Exploiting Domain Knowledge to address Multi-Class Imbalance and a Heterogeneous Feature Space in Classification Tasks for Manufacturing Data

Vitali Hirsch
Daimler Truck AG
vitali.hirsch@daimler.com

Peter Reimann
University of Stuttgart, GSaME
peter.reimann@gsame.uni-stuttgart.de

Bernhard Mitschang
University of Stuttgart, IPVS
bernhard.mitschang@ipvs.uni-stuttgart.de

ABSTRACT

Classification techniques are increasingly adopted for quality control in manufacturing, e. g., to help domain experts identify the cause of quality issues of defective products. However, real-world data often imply a set of analytical challenges, which lead to a reduced classification performance. Major challenges are a high degree of multi-class imbalance within data and a heterogeneous feature space that arises from the variety of underlying products. This paper considers such a challenging use case in the area of End-of-Line testing, i. e., the final functional test of complex products. Existing solutions to classification or data pre-processing only address individual analytical challenges in isolation. We propose a novel classification system that explicitly addresses both challenges of multi-class imbalance and a heterogeneous feature space together. As main contribution, this system exploits domain knowledge to systematically prepare the training data. Based on an experimental evaluation on real-world data, we show that our classification system outperforms any other classification technique in terms of accuracy. Furthermore, we can reduce the amount of rework required to solve a quality issue of a product.

PVLDB Reference Format:
Vitali Hirsch, Peter Reimann, Bernhard Mitschang. Exploiting Domain Knowledge to address Multi-Class Imbalance and a Heterogeneous Feature Space in Classification Tasks for Manufacturing Data. *PVLDB*, 13(12): 3258 - 3271, 2020.
DOI: https://doi.org/10.14778/3415478.3415549

1. INTRODUCTION

Manufacturing companies increasingly apply data-driven classification techniques to enhance tasks for product quality control [23, 38]. However, the characteristics of many real-world manufacturing data imply several analytical challenges that have a negative effect on the classification performance. For instance, real-world data usually contain a multiplicity of class labels that occur in an imbalanced way (multi-class imbalance) [16, 22, 36, 41]. Many learning algorithms tend to ignore the class labels that occur less frequently. In addition, underlying data often represents heterogeneous product variants with different physical properties [17, 38]. This increases the heterogeneity in the feature space, e. g., the value ranges representing certain class patterns differ among individual product variants. This finally makes it hard to detect clearly distinguishable patterns [18].

These challenges may be exemplified by a real-world use case we have developed at Daimler Truck AG. The use case comes from the area of End-of-Line (EoL) testing of assembled powertrain aggregates, e. g., engines. EoL testing constitutes the final functional check of such products after assembly. Thereby, product characteristics are tested based on sensor signals, e. g., the oil consumption of an engine. If sensor signals do not meet pre-defined tolerance limits, a quality issue is assumed. Then, quality engineers manually evaluate the sensor signals from the test bench and try to identify the faulty product component that causes the quality issue, e. g., a turbo charger. This task is referred to as *fault isolation* [21]. Operators replace the assumed faulty component, and the product is tested again. However, even experienced quality engineers need on average four attempts to identify the correct faulty component [17].

By considering the faulty components as classes and the sensor signals as features, we can transform fault isolation into a classification problem. This may help quality engineers and operators to determine the cause of a quality issue in a faster way by automatically recommending the most likely faulty product components. Assembled powertrain aggregates are composed of numerous individual components. Each of these components corresponds to one particular class label in the data. The majority of these class labels occur in only a small number of data samples, leading to a high degree of multi-class imbalance. Furthermore, powertrain aggregates come in diverse product variants. This product variety leads to a heterogeneous feature space.

Various research communities work on the design and optimization of algorithms for machine learning and data engineering. In a previous study, we have evaluated several common algorithms to see if they are suitable to meet the analytical challenges of our use case [18]. This mainly included algorithms for sampling, feature selection, binarization, and classification. The main conclusion of this study is that most of these algorithms are suitable to mitigate the negative effects of one particular analytical challenge. However, they worsen the effects of other challenges. Finally,

Proceedings of the VLDB Endowment, Vol. 13, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3415478.3415549

the application of these algorithms that are tailored to single challenges even reduces prediction performance. Thus, we argue that a classification system has to consider different analytical challenges and their mutual influences together.

In this paper, we propose a novel classification system that addresses both a multi-class imbalance and a heterogeneous product and feature space. In contrast to related work, we do not optimize or enhance any sophisticated algorithms. Instead, we make explicit use of knowledge that is available in the domain at hand to systematically prepare the training data and to mitigate the effects of the addressed challenges. Thereby, we make the following main contributions:

- We advance the understanding of classification tasks for quality control and fault isolation in manufacturing by providing insights into real-world data characteristics that represent obstacles for classification algorithms.We thereby bring attention back to analytical challenges of small data in the current big data era.
- We propose a design alternative for a classification system that effectively addresses these analytical challenges. The major contribution is that we explicitly use domain knowledge from a product hierarchy to segment the training data set into several sample subsets. Thereby, each subset corresponds to one product group with technically similar product variants. Hence, the sensor signals and thus the feature space within one sample subset are much more homogeneous. Furthermore, we discuss metrics to obtain information about the class distribution within the resulting sample subsets. Our system uses these metrics to come up with informed decisions how to further partition the subsets, thereby addressing multi-class imbalance.
- We discuss results of an evaluation of our classification system based on its application to the data of the EoL testing use case. The major outcome is that our system outperforms any baseline solution. It shows a higher prediction performance in terms of accuracy. Furthermore, it reduces the number of rework attempts that are required to solve a quality issue. In addition, we discuss generality issues, i. e., how our approach may be applied to other use cases than EoL testing.

The remainder of this paper is structured as follows: In Section 2, we detail the analytical challenges that are prevalent in real-world manufacturing data. In Section 3, we discuss related work. Section 4 describes our novel classification system. Section 5 discusses the results of our evaluation. Finally, we conclude and list future work in Section 6.

2. ANALYTICAL CHALLENGES

Fault isolation in manufacturing may typically be mapped to a *single-label multi-class* classification. Each observation, i. e., quality issue, is associated with exactly one of $C > 2$ class labels $c_i \in \mathcal{C} = \{c_1, \ldots, c_C\}$ that correspond to the possibly faulty product components [18]. We train a classifier $\mathcal{M}$ on a historical data set $\mathcal{X}$ with N samples (x_t, y_t). Each x_t is an element of an M-dimensional *feature space* $\mathcal{F} = \{f_1, \ldots, f_M\}$. The sensor signals $s_k \in \mathcal{S} = \{s_1, \ldots, s_S\}$ from the EoL test bench are part of $\mathcal{F}$, i. e., $\mathcal{S} \subset \mathcal{F}$. y_t represents the target class label c_i associated with x_t.

In our previous study, we derived analytical challenges for manufacturing data [18]. The first challenge is that the

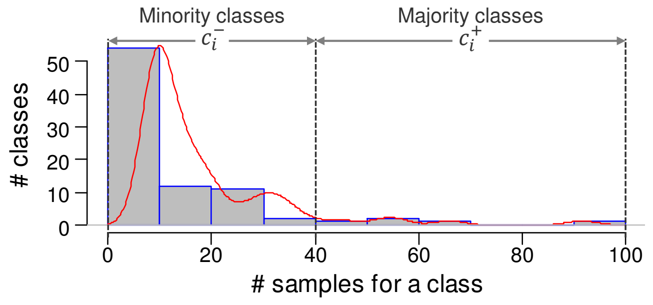

Figure 1: Class distribution for sample set $\mathcal{X}$. The grey boxes represent the histogram and the red line the density function of the class distribution.

sample set $\mathcal{X}$ is of small size (C1), i. e., it only contains $N = 1.050$ samples with 84 classes c_i and 115 features f_m. Further challenges are a multi-class imbalance (C2) and the heterogeneous product portfolio that increases the heterogeneity in the feature space $\mathcal{F}$ (C3). Our previous study reveals that many existing algorithms for sampling, binarization, feature selection, and classification can cope with a small amount of data (C1). However, they still struggle especially with the *combination* of multi-class imbalance (C2) and heterogeneous product portfolio (C3) [18]. This is the reason why the approach we propose in this paper addresses challenges C2 and C3. In the following, we analyze these two challenges in more detail compared to our previous study.

2.1 Multi-Class Imbalance

Figure 1 shows the distribution of all 84 classes (y-axis) with the amount of corresponding samples (x-axis). Most classes have fewer than 10 samples (cf. peak on the left), while only a few classes have more than 40 samples (cf. right side). The top 5 classes together are contained in 29% of all samples in $\mathcal{X}$, where each individual class has a comparatively high share of at least 4%. We denote such top classes as *majority classes* c_i^+ and their samples as *majority samples* $\mathcal{X}^+$. Each of the remaining 79 classes individually has a rather low share of samples, but all together are represented by 71% of the samples. We denote them as *minority classes* c_i^- that are represented by *minority samples* $\mathcal{X}^-$.

This uneven class distribution poses a multi-class imbalance problem. Most classification algorithms tend to ignore minority samples $\mathcal{X}^-$. This is because they try to maximize accuracy by predicting everything to be one of the majority classes c_i^+. Hence, the resulting classifier $\mathcal{M}$ is biased towards the majority classes c_i^+ [14]. We however require a classifier $\mathcal{M}$ that provides a balanced degree of accuracy for both the minority and majority classes. In a worst-case scenario, we are otherwise only able to predict 5 majority classes c_i^+, while the 79 minority classes c_i^- are ignored.

2.2 Heterogeneous Product Portfolio

The samples in our set $\mathcal{X}$ describe a variety of product variants, which differ in their technical specification. The manufacturing domain groups products with technically similar characteristics into product groups and organizes them in a *product hierarchy*. Figure 2 shows the product hierarchy for the engines of our use case. The first hierarchy level differentiates engines according to their *series*, i. e., whether they are heavy-duty or medium-duty engines

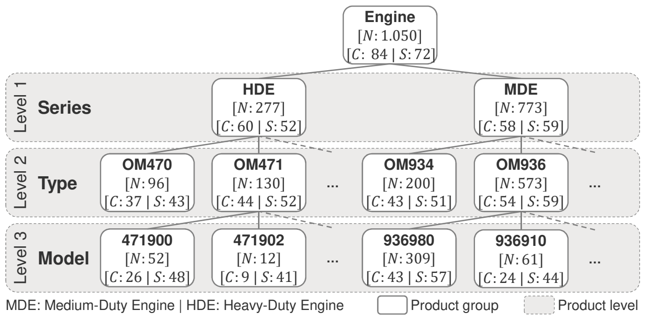

Figure 2: Product hierarchy and respective numbers of samples N, classes C and sensor signals S for different product groups in the sample set $\mathcal{X}$.

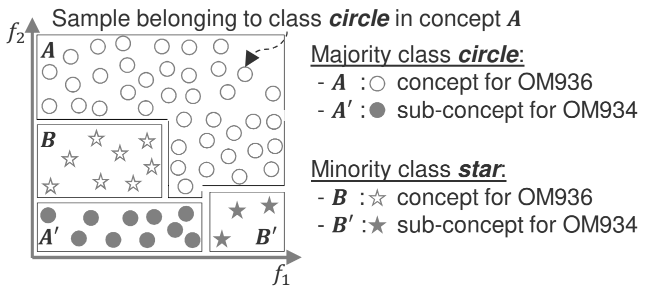
Sample belonging to class circle in concept A
f_2
f_1
A
B
A'
B'
Majority class circle:
- A : ○ concept for OM936
- A' : ● sub-concept for OM934
Minority class star:
- B : ☆ concept for OM936
- B' : ★ sub-concept for OM934

Figure 3: Illustration of analytical challenge C3.2 with two classes *circle* and *star*. Rectangles define the actual concepts, i. e., the decision rules.

(HDE, MDE). Level 2 divides these two groups into different *engine types*, e. g., group $OM934$ comprises four-cylinder $MDEs$, while $OM936$ comprises six-cylinder $MDEs$. The bottom level describes *engine models* that further specify the components of an engine. For example, to measure the oil level, group 936980 has a digital component, while group 936910 has an analog oil dipstick. The inherent product variety leads to the following three issues for classification.

(C3.1) Missing features: Some sensor signals s_k are only measured for certain variants. For example, each cylinder of an engine delivers one particular sensor signal s_k. So, product variants in group $OM934$ comprise four sensor signals for cylinders, while six-cylinder variants in the group $OM936$ comprise two additional signals. The data structure of the sample set $\mathcal{X}$ contains a column for each of the overall 72 sensor signals. So, a sample of a four-cylinder variant has six columns for cylinders, whereas two of these columns do not contain any value (i. e., "NA"). This issue leads to several missing feature values in the whole sample set $\mathcal{X}$. In our use case, about 17% of all values in $\mathcal{X}$ are "NA" values. To train a classifier $\mathcal{M}$, the missing values must be *imputed* or *removed*. However, we then either create artificial values for some features f_m that have no technical causality to a particular product variant, or we remove features that may characterize a specific class c_i. For the group $OM934$ with four cylinders, imputing values would mean that we generate artificial sensor signals s_k for the cylinders No.5 and No.6 that are however physically not available. By removing the sensor signals s_k for cylinders No.5 and No.6, we lose information for the group $OM936$ with six cylinders. Without these two sensor signals, it is likely that no error patterns for the cylinders No.5 and No.6 exist anymore in the data.

(C3.2) Sub-concepts: For different product variants, the same class c_i may be defined by the same features, but with different value ranges. Hence, a classifier $\mathcal{M}$ may have to learn multiple concepts for a single class. This even reduces the number of samples that is covered by each concept. This challenge is called *sub-concepts* in literature [16, 36]. Figure 3 shows an artificial data set with two example features and two classes. For example, the figure shows concept B for product group $OM936$ with six-cylinder variants and sub-concept B' for product group $OM934$ with four-cylinder variants. Both B and B' describe patterns for class star with the same features f_1 and f_2. However, B is characterized by low f_1 and medium f_2 values, while samples of B' have high f_1 and low f_2 values. While there is a sufficient number of samples for concept B, sub-concept B' is only represented by three minority samples of class star. This lack of representative samples in B' may cause a learning algorithm to neglect the three minority samples and thus the whole sub-concept B'. The feature ranges for B' are then assigned to the wrong concepts A or A' characterizing class circle.

(C3.3) Class membership: Not all product components are used throughout all product variants. For instance, cylinders No.5 and No.6 are only available in six-cylinder engines, but not in four-cylinder variants. Therefore, not all classes c_i are relevant for all product variants. We refer to this as *class membership* problem. A classifier $\mathcal{M}$ might recommend a faulty component that is not part of the relevant product. This leads to useless recommendations, which even reduces user acceptance of the classification system.

3. RELATED WORK

The first group of related work is associated to imbalanced learning, where numerous methods are summarized in review articles [11, 14–16, 26, 36]. However, the reviews show that most techniques from imbalanced learning are designed for two-class problems and are hence less effective for multi-class tasks [46]. Most solutions firstly use *class decomposition schemes* such as *One-vs-All* (OvA) to convert a multi-class problem into several two-class problems. Then, they work with two-class imbalance techniques to balance each binary sub-problem [43]. However, our previous study shows that such decomposition schemes reduce prediction performance, because they even increase issues with imbalance and with sub-concepts (C2 and C3.2) [18].

The few exceptions that explicitly deal with multi-class imbalance are *cost-sensitive* techniques that consider costs as penalty of different types of misclassification [11,16,36,43] [11, 16, 43]. One difficulty with such techniques is that the real costs are often unknown or hard to calculate for a given problem [15, 36]. In addition, cost-sensitive techniques are often limited to avoiding misclassification of minority classes c_i^-. This is because literature assumes that misclassification of minority classes is more costly. In applications such as detecting the type of cancer, this is welcome. In manufacturing, the misclassification costs are based on *monetary costs* of materials and the time taken by humans to complete a task. However, there is no clear dependency between monetary costs and the number of samples per class. In a worst case, majority classes c_i^+ may have higher monetary costs than minority classes c_i^-. Thus, cost-sensitive techniques would focus on majority classes and ignore minority classes.

The second group of related work considers the missing feature problem (C3.1). One approach uses ensembles that employ *random subspace selection* [28,30]. The essence is to generate a large number of *base learners* $\mathcal{L}_j$, while each $\mathcal{L}_j$ is trained with a random subset of features from the feature space $\mathcal{F}$. To classify a new sample x_t with missing features, only those base learners $\mathcal{L}_j$ trained with the features that are available in x_t are used. However, Polikar et al. [30] show that this paradigm has two assumptions: First, the set of features in $\mathcal{F}$ is partly *redundant*, so that the classification problem is solvable with a real subset of the features. Second, this redundancy is distributed randomly over the features in $\mathcal{F}$. Our previous study shows that the concepts in our sample set $\mathcal{X}$ are rather complex, since only 10 out of 115 features were rated as redundant by feature selection techniques [18]. So, the assumptions for these approaches to random subspace selection do not hold for our case.

The next group uses *hierarchical classification* to ensure class membership (C3.3) [33]. The assumption is that multiple classes have shared characteristics and relationships that may be used to organize the classes in a tree-like *class hierarchy*. We can now train a classifier $\mathcal{M}_l$ for each level l. Based on the prediction of the classifier $\mathcal{M}_l$, we apply the next classifier $\mathcal{M}_{l+1}$ one level lower. We can repeat this until we recognize our faulty components, i.e., classes c_i, on the leaf nodes. However, in our scenario, there are relationships between classes c_i and product groups in the sense that a product group restricts the subset of classes, i.e., the components that may be part of certain products. However, the classes themselves do not have relationships between each other. Hence, we cannot build a class hierarchy that may be used for classification as described above.

In summary, numerous techniques exist that address single challenges in isolation. However, these techniques imply other negative effects to classification. Our key statement is that a classification system has to consider all relevant challenges as well as their mutual influences. Here, we opt for an approach that explicitly uses available domain knowledge to systematically prepare the training data. Related approaches use domain knowledge to build logical rules or semantic descriptions of causal dependencies between several features or between features and class labels (e. g., [35,40]). These rules may be used for feature engineering or directly for classification. However, no appropriate knowledge about causalities among features or between features and classes exists in the domain of EoL testing. In contrast, any manufacturing company possesses domain knowledge in terms of a documentation of a company's product family, e. g., a product hierarchy [1]. A product hierarchy is not suitable to transform the feature space or to build classification rules. Instead, we may use the clearly distinguishable product groups described by the hierarchy to partition the sample set $\mathcal{X}$ into several sample subsets. In the following section, we describe how this and further steps to prepare the data helps to address both challenges C2 and C3 at once.

4. CLASSIFICATION SYSTEM

Figure 5 shows our classification system that addresses both analytical challenges C2 and C3 together. As mentioned before, the core idea is to exploit available domain knowledge in a training set preparation phase (4.1). Thereby, we partition the whole sample set $\mathcal{X}$ into several subsets in which the negative effects of analytical challenges C2 and C3 are mitigated. Figure 4 depicts this decomposition strategy. Afterwards, we describe how this strategy affects the *predictive modeling*, where we train multiple classifiers $\mathcal{M}$ for the sample subsets and combine the results of these classifiers to obtain a final *recommendation list* $\mathcal{R}_e$ (4.2).

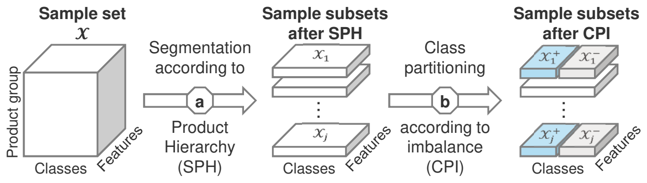

Figure 4: Decomposition strategy for (a) segmentation according to product hierarchy (SPH), and (b) class partition according to imbalance (CPI).

4.1 Training Set Preparation

The main step of the training set preparation and thus the major contribution of our classification system is the *segmentation according to the product hierarchy* (SPH) (4.1.1). Here, we use the knowledge from this hierarchy to divide our sample set $\mathcal{X}$ according to individual product groups and to obtain sample subsets with technically similar product variants. So, the sensor signals s_k within each subset are more homogeneous, i. e., SPH addresses challenge C3. The next step is a *class partitioning according to imbalance* (CPI) addressing challenge C2 (4.1.2). Here, we discuss the kinds of metrics that may be used to obtain relevant information on the class distributions of the sample subsets resulting from the preceding SPH. We use these metrics to make informed decisions on whether and how the subsets should be divided further among majority and minority classes. Finally, we briefly describe how to further pre-process the resulting sample subsets (4.1.3) for the subsequent training phase.

4.1.1 Segmentation according to Product Hierarchy

The standard design principle for classification is that a classifier $\mathcal{M}$ is trained on the entire sample set $\mathcal{X}$, i. e., on all samples available in the root "*Engine*" of the product hierarchy shown in Figure 2. In our approach, we however divide the sample set $\mathcal{X}$ into several subsets according to lower levels of the product hierarchy. For the hierarchy level "*Type*" as example, we generate subsets $\mathcal{X}_{(l,j)}$ for each product group from $OM470$ to $OM936$, where j indexes the product group on the hierarchy level l. For reasons of simplification, we use the notation $\mathcal{X}_j$ for a product group j on a certain level l if the level is obvious from the context.

By considering only technically similar product variants in one sample subset $\mathcal{X}_j$, we reduce the heterogeneity in the sensor signals s_k and hence mitigate the missing feature problem (C3.1). For example, by considering the four-cylinder product group $OM934$ on its own, the samples in the relevant subset $\mathcal{X}_j$ still have sensor signals for cylinders No.1 to 4. However, $\mathcal{X}_j$ does not contain signals for cylinders No.5 and 6, which actually do not exist in this product group. As $\mathcal{X}_j$ now comprises samples from similar product variants, we also reduce the variety in the value ranges of sensor signals s_k. This leads to a decreased number of subconcepts (C3.2). Furthermore, each subset $\mathcal{X}_j$ does not contain any classes anymore that are irrelevant for its product group, e. g., cylinders No.5 and 6 for group $OM934$ (C3.3).

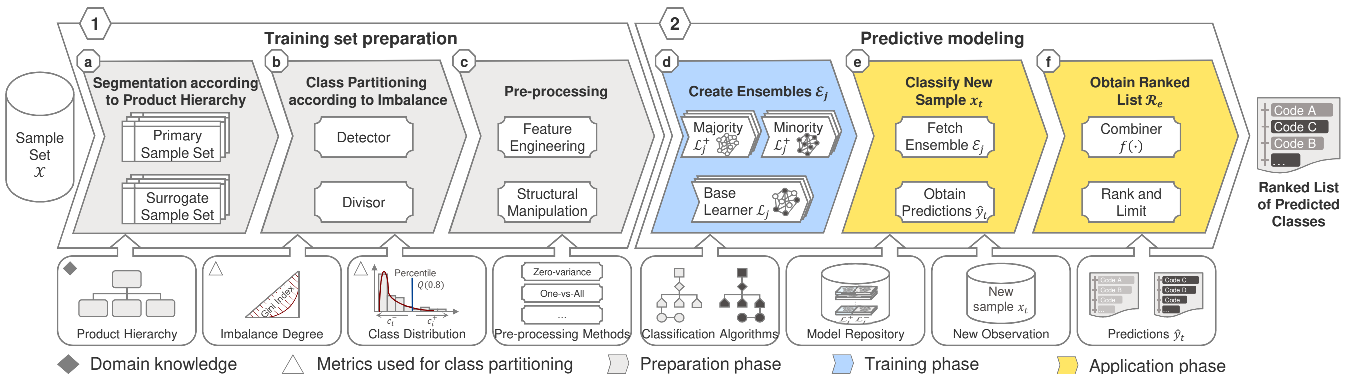

Figure 5: Classification system that exploits domain knowledge from the product hierarchy and reasonable metrics to address the analytical challenges discussed in Section 2. The whole system is composed of two major components: 1) training set preparation and 2) predictive modeling. The rectangles at the bottom of the figure indicate the inputs for the individual sub-components from (a) to (f).

It is critical to select a proper hierarchy level l when performing the segmentation into sample subsets for individual groups j. The deeper we go into the product hierarchy, the more we reduce the heterogeneity in the feature space. However, if the chosen level l is too deep, the number of samples for a particular group j may be too small to train a classifier. For example, the group 471902 at the level "*Model*" has only twelve samples to characterize nine classes c_i. Thus, almost every class in the group 471902 is represented by only one sample, so that no reasonable decision rules can be learned. Other groups on the same hierarchy level may have enough samples though. For example, the group 936980 has a sufficient number of 309 samples for its 43 classes. To handle these different product groups at a specific hierarchy level l, we introduce *primary* and *surrogate* sample subsets.

Primary sets contain samples of product groups that are located at a particular deeper level in the product hierarchy, i. e., a level where the groups are technically similar as much as possible. This way, we may mitigate the effects of challenge C3 in these primary sample sets to the greatest extent possible. In our product hierarchy shown in Figure 2, this is the third level "*Model*". Here, we may use the 309 samples of the group 936980 as a primary set. However, the sibling group 471902 does not contain enough samples to train a classifier. For such groups, we introduce surrogate sample sets that are located at least one hierarchy level higher. As a result, a surrogate set is less specific, but it contains more samples. For instance, we use group $OM471$ at level "*Type*" as a surrogate to represent group 471902 one level lower. Group $OM471$ contains 130 samples, which is enough to train a reasonable classifier. We later use this classifier for group $OM471$ as a surrogate to predict the class for new samples that belong to group 471902 (cf. Section 4.2).

To create primary and surrogate sample sets, we traverse the product hierarchy from the bottom to the top. We first specify the *maximum depth* (max_d) of the hierarchy to be examined. In our use case, we start the hierarchy traversal at the third level "*Model*", i. e., we set max_d to 3. We then intend to create a primary sample set for each product group j at the selected level max_d. Therefore, we check each group j at this level $l = max_d$ to see whether the primary sample subset $\mathcal{X}_{(l,j)}$ fulfills the requirements to train a classifier $\mathcal{M}_{(l,j)}$. However, it may happen that $\mathcal{X}_{(l,j)}$ exhibits an unfavorable sample distribution after SPH. For example, a certain class in $\mathcal{X}_{(l,j)}$ may now be characterized by only a few samples. Hence, we use a threshold for a *minimum number of samples* for a class. If a class has fewer samples than this threshold, we assume that no meaningful pattern can be learned for this class. So, we remove this class and its samples from the subset $\mathcal{X}_{(l,j)}$. For our use case, we have set the threshold to 2 to preserve as many classes as possible. This is because many classes show a low number of samples.

Afterwards, we check with two criteria whether it is reasonable or not to train a classifier $\mathcal{M}_{(l,j)}$ for a subset $\mathcal{X}_{(l,j)}$. The first criterion ensures that $\mathcal{X}_{(l,j)}$ contains at least two classes. With the second criterion, we assure that the *loss of information* due to the previous removal of classes with too few samples is not higher than another threshold parameter. We check that the removal of classes does not reduce the number of samples in $\mathcal{X}_{(l,j)}$ by more than, e. g., 25%-points. Otherwise, the risk increases that the loss of information overcompensates the positive effect of segmentation.

If a sample subset $\mathcal{X}_{(l,j)}$ meets both criteria, it becomes a primary set. If it does not meet at least one of these two criteria, we first visit the parent node of group j one level higher, i. e., $l = max_d - 1$, and check whether its sample set meets both criteria. If this is the case, we consider the sample set of this parent node as a surrogate set for the group j. Otherwise, we further traverse the hierarchy along the path to the root node. We do so until we are able to represent each group j by either its primary sample set at level max_d or by a surrogate set at a higher level.

Note that already this SPH reduces the class imbalance to a certain degree (C2). Figure 6 shows example distributions of the most frequent classes c_i (error codes A to G) after the segmentation into the groups $OM934$ ($\mathcal{X}_1$) and $OM470$ ($\mathcal{X}_2$). Only error code A occurs in both groups $OM934$ and $OM470$. If we consider both groups together, i. e., without performing the segmentation, this leads to a significant imbalance. Error code A is then the most common class with in total 61 samples. All other classes comprise between 5 and 16 samples only. Our SPH allows us to consider the groups $OM934$ and $OM470$ separately. Therefore, error code A has only 10 samples in $OM470$ and is thus not a dominant class anymore for this group. So, the segmentation positively influences the class imbalance for group $OM470$ (C2).

4.1.2 Class Partitioning according to Imbalance

In the sample set of group $OM934$ shown in Figure 6 however, error code A remains a majority class c_i^+ with 51 samples. In fact, about 46% of our primary and especially surrogate sample subsets contain such very dominant majority classes c_i^+. This still leads to a distinct class imbalance (C2). We hence perform a class partitioning to create disjoint majority subsets $\mathcal{X}_j^+$ and minority subsets $\mathcal{X}_j^-$ for each subset $\mathcal{X}_j$ that shows a distinct degree of class imbalance (cf. Figure 4(b)). For the group $OM934$ as example, we separate error code A from the remaining codes, i. e., we create a majority subset $\mathcal{X}_1^+$ with all samples belonging to error code A and a minority subset $\mathcal{X}_1^-$ with all other samples (cf. Figure 6). This way, we significantly reduce the class imbalance in the subsets $\mathcal{X}_1^+$ and $\mathcal{X}_1^-$ (C2). Thereby, we ensure that learning algorithms can recognize the error codes B to D within $\mathcal{X}_1^-$, which otherwise would possibly be ignored due to the dominance of error code A in the subset $\mathcal{X}_1$.

Prior to a class partitioning for a product group j, we first decide whether there is a distinct class imbalance in the subset $\mathcal{X}_j$. For example, the group $OM470$ in Figure 6 does not have a dominant imbalance so that no further partitioning is required. The group $OM934$, however, may benefit from partitioning. The *detector* sub-component shown in Figure 5(b) uses a statistical metric to make an informed decision whether a sample subset $\mathcal{X}_j$ has to be partitioned further or not. Afterwards, the *divisor* sub-component provides the necessary formalism how to appropriately partition $\mathcal{X}_j$ into majority and minority sample subsets $\mathcal{X}_j^+$ and $\mathcal{X}_j^-$.

Detector: There is no consensus on proper statistical metrics to determine the degree of class imbalance within data [11]. For our use case, the metric should have a normalized interval between 0 and 1, where the boundaries represent a *total balance* (0) or a *total imbalance* (1). This allows us to directly compare the results of the metric between all subsets $\mathcal{X}_j$. It also makes it easier to parameterize the detector component, because we can define one global threshold instead of parameterizing each subset $\mathcal{X}_j$ individually.

Cowell discusses several *inequality metrics* that meet our requirement [9]. The most prominent metrics are the *Hoover*, *Theil*, and *Gini* indexes. For our use case, we have tested these three indexes, each with seven threshold values from 0.1 to 0.7 in 0.1-steps to decide whether a subset $\mathcal{X}_j$ shall be partitioned or not. Finally, we opted for the Gini index, because we got the highest prediction performance in terms of accuracy with it. In addition, the domain experts of our use case have rated both the way to calculate the Gini index and its visualization using the *Lorenz curve* [9] as intuitive.

The Gini index of the samples in Figure 6 is about 0.28 for group $OM470$ and about 0.48 for group $OM934$. With a threshold value of 0.3, we detect a distinct class imbalance for group $OM934$. We divide the subset $\mathcal{X}_1$ in the next step into the disjoint subsets $\mathcal{X}_1^+$ and $\mathcal{X}_1^-$. For $OM470$ with a Gini index below 0.3, we do not perform a class partitioning. Our tests with different threshold values for our use case showed the best accuracy with a threshold of 0.3. Hence, we use this threshold as a global parameter for all subsets $\mathcal{X}_j$. This threshold is rather low, because we usually have only a few dominant classes with numerous samples in a subset $\mathcal{X}_j$, while most other classes are more balanced within $\mathcal{X}_j$.

Divisor: Similar to the Detector, we may use various metrics to determine the *point of intersection* to partition a set $\mathcal{X}_j$ into a majority set $\mathcal{X}_j^+$ and a minority set $\mathcal{X}_j^-$. The result of such a metric has to be a numerical number, which then acts as a threshold. Classes with more samples than the threshold are placed in the majority set $\mathcal{X}_j^+$, and the remaining ones in the minority set $\mathcal{X}_j^-$. Typical examples of metrics are the *arithmetic mean* or *standard deviation*. For our use case, we have chosen the *p-quantile* $Q(p)$. The major reason is that $Q(p)$ is the only of these metrics that allows for a parameterization with p to tune it for an improved prediction performance. We have tested different values for p, i. e., we started with 0.6 and increased p to 0.9 in 0.1-steps. Finally, we achieved the best performance with $p = 0.8$.

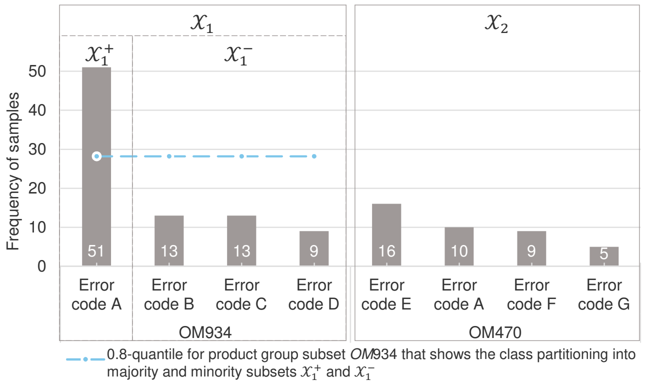

Figure 6: Class distribution after SPH ($\mathcal{X}_1$ and $\mathcal{X}_2$) and after CPI ($\mathcal{X}_1^+$ and $\mathcal{X}_1^-$). Error codes A to G represent the individual classes c_i in the sample subsets.

The calculation of $Q(p)$ is based on the *empirical cumulative distribution function* $F(x) = p$. Thereby, x is a number of samples and p is the relative share of classes in a subset $\mathcal{X}_j$ that are represented by x or less samples. Figure 7 shows the results of $F(x)$ for group $OM934$. For instance, $F(13)$ is 0.75, because three of four classes in $\mathcal{X}_1$ are represented by 13 or less samples. $Q(p)$ is calculated using the *inverse function* of $F(x)$, i. e., $Q(p) = F^{-1}(x)$. This means that those classes in $\mathcal{X}_j$ that are represented by $Q(p)$ or less samples cover a share of p of the class distribution of $\mathcal{X}_j$. With $p = 0.75$ for group $OM934$, we get $Q(0.75) = 13$, because the value of $F(13)$ is exactly 0.75. The idea is that we set a value for p, calculate $Q(p)$ for each subset $\mathcal{X}_j$ with a distinct class imbalance, and then place all classes with $Q(p)$ or less samples in the minority subset $\mathcal{X}_j^-$ and the remaining classes in the majority subset $\mathcal{X}_j^+$. This way, all minority subsets $\mathcal{X}_j^-$ cover a share of close to p of all classes in the original subsets $\mathcal{X}_j$. This again motivates our choice for using $Q(p)$ instead of other metrics, as we may influence the relative portions of majority and minority classes via parameter p.

The classes in our subsets $\mathcal{X}_j$ are distributed discretely, so that we are not able to calculate continuous distribution functions $F(x)$. For group $OM934$, we have only three values $F(9) = 0.25$, $F(13) = 0.75$ and $F(51) = 1$. As we do not have any x with $F(x) = 0.8$ for this group, we are not able to calculate the 0.8-quantile $Q(0.8)$. For this reason, we approximate the discrete distribution function $F(x)$ using *linear interpolation*, so that we are able to estimate the number of samples x for any p. We use linear interpolation, because it is efficient and simple to apply. In addition, we do not require an exact result for $Q(p)$, but only an estimate about the number of samples to define a class as a majority

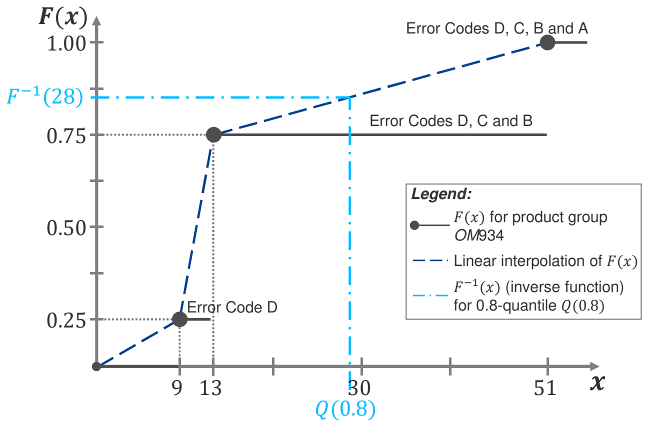

Figure 7: Calculation of 0.8-quantile ($Q(0.8)$) for product group $OM934$: Cumulative number of samples are reported on the x-axis and the empirical cumulative distribution function $F(x)$ on the y-axis.

class. The reason is that we finally apply $Q(p)$ to each subset $\mathcal{X}_j$ and that the class distribution of each $\mathcal{X}_j$ is discrete as well. For group $OM934$ as example, we only require a value of $Q(p)$ between 13 and 51 samples to separate the majority class from the minority classes.

As shown in Figures 6 and 7, the interpolated 0.8-quantile for group $OM934$ is about 28. Statistically speaking, all classes c_i having 28 or less samples represent in total 80% of the class distribution of group $OM934$. As mentioned before, we consider $Q(0.8) = 28$ as a threshold. So, error code A, which has more than 28 samples, is the only majority class c_i^+ and its samples are majority samples $\mathcal{X}_1^+$. Error codes B, C and D are minority classes c_i^- and their samples are minority samples $\mathcal{X}_1^-$. As shown in Figure 6, all resulting subsets $\mathcal{X}_1^+$, $\mathcal{X}_1^-$ and $\mathcal{X}_2$ are much more balanced. CPI reduces the Gini index from 0.48 for the subset $\mathcal{X}_1$ of group $OM934$ to a value of 0.19 for $\mathcal{X}_1^+$ and to 0.11 for $\mathcal{X}_1^-$.

For the rest of the paper, we denote the different subsets we have for product group j as $\mathcal{X}_j^\pm$. Thus, a subset $\mathcal{X}_j^\pm$ includes either the subset $\mathcal{X}_j$ in case no class partitioning has been performed or $\mathcal{X}_j^+$ and $\mathcal{X}_j^-$ in the other case.

4.1.3 *Pre-processing*

All subsets $\mathcal{X}_j^\pm$ must satisfy technical criteria, so that we can apply learning algorithms to them. We distinguish two tasks here, which are depicted in Figure 5, 1c.

Feature engineering: We remove *sparse*, *zero-* and *near-zero-variance* features f_m, as these can cause a classifier $\mathcal{M}$ to fail [18, 24]. Then, we *normalize* continuous values and perform *one-hot encoding* for categorical features [4].

Structural manipulation: Standard classification algorithms require at least two classes to train a classifier $\mathcal{M}$. However, some majority subsets $\mathcal{X}_j^+$, e. g., subset $\mathcal{X}_1^+$ shown in Figure 6, contain only one class. We treat such cases using the OvA binarization technique [11, 13]. Here, we first add the minority subset $\mathcal{X}_j^-$ to its majority subset $\mathcal{X}_j^+$ again. Then, we re-label all minority classes c_i^- and their samples in $\mathcal{X}_j^-$ to one combined “negative” class, and the single majority class c_i^+ in $\mathcal{X}_j^+$ to a “positive” class. We then may use standard classification algorithms to train a binary classifier for this combined and re-labeled sample subset.

4.2 Predictive Modeling

Predictive modeling includes a *training* and an *application* phase. We first discuss how to train the classifiers $\mathcal{M}_j$ for the sample subsets $\mathcal{X}_j^\pm$ (4.2.1). We then describe how to classify new observations, i. e., new samples x_t (4.2.2), and show how to obtain a final recommendation list $\mathcal{R}_e$ (4.2.3).

4.2.1 *Create Ensembles $\mathcal{E}_j$*

We train an individual classifier $\mathcal{M}_j$ for each subset $\mathcal{X}_j^\pm$. Thereby, we are widely free in the choice of classification algorithms. This is because we ensure that our subsets $\mathcal{X}_j^\pm$ meet the requirements of standard algorithms, such as containing a minimum number of classes. One restriction is that an algorithm must be able to train *probabilistic* classifiers $\mathcal{M}_j$. Probabilistic means that a classifier predicts a list of classes with associated *confidence values* as probabilities how sure the classifier is with its predictions. In addition, the algorithms must be able to handle more than two classes, since most of the subsets $\mathcal{X}_j^\pm$ contain multiple classes.

We recommend using ensemble procedures as classification algorithms. The subsets $\mathcal{X}_j^\pm$ resulting after our preparation phase often contain many class labels, but a rather small number of training samples and different kinds of noise. Our previous study shows that Random Forest is useful to handle such data characteristics [18]. This is because Random Forest‘s integrated sampling method reduces the risk of overfitting despite the small number of training samples.

When training a classifier $\mathcal{M}_j$ for a product group j, we have to distinguish between two cases: (1) sample subsets that have been partitioned into majority and minority subsets $\mathcal{X}_j^+$ and $\mathcal{X}_j^-$, as well as (2) subsets $\mathcal{X}_j$ that have not been partitioned. In the first case, we train separate classifiers, i. e., base learners, for each of the two subsets $\mathcal{X}_j^+$ and $\mathcal{X}_j^-$. We denote $\mathcal{L}_j^+$ as a *majority base learner*, which is trained on a majority subset $\mathcal{X}_j^+$. Accordingly, we denote $\mathcal{L}_j^-$ as a *minority base learner* being trained on $\mathcal{X}_j^-$. Thus, the base learners $\mathcal{L}_j^+$ and $\mathcal{L}_j^-$ are independent of each other and respectively tailored to predict either majority classes c_i^+ or minority classes c_i^-. By separately treating majority and minority classes, we ensure that classification algorithms do not ignore underrepresented minority classes.

For the group $OM934$ shown in Figure 6, we train a majority base learner $\mathcal{L}_1^+$ on the subset $\mathcal{X}_1^+$, i. e., $\mathcal{L}_1^+$ is tailored to predict error code A. Furthermore, we use $\mathcal{X}_1^-$ to train a minority base learner $\mathcal{L}_1^-$, which is able to predict error codes B to D. The resulting classifier system has to be able to predict all classes of a product group, e. g., all error codes from A to D. Hence, we combine each pair of base learners $\mathcal{L}_j^+$ and $\mathcal{L}_j^-$ to an ensemble $\mathcal{E}_j$. We store this ensemble $\mathcal{E}_j$ in a *model repository* and tag it with the number j, so that it is associated with product group j. Furthermore, we tag $\mathcal{E}_j$ with the hierarchy level l of the sample subsets $\mathcal{X}_{(l,j)}^\pm$ to indicate whether $\mathcal{E}_j$ is a primary or a surrogate ensemble. In case $l = max_d$, it is a primary ensemble and we denote it as $\mathcal{E}_{(max_d,j)}$. If $l = max_d - u$, with $u \geq 1$, we denote it as a surrogate ensemble $\mathcal{E}_{(max_d-u,j)}$. Other metadata stored for ensembles in the model repository are, e. g., the number of class labels and the Gini index of sample subsets $\mathcal{X}_{(l,j)}^\pm$.

In the second case, i. e., for sample subsets $\mathcal{X}_j$ that have not been partitioned into majority or minority subsets, we train one base learner $\mathcal{L}_j$ on the whole set $\mathcal{X}_j$. For instance, we train one base learner for group $OM470$ shown in Fig-

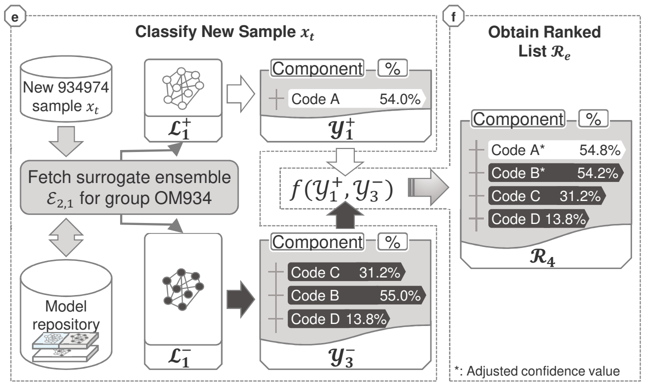

Figure 8: Computation of a recommendation list $\mathcal{R}_4$ for a new sample x_t of the product group 934974. Base learners $\mathcal{L}_1^+$ and $\mathcal{L}_1^-$ issue lists $\mathcal{Y}_1^+$ and $\mathcal{Y}_3^-$ that are merged by the combination function $f(\cdot)$ and ranked according to the confidence values to get $\mathcal{R}_4$.

ure 6 that is able to predict all four error codes A, E, F, and G. We also store such a base learner $\mathcal{L}_j$ as an ensemble $\mathcal{E}_j$ in our model repository.

4.2.2 Classify New Samples x_t

Given a new observation, we first determine the product group j at the lowest hierarchy level max_d to which this observation belongs. Then, we fetch the ensemble $\mathcal{E}_j$ from the model repository. This ensemble is also tagged with its hierarchy level l, indicating whether it is a primary ensemble $\mathcal{E}_{(max_d,j)}$ or surrogate ensemble $\mathcal{E}_{(max_d-u,j)}$ for the group j.

We then pass the sample x_t to the base learner(s) of the fetched ensemble $\mathcal{E}_j$. This way, we obtain a prediction for each of the base learners in the form of a list $\mathcal{Y}_e$ with the length e. $\mathcal{Y}_e$ contains the e most likely classes and ranks them according to their confidence values. In case $\mathcal{E}_j$ is composed of both a majority base learner $\mathcal{L}_j^+$ and a minority base learner $\mathcal{L}_j^-$, we accordingly get two lists $\mathcal{Y}_e^+$ and $\mathcal{Y}_e^-$. List $\mathcal{Y}_e^+$ contains the predicted majority classes c_i^+, and $\mathcal{Y}_e^-$ the predicted minority classes c_i^-. In case $\mathcal{E}_j$ has exactly one base learner $\mathcal{L}_j$, we obtain only one list $\mathcal{Y}_e$.

Figure 8 shows the application phase of our classification system, i. e., the steps (e) and (f) shown in Figure 5 for an example of a new failed EoL test x_t. The underlying product is a 934974 model at level $max_d = 3$ of our hierarchy. Since no primary ensemble for 934974 is available in the model repository at level 3, we fetch the surrogate ensemble $OM934$ at level 2 ($\mathcal{E}_{2,1}$). We then pass the new sample x_t as input to both base learners $\mathcal{L}_1^+$ and $\mathcal{L}_1^-$. For $\mathcal{L}_1^+$, we obtain the list $\mathcal{Y}_1^+$ with one majority error code A and a confidence value of 54.0%. The list $\mathcal{Y}_3^-$ is the prediction of $\mathcal{L}_1^-$ with the three most likely minority error codes C, B, and D, whose confidence values range between 13.8% and 55.0%.

4.2.3 Obtain Ranked List $\mathcal{R}_e$

In case of an ensemble that separately treats majority and minority classes, we need to combine both lists $\mathcal{Y}_e^+$ and $\mathcal{Y}_e^-$ into one list $\mathcal{R}_e$. Several reviews discuss numerous approaches to combine votes from different base learners [13,29,32,39,44]. However, all related approaches assume that the base learners predict completely or partly the same set of classes, i. e., the intersection of the sets of predicted classes is not empty. In our approach, the two involved base learners however predict disjoint sets of majority c_i^+ and minority classes c_i^-. Thus, our requirements differ from the approaches discussed in literature, so that these combination approaches are not applicable to our ensembles. Therefore, we consider the combination of disjoint classes from different base learners as an open research topic. For this reason, we opt for a first naive approach that is easy to implement. In most cases, the final recommendation list $\mathcal{R}_e$ is a union of the two lists $\mathcal{Y}_e^+$ and $\mathcal{Y}_e^-$ with unchanged confidence values.

Nevertheless, we consider some special cases where we scale a few of the original confidence values for certain classes. In our use case, a minority class c_i^- within $\mathcal{Y}_e^-$ sometimes has only a marginally higher confidence value than a majority class c_i^+ within $\mathcal{Y}_e^+$. Figure 8 shows such an example for the error codes A and B. Our interpretation is that the whole system is indifferent between the involved minority and majority classes. However, we typically assume a majority class to have higher confidence in probability values. This is because samples of majority classes occur much more often within the original sample set $\mathcal{X}$. Thus, we place the majority class above the minority class in the list $\mathcal{R}_e$. So, we upscale the confidence value of the relevant majority class and downscale the value of the minority class.

We consider only those cases where the difference in the confidence values of relevant classes is less than 1.5%-points. We use this low threshold, because we want to adjust the original confidence values as little as possible. This ensures that we do not distort the essential probabilistic statements of the base learners. In Figure 8, this only applies to the error codes A and B from the lists $\mathcal{Y}_1^+$ and $\mathcal{Y}_3^-$. In addition, we only adjust the confidence values of classes whose values are greater than the *random probability* of the respective base learners. The random probability of a base learner is one divided by the number of classes this base learner is trained on. For example, the base learner $\mathcal{L}_1^+$ is trained on two classes, i. e., one "positive" class for error code A and one "negative" class for other error codes. So, the random probability for error code A is 50%. The other base learner $\mathcal{L}_1^-$ is trained on the three error codes, B, C, and D, so that the random probability for each error code is about 33%. Thus, we adjust only the values for the error codes A and B, because their confidence values are higher than their random probability. This way, we only scale the confidence values of the classes for which the base learner is more confident. This further reduces the influence of our scaling.

We again opt for a straightforward approach to upscale the confidence value of the majority class and downscale the confidence value of the minority class. For error code A, we increase the original confidence value of 54.0% by the threshold value of 1.5%, i. e., by a factor of 1.015. So, we get a scaled confidence value of about 54.8%. For the minority class B, we reduce the confidence value by the same factor of 1.015. So, we obtain an adjusted value of about 54.2%. As a result, the adjusted confidence value of error code A is greater than the one for B.

Finally, we rank the classes in descending order regarding the scaled confidence values to generate the final recommendation list $\mathcal{R}_e$. Then, we limit $\mathcal{R}_e$ to the length e that is appropriate for the given use case. In EoL testing, the list is finally offered to operators. Thereby, they try to replace the faulty components in the order as they are recommended in the list $\mathcal{R}_e$. It would overwhelm these operators if we offered them a list with, e. g., 20 elements. So, we limit the list to a reasonable number of elements, e. g., to at most 10.

In case an ensemble only uses one base learner and we thus obtain only one list $\mathcal{Y}_e$, we may use this list as the final output of the process shown in Figure 5.

5. EVALUATION

We have carried out an extensive evaluation of our classification system based on its application to the real-world data of our use case. In the following, we discuss its potential to mitigate the negative effects of the analytical challenges illustrated in Section 2 (5.1). Afterwards, we report the results of our experimental evaluation (5.2). Then, we illustrate the real business impact of the proposed approach and how it improves the process of EoL testing (5.3). Finally, we discuss issues regarding the generality of our approach and ways to automate its parameterization (5.4).

5.1 Effects on Analytical Challenges

Table 1 summarizes how the two essential steps of our classification system affect the analytical challenges. It depicts these effects on the challenges separately for (a) SPH and (b) the subsequent CPI. To underpin these discussions, Table 2 reports statistical metrics that exemplify the effects on the challenges. We have collected these metrics by applying our approach to the real-world data of our use case.

SPH splits the original sample set $\mathcal{X}$ into 26 subsets $\mathcal{X}_j$. These are 21 primary subsets on level 3 of our product hierarchy, and five surrogate subsets one level higher. Each subset $\mathcal{X}_j$ contains on average 54 samples and eleven classes. This reduces the mean number of samples per class from about 12 samples in the original sample set $\mathcal{X}$ to now about 5 in each subset $\mathcal{X}_j$. Thus, we reduce the already small amount of representative samples in each subset (C1). This usually increases the risk of overfitting when applying learning algorithms [16,18,25]. Thus, we rate the effect of SPH on C1 as negative. As discussed in Section 4.2.1, this negative effect may though be mitigated in the later training phase by applying ensemble techniques for classification that are able to deal with smaller data sets, e. g., Random Forest [7, 18]. In general, many methods exist that can cope with challenge C1. However, only little is known how to tackle especially the combined effects of challenges C2 and C3 (cf. Section 3). This is the reason why our approach admits negative effects on C1 to mitigate the effects on C2 and C3.

SPH reduces the Gini index from 55% in the original sample set $\mathcal{X}$ to an average of 28% among all sample subsets $\mathcal{X}_j$. This already constitutes a significant reduction of the class imbalance. A detailed analysis reveals that SPH primarily reduces the class imbalance of the 21 primary subsets to this significant degree. The five surrogate subsets have Gini indexes between 30% and 50%. This is less than the Gini index of 55% of the original sample set $\mathcal{X}$. However, a Gini index between 30% and 50% for the five surrogate subsets still represents a remarkable class imbalance. Hence, we rate the influence of SPH on challenge C2 as partly positive.

The main advantage of SPH is apparent in its effect on challenge C3. SPH reduces the heterogeneity in the sensor signals s_k, since each subset $\mathcal{X}_j$ contains technically similar product variants. Firstly, we reduce the number of sensor signals s_k from 72 in the original sample set $\mathcal{X}$ to an average of about 41 in the subsets $\mathcal{X}_j$. Thereby, we remove

Table 1: Effect of SPH and CPI on analytical challenges: ++ Positive; + partly positive; 0 none significant; − partly negative; −− negative effect.

	Meets Analytical Challenge				
	C1	C2	C3.1	C3.2	C3.3
(a) SPH	−−	+	++	++	+
(b) CPI	−	+	0	0	0

Table 2: Numbers of samples x_t, classes c_i, sensor signals s_k, the portion of missing values ("NA") in these s_k, and the Gini index for the original sample set $\mathcal{X}$, as well as for the subsets $\mathcal{X}_j$ and $\mathcal{X}_j^{\pm}$ after (a) SPH and (b) the subsequent CPI.

Dimension	Original $\mathcal{X}$	(a) SPH $\varnothing$ of $26\mathcal{X}_j$	(b) SPH $\wedge$ CPI $\varnothing$ of $14\mathcal{X}_j \wedge 12\mathcal{X}_j^{\pm}$
Samples x_t	1050	54	37
Classes c_i	84	11	7
Sensor signals s_k	72	41	41
"NA" in s_k	17%	5%	5%
Gini index	55%	28%	21%

those sensor signals from a subset $\mathcal{X}_j$ that are not measured for the product variants of group j. For example, reconsider the four-cylinder variants in group $OM934$, where the sample set $\mathcal{X}$ originally contained sensor signals for six cylinders (cf. Section 2.2). SPH removes the sensor signals for the two cylinders that are not part of the four-cylinder variants in group $OM934$. A detailed analysis shows that this significantly reduces the number of missing sensor values. In our use case, about 17% of the values in the original sample set $\mathcal{X}$ were missing values, i. e., "NA"-values. SPH reduces this to an average of about 5% in the subsets $\mathcal{X}_j$. Thus, we rate the effect of SPH on challenge C3.1 as positive.

Furthermore, we reduce the variety in the value ranges of sensor signals within a subset $\mathcal{X}_j$ (C3.2). This reduces the number of sub-concepts each classifier $\mathcal{M}_j$ has to learn. For example, reconsider the concepts and sub-concepts shown for groups $OM934$ and $OM936$ in Figure 3. Originally, a combined classifier being applicable to both product groups had to learn all four concepts A, B, as well as A' and B'. Especially sub-concept B' was only represented by three minority samples of all 60 samples. Hence, learning algorithms would usually neglect these three minority samples and thus the whole sub-concept B'. After SPH, we train two separate classifiers on two separate sample subsets for the groups $OM934$ and $OM936$. The subset for group $OM936$ only contains the 47 samples to characterize concepts A and B. The subset for group $OM934$ only contains the 13 samples for sub-concepts A' and B'. So, the relative share of the three minority samples for sub-concept B' increases within this subset for group $OM934$. Thus, it is more likely that learning algorithms do not neglect the three minority samples and are thus able to learn a pattern for sub-concept B'. So, we rate the effect of SPH on challenge C3.2 as positive.

In a similar way, the subsets $\mathcal{X}_j$ do not contain error codes that are irrelevant for the associated product groups j (C3.3). For instance, the subset of group $OM934$ of four-cylinder variants does not contain class labels for cylinders No.5 and No.6 anymore. So, SPH ensures consistent predictions and thus increases user acceptance. However, SPH may also eliminate a few error codes in the sample subsets

$\mathcal{X}_j$ that are actually relevant for the product group j. In particular, we have at least one error code in each subset $\mathcal{X}_j$ that only occurs in one single sample in this subset. As discussed in Section 4.1, we remove such error codes and their single samples. SPH hence overreaches itself in solving the class membership problem, i. e., it removes few relevant classes from a sample subset $\mathcal{X}_j$. So, we rate the effect of SPH on challenge C3.3 as partly positive.

12 of the 26 sample subsets $\mathcal{X}_j$ resulting after SPH have a Gini index higher than 30%. The subsequent CPI hence partitions these 12 subsets into majority subsets $\mathcal{X}_j^+$ and minority subsets $\mathcal{X}_j^-$. Afterwards, all sample subsets $\mathcal{X}_j^\pm$ contain on average 37 samples and 7 classes, i. e., we again have about 5 samples per class as before the CPI. Note that Table 2 reports average numbers among both the 14 subsets $\mathcal{X}_j$ that have not been partitioned and the 12 majority and 12 minority subsets. A more detailed analysis reveals that the 12 majority subsets $\mathcal{X}_j^+$ contain on average 54 samples and 4 classes. The average values for the minority subsets $\mathcal{X}_j^-$ are however 44 samples and 12 classes. For these minority subsets $\mathcal{X}_j^-$, the number of representative samples for each class gets further reduced. Nevertheless note that this only affects 12 of the 26 previous subsets $\mathcal{X}_j$. So, we rate the effect of CPI on challenge C1 as partly negative.

In addition, CPI further reduces the average value of the Gini index for all resulting sample subsets $\mathcal{X}_j^\pm$ to about 21%. This additional reduction demonstrates a positive effect on challenge C2 of a class imbalance. Nevertheless, the reduction from 28% after SPH to now 21% is rather moderate, since CPI only partitions 12 of the 26 subsets $\mathcal{X}_j$. So, we rate this effect on C2 as partly positive. Note that CPI does not affect the number and nature of sensor signals. Hence, there is no effect on challenges C3.1 to C3.3.

5.2 Experimental Evaluation

Now we report the evaluation results for our classification system. We describe the experimental set-up (5.2.1), discuss how our classification system increases classification accuracy (5.2.2) and to what extend it reduces the number of ineffective rework attempts in EoL testing (5.2.3).

5.2.1 Experimental Set-Up

We use R 3.5.0 as a development environment for data pre-processing and for training the classifiers $\mathcal{M}_j$ [31]. For details about the hardware and software set-up, we refer to our previous study [18]. Here, we focus on a description of methodological aspects regarding the evaluation.

Training and test set: We split the sample set $\mathcal{X}$ into a training set with 750 samples and a test set with 300 samples. We have made sure that both sets contain all 84 classes and resemble the class distribution of the overall 1050 samples. We apply the training set preparation, i. e., steps a, b and c in Figure 5, on the 750 samples of the training set to get the corresponding sample subsets $\mathcal{X}_j^\pm$. Afterwards, we apply the training phase (step d) for each subset $\mathcal{X}_j^\pm$ to create the respective ensembles $\mathcal{E}_j$. We then carry out the application phase, i. e., steps e and f, for each of the 300 samples in our test set to evaluate the ensembles $\mathcal{E}_j$.

Parameterization: We parameterize our classification system as described throughout Section 4. For instance, we use a 0.8-quantile to split a sample subset into majority and minority subsets. We then apply Random Forest on each subset $\mathcal{X}_j^\pm$ using the same hyper-parameters as described in our previous study [18]. This allows us to compare the effects of our new approach with the previous results.

Evaluation: To evaluate the prediction performance, we report two *performance scores*. The first score represents accuracies for lists $\mathcal{R}_e$ of different lengths e. We classify the test samples to obtain a list $\mathcal{R}_e$ for each sample. A correct classification means that the real class label y_t of a test sample is contained at any of the first e positions of the associated list $\mathcal{R}_e$. Accuracy at e ($A@e$) then measures the relative portion of such correct classifications among all test samples. Thus, we have a dedicated accuracy value $A@e$ for each length e of the list $\mathcal{R}_e$. In our use case of EoL testing, operators may work through the list $\mathcal{R}_e$, i. e., they try to repair the faulty components in the order as they are listed in $\mathcal{R}_e$. Hence, $A@e$ measures how likely it is that an operator can solve a quality issue by solely using the list $\mathcal{R}_e$, i. e., without consulting the quality engineer. Note that a large list $\mathcal{R}_e$ would usually overwhelm operators. Hence, we limit the list $\mathcal{R}_e$ to ten error codes, i. e., $e \in \{1, \ldots, 10\}$.

The second score represents the number of *rework attempts* (RA) that operators need on average to solve a quality issue by working solely through the list $\mathcal{R}_e$. To calculate this score, we individually consider the number of correctly predicted faulty components for each position in the list $\mathcal{R}_e$. A hit at the first position means that the operator is able to solve the quality issue after one rework attempt. A hit at the second position means that s/he needs two attempts and so on. So, we respectively multiply the number of hits at a position with the ranking number. We then sum up the products and divide it by the number of all hits in $\mathcal{R}_e$.

Baseline: We compare the results of our proposed approach with the results of the best approach from our previous study [18]. This best previous approach is applying Random Forest in combination with the feature selection technique Boruta [25] (RF+B). In addition to our previous study, we experimentally evaluated a further baseline approach from the area of sequential data analysis. Here, we opt for an ensemble of several neural networks, since ensembles usually perform better than single classifiers [2]. Each neural network is trained using different random seeds. For a new observation, the ensemble averages the prediction scores of individual neural networks to obtain the final class prediction. So, we refer to this baseline as *averaged Neural Network* (avNN). Figure 9 compares the results of our approach with those of the baselines RF+B and avNN.

5.2.2 Increased Accuracy

In this subsection, we discuss the score $A@e$ on the left y-axis of Figure 9. A comparison of the two baselines shows that RF+B has a higher $A@e$ than avNN for all lengths of the list $\mathcal{R}_e$. The major reason is that sequential data analysis techniques such as neural networks usually require time series data with a high resolution. However, our sample set $\mathcal{X}$ contains only one static value for each sensor signal that is aggregated over time. Here, we see that conventional classification techniques such as RF achieve higher accuracies. Hence, we compare our approach only with RF+B.

Our approach dominates the baseline RF+B for all $A@e$ scores for different lengths e of the list $\mathcal{R}_e$. This means that the list $\mathcal{R}_e$ of our approach contains the correct faulty component more frequently compared to the list generated by RF+B. Thus, operators are able to solve a quality issue more often by solely working through the list $\mathcal{R}_e$ of our approach.

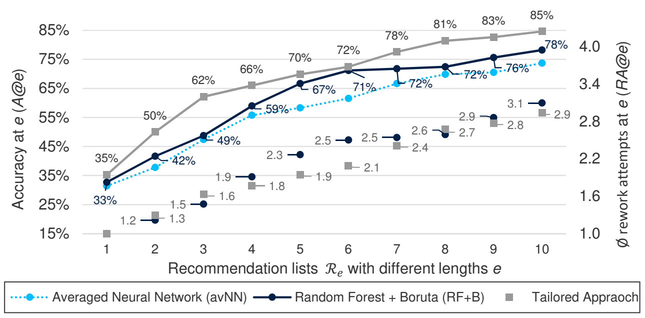

Figure 9: Evaluation results: $A@1$ to $A@10$ and $RA@1$ to $RA@10$ for lists $\mathcal{R}_e$, with different lengths e.

This also means that a quality engineer has to be consulted less often. The performance gains of our approach vary for individual lengths of the list $\mathcal{R}_e$. The lowest absolute gain is 1%-point for $\mathcal{R}_6$. Our approach especially outperforms the baseline for shorter lists. For instance, the highest performance gain is about 13%-points for $\mathcal{R}_3$. The average gain in accuracy among all lists is about 6.3%-points.

The scores $A@e$ reported in Figure 9 for our approach result from a combined application of SPH and CPI. Nevertheless, we have also evaluated both steps in isolation. This means that we divided our sample set $\mathcal{X}$ once only by SPH and once only by CPI. Then, we trained separate ensemble classifiers for each step and compared the results. We found that SPH has a higher contribution to increasing $A@e$ than CPI. The ten $A@e$ scores for applying only SPH exceed RF+B by an average of 2.9%-points. However, the scores for CPI are on average 2.7%-points below that baseline.

Nevertheless, applying CPI after SPH even adds 3.4%-points to the 2.9%-points accuracy gain of SPH, yielding the overall average of 6.3%-points mentioned above. This fact that CPI in isolation reduces accuracy, but increases it even further when applied after SPH supports our key statement. An approach focusing on either challenge C2 or challenge C3 in isolation is not sufficient. This especially holds for approaches that focus only on C2. Instead, it is much more beneficial to addresses both challenges at once, i. e., by applying SPH and CPI in a combined way.

5.2.3 *Reduced Number of Rework Attempts*

As mentioned before, we compare our approach with the best baseline RF+B, so we have excluded the $RA@e$ scores for avNN in Figure 9. While we want to increase $A@e$, now the goal is to reduce the scores $RA@e$, i. e., the average number of rework attempts operators need to solve a quality issue with the list $\mathcal{R}_e$. As shown on the right y-axis in Figure 9, our approach leads to a high reduction of the $RA@e$ scores for the lists with five and six elements. Here, operators need on average 0.4 less rework attempts. Nevertheless, the $A@e$ scores for our approach and the baseline are very close to each other for these lists $\mathcal{R}_5$ and $\mathcal{R}_6$. So, operators using our approach may solve the quality issue roughly as often as for the baseline, but they need a less number of rework attempts. The reason for this are the significant higher $A@e$ scores for lower lengths e, e. g., the performance gain of 13%-points for $A@3$. A high score $A@e$ on the first positions entails that the correct faulty component is contained more often at these first positions. Hence, it is more likely that operators solve a quality issue with fewer rework attempts.

For the lists with two, three, and eight elements, the $RA@2$, $RA@3$, $RA@8$ values of our approach are about 0.1 higher than those of the baseline RF+B. If the correct faulty component is part of the lists $\mathcal{R}_2$, $\mathcal{R}_3$, or $\mathcal{R}_8$, operators need on average only 0.1 additional rework attempts using the lists generated by our approach. Note again that our approach outperforms the baseline RF+B with a gain in $A@e$ between 9%-points and 13%-points with these three lists $\mathcal{R}_2$, $\mathcal{R}_3$, and $\mathcal{R}_8$. Here, the 0.1 additional rework attempts of the operator constitute a rather negligible price to pay, compared to this increase in accuracy. This is because a higher $A@e$ means that operators are much more likely to solve a quality issue without consulting the quality engineer.

5.3 Business Impact

The results reported for $A@e$ in Figure 9 with up to 85% are still less than typical results presented by the research community for other applications, e. g., see [27, 37, 42, 43]. Thereby, the research community usually employs synthetic or public data sets [3, 10] to design, optimize, and evaluate their approaches and algorithms. However, these data sets do not show all characteristics of real-world manufacturing data. In fact, real-world data characteristics and resulting analytical challenges make it hard to achieve similar levels of accuracy. We already show this in our previous study, where we tested a diverse set of methods with numerous configuration parameters [18]. The best combination of these methods with an accuracy of up to 78% is Random Forest with Boruta, which is used as baseline RF+B in this paper.

We also show in the previous study that even this baseline with its accuracy of up to 78% has a positive impact on the real business of EoL testing [18]. In particular, it reduces the overall costs for reworking on defective engines. More precisely, the personal costs for a quality engineer are usually 60% higher than for an operator. These quality engineers are now only involved in cases when the correct faulty component causing the quality issue was not part of the list $\mathcal{R}_e$. In all other cases, only the less expensive operators are involved in the process of EoL testing and reworking.

Compared to the baseline R+B, our approach leads to an even further reduction of the costs for reworking on defective engines. This is mainly because our approach results in higher $A@e$ scores for any list $\mathcal{R}_e$. The average gain in accuracy of 6.3%-points means that the operator can solve the quality issue without involving the more expensive quality engineer in 6.3%-points more cases. Furthermore, most of the lists $\mathcal{R}_e$ have a lower $RA@e$ score, so that our approach reduces the number of rework attempts an operator needs to solve a quality issue. The overall annual cost savings for a company such as Daimler Truck AG accumulates to a magnitude of up to a few millions of EURO. This shows that even small process improvements in the assembly of powertrain aggregates result in considerable cost savings.

5.4 Generality Issues and Parameterization

Our experiments show that our approach yields big performance gains for multi-class classification tasks that suffer from the two analytical challenges of multi-class imbalance (C2) and heterogeneous product portfolio and feature space (C3). In fact, these challenges are common in the whole manufacturing domain, which is mainly due to the increasing complexity and variety of production processes and products [20]. This statement is confirmed by review articles of Koeksal et al., Wüst et al., and Cheng et al. that examine literature describing several applications of machine learning to real-world manufacturing use cases [8, 23, 45].

To further confirm the generality of challenges C2 and C3, we have additionally examined several real-world use cases that are not covered by the mentioned reviews. For instance, Kassner et al. use machine learning, especially text analytics, to identify the root causes of quality problems related to customer warranty claims [22]. Thalmann et al. discuss analytical challenges for three use cases for fault detection, fault diagnosis, and predictive maintenance [38]. All these authors support our argumentation regarding analytical challenges in manufacturing. They all mention that a large number of possible error types in both production processes and product quality control leads to a multiplicity of imbalanced class labels for classification (C2). Furthermore, they discuss that machine learning suffers from the fact that underlying data often represent diverse product variants with different physical properties and technical specifications (C3).

The individual steps in the pipeline presented in Figure 5 are completely independent of the domain. Thus, the approach is generic. The only domain-specific aspect is the knowledge that is used as input for the first step. Here, our approach requires domain knowledge that allows for segmenting a sample set $\mathcal{X}$ into sample subsets that are more homogeneous regarding the feature space. In manufacturing, this domain knowledge is provided by documentations of product families, e. g., a product hierarchy as shown in Figure 2. As mentioned in Section 3, any manufacturing company has a documentation of its own product family [1]. So, SPH may be applied to machine learning applications across the whole manufacturing domain. In general, the domain knowledge that is necessary for SPH may be offered by a domain-specific concept model that organizes relevant domain concepts and entities into homogeneous groups. Such concept models are common in various domains, e. g., in chemistry or biology [34].

The next step CPI of our approach requires metrics to identify and quantify distinct imbalances between classes in sample sets. Here, we apply the Gini index and the p-quantile to the underlying class distributions of sample subsets $\mathcal{X}_j$. These two metrics are generic enough to be applicable to any kind of discrete class distribution. So, we may employ CPI regardless of the given use case or its domain.

SPH and especially CPI require to set or tune some parameters that influence classification performance (cf. Section 4.1). The first parameter of SPH a threshold for a *minimum number of samples* for a class. If a class has fewer samples than this threshold, SPH removes this class and its samples from the subset $\mathcal{X}_j$. The next parameter of SPH is another threshold to limit the *loss of information* due to the previous removal of some classes with too few samples. The first parameter of CPI is the *threshold of the Gini index* to identify a distinct class imbalance in a $\mathcal{X}_j$. CPI's second parameter is p of the p-quantile to differentiate minority and majority classes. For our use case of EoL testing, we have tuned these parameters and the resulting accuracy via a brute-force parameter search. However, a manual brute-force search is obviously too time-consuming when applying our approach to further use cases. To increase the generality by ease of use, we hence need an optimization algorithm that automates parameter tuning for SPH and CPI [12].

However, finding the best parameter setting for both SPH and CPI is a complex *multi-criteria* optimization problem. In fact, the individual parameters highly influence each other, so that it is hard to find parameter combinations that are close to the optimum. Furthermore, it is a *multi-objective* optimization problem. Besides increasing accuracy ($A@e$) and reducing the number of rework attempts ($RA@e$), the most important goal is to reduce monetary costs of EoL testing (Section 5.3). These different objectives usually compete with each other. For instance, a recent study shows that increasing accuracy does not necessarily reduce costs [19]. Altogether, these multi-criteria and multi-objective properties make optimization a very hard problem. An approach is needed that (1) finds a parameter setting that is close to the theoretical optimum of all competing objectives and that (2) achieves this with a minimum amount of computational effort. This calls for further research on and evaluation of optimization algorithms that are especially tailored to our approach and its two essential steps SPH and CPI.

A first good candidate for a tailored optimization method to be evaluated is the *stochastic gradient descent* [5]. This method is commonly used in machine learning, especially for backpropagation to train neural networks. We consider it especially suitable for our approach, because it reduces computational effort via faster optimization iterations, while it still finds a good parameter setting that is close to the optimum [6]. To further reduce computational effort of this optimization, we suggest to explore and apply heuristics to limit the search space for the parameters. The general idea is to heuristically identify proper initial parameter values, as well as lower and upper bounds within parameter spaces. For instance, initial values and bounds for p of the p-quantile may be obtained by analyzing the gradient of the density function of a class distribution (cf. red line in Figure 1).

6. CONCLUSION

The main contribution of this paper is an approach that exploits domain knowledge from a product hierarchy to mitigate negative effects of heterogeneous product and feature spaces to classification performance. Furthermore, we apply proper metrics to make an informed decision about the class distribution to address multi-class imbalance. Thereby, we show that related approaches focusing on only one of these two challenges are not sufficient. This statement is confirmed by our evaluation, where we applied our approach on real-world manufacturing data. Our approach dominates the baseline solutions in terms of accuracy. In most cases, we are also able to reduce the number of rework attempts an operator needs to solve a quality issue. In addition, we discussed generality issues, i. e., how our approach may be applied to increase performance in other use cases.

For ease of use, future work will concentrate on ways to automatically find proper parameterizations. Furthermore, extending the classification system to other use cases and domains will be interesting as well.

Acknowledgments. The authors thank the German Research Foundation (DFG), the Ministry of Science, Research and Arts of the State of Baden-Wurttemberg, and the Daimler Truck AG for financial support within the Graduate School of Excellence advanced Manaufacturing Engineering.

7. REFERENCES

[1] B. Agard and A. Kusiak. Data-Mining-based Methodology for the Design of Product Families. *International Journal of Production Research*, 42(15):2955–2969, 2004.

[2] M. A. H. Akhand and K. Murase. Neural Network Ensemble Training by Sequential Interaction. In J. Marques de Sá, editor, *Proceedings of the 17th International conference on Artificial Neural Networks - ICANN 2007*, Lecture Notes in Computer Science, pages 98–108. Springer, Porto, Portugal, 2007.

[3] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2011.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, NY, 8th edition, 2009.

[5] L. Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In Y. Lechevallier and G. Saporta, editors, *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT)*, pages 177–186, Paris, France, 2010. Springer-Verlag.

[6] L. Bottou and O. Bousquet. The Tradeoffs of Large Scale Learning. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*, pages 351–368. MIT Press, Cambridge, MA, USA.

[7] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[8] Y. Cheng, K. Chen, H. Sun, Y. Zhang, and F. Tao. Data and Knowledge Mining with Big Data towards Smart Production. *Journal of Industrial Information Integration*, 2017.

[9] F. Cowell. *Measuring Inequality*. 3th edition, 2000.

[10] D. Dua and C. Graff. UCI Machine Learning Repository, 2020.

[11] A. Fernández, V. López, M. Galar, M. J. del Jesus, and F. Herrera. Analysing the Classification of Imbalanced Data-Sets with Multiple Classes: Binarization Techniques and ad-hoc Approaches. *Knowledge-Based Systems*, 42:97–110, 2013.

[12] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter. Efficient and Robust Automated Machine Learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*, pages 2962–2970, Cambridge, MA, USA, 2015. MIT Press.

[13] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera. An Overview of Ensemble Methods for Binary Classifiers in Multi-Class Problems: Experimental Study on One-Vs-One and One-Vs-All Schemes. *Pattern Recognition*, 44(8):1761–1776, 2011.

[14] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.

[15] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing. Learning from Class-Imbalanced Data: Review of Methods and Applications. *Expert Systems with Applications*, 73:220–239, 2017.

[16] H. He and E. A. Garcia. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[17] V. Hirsch, P. Reimann, O. Kirn, and B. Mitschang. Analytical Approach to Support Fault Diagnosis and Quality Control in End-Of-Line Testing. *Procedia CIRP*, 72:1333–1338, 2018.

[18] V. Hirsch, P. Reimann, and B. Mitschang. Data-Driven Fault Diagnosis in End-of-Line Testing of Complex Products. In *Proceedings of the 6th IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 492–503. IEEE, 2019.

[19] V. Hirsch, P. Reimann, and B. Mitschang. Approach to Incorporate Cost Aspects into the Ordering of a Data-Driven Recommendation List for End-of-Line testing. *Procedia CIRP*, 74, 2020.

[20] S. Hu, X. Zhu, H. Wang, and Y. Koren. Product Variety and Manufacturing Complexity in Assembly Systems and Supply Chains. *CIRP Annals*, 57(1):45–48, 2008.

[21] R. Isermann. *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*. Springer-Verlag, Berlin, Heidelberg, 2006.

[22] L. Kassner and B. Mitschang. Exploring Text Classification for Messy Data: An Industry Use Case for Domain-Specific Analytics Technology. In E. Pitoura, S. Maabout, G. Koutrika, A. Marian, L. Tanca, I. Manolescu, and K. Stefanidis, editors, *Proceedings of the 19th International Conference on Extending Database Technology (EDBT)*, pages 491–502, Bordeaux, France, 2016.

[23] G. Köksal, İ. Batmaz, and M. C. Testik. A Review of Data Mining Applications for Quality Improvement in Manufacturing Industry. *Expert Systems with Applications*, 38(10):13448–13467, 2011.

[24] M. Kuhn. Building Predictive Models in R Using the caret Package. *Journal of Statistical Software*, 28(5), 2008.

[25] M. B. Kursa and W. R. Rudnicki. Feature Selection with the Boruta Package. *Journal of Statistical Software*, 36(11), 2010.

[26] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya. A Survey on Addressing High-Class Imbalance in Big Data. *Journal of Big Data*, 5(42), 2018.

[27] X.-Y. Liu, J. Wu, and Z.-H. Zhou. Exploratory Undersampling for Class-Imbalance Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2009.

[28] L. Nanni, A. Lumini, and S. Brahnam. A Classifier Ensemble Approach for the Missing Feature Problem. *Artificial Intelligence in Medicine*, 55(1):37–50, 2012.

[29] R. Polikar. Ensemble Based Systems in Decision Making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.

[30] R. Polikar, J. DePasquale, H. Syed Mohammed, G. Brown, and L. I. Kuncheva. Learn++.MF: A Random Subspace Approach for the Missing Feature Problem. *Pattern Recognition*, 43(11):3817–3832, 2010.

[31] R Core Team. R: A Language and Environment for Statistical Computing, 2018.

[32] L. Rokach. Ensemble-Based Classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.

[33] C. N. Silla and A. A. Freitas. A Survey of Hierarchical Classification Across Different Application Domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2011.

[34] R. Stevens, C. Wroe, P. Lord, and C. Goble. Ontologies in Bioinformatics. In *Handbook on Ontologies*, pages 635–657. Springer-Verlag, Berlin Heidelberg, 2004.

[35] C. Sun, N. Rampalli, F. Yang, and A. Doan. Chimera: Large-Scale Classification Using Machine Learning, Rules, and Crowdsourcing. *PVLDB*, 7(13):1529–1540, 2014.

[36] Y. Sun, A. K. C. Wong, and M. S. Kamel. Classification of Imbalanced Data: A Review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719, 2009.

[37] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, and Y. Zhou. A Novel Ensemble Method for Classifying Imbalanced Data. *Pattern Recognition*, 48(5):1623–1637, 2015.

[38] S. Thalmann, H. G. Gursch, J. Suschnigg, M. Gashi, H. Ennsbrunner, A. K. Fuchs, T. Schreck, B. Mutlu, J. Mangler, G. Kappl, C. Huemer, and S. Lindstaedt. Cognitive Decision Support for Industrial Product Life Cycles: A Position Paper. In *Proceedings of the 11th International Conference on Advanced Cognitive Technologies and Applications (COGNITIVE)*, Venice, Italy, 2019. IARIA.

[39] M. van Erp, L. Vuurpijl, and L. Schomaker. An Overview and Comparison of Voting Methods for Pattern Recognition. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*, pages 195–200, Los Alamitos, Calif, 2002. IEEE Computer Society.

[40] S. Verron, J. Li, and T. Tiplica. Fault Detection and Isolation of Faults in a Multivariate Process with Bayesian Network. *Journal of Process Control*, 20(8):902–911, 2010.

[41] S. Wang. *Ensemble Diversity for Class Imbalance Learning*. 2011.

[42] S. Wang, L. L. Minku, and X. Yao. A Systematic Study of Online Class Imbalance Learning With Concept Drift. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4802–4821, 2018.

[43] S. Wang and X. Yao. Multiclass Imbalance Problems: Analysis and Potential Solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1119–1130, 2012.

[44] M. Woźniak, M. Graña, and E. Corchado. A Survey of Multiple Classifier Systems as Hybrid Systems. *Information Fusion*, 16:3–17, 2014.

[45] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben. Machine Learning in Manufacturing: Advantages, Challenges, and Applications. *Production & Manufacturing Research*, 4(1):23–45, 2016.

[46] Z.-H. Zhou and X.-Y. Liu. On Multi-class Cost-sensitive Learning. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 567–572, Boston, Massachusetts, 2006. AAAI Press.