Rule-based Diagnostics of a Production Line

Osarenren Kennedy Aimiyekagbon[1], Lars Muth[2], Meike Wohlleben[3], Amelie Bender[4], Walter Sextro[5]

[1,2,3,4,5] *Paderborn University, Faculty of Mechanical Engineering, Dynamics and Mechatronics, Warburger Str. 100, 33098 Paderborn, Germany*
{*firstname.lastname*}*@uni-paderborn.de*

ABSTRACT

In the industry 4.0 era, there is a growing need to transform unstructured data acquired by a multitude of sources into information and subsequently into knowledge to improve the quality of manufactured products, to boost production, for predictive maintenance, etc. Data-driven approaches, such as machine learning techniques, are typically employed to model the underlying relationship from data. However, an increase in model accuracy with state-of-the-art methods, such as deep convolutional neural networks, results in less interpretability and transparency. Due to the ease of implementation, interpretation and transparency to both domain experts and non-experts, a rule-based method is proposed in this paper, for prognostics and health management (PHM) and specifically for diagnostics. The proposed method utilizes the most relevant sensor signals acquired via feature extraction and selection techniques and expert knowledge. As a case study, the presented method is evaluated on data from a real-world quality control set-up provided by the European prognostics and health management society (PHME) at the conference's 2021 data challenge. With the proposed method, our team took the third place, capable of successfully diagnosing different fault modes, irrespective of varying conditions.

1. INTRODUCTION

Fostered by digitalization and distributed networking, the fourth industrial revolution (industry 4.0) encompasses smart factories, cyber-physical systems, the Internet of Things (IoT), amongst others. In this era, development phases and subsequently time to market have become shorter, production flexibility to satisfy individual customer needs has become highly necessary and high system availability and efficiency have become more significant (Lasi, Fettke, Kemper, Feld, & Hoffmann, 2014; Lee et al., 2018). These have emphasized the need for advanced analytics and prognostics and health management (PHM) not just at the system level but also at the component level. In addition to diagnostics and prognostics of cyber-physical systems and IoT devices, PHM also involves health management through their life cycle (Lee et al., 2018). There exist a broad range of diagnostics and prognostics approaches, from physics-of-failure (PoF)-based methods to artificial intelligence techniques. On the one hand, PoF models build on domain experts and first principles, which foster model transparency and interpretability. Furthermore, high-fidelity models result in high model accuracy (Elattar, Elminir, & Riad, 2016). However, a major drawback of PoF models is deriving such high-fidelity models. On the other hand, with data available from various sources, artificial intelligence methods, such as machine learning techniques, are often employed to model the underlying relationship from data (Elattar et al., 2016). The recent development of deep learning techniques such as convolutional neural networks allows the modeling of complex relationships within data (Zhang et al., 2019). However, this results in high model accuracy at the cost of diminished model transparency and interpretability. Hybrid methods are conceivable to bridge the gap between these two techniques (Elattar et al., 2016).

Given that production, product quality, and maintenance are intertwined (Gu, He, Han, & Chen, 2017), a brief literature review on the application of PHM techniques in the industry and industry-related use cases is considered in the following. In their study, Jimenez-Cortadi et al. (Jimenez-Cortadi, Irigoien, Boto, Sierra, & Rodriguez, 2020) employed four machine learning techniques, including recurrent neural networks, to estimate the remaining useful life (RUL) of machining tools of a computer numerical control (CNC) lathe. Although their application was successful in the offline development phase, an online implementation within the production machine was impeded due to the complexity of the techniques.

Panicucci et al. (Panicucci et al., 2020) proposed a cloud- and edge-based solution. Their solution comprises three tree-based methods, including a decision tree and a random forest. According to the authors, their method choice is attributed to model interpretability, which is directly gained in the decision tree and indirectly via feature importance in the random forest. The proposed methods are successfully employed to diagnose

simulated component degradation of an experimental robotic arm. However, only the motor current signal was investigated in the course of their study.

For the RUL estimation of a CNC machine tool, Luo et al. (Luo, Hu, Ye, Zhang, & Wei, 2020) employed a digital twin-driven hybrid modeling approach. The approach consists of a particle filter method, a multi-domain simulation model and several data-driven methods, such as support vector regression. A high accuracy could be achieved with their proposed approach, even under varying conditions. However, the simulation model is not validated. Moreover, deriving high-fidelity models is typically not feasible.

To assist in the transition from corrective to predictive maintenance, Fernandes et al. (Fernandes et al., 2019) employed a rule-based modeling approach to anticipated predictive maintenance tasks of a precision parts manufacturing company. Because of the absence of fault instances, the described approach could not be validated. However, due to the transparency and interpretability of such a knowledge-driven rule-based approach, it is considered in this paper.

Per the general PHM steps (Elattar et al., 2016), the process involved in generating the data of the case study at hand is briefly described subsequently, alongside an overview of the data set and the tackled task. An overview of the employed techniques, ranging from feature extraction to modeling approaches, is provided. The methodology towards fault classification and operating condition clustering is put forward in the following section. Afterwards, the achieved results are presented, and finally, a concluding remark and some possible improvements are proposed.

1.1. Case Study

A case study is considered for the diagnostics of an industrial production line. The data set was provided by the European prognostics and health management society (PHME) at the conference's 2021 data challenge (PHM Society, 2021). A description of the monitored process, the data set, and the challenge is given in the following paragraphs.

Process description

The process, as depicted in Figure 1, is an automated quality control pipeline for electrical fuses. This quality control pipeline is a vivid example of part of a smart factory. It consists of components, such as a 4-axis SCARA-robot picking and placing the fuses with a vacuum gripper to and from a testing area, a thermal camera to detect fuses with unusually high temperatures and an industrial camera to monitor the fuses. As visualized in Figure 1, the relevant steps of the quality control pipeline are as follows: The fuses are picked up by the vacuum gripper of the robot from the part feeding system (feeder) and placed onto the testing area (1). In the testing area, the fuses undergo tests, including electrical conductivity, amperage and temperature. After the tests, the fuses are placed (2) on a conveyor belt, where they are sorted according to the test results by an actuated bar (3) and then transported (4) to a narrow conveyor belt (5). On this conveyor belt, the fuses are transported back to the feeder (6). The described cycle is repeated for up to about three hours. During this process, several health and process monitoring signals were acquired.

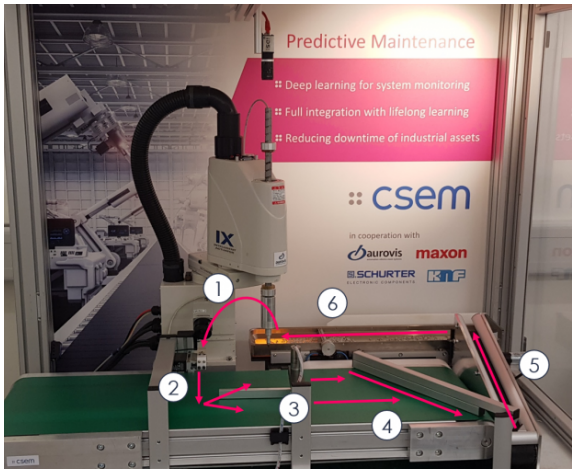

Figure 1. Structure and steps of the quality control pipeline for electrical fuses (PHM Society, 2021)

Data description

Data has been acquired under different health states and operating conditions by an automated data acquisition system during the described process. The health states comprise fault-free instances and artificial fault modes. These fault modes were introduced to simulate real-world scenarios for the challenge, and each mode influences one or more sensor signals. Furthermore, data was acquired under two different operating conditions for the fault-free state.

A total of 99 data sets (experiments) were provided for the model training and validation phase, comprising fault-free instances with class label 0 and faulty instances with class labels 2, 3, 4, 5, 7, 9, 11, and 12 as indicated in Table 1.

Table 1. An overview of the provided data sets

Class description	Label(s)	Quantity
Fault-free instance	0	70
Artificial fault instance	2, 3, 5, 7, 9	4 each
	4, 11, 12	3 each

Each data set comprises a set of 50 signals, which can be broadly categorized into health monitoring signals, such as `Vacuum` and `SmartMotorSpeed`, environmental monitoring signals, such as `Humidity` and `Temperature`, and general process monitoring signals, such as `ProcessCpuLoadNormalized` and `ProcessMemoryConsumption`.

Per sensor signal, up to seven attributes, as depicted in Figure 2, are calculated over a time window of 10 s and stored. Only these attributes are made available and not the raw sensor signals.

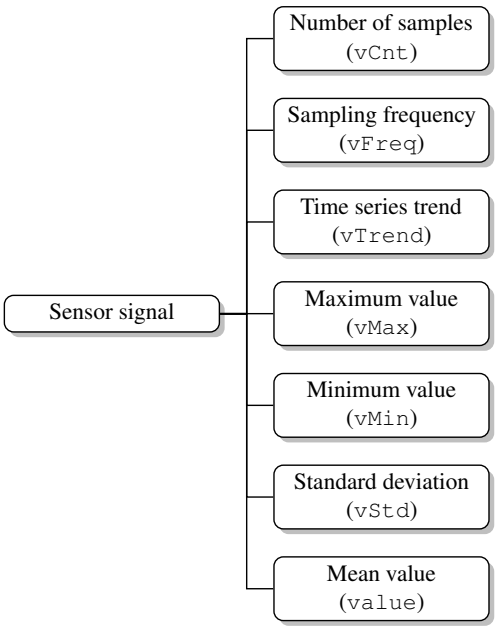

Figure 2. Possible attributes stored per sensor

Task description
For the data challenge, four tasks were formulated. Firstly, previously unseen data has to be correctly classified (fault detection and identification). Secondly, the signals have to be ranked according to their usefulness in diagnosing a fault, if present (root cause analysis). Thirdly, the data segments used to identify a fault, if present, should be as short as possible, according to the temporal order. Lastly, the fault-free instances acquired under two different operating conditions have to be clustered. A requirement of the challenge was the submission of the solution codes as Python 3 jupyter notebook files.

1.2. Theoretical framework

The tasks are located in the two fields classification/diagnosis and clustering. The first three tasks are strongly connected to the basic process of setting up a diagnosis system: First, measurements have to be acquired. From these measurements, features have to be extracted. A large number of possibly insignificant features increases the model complexity and might also diminish model accuracy. Hence, the most significant features have to be selected, and finally, a model for the classification of faults has to be set up.

The fourth task stems from the field of clustering. In the case at hand, it is known, that two groups (operating conditions) are present in the data, but it is unknown, which experiment belongs to which group. In this case, the feature extraction can be carried out as for a classification task. However, the feature selection cannot be based on variances within and between classes, since no labels are available.

Following the general PHM steps (Elattar et al., 2016), the employed feature extraction, feature selection and modeling approach to tackle the presented tasks are given in this section.

Feature extraction
Typically, features are extracted in the time-, frequency- and time-frequency domain of condition monitoring data, such as vibration and acoustic data, acquired over time (Luo et al., 2020). Given that the presented signals are not complex waveform signals, but already pre-processed, only a time domain analysis of the presented signals is adequate. Thus, statistical parameters, such as standard deviations and peak values, are derived from the signals.

Feature selection
Feature selection techniques can be broadly classified into filter, wrapper, and embedded methods (Guyon & Elisseeff, 2003). Filter methods are employed to select a subset of features independent of the selected classifier. Wrapper methods use classifiers or regressors to rank features and are usually more accurate than filter methods. Nonetheless, they are computationally more expensive. With embedded methods, the feature ranking and the classifier/regressor training process occur simultaneously (Guyon & Elisseeff, 2003).

Working with unlabelled data, i.e., unsupervised learning, presents a new challenge. In supervised learning, labels help to discern the importance of the given set of features and thus, they help in feature selection. However, this is not the case in unsupervised learning since no labels are available. Hence, the questions arise, which features are significant, which are redundant, which are irrelevant, and which are even misleading (Dy & Brodley, 2004). However, different approaches for selecting features in unsupervised learning can be found in scholarly literature (Roffo, 2016; Roffo, Melzi, & Cristani, 2015).

In this paper, filter methods are employed, independent of a specific classifier or clustering algorithm, to preselect the most significant features before modeling or algorithm training.

- **Filter methods for classification**
  Chi-square test and Analysis of variance (ANOVA) F-value were evaluated for the proposed tasks. However, given the data challenge requirements and the similarity of the results, only the ANOVA F-value was adopted in this paper.
  ANOVA F-value: Not every available signal and feature is relevant and significant. Therefore, when creating efficient models, only certain features have to be selected by the feature selection process. The first step in this process is to evaluate all available features with respect to the

classification task. Typically, a feature score is calculated based on how separable the feature values are between the classes. Such a score is the ANOVA F-value. It is defined as

$$F = \frac{MS_B}{MS_W}, \tag{1}$$

where MS_B is an estimator for the variance of the feature means between all classes, and MS_W is an estimator for the variance of the feature within a class (Sahai & Ojeda, 2004). F becomes larger, if a feature has very distinct values for a certain class as compared to all available samples, and it becomes a value close to 1, if the feature cannot explain the differences between the classes. Based on the F-value, the highest scoring features can be selected and used to build a classification model.

- **Filter methods for clustering**
  Infinite feature selection (Roffo et al., 2015), feature ranking based on correlation coefficients (Guyon, Weston, Barnhill, & Vapnik, 2002) and Laplacian score (He, Cai, & Niyogi, 2005) as implemented by Giorgio (Roffo, 2016; Giorgio, 2021), are evaluated for the clustering of the operating conditions. Due to the equally good results, the Laplacian score is employed for selecting a subset of features for the clustering task.
  Laplacian score (He et al., 2005): The underlying principle is to construct a graph with k nearest neighbours, assign weights to connected nodes of the graph, and then compute the Laplacian score from the derived Laplacian graph. The weights are defined by distance metrics, such as the Euclidean distance. Top-ranked features according to the Laplacian scores are selected for the clustering tasks.

Modeling approach
Several methods were employed to tackle the presented data challenge tasks, of which the rule-based approach is principal. A concise description of this method and two employed tree-based methods are presented in the following.

- **Rule-based diagnostics**
  Rule-based systems or expert systems utilize a set of `if-then` rules derived from expert knowledge for diagnostics, i.e., for each known fault mode, an `if-then` conditional statement is formulated while incorporating some threshold (Hayes-Roth, 1985; Rossi & Braun, 1997; Katipamula & Brambley, 2005). Rule-based systems have found application in various fields, such as in the engineering field and medical field. Their numerous application is accounted to their ease of implementation, transparency and interpretability (Hayes-Roth, 1985; Rossi & Braun, 1997; Katipamula & Brambley, 2005). A limitation is that the rule base builds on existing and known fault modes, which has to be updated by the occurrence of a new fault mode (Hayes-Roth, 1985; Rossi & Braun, 1997; Katipamula & Brambley, 2005).
  In an offline/static setting, as in the data challenge, all possible fault modes and operating conditions are available. In such cases, rule-based approaches can be adopted. If no decision rules can be defined through expert knowledge, decision trees or random forests are a good alternative.
- **Decision trees**
  Decision trees are a simple and explainable approach for classification. Algorithms for building decision trees recursively split the given training data into subsets using comparisons of one variable against a fixed value at a time, while minimising impurities in the subsets (Loh, 2011). These comparisons are the nodes of the tree. The tree is expanded and more nodes are created until the subsets are pure or a stopping criterion is met.
- **Random forests**
  Random forests are an extension of decision trees, promising better generalization capabilities for unseen data (Ho, 1995). A random forest consists of multiple decision trees, of which every tree is built using only a random subset of the available features and data. The final classification is then the class, which the most individual trees returned. This way, a random forest can prevent overfitting on the training data.

2. METHODOLOGY

This section presents the methods used to generate the fault classification model and to cluster the operating conditions of the fault-free state. Most of the methods are available in the Python package scikit-learn (Pedregosa et al., 2011).

2.1. Fault classification

According to the task, the classification should be done as quickly as possible, i.e., using as few data points at the beginning of an experiment as possible. Therefore, different features are generated only from short snippets of the given data. The extracted features are the rolling average, maximum, minimum, peak-to-peak-distance, and standard deviation. Since the given data points represent signal attributes generated from $10\,\mathrm{s}$ measurements, the new features result from two operations, e.g., the minimum of standard deviations. In the following, the signal attributes and the correspondingly extracted features are appended to the signal name with a dot separator as in `SmartMotorSpeed.value.min`, which implies the minimum of the mean values of the signal `SmartMotorSpeed`. Variable window sizes s are applied to analyze the features' dependency on the number of data points.

To evaluate the generated features, the task is reduced to a binary classification for every class, i.e., it is checked whether a feature contains enough information to decide if an experiment belongs to a given class or not.

This is done based on the ANOVA F-values as previously explained. To gain an insight into the significance of the

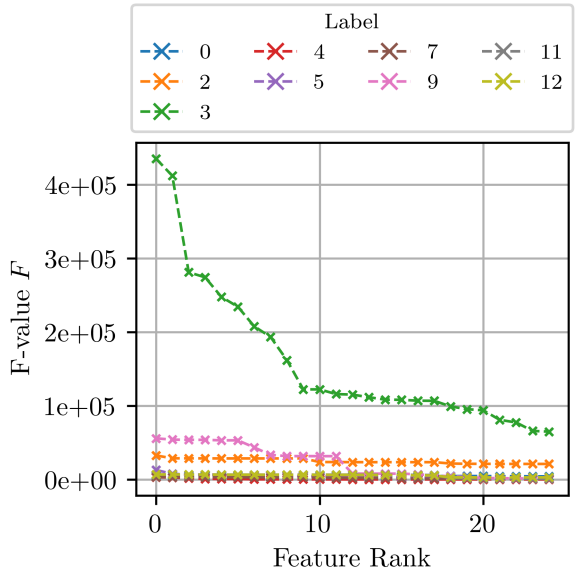

Figure 3. F-value F of the 25 best features for every class with window size $s = 3$

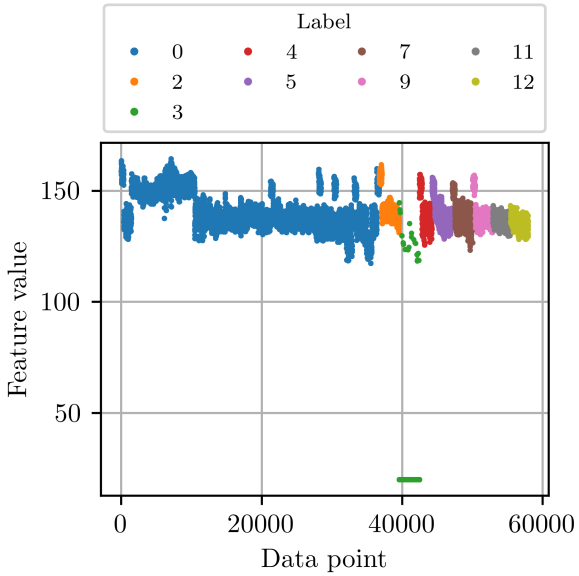

Figure 4. `FeederBackgroundIlluminationIntensity.vMin.min` with window size $s = 3$

extracted features, Figure 3 shows the F-values of the 25 highest scoring features for every class. The features have significantly higher scores for classes 2, 3, and 9 than for the remaining classes, even when using small window sizes $s \leq 3$. A visual inspection of the features reveals that these three classes are easily separable from the remaining classes using only a few of the highest scoring features. As exemplarily shown in Figure 4, class 3 is easily identifiable using only `FeederBackgroundIlluminationIntensity.vMin.min`, that is, an experiment unambiguously belongs to class 3, when this feature value drops below 100. With features based on `VacuumValveClosed.value` and a window size of six data points, class 5 is also clearly separable from the others.

Although the F-values of the top-ranked features for classes 11 and 12 are lower compared to classes 2, 3, and 9 in Figure 3, further investigation of the features also leads to a simple approach for separation of the classes: The lower F-values result from the fact, that class 0 contains some experiments, which lead to similar feature values as those, which distinguish classes 11 and 12 from all other classes. The signals affected are `DurationRobotFromFeederToTestBench`, `DurationRobotFromTestBenchToFeeder`, `SmartMotorSpeed`, and `SmartMotorPositionError`. The first two represent the robot's operating speed, and the last two correspond to the conveyor belt's operating speed. This set of signals has to be considered concurrently for classes 11 and 12. The rolling average of `DurationRobotFromFeederToTestBench.value` and the rolling minimum of `SmartMotorSpeed.vMin` are exemplarily depicted in Figures 5 and 6, respectively. As can be inferred, the experiments in class 0 have either high values for both `DurationRobotFromFeederToTestBench` and `SmartMotorSpeed`, which implies a slowed down condition or low values for both signals as in classes 2 to 9. The two levels of the feature values represent two fault-free operating conditions with different operating speeds. As can also be deduced from Figures 5 and 6, for class 11, the signal `SmartMotorSpeed` is elevated and not `DurationRobotFromFeederToTestBench` and vice versa for class 12. This finding also holds for the other signals corresponding to the robot's operating speed and the speed of the conveyor belt. Thus, a clear distinction between the classes 0, 11, and 12 can be made with the previously mentioned signals.

The remaining classes 4 and 7 are only clearly identifiable when further increasing the window size. As shown in Figure 7, the F-value of the best feature for class 7 increases significantly with the window size s. Therefore, class 7 is only identifiable after the experiment has been running for a while. The same holds for class 4. In both cases, the most accurate solution would be to use all available data points per experiment. However, this contradicts the goal of short prediction times.

Since most of the faulty classes (2, 3, 5, 9, 11, and 12) have clear boundaries separating them from the other classes, these classes are classified using simple checks against a fixed threshold for a maximum of four features per class (see appendix, Table 2 for the chosen features and rules). Additionally, the slowed-down condition of class 0 is identifiable this way. Only

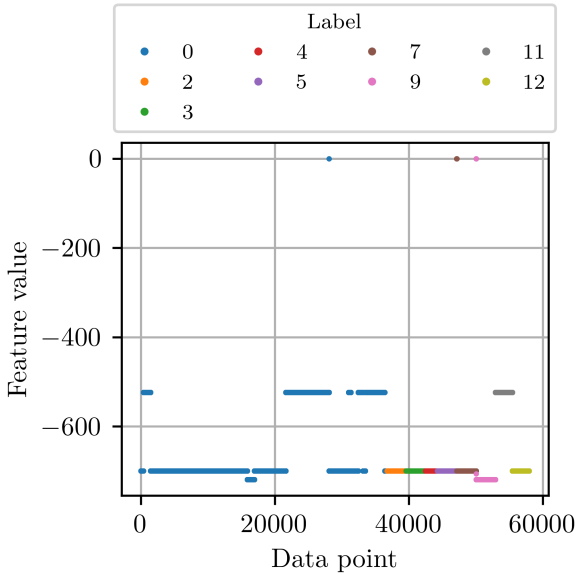

Figure 5. `SmartMotorSpeed.vMin.min` with window size $s = 6$

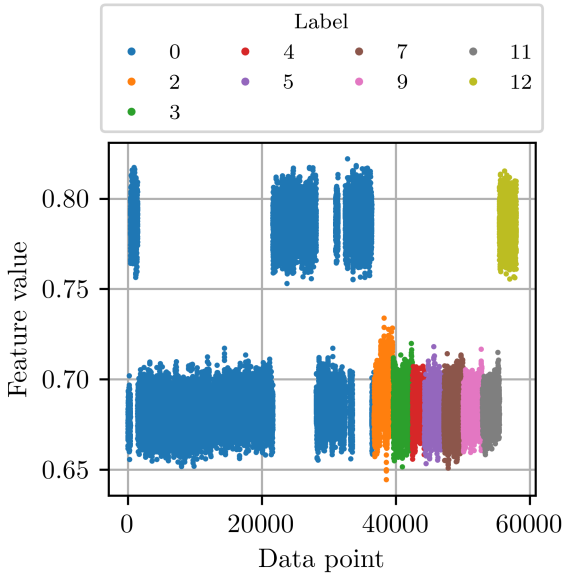

Figure 6. `DurationRobotFromFeederToTestBench.value.mean` with window size $s = 6$

classes 4, 7, and the default condition of class 0 cannot be easily separated. Therefore, for these classes, random forests are set up using 25 features. With 25, a medium number of features has been chosen for the random forests to prevent overfitting on the training data but still be able to discover complex dependencies of the classification on multiple features.

In the test case, the features used for classification are generated and continuously updated using the available data, i.e. the data points from the beginning of the experiment until the current time. Since the random forest identifying class 0 yields a high false positive rate, only the models identifying the faulty classes are applied at first. If no fault is identified, the next data point is added. As soon as a fault is identified, the loop is stopped and the the detected fault is returned. This way, in a real world use case with a fault-free system, the loop would be continued over the whole run time of the system (as long as no fault is returned, the system is considered to be fault-free). However, with respect to the presented data challenge task, the experiment is classified as fault-free, if no fault is detected after a maximum number of data points $t_{\max}$, to reduce the time needed to identify fault-free experiments. Additionally, the created random forest identifying class 0 is applied after a certain number of data points $t_{0,\min}$ with $t_{0,\min} < t_{\max}$. This lower limit on the number of points $t_{0,\min}$ is chosen to be 13 data points, since the longest `FuseCycleDuration` contained in all experiments is 129 s. During this time, all possible actions of the quality control line should have occurred at least once. However, the random forest identifying fault-free experiments yields a high false positive rate for experiments of the classes 4 and 7. Therefore, the faster identification of fault-free experiments comes at the cost of reduced accuracy when classes 4 or 7 are present.

The most important signals corresponding to each class are identified using the highest scoring features. The signals also allow for a physical interpretation of the faults (see appendix, Table 3).

2.2. Clustering of operating conditions

To cluster the fault-free experiments according to their operating condition, multiple feature extraction techniques are applied to the data of the fault-free experiments. From the generated features, all zero and not a number (NaN) features are removed. The remaining features are ranked using Laplacian scores. Finally, a kMeans algorithm is applied using the highest scoring features. For instance, Figure 8 shows the mean values of `Vacuum.vMax` of all data points per fault-free experiment. From this feature alone, already two clusters can be identified, which represent two different operating conditions affecting the vacuum system. The clusters either have average values of `Vacuum.vMax` above or below -0.2.

However, different clusters can be identified, when using `SmartMotorSpeed`, and `DurationFromFeederToTestBench` or `DurationFromTestBenchToFeeder`, as shown above for the fault classification. These signals represent the overall operating speed of the system. Since it is given that only two operating conditions were used and from the data available it cannot be decided for certain, if either the vacuum system or the operating speed were altered, the

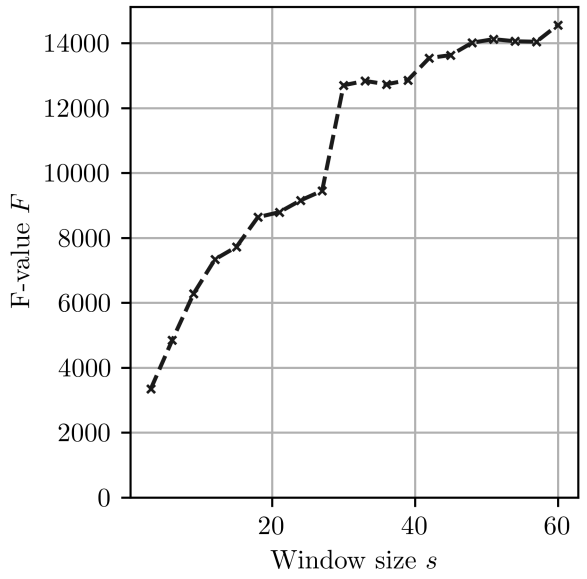

Figure 7. F-value F of the best feature for class 7 over window size s

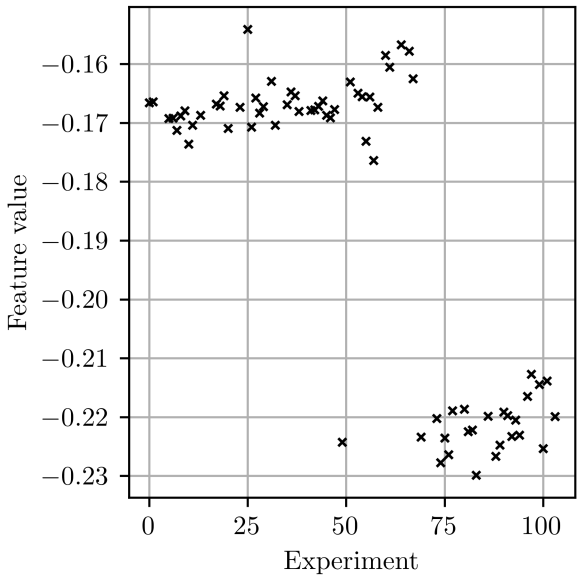

Figure 8. `Vacuum.vMax.mean` per fault-free experiment

decision is left to the kMeans algorithm.

3. Results

Figure 9 shows the resulting confusion matrix when applying the classification model to all available experiments. The values in every row are normalized by the total number of experiments per row (target). The model yields high accuracy when classes $0, 3, 5, 9, 11$, and 12 are present. Only three experiments of class 0 are not classified into class 0, since they look very similar to class 5, 7, and 9, respectively, and are classified accordingly. Due to the distinctive characteristics of classes 5 and 9 these two experiments are probably correctly classified by the model and were mistakenly labelled as class 0 in the given data.

One of four available experiments of class 2 is misclassified as class 0. However, an increase of the number of data points until the random forest for class 0 is evaluated to $t_{0,min} = 30$ (corresponding to $300\,\mathrm{s}$ run time), leads to the correct classification of this experiment. This can be explained by the fault which is associated with class 2: The number of fuses in the cycle is probably reduced and it takes a while after the start of an experiment until the feeder is empty.

As already expected during the feature selection, the model also yields low accuracy for experiments from classes 4 and 7. Only one of three experiments from class 4 is identified correctly. The other two are classified as fault-free (class 0). Three of four experiments from class 7 are identified correctly, and one is classified as fault-free. Again, the accuracy could be increased, when increasing the number of data points until the random forest for class 0 is evaluated as well as the maximum number of data points until a prediction is made. However, this comes at the cost of a much longer average prediction time for class 0. As discussed before, in a real use case this would not be a drawback, since the system can be regarded as fault-free as long as no fault is identified. In the data challenge though, also the fault-free state shall be identified in the shortest time

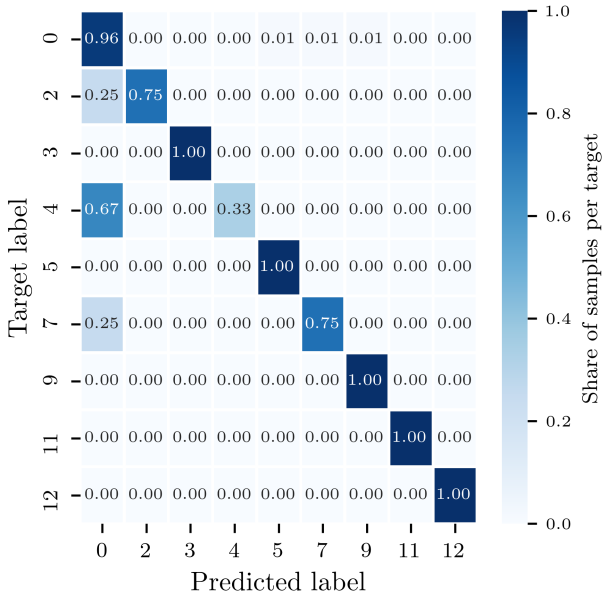

Figure 9. Confusion matrix for time until evaluation of random forest for class 0 $t_{0,\min} = 13$ and maximum time to prediction $t_{\max} = 30$

possible. Therefore, a reduced accuracy is accepted in favor of faster prediction times. The average time to prediction for every class can be found in the appendix, Table 2.

4. Conclusion

Although the number of use cases where machine learning is applied is continually rising, rule-based diagnostics are often a good alternative for machine health monitoring. In the presented use case, most faults are identifiable through their specific influence on a small number of features. The resulting rule-based fault classification model is easy to understand, explain and extend. The model is static, i.e. it does not automatically adapt itself over time when new fault-free operating conditions are introduced, for instance. However, this is not unusual, even for state-of-the-art machine learning techniques. For a possible adaptation, incremental learning methods, such as described by Yang et al. (Yang, Gu, & Wu, 2019), can be adopted. To maintain transparency and interpretability, we would propose an incremental random forest with a few trees.

References

Dy, J. G., & Brodley, C. E. (2004). Feature selection for unsupervised learning. *Journal of machine learning research*, *5*(Aug), 845–889.

Elattar, H. M., Elminir, H. K., & Riad, A. (2016). Prognostics: a literature review. *Complex & Intelligent Systems*, *2*(2), 125–154.

Fernandes, M., Canito, A., Bolón-Canedo, V., Conceição, L., Praça, I., & Marreiros, G. (2019). Data analysis and feature selection for predictive maintenance: A case-study in the metallurgic industry. *International journal of information management*, *46*, 252–262.

Giorgio. (2021). *Feature selection library.* Retrieved from `https://www.mathworks.com/matlabcentral/fileexchange/56937-feature-selection-library` (MATLAB Central File Exchange. Retrieved May)

Gu, C., He, Y., Han, X., & Chen, Z. (2017). Product quality oriented predictive maintenance strategy for manufacturing systems. In *2017 prognostics and system health management conference (phm-harbin)* (p. 1-7). doi: 10.1109/PHM.2017.8079213

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, *3*(Mar), 1157–1182.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, *46*(1), 389–422.

Hayes-Roth, F. (1985). Rule-based systems. *Communications of the ACM*, *28*(9), 921–932.

He, X., Cai, D., & Niyogi, P. (2005). Laplacian score for feature selection. In *Proceedings of the 18th international conference on neural information processing systems* (p. 507–514). Cambridge, MA, USA: MIT Press.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition* (Vol. 1, p. 278-282). doi: 10.1109/ICDAR.1995.598994

Jimenez-Cortadi, A., Irigoien, I., Boto, F., Sierra, B., & Rodriguez, G. (2020). Predictive maintenance on the machining process and machine tool. *Applied Sciences*, *10*(1), 224.

Katipamula, S., & Brambley, M. R. (2005). Methods for fault detection, diagnostics, and prognostics for building systems—a review, part i. *Hvac&R Research*, *11*(1), 3–25.

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., & Hoffmann, M. (2014). Industry 4.0. *Business & information systems engineering*, *6*(4), 239–242.

Lee, G.-Y., Kim, M., Quan, Y.-J., Kim, M.-S., Kim, T. J. Y., Yoon, H.-S., ... others (2018). Machine health management in smart factory: A review. *Journal of Mechanical Science and Technology*, *32*(3), 987–1009.

Loh, W.-Y. (2011). Classification and regression trees. *WIREs Data Mining and Knowledge Discovery*, *1*(1), 14-23. doi: 10.1002/widm.8

Luo, W., Hu, T., Ye, Y., Zhang, C., & Wei, Y. (2020). A hybrid predictive maintenance approach for cnc machine tool driven by digital twin. *Robotics and Computer-Integrated Manufacturing*, *65*, 101974.

Panicucci, S., Nikolakis, N., Cerquitelli, T., Ventura, F., Proto, S., Macii, E., ... others (2020). A cloud-to-edge approach to support predictive analytics in robotics industry. *Electronics*, *9*(3), 492.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

PHM Society. (2021). *Phme data challenge.* 2021 European prognostics and health management society (PHME) Data Challenge. Retrieved from `https://phm-europe.org/data-challenge` (Retrieved on: 01.06.2021)

Roffo, G. (2016). Feature selection library (matlab toolbox). *arXiv preprint arXiv:1607.01327*.

Roffo, G., Melzi, S., & Cristani, M. (2015). Infinite feature selection. In *Proceedings of the ieee international conference on computer vision* (pp. 4202–4210).

Rossi, T. M., & Braun, J. E. (1997). A statistical, rule-based fault detection and diagnostic method for vapor compression air conditioners. *Hvac&R Research*, *3*(1), 19–37.

Sahai, H., & Ojeda, M. M. (2004). *Analysis of variance for random models*. Boston, MA, USA: Birkhäuser.

Yang, Q., Gu, Y., & Wu, D. (2019). Survey of incremental learning. In *2019 chinese control and decision conference (ccdc)* (p. 399-404). doi: 10.1109/CCDC.2019.8832774

Zhang, L., Lin, J., Liu, B., Zhang, Z., Yan, X., & Wei, M. (2019). A review on deep learning applications in prognostics and health management. *IEEE Access*, *7*, 162415-162438. doi: 10.1109/ACCESS.2019.2950985

APPENDIX

Table 2. Rules defined and average time to prediction (number of data points used) for every class; class "0 slow" denotes the slowed down operating condition of the fault-free state

Class	Rules	Average time to prediction (number of data points used)
0	Random forest	14.6
0 slow	$0.75 <$ DurationRobotFromFeederToTestBench.value.mean $0.78 <$ DurationRobotFromTestBenchToFeeder.value.mean $-550 <$ SmartMotorSpeed.vMin.min < -500 $-400 <$ SmartMotorPositionError.vMin.min < -300	2.95
2	$5.5 <$ NumberFuseEstimated.vCnt.max	6.33
3	FeederBackgroundIlluminationIntensity.vMin.min < 100	3.75
4	Random forest	6.0
5	$-0.4 <$ VacuumValveClosed.value.max $0.3 <$ VacuumValveClosed.value.p2p $0.1 <$ VacuumValveClosed.value.std	4.0
7	Random forest	6.0
9	SmartMotorSpeed.vMin.min < -710	6.2
11	DurationRobotFromFeederToTestBench.value.mean < 0.72 DurationRobotFromTestBenchToFeeder.value.mean < 0.74 $-550 <$ SmartMotorSpeed.vMin.min < -500 $-400 <$ SmartMotorPositionError.vMin.min < -300	2.67
12	$0.75 <$ DurationRobotFromFeederToTestBench.value.mean $0.78 <$ DurationRobotFromTestBenchToFeeder.value.mean $-710 <$ SmartMotorSpeed.vMin.min < -690 $-600 <$ SmartMotorPositionError.vMin.min < -400	4.33

Table 3. Most important signals and physical interpretation for every fault

Class	Most important signals	Physical interpretation
0	None or DurationRobotFromFeederToTestBench DurationRobotFromTestBenchToFeeder SmartMotorSpeed SmartMotorPositionError	Fault-free, normal operating speed or Fault-free, robot and conveyor belt slowed down
2	NumberFuseEstimated NumberFuseDetected NumberEmptyFeeder	Less fuses estimated and estimation is done more frequently, significantly less fuses may be circulating
3	FeederBackgroundIlluminationIntensity SharpnessImage IntensityTotalImage	Feeder background illumination defective
4	TotalCpuLoadNormalized	Total CPU load is fluctuating while process CPU load seems normal, additional processes may be running on the computer
5	VacuumValveClosed Vacuum	Vacuum frequently reaches values close to 0, vacuum system defective
7	FusePicked	Less fuses picked over time, but everything else seems normal, can only be detected after a long time, gripper may be defective
9	SmartMotorSpeed SmartMotorPositionError	Conveyor belt's motor speed fluctuating, motor defective
11	SmartMotorSpeed SmartMotorPositionError	Only conveyor belt slowed down
12	DurationRobotFromFeederToTestBench DurationRobotFromTestBenchToFeeder	Only robot slowed down