Hierarchical Multi-class Classification for Fault Diagnosis

Pablo del Moral
E-mail: pablo.del_moral@hh.se

Sławomir Nowaczyk
E-mail: slawomir.nowaczyk@hh.se

Sepideh Pashami
E-mail: sepideh.pashami@hh.se
CAISR, Center for Applied Intelligent Systems Research, Halmstad, Sweden

This paper formulates the problem of predictive maintenance for complex systems as a hierarchical multi-class classification task. This formulation is useful for equipment with multiple sub-systems and components performing heterogeneous tasks. Often, the data available describes the whole system's operation and is not ideal for accurate condition monitoring. In this setup, specialized predictive models analyzing one component at a time rarely perform much better than random. However, using machine learning and hierarchical approaches, we can still exploit the data to build a fault isolation system that provides measurable benefits for technicians in the field.
We propose a method for creating a taxonomy of components to train hierarchical classifiers that aim to identify the faulty component. The output of this model is a structured set of predictions with different probabilities for each component.
In this setup, traditional machine learning metrics fail to capture the relationship between the performance of the models and its usefulness in the field. We introduce a new metric to evaluate our approach's benefits; it measures the number of tests a technician needs to perform before pinpointing the faulty component.
Using a dataset from a real-case problem coming fro the automotive industry, we demonstrate how traditional machine learning performance metrics, like accuracy, fail to capture practical benefits. Our proposed hierarchical approach succeeds in exploiting the information in the data and outperforms non-hierarchical machine learning solutions. In addition, we can identify the weakest link of our fault isolation model, allowing us to improve it efficiently.

Keywords: Fault Diagnosis, Multi-class Classification, Hierarchical Classification, Automotive Industry, Integral Fault Diagnosis, Structure Prediction

1. Introduction

Monitoring complex equipment for maintenance purposes is of great interest to the industry. Knowing the system's state allows us to devise maintenance strategies that optimize uptime, reduce maintenance costs, and increase reliability.

In the literature, there are two main groups of industrial applications for automated self-monitoring tasks. The first is simple machines, and the second is complex safety-critical systems. The perfect representative of the first group is the monitoring of rolling bearings using vibration measurements; in this case, there is an expected physical relation between the component's health and the data collected. An example of the second group is the monitoring of aircraft; the criticality of these systems justifies the cost of installing an extensive network of sensors and processing the resulting amount of data. The common characteristic of these two scenarios is that the resulting models are objectively very accurate – either because the problem is simple or because the information we have is quite exhaustive.

Motivated by the interactions with our industrial partners from the automotive and health care industry, we want to focus on a rarely-tackled third type of scenario, which is quite different from the previous two. A vehicle is a quite complex machine comprising multiple heterogeneous subsystems responsible for various tasks. However, the economic benefits of an extensive network of sensors do not justify the cost of installation, processing, storing, and modeling of the data. On the other hand, dividing it into simple components and providing hundreds of per-component condition monitoring solutions is not any cheaper. It also neglects to consider the behavior of the vehicle as a whole and the interactions between

Proceedings of the 31st European Safety and Reliability Conference
Edited by Bruno Castanier, Marko Cepin, David Bigaud, and Christophe Berenguer
Published by Research Publishing, Singapore.
ISBN: 978-981-18-2016-8; doi:10.3850/978-981-18-2016-8_524-cd

subsystems.

In the past decades, more and more sensors have been installed in vehicles and similar types of machines. The data available is not collected with the design of a monitoring system in mind, but mainly for other purposes such as compliance with regulations, control, safety or security. With this data, an accurate description of the machine's state in general, and each of its components specifically, is not possible. This leads to an important qualitative difference between this third scenario and the previous two – objectively, the models that can be created are quite inaccurate. At this stage, the requirement from a company is to find value in the existing data and identify the next steps to create a successful monitoring solution.

In this paper, we focus on the problem of fault diagnosis in the framework of automated monitoring. Given a breakdown of a machine, we want to discover which specific component is responsible for the fault. To build our models, we use the sensor data describing the operation of a fleet of trucks and the historical logs of their repairs.

The output of our machine learning models must be exploited by the technician carrying out the maintenance operation. Surprisingly enough, the link between the usefulness of the solution and the evaluation of the machine learning models has not been studied extensively in the literature.

We propose a simple framework for an expert to interact with our models. The technician sequentially tests the components identified by our models as the most likely to be faulty, one by one. This framework offers us a way to evaluate how useful the models will really be, based on how many tests must be performed before the fault is isolated. We will demonstrate how traditional performance metrics used within the machine learning community do not necessarily correspond to this practical utility.

In this scenario, we assume that classification models will have a low performance at the component level. However, we should be able to find multiple groups of components that can be easily distinguishable by a classifier. In our approach, we first create a taxonomy by automatically extracting a hierarchy of components (see Fig. 1) from the data. We later use this taxonomy to build a hierarchical classification model: training binary classifiers at every node of the hierarchy to discriminate between this node's children.

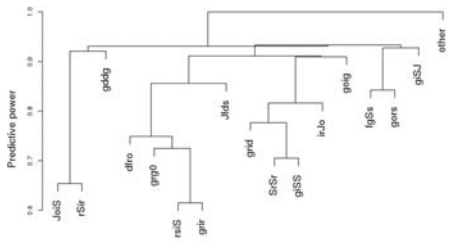

Fig. 1. An example of an extracted hierarchy with 17 components. On the y-axis, we can see the expected quality of the split. The name of the components has been anonymized.

We compare our hierarchical classifiers' results against a classifier that does not take into account the hierarchical structure. We show how our method of extracting hierarchies helps us to create models that outperform non-hierarchical ones in terms of the usefulness for the technician making the diagnosis – even though they perform very similar in terms of classification accuracy.

Finally, we use the structure of the hierarchy to identify which nodes of our hierarchy corresponds to the weakest models. We use this information to identify where to install an extra sensor tailored to enhance the performance of this model. We simulate the addition of this sensor to enhance the performance of the models in different nodes of the hierarchy and evaluate its benefits.

2. Literature review

Self-monitoring of industrial equipment has become a hot topic, especially in the context of Industry 4.0 Yan et al. (2017), Li et al. (2017).

There are examples of data-based models applied to critical systems. In Verhagen and Boer] (2018), the authors propose a data-driven fault prediction architecture for different components of an aircraft. In their case, they use 1597 features to build their models. In Michau et al. (2017), the authors present a solution to detect and isolate faults in a power plant generator using readings from 320 sensors. In Canizo et al. (2017), the authors present an architecture to predict failures in wind turbines using 552 features, including sensor readings and alarm signals. In all these cases, there is an extensive network of sensors describing the functioning of the system.

On the other side, we can find a body of research focused on monitoring simple systems with few sensors. These approaches are based on physical modeling and signal analysis. In Lei et al. (2020), we can find an extensive survey on this type of systems, especifically on bearings, gears, motors and engines.

Regarding fault diagnosis, we can see two main lines of research : signal importance methods and supervised methods.

In the first group, we can find Cheng et al. (2016). In this work, the authors analyze the correlation between different signals before and after a failure. Using causality, they trace back the original signals responsible for the fault. In Michau et al. (2017), the authors first trained a classifica-

tion model to predict whether a fault is going to occur or not. Then, when a fault is predicted, they trace back the signals most responsible for the prediction. These works are based on the assumption that the signal recorded by a sensor can identify a particular component.

Necessarily, the problem of supervised classification for fault diagnosis is a multi-class classification problem. An example of a supervised method for failure diagnosis is Wen et al. (2018). In this work, authors use convolutional neural networks on three different types of machinery with 10, 4, and 2 types of failures, respectively. They evaluate their results based on global and per-component accuracy. In Baraldi et al. (2016), the authors use multi-class classification to identify different types of failures for the same component. In their case, they also evaluate their approach based on global and per-component accuracy.

3. Motivation

In this paper, we formulate the problem of fault diagnosis as a multi-class classification problem. In other words, once we have a faulty machine, we want to identify which of its many components is responsible for the fault. We want to introduce hierarchical multi-class classification for fault diagnosis. It uses a hierarchy of classes (in our case, components of the machines) in the shape of a tree. Starting with all components grouped at the root of the tree, we will train a classifier at each node to separate the left sub-branch components from those part of the right sub-branch.

This classification task aims to assist the technician in performing the diagnosis on the machine, reducing the amount of work needed to find a fault. The first thing we need to do is to define a framework in which our models and the technician can interact. We propose the following setup, which we believe is quite realistic, although, of course, more sophisticated alternatives are also possible. The output of a classification model is a numeric vector of probabilities, indicating how likely it is for each component to be the one responsible for the fault. The technician receives this input, selects the component with the highest probability, and performs a test to validate the predictions. In the simplest case, the test could be to replace the component and see whether the machine is now fixed, but exact details of that are not critical for our further analysis. If the component is not faulty, the technician will move to the next most likely one until he finally finds the fault.

The quality of any model in this framework can be evaluated based on its utility. We measure this utility by counting the number of tests that the technician needs to perform to identify the faulty component. In the experiments presented here, our technician is "simulated": based on the ground truth, we always know how many tests are necessary to find the real fault for every possible classifier output. We will make three assumptions to make the experimental evaluation simpler: all the tests consider a single component, the cost of every test is the same, the tests are perfect.

These assumptions do not necessarily hold. The framework proposed here can be easily extended to more complex cases, but doing that here would distract from this work's central message. In practice, the technician can often perform multi-component tests, i.e., evaluate groups of components rather than individual ones. For example, if the engine of a truck does not start, one could easily discard the braking system as a possible fault. In this work, we do not consider such a possibility since it is usually not obvious which groups are realistic. Such information, if available, would be beneficial and could be used to adapt our hierarchical approach that is built. Unfortunately, we have no access to it in the use-case we study in this paper. The other possible extension is for each of the tests to have different costs, which can be monetary, timely, etc. Again, without domain knowledge, it is not possible to account for that – even though incorporating such information into our evaluation would provide additional benefits.

We expect that the symptoms of failures from components that perform similar tasks or are part of the same subsystem will be very similar. In these circumstances, the performance of a classifier that separates each of the components of the truck will be low. However, we also expect different groups of components to have very distinct effects on the data. For example, an electrical fault and a fault in the braking system should have very different symptoms and thus be represented in the data in a quite distinct way. Our idea of a good hierarchy is the one that captures these similarities and dissimilarities between the components. By training classifiers to separate easily distinguishable sets of the components on the hierarchy's higher levels, we can avoid unnecessary classification errors. On the other hand, near the bottom of the hierarchy, we can build specialized classifiers to distinguish between those components that have a similar effect on the data, which can lead to better performance.

Using hierarchies for fault diagnosis is consistent with how manual fault diagnosis is typically made. The manual fault diagnosis method can be regarded as a hierarchical structure. The technician follows a deductive process, performing tests and isolating the faulty component. A hierarchical structure of components can accommodate this type of process and accommodate multi-component testing naturally.

4. Method

4.1. *Data*

For this work, we use the data provided by one of our industrial partners, Volvo Trucks. We perform our analysis on a fleet of approximately 4000 trucks operating in Europe for three years. We will use two sources of data: operational data and repair logs.

The operational data consists of on-board sensor readings, presented as interpolated timestamped instances with two weeks frequency. In total, our dataset has 365 different numerical features collected from these sensors. In addition, we have 74 categorical features describing the particular configuration of each truck.

The repair logs contain information about the workshop visits of each truck. They specify which components have been replaced and when. It is important to notice that this information includes both replacements due to a breakdown and a scheduled maintenance operation. Some of the replaced components might not have been faulty at the time of replacement, which adds an extra layer of uncertainty. The name of the components have anonymized.

4.2. *Processing of the data*

We characterize each visit to the workshop with two months of operational data previous to that visit. On average, this means about four data instances. Since we intend to emulate the real deployment of a fault isolation system, we use the initial two-thirds of the visits as training, and the rest for testing and evaluation of the models. It is important to guarantee that we never use future data to classify any of the failures.

There are 284 different components in this dataset and 365 different sensors describing the general state of the truck. We assume that for most components, their failures will be hard to identify since the data can not provide a relevant description for each of them.

We aim to discover the relationships between the symptoms of different components' failures. Then, use this information to extract a hierarchy that encodes these relationships.

First, we need to make sure that a particular repair is due to a component failure and not an scheduled maintenance operation. We assume that by selecting those components that are rarely replaced simultaneously on the same truck, we can filter out most of the regular maintenance operations.

Then, using only the training set, for each component, we train a simple classifier to distinguish between the visits to the workshop when the component was replaced and the rest of the visits. Using stratified (based on the truck identity) cross-validation, we can get an estimate of how much predictive power there is in the data and we can rank the components from easier to harder to predict a failure.

We want our diagnostic system to be integral, i.e., identify the failures of all possible components. We have already stated that for many components this will be possible. However, we want our model to include the information of what can and can not predicted. Therefore, once we have selected a group of components that can be predicted, we label each visit in the training set with the component that was replaced during that visit. If none of the selected components is replaced, we use an extra class, called "other."

4.3. *Extracting the hierarchy*

To extract the hierarchy of selected components, we follow a greedy top-down algorithm. Starting with all the classes, we want to find a way of splitting them in such a way that a classifier trained to distinguish the "left" group from the "right" group has the highest possible accuracy. This process is repeated recursively, building a tree until individual components are placed at the leaves (see Fig. 1).

The number of possible splits is prohibitively large even for a modest number of components. There are 511 possible splits for a node with 10 components, more than a million for anode with 21 components. Given that trying every alternative is not feasible, we use an off-the-shelf genetic algorithm from the r package *GA* (with default parameters) to find a near-optimum solution. First, we train a single classifier on the full dataset using 5-fold stratified (based on the truck identity) cross-validation and keep the probability predictions.

As an evaluation function for the genetic algorithm, we are going to use the area under the ROC curve function estimated from the 5-fold cross-validation. Instead of training a new classifier, we will use the probability predictions from the single classifier. As an example, in Fig. 1, we want to evaluate the split between the com-

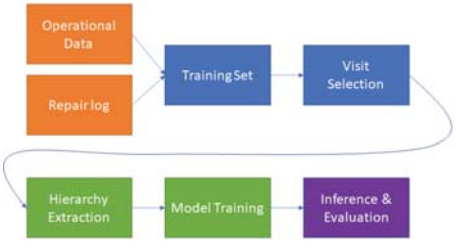

Fig. 2. Workflow of our method. Different steps corresponding with each section are in a different color.

ponents {"fgSs", "gors"}, and "giSJ". We create a meta-class "left" that combines the probability predictions of "fgSs" and "gors", and a meta-class "right" that does the same for the "giSJ". Equipped with the probability predictions of the "left" and "right" classes and the ground truth, we calculate the area under the ROC curve. This is the final output of the evaluation function. We let the genetic algorithm run until it can not find a better solution for more than 10 iterations. The final output of the genetic algorithm will indicate us how to split into two groups the comonents of a node.

To accommodate the "other" class, we artificially place it at the top of the hierarchy. In this way, the hierarchical model first decides whether a new instance is part of our group of selected components. If yes, it then identifies which of them is responsible for the fault.

Once we have extracted the hierarchy, we train new classifiers at each node, with the goal to distinguish all the components of the left sub-branch from the components on the right sub-branch.

4.4. *Inference*

To obtain a prediction from the hierarchical model, all the individual models at the nodes are evaluated, and their probabilistic output are propagated from the root of the hierarchy to the leaves. As an example, the final probability prediction of class "gddg" in Fig. 1 would be the probability of the classifier at the top node to choose left, times the probability of the classifier at the next node to choose left, times the probability of the classifier at the next node to choose right.

In our data, for both test and training sets, each of the visits to the workshop is characterized by the last four instances in the dataset. To output a final prediction, we run each of the four instances through our hierarchical classifier. The prediction on each instance is a vector of probabilities, assigning a number between 0 and 1 to each of the components. We expect that a fault has a more significant effect on the data as time evolves; therefore, the final prediction for a visit is a weighted average of its four instances, assigning a larger weight to the instances closer to the workshop visit.

5. Experiments

We expect to observe a measurable improvement after using our extracted hierarchy for building our models, compared to models that do not take into account this structure. How beneficial they can be depends on two factors: the number of components and the complexity of the task.

Our extracted hierarchy will be helpful if it can capture the interesting relationships of similarity and dissimilarity between the different components. We assume that there exist different levels of similarity. For example, the faults of two components of the braking system will have very similar effects on the data, and they will be more similar to the faults on components of the steering system than to the faults on components of the air conditioning system.

If we had a classifier that obtains perfect classification score, there would not be any added value from using a hierarchical structure – as the models become better, there is less and less room for improvement. An equivalent statement can be made about very poor models whose outputs are random – since there is no useful information about any of the components, grouping them together in a smart way has nothing to exploit. In both cases, the explanation is the same: all components are equally similar, i.e., there is no structure (see Fig. 3). We will only see a benefit in using hierarchies when non-hierarchical classifiers fail to account for the structure in the similarity of components.

With very few components, there is not much structure to exploit. With more components, we can expect more complex and heterogeneous relationships of similarity. Deciphering these increasingly complex relationships and encoding them in a hierarchy will boost the performance of hierarchical classifiers compared to non-hierarchical ones. However, as the number of components continues to grow, the complexity of the task increases to the point where individual base classifiers cannot cope with it.

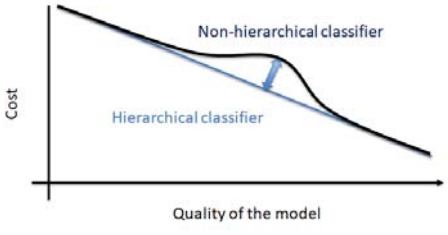

Fig. 3. For classifiers that achieve high or low performance classification, the benefits of a good hierarchy will not be noticeable.

5.1. *Experimental setup*

For our experiments, we select subsets of components based on the two criteria presented in the method section. First, we filter out the components that are usually replaced simultaneously with others. Then, we pick a subset of components with the highest predictive power. We present below our results for seven differently-sized subsets of components, based on seven different thresholds of how easy they are to diagnose individually. In

the rest of this section, we summarize our major findings.

As a base classifier, we use a decision tree classifier, as implemented in the r package *rpart*. Through 5-fold cross-validation, we will tune the regularization parameter *cp* and the number of instances per leaf *minleaf*.

5.2. *Accuracy*

First, we measure the accuracies of a single and a hierarchical classifier. In table 1, we can see the comparison. There does not exist a clear difference between the accuracies of the non-hierarchical and the hierarchical classifiers.

As we increase the number of components, there is a clear drop in the accuracy. This is an expected result. Every visit to the workshop that does not include any of the selected components is labeled with the "other" class. This means that the lower the number of selected components, the more dominant the "other" class is. Besides, we have selected the components based on their predictive power, so the subsets with more components are on average harder to predict.

Table 1. Accuracy of a single classifier and a hierarchical approach with different sets of selected components.

	Accuracy	
# Comp	Single	Hierarchical
12	0.91	0.91
17	0.85	0.86
20	0.80	0.82
24	0.73	0.71
28	0.70	0.67
32	0.68	0.70
42	0.5	0.52

5.3. *Number of interventions.*

In Table 2, we can see the comparison of a single and a hierarchical classifier based on the number of tests needed to isolate the faulty component.

With very few components, there does not exist a significant difference between the two approaches. However, as we add more components, the differences increase – in favor of the hierarchical approach. As we expected, as the number of components increases, so does the complexity of the relationship between them. Our hierarchical approach is able to encode these complex relationships and boost the performance.

Nevertheless, as we continue to increase the number of components further, we see that this difference begins to decrease. By adding new components to our subset, we are introducing components with lower predictive power, and making the complete problem harder, until the benefits of hierarchy disappear.

We only show the average number of tests when the component replaced is among the selected one. Realistically, we can not consider the "other" class as equivalent, since it consists on many different components and no single test could tell us whether the faulty component is part of it or not. Since we are placing the "other" class at the top of the hierarchy, we do not expect to have significant differences between the single classifier and the hierarchical one in how they identify it.

T

Table 2. Number of tests needed to isolate the faulty component, when the faulty component is among the selected ones.

	# of Tests	
# Comp	Single	Hierarchical
12	3.16	3.45
17	7.48	7.28
20	12	7.61
24	8.86	6.89
28	6.77	5.93
32	7.52	6.96
42	7.50	7.56

If we had evaluated our hierarchical approach only based on accuracy, we would have concluded that it does not provide any measurable improvement. However, in a more realistic scenario of interaction with the technician carrying out the diagnostics, we can measure significant benefits. Our hierarchical approach does not provide a better classification performance in the per-component level; however, by grouping the components at different levels, it allows us to create better models in these intermediate levels. It is precisely this structure in the predictions that can be exploited.

We are choosing subsets of components using a ranking of how predictable they are. The more components we add, the lower the average predictive power will be. With 20 components, we were able to obtain the maximum added value of using hierarchical classification. We can assume that there exists some structure in the components and that our hierarchy has captured it.

However, that benefit disappears as we add more components to our subset. If we add a component with a low predictive power, and it is equally similar to the rest of the components, there is not a clear, logical position in the hierarchy to place it. For example, in Fig. 1, if we wanted to add a component with a predictive power of 0.6 with respect to the rest of components, we could place it together with "JoIs" and "rSir" components, but then the predictive power of that

group with respect to the "Jfds" would drop, and not be 0.92 anymore. Adding a component with low predictive power adds noise not only to our classifiers, but also to our method of extracting hierarchies.

It is critical to identify which subsets of components boost the performance of the hierarchical approach when compared to the non-hierarchical one. We need enough components so that there exists a complex structure that can be exploited, but we also need to keep in mind that components with very low predictive power will actually harm the hierarchy extraction.

5.4. *Enhancing the hierarchy*

If we want to enhance the overall performance of the diagnosis system, we need better data by, for example, adding new sensors to the machine. Encoded in the hierarchy is the information of similarity and dissimilarity between components based on their effect on the data. We can analyze this information to identify what kind of data is needed to obtain the biggest benefit.

In this experiment we simulate the installation of a new sensor that helps us enhancing the performance of the hierarchical model. It is not reasonable to assume that there exists a sensor that could distinguish between two random sets with an arbitrary number of components. However, we assume that it is possible to install a sensor that could distinguish between any two individual components.

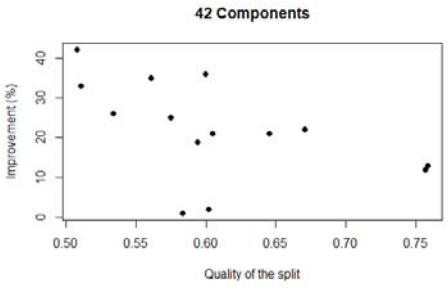

Fig. 4. Relative improvement of the number of tests to isolate the selected components as a function of the quality of the split as measured in the hierarchy.

Using the hierarchy, we can identify which pair of components are harder to distinguish. We simulate this sensor by adding a feature highly correlated with the true class when it is one of the two components and random otherwise. For this experiment, we choose the case with 42 components. We choose the 14 nodes of the hierarchy where two components are distinguished. For each node, we simulate the addition of the new sensor and reevaluate the number of tests needed to isolate those two components. In Fig. 4, we can see the relationship between the relative improvement in the number of tests and the quality of the split as measured in the height of the hierarchy. The components at those nodes that are lower in the hierarchy present the larger benefit from adding the new sensor. However, for those nodes where the classifiers were already good enough, the benefit is smaller.

6. Conclusions and future work

In this paper, we have shown that multi-class classification can be used in the field of maintenance in general, and fault diagnosis in particular.

We have introduced a hierarchy of components as a logical way to structure the output of our models. The motivation to use hierarchies is that there exists a structure in how the components relate to each other, i.e., we assume that some components are more similar than the others in how they affect the data. We have shown that this assumption holds and that this type of structure has proven to give measurable benefits when compared to approaches that do not take into account the structure of components.

We have established a framework for how the field technician and the classification models can interact. We have evaluated our models with a simple measure of usefulness. We have shown that the most traditional machine learning metrics fail to capture the practical utility in this framework of interaction.

A hierarchy comes with its inconveniences. The main prerequisite for a hierarchy to be useful is that each of the components has different relationships with the rest of the components. If this assumption does not hold, and there is no structure in how components relate to each other, a hierarchy might not be the best option.

It has proven to be crucial to identify which components to include in the hierarchy and which not. With the right selection, we can see significant improvements in a hierarchical classifier over a non-hierarchical one. However, these benefits can disappear if we include the wrong components.

One of the main advantages of this type of hierarchies of components is that it allows us to identify where the models are failing, allowing us to redesign and enhance our diagnostic solution. We have shown a simple example of how this can implemented, by simulating the addition of a sensor to distinguish between two specific components.

We believe that the benefits of using extracted hierarchies shown in this paper are just the starting point of how hierarchies can be exploited.

Hybrid hierarchies that incorporate the expert

knowledge and automatically generated relationships between components look like a promising idea. We can add multi-component testing, and have a more precise evaluation based on the different costs of different tests. We can enhance our diagnostic solution by adding sensors that can distinguish different groups with different number of components, and this information can be coded as well in the hierarchy.

Finally, we have limited ourselves to work with multi-class classification. However, in this type of systems, several components can be faulty at the same time. The obvious next step in this direction would be to introduce multi-label classification. Multi-label classification allows for a new instance to be classified into more than one class.

References

Baraldi, P., F. Cannarile, F. Di Maio, and E. Zio (2016). Hierarchical k-nearest neighbours classification and binary differential evolution for fault diagnostics of automotive bearings operating under variable conditions. *Engineering Applications of Artificial Intelligence 56*, 1–13.

Canizo, M., E. Onieva, A. Conde, S. Charramendieta, and S. Trujillo (2017). Real-time predictive maintenance for wind turbines using Big Data frameworks. *2017 IEEE International Conference on Prognostics and Health Management, ICPHM 2017*, 70–77.

Cheng, W., K. Zhang, H. Chen, G. Jiang, Z. Chen, and W. Wang (2016). Ranking causal anomalies via temporal and dynamical analysis on vanishing correlations. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 805–814.

Lei, Y., B. Yang, X. Jiang, F. Jia, N. Li, and A. K. Nandi (2020). Applications of machine learning to machine fault diagnosis: A review and roadmap. *Mechanical Systems and Signal Processing 138*, 106587.

Li, Z., Y. Wang, and K. S. Wang (2017). Intelligent predictive maintenance for fault diagnosis and prognosis in machine centers: Industry 4.0 scenario. *Advances in Manufacturing 5*(4), 377–387.

Michau, G., T. Palme, and O. Fink (2017). Deep feature learning network for fault detection and isolation. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society, PHM*.

Verhagen, W. J. and L. W. D. Boer] (2018). Predictive maintenance for aircraft components using proportional hazard models. *Journal of Industrial Information Integration 12*, 23 – 30.

Wen, L., X. Li, L. Gao, and Y. Zhang (2018). A new convolutional neural network-based data-driven fault diagnosis method. *IEEE Transactions on Industrial Electronics 65*(7), 5990–5998.

Yan, J., Y. Meng, L. Lu, and L. Li (2017). Industrial Big Data in an Industry 4.0 Environment: Challenges, Schemes, and Applications for Predictive Maintenance. *IEEE Access 5*, 23484–23491.