Journal of Big Data

METHODOLOGY Open Access

A predictive noise correction methodology for manufacturing process datasets

Omogbai Oleghe*

*Correspondence:
o.oleghe@ritepaklimited.com
Rite Pak Co. Ltd, Ogun, Km 38 Abeokuta Expressway, Ota, Nigeria

Abstract

In manufacturing processes, datasets intended for data driven decisions are majorly generated from time-sequenced sensor readings. Industrial sensor systems are prone to transmit inaccurate readings, which result in noisy datasets. Noisy datasets inhibit machine learning and knowledge discovery. Using a multi-stage, multi-output process dataset as an experimental case, this article reports a methodology for replacing erroneous sensor values with their predicted likely values. In the methodology, invalid values specified by process owners are first converted to missing values. Then, ReliefF algorithm is used to select the most relevant features to progress for prediction modelling, and also to boost the performance of the prediction model. A Random Forest classifier model is built to predict replacement values for the missing values. Finally, predicted values are inserted into the dataset to fill in the missing entries. With many attributes having a significant number of erroneous values, the invalid values replacement is done one attribute at a time. To do this systematically, the process flow direction and stages in the manufacturing process are exploited to partition the dataset into subsets for model building. The results indicate that the methodology is able to replace erroneous values with likely true values, to a very high degree of accuracy. There is a paucity of this type of methodology for dealing with invalid entries in process datasets. The methodology is useful for both missing and invalid value correction in process datasets. In the future, the plan is to inject the prediction models into streaming data to simultaneously enable erroneous value correction and predictive process monitoring in real-time.

Keywords: Machine learning, Manufacturing process, Noise correction, Feature reduction, Classification prediction

Introduction

Data driven decisions in manufacturing systems rely much on time-sequenced readings from multiple time-synchronized sensors. In real-life settings, sensors are perturbed by mechanical vibrations and electronic signals. They are also prone to generate erroneous values when sensing the product feature characteristics in high-speed production lines, for example where the product frequently shifts from desired position. In other processes, such as extrusion lines, the extruded product may be stained, for example with cooling liquids, and this can cause sensors to misread. Erroneous sensor values can lead to inaccurate decisions, for example, raising false alarms on defects [1]. Additionally, noisy datasets are difficult to interpret by those not familiar with the system. Where noisy values are significant in the dataset, they inhibit machine learning performance and can also lead to false predictions [2]. It may be infeasible or expensive to change sensors or the system, and so decision makers are faced with no other option than to rely on noisy datasets.

This research explores the use of classification prediction modelling to replace erroneous values in time series datasets of multi-stage manufacturing processes. Given that such datasets may have hundreds to thousands of sensor-generated parameters, feature set reduction through feature relevance ranking is considered for boosting the performance of the prediction model. A dataset [3] that was generated using a Manufacturing Execution System was used for the research. The 116-feature, 14,088-entries dataset included timeseries information for: ambient factory conditions; material property specifications; machine operating parameters and dimensional characteristics of the output exiting from each of the two stages. In addition to the dataset being multivariate and of medium dimensionality, it contained a lot of erroneous values. The objective was to denoise the dataset by replacing the erroneous values with values predicted using classification modelling. Noisy erroneous values in the batched dataset are first replaced with missing values. Then missing values are inputted with predicted values using a built classifier model.

An end-to-end methodology for replacing erroneous values in time series datasets of multi-stage processes is proposed. The proposed methodology comprises data exploration, data pre-processing and preparation, feature reduction to boost the performance of the classification model that is built, results parsing with cleaning and predicted values inputting. Data exploration, preparation, parsing and inputting are undertaken using python-based libraries and codes. The machine learning tasks are performed using WEKA open source software. Although the methodology is described for offline modelling, the methodology culminates in a model that can be used for correcting erroneous streaming data. The built model can also be used for real-time predictive process monitoring. In other words, the methodology delivers multiple benefits aside correcting erroneous values in a dataset.

Consequently, this article is structured as follows: "Dataset de-noising and erroneous values correction " section discusses data de-noising and erroneous values correction in large datasets; "Materials and methods" describes the materials and methods and presents the proposed methodology; "Results and analysis" section reports the application of the methodology with results and analysis; "Discussions and conclusions" section discusses the research implications with concluding remarks.

Dataset de-noising and erroneous values correction

Noise as it relates to data is the existence of irrelevant and erroneous values in a dataset. Examples of irrelevant values are data points or attributes with constant (or near constant), non-changing values. An attribute with serialized values such as row ID numbers or time stamps, where these are not needed for a data mining task at hand, may also be considered as noise. Errors in a dataset can occur in various forms such as extreme and unrealistic values, missing values and suspicious values such as negative values.

Classification of noise is generally case-specific [4], and data modellers need to establish what is noise and what is not noise in the dataset.

Real-life datasets are often noisy [5, 6], to varying degrees such as low, moderate, high and extreme [7]. Table 1 can be used to explain these degrees in a given *m* x *n* dataset matrix, where *m* is the number of attributes or columns and *n* the number of instances or rows. In the Table 1, ν are the valid values while ε represent erroneous values. Attributes A1 and A3 have less than 5% erroneous values and can be considered as having low noise. Attributes A2, A4, A5 and A7 may be termed as moderately noisy, A6 has high noise, while A8 is extremely noisy.

Noise may be inevitable is some datasets, for example, sensors which are the most ubiquitous data source in many data driven decisions, are known to generate erroneous data as a result of mechanical and electronic disturbances [7, 8, 9]. If the system is such that the sensors are very sensitive, then such systems may regularly generate noisy

Table 1 Example dataset showing pattern for erroneous values

Instance #	Attributes							
	A1	A2	A3	A4	A5	A6	A7	A8
1	*v*	*v*	*v*	*v*	*v*	*v*	*v*	*v*
2	*v*	*v*	*v*	*v*	*v*	*v*	*v*	*v*
3	*v*	*v*	*v*	*ε*	*v*	*v*	*v*	*v*
4	*v*	*v*	*v*	*v*	*v*	*v*	*ε*	*ε*
5	*v*	*v*	*v*	*ε*	*ε*	*v*	*v*	*ε*
6	*v*	*v*	*v*	*v*	*v*	*v*	*v*	*ε*
7	*v*	*v*	*v*	*v*	*v*	*v*	*ε*	*ε*
8	*v*	*v*	*v*	*ε*	*v*	*ε*	*v*	*ε*
9	*v*	*v*	*v*	*v*	*ε*	*ε*	*v*	*ε*
10	*v*	*v*	*v*	*v*	*v*	*ε*	*v*	*ε*
11	*v*	*v*	*v*	*ε*	*v*	*ε*	*ε*	*ε*
12	*v*	*v*	*v*	*v*	*v*	*ε*	*v*	*v*
13	*v*	*v*	*v*	*v*	*v*	*ε*	*v*	*ε*
14	*v*	*v*	*v*	*ε*	*v*	*ε*	*ε*	*ε*
15	*v*	*v*	*v*	*v*	*v*	*ε*	*v*	*v*
16	*v*	*v*	*ε*	*v*	*v*	*ε*	*v*	*v*
17	*v*	*v*	*v*	*v*	*v*	*ε*	*v*	*v*
18	*v*	*v*	*v*	*v*	*v*	*ε*	*ε*	*ε*
19	*v*	*v*	*v*	*v*	*v*	*ε*	*ε*	*ε*
20	*v*	*v*	*v*	*v*	*v*	*ε*	*ε*	*ε*
21	*v*	*v*	*v*	*v*	*v*	*ε*	*ε*	*ε*
22	*v*	*v*	*v*	*v*	*v*	*ε*	*v*	*ε*
23	*v*	*v*	*v*	*v*	*v*	*ε*	*v*	*ε*
24	*v*	*v*	*v*	*v*	*v*	*v*	*v*	*ε*
25	*v*	*v*	*v*	*v*	*v*	*v*	*v*	*ε*
26	*v*	*ε*	*v*	*ε*	*v*	*v*	*ε*	*ε*
27	*v*	*ε*	*v*	*v*	*v*	*v*	*v*	*ε*
28	*v*	*ε*	*v*	*ε*	*v*	*v*	*v*	*ε*
29	*v*	*v*	*v*	*v*	*v*	*v*	*v*	*ε*
30	*v*	*v*	*v*	*v*	*v*	*v*	*v*	*ε*
Percentage ε	0%	10%	3%	23%	7%	53%	30%	77%

datasets where mechanical disturbances and electrical fluctuations are regular. Significant noise in a dataset can lead to erroneous statistical inferences which may misinform decision makers. Where noise ranges from moderate to extreme, it can negatively affect the speed and accuracy performance of machine learning algorithms [10]. Robust approaches are need when handling noisy datasets.

Data cleaning is a time consuming and effort demanding task in a machine learning process. Data cleaning can be approached using integrity constraints methods [11], statistical tools [12] and machine learning models [13]. A number of options exist for addressing erroneous values in a multivariate dataset [14]. In one method, data instances with error values can be discarded, but this may significantly reduce the size of the dataset where the dataset is small to medium size and where the noisy instances are substantial. If columns or attributes with erroneous values are discarded [as in 15], decision makers may miss some critical attribute relationships and the dynamics that take place in the system. Additionally, the expunged readings may hold valuable time-related information, and so cannot be excluded for a particular decision. Another method is to replace erroneous values with mean or mode values for the attribute that has the erroneous values, but this may lead to wrong statistical or learning inferences where a significant number of values are replaced. Yet another method is to use a non-noisy surrogate dataset [16], to replace the erroneous values. The surrogate dataset can come from the same system, such as from prior data collection. This may not be an option if the system is prone to generating noisy data (as it is the case with sensors), or if the conditions are different. A fourth way of dealing with erroneous values is to mine them using a machine learning approach, either learning with noise [7] or replacing the erroneous values [13, 17]. Replacing erroneous values enables a reconstructed dataset to evolve. A reconstructed dataset can be of value in other ways aside the prediction of some values.

In the literature, researchers' efforts have been mainly focused on developing and evaluating algorithms and methods that are robust to noise [6, 7, 18–20] or for filtering out noise [21, 22]. There is a paucity of studies that detail the methodological end-to-end process for correcting erroneous values in a dataset. There is also scarcity of studies that describe how process datasets can be de-noised through erroneous values correction. Meanwhile, the heterogeneity of datasets and variety of machine learning goals makes a generalized de-noising method to be infeasible. As such, studies and methods relating to data noise correction are needed to account for the uniqueness of datasets and machine learning goals.

Materials and methods

This research aims to develop a methodology that uses classification prediction modelling to replace erroneous sensors-generated values of multi-stage process datasets. This section details the tools for analysis.

Dataset of multi-stage, multi-output process

In early 2020, a dataset was posted by Liveline Technologies, on Kaggle [3], which hosts open source dataset repositories for different types of datasets representing a variety of systems and problem types. The timeseries dataset was posted to elicit responses from data modellers on how accurately they can predict values into the future. The dataset was found to be expedient in other ways. Firstly, the dataset comes from a real-life multi-stage continuous flow manufacturing process, so mining it would generate real-life insights. Secondly, the system that the dataset represents is typical of many process types, with activities in parallel and in series as well as assembly points. Additionally, it is a time series dataset which typifies the type of data that is streamed and batched in many systems. The results of a machine-learning based modelling approach could be generalizable for many types of systems and processes. The dataset contained a lot of erroneous values. It is possible that the original dataset was perturbed by the dataset providers, but erroneous data entries occur in real-life systems datasets, especially those from sensors. The occurrence may be consequential, whether frequent or infrequent. Documenting the process of correcting a very noisy dataset would be beneficial to decision makers in systems who are at any time faced with dealing with such datasets.

The dataset, the description of the data sampling rate and the background information about the system are available at Kaggle.com [3]. Based on the information disclosed by the dataset providers, the system is a two-stage continuous flow manufacturing process, which can be depicted as shown in Fig. 1 [23]. Outputs from each stage are measured in 15 locations each, and they are the primary measurements to correct. The raw dataset (in csv format) consisted of 116 variables (column labels, see Appendixs 1, 2, 3, 4) and 14,088 observations (row data). The 14,088 observations are based on time stamps in 1second interval. As imposed by the dataset providers, the key variables of concern are the 15 stage 1 output measurements and 15 stage 2 output measurements (see Fig. 1). These measurements were given in mm, suggesting that they are dimensional features.

The dataset owners provided information that the dataset was very noisy (see Fig. 2 excerpts for time series plots of some of the variables). It was observed, from the raw dataset, that there is value variation in some key process output features, where the values are required to be fixed (see example charts in Fig. 2a, b). This is an indication that the values may be noisy. Dataset providers confirmed this. A lot of the values are also zero, suggesting they are inaccurate readings, since dimensional features must take on positive values. So, the main challenge was to de-noise and correct the dataset.

Feature selection modelling

A feature (or attribute) is an individual measurable property of the system that is being investigated [24]. In a dataset, it is referenced using the column labels. The focus of feature selection is to select a subset of variables from the input which can efficiently

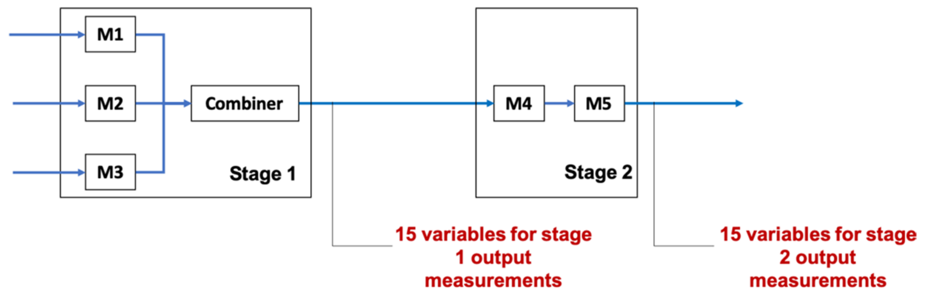

M1
M2
M3
Combiner
Stage 1
M4
M5
Stage 2
15 variables for stage 1 output measurements
15 variables for stage 2 output measurements

Fig. 1 Process flow representing the multi-stage continuous flow production system

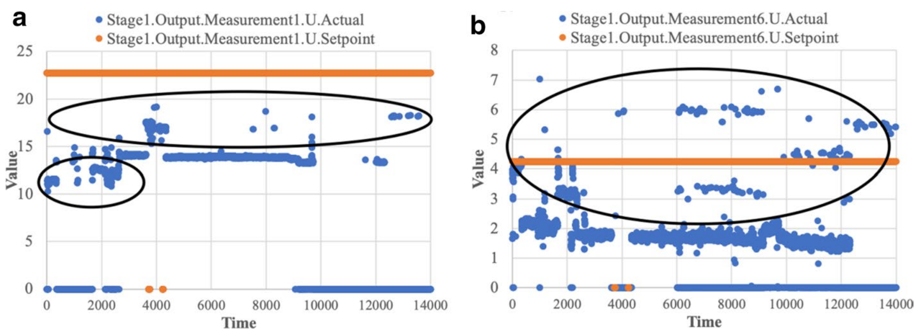


Fig. 2 Time series plot: (**a**) stage 1 output measurement at location 1; (**b**) stage 1 output measurement at location 6

describe the input data while reducing effects from noise or irrelevant variables and still provide good prediction results. Feature selection is the systematic reduction of the feature space of a dataset. The key objective is to improve the accuracy and cost of the learning model [7, 25]. In feature selection, only those features (variables or attributes) that show high sensitivity to the target output are retained [26]. Others are considered as noise or irrelevant to a given machine learning task [27]. Other benefits include reducing the dataset size for storage [28] and to pinpoint key input variables that affect an output variable [29]. Feature selection or reduction allows the application of more complex algorithms that would otherwise be infeasible where the full dataset is used [30].

Examples of feature selection algorithms that have been widely used within a manufacturing system context include Principal Component Analysis [31], ReliefF [32] and Correlation Based Filter Selection [33]. These feature selection algorithms have their specific uses and scope, benefits and limitations and types of dataset they can handle. For instance, Principal Component Analysis is very useful for extracting relevant features from high dimensional data such as image data [34], while Correlation Based Filter Selection works well on numeric data [35]. ReliefF and other tree-based algorithms are able to handle noise better than others [36].

In its simplest form, a feature selection workflow can be described as shown in Fig. 3. The workflow describes the use of a filter-based algorithm that uses a ranking scheme to rank features from most relevant to least relevant, in a supervised manner.

As shown in the Fig. 3, the dataset containing all features is evaluated using candidate feature reduction algorithms. The subset is then evaluated using candidate machine learning models [24, 37] or one that has been chosen previously. For ranking-based feature selection approach, different cut-off points for the ranked features are chosen and evaluated to identify the optimal cut-off point beyond which learning performance becomes marginal. Selected features are not generalizable for the entire dataset. They are specific to the target or output attribute, and so in a multi-target attribute dataset, different feature sets need to be extracted for each target [38]. In other words, the process (Fig. 3) is repeated as many times as a target attribute is to be predicted.

There is a trade-off between reduced feature space dimensionality and prediction accuracy [15]. Data modellers have to choose whether to proceed with the full set or a reduced set, depending on the prediction accuracy of using the reduced set in comparison to using the full set. In a very large dataset with thousands of features, it is good

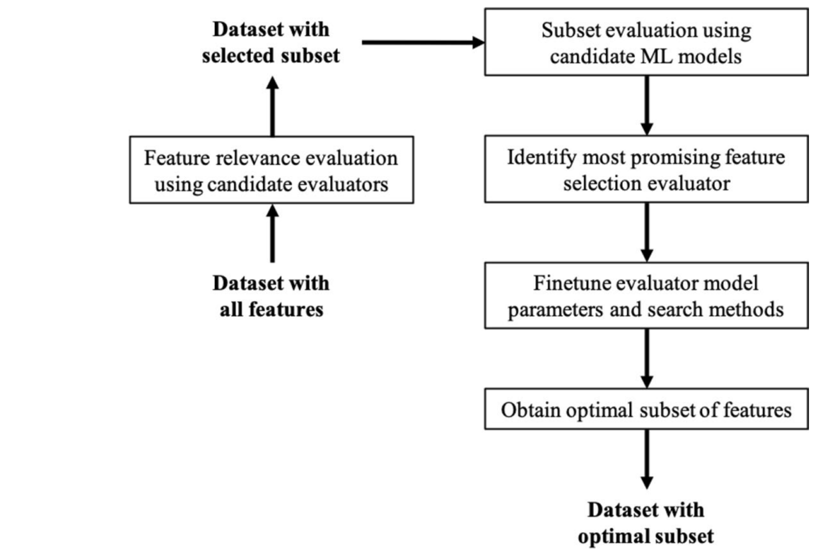

Dataset with selected subset
Subset evaluation using candidate ML models
Feature relevance evaluation using candidate evaluators
Identify most promising feature selection evaluator
Dataset with all features
Finetune evaluator model parameters and search methods
Obtain optimal subset of features
Dataset with optimal subset

Fig. 3 Workflow for a feature selection process

practice to exclude non-informative and irrelevant features first [39] as this can enhance the feature selection process. Other pre-processing tasks that enhance feature selection modelling include using clean instances of the dataset [40] and data type transformation, such as normalization, discretization and nominalization, as would normally be fulfilled prior to classification, clustering and association modelling, for example.

ReliefF ranks features from most sensitive to least sensitive to the class attribute. Data modellers can then decide which cut off point to test on the classification model. For this reason, ReliefF feature selection algorithm was chosen. Principal Component Analysis was ruled out as it is computationally costlier. Additionally, it uses orthogonal transformation to map features that may be correlated into a new feature space where there is minimum correlation between the new features, thereby creating new features from existing ones [15]. The new features diminish the physical meaning of the original features, and requires additional modelling and computation to understand what the new features represent. Correlation Based Filter Selection was also considered, but this does not give a ranking of how features are important in explaining a class attribute.

Classification modelling

Classification prediction remains the most utilized machine learning technique [37], ahead of clustering and association rules, even in the context of manufacturing systems [41]. Classification prediction involves the process of using observations of a known group to estimate observations of an unknown group [8, 42], on the basis of the least estimation error [43] and modelling cost [44]. Accuracy of prediction or estimation error is based on measurement error (or distance) between the predicted and observed actual value. Cost of modelling is a function of the total time to train and test the model.

Classification prediction, as it is with other machine learning methods, is fulfilled in two main stages [45]. In the first stage, the classifier model (algorithm and specified parameters), is built by training it to memorize observations and patterns of a portion (train set) of the known group. In the second stage, the trained model is validated on the other portion of the known group (test set) to estimate the accuracy rate when using the classifier model on an unknown group. The use of the classifier model on an unknown group is based on the known group and unknown group being homogenous in terms of identical number and type of attributes, and representing the same system.

Neural Networks, Linear Regression, Support Vector Machines and Tree-based Classifiers, with their variants, have been the most prevalent learning algorithms in use in manufacturing [46]. These classifier learning algorithms handle data differently [46], deal with noise differently [47], and respond differently to data types. For instance, tree-based classifiers work well with nominal data types, they are also more responsive to certain data types than with others. For example, Neural Networks and Linear Regression models are well suited for numeric data types; J48 learner accepts only nominal-type data (such as discrete and categorical), and Random Forest works is able to handle both numeric and nominal data types. Tree-based classifiers can handle small to medium sized datasets very well [48, 49]. The tree-based classifiers such as Decision Tree and Random Forest have been the dominant classifiers in machine learning-based noise correction tasks. Some algorithms, such as Neural Networks, Logistic Regression and Support Vector Machine, perform badly when data is not normalized and where scale differences exist amongst features. The tree-based classifiers are rule-based, and so do not require pre-processing through data normalization. Having foreknowledge of these information helps data modellers narrow down the candidate classifier algorithms to test on a dataset, and not test the full spectrum.

In data modelling with classification learning, it is common to use a boosting method such as an ensemble of multiple classifiers [8] in order to improve prediction accuracy. Random Forest classifier is an ensemble of tree-based classification models, and is known to be more accurate than J48 and Decision Tree [5]. Feature reduction has also been used to boost the performance of classifier models.

In choosing the modelling algorithm to use in building a classification prediction model, it is common to test different algorithms [7, 8, 15] and choose the one with the most promising performance (prediction accuracy and training/testing speed).

Software tools for building machine learning models

Identification and selection of software tools is rarely reported in detail [50]. Familiarity with the software, application domain of the software, versatility of the software in terms of data types and formats and learning algorithms handled, ease of communicating with other software and other factors are used to identify and select candidate machine learning software tools [50]. Software tools with Graphical User Interface (GUI) such as KNIME and WEKA have been used extensively in data driven decisions relating to manufacturing systems [50]. These and other GUIs enable machine learning models to be built quickly with little or no programming code input. One major limitation of these GUIs is that programming codes may be needed for data transmission between the software and the database that warehouses the dataset. Additionally, they individually have their specific limitations. It is therefore possible that one may need to rely on codes written in C, Python, Java or R language, to automate some tasks such as data transmission and text extraction.

The use of a relational database is inevitable when learning on batched data, since batched data are by nature stored data. SQL and other relational databases are common in manufacturing [51]. These databases provide the necessary structure for storing and organizing data to facilitate quick and easy access and manipulation. Microsoft Excel has also proven useful [50]. Relational databases have the type of structure (columns and rows) that is responsive to queries and manipulation using programming codes. In the current study, WEKA was used for the machine learning tasks. Python was used to generate computer codes to enable data pre-processing, data parsing and data extraction of relevant information. Microsoft Excel was used as the database and for opening csv data file formats.

Methodology for correcting noise in multiple attributes in a dataset.

The methodology that was developed is depicted in Fig. 4. It is based on swapping noisy erroneous values with missing values. Machine learning algorithms treat missing values as values to be skipped during learning, whereas noisy values (which are erroneous) may be included in learning, unless otherwise specified. The methodology proposes to replace erroneous values with missing values and then impute most probable values in the missing values. The most probable values are predicted using a classification prediction model. The classification prediction model is built and trained using the clean (not missing) instances of the dataset. The trained model is then used to predict values for the missing instances.

The first three steps represent the data exploration and pre-processing of the data, including data transformation. The subsequent seven steps are the prediction learning steps. In the methodology, prediction learning is fulfilled one attribute at a time. In a situation where there are multiple attributes to be corrected, these steps are repeated for each attribute that is noisy. The final step involves replacing the missing values with the predicted values, which culminates in the corrected dataset.

In the developed methodology, classification was used. A clustering or association mining approach can also be used. In order to reduce the feature space of the dataset and boost the performance of the classification model, feature reduction modelling was applied. Classification modelling would therefore serve two purposes to: (a) validate the ability of few selected features to represent the entire feature space and (b) to predict replacement values for erroneous values. For the above stated reasons, the prediction learning aspect is majorly based on feature selection and classification prediction modelling methods.

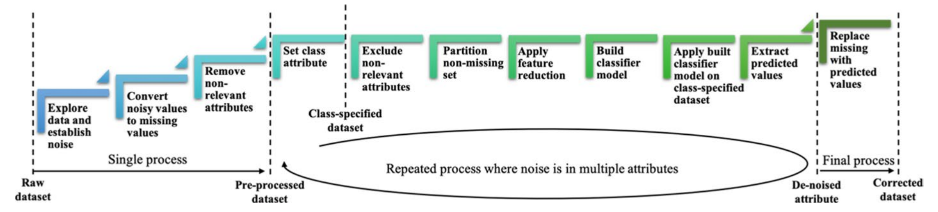

Explore data and establish noise
Convert noisy values to missing values
Remove non-relevant attributes
Set class attribute
Exclude non-relevant attributes
Partition non-missing set
Apply feature reduction
Build classifier model
Apply built classifier model on class-specified dataset
Extract predicted values
Replace missing with predicted values
Class-specified dataset
Single process
Repeated process where noise is in multiple attributes
Final process
Raw dataset
Pre-processed dataset
De-noised attribute
Corrected dataset

Fig. 4 Flow process for correcting noise in multiple attributes in a dataset

The methodology that is depicted in Fig. 4 was developed while correcting the noisy dataset described in "Dataset de-noising and erroneous values correction" section. It was developed through an iterative process that tested other possible pipelines [50]. In the next section, the developed methodology is re-applied to the same noisy dataset to explicate it as a reusable approach.

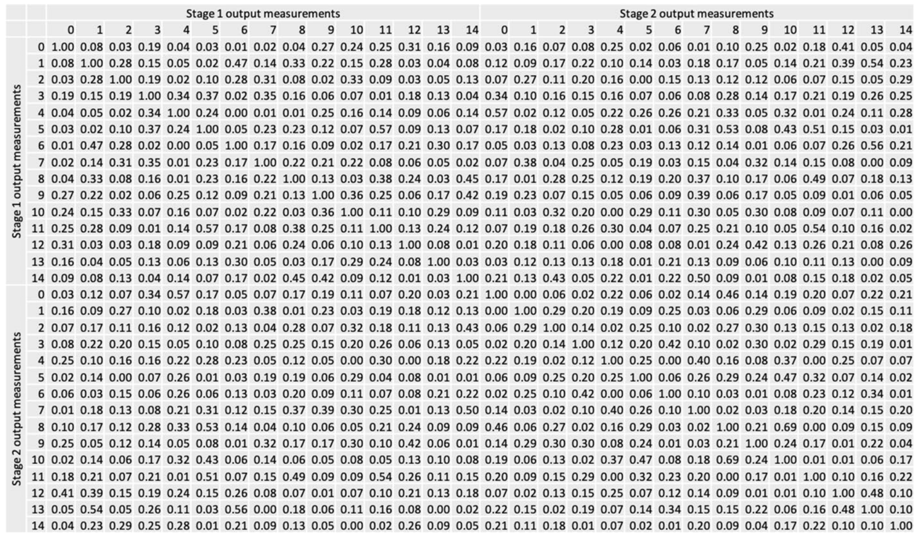
Results and analysis

The dataset used in this study contained a significant amount of erroneous values in those attributes representing output variables. The objective was to denoise the dataset by replacing the erroneous values with values predicted using classification modelling. The results presented here are the steps and outcomes from adopting the methodology shown in Fig. 4

Explore data and establish noise

Data exploration was fulfilled using Panda, a data analysis tool for python programming. Time series plots of attributes were analysed for trends and anomalies. Many of the stage 1 and stage 2 output measurements exhibited trends similar to those shown in Fig. 2. In the dataset, noisy values were identified as values outside of a user-specified range, including zeros and outliers.

A correlation matrix was generated to determine if attributes were strongly, moderately or weakly correlated. Strong correlations are an indication that one in a pair of attributes may be redundant. It is also an indication that statistical tools can be used to explain pairwise relationships. For example, if stage 1 measurement output and stage 2 measurement output are highly correlated, then either can be used to explain the other, through simple statistical expressions. The correlation matrix however indicated mainly weak correlations, see correlation coefficients shown in matrix of Fig. 5.



		Stage 1 output measurements															Stage 2 output measurements														
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Stage 1 output measurements	0	1.00	0.08	0.03	0.19	0.04	0.03	0.01	0.02	0.04	0.27	0.24	0.25	0.31	0.16	0.09	0.03	0.16	0.07	0.08	0.25	0.02	0.06	0.01	0.10	0.25	0.02	0.18	0.41	0.05	0.04
	1	0.08	1.00	0.28	0.15	0.05	0.02	0.47	0.14	0.33	0.22	0.15	0.28	0.03	0.04	0.08	0.12	0.09	0.17	0.22	0.10	0.14	0.03	0.18	0.17	0.05	0.14	0.21	0.39	0.54	0.23
	2	0.03	0.28	1.00	0.19	0.02	0.10	0.28	0.31	0.08	0.02	0.33	0.09	0.03	0.05	0.13	0.07	0.27	0.11	0.20	0.16	0.00	0.15	0.13	0.12	0.12	0.06	0.07	0.15	0.05	0.29
	3	0.19	0.15	0.19	1.00	0.34	0.37	0.02	0.35	0.16	0.06	0.07	0.01	0.18	0.13	0.04	0.34	0.10	0.16	0.15	0.16	0.07	0.06	0.08	0.28	0.14	0.17	0.21	0.19	0.26	0.25
	4	0.04	0.05	0.02	0.34	1.00	0.24	0.00	0.01	0.01	0.25	0.16	0.14	0.09	0.06	0.14	0.57	0.02	0.12	0.05	0.22	0.26	0.26	0.21	0.33	0.05	0.32	0.01	0.24	0.11	0.28
	5	0.03	0.02	0.10	0.37	0.24	1.00	0.05	0.23	0.23	0.12	0.07	0.57	0.09	0.13	0.07	0.17	0.18	0.02	0.10	0.28	0.01	0.06	0.31	0.53	0.08	0.43	0.51	0.15	0.03	0.01
	6	0.01	0.47	0.28	0.02	0.00	0.05	1.00	0.17	0.16	0.09	0.02	0.17	0.21	0.30	0.17	0.05	0.03	0.13	0.08	0.23	0.03	0.13	0.12	0.14	0.01	0.06	0.07	0.26	0.56	0.21
	7	0.02	0.14	0.31	0.35	0.01	0.23	0.17	1.00	0.22	0.21	0.22	0.08	0.06	0.05	0.02	0.07	0.38	0.04	0.25	0.05	0.19	0.03	0.15	0.04	0.32	0.14	0.15	0.08	0.00	0.09
	8	0.04	0.33	0.08	0.16	0.01	0.23	0.16	0.22	1.00	0.13	0.03	0.38	0.24	0.03	0.45	0.17	0.01	0.28	0.25	0.12	0.19	0.20	0.37	0.10	0.17	0.06	0.49	0.07	0.18	0.13
	9	0.27	0.22	0.02	0.06	0.25	0.12	0.09	0.21	0.13	1.00	0.36	0.25	0.06	0.17	0.42	0.19	0.23	0.07	0.15	0.05	0.06	0.09	0.39	0.06	0.17	0.05	0.09	0.01	0.06	0.05
	10	0.24	0.15	0.33	0.07	0.16	0.07	0.02	0.22	0.03	0.36	1.00	0.11	0.10	0.29	0.09	0.11	0.03	0.32	0.20	0.00	0.29	0.11	0.30	0.05	0.30	0.08	0.09	0.07	0.11	0.00
	11	0.25	0.28	0.09	0.01	0.14	0.57	0.17	0.08	0.38	0.25	0.11	1.00	0.13	0.24	0.12	0.07	0.19	0.18	0.26	0.30	0.04	0.07	0.25	0.21	0.10	0.05	0.54	0.10	0.16	0.02
	12	0.31	0.03	0.03	0.18	0.09	0.09	0.21	0.06	0.24	0.06	0.10	0.13	1.00	0.08	0.01	0.20	0.18	0.11	0.06	0.00	0.08	0.08	0.01	0.24	0.42	0.13	0.26	0.21	0.08	0.26
	13	0.16	0.04	0.05	0.13	0.06	0.13	0.30	0.05	0.03	0.17	0.29	0.24	0.08	1.00	0.03	0.03	0.12	0.13	0.13	0.18	0.01	0.21	0.13	0.09	0.06	0.10	0.11	0.13	0.00	0.09
	14	0.09	0.08	0.13	0.04	0.14	0.07	0.17	0.02	0.45	0.42	0.09	0.12	0.01	0.03	1.00	0.21	0.13	0.43	0.05	0.22	0.01	0.22	0.50	0.09	0.01	0.08	0.15	0.18	0.02	0.05
Stage 2 output measurements	0	0.03	0.12	0.07	0.34	0.57	0.17	0.05	0.07	0.17	0.19	0.11	0.07	0.20	0.03	0.21	1.00	0.00	0.06	0.02	0.22	0.06	0.02	0.14	0.46	0.14	0.19	0.20	0.07	0.22	0.21
	1	0.16	0.09	0.27	0.10	0.02	0.18	0.03	0.38	0.01	0.23	0.03	0.19	0.18	0.12	0.13	0.00	1.00	0.29	0.20	0.19	0.09	0.25	0.03	0.06	0.29	0.06	0.09	0.02	0.15	0.11
	2	0.07	0.17	0.11	0.16	0.12	0.02	0.13	0.04	0.28	0.07	0.32	0.18	0.11	0.13	0.43	0.06	0.29	1.00	0.14	0.02	0.25	0.10	0.02	0.27	0.30	0.13	0.15	0.13	0.02	0.18
	3	0.08	0.22	0.20	0.15	0.05	0.10	0.08	0.25	0.25	0.15	0.20	0.26	0.06	0.13	0.05	0.02	0.20	0.14	1.00	0.12	0.20	0.42	0.10	0.02	0.30	0.02	0.29	0.15	0.19	0.01
	4	0.25	0.10	0.16	0.16	0.22	0.28	0.23	0.05	0.12	0.05	0.00	0.30	0.00	0.18	0.22	0.22	0.19	0.02	0.12	1.00	0.25	0.00	0.40	0.16	0.08	0.37	0.00	0.25	0.07	0.07
	5	0.02	0.14	0.00	0.07	0.26	0.01	0.03	0.19	0.19	0.06	0.29	0.04	0.08	0.01	0.01	0.06	0.09	0.25	0.20	0.25	1.00	0.06	0.26	0.29	0.24	0.47	0.32	0.07	0.14	0.02
	6	0.06	0.03	0.15	0.06	0.26	0.06	0.13	0.03	0.20	0.09	0.11	0.07	0.08	0.21	0.22	0.02	0.25	0.10	0.42	0.00	0.06	1.00	0.10	0.03	0.01	0.08	0.23	0.12	0.34	0.01
	7	0.01	0.18	0.13	0.08	0.21	0.31	0.12	0.15	0.37	0.39	0.30	0.25	0.01	0.13	0.50	0.14	0.03	0.02	0.10	0.40	0.26	0.10	1.00	0.02	0.03	0.18	0.20	0.14	0.15	0.20
	8	0.10	0.17	0.12	0.28	0.33	0.53	0.14	0.04	0.10	0.06	0.05	0.21	0.24	0.09	0.09	0.46	0.06	0.27	0.02	0.16	0.29	0.03	0.02	1.00	0.21	0.69	0.00	0.09	0.15	0.09
	9	0.25	0.05	0.12	0.14	0.05	0.08	0.01	0.32	0.17	0.17	0.30	0.10	0.42	0.06	0.01	0.14	0.29	0.30	0.30	0.08	0.24	0.01	0.03	0.21	1.00	0.24	0.17	0.01	0.22	0.04
	10	0.02	0.14	0.06	0.17	0.32	0.43	0.06	0.14	0.06	0.05	0.08	0.05	0.13	0.10	0.08	0.19	0.06	0.13	0.02	0.37	0.47	0.08	0.18	0.69	0.24	1.00	0.01	0.01	0.06	0.17
	11	0.18	0.21	0.07	0.21	0.01	0.51	0.07	0.15	0.49	0.09	0.09	0.54	0.26	0.11	0.15	0.20	0.09	0.15	0.29	0.00	0.32	0.23	0.20	0.00	0.17	0.01	1.00	0.10	0.16	0.22
	12	0.41	0.39	0.15	0.19	0.24	0.15	0.26	0.08	0.07	0.01	0.07	0.10	0.21	0.13	0.18	0.07	0.02	0.13	0.15	0.25	0.07	0.12	0.14	0.09	0.01	0.01	0.10	1.00	0.48	0.10
	13	0.05	0.54	0.05	0.26	0.11	0.03	0.56	0.00	0.18	0.06	0.11	0.16	0.08	0.00	0.02	0.22	0.15	0.02	0.19	0.07	0.14	0.34	0.15	0.15	0.22	0.06	0.16	0.48	1.00	0.10
	14	0.04	0.23	0.29	0.25	0.28	0.01	0.21	0.09	0.13	0.05	0.00	0.02	0.26	0.09	0.05	0.21	0.11	0.18	0.01	0.07	0.02	0.01	0.20	0.09	0.04	0.17	0.22	0.10	0.10	1.00

Fig. 5 Matrix plot showing correlation coefficients between the output measurements

Table 2 Attributes in the dataset with their percentage missing values

Output measurement at location	% missing for stage 1	% missing for stage 2
0	0.5	9
1	58	41
2	3	12
3	1	6
4	1	91
5	95	1
6	88	1
7	62	7
8	6	7
9	5	30
10	2	5
11	74	4
12	23	6
13	14	3
14	57	15

Convert noisy values to missing values

Sensor readings are erroneous due to vibrations and interferences on the sensing surface of the output feature. User-specified range of true values was used to sift out erroneous sensor readings falling outside of the range. Values outside the range were converted to 0 values, while those values within true range were retained. Then zero values were then converted to missing (NaN) values. Focus was on those stage 1 and stage 2 output measurements that exhibited significant erroneous values. The conversions were performed using Pandas and NumPy python-based libraries and codes (see Algorithm 1). Table 2 shows the attributes with missing values.

Algorithm 1. Python-based code to convert error readings to missing values for Stage 1 output measurement 1 Actual

```
#import libraries
import numpy
import pandas

#open csv file containing the dataset
df = pd.read_csv(“file pathname of where data is stored”)

#replace error readings with 0 (values between 14 and 16 are considered true values)
df['Stage1.Output.Measurement1.U.Actual'] =
np.where(df['Stage1.Output.Measurement1.U.Actual'] < 14, 0,
df['Stage1.Output.Measurement1.U.Actual'])

df['Stage1.Output.Measurement1.U.Actual'] =
np.where(df['Stage1.Output.Measurement1.U.Actual'] > 16, 0,
df['Stage1.Output.Measurement1.U.Actual'])

#convert 0 values to missing values
df= df.replace(0.00, np.nan)
```

Remove non-relevant attributes

Some attributes do not add value to a learning task. When included in the dataset for learning, it hinders the model performance. For the learning task at hand, the time stamp attribute was not needed. In addition, the set point measurements in the dataset were also not needed, since they were constant, non-changing values. If the correlation analysis had established correlation between any two attributes, one can be discarded as being redundant.

Attributes with significant missing values can also be discarded. Attributes with more than 70% missing values fall into this category. The reasoning behind this is because the non-missing which would have been used to predict the missing values is insufficient to comprehensively describe the dataset pattern for the entire set of values for the attribute. Three attributes in stage 1 output measurements (5, 6 and 11) and one in stage 2 output measurements (4) fell into this category.

Set class attribute

A class attribute can be described as the particular attribute whose values are to be predicted, also known as the target, dependent or output variable. In the dataset being worked on, there are multiple class attributes since there are multiple attributes whose values are noisy, something that is common in real life datasets. This is a classic multi-target classification problem, where many attributes are considered as the target. There are two known approaches for treating this type of multi-class classification problem [52]. One is to build a separate model for each target. The second approach is to use a multi-target classification algorithm [53], such as Random Linear Target Combination and Multi-Objective Random Forest [54]. Using single-target classifiers for a dataset that has many class attributes would be an arduous and time-consuming task, if one is to build a separate model for each class attribute. Despite this, they perform just as well [52] and even sometimes better [54] than multi-target classifiers. Moreover, single-target classification approach is more prevalent and more straightforward [52]. For the few numbers of attributes that are to be corrected in the current dataset, a single-target classification approach is sufficient.

In this research, the prediction learning is on the basis of a single-target classification approach, hence steps four to ten (see Fig. 4) is done repeatedly for each of the class attribute that is to be corrected for noise. The approach is a systematic one where attributes with erroneous values are corrected, one attribute at a time [55]

Exclude non-relevant attributes

The dataset being corrected represents a multi-stage manufacturing process. Specifically, the system is a two-stage continuous flow manufacturing process. It is better to split the class-specified dataset into relevant and not-relevant sets to improve learning performance [56]. With reference to Fig. 1, relevant to predicting stage 1 output measurements are those variables upstream of stage 1 output measures. Stage 2 processes and stage 2 output measures are not relevant, since they are downstream of stage 1 output measurements. Similarly, stage 1 output measurements and stage 2 processes are relevant for predicting stage 2 output measurements. Including stage

1 process parameters in the set for predicting stage 2 output measurements will be a redundancy, because stage 1 output measures are the results of stage 1 process parameters.

Bearing this in mind a dataset is carved out of the class-specified dataset. The generated dataset includes the class attribute as well as those attributes that represent process parameters upstream of the attribute. As an example, if stage 1 output measurement 2 (S1M2) has been chosen as the class attribute, then the dataset is pruned to exclude all other stage 1 output measurements and stage 2 process attributes with stage 2 output measurements. On the other hand, if stage 2 output measurement 7 (S2M7) is selected as the class attribute, the dataset includes the stage 1 output measurements and stage 2 attributes, but excludes other stage 2 output measurement attributes. The stage 1 output measurements should have been corrected prior to this, as noisy attributes can cause prediction errors. It is possible to use only the stage 2 process attributes (representing machines 4 and 5) to predict the stage 2 output measurement values, but by including stage 1 output measurement attributes, it is assumed that stage 2 output is formed from stage 1 output. In other words, stage 1 output measurements can be used to explain stage 2 output measurements. If a stage 3 process existed, the same analogy is applied, and so on in subsequent stages. To eliminate confusion, it is best that the dataset attributes are ordered is such a way that the ordering represents the actual flow process.

Partition non-missing dataset

Although tree-based algorithms such as Decision Tree and Random Forest have been known to learn noisy datasets very well [47], performance is often degraded where noise exists in a dataset. As a result, the methodology proposes using a clean (not missing values) dataset to build and test the prediction model. In order to do this, a dataset that has non-erroneous (non-missing) values for the class attribute is carved out of the class-specified dataset (which has both missing and non-missing values). This dataset is progressed for building the classifier model for predicting the class attribute. The built model would be applied to the class-specified dataset to predict values for the missing values. The data partitioning was accomplished in the WEKA explorer environment.

Apply feature reduction modelling

In the methodology, the main function of feature reduction is to boost the performance of the classifier model. To confirm that the few selected features can adequately represent the entire feature space, a classifier model is applied to the feature-reduced dataset (as described in Fig. 3). The performance is compared with learning using the full feature space. If prediction accuracy is not degraded significantly but modelling cost is improved, then the selected features are capable of representing the entire feature space, otherwise, the full dataset is used.

For the dataset being corrected, a feature reduction model was built on the basis of ReliefF feature selection algorithm [57, 58]. In the current research, subsets of the dataset made up of the top ranked 5, 10, 15 and 20 ReliefF selected attributes were assessed

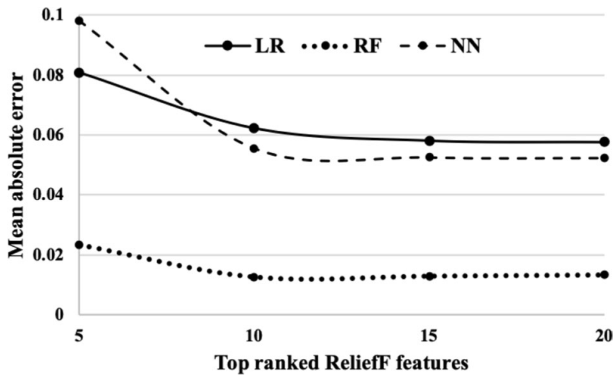


Fig. 6 Prediction accuracy performance comparison of top ranked features for predicting stage 1 output measurement 1

Table 3 Comparison of prediction accuracy on the dataset and feature selection-based dataset

Dataset	MAE	% improvement
Full dataset	0.0421	–
Top 10 ReliefF (missing and non-missing)	0.0363	13.78%
Top 10 ReliefF using non-missing set	0.0355	2.20%

using models built with Logistic Regression (LR), Random Forest (RF), and Neural Networks (NN). The results (see Fig. 6) indicate the optimal number to be the first 10 ReliefF selected attributes, beyond which there is marginal or no improvement in prediction accuracy.

Table 3 has been used to compare a Random Forest classifier model prediction accuracy using: (a) the full dataset; (b) the top 10 ReliefF (missing and non-missing) and (c) the top 10 ReliefF using non-missing set only. From the results presented in Table 3, it can be concluded that there is improvement in the prediction accuracy when using the ReliefF reduced subset instead of using the full dataset. There is only very marginal improvement of 2.2% by using the non-missing (clean) subset instead of using the subset that contains both missing and non-missing values for the class attribute. For the current dataset problem, there is therefore no need to set apart a clean subset to be used in building the classifier model, as this would add to the computational steps and time. The partitioning step 6 in the methodology (see Fig. 4) is therefore bypassed for the current dataset, but would be needed for a dataset where missing values significantly degrade the performance of the built classifier model.

The results indicate that the top 10 ReliefF selected attributes are able to represent the entire feature set for predicting S1M1, and are also able to boost the performance of the classifier model. The prediction results for the three candidate classifiers (see Fig. 4), suggest that a Random Forest-based classifier model is the best learner for the dataset. On this basis, the classification prediction is progressed on a hybrid modelling approach namely, ReliefF to reduce the feature space and Random Forest for value prediction.

Build classifier model

A Random Forest classifier model was used to evaluate the feature selection subset. Random forest algorithm is a combination of tree-based classifiers, such that after a large number of trees is generated, they vote for the most popular class [48]. Random Forest by nature is an ensemble i.e. consisting of multiple classifiers (tree models), the accuracy of which has been shown to be quite high for small to medium sized datasets [59]. The drawbacks for Random Forest are speed deficiencies as a result of combining multiple classifiers. In the current research the dataset is considered small to medium sized and so speed deficiency was not noticed. Moreover, the feature space has been reduced through feature reduction, which is supposed to boost the speed performance of the classifier model. Another drawback with Random Forest, like other tree-based algorithms, is that they cannot provide good estimations outside the boundaries of the training dataset, in other words they extrapolate poorly, unlike regression algorithms [59]. The current prediction task does not warrant extrapolating the prediction forecasts outside the boundaries of the current dataset, so a tree-based classifier is sufficient for the present purpose.

The dataset used for training is the top 10 ReliefF reduced dataset which included missing and non-missing values for the class attribute: an 11-attribute $\times$ 14,074 instance dataset. The dataset instances were partitioned 80:20 into train and test sets, using random sampling. Mean absolute error was used as the evaluation metric. It is a prediction accuracy metric for classification models that learn numeric data types. It is indicative of the error variance between the predicted values and the actual values. It is given by the Eq. 1, where e_i is the prediction error of the ith sample, and n is the number of samples.

$$Mean\ absolute\ error = \frac{\sum_i^n |e_i|}{n} \tag{1}$$

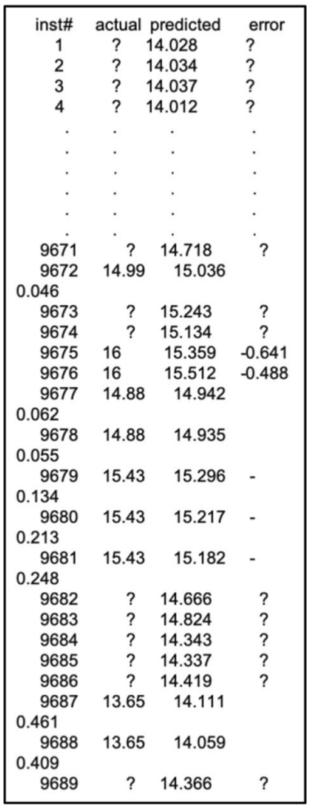
Values of mean absolute error closer to 0 show good prediction capability of the model. Mean absolute error for the Random Forest classifier model for predicting Stage 1 output measurement 1 (S1M1) values, using the top 10 ReliefF selected attributes, was 0.0363, see Table 3. The model is saved to be applied for missing value imputation.

Apply built classifier model on class-specified dataset

The saved model is re-applied to the same dataset, but this time without partitioning the dataset instances. The reason for this is so that the dataset instances are not disordered as is the case with random sampling and partitioning. Re-ordering a disordered dataset after generating prediction results, would add to the computational complexity of the machine learning process.

An excerpt of the prediction results (instance 1 to 4 and 9671 to 9689) is shown in Fig. 7. Under the column with actual values, '?' denotes a missing value. These results are saved to a text file to enable extraction of relevant information.

Figure 8a is a plot of the prediction error for predicting S1M1. The points lying along the 0 axis are majorly the missing entries for the error values due to missing actual values. An analysis of the prediction error for the actual values showed that 99% of values lie below 0.05 mm and 56% below 0.01 mm (see Table 4). The results indicate that if 100 samples are taken, 99 can be predicted to an accuracy that is within 0.05 mm, using the



```
    inst#   actual  predicted    error
      1       ?    14.028        ?
      2       ?    14.034        ?
      3       ?    14.037        ?
      4       ?    14.012        ?
      .       .      .           .
      .       .      .           .
      .       .      .           .
      .       .      .           .
      .       .      .           .
      .       .      .           .
   9671       ?    14.718        ?
   9672   14.99    15.036
0.046
   9673       ?    15.243        ?
   9674       ?    15.134        ?
   9675   16        15.359   -0.641
   9676   16        15.512   -0.488
   9677   14.88    14.942
0.062
   9678   14.88    14.935
0.055
   9679   15.43    15.296    -
0.134
   9680   15.43    15.217    -
0.213
   9681   15.43    15.182    -
0.248
   9682       ?    14.666        ?
   9683       ?    14.824        ?
   9684       ?    14.343        ?
   9685       ?    14.337        ?
   9686       ?    14.419        ?
   9687   13.65    14.111
0.461
   9688   13.65    14.059
0.409
   9689       ?    14.366        ?
```

Fig. 7 Excerpt of prediction results for stage 1 output measurement 1

model. Figure 8b shows the scatter plot for predicted vs actual values. Most data points lie along the line of best fit, indicating good prediction accuracy of the model.

Figure 8a reveals that some prediction errors are significant, values above 0.05, and they are about 1% of samples. From the plot, most of this type of errors occur where there is a significant amount of missing values. These results can be taken to imply that prediction accuracy drops significantly around the vicinity of erroneous or missing values. Prediction accuracy to ± 0.6 mm may be good or bad depending on the situational requirements.

Extract predicted values

A python-based code was used to extract and organize the relevant information from the WEKA generated result. The logic in the code created a csv file for the results. Where the actual column value is '?' the predicted value is selected, otherwise the actual value is selected. By so doing, the column in the generated csv data contains the actual values and the predicted values for the class attribute. The code is shown in Algorithm 2. The code truncates information (by extracting only relevant column values) and parses information (from text to csv). The csv data is saved as a 1

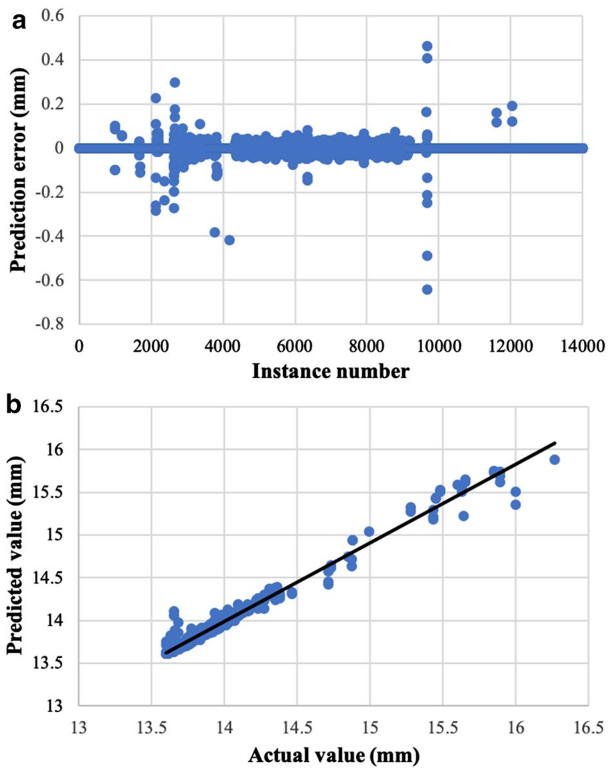


Fig. 8 Plot for stage 1 output measurement: (**a**) prediction error for each instance (**b**) scatter diagram for predicted vs actual values

Table 4 Prediction accuracy summary

Samples with prediction errors	Percentage of values
0.01 and below	56
0.03 and below	94
0.05 and below	99

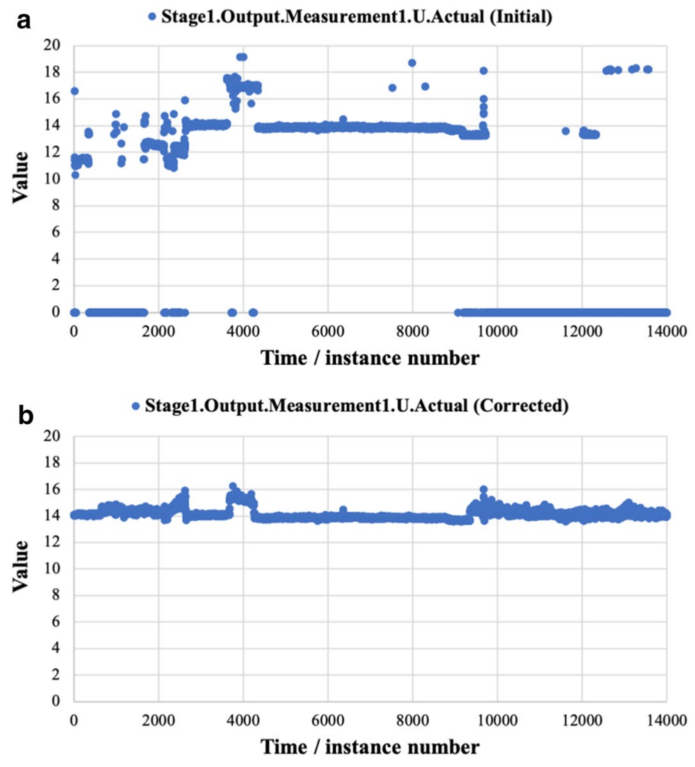


Fig. 9 Time series plot comparing initial dataset with corrected dataset

column × 14,074 instances dataset with missing values replaced with predicted values. The column represents the corrected (non-missing, non-erroneous) data for

Algorithm 2. Python-based code to extract and save relevant information from results buffer.

```
import csv
fhand = open ('S1M1.txt') # open saved text results of predicted values
for line in fhand :
    word = line.split() # split words on each line
    if word [1] == '?' : # if first indexed word is '?'
        value = (word [2]) # set value to second indexed word
    else :
        value = (word [1]) # otherwise, set value to first indexed word
    with open ('S1M1_predicted.csv', 'a', newline='') as f :
        thewriter = csv.writer(f)
        thewriter.writerow([value]) # append each word (value) to a row
```

S1M1.

Replace missing with predicted values

Being a medium-sized dataset, Microsoft Excel was sufficient for replacing the column (S1M1) of missing values with the column of non-missing (corrected) values. Figure 9 plot has been used to show the dataset for attribute S1M1 before and after

Table 5 Summary of number of erroneous values in the dataset, before and after correction

Before correction	After correction	
	Including attributes with more than 65% erroneous values	Excluding attributes with more than 65% erroneous values
79.456	52.948	3.929

correcting for erroneous values. The plot shows that the corrected dataset has values that are consistent with true sensors readings for the system, compared with the initial dataset. This indicates the viability of the method and on this basis, the steps four to ten are repeated until the missing values in the selected stage 1 output measurement values have all been replaced with their predicted values. The combination of ReliefF and Random Forest is applied. The corrected dataset is then progressed for correcting stage 2 output measurement values. And steps four to ten are applied accordingly.

It is important to note that missing values of less than 5% are considered trivial [22, 60]. They are insignificant to cause any major performance degradation or cast suspicion on the results of applying a machine learning algorithm. And so, attributes with less than 5% missing values were not corrected for this research.

Table 5 shows the summary of the total number of erroneous values that were in the dataset before and after correction. The erroneous values are equivalent to the number of missing values, given that erroneous values were converted to missing values. From the Table 5, if those attributes with significant missing values (see Table 2) are excluded, the erroneous values in the dataset is reduced by about 95%. The percentage can be 100% if those attributes with less than 5% erroneous values are also corrected. For a process flow dataset, the results can be taken to mean that the methodology is able to correct erroneous values in an attribute having as much as 65% spurious values.

Discussions and conclusions

The methodology yields a number of benefits for industry practitioners. If every noisy instance were to be discarded in a dataset, decision makers may be left with a much-reduced dataset for decision making. With the methodology, dataset owners and users in manufacturing systems and systems alike, do not need to discard very noisy attributes or noisy instances. The methodology helps in the correction of erroneous values in a dataset.

Data driven predictions along a multi-stage manufacturing process has been researched [61–63]. The studies do not explicate how to navigate the entire process of mining a multi-stage manufacturing process dataset. In other words, they do not describe in detail, data exploration, pre-processing, dataset partitioning, model building, results parsing and cleaning and analysis. The methodology and the study reported in this article bridges the above stated knowledge gap, and so provides process managers with the workflow for mining and correcting their datasets.

A manufacturing process can be defined as a sequence of interlinked operational activities in a manufacturing system, where each activity leads to the next, and the flow of activities forms a whole [64]. Manufacturing is generally fulfilled in stages. It was shown in this article that each stage in a manufacturing process can be used to demarcate relevant process variables that should be progressed to build a learning model. This helps in pruning the dataset so that model performance is enhanced. Data modellers for multi-stage process learning would find this very useful where there are thousands of process variables and tens to hundreds of stages.

Machine learning is an enabler for knowledge discovery. The methodology uses both feature engineering and classification modelling methods. Both methods on their own are able to extract complex patterns in large datasets that help provide a deeper understanding of the system that generated the dataset. The feature reduction component of the methodology was able to indicate those few attributes that can be used to explain each output measurement. This is a noise reduction strategy, as it reduces the feature space to only those that are most relevant to explaining the target attribute. Then, decision makers can use such information to understand the core relationships in their system. The classification prediction modelling process to replace missing values is similar to the process one would follow when building a classifier prediction model for predictive process monitoring. The prediction models built to replace missing values can also be taken as the prediction models to use in subsequent prediction tasks. In other words, there is no need to build another model, rather the task would be to improve training by increasing the number of training samples, which ultimately leads to an increased prediction accuracy of the model.

The dataset correction approach uses prediction to replace erroneous values in the dataset. With the approach, a predictive model is built. The built model has a very high predictive accuracy, see "Apply built classifier model on class-specified dataset" section, giving one confidence that the replacement values accurately represent what would have been true values.

The emphasis of denoising the dataset by replacing erroneous values with their likely values is so that the corrected dataset can be used for data modelling. With the corrected dataset, analysis of the system that generated the dataset can be enhanced. For instance, the important patterns in the corrected dataset can be learnt using any classification or clustering algorithm that fits the dataset. A research effort that has emanated from the current one is the development of a real-time predictive process control system, that can be used to circumvent unqualified outputs at the end of each stage in the manufacturing process. This is made possible because of the corrected dataset, the learning of the dataset and the discovery of the dataset patterns, all of which are outcomes of the current de-noising study.

The methodology presented in this article represents noise correction of batched data. The built models can be injected into streaming data to enable real-time predictive process monitoring [65]. In addition, the process and model building steps can be collapsed into one algorithm to significantly reduce the time taken to manually correct the dataset. This advancement to the methodology is being undertaken by the author.

Researchers have developed various approaches to dealing with data noise (refs). One approach cannot be said to be more superior than others, and it would appear that approaches are case specific. The specificity is due to the characteristics of the dataset as well as the objective of the data correction task. For instance, in this study, the dataset represented a multi-stage, multi-feature process. The inclusion of data partitioning in the methodological steps arises dues to the type of dataset. The use of feature relevance reduction was to improve the performance of the classifier model that was built to predict replacement values for the erroneous values.

Due to the specificity of the approaches to deal with data noise, results can hardly be compared. Approaches may be compared, but only where similar systems or datasets have been modelled. The uniqueness of the approach presented in this study is in the use of the end-to-end data mining process to implement noise correction in a dataset. The data exploration and pre-processing with feature reduction and classification modelling are all important tasks in the de-noising and dataset correction workflow.

[10] list some likely drawbacks of replacing noisy values with values predicted using a classifier model. They argue that it carries a high computational cost. This is true, considering the number of models to build when there are many attributes to correct for erroneous values. A multi-target algorithm can be considered as a likely alternative worth testing and is an area for further research.

During the analysis, it was found that prediction accuracy drops around the vicinity of erroneous or missing values. This finding has implications for real-time predictive process monitoring. Managers should be aware that prediction may degrade where there are many erroneous values prior to the value being predicted. This is another area worthy of further research.

There is a paucity of the type of methodology presented in the current research. To the best of the author's knowledge, it has not been reported in the literature. The methodology is generalizable in manufacturing systems and can be applied to non-manufacturing related datasets. In this article, the methodology has been described with reproducibility in view. In addition to this, the software tools are generic and opensource.

Acknowledgements
Special thanks to Liveline Technologies, the dataset providers, who gave additional information regarding the system that generated the dataset that was used in this research. Deep gratitude to Tanya Fielder and Joseph Hernandez both of Liveline. Further thanks to the anonymous reviewers whose comments and suggestions enhanced this article.nnn

Authors' contributions
Author provided work structure, conceptualization, data exploration and processing, data analysis and original draft preparation. All authors read and approved the final manuscript.

Author information
The author is a manufacturing analytics consultant and an independent researcher.

Funding
Not applicable.

Availability of data and materials
The original dataset used for this study is available in the Kaggle repository https://www.kaggle.com/supergus/multistage-continuousflow-manufacturing-process

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The author declares that there are no competing interests.

Appendix 1. Variables 1–3 in the raw dataset: Time stamp and factory ambient conditions

No.	Time stamp	No.	Factory ambient conditions
1	Time_stamp	2	Ambient Conditions. Ambient Humidity. U. Actual
		3	Ambient Conditions. Ambient Temperature. U. Actual

Appendix 2. Variables 4–72 in the raw dataset: Stage 1 input process variables and Stage 1 output process variables

No.	Stage 1 input process variables	No.	Stage 1 output process variables
4	Machine1. RawMaterial. Property1	43	Stage1. Output. Measurement0. U. Actual
5	Machine1. RawMaterial. Property2	44	Stage1. Output. Measurement0. U. Setpoint
6	Machine1. RawMaterial. Property3	45	Stage1. Output. Measurement1. U. Actual
7	Machine1. RawMaterial. Property4	46	Stage1. Output. Measurement1. U. Setpoint
8	Machine1. RawMaterialFeederParameter. U. Actual	47	Stage1. Output. Measurement2. U. Actual
9	Machine1. Zone1Temperature. C. Actual	48	Stage1. Output. Measurement2. U. Setpoint
10	Machine1. Zone2Temperature. C. Actual	49	Stage1. Output. Measurement3. U. Actual
11	Machine1. MotorAmperage. U. Actual	50	Stage1. Output. Measurement3. U. Setpoint
12	Machine1. MotorRPM. C. Actual	51	Stage1. Output. Measurement4. U. Actual
13	Machine1. MaterialPressure. U. Actual	52	Stage1. Output. Measurement4. U. Setpoint
14	Machine1. MaterialTemperature. U. Actual	53	Stage1. Output. Measurement5. U. Actual
15	Machine1. ExitZoneTemperature. C. Actual	54	Stage1. Output. Measurement5. U. Setpoint
16	Machine2. RawMaterial. Property1	55	Stage1. Output. Measurement6. U. Actual
17	Machine2. RawMaterial. Property2	56	Stage1. Output. Measurement6. U. Setpoint
18	Machine2. RawMaterial. Property3	57	Stage1. Output. Measurement7. U. Actual
19	Machine2. RawMaterial. Property4	58	Stage1. Output. Measurement7. U. Setpoint
20	Machine2. RawMaterialFeederParameter. U. Actual	59	Stage1. Output. Measurement8. U. Actual
21	Machine2. Zone1Temperature. C. Actual	60	Stage1. Output. Measurement8. U. Setpoint
22	Machine2. Zone2Temperature. C. Actual	61	Stage1. Output. Measurement9. U. Actual
23	Machine2. MotorAmperage. U. Actual	62	Stage1. Output. Measurement9. U. Setpoint
24	Machine2. MotorRPM. C. Actual	63	Stage1. Output. Measurement10. U. Actual
25	Machine2. MaterialPressure. U. Actual	64	Stage1. Output. Measurement10. U. Setpoint
26	Machine2. MaterialTemperature. U. Actual	65	Stage1. Output. Measurement11. U. Actual
27	Machine2. ExitZoneTemperature. C. Actual	66	Stage1. Output. Measurement11. U. Setpoint
28	Machine3. RawMaterial. Property1	67	Stage1. Output. Measurement12. U. Actual
29	Machine3. RawMaterial. Property2	68	Stage1. Output. Measurement12. U. Setpoint
30	Machine3. RawMaterial. Property3	69	Stage1. Output. Measurement13. U. Actual
31	Machine3. RawMaterial. Property4	70	Stage1. Output. Measurement13. U. Setpoint
32	Machine3. RawMaterialFeederParameter. U. Actual	71	Stage1. Output. Measurement14. U. Actual
33	Machine3. Zone1Temperature. C. Actual	72	Stage1. Output. Measurement14. U. Setpoint
34	Machine3. Zone2Temperature. C. Actual		
35	Machine3. MotorAmperage. U. Actual		
36	Machine3. MotorRPM. C. Actual		
37	Machine3. MaterialPressure. U. Actual		
38	Machine3. MaterialTemperature. U. Actual		
39	Machine3. ExitZoneTemperature. C. Actual		
40	FirstStage. CombinerOperation. Temperature1. U. Actual		
41	FirstStage. CombinerOperation. Temperature2. U. Actual		
42	FirstStage. CombinerOperation. Temperature3. C. Actual		

Appendix 3. Variables 73–116 in the raw dataset: Stage 2 input process variables and Stage 2 output process variables

No.	Stage 2 input process variables	No.	Stage 2 output process variables
73	Machine4. Temperature1. C. Actual	87	Stage2. Output. Measurement0. U. Actual
74	Machine4. Temperature2. C. Actual	88	Stage2. Output. Measurement0. U. Setpoint

No.	Stage 2 input process variables	No.	Stage 2 output process variables
75	Machine4. Pressure. C. Actual	89	Stage2. Output. Measurement1. U. Actual
76	Machine4. Temperature3. C. Actual	90	Stage2. Output. Measurement1. U. Setpoint
77	Machine4. Temperature4. C. Actual	91	Stage2. Output. Measurement2. U. Actual
78	Machine4. Temperature5. C. Actual	92	Stage2. Output. Measurement2. U. Setpoint
79	Machine4. ExitTemperature. U. Actual	93	Stage2. Output. Measurement3. U. Actual
80	Machine5. Temperature1. C. Actual	94	Stage2. Output. Measurement3. U. Setpoint
81	Machine5. Temperature2. C. Actual	95	Stage2. Output. Measurement4. U. Actual
82	Machine5. Temperature3. C. Actual	96	Stage2. Output. Measurement4. U. Setpoint
83	Machine5. Temperature4. C. Actual	97	Stage2. Output. Measurement5. U. Actual
84	Machine5. Temperature5. C. Actual	98	Stage2. Output. Measurement5. U. Setpoint
85	Machine5. Temperature6. C. Actual	99	Stage2. Output. Measurement6. U. Actual
86	Machine5. ExitTemperature. U. Actual	100	Stage2. Output. Measurement6. U. Setpoint
		101	Stage2. Output. Measurement7. U. Actual
		102	Stage2. Output. Measurement7. U. Setpoint
		103	Stage2. Output. Measurement8. U. Actual
		104	Stage2. Output. Measurement8. U. Setpoint
		105	Stage2. Output. Measurement9. U. Actual
		106	Stage2. Output. Measurement9. U. Setpoint
		107	Stage2. Output. Measurement10. U. Actual
		108	Stage2. Output. Measurement10. U. Setpoint
		109	Stage2. Output. Measurement11. U. Actual
		110	Stage2. Output. Measurement11. U. Setpoint
		111	Stage2. Output. Measurement12. U. Actual
		112	Stage2. Output. Measurement12. U. Setpoint
		113	Stage2. Output. Measurement13. U. Actual
		114	Stage2. Output. Measurement13. U. Setpoint
		115	Stage2. Output. Measurement14. U. Actual
		116	Stage2. Output. Measurement14. U. Setpoint

C. Setpoint - Setpoint for Controlled variable, C. Actual - Actual value of Controlled variable, U. Actual - Actual value of Uncontrolled variable .

Appendix 4. Notes to the. csv dataset

Process description

Start column	End column	Column description
0	0	Time stamp
1	2	Factory ambient conditions
3	6	First stage, Machine 1, raw material properties (material going in to Machine 1)
7	14	First stage, Machine 1 process variables
15	18	First stage, Machine 2, raw material properties (material going in to Machine 2)
19	26	First stage, Machine 2 process variables
27	30	First stage, Machine 3, raw material properties (material going in to Machine 3)
31	38	First stage, Machine 3 process variables
39	41	Combiner stage process parameters. Combines the outputs from Machines 1, 2, and 3
42	71	Measurements of 15 features (in mm), along with setpoint or target for each
72	78	Second stage, Machine 4 process variables
79	85	Second stage, Machine 5 process variables
86	115	Measurements of 15 features (in mm), along with setpoint or target for each

The process is a continuous flow process consisting of two stages (see Fig.1). In the first stage, Machines 1, 2, and 3 operate in parallel and feed into a Combiner. The output from the Combiner is measured in 15 different locations (Stage1.Output.Measurement0.U.Actual to Stage1.Output.Measurement14.U.Actual) against their set points (Stage1.Output.Measurement0.U.Setpoint to Stage1.Output.Measurement14.U.Setpoint). The output from the combiner flows into the second stage, where Machines 4 and 5 process in series. The output from Machine 5 is measured in the same 15 locations (Stage2.Output.Measurement0.U.Actual to Stage2.Output.Measurement14.U.Actual) as the output from the Combiner and compared with their setpoint measurements (Stage2.Output.Measurement0.U.Setpoint to Stage2.Output.Measurement14.U.Setpoint).

The output measurements are collected via sensors mounted on the machines. Zero measurements are as a result of sensor and/or system failures. Erroneous values are as a result of the product unsteady movement and product surface contamination

Received: 30 July 2020 Accepted: 8 October 2020
Published online: 17 October 2020

References

1. Shao J, et al. Automatic weld defect detection based on potential defect tracking in real-time radiographic image sequence. NDT and E Int. 2012;46:14–21.
2. Kim S. et al. Dealing with noise in defect prediction. In: 2011 33rd International Conference on Software Engineering (ICSE). IEEE. 2011.
3. Kaggle. Multi-Stage Continuous-Flow Manufacturing Process. 2020. https://www.kaggle.com/supergus/multistage-continuousflow-manufacturing-process. Accessed 20 Mar 2020]
4. Müller H, Freytag J-C, Problems, methods, and challenges in comprehensive data cleansing. Professoren des Inst. Für Informatik. 2005.
5. Peres RS, et al. Multistage quality control using machine learning in the automotive industry. IEEE Access. 2019;7:79908–16.
6. Zeng J-S, Gao C-H. Improvement of identification of blast furnace ironmaking process by outlier detection and missing value imputation. J Process Control. 2009;19(9):1519–28.
7. Lee H, Kim Y, Kim CO. A deep learning model for robust wafer fault monitoring with sensor measurement noise. IEEE Trans Semicond Manuf. 2016;30(1):23–31.
8. Sáez JA, et al. Tackling the problem of classification with noisy data using multiple classifier systems: analysis of the performance and robustness. Inf Sci. 2013;247:1–20.
9. Cheng H-J, Kumar A. Process mining on noisy logs—Can log sanitization help to improve performance? Decis Support Syst. 2015;79:138–49.
10. Acuna E, Rodriguez C. The treatment of missing values and its effect on classifier accuracy. In: Classification, clustering, and data mining applications. 2004, Springer. p. 639–647.
11. Bohannon P. et al. A cost-based model and effective heuristic for repairing constraints by value modification. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. 2005.
12. Mayfield C, Neville J, Prabhakar S. ERACER: a database approach for statistical inference and data cleaning. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. 2010.
13. Gupta A, Lam M. The weight decay backpropagation for generalizations with missing values. Ann Oper Res. 1998;78:165–87.
14. Magnani M. Techniques for dealing with missing data in knowledge discovery tasks. Obtido https://magnanim.web.cs.unibo.it/index.html, 2004. 15(01): p. 2007.
15. Adams S. et al. A comparison of feature selection and feature extraction techniques for condition monitoring of a hydraulic actuator. In: Annual Conference of the Prognostics and Health Management Society 2017. 2017.
16. Matsumura T, Haftka RT, Kim NH. Accurate predictions from noisy data: replication versus exploration with applications to structural failure. Struct Multidisciplin Optimizat. 2015;51(1):23–40.
17. Folguera L, et al. Self-organizing maps for imputation of missing data in incomplete data matrices. Chemometrics Intell Labo Syst. 2015;143:146–51.
18. Zhang Y, Wu X. Integrating induction and deduction for noisy data mining. Inf Sci. 2010;180(14):2663–73.
19. Atla A, et al. Sensitivity of different machine learning algorithms to noise. J Comput Sci Coll. 2011;26(5):96–103.
20. Lee D-H, et al. A data-driven approach to selection of critical process steps in the semiconductor manufacturing process considering missing and imbalanced data. J Manufact Syst. 2019;52:146–56.
21. Boukouvala F, Muzzio FJ, Ierapetritou MG. Predictive modeling of pharmaceutical processes with missing and noisy data. AIChE J. 2010;56(11):2860–72.

22. García-Gil D, et al. Enabling smart data: noise filtering in big data classification. Inf Sci. 2019;479:135–52.
23. Jearkpaporn D, et al. Process monitoring for mean shifts for multiple stage processes. Int J Prod Res. 2007;45(23):5547–70.
24. Chandrashekar G, Sahin F. A survey on feature selection methods. Comput Electr Eng. 2014;40(1):16–28.
25. Dash M, Liu H. Feature selection for classification. Intell Data Analy. 1997;1(3):131–56.
26. Liu T-I, et al. Real-time recognition of ball bearing states for the enhancement of precision, quality, efficiency, safety, and automation of manufacturing. Int J Adv Manufact Technol. 2014;71(5–8):809–16.
27. Flath CM, Stein N. Towards a data science toolbox for industrial analytics applications. Comput Ind. 2018;94:16–25.
28. Guyon I, Elisseeff A. An introduction to variable and feature selection. J Mach Learn Res. 2003;3(Mar):1157–82.
29. Schuh G, et al. Databased prediction of order-specific transition times. CIRP Ann. 2019;68(1):467–70.
30. Cortés-Ibáñez JA, et al. Preprocessing methodology for time series: an industrial world application case study. Inf Sci. 2020;514:385–401.
31. Lin T-K. Adaptive Principal Component Analysis Combined with Feature Extraction-Based Method for Feature Identification in Manufacturing. J Sens. 2019;2019:19.
32. Kononenko I. Estimating attributes: analysis and extensions of RELIEF. In: European conference on machine learning. 1994. Springer.
33. Yu L, Liu H. Feature selection for high-dimensional data: A fast correlation-based filter solution. In: Proceedings of the 20th international conference on machine learning (ICML-03). 2003.
34. Lu Y. et al. Feature selection using principal feature analysis. In: Proceedings of the 15th ACM international conference on Multimedia. 2007.
35. Hall MA. Correlation-based feature selection of discrete and numeric class machine learning. 2000.
36. Arauzo-Azofra A, Aznarte JL, Benítez JM. Empirical study of feature selection methods based on individual feature evaluation for classification problems. Expert Syst Appl. 2011;38(7):8170–7.
37. Géron A. Hands-on machine learning with scikit-learn and tensorflow: concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc." 2017.
38. Fu Y, et al. Analysis of feature extracting ability for cutting state monitoring using deep belief networks. Procedia Cirp. 2015;31(Suppl. C):29–34.
39. Chen Z, et al. Characterizing strip snap in cold rolling process using advanced data analytics. Procedia CIRP. 2019;81:453–8.
40. Chen C, et al. Energy consumption modelling using deep learning technique—a case study of EAF. Procedia CIRP. 2018;72:1063–8.
41. Köksal G, Batmaz İ, Testik MC. A review of data mining applications for quality improvement in manufacturing industry. Expert Syst Appl. 2011;38(10):13448–67.
42. Clancey WJ. Heuristic classification. Artif Intell. 1985;27(3):289–350.
43. Nikam SS. A comparative study of classification techniques in data mining algorithms. Oriental J Comput Sci Technol. 2015;8(1):13–9.
44. Chizi B, Maimon O. Dimension reduction and feature selection. In: Data mining and knowledge discovery handbook. Springer; 2009. p. 83–100.
45. Choudhary AK, Harding JA, Tiwari MK. Data mining in manufacturing: a review based on the kind of knowledge. J Intell Manuf. 2009;20(5):501.
46. Rostami H, Dantan J-Y, Homri L. Review of data mining applications for quality assessment in manufacturing industry: support vector machines. Int J Metrol Quality Eng. 2015;6(4):401.
47. Frénay B, Verleysen M. Classification in the presence of label noise: a survey. IEEE Transact Neural Netw Learn Syst. 2013;25(5):845–69.
48. Breiman L. Random forests. Machine Learn. 2001;45(1):5–32.
49. Farid DM, Maruf GM, Rahman CM. A new approach of Boosting using decision tree classifier for classifying noisy data. In: 2013 International Conference on Informatics, Electronics and Vision (ICIEV). IEEE. 2013.
50. Oleghe O, Salonitis K. A framework for designing data pipelines for manufacturing systems. In: 53rd CIRP Conference on Manufacturing Systems. Il. July 1–3. 2020.
51. Ismail A, Truong H, Kastner W. Manufacturing process data analysis pipelines- a requirements analysis and survey. J Big Data. 2019. 6(1–26).
52. Last, M., A. Sinaiski, and H.S. Subramania. Predictive maintenance with multi-target classification models. In: Asian Conference on Intelligent Information and Database Systems. Springer. 2010.
53. Tsoumakas G. et al. Multi-target regression via random linear target combinations. In: Joint European conference on machine learning and knowledge discovery in databases. Springer. 2014.
54. Sheng VS. An empirical comparison on multi-target regression learning. Comput Mater Continua. 2018;56(2):185–98.
55. Modi H, Panchal M. Experimental comparison of different problem transformation methods for multi-label classification using MEKA. Int J Comput Applic. 2012;59(15):10–5.
56. Gittler T, et al. Towards predictive quality management in assembly systems with low quality low quantity data–a methodological approach. Procedia CIRP. 2019;79:125–30.
57. Kira K, Rendell LA. A practical approach to feature selection. In: Machine Learning Proceedings. Elsevier. 1992. 1992, p. 249–256.
58. Robnik-Šikonja M, Kononenko I. Theoretical and empirical analysis of ReliefF and RReliefF. Machine Learn. 2003;53(1–2):23–69.
59. Gyulai D, et al. Lead time prediction in a flow-shop environment with analytical and machine learning approaches. IFAC-PapersOnLine. 2018;51(11):1029–34.
60. Vagin V, Fomina M. Problem of knowledge discovery in noisy databases. Int J Mach Learn Cybern. 2011;2(3):135.
61. Arif F, Suryana N, Hussin B. A data mining approach for developing quality prediction model in multi-stage manufacturing. Int J Comput Appl. 2013;69(22):35–40.

62. Gazzola G, et al. Integrated variable importance assessment in multi-stage processes. IEEE Trans Semicond Manuf. 2018;31(3):343–55.
63. Lughofer E. et al. Evolving time-series based prediction models for quality criteria in a multi-stage production process. In: 2018 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS). IEEE. 2018.
64. Etgar M. A descriptive model of the consumer co-production process. J Acad Mark Sci. 2008;36(1):97–108.
65. Kenda K, Mladenić D. Autonomous sensor data cleaning in stream mining setting. Business Syst Res J. 2018;9(2):69–79.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.