the data points, the predictions are much better. In Fig. 4 this is seen as over-predictions in the true range of approx -3 to -1.

4. Discussion

The current work describes a case of highly time-dependent multivariate data. Despite testing multiple algorithms for predicting physical dimensions based on process variables, the trained models are only partially able to predict product geometry. The time dependency of the data appears to outweigh the influence of the variables captured for predictions. The following explanations may hold merit:

1. The sampling rate (around 1/1000) is too low to capture the data needed to properly train any of the tested ML algorithms;

Table 5. Regression model accuracy in the form or standard error per output variable. D1 and D2 indicate datasets with and without lag features, respectively. Values are given as a 95% confidence interval based on 20 repeated training and testing iterations. Bold face numbers indicate the highest accuracy for each variable/average.

	$S_{e,1}$	$S_{e,2}$	$S_{e,3}$	$S_{e,4}$	$S_{e,5}$	$S_{e,6}$	average
GB (D1)	1.246±0.116	0.697±0.058	1.068±0.058	**0.738±0.042**	0.874±0.050	1.085±0.096	0.951±0.030
XGB (D1)	1.127±0.064	**0.693±0.018**	1.043±0.026	0.768±0.022	0.835±0.022	0.996±0.044	0.910±0.018
RF (D1)	1.092±0.028	0.704±0.014	1.058±0.020	0.796±0.010	0.814±0.014	0.949±0.024	0.902±0.004
GB (D2)	1.030±0.074	0.729±0.040	0.901±0.056	0.742±0.050	0.767±0.024	0.877±0.044	0.841±0.022
XGB (D2)	0.966±0.036	0.723±0.018	0.902±0.032	0.749±0.022	**0.716±0.020**	0.833±0.034	0.815±0.010
RF (D2)	**0.926±0.018**	0.760±0.012	**0.889±0.020**	0.745±0.010	0.747±0.016	**0.785±0.016**	**0.809±0.008**

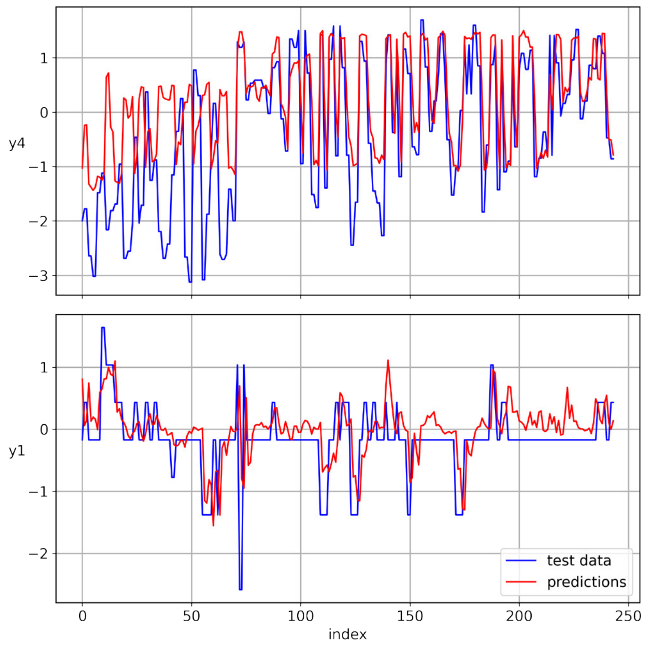

Fig. 3. Time plot of the true values of y_1 and y_4 together with and their corresponding Random Forest estimates trained on the D2 dataset.

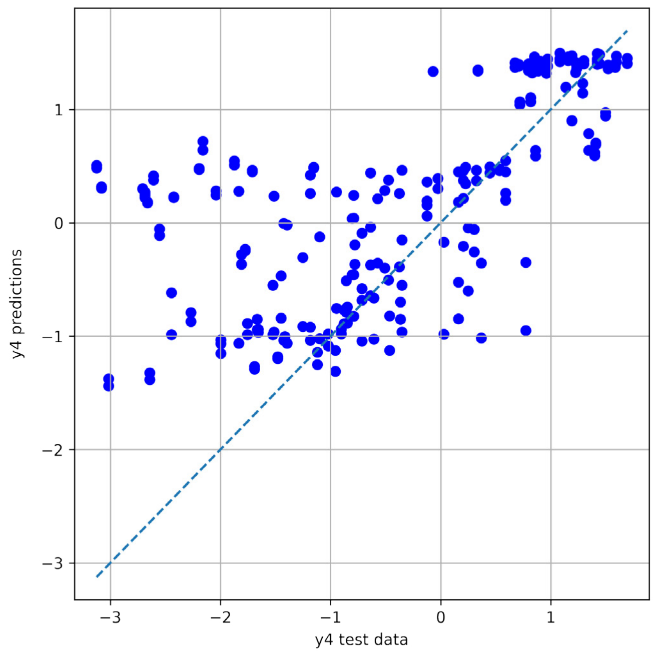

Fig. 4. Scatter plot of the true vs. predicted values of y_4 based on the Random Forest model trained on the D2 dataset.

2. The process is not stable, and there are unknown factors that outweigh the significance of the recorded variables; or
3. The dataset is too small to allow accurate predictions without over-fitting

Even though the added lag features in the D2 dataset do contribute, signalling that some dynamic effects are learnt, the increase in prediction accuracy is marginal. Referring to the chosen metric of regression standard error, a result of $S_e = 1$ would indicate a range of prediction error similar to the range of true value variation. Consequently, statically predicting the average $\hat{y} = \bar{y}$ would give the same accuracy, so a good model would yield a result closer to zero that what was achieved in this paper. In summary, we can assume that the label values (part measurements) are significantly affected by relations or dynamics that can't be trained from the gathered data. This is due to the fact that the measurement equipment is routinely calibrated and the uncertainty controlled, so that random error is not likely the reason for poor performance.

It is notable that RF yielded the highest accuracy on both the D1 and D2 datasets, since especially XGB is regarded as an improved algorithm over RF and GB due to its clever regularization and learning setup. As noted in section 3, Fig. 3 shows a pattern of opposite predictions on the first part of the test set, and a possible explanation for this could be multi-modality of the studied manufacturing process leading to learning instabilities. It could seem that XGB in this case is more prone to this complexity.

Regardless of whether any of the above is true, the time dependency is believed to have a significant impact on the efficacy of the ML algorithms. This is evident because the manner in which the dataset is split into training, testing and validation sets – chronologically or randomly shuffled – greatly affects the performance of the trained models. While a model trained on randomly shuffled data will yield excellent test results, the same model performs poorly if the test set is chronologically split.

To answer the defined research question, the studied ML approach seems to be able to predict only parts of the variations seen in product geometry based on EBM process data. Consequently, it is questionable whether such predictions may provide value in a feed-forward control system. Nevertheless, it could be evaluated whether a classification approach would be more suitable than estimating continuous labels with a regression approach. In a simplified feed-forward control application, where a proceeding manufacturing process should be adjusted according to ML-based predictions, the adjustments could indeed be based on a discrete scheme instead of a continuous one.

5. Conclusions and future work

This paper presented a multivariate dataset from an EBM process that exhibits time dependent traits. The performance of three ML algorithms – namely RF, GB, and XGB – were compared in terms of prediction accuracy. The results show that RF performed the best on this dataset, but more importantly that all of the trained models were only partially able to predict product geometry. Further work is therefore required in order for such a model to be suitable for feed-forward control or subsequent manufacturing processes.

The findings of the present study warrant further research into the dynamics of the EBM process. For future work, designed experiments may reveal patterns that are unclear due to sampling frequency. Further on, a larger dataset may address the same challenge, especially with a higher sampling rate. Finally, possible directions for future work are deep learning methods such as recurrent neural networks (RNNs) to inherently learn dynamics based on preceding data points, or hybrid machine learning methods such as pseudo-Hamiltonian neural networks [6] to actually learn the physics and disturbances of the EBM process.

Acknowledgements

This work was funded by the Norwegian Research Council through DataVar (310095) and AluGreen (328831). We also wish to thank prof. Geir Ringen and prof. Are Strandlie for fruitful discussions.

References

[1] I. Baturynska and K. Martinsen. "Prediction of geometry deviations in additive manufactured parts: comparison of linear regression with machine learning algorithms". In: *Journal of Intelligent Manufacturing* 32.1 (2020), pp. 179–200. DOI: 10.1007/s10845-020-01567-0.

[2] L. Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[3] M. Brettel et al. "Enablers for Self-optimizing Production Systems in the Context of Industrie 4.0". In: *Procedia CIRP* 41 (2016), pp. 93–98. DOI: 10.1016/j.procir.2015.12.065.

[4] T. Chen and C. Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, California, USA: ACM, Aug. 13, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.

[5] R. W. Diraddo and A. Garcia-Rejon. "On-line prediction of final part dimensions in blow molding: A neural network computing approach". In: *Polymer Engineering and Science* 33.11 (1993), pp. 653–664. DOI: 10.1002/pen.760331102.

[6] S. Eidnes et al. "Pseudo-Hamiltonian neural networks with state-dependent external forces". In: *Physica D: Nonlinear Phenomena* 446 (2023), p. 133673. DOI: 10.1016/j.physd.2023.133673.

[7] J. H. Friedman. "Stochastic gradient boosting". In: *Computational Statistics & Data Analysis* 38.4 (2002), pp. 367–378. DOI: 10.1016/S0167-9473(01)00065-2.

[8] C. Gauvin, F. Thibault, and D. Laroche. "Optimization of blow molded part performance through process simulation". In: *Polymer Engineering & Science* 43.7 (2003), pp. 1407–1414. DOI: 10.1002/pen.10119.

[9] T. Ghosh and K. Martinsen. "Generalized approach for multi-response machining process optimization using machine learning and evolutionary algorithms". In: *Engineering Science and Technology, an International Journal* 23.3 (2020), pp. 650–663. DOI: 10.1016/j.jestch.2019.09.003.

[10] T. K. Ho. "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1. Montreal, Que., Canada: IEEE Comput. Soc. Press, 1995, pp. 278–282. DOI: 10.1109/ICDAR.1995.598994.

[11] G.-Q. Huang and H.-X. Huang. "Optimizing parison thickness for extrusion blow molding by hybrid method". In: *Journal of Materials Processing Technology* 182.1 (2007), pp. 512–518. DOI: 10.1016/j.jmatprotec.2006.09.015.

[12] H.-X. Huang et al. "New Strategies for Predicting Parison Dimensions in Extrusion Blow Molding". In: *Polymer-Plastics Technology and Engineering* 50.13 (2011), pp. 1329–1337. DOI: 10.1080/03602559.2011.584234.

[13] W. McKinney. "Data Structures for Statistical Computing in Python". In: Python in Science Conference. Austin, Texas, 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[14] L. Monostori et al. "Cyber-physical systems in manufacturing". In: *CIRP Annals* 65.2 (2016), pp. 621–641. DOI: 10.1016/j.cirp.2016.06.005.

[15] L. Monostori et al. "Machine Learning Approaches to Manufacturing". In: *CIRP Annals* 45.2 (1996), pp. 675–712. DOI: 10.1016/S0007-8506(18)30216-6.

[16] O. Ogorodnyk et al. "Application of feature selection methods for defining critical parameters in thermoplastics injection molding". In: *Procedia CIRP* 81 (2019), pp. 110–114. DOI: 10.1016/j.procir.2019.03.020.

[17] C. D. Øien. *publication1*. Version v1.1. May 5, 2023. DOI: 10.5281/ZENODO.7898886.

[18] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html (visited on 04/28/2023).

[19] J.-C. Yu and J.-Y. Juang. "Design optimization of extrusion-blow-molded parts using prediction-reliability-guided search of evolving network modeling". In: *Journal of Applied Polymer Science* (2010), NA–NA. DOI: 10.1002/app.31954.

[20] J.-C. Yu et al. "Optimization of extrusion blow molding processes using soft computing and Taguchi's method". In: *Journal of Intelligent Manufacturing* 15.5 (2004), pp. 625–634. DOI: 10.1023/B:JIMS.0000037712.33636.41.

[21] A. Zheng and A. Casari. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. 1st. O'Reilly Media, Inc., 2018.