Learning with supervised data for anomaly detection in smart manufacturing

Meiling He, Matthew Petering, Phillip LaCasse, Wilkistar Otieno & Francisco Maturana

INTERNATIONAL JOURNAL OF COMPUTER INTEGRATED MANUFACTURING
2023, VOL. 36, NO. 9, 1331–1344
https://doi.org/10.1080/0951192X.2023.2177747

Learning with supervised data for anomaly detection in smart manufacturing

Meiling He[a], Matthew Petering[b], Phillip LaCasse[c], Wilkistar Otieno[b] and Francisco Maturana[a]

[a]Department of AI/IoT DA&I-IT, Rockwell Automation, Inc., Milwaukee, USA; [b]University of Wisconsin – Milwaukee, Milwaukee, WI, USA; [c]Department of Industrial and Manufacturing Engineering, University of Wisconsin–Milwaukee, Milwaukee, USA

ABSTRACT
The emergence of the Internet of Things (IoT), cloud computing, cyber-physical systems, system integration, big data, and data analytics for Industry 4.0 have transformed the world of traditional manufacturing into an era of smart manufacturing (SM). Smart manufacturing's central focus is to process real-time IoT data and leverage advanced analytical approaches to detect abnormal behaviors. Social smart manufacturing applies analytics tools to empower decision makers and minimize duplication by executing the repetitive data processing work more consistently and precisely than can be done by a human operator. In smart manufacturing, the majority of industrial data is imbalanced. However, most traditional machine learning algorithms tend to be biased toward the majority class and under-represent the minority class. This research proposes a model selection architecture to automate the procedure of preprocessing input data and selecting the best combination of algorithms for anomaly detection. This design will play an essential role in producing high-quality products and improving quality control and business processes in diverse applications including predictive maintenance and fault detection. The framework is transferrable to any smart manufacturing task in the supervised learning domain.

ARTICLE HISTORY
Received 1 July 2022
Accepted 30 January 2023

KEYWORDS
Advanced analytics; smart manufacturing; imbalanced data; generic model selection; big data; anomaly detection

1. Introduction

Industry is now going through its fourth generation of the industrial revolution or what is known as Industry 4.0. The term Industry 4.0 was first introduced by the German government for defining smart manufacturing (Kang, Kim, and Cho 2016).

The growing application of sensors in manufacturing lines, the collection of environmental data, and increased access to various machine parameters have resulted in an explosion in the quantity of available data. The availability of big data has motivated extensive research on developing approaches for handling a host of complicated industrial problems and can be utilized to help leaders make better business decisions (Maddikunta et al. 2021). Of primary importance has been the development of conceptual frameworks to deal assist the analyst in converting the volumes of data and associated multiplicity of approaches into useful information (Kozjek et al. 2020). Such approaches have been successfully employed in applied case studies featuring semiconductor manufacturing (Moyne and Iskandar 2017), tool wear (Wu et al. 2017), remanufacturing (Zhang et al. 2022), and affective design (Yan Chan et al. 2018). Big data has also motivated the development of new approaches, with the progression of research migrating from traditional machine learning models (Wuest et al. 2016) to artificial intelligence approaches such as digital twin implementations or reinforcement learning (Liu et al. 2019).

Notwithstanding the tremendous opportunity afforded by big data, there are nontrivial challenges that must be navigated. Security and safety concerns must be adequately addressed. Data cannot be effectively used unless it is properly formatted, a process that can be extensive. Legacy data, perhaps collected for one purpose but retained long term in archives, may be leveraged in an attempt to solve problems that are not the same as the original purpose for which the data were collected. This does not preclude the development of useful models but care must be taken to ensure that data is not misused or that inappropriate conclusions are not drawn from it. An additional observation is the phenomenon by which manufacturers collect large amounts of data but analyze only a small portion of it. The consequence is that

CONTACT Meiling He mhe2@rockwellautomation.com Department of Operational Sciences, Air Force Institute of Technology, 2950 Hobson Way, Bldg 641 Wright Patterson, OH 45433

manufacturers may possess only limited knowledge of the relative value of the smart manufacturing data that they collect (LaCasse, Otieno, and Maturana 2019). This challenge was identified early in the development of big data, coining the phrase *data-rich but knowledge poor* (Lu 1990).

This research explores the challenges associated with anomaly detection using IoT data that exhibit the following difficulties: high dimensionality, data class imbalance, or a need for preprocessing. High-dimensional data usually carry redundant and irrelevant information that will impair machine learning performance; this is sometimes referred to as the *curse of dimensionality*. Therefore, dimension reduction or projection should be applied to reduce noise and redundancies. LaCasse, Otieno, and Maturana (2018) propose a framework for feature selection and prioritization by which high-dimensional data is screened via an initial test for statistical independence and a subsequent assignment of a numeric value and linguistic label via fuzzy inference (LaCasse, Otieno, and Maturana 2018). The approach is best suited for exploratory analysis with a large potential feature set and was tested in a case study in defect prediction for printed circuit boards employing ball grid array (BGA) package types (LaCasse, Otieno, and Maturana 2020).

Imbalance within datasets happens when the number of normal class observations is much larger than that for the abnormal class. Imbalance in data can diminish the detection performance of most traditional machine learning algorithms, as those algorithms are designed to generate high accuracy, which often over-represents the majority class and under-represents the minority class. Therefore, some pre-processing techniques such as over-sampling of the minority class, under-sampling of the majority class, or combining both methods can be used prior to using traditional ML algorithms. Alternatively, some ensemble-based ML tools that were designed to process highly imbalanced data can be used independently to minimize poor ML learning outcomes caused by imbalance in data. A variety of factors make the datasets involved in smart manufacturing well-suited to the application of anomaly detection in general (Pittino et al. 2020). Therefore, to operate a reliable, resilient, and secured manufacturing system, any abnormal behavior in operating machines, running conditions, control procedures, communication systems, and generated products have to be continuously monitored (Hansson et al. 2016; Pittino et al. 2020)

In the past decade, many ML tools have been introduced to enhance fault detection and diagnosis (Cinar et al. 2020). Additionally, in a complex, dynamic, and even chaotic environment, applying a single machine learning tool to learn IoT data with high diversity is not realistic (Lee and Ha 2009; Koksal, Batmaz, and Testik 2011). Instead, in order to best adapt to the complexities of processing diverse data in the field, this research proposes a supervised anomaly detection framework to generate better overall learning performances. In this supervised machine learning framework, a combination of different machine learning algorithms is applied and the one which produces the best results is automatically selected.

This research makes a novel contribution that is of practical relevance to the smart manufacturing community by addressing a specific gap in the current body of knowledge. While it is true that some previous studies have compared multiple machine learning algorithms for specific problems in certain domains (Tao et al. 2018; Wu et al. 2017; Hansson et al. 2016; Kang, Kim, and Cho 2016), there is heretofore no developed architecture for auto-selecting the best classifier from running multiple algorithms within a framework for solving industrial problems. Similarly, studies have been performed employing a single algorithm such as Naïve Bayes (Brundage, Ademujimi, and Rakshith 2017) or using combinations of algorithms (Fathy, Jaber, and Brintrup 2020); however, this research is distinct in that it automates the machine learning model selection.

The auto-selection framework eliminates the duplication of coding and work from human analysts and data scientists when solving supervised problems. It creates collaboration between humans and our advanced analytics tool, saving more labor effort for implementing tasks that automated algorithms cannot perform such as analysis of results and formulation of courses of action. Additionally, this framework is likely to improve overall anomaly detection performances and minimize unplanned downtimes in Smart Manufacturing by virtue of automatically creating the best performing model for processing both balanced and imbalanced data.

This rest of this paper is organized as follows. Section 2 describes our first ML auto-selection architecture for supervised data processing. In Section 3, experimental results from real-world applications of the proposed architectures are presented. Section 4 states conclusions and discusses opportunities for future research.

2. Supervised machine learning algorithm auto selection architecture for solving data class imbalance problem

2.1 Supervised learning

In industrial supervised classification tasks, the data $\mathcal{D}$ consists of a matrix $X \in \mathbb{R}^{n \times d}$ as

$$X = \{x_i^d\}_{i=1}^n \tag{1}$$

and a one-dimensional binary labeled vector y as

$$y = \{y_i\}_{i=1}^n \tag{2}$$

where x_i^d is the i^{th} observation of d-dimensional sample data and $y_i \in \{0, 1\}$ is the corresponding label that indicates the class to which x_i^d belongs.

Supervised learning can be formulated as:

$$\mathcal{D} = \left\{ \left(x_i^d, y_i\right) | x_i^d \in \mathbb{R}^d, y_i \in \{0, 1\} \right\}_{i=1}^n \tag{3}$$

The objective of supervised learning is to find an optimized and generalized transformation $\mathcal{H}$of input X to the output labels y based on certain evaluation metrics that minimize the error between y and $\hat{y}$:

$$\mathcal{H}(X) = \hat{y} \tag{4}$$

2.2 Preprocessing

IoT labeled data $\mathcal{D}$ can consist of both categorical and numerical information. All categorical information is converted to numerical information in the preprocessing stage by using label encoding. In addition, best features can be selected if desired. Pearson correlation coefficient, ρ^j, is used to rank the significance of information carried by each feature x^j.

$$\rho^j = \frac{cov\left(x^j, y\right)}{\sigma_{x^j}, \sigma_y} \tag{5}$$

In Equation 5, $cov\left(x^j, y\right)$ indicates the covariance between the j^{th} feature x^j and label vector y, σ_{x^j} represents the standard deviation of attribute x^j, and σ_y is the standard deviation of attribute y. $\rho^j \in [0, 1]$ indicates the linear correlation between feature j and label vector y. In this paper, the top k most related features are selected for further processing.

2.3 Imbalance problem

The imbalance problem arises when there is significant inequality between the number of samples from different classes. The input distribution of the imbalance problem, if not processed appropriately, can have adverse effects on the classification performances of most machine learning algorithms (Oksuz et al. 2020; Fathy, Jaber, and Brintrup 2020). In real-world applications, datasets are usually dominated by samples from the 'normal' class (or class negatives) with a small proportion of samples from the 'abnormal' or 'interest' class (or class positives). Also, misclassifying 'abnormal' class samples usually costs more than misclassifying 'normal' samples.

Therefore, it is typically preferable to obtain an excellent detection of positive instances while maintaining highest possible negative instances classification rate.

2.4 Evaluating metrics for imbalance classification

The most frequently used performance evaluation metrics for classification models are Accuracy or Error Rate. It indicates how well the models correctly classify both normal (negatives) and abnormal (positives) behaviors or objects (He and Garcia 2009).

Let {TP, TN, FP, FN} in Table 1 represent the correctly classified positive samples (true positive), correctly classified negative samples (true negative), incorrectly classified negative samples (false positive), and incorrectly classified positive samples (false negative). Then, the Accuracy can be formulated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

And the Error Rate is:

$$Error\ Rate = 1 - Accuracy \tag{7}$$

Table 1. Confusion matrix for performance evaluation.

	Actual Positive	Actual Negative
Predicted Positive	TP (True Positive)	FP (False Positive)
Predicted Negative	FN (False Negative)	TN (True Negative)

In the case of imbalanced data, neither Accuracy nor Error Rate are likely to be appropriate to evaluate the classification performance. For example, if a dataset holds 1% positive samples and 99% negative samples, incorrectly classifying all positive samples would generate accuracy as high as 99%, which does not show good detection performance because all positives (which are usually more important) are misclassified. Such a model would not be useful to a decision maker.

In place of Accuracy and Error rate, a few other assessment metrics are suggested (He and Garcia 2009) to be used in the imbalanced data classification community, such as Precision, Sensitivity, F-Measure, and G-mean as:

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Sensitivity = TPR = \frac{TP}{TP + FN} \tag{9}$$

$$Specificity = TNR = \frac{TN}{TN + FP} \tag{10}$$

$$\text{F-Measure} = \frac{(1+\beta)^2 \cdot Sensitivity \cdot Precision}{\beta^2 \cdot Sensitivity \cdot Precision} \tag{11}$$

$$\text{G-Mean} = \sqrt{Sensitivity \times Specificity} \tag{12}$$

Sensitivity, also called recall, determines what proportion of positive samples are correctly classified (completeness). A model can artificially inflate Sensitivity simply by classifying every sample as positive. Precision indicates the proportion of positive classifications that are correct (exactness). A model can artificially inflate its Precision score by classifying only a very small number of samples as positive, those being the samples that it is most sure of. Precision is distribution-dependent since it carries information about how many negative samples are misclassified to the positive class but is not sensitive to how many positive samples are misclassified. Sensitivity, on the other hand, is sensitive to correctly classified positives but not misclassified negatives. Depending on the cost associated with a false-positive or a false-negative classification, either Precision or Sensitivity could be the metric of choice. For such scenarios as neither metric is preferable, F-Measure is the combination of Precision and Sensitivity with coefficient β, used to determine the weighted importance of Precision or Recall. G-Mean is evaluating the combination of correctly classified positive samples and negative samples. Both G-Mean and F-Measure are good evaluation metrics for assessing the performance of imbalanced data classification.

2.5 Re-sampling

Re-sampling methods (over-sampling and under-sampling) can often be used to improve the performance of most classification algorithms on highly imbalanced datasets by balancing the sample sizes from different classes. Typically, over-sampling tools are preferable to under-sampling tools because over-sampling would not arbitrarily eliminate samples thereby causing loss of information. This research employs the representative random over-sampling tool adaptive synthetic (ADASYN) (He et al. 2008) and the modified version of synthetic minority over-sampling technique (SMOTE) (He and Garcia 2009). Both ADASYN and SMOTE over-sample the minority class samples by generating synthetic samples based on the information of the existing ones rather than repeating the original samples. This over-sampling process has eliminated the overfitting problem of ML algorithms caused by only fitting the repeated samples (Fernández et al. 2018).

SMOTE is one of the most popular preprocessing tools to balance data. SMOTE oversamples the minority class by introducing extra samples from the neighborhood. Figure 1 illustrates SMOTE for synthetic sample generation. ADASYN uses a similar sampling process except that it adds small random values to the synthetic samples. The following steps explain the synthetic sample generation using SMOTE. First, a minority class sample x_i is selected as a base sample. Next, k-Nearest neighbor are found for sample x_i. Finally, new minority class samples are generated by interpolation.

Interpolation is performed by computing the difference between the feature vector of the sample x_i and the vector of each randomly sampled $L < k$ neighbors. The difference is then multiplied by a random number in (0, 1), and the result is added to the feature vectors of the selected sample x_i to generate the new sample. This process aims to generate the synthetic samples along the line segment between the base sample and its neighbors. ADASYN is the improved version of SMOTE. Besides generating samples in the line segment between the base sample and the

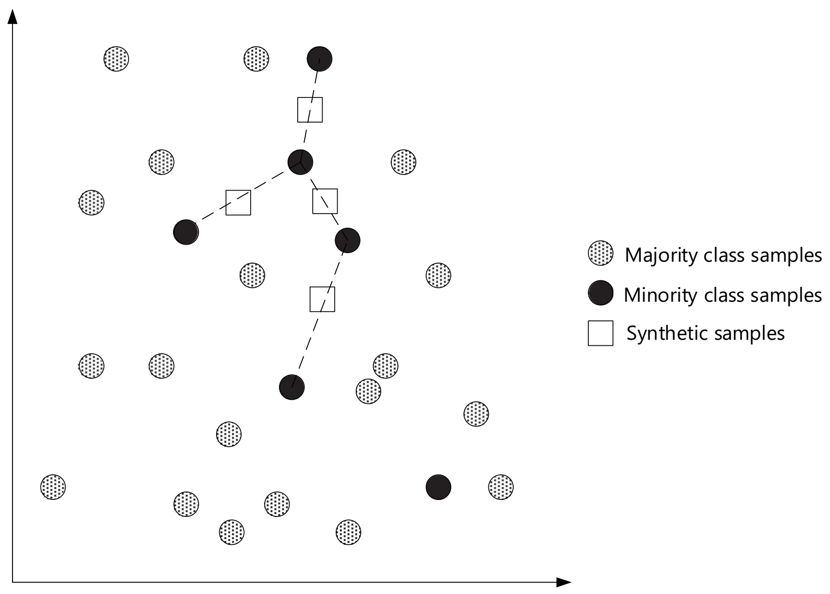

Figure 1. Illustration of SMOTE for synthetic sample generation. ADASYN uses a similar sampling process except it adds small random values to the synthetic samples.

neighbors, ADASYN adds a small random value to each synthetic sample to increase the generality of the oversampling process.

2.6 Supervised machine learning algorithm used

In this section, eight traditional machine learning algorithms used in supervised auto-selection framework are described. These include single model-based algorithms such as neural networks, naïve Bayes, and Decision trees. There are also ensemble-based classifiers such as bagging, extra tree, random forest, and Adaboost.

2.6.1 Decision tree

Decision trees, also known as hierarchical classifiers (Silla Jr. & Freitas, 2011), are a special case of multi-stage classifiers. Decision tree classifiers break down a complex classification problem into a series of simple stump classifiers in a hierarchy of stages which allows rejecting data at intermediate stages. In some applications of decision trees, such as medical diagnosis, simple decision rules make it easier for doctors or other users to interpret results.

Traditional decision trees, such as C4.5, use information gain as a splitting criterion. The goal of decision tree is to choose a split of an attribute (feature) that maximizes a global cost function, such as information gain (Loh 2011).

There are multiple advantages of applying decision trees on classification problems:

(1) In multivariate analysis, one usually has to estimate the distributions of the data or the parameters of the known distributions. In those cases, the 'curse of dimensionality' may be an issue for estimations. A decision tree avoids this problem since it uses small proportions of features in each internal node. In addition, decision tree can generate complex decision boundaries between two classes in binary classification problems. No assumption for the underlying distribution of the training data is required.
(2) Unlike other single-stage machine learning algorithms, such as SVM, a decision tree is designed to decrease computational complexity and time. For the decision tree, only a subset of features is selected by the global optimization criterion. Also, only specific subsets of classifiers are used to evaluate an instance, saving unnecessary computational costs.
(3) A decision tree is well suited to simple, clear interpretations that can be understood by a non-technical audience, translated into metrics, and

coded into business intelligence dashboards for tracking and threshold monitoring.

On the other hand, decision trees have some drawbacks. Similar to neural networks, a decision tree can be easily over-trained. This problem can be improved by pruning redundant nodes or using cross-validation.

2.6.2 Random forest

Random forest is an ensemble-based classifier (Leo 2001). Each base tree classifier is trained based on a bootstrap sample of observations. During the training process, instead of choosing the best split among all the attributes or features for each tree, only a subset of attributes is chosen to obtain the optimal split. In the end, the classification decision is made based on the majority votes from all base trees for predicting each instance. Random forest has shown to be an advantage in processing data when the sample size exceeds the data dimension.

2.6.3 Extremely randomized trees

Extremely randomized tree models (extra tree) are similar to random forest models. Both classifiers are trained based on multiple decision tree classifiers (Geurts, Ernst, and Wehenkel 2006). In extra tree, the threshold of each attribute is assigned randomly rather than choosing the optimal threshold for each feature as random forest does. With the additional randomness, extra tree classifier generates higher variance. Extra tree is typically better than random forest in that it typically has lower computing cost and provides better classification performance.

2.6.4 AdaBoost

AdaBoost was proposed by Freund and Schapire in 1996 (Freund and Schapire 1997) and has since become one of the most important classification methodologies. The same year, Breiman named AdaBoost with tree-based weak classifiers as the 'best off-the-shelf classifier in the world' (Breiman 1996). AdaBoost is constructed using a succession of weak classifiers. In each learning stage, extra weight is assigned to incorrectly classified instances and less weight is given to correctly classified instances by the previous classifier. Eventually, each chosen base classifier is weighted in proportion to the classification performance at each stage. Because of the structure, AdaBoost can convert the base 'weak' classifiers, which perform just slightly better than random guessing, into a more robust classifier with much higher accuracy. The base classifiers can be decision tree, decision tree stumps, neural networks, support vector machines, or other machine learning algorithms. AdaBoost usually performs better than the bagging method and different individual classifiers. However, AdaBoost is generally more sensitive to noise. In this research, AdaBoost typically does not perform better than other algorithms on imbalanced data processing, although this could certainly be a function of the data and the relationships that are contained within.

2.6.5 Nearest neighbor

As one of the oldest and simplest machine learning algorithms, nearest neighbor is very effective for classification. Unlike other machine learning algorithms, the nearest neighbor classifier does not assume the class distribution or the boundaries between the classes. The construction of a nearest neighbor model is dependent on the distance metric. Samples are classified according to the label associated with the majority of its k nearest neighbors in the training set, where k is a hyperparameter (Goldberger et al. 2004). The performance of k-NN can be highly affected by choosing the appropriate distance metric for learning. Like other machine learning algorithms, k-NN performs better when more data is used for training.

2.6.6 Neural networks

Neural networks (NN) are designed to imitate brain function, and this approach has since become a state-of-the-art ML algorithm used in numerous applications (Lee et al. 2019; Bottou 2012; David, Hinton, and Williams 1986). There exist many different versions of NN. This paper applies one of the most common NN, also called multi-layer perception (MLP). The two layers of MLP can be formulated as:

$$\mathcal{H}(X) = \phi^{(2)}\left(\phi^{(1)}\left(X^T w^{(1)}\right)^T w^{(2)}\right) \tag{13}$$

where $\phi(\cdot)$ is the activation function. One of most common activation functions used for training NN is sigmoid function. Here is an example of using a sigmoid function to convert the weighted sum of inputs and a bias in layer m:

$$\phi^{(m)}(\theta) = \frac{1}{1 + e^{-\theta}} \tag{14}$$

If NN has s_m units in layer m and s_{m+1} units in layer $m+1$, then the weight matrix $w^{(m)}$ will be of dimension $s_m \times s_{m+1}$. The weight matrix is obtained by using backpropagation during the learning process. It is essential to optimize the combination of the number of hidden layers and neurons in those layers to obtain a good MLP model; however, the optimized combination is often difficult to obtain and MLP tends to perform poorly with imbalanced data.

2.6.7 Naïve bayes

The Naive Bayes (NB) classifier is built upon Bayes' Theorem. It has simplified the learning process by assuming all data features are independent of one another given class.

Although independence can be a strong assumption in real applications, NB classifier often outperforms over other algorithms in anomaly detection problems by generating a higher true positive rate and lower false-positive rate (Guzella and Caminhas 2009). The NB classifier assigns the most probable class to each sample following the algorithms illustrated as following:

NB classifier uses Bayes' theorem to obtain the probability of a previously unknown data, x, belonging to class c_k as:

$$p(C = c_k|X = x) = \frac{p(C = c_k) \cdot p(X = x|C = c_k)}{p(X = x)} \tag{15}$$

where: $p(C = c_k)$ is the prior probability of class c_k; $p(X = x)$ is the prior probability of input data x; $p(X = x|C = c_k)$ is probability of x given class c_k; $p(C = c_k|X = x)$ is probability of c_k given class x.

Because of the feature independence assumption, $p(X = x|C = c_k)$ can be rewritten as:

$$p(X = x|C = c_k) = \prod_{j=1}^{d} p(X = x^j|C = c_k) \tag{16}$$

In this case, the denominator is effectively constant because it does not depend on the hypothesis C and all the feature values are known.

Then, Eq. (15) can be updated as:

$$p(C = c_k|X = x) = \frac{1}{Z} p(C = c_k) \cdot \prod_{j=1}^{d} p(X = x^j|C = c_k) \tag{17}$$

where $Z = p(X = x)$.

During the training process, the machine learning framework can be optimized as:

$$\hat{y} = argma\ x_y \frac{1}{Z} p(C = c_k) \cdot \prod_{j=1}^{d} p(X = x^j|C = c_k) \tag{18}$$

which is can be reduced to:

$$\hat{y} = argma\ x_y \prod_{j=1}^{d} p(X = x^j|C = c_k) \tag{19}$$

The value of predicted $\hat{y}$ depends on the event model selected to fit the training data and also depends on the occurrence of term x^{tr} in the training datasets. Because of the feature-independent assumption naïve Bayes follows, it typically works better when features are independent or functionally independent (Silla and Freitas 2011).

2.6.8 Bagging

Bagging, also called 'bootstrap aggregating', one of the most popular ensemble-based tools developed by Breiman (Breiman 1996). It is built from growing each base tree classifier from randomly selected bootstrap samples in the training datasets. Bootstrap is a random sampling technique. Many subsets of equal sample size are drawn, with replacement, from the original dataset. During testing, data class is generated based on the majority vote from all selected base tree classifiers. Bagging technique is often superior to most single classifiers and generates less variance apart from nearest neighbors and Naive Bayes classifiers (Liang, Zhu, and Zhang 2011).

2.7 Supervised ensemble-based tools designed to learn imbalanced data

Over-sampling can introduce noise, and the extra samples generated do not always represent the data from the minority class (also referred to as the positive class). On the other hand, under-sampling only uses a subset of the majority class and tends to eliminate potential important information from the majority class. Therefore, ensemble-based imbalanced data classification tools are developed for exploring imbalanced data while not adding in extra information or missing out on important information during the learning process (Liu, Wu, and Zhou 2008).

2.7.1 EasyEnsemble classifier

In the EasyEnsemble classifier, instances in the majority class X_N (which are referred to as 'negative' class) are independently sampled into T subsets as: $X_{N_1}, \ldots, X_{N_T}$. A classifier h_i is trained on each subset, X_{N_t}, $1 \le t \le T$ and all samples from the positive class X_P. In this case, the number of positive samples can be significantly smaller than the number of negative samples as $n_P \ll n_N$. The size of each negative subset sampled should be the same as the number of the positive samples from the positive set as $n_P = n_{N_t}$. The algorithm can be illustrated as Algorithm 1.

ALGORITHM 1: The EasyEnsemble Algorithm

```
Input: The sets of minority class samples, $X_P$, and majority class samples,
  $X_N$. The number of subsets, $T$, sampled from $X_N$. The number of
  iterations, $K$, to train an AdaBoost classifier, $H_t$.
Output: $H(x) = sign\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \alpha_{t,k} h_{t,k}(x) - \theta_t\right)$
while $t \le T$ do:
1. $t \leftarrow t + 1$
2. Randomly sample a subset $X_{N_t}$ from $X_N$, and $n_P = n_{N_t}$
3. Train an AdaBoost classifier $H_t$ using $X_{N_t}$ and $X_P$. The AdaBoost classifier
   consists of $K$ weak classifiers $h_{t,k}$ and the corresponding weight $\alpha_{t,k}$
   and threshold $\theta_t$ as:
           $H_t(x) = sign\left(\sum_{k=1}^{K} \alpha_{t,k}, h_{t,k}(x) - \theta_t\right)$
end while
```

EasyEnsemble uses balanced bootstrap samples to generate an ensemble-based classifier to process imbalanced problems. It uses minority class repeatedly and makes sure no extra information is introduced and no information is lost.

2.7.2 Balanced random forest classifier

The random forest (RF) algorithm is comparable to AdaBoost but even more resistant to noise. However, as with other machine learning algorithms, RF is designed to obtain high accuracy. Therefore, RF is biased towards the majority class in classifying imbalanced data problems.

To correct this, balanced random forest uses balanced bootstrap samples to train decision trees to solve the imbalanced problem. The details of constructing balanced random forest are illustrated in Algorithm 2.

ALGORITHM 2: Balanced Random Forest Algorithm

```
Input: The sets of minority class samples, $X_P$, and majority class samples,
  $X_N$. The number of subsets, $T$, sampled from $X_N$. The number of
  iterations, $K$, to train a random forest classifier, $H_t$.
Output: Aggregate the balanced random forest classifier, $H$, aggregated
  from all the trained trees for the final prediction.
while $t \le T$ do:
1. $t \leftarrow t + 1$
2. Randomly sample a subset $X_{N_t}$ from $X_N$, and $n_P = n_{N_t}$
3. Train a tree classifier $H_t$ using $X_{N_t}$ and $X_P$. Grow the tree to its maximum
   depth without pruning. At each node, the cost function only searches
   through a subset of variables for the optimal split rather than
   searching through all the variables.
end while
```

2.8 Supervised machine learning auto selection framework

In this architecture, a learning and predictive analytics architecture is built to select the best supervised anomaly detection classifier for processing both imbalanced and balanced datasets in five stages (Figure 3).

Stage one is to separate the input stream data collected from the manufacturing plant into two branches. The left branch is used to store the historical data records for learning purposes. The right branch is intended for testing the real-time data by using the best selected algorithm based on the pre-stored data. In stage two, the preprocessing unit collects the information for learning by cleaning the data, label encoding all categorical information to numerical values, and finally selecting the optimal feature combination for model selection. Stage three feeds the preprocessed storage data into the traditional supervised machine learning structure to evaluate the balances of the input samples and select the best regular ML model for classifying the data.

Suppose the classification results generated from traditional machine learning classifiers produce extremely low TPR or G-Mean ($\le \beta$). In that case, it either means the input data is highly imbalanced or positive class samples are hard to segment from the negative samples or both. In that respect, more advanced classifiers (hybrid classifiers and ensemble-based classifiers) designed for processing highly imbalanced data (more biased toward minority class) should be used to learn the data.

Stage four is to learn the data based on the hybrid classifiers and ensemble-based classifiers. Hybrid classifiers consist of the oversampling tool ADASYN and each of the eight traditional classifiers, which usually would generate better Sensitivity or G-Mean when input data is highly imbalanced. Ensemble-based imbalanced data classifiers are used to compare to other hybrid classifiers to develop the best final classifier. Finally, the last stage is packing and deploying the best classifier generated through stages three and four for real-time detection.

3. Experimental results

In this section, the supervised anomaly detection framework Figure 2 is applied to detect abnormal solder paste deposits and rolling element bearing anomaly detection. Then, performance metrics are used to compare the algorithms suggested in the framework.

3.1 Solder Paste Inspection (SPI)

Printed Circuit Board (PCB) assembly is a key process during the manufacturing of electronic products, in which various electronic components are connected to the bare PCB through the solder printing process. It is estimated that 50–70% of PCB defects are produced at the solder printing stage. Solder Paste Inspection (SPI) is the process of evaluating the solder paste's quality when printing and is an essential process to minimize cost of rework and maintain good product quality by detecting faults in the early stages of the production line. However, SPI is far from perfect, and it generates a low true positive rate. In this experiment, the supervised anomaly detection framework is used to train ML models that generate a decent true positive rate using the SPI data. The SPI dataset is highly imbalanced, containing only 331 positive samples compared to 199,387 negative.

In this experiment, 80% of the original data is randomly selected for training and the remaining 20% is set aside for testing, which generates 260 positive samples and 159,514 negative samples in the training set, and 71 positive samples and 39,873 negative samples in the test set. The classification performance of the algorithms is presented in Table 2.

Observe that the accuracy in both training and test sets are nearly perfect. However, this is simply due to the imbalance in the data as evidence by the very low Sensitivity (TPR). The top performing algorithm based on Sensitivity is Decision Tree at only 43.66%. For the purposes of the algorithm selection framework, the default Sensitivity threshold, β, in stage three is set as 50%, which is not satisfied by any of the proposed algorithms. This threshold was chosen because the desire is for the models to generate better classification than random guesses in balanced classification problem and generate at least 50% TPR in imbalanced problems. Since stage three does not generate the ideal best classifier, as indicated by the mediocre performance in the TPR column of Table 2, the training datasets are forwarded to stage four.

To choose between SMOTE and ADASYN as the best re-sampling tool in stage four, the experiment is set up as follows. First, both tools are used to oversample the minority class samples in the training data so that the minority class contains same number of the samples as the majority class. After oversampling the monitoring class in the training dataset, the numbers of both positive and negative instances in data are identical, numbering 159,514. Then, stage three is repeated using the balanced data. Finally, those trained algorithms are applied to the test sets. The testing results are listed in Tables 3 and 4.

Based on the experiment using SPI data, it is clear that both SMOTE and ADASYN have positive effect on the imbalance data classification. The TPR and G-Means generated by most classifiers after using those two tools are improved with the neural network models exhibiting the most dramatic improvement.

Overall, ADASYN over-sampling facilitated better detection performance for most of the classifiers, specifically Adaboost, nearest neighbor, neural network, decision tree, and bagging. Therefore, ADASYN supervised anomaly detection framework, was selected as the over-sampling tool in stage four.

In stage four, two ensemble-based tools were used, balanced random forest and EasyEnsemble, on the same training and testing datasets from stage three. The performance results are listed in Table 5.

Based on the performance of the tools discussed in stage three and four, balanced random forest gives the best classification performance for processing SPI data.

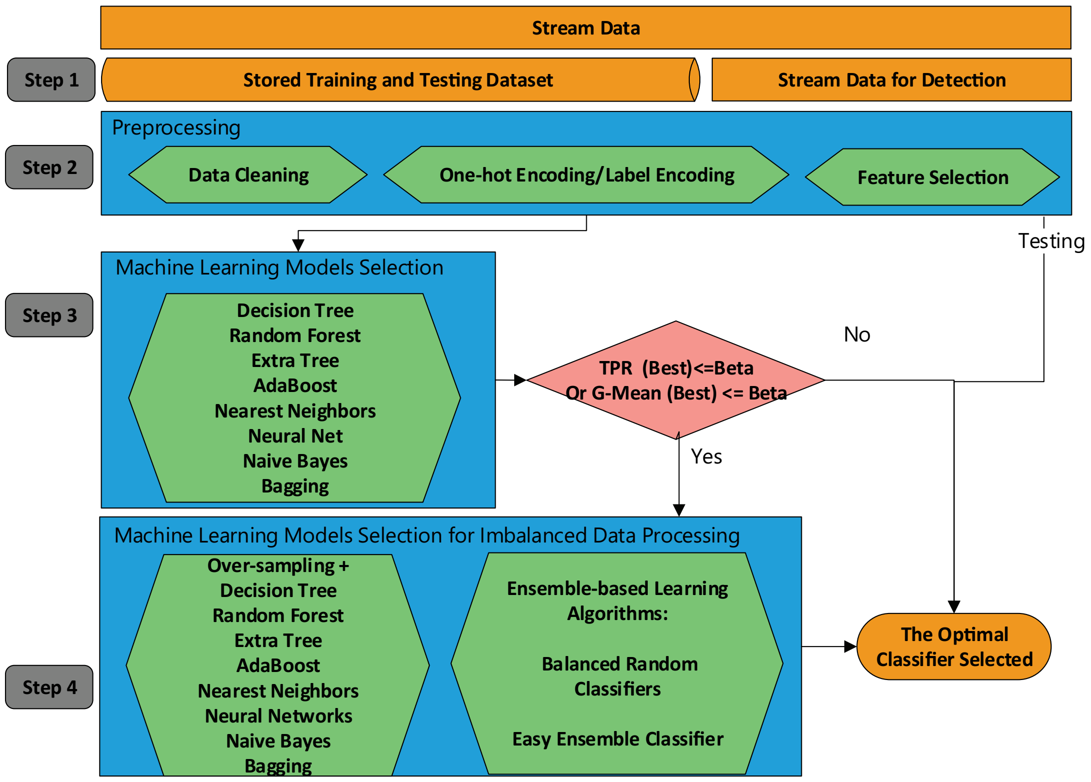

Figure 2. Proposed supervised learning and predictive analytics framework for imbalanced data processing.

3.1.2 Rolling element bearing anomaly detection

Rolling element bearing is among the most common components in rotating machinery, and bearing failure is one of the primary causes of breakdowns in rotating machinery.

The second dataset used to test this supervised anomaly detection framework is IMS bearing data collected by the National Instruments LabVIEW program supported by Rexnord Corp in Milwaukee, WI. In the test rig, a shaft hosts four test bearings, and an AC motor used to rotate the shaft was running at a constant speed at 2000 RPM with a radial load of 6000 lbs. (Qiu et al. 2006).

The data from one of the bearings (bearing 3) has an inner race defect in bearing during the test-to-failure running process. The experiment was conducted over the span of a 35-day period. In the first 30 days, there was no significant trend of bearing component damage observed, and in the later 5 days of running, there was an increase of changes that occurred significantly.

Therefore, the first 30 days of data are used as healthy (33,116,160 sample points) and the data collected in the last 5 days as unhealthy (11,038,720 sample points). Healthy and unhealthy data are then segmented with 10,000 points per row to construct the labeled data matrix (441547 rows x 101 columns with 331,161 rows of negatives and 110,386 rows of positives). 80% of the data is randomly selected from the dataset to train the models, and the remaining 20% is reserved for testing the models.

Table 2. Supervised anomaly detection for solder paste inspection by using traditional machine learning algorithms.

Algorithms	Training Accuracy	Testing Accuracy	TPR	TNR	G-Mean
Random forest	99.98%	99.84%	25.35%	99.98%	50.35%
Extra tree	99.98%	99.84%	25.35%	99.98%	50.35%
AdaBoost	99.98%	99.84%	29.58%	99.97%	54.38%
Nearest neighbor	99.88%	99.79%	15.49%	99.94%	39.35%
Neural network	99.84%	99.82%	0.00%	100.00%	0.00%
Naive Bayes	97.40%	97.23%	43.66%	97.33%	65.19%
Decision tree	99.98%	99.74%	32.39%	99.86%	56.88%
Bagging	99.97%	99.84%	28.17%	99.97%	53.07%

Table 3. Anomaly detection with traditional machine learning algorithms after using SMOTE over-sampling.

Algorithms	Training accuracy	Testing accuracy	TPR	TNR	G-Mean
SMOTE + Random Forest	99.99%	99.77%	33.80%	99.89%	58.11%
SMOTE + Extra tree	99.99%	99.82%	30.99%	99.94%	55.65%
SMOTE + AdaBoost	99.89%	99.42%	32.39%	99.54%	56.79%
SMOTE + Nearest neighbor	99.70%	99.29%	43.66%	99.39%	65.87%
SMOTE + Neural network	86.68%	90.68%	78.87%	90.71%	84.58%
SMOTE + Naive Bayes	71.23%	96.45%	47.89%	96.53%	67.99%
SMOTE + Decision tree	99.99%	99.46%	30.99%	99.58%	55.55%
SMOTE + Bagging	99.99%	99.64%	29.58%	99.76%	54.32%

Table 4. Anomaly detection for solder paste inspection with traditional machine learning algorithms after using ADASYN over-sampling.

Algorithms	Training accuracy	Testing accuracy	TPR	TNR	G-Mean
ADASYN + Random forest	99.99%	99.72%	32.39%	99.84%	56.87%
ADASYN + Extra tree	99.99%	99.81%	30.99%	99.93%	55.65%
ADASYN + AdaBoost	99.72%	99.09%	43.66%	99.19%	65.81%
ADASYN + Nearest neighbor	99.68%	99.02%	46.48%	99.11%	67.87%
ADASYN + Neural network	83.60%	90.44%	81.69%	90.45%	85.96%
ADASYN + Naive Bayes	70.39%	95.18%	54.93%	95.25%	72.33%
ADASYN + Decision tree	99.99%	99.18%	28.17%	99.31%	52.89%
ADASYN + Bagging	99.99%	99.54%	32.39%	99.66%	56.82%

Table 5. Anomaly detection using the ensemble-based tools: balanced random forest and EasyEnsemble.

Algorithms	Training Accuracy	Testing Accuracy	Sensitivity	Specificity	G-Mean
Balanced Random Forest	91.26%	90.93%	85.92%	90.94%	88.39%
EasyEnsemble	85.07%	84.68%	81.69%	84.68%	83.17%

The best-performing classifier chosen by the auto-selection framework (Figure 2) for classifying bearing conditions is determined based on the performance data listed in Table 6. In this experiment, the G-mean scores of eight algorithms in step 3 do not pass the G-mean threshold at 1.0, and the data is further passed through the model selection session in step 4. Finally, Balanced random forest is selected as it generates the best classifier with the highest G-Mean score, comparing all other models and combinations. Table 6 shows that the embedded over-sampling tool in step 4 has improved the performance of most classifiers from step 3, including random forest, AdaBoost, nearest neighbor, neural network, naive Bayes, decision tree, and bagging. It also verifies that our multi-layer model selection

Table 6. Bearing fault detection using supervised AutoAI and the best model, EasyEnsemble, is selected based on the G-mean.

Algorithms	Training accuracy	Testing accuracy	TPR	TNR	G-Mean
Random forest	100.00%	86.50%	73.77%	90.81%	81.85%
Extra tree	100.00%	84.29%	51.15%	95.51%	69.89%
AdaBoost	90.19%	86.60%	71.40%	91.75%	80.93%
Nearest neighbor	81.60%	65.92%	17.09%	82.45%	37.54%
Neural network	75.01%	74.96%	39.67%	86.91%	58.72%
Naive Bayes	75.03%	74.90%	72.59%	75.68%	74.12%
Decision tree	100.00%	79.88%	59.04%	86.93%	71.64%
Bagging	99.53%	83.15%	57.93%	91.70%	72.88%
ADASYN+Random forest	100.00%	76.54%	99.40%	68.81%	82.70%
ADASYN+Extra tree	100.00%	77.75%	98.79%	70.62%	83.53%
ADASYN+AdaBoost	90.91%	84.87%	76.23%	87.80%	81.81%
ADASYN+Nearest neighbor	86.93%	69.49%	67.24%	70.25%	68.73%
ADASYN+Neural network	74.57%	74.86%	75.36%	74.69%	75.02%
ADASYN+Naive Bayes	82.23%	75.01%	97.23%	67.49%	81.01%
ADASYN+Decision tree	100.00%	77.27%	60.42%	82.97%	70.80%
ADASYN+Bagging	99.79%	80.04%	62.77%	85.88%	73.42%
EasyEnsemble	78.25%	78.22%	97.89%	71.57%	83.70%
Balanced random forest	85.31%	79.45%	99.83%	72.56%	85.11%

framework generates better models than only using the general selection layer in step 3.

Conclusion

This research addresses a gap in the practical incorporation of advanced analytical approaches to solving abnormality detection problems in the Industry 4.0 environment. The auto-selection framework performs data cleaning, cycles through predefined classification algorithms, and outputs metrics. Subject to performance against predefined stopping criteria, the framework then cycles through oversampling techniques to mitigate the challenges posed by imbalanced data. The framework terminates with the application of ensemble approaches to the models.

The framework was applied to two datasets created from two diverse experiments, one in the domain of fault detection in PCB assembly and the other in bearing fault detection. The experiments demonstrate the flexibility of the auto-selection approach and illustrate how seemingly similar experiments might be optimally solved by a diverse array of different approaches. By performing the cycling and screening automatically, the human analyst can better dedicate his or her time to the analysis and interpretation of model results versus model creation.

Ultimately, Industry 4.0 offers an opportunity by generating big data for training better machine learning algorithms. Making good use of machine learning in IoT will significantly contribute to better quality control, automation, and collaboration between human analysts and automated tools to improve working efficiency and further advance the transition from Industry 4.0 to Industry 5.0.

Future work will explore an evaluation framework for data quality over the time to automatically determine when model retraining should take place. The robustness of models over time is highly dependent on many factors, and the natural next step to this research is to explore the degree to which this decision can be supported by an automated and standardized framework versus the ad hoc approach that is typically employed at present time.

Acknowledgements

This research was supported by Rockwell Automation.

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCID

Meiling He http://orcid.org/0000-0003-1759-4223
Phillip LaCasse http://orcid.org/0000-0003-2351-0372

References

Bottou, L. 2012. “Stochastic Gradient Descent Tricks.” *Neural Networks: Tricks of the Trade* 7700: 421–436.

Breiman, L. 1996. “Bagging Predictors.” *Machine Learning* 24 (2): 123–140. doi:10.1007/BF00058655.

Brundage, M. P., T. Ademujimi, and B. Rakshith. 2017. “Smart Manufacturing Through a Framework for a Knowledge-Based Diagnosis System.” *International Manufacturing Science and Engineering Conference*, Los Angeles, California, USA. American Society of Mechanical Engineers.

Cinar, Z., A. Nuhu, Q. Zeeshan, O. Korhan, M. Asmael, and B. Safaei. 2020. “Machine Learning in Predictive Maintenance Towards Sustainable Smart Manufacturing in Industry 4.0.” *Sustainability* 12 (19): 8211. doi:10.3390/su12198211.

David, R. E., G. E. Hinton, and R. J. Williams. 1986. “Learning Representations by Back-Propagating Errors.” *Nature* 323 (6088): 533–536. doi:10.1038/323533a0.

Fathy, Y., M. Jaber, and A. Brintrup. 2020. “Learning with Imbalanced Data in Smart Manufacturing: A Comparative Analysis.” *IEEE Access* 9: 2734–2757. doi:10.1109/ACCESS.2020.3047838.

Fernández, A., S. García, F. Herrera, and N. V. Chawla. 2018. “SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-Year Anniversary.” *The Journal of Artificial Intelligence Research* 61: 863–905. doi:10.1613/jair.1.11192.

Freund, Y., and R. E. Schapire. 1997. “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.” *Journal of Computer and System Sciences* 55 (1): 119–139. doi:10.1006/jcss.1997.1504.

Geurts, P., D. Ernst, and L. Wehenkel. 2006. “Extremely Randomized Trees.” *Machine Learning* 63 (1): 3–42. doi:10.1007/s10994-006-6226-1.

Goldberger, J., S. Roweis, G. Hinton, and R. Salakhutdinov. 2004. “Neighbourhood Components Analysis.” *Advances in Neural Information Processing Systems* 17: 513–520.

Guzella, T. S., and W. M. Caminhas. 2009. "A Review of Machine Learning Approaches to Spam Filtering." *Expert Systems with Applications* 36 (7): 10206–10222. doi:10.1016/j.eswa.2009.02.037.

Hansson, K., S. Yella, M. Dougherty, and H. Fleyeh. 2016. "Machine Learning Algorithms in Heavy Process Manufacturing." *American Journal of Intelligent Systems* 6: 1–13.

He, H., Y. Bai, E. A. Garcia, and S. Li. 2008. "ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning." *IEEE international joint conference on neural networks*, Hong Kong.

He, H., and E. A. Garcia. 2009. "Learning from Imbalanced Data." *IEEE Transactions on Knowledge and Data Engineering* 21 (9): 1263–1284. doi:10.1109/TKDE.2008.239.

Kang, P., D. Kim, and S. Cho. 2016. "Semi-Supervised Support Vector Regression Based on Self-Training with Label Uncertainty: An Application to Virtual Metrology in Semiconductor Manufacturing." *Expert Systems with Applications* 51: 85–106. doi:10.1016/j.eswa.2015.12.027.

Koksal, G., İ. Batmaz, and M. C. Testik. 2011. "A Review of Data Mining Applications for Quality Improvement in Manufacturing Industry." *Expert Systems with Applications* 38 (10): 13448–13467. doi:10.1016/j.eswa.2011.04.063.

Kozjek, D., R. Vrabic, B. Rihtarsic, N. Lavrac, and P. Butala. 2020. "Advancing Manufacturing Systems with Big-Data Analytics: A Conceptual Framework." *International Journal of Computer Integrated Manufacturing* 33 (2): 169–188. doi:10.1080/0951192X.2020.1718765.

LaCasse, P., W. Otieno, and F. Maturana. 2018. "A Hierarchical, Fuzzy Inference Approach to Data Filtration and Feature Prioritization in the Connected Manufacturing Enterprise." *Journal of Big Data* 5 (1): 1–31. doi:10.1186/s40537-018-0155-2.

LaCasse, P., W. Otieno, and F. Maturana. 2019. "A Survey of Feature Set Reduction Approaches for Predictive Analytics Models in the Connected Manufacturing Enterprise." *Applied Sciences* 9 (5): 843. doi:10.3390/app9050843.

LaCasse, P., W. Otieno, and F. Maturana. 2020. "Predicting Contact-Without-Connection Defects on Printed Circuit Boards Employing Ball Grid Array Package Types: A Data Analytics Case Study in the Smart Manufacturing Environment." *SN Applied Sciences* 2 (2): 1–13. doi:10.1007/s42452-019-1924-z.

Lee, J., and S. Ha. 2009. "Recognizing Yield Patterns Through Hybrid Applications of Machine Learning Techniques." *Information Sciences* 179 (6): 844–850. doi:10.1016/j.ins.2008.11.008.

Lee, D. -H., J. -K. Yang, C. -H. Lee, and K. -J. Kim. 2019. "A Data-Driven Approach to Selection of Critical Process Steps in the Semiconductor Manufacturing Process Considering Missing and Imbalanced Data." *Journal of Manufacturing Systems* 52: 146–156. doi:10.1016/j.jmsy.2019.07.001.

Leo, B. 2001. "Random Forests." *Machine Learning* 45: 5–32.

Liang, G., X. Zhu, and C. Zhang. 2011. "An Empirical Study of Bagging Predictors for Different Learning Algorithms." *Proceedings of the AAAI Conference on Artificial Intelligence*, San Francisco, California USA, (p. 1).

Liu, C., H. Li, Y. Tang, D. Lin, and J. Liu. 2019. "Next Generation Integrated Smart Manufacturing Based on Big Data Analytics, Reinforced Learning, and Optimal Routes Planning Methods." *International Journal of Computer Integrated Manufacturing* 32 (9): 820–831. doi:10.1080/0951192X.2019.1636412.

Liu, X. -Y., J. Wu, and Z. -H. Zhou. 2008. "Exploratory Undersampling for Class-Imbalance Learning." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39 (2): 539–550. doi:10.1109/TSMCB.2008.2007853.

Loh, W. -Y. 2011. "Classification and Regression Trees." *Wiley Interdisciplinary Reviews Data Mining and Knowledge Discovery* 1 (1): 14–23. doi:10.1002/widm.8.

Lu, S. 1990. "Machine Learning Approaches to Knowledge Synthesis and Integration Tasks for Advanced Engineering Automation." *Computers in Industry* 15 (1–2): 105–120. doi:10.1016/0166-3615(90)90088-7.

Maddikunta, P., Q. -V. Pham, B. Prabadevi, N. Deepa, K. Dev, T. Reddy Gadekallu, and M. Liyanage. 2021. "Industry 50: A Survey on Enabling Technologies and Potential Applications." *Journal of Industrial Information Integration* 26 100257.

Moyne, J., and J. Iskandar. 2017. "Big Data Analytics for Smart Manufacturing: Case Studies in Semiconductor Manufacturing." *Processes* 5 (3): 39. doi:10.3390/pr5030039.

Oksuz, K., B. Cam, S. Kalkan, and E. Akbas. 2020. "Imbalance Problems in Object Detection: A Review." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43 (10): 3388–3415. doi:10.1109/TPAMI.2020.2981890.

Pittino, F., M. Puggl, T. Moldaschl, and C. Hirschl. 2020. "Automatic Anomaly Detection on In-Production Manufacturing Machines Using Statistical Learning Methods." *Sensors* 20: 2344.

Qiu, H., J. Lee, J. Lin, and G. Yu. 2006. "Wavelet Filter-Based Weak Signature Detection Method and Its Application on Rolling Element Bearing Prognostics." *Journal of Sound and Vibration* 289 (4–5): 1066–1090. doi:10.1016/j.jsv.2005.03.007.

Silla, C. N., Jr., and A. A. Freitas. 2011. "A Survey of Hierarchical Classification Across Different Application Domains." *Data Mining and Knowledge Discovery* 22 (1–2): 31–72. doi:10.1007/s10618-010-0175-9.

Tao, F., Q. Qi, A. Liu, and A. Kusiak. 2018. "Data-Driven Smart Manufacturing." *Journal of Manufacturing Systems* 48: 157–169. doi:10.1016/j.jmsy.2018.01.006.

Wuest, T., D. Weimer, C. Irgens, and K. -D. Thoben. 2016. *Machine Learning in Manufacturing: Advantages, Challenges, and Applications*. 4, 23–45. Production & Manufacturing Research.

Wu, D., C. Jennings, J. Terpenny, R. Gao, and S. Kumara. 2017. "A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests." *Journal of Manufacturing Science and Engineering*, Vol. 139, 7, p. 071018.

Yan Chan, K., C. Kwong, P. Wongthongtham, H. Jiang, C. K. Fung, and B. Abu-Salih. 2018. "Affective Design Using Machine Learning: A Survey and Its Prospect of Conjoining Big Data." *International Journal of Computer Integrated Manufacturing* 33 (7): 645–669. doi:10.1080/0951192X.2018.1526412.

Zhang, X., Q. He, H. Zhang, Z. Jiang, and Y. Wang. 2022. "Big Data-Based Research on Active Remanufacturing Comprehensive Benefits Evaluation of Mechanical Product." *International Journal of Computer Integrated Manufacturing* 1–21. doi:10.1080/0951192X.2022.2128214.