Product failure detection for production lines using a data-driven model

Expert Systems with Applications

Kang, Ziqiu; Catal, Cagatay; Tekinerdogan, Bedir

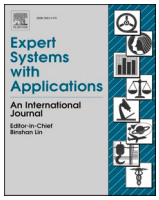
https://doi.org/10.1016/j.eswa.2022.117398

Expert Systems With Applications 202 (2022) 117398

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Product failure detection for production lines using a data-driven model

Ziqiu Kang [a,b], Cagatay Catal [c,*], Bedir Tekinerdogan [a]

[a] Wageningen University & Research, Information Technology Group, Wageningen, The Netherlands
[b] Chinese Academy of Agricultural Sciences, Institute of Urban Agriculture, Chengdu 610213, China
[c] Qatar University, Department of Computer Science and Engineering, Doha, Qatar

ARTICLE INFO

Keywords:
Machine learning
Production lines
Data analytics
Data mining
Product failure detection

ABSTRACT

For a healthy production line, it is essential to ensure a low failure rate of products. Product quality in production lines can be inspected using several techniques at the end of a production process, including a manual inspection. Different methods are applied to inspect the product quality at the end of the production process and sometimes during the production. This is often done using manual inspection, but this is less efficient, expensive, and time-consuming. Machine learning algorithms have the potential for evaluating and predicting product quality in a production line. In this paper, a novel product failure detection model that applies ANOVA (Analysis of Variance) feature selection method, Min-Max Scaling normalization method, mean imputation technique, Random Forest classification algorithm, a data sampling technique, and Grid Search parameter optimization approach is proposed and validated. For the comparison of the proposed model, several experiments have been performed using five classification algorithms, including RUSBoosted Tree. Experimental results demonstrated that the proposed model using the Random Forest algorithm, ANOVA feature selection, and sampling method achieves the best performance among other models and detects the faulty products effectively. It was also shown that the RUSBoosted Tree algorithm can be considered by practitioners for building the faulty product prediction model when data sampling and feature selection techniques are not integrated into the prediction model.

1. Introduction

For a healthy production line, it is essential to ensure a low failure rate of products. Very often, defective products are inevitable, which results in products that are of no use; that is, these cannot be sold or passed to the next production stage. This lack of quality can create a tremendous economic loss for the production line business. The high failure rate can also lead to waste and unnecessary energy consumption because defective products are regularly deposed as waste materials. As lives are increasingly dependent on industrial products, expectations for high-quality products are rising day by day.

Because of these reasons, an effective quality control strategy is increasingly essential for production line management. Different methods are applied to inspect the product quality at the end of the production process and sometimes during the production. This is often done using manual inspection, but this is less efficient, expensive, and time-consuming. Professional quality test equipment can be adopted, but this may require substantial adjustment of the production line and also, a high up-front investment. One solution is to perform an inspection for each processing step and ship only the products that pass all inspections (Chun, 2016). Nevertheless, failures still occur due to unsatisfactory inspections, poor quality standards, and variations in the environment (Kang et al., 2018). This results in customer dissatisfaction and warranty claim costs (Yang, 2008). It is indeed challenging to identify product failures from a production line using low-cost and high-efficient approaches.

To overcome the above problems of manual inspection, predictive analysis has been increasingly applied in different application domains (Köksal et al., 2011; Choudhary et al., 2009; Kusiak, 2006). These prediction models predict the defective products and market failure rate decrease (Lughofer et al., 2017). Also, the analysis of significant variables in models can help to identify the root causes of faults (Kang et al., 2018), which can help to improve the quality of future products.

Machine learning algorithms have the potential for evaluating and predicting product quality in a production line. In production lines, most equipment types generate a large amount of data. Product failures tend to generate outlier data in production lines. To this end, machine learning can use this generated data to build a prediction model. In this case, no extra modification for the production line or additional labor is required.

* Corresponding author.
E-mail addresses: ziqiu.kang@wur.nl (Z. Kang), ccatal@qu.edu.qa (C. Catal), bedir.tekinerdogan@wur.nl (B. Tekinerdogan).

https://doi.org/10.1016/j.eswa.2022.117398
Received 30 October 2021; Received in revised form 17 March 2022; Accepted 25 April 2022
Available online 27 April 2022

Raw production data is different than other types of data used in predictive modeling studies. For instance, they may often consist of many missing values (Kang et al., 2017) due to the environment (Kwak & Kim, 2012). Furthermore, data is highly unbalanced because most of the data points belong to the majority class (i.e., non-faulty production data), and very few data points belong to the minority class (i.e., faulty production data). This characteristic of the data requires additional methods to apply.

The main motivation of this study is to reduce the cost of faulty product detection systems by automating the faulty product prediction process. Although several prediction models have been investigated in the literature to automate this task, most of these studies considered only a few contributing factors and/or algorithms (e.g., feature selection technique or data balancing algorithm) for building a highly accurate model. This study aims to utilize many different feature engineering techniques to propose a specific faulty product prediction framework for a particular case study (i.e., a semiconductor manufacturing process), however, a similar experimental setup must be designed easily for different major industries. In a recent survey paper (Kang et al., 2020), we showed that product failure detection is one of the most important problems using machine learning in production lines and, classification and anomaly detection approaches are mostly applied to address this problem. In the same survey study (Kang et al., 2020); we observed that preventive maintenance, which requires the prediction of Remaining Useful Life (RUL), is also one of the problems related to the production lines and is difficult to measure in practice. As such, we developed a machine learning-based novel RUL prediction approach recently (Kang et al., 2021) and used the run-to-failure data of similar jet engines to predict the failures. Due to our above-mentioned recent research papers and motivation in this field, we aimed to design a faulty product prediction model.

In this current study, several machine learning algorithms and relevant techniques are applied to build a novel model, and experimental results are presented. A normalization technique (i.e., MinMax Scaler), a data imputation technique for missing values (i.e., mean imputation), feature selection technique (i.e., ANOVA), classification algorithms, parameter optimization technique (i.e., GridSearch), and data balancing approaches (i.e., over-sampling and under-sampling) were investigated to improve the performance of classification algorithms.

This study provides practitioners an insight into the potential of machine learning approaches in product quality control. For researchers, this study provides an overview of the strength and weaknesses of different machine learning algorithms in product failure detection.

In this study, as the dataset is highly imbalanced, special algorithms such as the RUSBoosted Trees (RUSBT) algorithm and data balancing techniques were investigated. Several machine learning algorithms were investigated and integrated to build a highly accurate prediction model. Also, several evaluation metrics were evaluated to measure the performance of the prediction model.

The main contributions of this study are specified as follows:

1. We presented a novel machine learning-based methodology to detect faulty products and compared the performance with other machine learning-based models.
2. It was demonstrated that the faulty product detection system can be automated using machine learning algorithms and highly accurate prediction models can be built.
3. The ANOVA feature selection technique improved the performance of the prediction model.
4. Data sampling techniques improved the performance of the proposed model.
5. It was shown that simpler prediction models can be built using the RUSBoosted Tree algorithm and therefore, practitioners can consider the implementations using this algorithm before investigating more complex models.
6. The proposed methodology is flexible to be adapted to different major industries and similar experiments can be performed to achieve the highly accurate faulty product detection models.

The remainder of this paper is organized as follows: Section 2 presents the background and related work. Section 3 explains the analysis of the data, and Section 4 discusses the methodology applied in this study. Section 5 shows the experimental results. Section 6 presents the discussion, and Section 7 explains the conclusion of this paper.

2. Background and related work

According to Miljkovic's survey (2011), there are three categories of defects detection methods, which are explained as follows:

- *Process model-based methods:* These methods compare the output of the measuring system with the output of the mathematic model. Then, the residue of the comparison is used to adjust and improve the mathematic model. Many studies apply different process model-based methods, including Parity Equations (Frank, 1990), state observers (Isermann, 2005), and parameter estimation (Isermann, 2006; Venkatasubramanian, Rengaswamy, Yin, & Kavuri, 2003).
- *Knowledge-based methods:* These methods, which are rule-based, mainly rely on expert knowledge. This kind of model is easy to interpret and runs fast. However, they are inflexible and expensive to maintain. An online fault diagnosis system that applies this method is discussed in the paper of Angeli (2010). According to Miljkovic (2011), expert knowledge-based methods are more suitable to be used in well-defined processes.
- *Data-driven methods:* These methods can be categorized into signal analysis, spectrum analysis, and pattern analysis. Isermann (2006) presented some examples that identify faulty products by analyzing the normal and faulty signals from sensors.

The novel machine learning-based approach that was developed in this study can be considered as a data-driven method because different patterns in the dataset are discovered with the help of machine learning techniques, and numerous data points are used during the training phase. Four types of machine learning techniques exist in the literature, namely supervised learning, unsupervised learning (e.g., clustering algorithms), semi-supervised learning, and reinforcement learning. In different studies, several algorithms were proposed and used from these categories for modelling and classification purposes. Recently, Borlea et al. (2021) used Fuzzy C-means and K-means clustering algorithms in a unified form and implemented this novel model in a distributed platform. Jodas et al. (2020) developed a classification model using machine learning techniques. Pozna and Precup (2014) proposed an approach for expert systems modelling. These successful modelling applications in various fields have emerged in recent years. This type of machine learning-based studies helped the transformation of the manufacturing sector and increased the use of digitalization within the context of Industry 4.0 (Catal and Tekinerdogan, 2019).

From the machine learning perspective, our problem is a binary classification task because there exist two kinds of classes, namely passing and failing products. In a research study, K-Nearest Neighbour (KNN), a supervised learning approach, was applied to identify the faulty products in a semiconductor production line (Verdier & Ferreira, 2011). Russ et al. (2005) applied the dynamic growing self-organizing map to separate the good and faulty wafers from raw production data.

However, datasets are mostly imbalanced in this problem, and this characteristic of the datasets makes the problem more complex. During the production process of products, most of the products are normal, and only a small portion of the data points belongs to the faulty class. To this end, models cannot have sufficient data that belongs to the positive class (i.e., minority class that consists of failing data), and much more data exist that belongs to the negative class (i.e., majority class that consists

of passing data). For example, during the semiconductor manufacturing process, a product might be produced faulty and therefore, these products are labeled as positive data instances in the dataset. If the product that is produced is not faulty, it is labeled as a negative class. To handle imbalanced data problems, data sampling techniques can be used to balance the dataset so that the model can learn from a dataset, including a similar number of positive and negative data points. Some data balancing algorithms (i.e., SMOTE, ClusterSMOTE, BorderlineSMOTE, ADASYN) and classification algorithms (i.e., RUSBoosted) were designed to address this problem.

To deal with the imbalance sample structure, the following two approaches are mostly applied (Chen, Liaw, & Breiman, 2004):

- *Cost-sensitive learning:* This method increases the positive class' weight during the training process so that more penalty is applied to the misclassification of positive samples.
- *Sampling:* This category of approaches uses sampling techniques to balance the class distribution.

There are mainly two kinds of sampling methods that are over-sampling and under-sampling. Veni (2018) explained the pros and cons of these two sampling methods. The over-sampling method replicates the minority class samples so that the number of data points in each class is similar to each other. The advantage of the over-sampling method is that no information is lost during the sampling. The disadvantage of over-sampling is that it is prone to overfitting. Opposite to the over-sampling, under-sampling balances the data by removing the data points that belong to the majority class so that the number of data points in each class is similar to each other. It avoids overfitting better than overfitting; however, it is more suitable for a large dataset because some of the data points in the dataset are lost while balancing the distribution of classes.

Recently, deep learning, which is a sub-branch of machine learning, has been applied in this domain. Li et al. (2019) applied the Deep Neural Network (DNN) algorithm for degradation assessment in mechanical equipment and demonstrated that the proposed approach outperforms other traditional machine learning-based approaches. Wang et al. (2016) developed a Deep Neural Network-based framework for the identification of the health of wind turbine (WT) gearboxes and showed that the DNN model provides the best performance among six machine learning algorithms. Khumprom and Yodo (2019) used the Deep Neural Networks approach to predict the State of Health (SoH) of the lithium-ion battery and showed that they provide better performance than the traditional machine learning algorithms. Li et al. (2020a) applied the Deep Belief Network (DBN) algorithm for predicting the backlash error that can affect the geometry of the components and showed that their model provides superior performance in machining centers. Iqbal et al. (2019) proposed a Fault Detection and Isolation (FDI) system using deep learning algorithms and demonstrated that their approach can locate several faults under real-time working conditions. Liong et al. (2020) designed a leather defect detection technique by employing a deep learning approach and showed its effectiveness on a real dataset. A modified AlexNet architecture was used for feature extraction and the U-net architecture was applied for segmentation. Liang et al. (2018) developed a DNN model for detecting energy anomalous patterns during the aluminum extrusion process and applied a transfer learning approach effectively.

Despite deep learning algorithms providing superior performance for failure detection, there are several drawbacks to these approaches. For example, these algorithms require additional expertise, required computational power is higher, hyperparameter tuning is mostly needed, and a lot of data must be collected for achieving high performance. In order to avoid these drawbacks, we decided to develop a novel machine learning model, which is highly accurate but does not include these drawbacks. Also, nowadays every major industry is using automation for faulty product detection, however, mostly the existing solutions are quite costly for organizations. In this study, we aimed to reduce the cost of faulty product detection systems by applying machine learning algorithms.

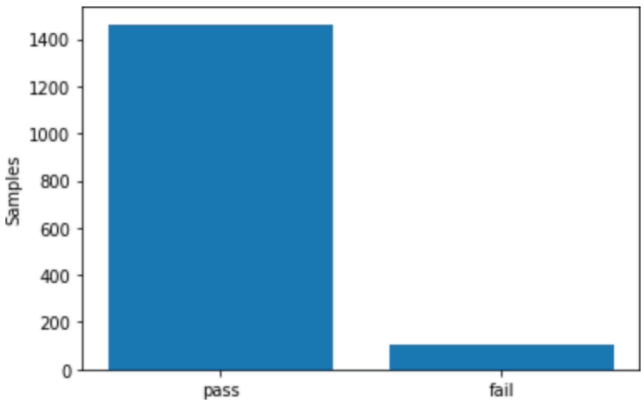


Fig. 1. The Number of Passing and Failing Samples in the Dataset.

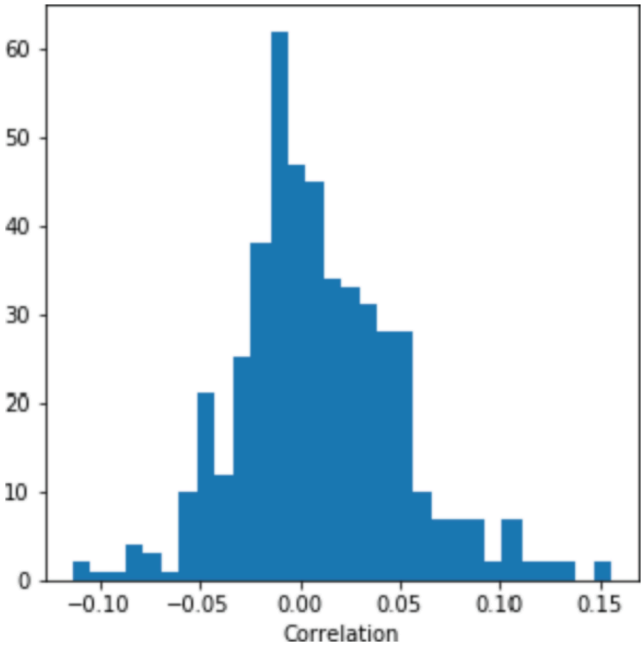


Fig. 2. The Correlation Index between Features and the Dependent Variable.

3. Data analysis

In this study, a dataset that contains sensory data from a semiconductor manufacturing process (McCann & Johnston, 2008) is used, and the task is to identify the product failure from the production line.

The dataset has 1567 samples with 592 features, including 591 sensor features (i.e., float data types) and a time feature. Each record of the dataset is labeled by −1 for failing products and by 1 for passing products. A data instance of the dataset (i.e., only 15 features of the first data point) is presented as follows:

[3030.93, 2564, 2187.7333, 1411.1265, 1.3602, 100, 97.6133, 0.1242, 1.5005, 0.0162, −0.0034, 0.9455, 202.4396, 0, 7.9558, …].

There are four challenges that need to be addressed in this dataset. The first challenge of this dataset is that the feature-sample ratio is too high. The number of features is one-third of the number of data points, which means that there is less information to train on each feature. The second challenge is that the dataset is highly imbalanced, as shown in Fig. 1. There are 1463 normal samples, while there are only 104 failure

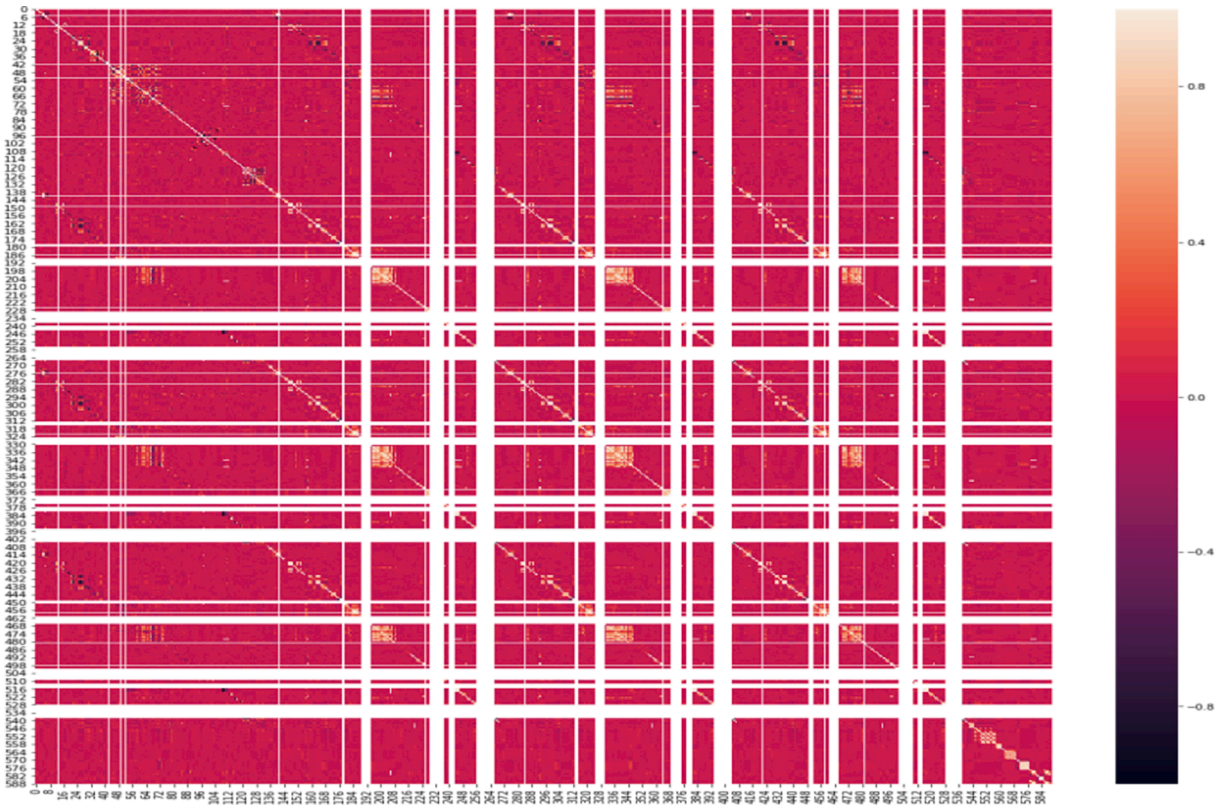


Fig. 3. The Heatmap of the Features in the Dataset.

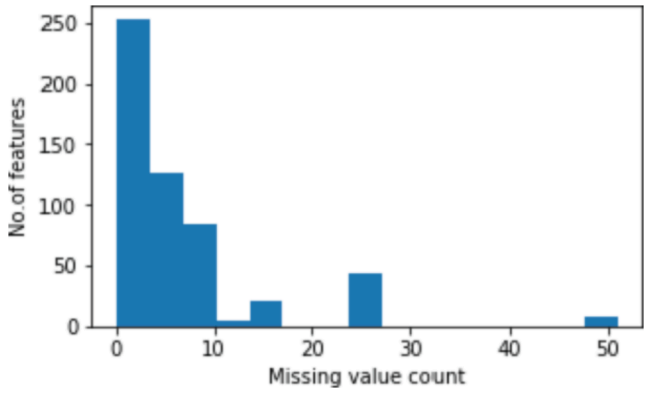


Fig. 4. Feature Count Distribution for a Missing Value Smaller than 10% of the Total Data Size.

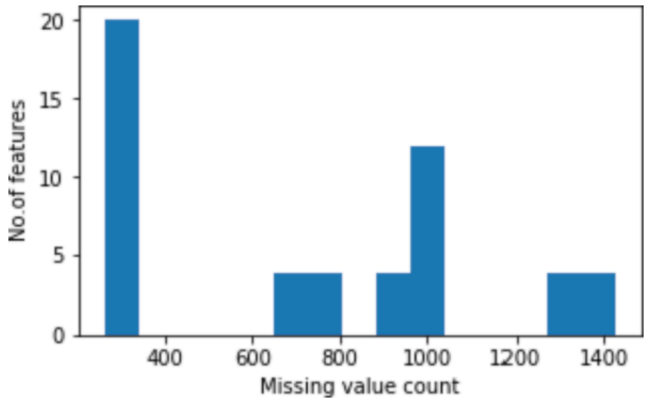


Fig. 5. Feature Count Distribution for a Missing Value Greater than 10% of the Total Data Size.

samples.

The third challenge is that many of the features in the dataset are noisy or irrelevant signals. As shown in Fig. 2, only 3.2% of features have an absolute correlation greater than 0.1. Fig. 3 also shows that the correlation among most features is very weak. Most of the correlation indexes are between −0.1 and 0.1. As such, it can be concluded that the majority of the features are noisy or irrelevant for the dependent variable.

The last challenge is that the dataset contains a lot of missing values. 538 features have missing values smaller than 10% of the data size, as shown in Fig. 4, and 52 features have missing values greater than 10% of the data size, as shown in Fig. 5.

Fig. 6 shows the distributions of feature values. The scales of values are very different among features. Some sensors are on a scale of thousand, while some features are on a scale of decimal. Additionally, some features have a significant tail such as sensor 589, which can be useful to identify the outliers. Also, as shown in Fig. 7, there are 116 features (16% of features) with a constant value, which is not helpful for machine learning models because they cannot help to distinguish the data points.

Before presenting our methodology, we establish the problem statement as follows (He & Garcia, 2009).

Classification problem: Given a training set **T** including **n** number of data points, the following definitions are presented. $\mathbf{T} = \{(\mathbf{x_i}, \mathbf{y_i})\}$, $i = 1, 2, \ldots, n$, where $\mathbf{x_i} \in \mathbf{X}$ is an instance in the m-dimensional feature space $\mathbf{X} = \{\mathbf{f_1}, \mathbf{f_2}, \ldots, \mathbf{f_m}\}$ and $\mathbf{y_i} \in \mathbf{Y} = \{\mathbf{0}, \mathbf{1}\}$ is the class label of the x_i. Also, the following subsets are defined in this problem. $\mathbf{T_{min}} \subset \mathbf{T}$ and $\mathbf{T_{maj}} \subset \mathbf{T}$, where T_{min} represents the minority class data points in T, and T_{maj} represents the majority class data points such that $\mathbf{T_{min}} \cap \mathbf{T_{maj}} = \{\Phi\}$ and $\mathbf{T_{min}} \cup \mathbf{T_{maj}} = \mathbf{T}$. We must find a classification model that can effectively classify data points in T using T_{min} and T_{maj} data points.

In this study, Support Vector Classifier, Multilayer Perceptron,

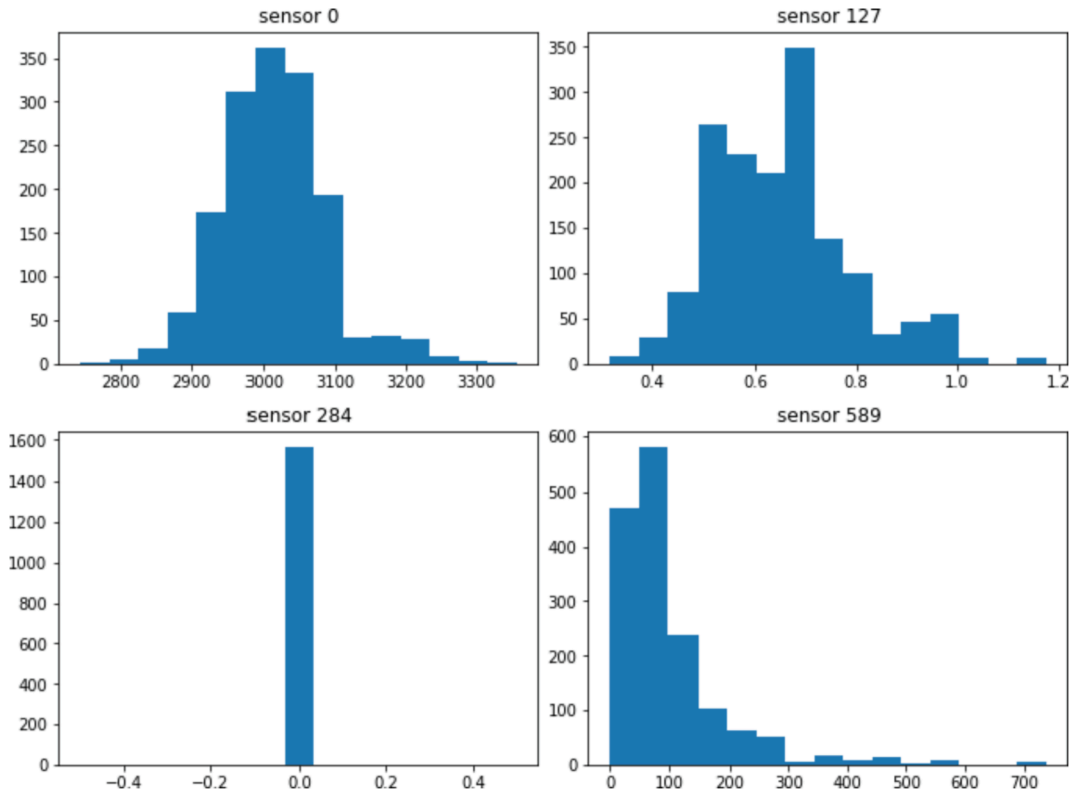


Fig. 6. Distribution of Feature Values for Sensors 0, 127, 284, and 589.

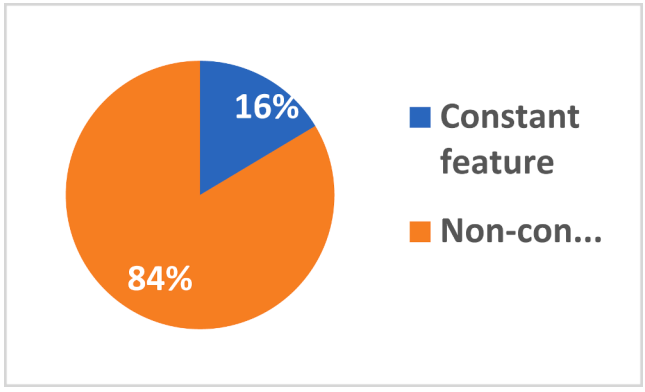


Fig. 7. Distribution of Features.

Random Forest, Gradient Boosted Trees, and RUSBoostedTree classification algorithms together with other feature engineering techniques are investigated to solve this challenging problem.

4. Methodology

An overview of the methodology is presented in Fig. 8. As shown in this figure, data pre-processing, feature selection, data sampling, parameter optimization techniques are applied together with the machine learning algorithm. First, the dataset is divided into training and testing sets using the hold-out approach (i.e., 80% for training and 20% for testing). The data of the training set is pre-processed before training the models. During the pre-processing stage, the mean imputation method is applied for replacing the missing values, and the Min-Max scaling normalization method is used for the normalization of the data. For feature selection, the ANOVA (Analysis of Variance) feature selection (FS) approach is applied to select the most relevant features. For sampling, over-sampling (OS) and under-sampling (US) techniques are investigated to improve the performance of the algorithms. For parameter optimization, the Grid Search optimization technique is used, and as such, hyper-parameters are tuned accordingly. To evaluate the performance of the model, the trained model is tested with the testing set. Five classification algorithms, namely Support Vector Classifier (SVC), Multilayer Perceptron (MLP), Random Forest (RF), Gradient Boosted Trees (GDBT), and RUSBoostedTree (RUSBT) are applied during experiments. As baseline models, all the machine learning algorithms are initially applied without the use of feature selection and sampling techniques. Later, these techniques are also applied to investigate the performance variation of the models. In the following subsections, the data pre-processing step, classification algorithms, feature selection approach, data sampling techniques, model configuration, parameter optimization stage, and evaluation strategy are explained.

4.1. Data PreProcessing

According to data analysis, a large number of features have a high portion of missing values. These features do not provide much useful information to the machine learning models, and even they require extra computational power. Replacing missing values with zero or mean values is an approach to handle the missing values, but features with too many missing values can have a negative impact on the prediction models. To this end, features with more than 80% missing value samples are dropped from the dataset. Furthermore, the time feature is removed as the quality of the product is not dependent on the time in this case study. As shown in Table 1, 125 features are removed from the original dataset, and hence, the adjusted dataset includes 466 features at the end of this process.

After dropping features with substantial missing values, there are still missing value cells in the dataset. As such, the mean imputation method is used to replace all missing values with the mean of the corresponding feature, and a dataset without missing values is generated. Because no separate testing set is provided for this analysis, a testing

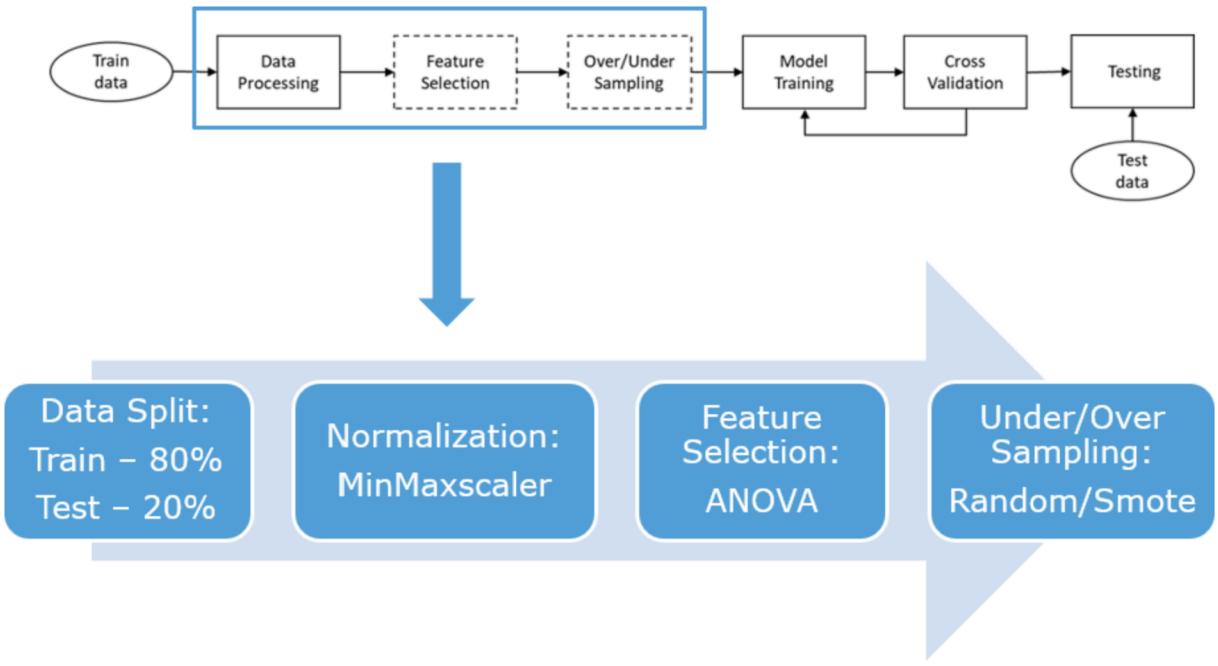


Fig. 8. The Methodology of the Proposed Machine-Learning based Failure Detection Model.

Table 1
Summary of the Training Data and Testing Data.

Original dataset			
No. of samples	1567	No. of features	591
Training dataset			
No. of samples	1253	No. of features	466
No. of failure samples	91	Failure ratio	7%
Testing dataset			
No. of samples	314	No. of features	466
No. of failure samples	13	Failure ratio	4%

dataset is created from the original dataset. 20% of the dataset is randomly sampled from the dataset as a testing set, and the remaining part is used as the training set. The number of data points in each set is presented in Table 1.

The value ranges and scales of features are very different in the dataset, which makes machine learning algorithms difficult to converge based on the loss function. To improve the performance of the machine learning models, normalization is implemented to remove the negative impact of the diverse value scales and value ranges. The Min-Max Scaling is used to normalize the feature value to the range of [0, 1], as shown in Eq. (1).

$$x^{`} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

4.2. Classification algorithms

In this study, the following classification algorithms were utilized: Support Vector Classifier (SVC), Multilayer Perceptron (MLP), Random Forest (RF), Gradient Boosted Trees (GDBT), and RUSBoostedTree (RUSBT) algorithms.

Support Vector Classifier (SVC) is a Support Vector Machine (SVM) implementation that performs the classification task. It has a low computational cost and is effective in classifying high-dimensional data with a low sample size. It performs well when the data separation margin is large. Eq. (2) shows the linear form of the SVM algorithm.

$$f(x) = w^{Tr}.x + b \tag{2}$$

x represents the input variable, w shows the weight, b is the bias, and Tr represents the transpose. The algorithm aims to minimize the error that must be within the permissible range (∈). The optimization problem can be formulated as shown in Eq. (3) (Banadkooki et al., 2019). C represents the penalty factor, ξ_i^- and ξ_i^+ are penalties related to the training data, w shows the weight, y is the output variable, and X is the input variable.

$$\begin{aligned} &Minimize \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\left(\xi_i^- + \xi_i^+\right) \\ &subject(to)(w_i \cdot x_i + b) - y_i < \varepsilon + \xi_i^+ \end{aligned} \tag{3}$$

Multilayer Perceptron (MLP) neural network has been widely applied in recent years due to the state-of-the-art results of deep learning algorithms for many problems and the increasing computational power of systems. However, neural network algorithms require a large amount of data for training. Eq. (4) shows how the MLP algorithm works. w_j represents the weight coefficient of neurons, f_j is the activation function of the corresponding neuron, n represents the number of neurons in input layers, m shows the number of neurons in hidden layer, $x_i(k)$ represents the input variable, T is the transpose of the matrix, and w_0 shows the bias of the output layer (Banadkooki et al., 2019).

$$y(k) = \sum_{j=1}^{m} w_j f_j \left(\sum_{j=1}^{n} w_{ij} x_i(k) + w_0 \right) \tag{4}$$

Random Forest (RF) is a tree-based method and applies to bootstrap during the training process. RF (Cutler, Cutler, & Stevens, 2011) consists of multiple trees, and each tree only uses a subset of all the features. Each tree generates a prediction, and the final prediction is an aggregation of all predictions. The RF has three major tuning parameters, which are the tree size, number of predictor variables, and tree depth. RF can also deal with imbalanced data by adjusting the weight of each class. The performance of RF generally outperforms single tree prediction algorithms (Chen et al., 2004). Chen et al. (2004) used both weighed RF and balanced RF to classify imbalanced data, and their results outperform other algorithms. The decision function of the RF algorithm is presented in Eq. (5). H(x) shows the combination of the classification models, h_i represents a single decision tree model, I is the indicator function, and Y shows the output variable (Liu et al., 2012).

Table 2
Percentile and Number of Features Selected for Training.

	Percentile (%)	No. of Features
RF + Over-sampling	80	372
RF + Under-sampling	30	117
GDBT + Over-sampling	50	233
GDBT + Under-sampling	30	117
SVC + Over-sampling	30	117
SVC + Under-sampling	20	93
RUSBT	40	186

$$H(x) = \underset{Y}{\operatorname{argmax}} \sum_{i=1}^{k} I(h_i(x = Y)) \tag{5}$$

Gradient Boosted Trees (GDBT) algorithm converts a series of weak learners into stronger learners (Natekin & Knoll, 2013). Each tree of GDBT improves the prediction from the result of the previous tree algorithms. This makes the GDBT highly flexible and increasingly popular in recent years. GDBT has been largely used to perform prognostic genetic tasks (Teramoto, 2009) that are normally imbalanced classification problems. Teramoto's study (2009) shows that GDBT outperforms RF and SVM. The GBDT algorithm aims to minimize the regularized objective shown in Eq. (6). $\Omega(f)$is a regularization term and each f_k is related to a decision tree (Li et al., 2020b).

$$\widetilde{\mathscr{L}} = \sum_{i} i(\widehat{y}_i, y_i) + \sum_{k} \Omega(f_k) \tag{6}$$

RUSBoosted Tree (RUSBT) is a hybrid of boosted tree and under-sampling, which saves some time for data pre-processing. RUSBT performs random under-sampling to balance the class before boosting. Compared to SMOTEBoost, RUSBT is much faster and less complex (Seiffert, Khoshgoftaar, Van Hulse, & Napolitano, 2009). During the boosting, RUSBT combines multiple weak learners to form a strong leaner, which is similar to GDBT. In Seiffert et al. s study (2009), RUSBT provided a better performance compared to SMOTEBoost and Adaboost algorithms. For the RUSBT algorithm, the following equations are utilized (Mounce et al., 2017). Eq. (7) shows how to calculate pseudo-loss. Eq. (8) explains how to calculate the weight update parameter. Eq. (9) shows how to update D_t. Eq. (10) and Eq. (11) explains how to normalize D_{t+1}. Eq. (12) represents the final hypothesis output.

$$\in_t = \sum_{(i,y),y_i \neq y} D_t(i)(1 - h_t(x_i, y_i) + h_t(x_i, y)) \tag{7}$$

$$\alpha_t = \frac{\in_t}{1 - \in_t} \tag{8}$$

$$D_{t+1}(i) = D_t(i)\alpha_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_i,y;y\neq y_i))} \tag{9}$$

$$Z_t = \sum_i D_{t+1}(i) \tag{10}$$

$$D_{t+1}(i) = \frac{D_{t+1}(i)}{Z_t} \tag{11}$$

$$H(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{t=1}^{T} h_t(x, y) \log \frac{1}{\alpha_t} \tag{12}$$

4.3. Feature selection

The dataset has a very high feature-to-sample ratio, and a substantial number of features have a low correlation with the dependent variable. In this case, feature selection is required to reduce the number of irrelevant features before the data is applied for training the model. ANOVA f-score is used to judge whether a feature is important for the dependent variable or not. A higher f-score rejects the null hypothesis, which means that the variable variance has an impact on the dependent variable variance. A percentage of the top f-score variables are selected as the training features. In this study, ANOVA was applied to measure the correlation between a feature and all features. For this purpose, the *F* statistic of the feature was used. In statistics, the F-statistic of the feature satisfies the *F*-distribution and is applied for the significance test.

The evaluation process determines the percentage, and different machine learning algorithms may have different percent of feature selection. In Table 2, the number of features and percentages are presented for each algorithm. For the RUSBT algorithm, sampling is not applied because it already includes sampling as its internal mechanism.

Table 3
Summary of the Datasets.

	Original train dataset	Under-sampled dataset	Over-sampled dataset
No. of normal samples	1162	91	1162
No. of faulty samples	91	91	1162
Total sample size	1253	182	2324

Table 4
Hyper-parameters of RF in Different Configurations.

	Number of estimators	Max tree depth
RF	50	7
RF + OS	200	7
RF + US	2	200
RF + OS + FS	200	7
RF + US + FS	3	100

4.4. Under-Sampling and Over-Sampling

The training data is highly imbalanced, and only 7% of the data points belong to the faulty class. Imbalanced datasets are always more challenging and may lead to biased and misleading prediction accuracy. Under-sampling and over-sampling are two commonly used methods to tackle imbalanced classification problems. Under-sampling samples all minority class samples and randomly selects an equal number of majority class samples. Then, it combines the two sampled subsets to form a new balanced dataset. In this training set, normal samples belong to the majority class, and the faulty samples belong to the minority class. As shown in Table 3, after the under-sampling, the total data size reduces from 1253 to 182. As shown here, the under-sampling method balances the data by losing some data points and useful information.

Opposite to the under-sampling, the over-sampling balances the data by replicating the minority class samples. As shown in Table 3, 91 faulty samples are replicated to 1162 samples, which is equal to the size of normal samples. Consequently, the total training size increases from 1253 to 2324, which is almost double. The over-sampling method balances the data and increases the data size; however, the drawback is that it is prone to overfitting due to the data replication. Synthetic Minority Over-sampling Technique (SMOTE) is one of the most commonly used over-sampling methods to reduce the overfitting problem. Instead of repeating the minority samples, SMOTE synthesizes similar data points. For instance, a minority point finds its k-nearest neighbors using Euclidean Distance and then creates one or multiple new points in between. In this case study, the SMOTE technique is applied as the underlying over-sampling approach.

4.5. Model configuration

First, we trained and applied the Random Forest algorithm with default parameters, and later, we aimed to optimize the hyper-

Table 5
Hyper-parameters of RUSBT in Different Settings.

	Number of estimators	Learning Rate
RUSBT	150	0.4
RF + FS	150	0.5

Table 6
Hyper-parameters of GDBT in Different Settings.

	Number of estimators	Max tree depth	Learning rate
GDBT	50	3	1
GDBT + OS	100	5	0.5
GDBT + US	100	1	0.7
GDBT + OS + FS	150	3	0.5
GDBT + US + FS	100	3	0.3

Table 7
Hyper-parameters of SVC in Different Settings.

	Kernel function	C	Degree	Coef0
SVC	Poly	1	2	2
SVC + OS	Poly	1	5	0.1
SVC + US	Poly	1	5	0
SVC + OS + FS	Poly	1	5	2
SVC + US + FS	Poly	1	5	0.1

parameters of the algorithm. The number of estimators and the max tree depth parameters were determined using a parameter optimization approach. In this case study, hyper-parameters of RF are optimized via the Grid Search Cross-Validation (GridSearchCV) method. Table 4 shows the parameters of the RF algorithm used with sampling and feature selection techniques.

After the GridSearchCV is applied, the hyper-parameters of RUSBT are shown in Table 5.

GDBT is computationally intensive, and thus, the optimization of all hyper-parameters is not feasible and practical. Only hyper-parameters shown in Table 6 are tuned via the GridSearchCV method.

SVC also provides several hyper-parameters to tune with the GridSearchCV method. These parameters are presented in Table 7.

The configuration of the MLP algorithm is presented in Table 8. GridSearchCV method is also applied for the MLP algorithm to optimize these parameters.

4.6. Evaluation

For binary classification problems, different performance metrics are defined based on the confusion matrix shown in Table 9. Accuracy is the most common performance metric to evaluate classification models due to its intuitiveness (Chawla, Japkowicz, & Kotcz, 2004). As shown in Eq. (13), it represents the percentage of correct predictions of all samples.

The low accuracy shows that it does not distinguish classes and does not work well for imbalanced classification, where the positive response is more important than the negative response (Hossin & Sulaiman, 2015). Precision (Eq. (14)) and recall (Eq. (15)) are the metrics with more discriminating power over different classes. Precision measures the percentage of correct prediction among all positive predictions while recall measures the percentage of correct prediction among all positive samples. However, none of these two metrics consider all elements in the confusion matrix. According to Hossin and Sulaiman (2015), F1-score (Eq.16) is the best metric to evaluate the performance of imbalanced binary classification tasks as it is a balance between precision and recall.

$$Accuracy = \frac{TP + TN}{FP + FN + TP + TN} \tag{13}$$

Table 8
MLP Configuration.

	Connection	**Number of units**	**Input dimension**	**Activation fun**
Input Layer	Dense	60	466	Relu
Hidden Layer	Dense	30	–	Relu
Output Layer	Dense	1	–	Linear
	Loss fun	**Optimizer**	**Learning Rate**	**metrics**
Compiling	Binary cross-entropy	Adam	0.001	accuracy

Table 9
Confusion Matrix.

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

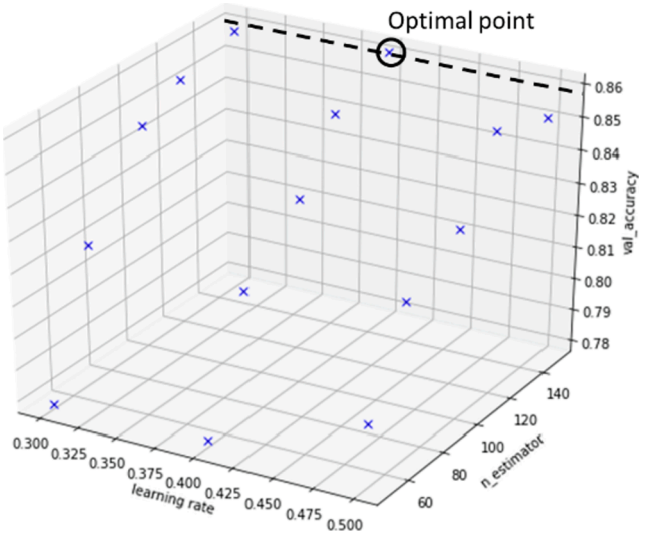


Fig. 9. GridsearchCV of RUSBT.

$$Precision = \frac{TP}{TP + FP} \tag{14}$$

$$Recall = \frac{TP}{TP + FN} \tag{15}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{16}$$

Precision and recall are usually used to evaluate the performance of imbalanced classification tasks. For this case study, precision measures the percentage of true faults in the predicted faults (Eq. (14)), and recall measures the percentage of predicted faults in the true faults (Eq. (15)). In practice, increasing recall tends to reduce the precision as the high recall needs a low threshold. Thus, F1-score is introduced (Eq. (16)) to balance between precision and recall. In this case study, F1-score is chosen to be the indicator of the classification performance. The best model should have the highest F1-score.

The hyper-parameters are tuned to achieve the best F1-Score. GridSearchCV is used to check the F1-Score for different combinations of hyper-parameters. Fig. 9 shows the process of applying GridSearchCV to find the best hyper-parameters of RUSBT. Fig. 10 shows the process of cross-validation of the MLP. The neuron weights stop when the validation accuracy is stabilized.

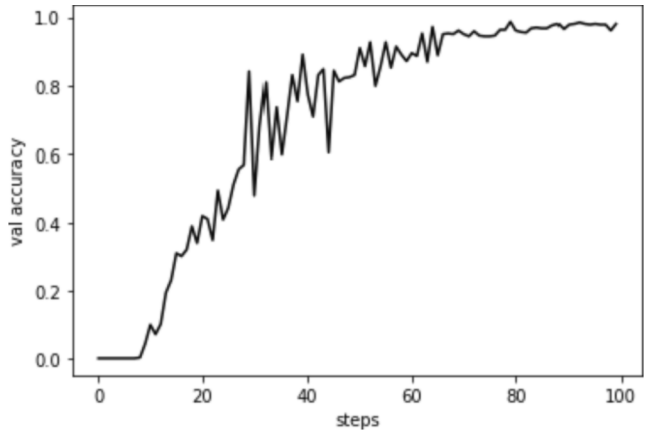


Fig. 10. Cross-validation of MLP.

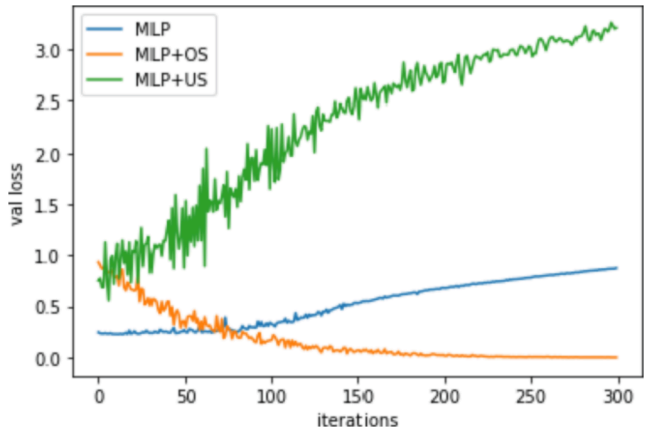


Fig. 11. MLP convergence rate of different data processing treatments.

The test dataset has 314 samples, including 4% faulty samples. The test data is also normalized before it is used for the model. The top-five F1-score models are also evaluated using Precision-Recall Curves (PR-Curve), which is another way of imbalanced classification evaluation.

5. Experimental results

As shown in Fig. 11, MLP with under-sampling converged around 200 iterations while the MLP and MLP with over-sampling methods could not reach convergence within 300 iterations. The MLP without sampling only contained a small number of failure samples reducing the rate of convergence. The MLP with under-sampling had balanced training data, but sample size is inadequate for neural network to reach a convergence. As a result, over-sampling raised the training speed and convergence rate and under-sampling method had the opposite effect.

As the baseline models, classification algorithms were applied without the use of sampling and feature selection techniques. In Table 10, the experimental results of these models are presented. According to these results, the RUSBT algorithm outperforms other algorithms in terms of F1-score because RUSBT has an embedded sampling algorithm that performs random under-sampling of the training data before the boosting is applied. The Adaboost component of the RUSBT algorithm is trained with this balanced data, but other models are trained with imbalanced data. To this end, the main advantage of the RUSBT algorithm is that it does not need an extra sampling stage to train the model. RF has a lower F1-score than the RUSBT, but it has a much higher recall value than the RUSBT. Because the training data is highly imbalanced and consists of noisy instances, RF is also effective in dealing with an imbalance classification problem. GDBT fails to identify any faulty samples and has zero F1-score, which indicates that GDBT is not suited for this problem.

Table 10
Results of the Five Models without Sampling and Feature Selection.

Algorithms	Accuracy	Precision	Recall	F1-Score
RUSBT	0.88	0.13	0.30	0.19
RF	0.63	0.07	0.62	0.12
SVC	0.80	0.05	0.23	0.09
MLP	0.92	0.08	0.08	0.08
GDBT	0.96	0.00	0.00	0.00

Table 11
Results of Models with Over-sampling and Under-sampling.

Algorithms	Sampling	Accuracy	Precision	Recall	F1-Score
RF	Over-sampling	0.92	0.20	0.31	0.24
RUSBT	–	0.88	0.13	0.30	0.19
GDBT	Over-sampling	0.94	0.22	0.15	0.18
RF	Under-sampling	0.65	0.07	0.61	0.13
GDBT	Under-sampling	0.65	0.07	0.61	0.13
SVC	Under-sampling	0.58	0.06	0.62	0.11
MLP	Under-sampling	0.63	0.05	0.53	0.10
MLP	Over-sampling	0.89	0.00	0.00	0.00
SVC	Over-sampling	0.89	0.00	0.00	0.00

As shown in Table 11, after the application of under-sampling and over-sampling, RF with over-sampling provides the best performance in terms of F1-score, which is followed by the RUSBT algorithm. As shown in Fig. 12, when the oversampling is applied, the recall of RF reduces by half (i.e., from 0.62 to 0.31), but the precision becomes nearly double (i.e., from 0.13 to 0.20). However, the under-sampling method does not make significant changes to the RF algorithm. The performance of GDBT increases dramatically after applying sampling before the training step. Fig. 13 shows that both over-sampling and under-sampling raise the F1-score of the prediction. Particularly, the recall of the under-sampling GDBT increases from 0 to 0.61. Over-sampled GDBT slightly overperforms the under-sampled model in terms of F1-score. Sampling methods do not significantly increase the performance of MLP and SVC. The performance of the SVC model even deteriorates due to oversampling. To this end, it is concluded that MLP and SVC are not suitable algorithms, and they are filtered out from the classification list. RF, RUSBT, and GDBT algorithms are then applied with both sampling and feature selection methods.

As shown in Table 12, after applying the feature selection method (i.e., ANOVA), RF achieves the highest F1-score (i.e., 0.29), which is almost twice of the second-best model GDBT (i.e., 0.15). Fig. 14 shows that all evaluation parameters of RF increase after implementing the feature selection. However, the RUSBT and GDBT algorithms predict less accurately compared to their corresponding models that do not use feature selection. The reason for this observation might be related to the internal mechanisms of the algorithms because boosting is resistant to noise, and additional feature selection may not provide extra performance. Removing some features may cause us to lose some information, and therefore, this might adversely impact the prediction performance of boosting mechanism.

As shown in Table 12, RF with over-sampling and feature selection is the best model in terms of F1-score value. However, different evaluation parameters may cause different results. As such, the PR-Curves and Receiver Operating Characteristics (ROC) Curves were investigated. For ROC Curves, the x-axis plots the false positive rate, and the y-axis shows the true positive rate. For PR-Curves, the x-axis is the recall, and the y-axis is the precision.

As shown in Fig. 15, PR-Curve and ROC Curve demonstrate that RF with under-sampling and feature selection provides the best performance in terms of Area Under Curve (AUC) parameters. Table 13 shows

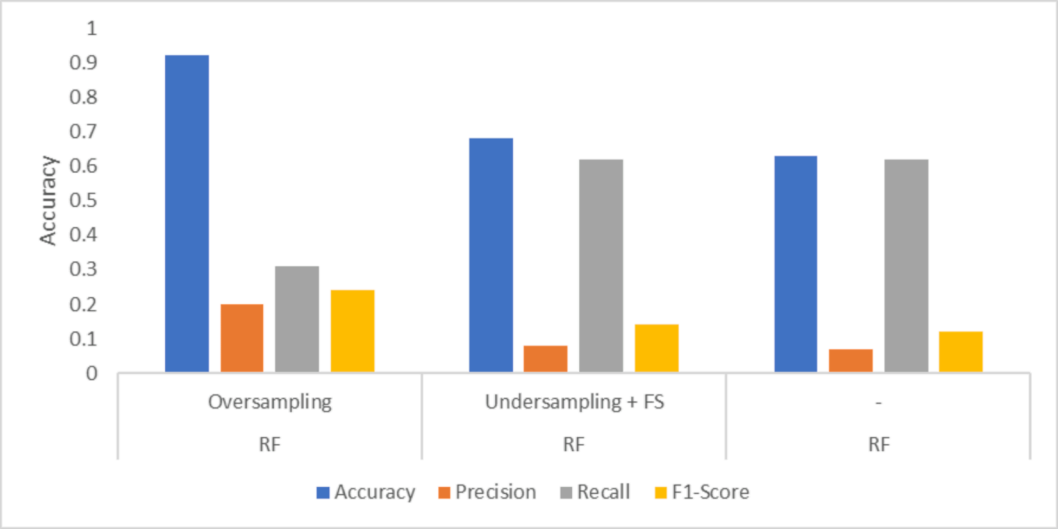


Fig. 12. RF Results Before and After Sampling.

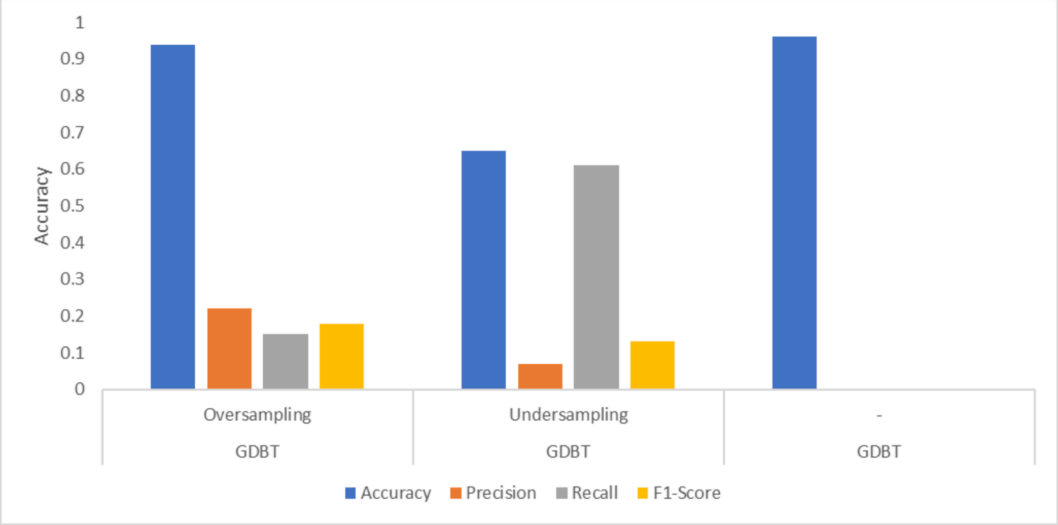


Fig. 13. GDBT Results Before and After Sampling.

Table 12
Results of RF, GDBT, and RUSBT with Sampling and Feature Selection.

Algorithms	Data Processing	Accuracy	Precision	Recall	F1-Score
RF	Over-sampling + Feature Selection	0.92	0.24	0.38	0.29
GDBT	Over-sampling + Feature Selection	0.93	0.15	0.15	0.15
RF	Under-sampling + Feature Selection	0.68	0.08	0.62	0.14
GDBT	Under-sampling + Feature Selection	0.69	0.07	0.54	0.13
RUSBT	Feature Selection	0.87	0.09	0.23	0.13

that this model achieves the best performance (i.e., ROC AUC = 0.72 and PR AUC = 0.14). RF with over-sampling and feature selection is the second-best algorithm in terms of AUC values. This table shows that RF overperforms GDBT in terms of PR and ROC curves.

According to experimental results, the RUSBT algorithm has the best performance when no sampling and feature selection methods are applied. RF outperforms other algorithms if the sampling method is applied to balance the data points in the dataset. It was observed that the feature selection significantly improves the performance of RF. RF with under-sampling and feature selection provided the best performance to predict the dependent variable in this study. 0.72 AUC score is achieved with RF under-sampling and feature selection model. 0.69 AUC score is achieved with the RF over-sampling and feature selection model.

6. Discussion

This study illustrated the effectiveness of sampling methods and feature selection methods over different machine learning algorithms. It showed that both under-sampling and over-sampling methods were effective in dealing with the imbalanced classification problem. Particularly, the under-sampling and the over-sampling methods can significantly improves the performance of RF and GDBT, which are decision-tree based algorithms. However, under-sampling methods might decrease the accuracy of prediction due to sample size reduction. Feature selection can effectively remove noises from the less correlated features and improves the accuracy of all algorithms. The combination of imbalance data resampling and feature selection are useful methods to improve the accuracy of product failure detection algorithms.

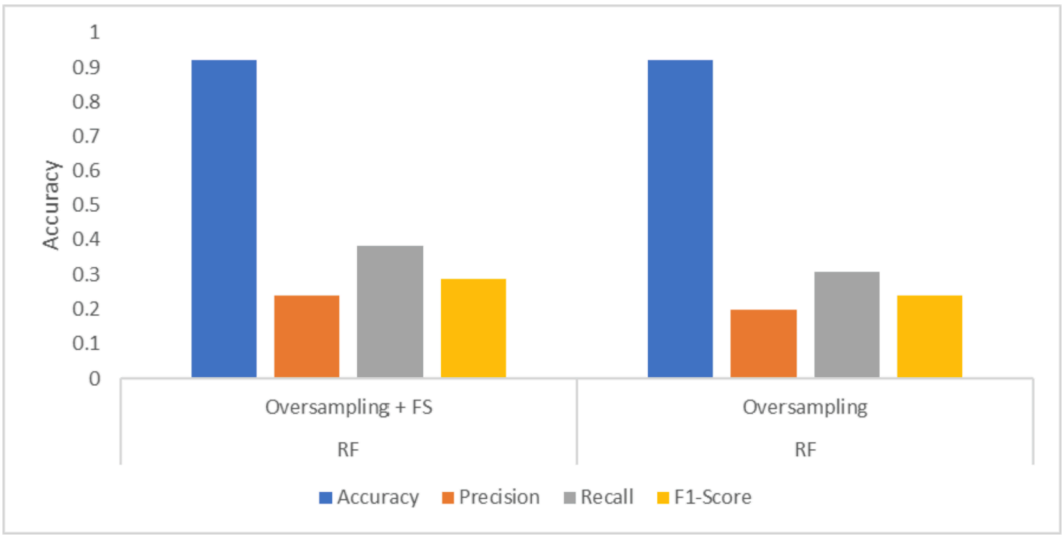


Fig. 14. Results of RF Over-sampling Before and After Feature Selection.

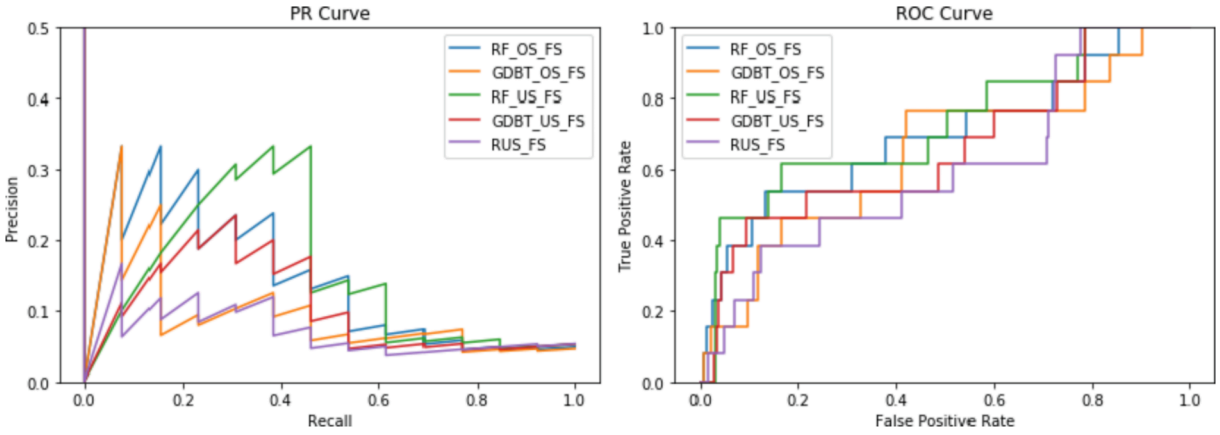


Fig. 15. PR-Curve and ROC-Curve of the Top-five F1-score Models.

Table 13
PR-Curve AUC and ROC-Curve AUC values with Sampling and Feature Selection.

Algorithms	Data Processing	PR AUC	ROC AUC
RF	Under-sampling + FS	0.14	0.72
RF	Over-sampling + FS	0.13	0.69
GDBT	Over-sampling + FS	0.09	0.64
GDBT	Under-sampling + FS	0.07	0.66
RUSBT	FS	0.07	0.6

In this study, it was demonstrated that machine learning could effectively detect faulty products in production lines. Particularly, the RF-based model provided the best performance, however, this model requires the use of a data sampling algorithm. As a result of bootstrapping, the training data may not contain or only contain a small portion of the minority data, causing that RF is prone to imbalanced dataset. If the practitioners do not want to investigate the effect of data sampling approaches, it is also possible to design the prediction model using the RUSBT algorithm because this algorithm has an embedded sampling algorithm. When feature selection algorithms are applied, the best performance was achieved with the RF algorithm. When all the experiments are considered, the RF with under-sampling and feature selection provided the best performance among other models. According to these observations, we suggest practitioners apply our RF-based model to improve their production activities. Although the performance of the proposed model is sufficient, there are still some possibilities to improve the overall performance. Because the dataset size is limited and the sample-to-feature ratio is low, the prediction performance can be further improved by increasing the dataset size. Also, additional data pre-processing methods (i.e., normalization techniques and data imputation approaches), classification algorithms, feature selection techniques, data sampling approaches, and parameter optimization techniques can be applied to improve the performance of the proposed model. Due to the limited size of the dataset, traditional classification algorithms were focused on; however, deep learning algorithms might improve the performance on a larger dataset though they have several limitations as explained in the Background and Related Work section.

In this study, a public production line dataset is used to evaluate the effectiveness of the machine learning approaches. One of the major drawbacks of this public dataset is that the feature names are unknown, and as such, this makes it more difficult to select the most relevant features. However, we believe that feature names should be known to the data scientists and engineers, and in such a case, the irrelevant or noisy features can be filtered out before applying the feature selection. This additional information on the features can increase the performance significantly and reduce the workload of data pre-processing. The proposed model using machine learning has been applied in a specific production line context, and the performance of this model might differ in a different production line context. The difference between

production lines is vast, and the sensory data quality varies among different production lines. Some production lines may not include many features, and some of them can have parallel paths, which means that one product does not pass through all the sensors. All these differences make it hard to design a universal machine learning-based model to detect faulty products in a production line.

7. Conclusion

In this study, the effectiveness of machine learning-based prediction models was investigated for the identification of faulty products in production lines. For this purpose, normalization, data imputation, feature selection, data balancing, classifiers, and parameter optimization techniques have been used to build a novel prediction model. It was observed that under-sampling and over-sampling are both effective methods, which improve the prediction performance of RF-based models. Feature selection reduces the noise and increases the performance of classification algorithms. Random Forest with sampling and feature selection achieved the best performance score in terms of AUC value for faulty product detection. The prediction results achieved in this study demonstrate the potential of data-driven methods in the domain of prognostic diagnosis in production lines. The performance of the proposed machine learning-based model can be further improved if more data points per feature are available, and more information about features is provided in the datasets. The future plan is to apply deep learning algorithms such as convolutional neural networks and recurrent neural networks when a large dataset is reached.

Declarations

I. Ethical approval: Not applicable.

II. Funding Details: No funding was received.

III. Conflicts of interest/Competing interests: There is no conflict of interest.

IV. Informed Consent: The Author transfers to Elsevier the non-exclusive publication rights.

V. Availability of data and material: Public dataset was used (McCann & Johnston, 2008).

VI. Code availability: Code is allowed upon request (https://github.com/frankmp40/Product-Failure-Detection-for-Production-Lines-using-A-Data-Driven-Model).

CRediT authorship contribution statement

Ziqiu Kang: Conceptualization, Methodology, Software, Validation, Writing – review & editing. **Cagatay Catal:** Methodology, Validation, Writing – review & editing. **Bedir Tekinerdogan:** Methodology, Validation, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Angeli, C. (2010). Diagnostic expert systems: From expert's knowledge to real-time systems. *Advanced Knowledge Based Systems: Model, Applications & Zesearch, 1*, 50–73.

Banadkooki, F. B., Ehteram, M., Ahmed, A. N., Fai, C. M., Afan, H. A., Ridwam, W. M., … El-Shafie, A. (2019). Precipitation forecasting using multilayer neural network and support vector machine optimization based on flow regime algorithm taking into account uncertainties of soft computing models. *Sustainability, 11*(23), 6681.

Borlea, I. D., Precup, R. E., Borlea, A. B., & Iercan, D. (2021). A unified form of fuzzy C-means and K-means algorithms and its partitional implementation. *Knowledge-Based Systems, 214*, Article 106731.

Catal, C., & Tekinerdogan, B. (2019). Aligning education for the life sciences domain to support digitalization and industry 4.0. *Procedia computer science, 158*, 99–106.

Chawla, N., Japkowicz, N., & Kotcz, A. Editorial: special issue on learning from imbalanced data sets, SIGKDD Explor. Newslett. 6 (1)(2004) 1–6. In.

Chen, C., Liaw, A., & Breiman, L. (2004). Using random forest to learn imbalanced data. *University of California, Berkeley, 110*(1–12), 24.

Choudhary, A. K., Harding, J. A., & Tiwari, M. K. (2009). Data mining in manufacturing: A review based on the kind of knowledge. *Journal of Intelligent Manufacturing, 20*(5), 501.

Chun, Y. H. (2016). Improved method of estimating the product quality after multiple inspections. *International Journal of Production Research, 54*(19), 5686–5696.

Cutler, A., Cutler, D., & Stevens, J. (2011). Random Forests. *In (Vol., 45*, 157–176.

Frank, P. M. (1990). Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy: A survey and some new results. *Automatica, 26*(3), 459–474. https://doi.org/10.1016/0005-1098(90)90018-D

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*(9), 1263–1284.

Hossin, M., & Sulaiman, M. N. (2015). A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process, 5*, 01–11. https://doi.org/10.5121/ijdkp.2015.5201

Iqbal, R., Maniak, T., Doctor, F., & Karyotis, C. (2019). Fault detection and isolation in industrial processes using deep learning approaches. *IEEE Transactions on Industrial Informatics, 15*(5), 3077–3084.

Isermann, R. (2005). Model-based fault-detection and diagnosis–status and applications. *Annual Reviews in control, 29*(1), 71–85.

Isermann, R. (2006). *Fault-diagnosis systems: An introduction from fault detection to fault tolerance*. Springer Science & Business Media.

Jodas, D. S., Pereira, A. S., & Tavares, J. M. R. (2020). Classification of calcified regions in atherosclerotic lesions of the carotid artery in computed tomography angiography images. *Neural Computing and Applications, 32*(7), 2553–2573.

Kang, S., Kim, E., Shim, J., Chang, W., & Cho, S. (2018). Product failure prediction with missing data. *International Journal of Production Research, 56*(14), 4849–4859.

Kang, S., Kim, E., Shim, J., Cho, S., Chang, W., & Kim, J. (2017). Mining the relationship between production and customer service data for failure analysis of industrial products. *Computers & Industrial Engineering, 106*, 137–146.

Kang, Z., Catal, C., & Tekinerdogan, B. (2020). Machine learning applications in production lines: A systematic literature review. *Computers & Industrial Engineering, 149*, Article 106773.

Kang, Z., Catal, C., & Tekinerdogan, B. (2021). Remaining useful life (RUL) prediction of equipment in production lines using artificial neural networks. *Sensors, 21*(3), 932.

Khumprom, P., & Yodo, N. (2019). A data-driven predictive prognostic model for lithium-ion batteries based on a deep learning algorithm. *Energies, 12*(4), 660.

Köksal, G., Batmaz, İ., & Testik, M. C. (2011). A review of data mining applications for quality improvement in manufacturing industry. *Expert systems with Applications, 38*(10), 13448–13467.

Kusiak, A. (2006). Data mining: Manufacturing and service applications. *International Journal of Production Research, 44*(18–19), 4175–4191.

Kwak, D. S., & Kim, K. J. (2012). A data mining approach considering missing values for the optimization of semiconductor-manufacturing processes. *Expert Systems with Applications, 39*(3), 2590–2596.

Li, Z., Wang, Y., & Wang, K. (2019). A deep learning driven method for fault classification and degradation assessment in mechanical equipment. *Computers in Industry, 104*, 1–10.

Li, Z., Wang, Y., & Wang, K. (2020a). A data-driven method based on deep belief networks for backlash error prediction in machining centers. *Journal of Intelligent Manufacturing, 31*(7), 1693–1705.

Li, Q., Wu, Z., Wen, Z., & He, B. (2020b). Privacy-preserving gradient boosting decision trees. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 01, pp. 784-791).

Liang, P., Yang, H. D., Chen, W. S., Xiao, S. Y., & Lan, Z. Z. (2018). Transfer learning for aluminium extrusion electricity consumption anomaly detection via deep neural networks. *International Journal of Computer Integrated Manufacturing, 31*(4–5), 396–405.

Liong, S. T., Zheng, D., Huang, Y. C., & Gan, Y. S. (2020). Leather defect classification and segmentation using deep learning architecture. *International Journal of Computer Integrated Manufacturing, 33*(10–11), 1105–1117.

Liu, Y., Wang, Y., & Zhang, J. (2012, September). New machine learning algorithm: Random forest. In International Conference on Information Computing and Applications (pp. 246-252). Springer, Berlin, Heidelberg.

Lughofer, E., Pollak, R., Zavoianu, A. C., Meyer-Heye, P., Zörrer, H., Eitzinger, C., ... & Radauer, T. (2017, June). Self-adaptive time-series based forecast models for predicting quality criteria in microfluidics chip production. In: 2017 3rd IEEE International Conference on Cybernetics (CYBCONF) (pp. 1-8). IEEE.

McCann, M., & Johnston, A. (2008). Secom Dataset, UCI Machine Learning Repository. In.

Miljković, D. (2011). Fault detection methods: A literature survey. In 2011 Proceedings of the 34th international convention MIPRO (pp. 750-755). IEEE.Natekin, A., & Knoll, A. (2013). Gradient Boosting Machines, A Tutorial. *Frontiers in neurorobotics, 7*, 21. doi:10.3389/fnbot.2013.00021.

Mounce, S. R., Ellis, K., Edwards, J. M., Speight, V. L., Jakomis, N., & Boxall, J. B. (2017). Ensemble decision tree models using RUSBoost for estimating risk of iron failure in drinking water distribution systems. *Water Resources Management, 31*(5), 1575–1589.

Natekin, A., & Knoll, A. (2013). Gradient Boosting Machines, A Tutorial. *Frontiers in Neurorobotics, 7*, 21. https://doi.org/10.3389/fnbot.2013.00021

Pozna, C., & Precup, R. E. (2014). Applications of signatures to expert systems modelling. *Acta Polytechnica Hungarica, 11*(2), 21–39.

Russ, G., Kruse, R., Karim, M. A., Hsu, A. L., Halgamuge, S., Smith, A. J., & Islam, A. (2005). In *November). Detection of faulty semiconductor wafers using dynamic growing self organizing map* (pp. 1–6). IEEE.

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2009). RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 40*(1), 185–197.

Teramoto, R. (2009). Balanced Gradient Boosting from Imbalanced Data for Clinical Outcome Prediction. *Statistical Applications in Genetics and Molecular Biology, 8*, Article20. https://doi.org/10.2202/1544-6115.1422

Veni, C. V. (2018). *On the Classification of Imbalanced Data Sets*.

Venkatasubramanian, V., Rengaswamy, R., Yin, K., & Kavuri, S. N. (2003). A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. *Computers & Chemical Engineering, 27*(3), 293–311.

Verdier, G., & Ferreira, A. (2011). Adaptive Mahalanobis Distance and k-Nearest Neighbor Rule for Fault Detection in Semiconductor Manufacturing. *IEEE Transactions on Semiconductor Manufacturing, 24*(1), 59–68. https://doi.org/10.1109/TSM.2010.2065531

Wang, L., Zhang, Z., Long, H., Xu, J., & Liu, R. (2016). Wind turbine gearbox failure identification with deep neural networks. *IEEE Transactions on Industrial Informatics, 13*(3), 1360–1368.

Yang, C. C. (2008). Improving the definition and quantification of quality costs. *Total Quality Management, 19*(3), 175–191.