Institut für Parallele und Verteilte Systeme
Abteilung Anwendersoftware
Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart

Masterarbeit

Partitioning Training Data for Complex Multi-class Problems using Constraint-based Clustering

Ulf Kunze

Course of Study:	Softwareenginering
Examiner:	Apl. Prof. Dr. rer. nat. habil. Holger Schwarz
Supervisor:	M. Sc. Dennis Tschechlov
Commenced:	July 12, 2021
Completed:	January 12, 2022

Kurzfassung

Die Qualitätskontrolle ist eines der wichtigsten Instrumente zum Schutz der Verbraucher vor Produkten minderer Qualität und daher unerlässlich. Aber es ist nicht nur wichtig, fehlerhafte und minderwertige Produkte durch die Qualitätskontrolle vom Markt fernzuhalten, sondern sie nach Möglichkeit zu reparieren. Dies spart unnötigen Abfall und ist ein nachhaltiger Umgang mit Ressourcen. Mit dieser Arbeit werden Constraint-basierte clustering Algorithmen im Anwendungsfall der Qualitätskontrolle untersucht. Constraint-basierte Clustering Algorithmen werden eingesetzt, weil sie mehr Flexibilität versprechen als starre Partitionen. Die Neuzuordnung von Dateninstanzen zu neuen Clustern kann dazu beitragen, die Auswirkungen der folgenden analytischen Herausforderungen zu verringern: heterogenes Produktportfolio, Multi-class imbalance und Small Sample size. Für diese Arbeit werden die Algorithmen CDBSCAN, COP-Kmeans und MPCK-Means evaluiert. Die verwendeten Constraint-Sets sind Constraints nach Produktgruppen, Motortypen und Fehlerklassen. In dieser Arbeit wird auch die bestehende Methode genauer untersucht, um die unterschiedlichen Verhaltensweisen zu verstehen. Das Endergebnis ist eine durchschnittliche Verbesserung von 5% gegenüber dem bestehenden Ansatz und eine Steigerung von 13% gegenüber einem Random Forest Klassifikator. Zusätzlich werden Verfahren untersucht um Domänenwissen aus Datensätzen zu extrahieren. Dafür wird ein Active Learning und ein algorithmischer Ansatz in die bestehende Pipeline integriert.

Abstract

Quality control is one of the most important tools for protecting consumers of low quality products and is therefore essential. But it is not only important to keep defective and substandard products off the market through quality control, but to repair them whenever possible. This saves unnecessary waste and is a sustainable use of resources. In this thesis, constraint-based clustering algorithms are evaluated in the use case of quality control. Constraint-based clustering algorithms are used because they promise more flexibility than rigid partitions. The reallocation of data instance into new clusters can help to reduce the influence of analytic challenges for example: heterogeneous product portfolio, Multi-class imbalance and small sample size. For this thesis the algorithms CDBSCAN, COP-Kmeans and MPCK-Means are evaluated. The used constraint sets are Constraints by: Product group, engine type and error classes. This work also examines the existing method in more detail to understand the different behaviours. The end result is an average improvement of 5% over the existing approach and an increase of 13% over a random forest classifier. Furthermore, methods for extracting domain knowledge from data sets are investigated. For this purpose, an active learning and an algorithmic approach are integrated into the existing pipeline.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Quality control is one of the most important tools for protecting consumers of bad products and is therefore essential. But it is not only important to keep defective and substandard products off the market, but also to repair them where possible. To avoid unnecessary waste and act more sustainably for our future. Furthermore, repairing products that have become faulty or broken during production turns a company's loss into a profit and should therefore always be considered by manufacturers. It is important that the manufacturer has access to tools that enable them to repair at a reasonable cost. However, these tools need to cope with the analytical challenges of these real-world data sets. In the case of Hirsch et al.[HRM20] these challenges are as follows: Multi-class imbalance, heterogeneous product portfolio, small sample size and missing features in a data set of different engines. Engines are complex products that use multiple components and therefore have different error classes in different quantities, leading to multi-class imbalance. This heterogeneity of product classes also leads to missing features, as not every engine has the same components and therefore sensor data is missing for some engines and available for others. In addition, the sample size is small with 84 error classes, 28 product groups and only 1050 data instances. These challenges are well researched in their own right and are often solved by classification, but combining these analytic challenges is not straightforward. However, classification can be improved by using an ensemble that focuses on the instances relevant to the current prediction. These approaches require selection or partitioning of the relevant instances to work well. Therefore, this work focuses on the evaluation of constraint-based clustering algorithms for partitioning such relevant instances.

Prior to this thesis, research was conducted at the University of Stuttgart on the topic of end-of-line product testing. This research was mainly carried out by IPVS researchers and Vitali Hirsch was significantly involved. During his research, he developed an approach to predict faulty parts in an engine. Normally, quality engineers have to judge which part of an engine is defective based on their experience and knowledge. With his approach, Hirsch et al. [HRM19] where able to predict the defective part already on the second rework attempt, whereas quality engineers usually need about four rework attempts. His approach is based on a real world data set provided by a German car manufacturer. The data set was collected during end-of-line testing of truck engines and contains a product hierarchy, information about the different engine groups and the error class designation.

While Hirsch et al. use a specific approach based on the product hierarchy, it is an open question how generic approaches perform in this scenario. Therefore, three different constraint-based clustering algorithms are evaluated. The evaluated algorithms are: CDBSCAN, COP-Kmeans and MPCK-Means. Constraint-based clustering also has the advantage that it can produce results when little or no prior domain knowledge is available. However, the use of existing domain knowledge allows for improvements over regular clustering algorithms that would normally struggle with the challenges of the provided dataset. In addition, constraints enable the creation of new partitions so that new possible partitions can be evaluated using existing domain knowledge. This is done by

generating constraints from: Product groups, different engine types and the existing error classes, which are used to train the classifier. Furthermore, an attempt is made to show the behaviour of the implemented and existing algorithms with respect to the analytical challenges.

1.1 Goals

The goals of this master thesis are:

- The development and investigation of constraint-based clustering techniques as an tool to create partitions for a classification based prediction.
- If no prior domain knowledge is available, a set of constraints can be derived from the existing error classes.
- Constraint sets in cases where domain knowledge is present in the form of a product hierarchy:
  - A broad constraint set would be based on the different types of engines
  - An exact group assignment can be achieved with constraints derived from product groups.
- Through implicit domain knowledge, constraints can also be generated by domain experts with the help of active learning.
- Evaluate the behaviour of the approaches on different degrees of imbalance.
- Evaluation of active learning techniques as an interface for domain experts.
- Implementation of a partitioning based on missing features.
- Ensure compatibility with new partitions for all approaches.
- Evaluate the behaviour of constraint-based clustering on a real-world data set with multi-class imbalance, small sample size and missing features.

1.2 Outline

This thesis is separated in the following chapters:

Chapter 2- Previous Research In this chapter, the approach of Hirsch et al. is described in detail. The data set used is presented with its analytical challenges. This is followed by the SPH + CPI partitioning method to reduce the imbalance between classes and overcome some of the challenges mentioned earlier. Ensemble classification and other classification methods are then described further.

Chapter 3 - Related Work This chapter starts with a broad overview on the topics of real-world Applications for Data Analysis. Then goes into greater detail about the analytic challenges of multi-class imbalance, classification with a data set with small sample sizes and the challenges of a heterogeneous product portfolio. Finally the chapter ends with a classification of this work.

Chapter 4 - Constraint-based clustering In this chapter the basics of constraint-based clustering are explained, starting with the constraints used in this thesis. Then a introduction into the three used constraint-based clustering algorithms is given starting with COP-Kmeans, followed by MPCK-Means and CDBSCAN. At the end of the chapter, active learning is introduced.

Chapter 5 - Implementation This chapter gives an overview of the integration of the various new methods into the existing framework. It starts with the implementation of active learning, partitioning by missing features and some algorithms that change the existing product hierarchy in the domain knowledge extraction. This is followed by the preparation of the training set, which includes the new constraint sets, the constraint-based clustering algorithms and the Random Forest approach to create new partitions for the ensemble.

Chapter 6 - Evaluation This chapter evaluates all new implementations and provides insight into the behaviour of the algorithms, it starts with an explanation of the metrics used for the evaluation. More background information is then given on the changes to the data set and data generation, with a focus on how values that are not numbers are handled. The results of constraint-based clustering and different constraint sets are then assessed. In addition, the different approaches are evaluated with varying degrees of multi-class imbalances. The chapter ends with an evaluation of domain knowledge extraction for cases where little domain knowledge is initially available.

Chapter 7 - Conclusion The last chapter then draws a conclusion over the changes to the approaches and suggests further research in the outlook.

2 Previous research

This chapter provides a detailed look at the approaches of Hirsch et al. [HRM19] and the follow-up paper [HRM20], both deal with fault isolation on the same data set. In the first paper, different standard machine learning algorithms are compared with respect to their efficiency on the data set. The second paper presents a self-developed approach based on the challenges of the data set. This self-developed approach focuses on two of the four analytical challenges of the data set: dealing with the multi-class imbalance and the heterogeneous product portfolio. A flowchart describing the steps of the approach can be seen in the Figure 2.1. The steps are explained in more detail in the following sections, starting with the data set and the analytical challenges in Section 2.1. Afterwards the SPH + CPI algorithm, which is the main approach to training set preparation, is explained in Section 2.2, The chapter then ends with an overview of the Hirsch et al. pipelines prediction step in Section 2.3.

2.1 Data set

The data set used by Hirsch et al. [HRM19] contains 1050 instances, with 84 error classes. It also contains a product hierarchy, seen in Figure 2.2, that divides each engine group as follows: The first level distinguishes between medium-heavy and heavy engines. The second level distinguishes between the different engine types, e.g. 4 or 6 cylinders. On the third level, the actual models are differentiated according to their characteristics. This last level contains 29 different engine groups. Hirsch et al. identify a number of analytical challenges related to the data set, namely: (C1) small size of the sample set provided, (C2) Multi-Class imbalance, (C3) heterogeneous product portfolio and (C4) noise and overlap of samples:

Small size of the sample set provided (C1): The 1050 instances have 115 features, which in combination with the 84 error classes can lead to overfitting. Furthermore, it can also increase the difficulty of learning decision rules for the minority error classes. As an example of overfitting, Hirsch et al. [HRM19] points out that a decision tree would generate many decision nodes due to the number of features and instances, and fail outside of the training data.

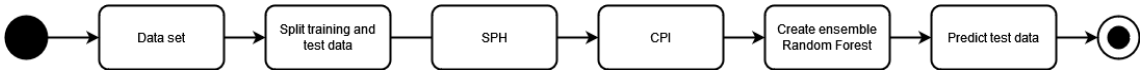


Figure 2.1: Simplyfied verison of the pipeline used by Hirsch et al. in [HRM20]

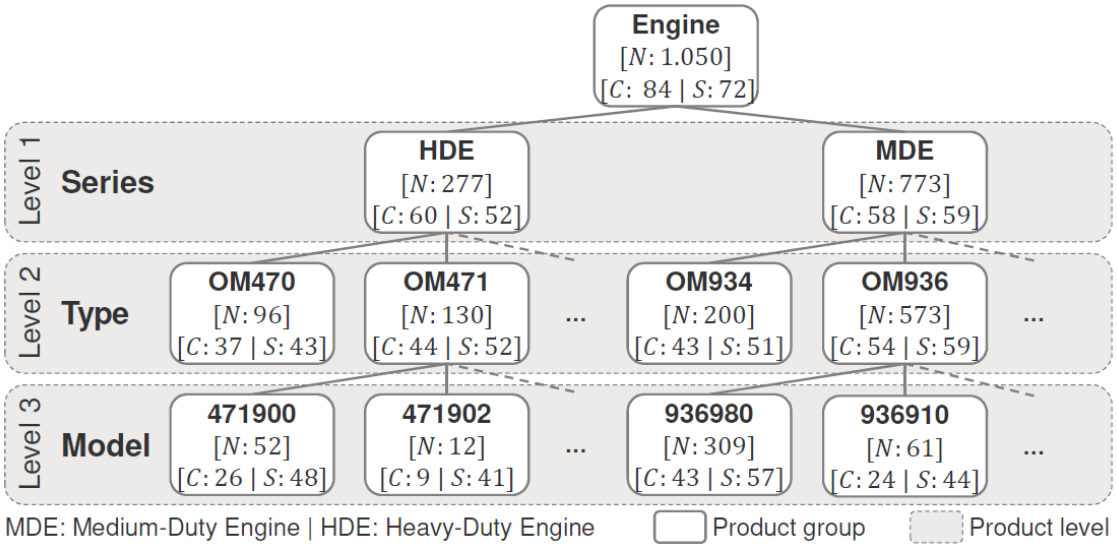


Figure 2.2: Representation of the data set, by Hirsch et al. in [HRM20]

Multi-Class imbalance (C2): The evaluation of Hirsch et al. [HRM19] observed that the five biggest error classes contain about 29% of all instances. The remaining 79 error classes contain 71% of the instances. Therefore, there exists a multi-class imbalance between the error classes and the distributions of the instances. Another degree of imbalance is the unbalanced distribution of the 1050 instances across the 29 engine groups. Most groups have between 20 and 60 instances, while the largest group contains about 300 instances. However, this imbalance does not directly contribute to the problem of multi-class imbalance.

Heterogeneous product portfolio (C3): The product diversity creates challenges, Hirsch et al. calls "heterogeneous product portfolio", that consists of the following three parts.

The first part of this challenge is **missing features**. An engine consists of many components and each of these components must be checked for errors. Since one engine may have more cylinders than another engine, values may be missing from the data set. One possibility is to replace these missing values, which can lead to inaccuracies in the evaluation. Another possibility is to delete the feature for all engines, but this would mean that errors in the additional cylinders would not be detected. Approximately 17% of the data set consists of Not a number (Nan) values.

The second part of this challenge is called **sub-concepts**. Since the same sensors are sometimes used for different engines for the quality tests, the problem arises that the same sensor can provide data in different value ranges for two engines. This means that the classification algorithm has to learn different patterns describing the respective engines. Since modularisation is also part of the automotive industry, individual components of an engine are also used in other engines. Furthermore, that even these same components can have different value ranges, even though they have the same error class. Hirsch et al. points out that this is called a sub-concept in literature.

The third and last part of this challenge is called **class membership**. Since not all product components are used in all engines, some error classes are not relevant for certain engines. This can lead to incorrect predictions about a faulty component, as it is simply not present in that engine.

Noise and overlapping samples (C4): In this analytic challenge, Hirsch et al. summarise human and sensor error. A human error would be, for example, the incorrect error class for a data instance. Outliers in the data set are also counted in this category of noise samples. Due to fluctuations and inaccuracies in the sensors, it can happen that data samples are in the value ranges of other engine groups, this is called overlapping of samples.

2.2 Training set Preparation

In this step, the training data is pre-processed to achieve a more accurate prediction result. Hirsch et al. have developed methods for this that specifically address the C2 and C3 challenges. The first method is Segmentation according to the Product Hierarchy (SPH), a grouping based on the 29 product groups. The second method is class partitioning according to imbalance (CPI), an attempt to reduce class imbalance within a group by dividing the majority error classes.

Segmentation according to Product Hierarchy (SPH): The SPH step uses the 29 engine groups as the basis for an ensemble. Likewise, the problem of **missing features** becomes less relevant, since the same features are missing within a group. This also reduces the impact of different value ranges on a sensor, so called **sub-concepts**. Focusing on one group also reduces the number of possible error classes and thus the problem of class membership. Although heterogeneity is likely to be lowest at the bottom level of the hierarchy, most groups are comparatively small in terms of sample size. This may mean that there are not enough samples to train a classifier. For this case, Hirsch et al. introduce surrogate samples. If a group has too few samples (information loss), only one sample or less than two error classes, the surrogate group is used in its place. The surrogate group is the parent node of the insufficient leaf node. Surrogate group are already a step towards dealing with the multi-class imbalance challenge (C2).

Class partitioning according to imbalance (CPI): The focus of the CPI step is on addressing the multi-class imbalance. The individual groups from the SPH step are analysed. In each group, with a Gini coefficient of greater than 0.3, the majority classes are separated from the minority classes. The majority class is given a subgroup of its own, while all minority classes are pooled together in one joint group. This partitioning is done by calculating the 0.8 quantile of the current partition as a threshold. Everything greater than the threshold is put into the majority partition and everything less or equal is put into the minority partition.

2.3 Prediction

This section shows which adjustments were made for the prediction. In general, a random forest classifier is trained for all predictions, but theoretically any other classifier can be used. The result of a prediction is the error class with the highest probability. In the following, we will show what this means for the SPH + CPI and Random Forest groups.

Ensemble: SPH + CPI modifies the 29 original product groups, some of which have subgroups to separate majority and minority. For each of these 29 groups and subgroups, a separate classifier is now trained. This further reduces heterogeneity (C3) as only engines from one product group are considered. In the application case, the classifier used is a Random Forest. When the minority and majority results are combined, the majority results are scaled upwards to fit the expectation of the imbalance. From the results of the combined minority and majority groups the best result is selected.

Random Forest: Hirsch te al. [HRM19] selected Random Forest based on its ability to cope with the four main analytic challenges listed in Section 2.1. For the prediction, the entire data set, without groups, partitioning or further processing is used. In this thesis, the Random Forest is used as a basis for comparing different methods of partitioning and data aggregation.

3 Related Work

In this chapter, an overview of related work by different authors is given. Therefore, the challenges of data analysis in possible industry scenarios are looked upon in Section 3.1 and the more specific challenges in the later sections. These include the methods used in research to deal with multi-class data imbalance, presented in Section 3.2, and the challenge of missing features in Section 3.3. Then the results of other researchers on the impact of small sample sizes in Section 3.4 are presented, followed by the heterogeneous product portfolio in Section 3.5. This chapter then ends with the classification of this thesis in Section 3.6.

3.1 Challenges of Data Analysis in real-World Applications

Manufacturers are required to ensure their products comply with certain standards in a vast majority of cases, be it EU directives, standards, or requirements of other companies that use their product as an intermediate. Quality assurance and end-of-line testing are therefore an absolute must for every manufacturer. However, each time a data set is analysed, different challenges arise.

Hesari et al. [ZASA18] uses a data set of diesel engines. The data itself comes from vibration sensors that check the final state of the engine. More specifically, the authors focus their diagnosis on misfires caused by faulty fuel injection. Although the neural network approach is of interest and the results are impressive, an engine consists of several parts and therefore has several sources of error. Unfortunately, these are ignored, so a defective gasket, or a badly adjusted ignition, can remain an undetected problem. Though Hesari et al.'s approach has merit, it should be extended to all components of an engine, or all parts of an engine function.

Leitner et al. [LLE16] work with a data set limited to a specific engine type. Since their data set contains about 1700 good engines and only about 80 faulty engines, they go the outlier detection route and try to identify faulty engines, thereby tackling a binary classification problem. This is commonly referred to as fault detection and determines whether the motor is faulty or not. The data set contains features from noise, vibration and harshness sensors generated by a one-minute engine run with an electric motor. An engineer has already set the thresholds for each of the 348 features. They uses a support vector data description and evaluates three different hyperparameters to find a good optimisation. However, it only determines whether a motor is defective, not why it is defective, which is a much more difficult problem.

Wüst et al. [WWIT16] present a comprehensive overview of machine learning in real-world applications. The main challenges are high dimensionality, imbalance and the selection of a machine learning technique that can deal with these challenges. Then, the different techniques of unsupervised machine learning, reinforcement learning and supervised machine learning are

explained in detail. Followed, by a quick overview of different supervised machine learning algorithms and their strengths. Finally, applications where machine learning is currently used are discussed, e.g. in manufacturing, medicine, image recognition and credit scoring.

Köksal et al. [KBT11] present an overview of data mining work from 1997 to 2007 in the context of quality improvement. They focus on so-called quality tasks, which are: Description of product and process quality, prediction of quality, classification quality and parameter optimisation. All data mining applications are evaluated against these quality tasks. Furthermore, programming languages, data mining packages and other software are classified for the purpose of data mining.

3.2 Multi-class imbalance

The problem of multi-class imbalance can occur in many different scenarios. The mere fact that products are manufactured in different quantities already leads to an imbalance of available data and error classes. Especially in end-of-line tests this is often the case. There are many differences between the individual engines, but also overlaps due to the use of the same components, which lead to an unbalanced distribution of data instances.

Krawczyk [Kra16] provides an overview of the different approaches and challenges in dealing with imbalanced data. He also provides suggested solutions and guidance for researchers to overcome these problems. Since the focus of this paper is on constraint-based clustering, the section of his paper that deals with clustering is of particular interest. Krawczyk sees problems with clustering methods that cluster data sets with an imbalance between different classes. **This particularly concerns methods that use centroids. Density-based clustering methods are more resistant to imbalanced data**. He also points out that no clustering method takes into account the overall distribution of data points, which is important for detecting minority classes. Krawczyk suggests developing a method that uses clustering as a basis for partitioning data. The partitions are then further analysed using different methods. He also suggests applying clustering only to the minority class to discover structures within minorities. An example of this would be to detect and consider the outlying areas around large clusters. The goal would be to find a rough partition in a first step and then determine the peripheral areas in detail.

Weiss et al. [Wei04] is pointing out that the multi-class imbalance is also effecting the accuracy score, since the majority classes have a higher influence on the accuracy than the minority classes. He further points out that minority classes usually have a error rate of two to three times higher than the majority classes.

Xuan et al. [XZF13] evaluates the K-Means clustering algorithm on class imbalanced data. They find that class imbalance leads to overall degradation when using K-Means. The performance of the clustering algorithm decreases as class imbalance increases. Although this is an indicator of how clustering algorithms can behave, only a partition-based clustering algorithm is evaluated, giving no attention to hierarchical and density-based clustering algorithms. Moreover, the performance of any clustering algorithm depends on the characteristics of a data set, and the degree of imbalance may not even be the biggest challenge. For example, density-based clustering often struggles with the variance of a data set's density in certain areas. In our case, constraint-based clustering is used to overcome these challenges. The constraint of cannot-link allows to preserve minorities in imbalanced data sets.

3.3 Missing Features

The provided data set has the same problem as many other real-world data sets: It contains missing values. However, usually missing values arise from one of the following three reasons: a defective sensor or incorrect use of the measurement equipment, a lost or forgotten value, or simply a case of human error can cause the missing value. Houari et al. [HBTK14] also show different methods to solve the problem of missing features:

- Mean imputation: replaces the missing value with a mean calculated over the whole sample.
- Regression imputation: uses regression to replace a missing value.
- K-nearest neighbours: uses k neighbours to create a replacement value.
- Multiple imputation: uses a predictive distribution to create replacement values.

Houari et al. propose a sampling procedure that works in two steps: First, they estimate with copulas the best imputation method for a data value. Second, they impute the missing values with the best method.

Ehrling et al. [EGV+18] provide a classification of different patterns of missing data and go further by also determining whether the missing features are random or in correlation with something. Depending on the patterns and correlations found, a decision is made whether to ignore or impute the missing values. An example with an industrial data set is then used to show the results of using different imputation methods in combination with their approach.

In the case of the present data set, the reason for missing values is the heterogeneity of the data and the fact that the data set contains different products with different sensors. Additionally a smaller percentage of the missing features is noise related to malfunctioning sensors, or human error.

3.4 Small Sample size

Small sample size presents a challenge because any classification needs data to work properly. He et al. [HG09] describes this as special challenge in combination with imbalance and high dimensionality, since it leads to inductive rules failing. Also, it is more likely that the decision trees degenerate because of the high dimensionality and sample size. In case of this thesis this can effect the decision trees of the Random Forest algorithms used, by overfitting. Weiss et al. [Wei04] points out further that the lack of samples is challenging for classification algorithms since it is difficult to find regularities.

However, it is possible to overcome the small samples by oversampling. Small sample size is further increasing the challenge when combined with sub-concepts.

3.5 Heterogeneous product portfolio

A heterogeneous product portfolio as described in Section 2.1 comes with its own challenges, namely **class membership**, **sub-concepts** and Hirsche et al.[HRM19] also classifies **missing features** as part of the Heterogeneous product portfolio. He et al. [HG09] further points out that **sub-concepts** might be ignored if deemed not statistically meaningful, by the classification algorithm. This is further impaired by the different value ranges for the different product groups, since the values can become contradictory. Class membership is creating more contradiction for simple classifiers since, not all error classes are relevant for every product group.

Suresh et al. [SGG18] uses a two-stage pipeline to generate predictions for patient groups. This approach is similar to that of Hirsch et al. [HRM20], but is optimised for their use case: predicting the mortality of patients in critical condition in the next 24 or 28 hours. Her pipeline creates a grouping based on similar features by creating an embedding with representation learning. It then uses cohort detection to split the embedding into different groups with similar properties to ensure that there is enough information to make a prediction. A cohort in this context is a group with the same properties, which is a similar concept to domain knowledge. Multitask learning is used to create a grouping for predictions. In doing so, Suresh et al. deal with the heterogeneity of their data set by using unsupervised learning to form groups that match criteria through automatic optimisation features.

3.6 Classification of this work

This work is about the evaluation of new methods that create a partition for the prediction of different fault classes for different engines. The focus of this work is on two tasks: The first task is related to a pattern recognition task, since the goal is to recognise structures or patterns within a data set. For this purpose, a method is developed that uses the specific properties of the provided data set. With this approach, new domain knowledge of the data set can be explored. The second task falls in the domain of machine learning, but the goal is to evaluate techniques that use semi-supervised learning for the classification process of multi-class imbalance data problems. This uses existing and explored domain knowledge to generate different sets of constraints.

The decision to use a semi-supervised learning approach is due to the real-world nature of the problem: Usually there is some kind of domain knowledge that is available or can be generated by a domain expert. Using domain knowledge and semi-supervised learning could be another possible approach for data sets with the above challenges.

4 Constraint-based clustering

Constraint-based clustering, or (constraint clustering), is a category of semi-supervised machine learning algorithms. These algorithms use constraints to integrate domain knowledge of experts, a deeper look into constraints can be found in Section 4.1. Afterwards, the used algorithms are introduced starting with two partition-based clustering algorithms. The first algorithm is COP-Kmeans detailed in Section 4.2, followed by the second algorithm MPCK-Means presented in Section 4.3. The last algorithm is a density-based clustering algorithm named CDBSCAN in Section 4.4. Then the chapter ends with an introduction into active learning in Section 4.5.

4.1 Constraints

There are different types of constraints. In this thesis, we primarily use instance-based constraints, namely must-link and cannot-link. As can be seen in Figure 4.1, cannot-links can be used to separate two clusters, while must-link allows two points within a cluster to be connected. Unfortunately, these pairwise constraints can lead to error conditions, some Constraint-based algorithms can deal with the errors. It then depends on the algorithm how it deals with the error conditions. Some algorithms are able to ignore constraints, others abort. We use constraints to map complex relationships between different data points, or to map domain knowledge about product groups. In the following, we explain the origin and properties of the constraint-based clustering algorithms used in this work.

4.2 COP-Kmeans

Wagstaff et al. [WCRS01] presented their algorithm Constraint K-Means (COP-Kmeans) in 2001. This modifies the K-Means clustering algorithm proposed by Stuart Lloyd [Llo82] by adding constraints to the clustering process. However, the COP-Kmeans algorithm allows constraints specifying the relationship between two data points to be included in the clustering process. Like the regular K-Means algorithm, the COP-Kmeans algorithm uses centroids to calculate the distances between all points and the cluster centroid. The point is assigned to the cluster with the shortest

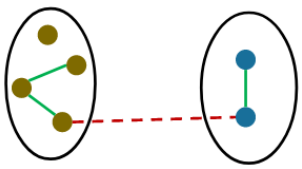


Figure 4.1: Must-link (green) and cannot-link (red dashed) constraints

distance to the cluster centroid. When points are assigned regularly, the algorithm checks whether the assignment violates the constraints. The algorithm uses a set of must-link and a set of cannot-link constraints. COP-Kmeans can not decide against constraints, so it can abort during execution. By integrating constraints, the accuracy of cluster detection could be significantly improved.

4.3 MPCK-Means

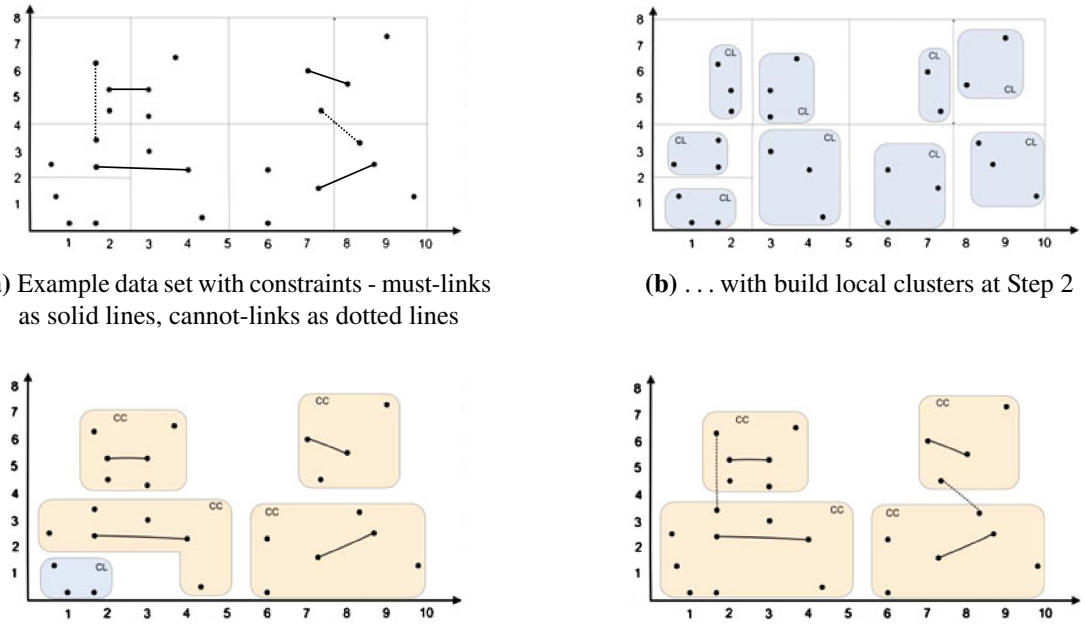
Bileno et al. [BBM04] published MPCK-Means, another variant of the K-Means algorithm, in 2004. Like COP-Kmeans, MPCK-Means uses constraints to represent domain knowledge. It uses constraints in cluster initialisation and point assignment. However, MPCK-Means additionally uses metric learning during the point assignment step, which allows it to ignore constraints, albeit at a cost. For example, it is more expensive to violate must-link constraints with a larger distance than must-link constraints with a smaller distance. Violating cannot-link points that are close together is also more expensive than violating cannot-link constraints with a larger distance. This is done with the intention of preventing the algorithm from breaking down if the constraints are inconsistent. Using the cost of the constraints, as well as the calculated distances, the cost of each point is calculated pairwise. The metric learning function then decides on the most favourable connection of two points and assigns the clusters accordingly.

4.4 CDBSCAN

CDBSCAN, introduced in 2007 by Ruiz et al.[RSM07] is an extension of the density-based DBSCAN algorithm of Ester et al. [EKSX96]. Density-based clustering is based on adding all points in the radius epsilon around a point that is already in a cluster or a starting point. Thereafter, is then repeated for all points added to the cluster. CDBSCAN works on the basic principle of density based clustering, but with some changes described in the following four steps:

- Step one: Partitioning the data sets depending on the constraints Figure 4.2a.
- Step two: Create of local clusters based on density-based clustering, taking into account the cannot-link constraints Figure 4.2b.
- Step three: Merging local clusters, taking into account the must-link constraints Figure 4.2c.
- Step four: Merging clusters considering density clusters and cannot-link constraints Figure 4.2d.

CDBSCAN does not allow constraint violation, so the algorithm may fail in the presence of inconsistent constraints. The algorithm is partially resistant to noise due to the density-based clustering approach, depending on the configuration, or density of the noise. Thanks to the constraints, CDBSCAN can reliably work with different densities in a data set, whereas normal DBSCAN is vulnerable to this.



(a) Example data set with constraints - must-links as solid lines, cannot-links as dotted lines

(b) . . . with build local clusters at Step 2

(c) Step 3 - merge clusters and enforce must-link constraints

(d) . . . and Step 4 - build final clusters

Figure 4.2: Example for CDBSCAN from Ruiz et al. [RSM09]

4.5 Active Learning

Active learning is usually used when labelled data is not available, it allows the labelling of selected data points that promise a good value by a metric. Figure 4.3 shows an abstract version of the active learning loop. It starts on the left with the initial calculation of a set of scores for all unlabeled data. The next step is to select the best-fitting data point according to the used metric. Following this, the unlabeled data is retrieved in correlation to the values and the domain expert is asked to label the point. The domain expert then answers the question, and the loop then continues with the newly labelled data.

Generally, the data is labelled by a human or a domain expert. However, the user is not asked to label all the data, usually he has the option to just finish the labeling process and the following algorithms have to deal with the amount of labeled data. The data that should be labeled next can be selected by any method: from a simple distance metric to the calculation of a machine learning model. Here are a few examples of active learning:

- Kumar et al. [KG20] present an overview of different pool-based active learning scenarios.
- Settles et al. [SC08] present different active learning methods based on uncertainty, Committee and Information Density.
- Zhaoet et al. [ZSWC19] presents a Gaussian mixture approach for active learning and demonstrates it's capabilities on common clustering data sets
- Dasgupta et al. [DH08] present an AL approach based on hierarchical clustering, capable of finding rare classes in the data set.

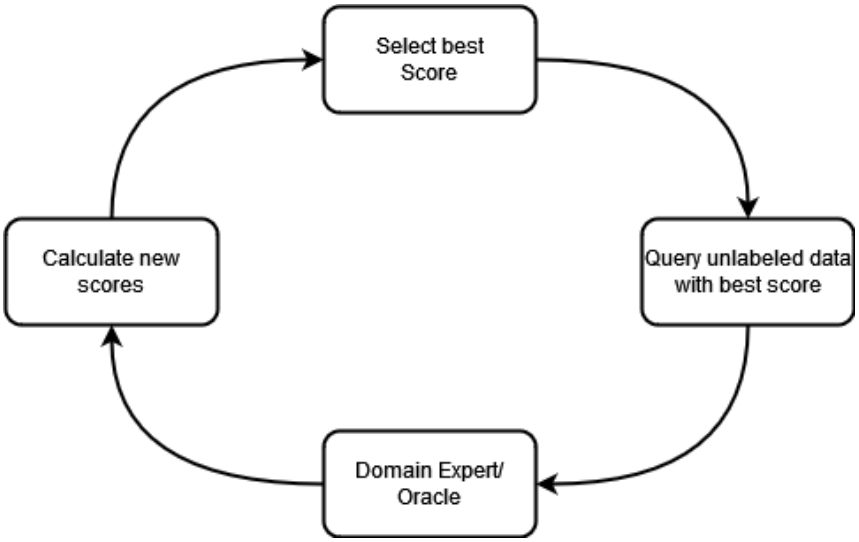


Figure 4.3: Active learning loop

- Vu et al. [VLB10] present a simple modified MinMax algorithm based on distance metrics for selecting data instances.
- Bilgic et al. [BMG10] presents active learning for graphs representing domain knowledge by relations between data instances.

The query itself can take a variety of forms, e.g. "Are point x and point y in the same group"or "Is point x an outlier". Typically, these questions aim to label the data that the metric finds useful, or to gain valuable insight into a data set that machine learning would otherwise struggle to understand. In our case, we emulate the domain expert with an oracle since we already have a fully labeled data set. However, this allows us to use this technique to create partitions in new ways.

5 Implementation

This chapter focuses on the various extensions to the existing Hirsch et al. implementation. As can be seen in Figure 5.1, the original pipeline has been extended in two places: the new domain knowledge extraction section and the training set preparation section. Section 5.1 on domain knowledge extraction contains ways to create new partitions. Using active learning, new partitions can be created by the involvement of a domain expert. Furthermore, a method is introduced that allows to extract domain knowledge from the provided data set and creating a partitioning. Additionally, further modified versions of the existing hierarchy are implemented. In the Section 5.2, the constraint-based clustering algorithms are implemented. In addition, an algorithm is implemented that represents the maximum convergence of the grouping for constraint-based clustering algorithms. The derivation of constraints from domain knowledge is also explained in this section.

5.1 Domain Knowledge Extraction

Two strategies are being implemented, for domain knowledge extraction, in this new part of the pipeline, as shown in Figure 5.2. These are required for the use cases in which no hierarchy or product grouping is available. They focus on creating either a new grouping or a new hierarchy by using the existing data set.

The first strategy in Section 5.1.1 is called active learning. The result of active learning is then used to create a grouping either by clustering with a CDBSCAN algorithm or classification with Random Forest. It takes the provided data set and labels the data instances with the help of a domain expert to create a new partitioning of the data instances. It is then followed up by a partitioning algorithm in Section 5.1.2, which uses missing features and then transforms these partitions into a new hierarchy. Furthermore, it is now possible to modify the original and the new hierarchy. However, these new groupings and hierarchies need to update the existing data structures. This

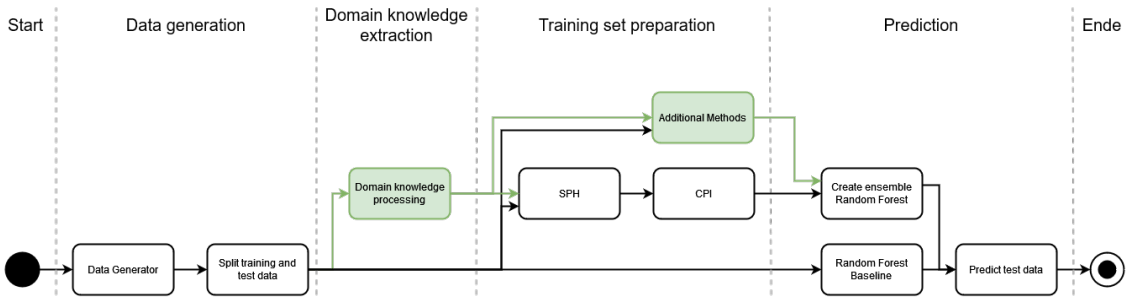


Figure 5.1: Pipeline overview with added features

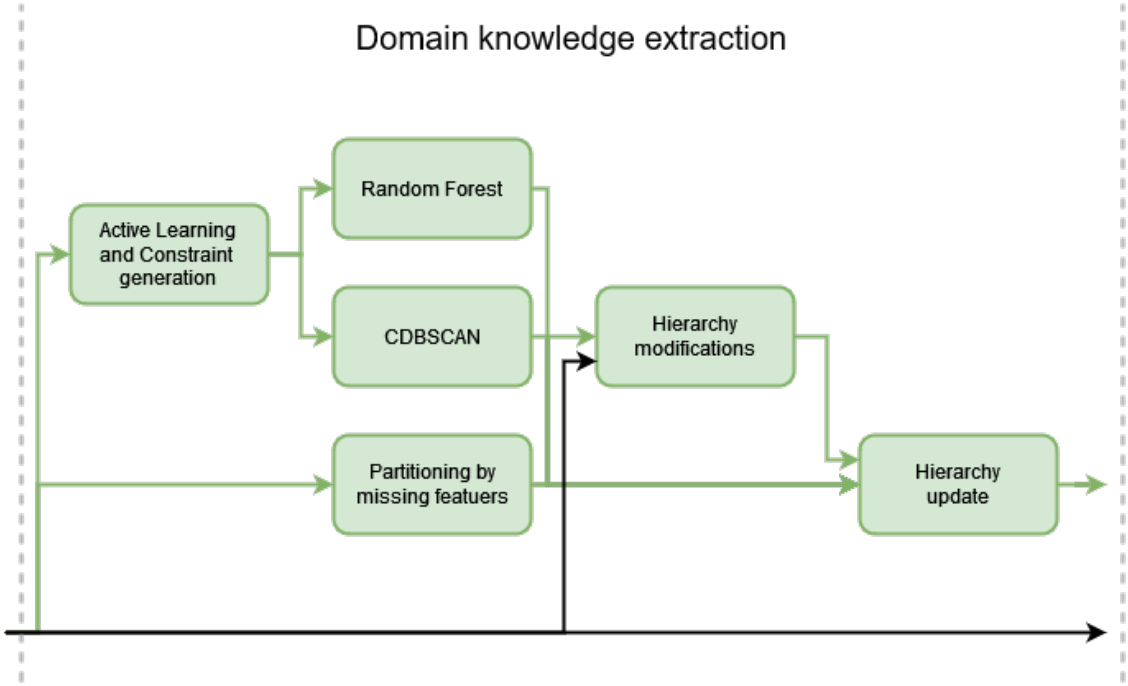


Figure 5.2: Pipeline, Domain knowledge extraction

is done in the hierarchy update step. Additionally, any strategy or machine learning technique capable of handling the analytic challenges presented in Section 2.1, and which creates a grouping or partition of sorts, could be used here.

5.1.1 Active learning and constraint generation

There are many applications for active Learning in the context of data analysis. Active learning is used for the situation where a domain expert is available, but no explicit formalized domain knowledge about the data set. The domain expert is trying to recreate the original group distribution from the data set. The active Learning algorithm asks questions of the form: "Is point A in the same group as point B?". The MinMax approach for active learning chooses data points that are as far away as possible from all the already identified points. Since the data instances are not evenly distributed over all groups, many queries are required to compensate for the imbalance. The data set is a high dimensional space, therefore the Manhattan distance is used for the distance calculation, which performs better in high dimensional feature space according to [AHK01]. Otherwise, active learning would mostly select instances of the majority groups. Furthermore, the different features in the provided data set represent sensor values and in a real world case can't be compared against each other. Therefore, a pairwise metric is necessary to minimize the influence of different value ranges. Since the product groups have different value ranges, the distance between the points can have a greater impact on the amount of queries required to find a decent amount of product groups. By questioning the domain expert or Oracle, usually only a part of the instances of the data set is assigned to a group. The following paragraphs show two solutions for dealing with not fully labeled data sets.

Random Forest: Random Forest uses the groups of data sets already identified through active Learning as training data. A forest of size 200 is trained to classify all remaining unlabelled points.

CDBSCAN: Active learning has already identified groups distributed over a certain set of data instances. Must-link and cannot-link constraints are generated from these groups between the instances. CDBSCAN then uses these constraints to cluster the remaining data instances.

5.1.2 Partitioning by missing features

In this section we consider the case where no hierarchy or product group is known about the data set and no domain expert is available to provide further information. A good start is always to look at the data and its properties. In the context of this work, where a variety of properties that can be used exists, this algorithm focuses on: missing sensor data and different value ranges of individual sensors. They represent the analytic challenge: Heterogeneous product portfolio, missing features and Sub-concepts. This is possible since the data set consist of different products with different components and therefore different missing features, thereby making this approach a speciality and not something that will work commonly on every data set. Since Nan values are easier to differentiate than different value ranges, the focus is initially on the former. The goal of the algorithm is to find partitions that are as big as possible. However, a few rows can not be unambiguously assigned to a partition, those rows are assigned by a classifier in each assignment iteration. The algorithm uses the Nans in the different columns to find a combination of columns that satisfies the following conditions: First, as few Nans as possible that are in the same row across all columns. Secondly, as few rows as possible without Nans. Rows with no or more than one Nan are then assigned to the different partitions by a classifier. The classifier uses the partitioning created by the Nans for training and then predicts the rows that could be located in two or more of the trained partitions. Then the partitioning step is repeated until no smaller partitions can be found. In another separate algorithm the created partitions are then converted to a hierarchy. For each new partition found a new node is then added to the hierarchy. The algorithm itself is separated in the following three functions:

The first function of the algorithm *Iterator* shown in Algorithm 5.1 focuses on iterating over the individual stages of partitioning. The code begins with the creation of the first partition in line 2 with the *findPartition* method explained in the second function. In addition, a list of the individual partitioning steps is created, in line 3. This can later be converted into a hierarchy. The algorithm runs through the partitioning until no more sub-partitions are created, in line 5. For each partition already found, the corresponding subsets are calculated and then partitioned again, lines 8 to 14. If the partition is no longer changed, the iteration is terminated, lines 15 and 16. The new partitions are then added to the list of all partitions for later use, in line 19. Finally, the collected partitions are returned, in line 22.

The second function in pseudo code is *findPartition* presented in Algorithm 5.2, and it works with the different combinations of columns. It first creates a collection of all columns with a Nan value and then filters all columns with less than at least $\theta = 2$ instances in lines 2 and 3. This is done because we need at least two instances for the classification step later in the pipeline. There is another filter value that is set by the analyst ϵ with the default value of 0.2. The filter is set so that

Algorithm 5.1 Iteration

```
1:  procedure ITERATOR(nanMatrix, points, ϵ, θ)
2:      partition ← findPartition(nanMatrix, points, ϵ, θ)        // Create first partition
3:      labels ← empty list
4:      labels append partition                                    // Save first partition
5:      while not finished do          // Continues as long as new partitions are found
6:          finished ← True
7:          currenPartition ← partition
8:          for partitionIndex in Partition do
9:                                     // Look for further partitioning for the existing partitions
10:             nanMatrixSubset ← subset(partitionIndex, partition, nanMatrix)
11:             pointsSubset ← subset(partitionIndex, partition, points)
12:             subsetLabels ← findPartition(nanMatrixSubset, pointsSubset, ϵ, θ)
13:                                                // Further information in Algorithm 5.2
14:             currentPartition ← putAtPosition(partitionIndex, partition, subsetLabels)
15:             if subsetLabels not False then
16:                 finished ← False                          // Reset while escape condition
17:             end if
18:         end for
19:         labels append currentPartition                         // Save new partition
20:         partition ← currentPartition
21:     end while
22:     return labels
23: end procedure
```

each column has at least column size * ϵ and at most column size * (1-ϵ) Nans in the column. ϵ can be selected in two ways: A small value, for larger partitions and more iterations, or a higher value, for more iterations and medium sized partitions. This is important so that no column full of Nans is used to calculate a partition, but also so that noise is filtered out to some extent. The recursive function *findSolution* Algorithm 5.3 is then called with the first column and the remaining columns, which satisfying the filter, in line 7. This then returns a set of solutions, with the best solution simply being selected. If no solutions are returned, the function returns False, lines 9 and 10. However, the solution still has data instances that are not evaluated during the recursive algorithm. This was not done in earlier steps, because it had to be calculated for more than one permutation, line 12. Therefore, for the best solution from the set, the undecided indices are classified with the imputed instance values, lines 15 to 20. Finally, a set of labels for the current partitioning step is returned.

In the third function *findSolution* in Algorithm 5.3 the recursive part of the code is handled. If the current solution has at least two columns, the columns are evaluated, line 2. First, the indices of the rows in which a Nan value lies are at the same index in at least two columns, this set is called intersection, line 4. Next, the indices of the rows in which only one Nan value occurs are selected, denoted as uniques, line 5. Finally, the indices of the rows in which no Nan values occur are selected, which is called missing, line 6. If the number of overlapping and missing rows is greater than half the size of the column, the solution is skipped and the recursion is stopped, lines 7 to 9. This has the benefit that the amount of data instances that have to be predicted in the classification step is small enough, but it also reduces the execution time, since no time is wasted with column

Algorithm 5.2 Partition Finder

```
1:  procedure FINDPARTITION(nanMatrix, points, ϵ, θ )
2:      columns ← findColumusWithNan(nanMatrix)
3:      columns ← filterColumns(columns, ϵ, θ)
4:                                          // Reduces the number of Permutations drastically
5:      for column in columns do
6:          aktSolution ← column
7:          solutions ← findSolution(aktSolution, columns)
8:      end for
9:      if solutions is None then
10:         return False
11:     end if
12:     bestSolution ← getBestSolution(solutions)      // Picks lowest score from all solutions
13:     undecidedInstances ← getUndecided(bestSolution, points)
14:                                          // Use Indices with real values for classification
15:     if undecidedInstances ≥ 0 then
16:         decidedInstances ← getDecided(bestSolution, points)
17:         clf ← trainClassifier(decidedInstances)
18:         classifiedLabels ← clf classify(undecidedInstances)
19:         updateLabels(bestSolution, classifiedLabels)
20:     end if
21:     return getLabels(bestSolution)
22: end procedure
```

combinations that would lead to an even worse result. If the number of missing rows is less than about one-fifth the size of the column, the current solution is added to the set of returned solutions with all values evaluated, line 11. If a solution is added or if no solution is evaluated, the remaining columns are added one after the other in a loop, lines 18 to 20. In this way, all possible permutations are modelled by the recursive call of this function, line 21. Following the recursive call, the column is removed again, line 22. After this for loop, all possible solutions are returned, line 24.

Create new hierarchy: By logging the individual partitions, a hierarchy can be created after the partitioning is done in an extra step: Starting with a root node that contains all data instances, then taking the first partitions form the first iteration, and create a new node for each changed partition index. This is then appended as child and all points with the index are appended to the new node. Accordingly, this is repeated for every subsequent logged iteration of the partitioning algorithm. If a partition index in a subsequent partition does not change, it is ignored and no new node is added.

5.1.3 Hierarchy modifications

Some of the leaves of the hierarchy have a very small number of data instances. One approach is to combine small leaf nodes that have the same parent node. This theoretically reduces class imbalance, similar to the SPH step of Hirsch et al. [HRM20] For the present data set, our goal is to combine all leaves of a node with less than 15 instances. Furthermore, new methods have

Algorithm 5.3 Solution Finder

```
1:  procedure FINDSOLUTION(aktSolution, columns, solutions)
2:      if length(aktSolution) > 2 then                          // Test possible solution
3:          columnSize ← length(column[0])
4:          intersection ← countRowsWithOverlappingNan(aktSolution)
5:          uniques ← getRowsWithOneNan(aktSolution)
6:          missing ← getRowsWithNoNan(aktSolution)
7:          if length(intersection) + length(missing) ≥ (columnSize * 0.5) then
8:                  // If more than half the data is undecided, the solution is not good enough
9:              return solutions
10:         end if
11:         if length(missing) ≤ (columnSize * 0.2) then
12:             // The Solution is good enough that it can be added to the list of possible solutions
13:             score ← length(missing) + length(intersection)
14:             solution ← (aktSolution, intersection, missing, uniques, score)
15:             solutions append solution
16:         end if
17:     end if
18:     remaining ← columns - aktSolution
19:     for column in remaining do                               // Try all permutations of columns
20:         aktSolution append column
21:         solutions ← findSolution(aktSolution, remaining, solutions)
22:         aktSolution remove column
23:     end for
24:     return solutions
25: end procedure
```

been implemented to change the hierarchy itself. One such methods is to remove all levels of the hierarchy between the leaf and the root node to create a flat hierarchy. Another method is the removal of all leaf nodes.

5.1.4 Hierarchy updates

The active learning approach, the hierarchy modification, and partitioning by missing features change the distribution of the original product groups. This means that the hierarchy has to be adjusted, otherwise the SPH + CPI algorithm can not handle the new hierarchy. This solves issues that otherwise would occur during the prediction step. The constraint-based clustering algorithms are solving this problem by matching the test data to clusters. This matching calculates the distance for every test data instance and the cluster centroids. The closest cluster of the centroid is then used for the prediction in the ensemble step according to the clustering partition.

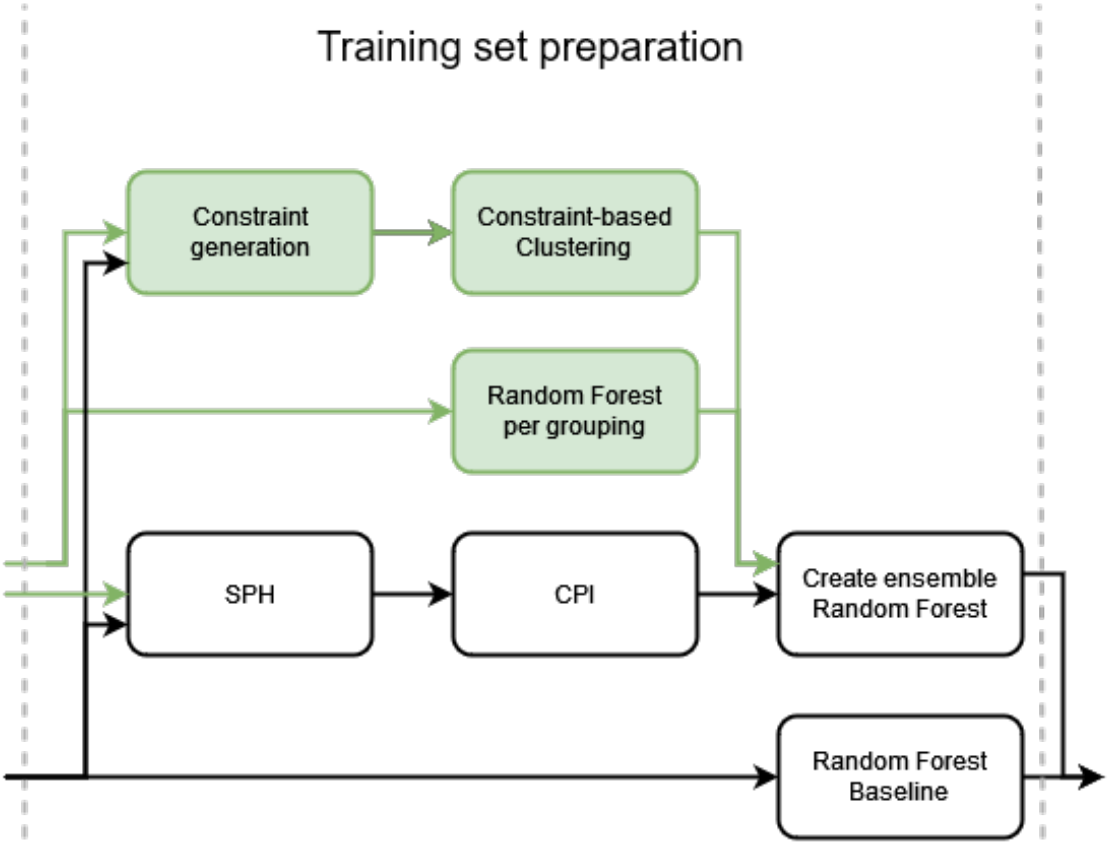


Figure 5.3: Pipeline, Training set preperation

5.2 Training set preparation

The Training set preparation uses the available Domain Knowledge, which is already extracted and pre-processed into the form of a group distribution or a hierarchy. Accordingly, it creates a partitioning of the training data that then can be used to create an ensemble for the final classification step. In the area of training set preparation seen in Figure 5.3, the constraint generation methods and constraint-based clustering methods were integrated. Additionally, the RF per Group algorithm was implemented. Also, the implemented algorithms can use all variants of the groupings of the domain knowledge extraction of the training set. In addition, the structure of the existing pipeline has been extended so that more different clustering algorithms can be evaluated. Furthermore, it is now also possible to evaluate other methods for partitioning.

5.2.1 Constraint sets

Constraints are used to support the clustering process by using domain knowledge. The aim of this work is to evaluate the effects of different sets of constraints. For the generation of constraints all information available about the training data set can be used, the focus is on the hierarchy and the original product groups. However, all domain knowledge should have been integrated into the possible product groupings and hierarchies by this point.

Constraints from error classes The first variant is the creation of constraints according to error classes. Within a error class, all instances are connected by must-link constraints. Cannot-link constraints are used to separate error classes from each other. The partitioning is represented in Table 5.1, each color and number representing one error class. The existing hierarchy is greyed out since it has no influence over the partitioning. During the ensemble step, this would train a Random Forest per error class.

root	Engine																	
Level-1	mde								hde									
Level-2	mde...		mde3						hde1							hde...		
Level-3	mde...		mde3-...			mde3-13			hde1-1			hde1-...				hde...		
Classes	5	6	7	2	4	5	5	2	1	8	3	1	3	3	6	7	9	8	1

Table 5.1: Constraints from error classes

Constraints from product groups This constraint generation attempts to work in a similar fashion as the partitioning of the SPH step by Hirsch et al. [HRM20]. Constraint sets with a low amount of constraints can be used to ensure that there is room to merge smaller groups. Since not every time every product group is separated through cannot-link constraints, additionally MPCK-Means is able to overrule cannot-link constraints. Must-link constraints connect the individual instances within a product group or leaf node. Cannot-link constraints prevent the links between product groups or leaf node. This is visualised in Table 5.2, inside one of the colors must-links are used. Meanwhile, the red lines mark boundaries through cannot-link constraints.

root	Engine																	
Level-1	mde								hde									
Level-2	mde...		mde3						hde1							hde...		
Level-3	mde...		mde3-...			mde3-13			hde1-1			hde1-...				hde...		
Classes	5	6	7	2	4	5	5	2	1	8	3	1	3	3	6	7	9	8	1

Table 5.2: Constraints from product groups

Constraints from different engine types Another version similar to the partitioning by product groups, but with more degrees of freedom. It uses cannot-link between the two main groups of engines mde and hde, marked in red in Table 5.3. Must-link constraints will still be used to connect the indices inside of one product group or leaf node. Or for other hierarchy forms, cannot-link constraints are between the children of the root node.

root	Engine																	
Level-1	mde								hde									
Level-2	mde...		mde3						hde1							hde...		
Level-3	mde...		mde3-...			mde3-13			hde1-1			hde1-...				hde...		
Classes	5	6	7	2	4	5	5	2	1	8	3	1	3	3	6	7	9	8	1

Table 5.3: Constraints from different engine types

5.2.2 Constraint-based clustering

This Subsection contains the small changes made to the constraint-based clustering algorithms described in Chapter 4. The CDBSCAN, MPCK-Means and COP-Kmeans procedures have already been implemented in a preliminary project. Therefore, this subsection will go into more detail about the necessary modifications. For use with the provided data set, the distance metrics were updated if they were still based on Euclidean distance. For this, the Manhattan distance was used again. The original group distribution can easily be changed by the constraint-based algorithms, which is why an extension was added. It allows the instances of the test set to be classified according to their proximity to the different cluster centres, thus fixing the problem of changes in the partition. All existing product groups are relabeled during the clustering process, and their data instances get mixed up. In case of the CDBSCAN, changes to the partition can occur during merging of clusters or outright declaring them as noise. K-Means based clustering algorithms use a preset which lets them know how many clusters are to expect, but doesn't solve the allocation of the test set. This preset is chosen based on the cannot-link constraints, since they are the most restricting. There can not be more clusters than the cannot-link constraints allow, thereby the algorithms should not throw a Empty cluster exception. However, there is the case where not all clusters will be found. Cannot-link constraints are either generated between product groups so the count of product groups is chosen or they are based on the error classes so the count of the error classes is used.

5.2.3 Random Forest per Group

Random Forest per Group (RF per Group) works in such a way that a separate Random Forest is generated for each product group or leaf node. The other approaches: SPH + CPI or constraint-based clustering make changes to the existing grouping. This gives us a comparative value that allows us to better evaluate the changes in the different product groups by our methods. Specifically for constraint-based clustering on the original product groups, it gives us a picture of what 100% of the constraints would amount to. Furthermore, it shows how different sets of constraints behave.

6 Evaluation

In this chapter, the results of this thesis are presented and discussed. At the beginning of the chapter, the evaluation setup is introduced explaining the data generation, missing feature challenge and the used metrics in Section 6.1. Section 6.2 focuses on the constraint-based clustering algorithms. Furthermore, it explains the behaviour of convergence towards the partition the constraints are derived from. Then the main evaluation of the results from the constraint-based clustering algorithms and other partitioning algorithms is given, followed by an overview of the accuracy of the different algorithms and a discussion of the overall behaviour. At the end of the section, additional constraint sets are evaluated. Section 6.3 provides an overview into the behaviour of the partitioning algorithms when influenced by different degrees of multi-class imbalance. Section 6.4 then attempts to further explain the behavior of used algorithms on a flat hierarchy and a hierarchy without leaf nodes. Finally, the different techniques for extracting domain knowledge which modify the product hierarchy are compared in Section 6.5, thereby evaluating active learning and an algorithmic approach to extract a new hierarchy from the data set.

6.1 Evaluation setup

In this section, the structure of the evaluation is explained in more detail, starting with an update about the data generator. Next, the problem of missing features for the different partitioning algorithms is addressed. This is followed by an overview of the approaches evaluated, and finally the different metrics used in the evaluation are explained.

6.1.1 Data generation

Due to the confidentiality agreement, the original data set used by Hirsch et al.[HRM19], [HRM20] is not accessible. Furthermore, Vitali Hirsch is no longer at the University of Stuttgart. However, Dennis Tschechlov, M.Sc., currently a researcher at the Institute for Parallel and Distributed Systems at the University of Stuttgart, has taken up the research, and developed a data generator. This data generator is capable of generating synthetic data sets that contain the analytical challenges listed in Section 2.1. The data generator additionally allows the regulation of the class imbalance in five different degrees, namely: very low, low, normal, high, very high. Very low has an average Gini coefficient of 0.32, low has a Gini coefficient of 0.42, the normals Gini coefficient is 0.52, the Gini coefficient of high is 0.54, very high has the Gini coefficent of: 0.57. Furthermore, the values of the individual data instances are different for each generation, which causes variance in the data set. The training data set for the prediction consists of 750 data instances, while the test data set consists of 300 data instances.

6.1.2 Addressing the Missing features challenge

The missing feature challenge, presented in Section 2.1, is addressed in two ways in this thesis. Missing features are identified by Nan values in the data set. However, there are also randomly missing values, these Nan values represent another reason a value might not exist in the data set. The first way is kNN-imputation, but this can have an negative impact in the context of our data set. If kNN-imputation uses data from other columns and rows, it is likely that the value obtained will make no sense in the context of the feature. If kNN-imputation only considers the current column, it is possible that the current column consists only of Nan values, which is a known problem of the heterogeneous product portfolio. The second option is a simple substitution by zero. This method is not accurate in comparison to an imputed value, but most of the Nans inside the data set should not have a value since the feature doesn't exist for a certain product group. By not imputing the randomly missing values, there is theoretically loss in possible accuracy. Both options can be used, but generally yield different results. A combination of both techniques is possible by using a detection for single Nan values followed by a kNN-imputation and a substitution for the rest. However, in the case of this thesis the substitution with zero is chosen, mainly because there are more missing features than randomly missing values.

6.1.3 Evaluated approaches

During this chapter different approaches are evaluated. Starting with the general behaviour of constraint-based clustering and their convergence to the used partition, using the product group. Afterwards, a comparison between the different algorithms is given using constraint-based algorithms, SPH + CPI and RF per Group, this will provide an overview of the prediction accuracy. Then different sets of constraints are evaluated, one set based on the error classes and another based on a single forced separation of the two main engine types. This is done to find a better way to partition the data set with the already existing product hierarchy. Furthermore, different degrees of multi-class imbalance are evaluated to get a further insight in the behaviour and stability of the partitioning algorithms. Afterwards, the scenario of not having prior domain knowledge is addressed with two approaches. The first approach is based on active learning and uses a domain expert to create a new partitioning of groups. The second approach uses missing features and different value ranges challenge to create a new hierarchy. Finally, two forms of hierarchy are evaluated to further validate the behaviour of the the used algorithms.

6.1.4 Metrics

An accuracy score called $A@e$ is used, which describes the accuracy (A) of a prediction for a faulty component at ($@$) a number of attempts (e). The score also is an average, usually over multiple runs at the specific attempt. R_e is the number of the prediction or rework attempt, in this case it's a list of ten entries. In addition, R_e correlates with $A@e$ and describes the accuracy for the prediction attempts. Furthermore, the Gini coefficient is used to describe the imbalance between the classes. The value 1 for the Gini coefficient represents maximum imbalance, while a 0 represents an equal distribution. Class accuracy represents the percentage of correctly predicted error classes. The F-Score describes the accuracy of the predictions for the error classes, based on precision and recall. The value 1 represents maximum recall and precision while a 0 is a total miss.

6.2 Constraint-based clustering results

This section evaluates the methods used in preparation of the training set area in the pipeline. First, the convergence of the constraint-based clustering algorithms is examined. Then, the constraint-based clustering evaluation using the provided hierarchy, while the accuracy of the evaluated partitioning algorithms is evaluated. Furthermore, a discussion about the results of SPH + CPI, and RF per Group is presented. The section then ends with an evaluation of the different possibilities for generating constraints from the provided product hierarchy. During this section Table 6.1, Figure 6.2 and Figure 6.3 use the same underlying predictions of 100 runs.

6.2.1 Convergence of Constraint-based clustering algorithms

Figure 6.1 shows the effects of different percentages of constraints. The constraints used for this section are derived from product groups, thereby creating cannot-link constraints between the product groups and must-link constraints inside the product group. CDBSCAN is used as the representation for all constraint-based clustering algorithms used in this work. The set of constraints used is shown in the legend of Figure 6.1 after CDBSCAN. Increasing the number of constraints also increases the accuracy. 2% constraints result in a CDBSCAN that matches the predictions of the first three of the Random Forest baseline, but can not keep up. With 5% CDBSCAN is already better than the baseline up to the ninth prediction. 15% constraints already provide an improvement in accuracy of more than 10% in accuracy from the second prediction onward. **The sweet spot for this data set and constraint set is at about 30%**, for a good accuracy and stability in the result. As the number of constraints used increases, the CDBSCAN algorithm begins to approximate RF by grouping partitioning algorithm. This is because the constraints per product group recreate parts of the original product group.

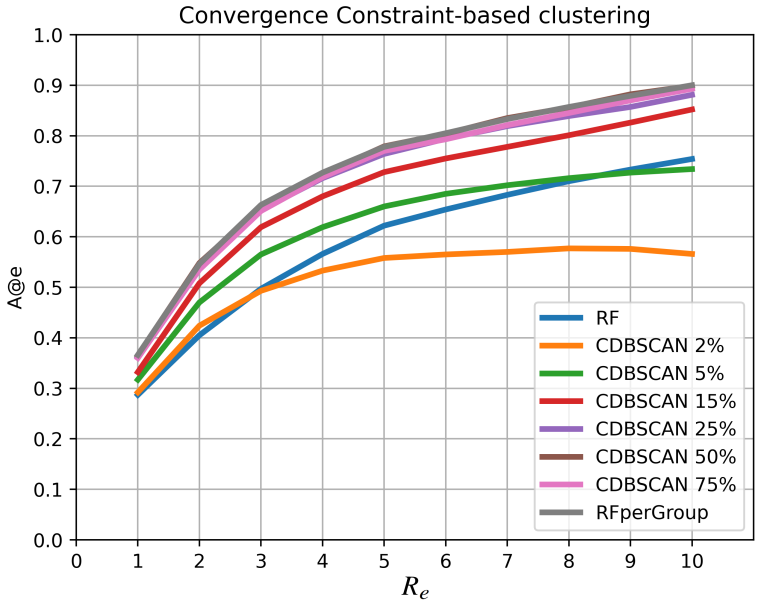


Figure 6.1: Convergence in constraint based clustering

6.2.2 Evaluation of different partitioning algorithms

This evaluation uses the provided data set from data generation with the normal degree of imbalance. In addition, Figure 6.2 and Table 6.1 show the average results of all partitioning algorithms over 100 generated data sets with the same parameters. This generates 28 partitions with 69 error classes, with a Gini coefficient of 0.52. For these evaluations, the number of constraints used is set to 30%. The parameter for the number of clusters for K-Means based algorithm is picked by the count of product groups.

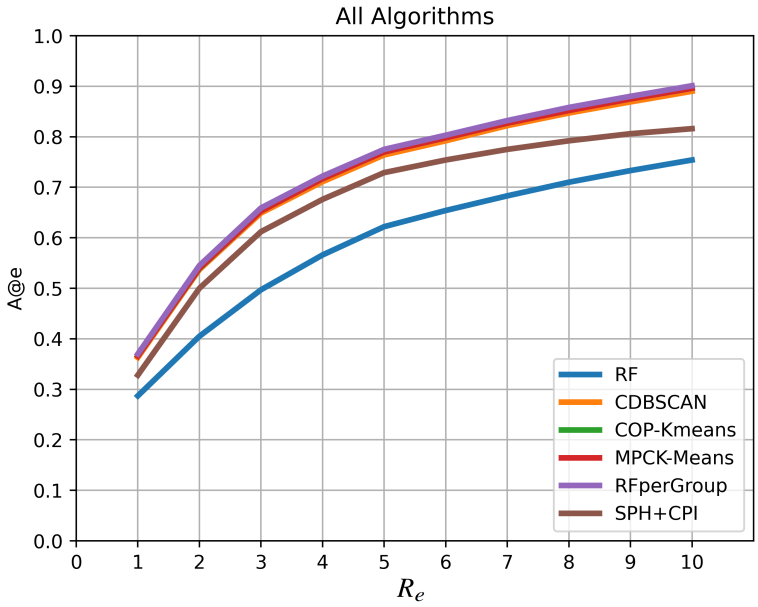


Figure 6.2: Accuracy of different partitioning algorithms

Algorithm	R_e 1	R_e 2	R_e 3	R_e 4	ø10 Δ
RF Baseline	28.7	40.5	49.7	56.6	0.00
ΔSPH + CPI	4.1	9.5	11.5	11.0	8.77
ΔCDBSCAN	7.6	13.2	15.2	14.5	13.33
ΔMPCK-Means	7.9	13.5	15.6	15.1	13.83
ΔCOP-Kmeans	7.9	13.4	15.6	15.1	13.83
ΔRFperGroup	**8.3**	**14.0**	**16.2**	**15.6**	**14.34**

Table 6.1: Evaluation of partitioning methods

Figure 6.2 already gives an overview of the averaged results, showing the general order. The Random Forest baseline makes starts with the least accurate predictions, followed by the SPH + CPI partitioning algorithm. Accompanied by the constraint-based algorithms, and finally the RF per Group algorithm with the best average result. Table 6.1 provides accurate averaged results for the different partitioning algorithms, while Figure 6.2 provides the general order as an overview. The table shows the $A@e$ values, and has the Random Forest baseline. For all other algorithms, the Random Forest baseline is subtracted. This makes it easy to judge whether a method gives a better

result than a Random Forest execution. Negative scores are worse than the baseline, and positive scores are better. In addition, the scores are given as percentages for better readability. Furthermore, the best accuracy values per prediction attempt are highlighted. The score ø10Δ represents the average increase or decrease in accuracy over all ten predictions. Furthermore, only the first four of the fault isolation attempts are shown in the tables, since these are relevant as stated by Hirsch et al. [HRM20].

The Random Forest baseline starts with an average accuracy of 28.7% for the first prediction of the faulty part. The second prediction climbs to an accuracy of 40.5% while the third prediction is at 49.7%. Prediction number four passes through the 50% mark with 56.6%, all other partitioning algorithms surpass the 50% accuracy by their second prediction. SPH + CPI improves on all RF baseline results, with 4.1% for the first prediction, 9.5% for the second prediction, followed by 11.5% for the third and 11.0% for the fourth prediction. Finally, the accuracy increases by 8.77% for all ten predictions compared to Random Forest. The constraint-based algorithms perform similarly, such that the increase in accuracy for prediction one is 7.8%, followed by 13.4% for prediction two. Prediction three increases accuracy by 15.5% compared to the Random Forest baseline. The last prediction increases the accuracy by 14.9%, and on average over all ten predictions the increase in accuracy is 13.66%. The RF per Group algorithm yields the best results of all partitioning algorithms. Its first accuracy increase is 8.3%, the second 14.0%, the third 16.2%, the fourth 15.6% with an average over all ten predictions of 14.34%. **This shows that the partitioning by product group is already close to the local maximum or close to the optimum**. All partitioning algorithms are better than the Random Forest baseline. However, SPH + CPI is falling behind the constraint-based clustering algorithms by approximately 4,5%, since **SPH + CPI either adds data instances that are misleading the classifier or it removes important data instances**. Since, the CDBSCAN algorithms converge as shown in Section 6.2.1 it is to expect that the results of the different constraint-based algorithms are similar to each other. Therefore, in the current configuration, they can not surpass RF per Group in accuracy. RF per Group seems to be the best solution for the current partitioning, as the other algorithms that try to improve the results beyond a product group partitioning lead to a loss of accuracy.

6.2.3 Result accuracy

Figure 6.3 also gives an insight into the distribution of the accuracy of the prediction results of the 100 runs. Only the first four predictions of the fault isolation attempts are shown, as these are relevant according to Hirsch et al. [HRM19]. On average, the results of the different partitioning algorithms scatter around the median by about 1.25%, as the upper and lower quartiles are about 2.5% apart. Therefore, the **1.25% is used as non significant margin for comparisons between the different partitioning algorithms**. Overall, the SPH + CPI partitioning algorithm has the most outliers. The constraint-based clustering algorithms are behaving very similar due to the convergence shown in Section 6.2.1. However, their quartils and medians are very similar as well. During the four predictions the distribution of the accuracy results is not changing significantly.

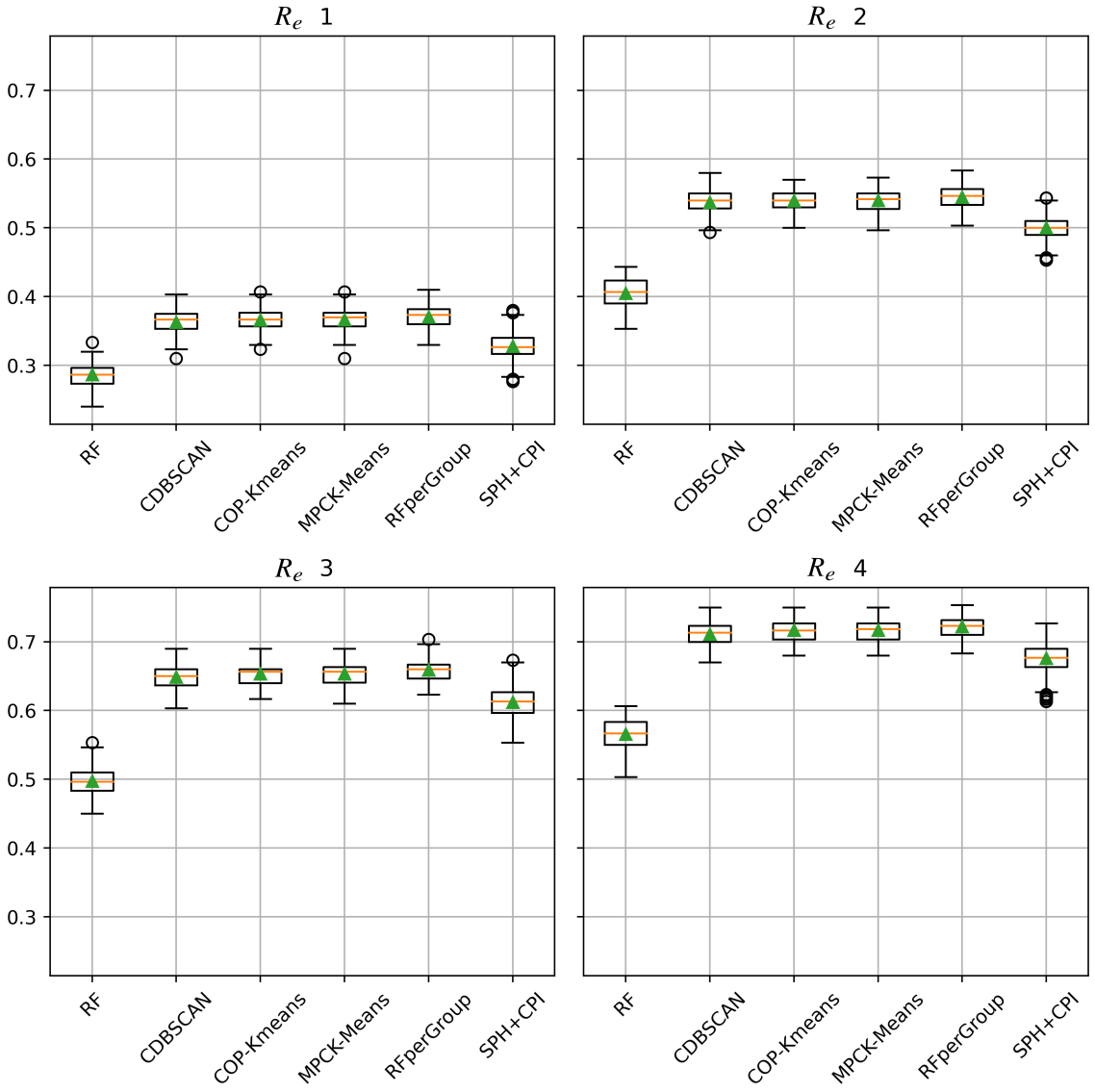


Figure 6.3: Box plot for the first four picks of faulty components

6.2.4 Discussion

Behaviour constraint-based clustering Constraint-based clustering is highly dependent on constraints, as can be seen in Section 6.2.1. A look into the different accuracy for partitions, or clusters in this case, shows that some partitions have no predictions. This is due to the workaround for the ensemble generation. The ensemble for a data instance is selected by finding the nearest cluster centre. Therefore, some data instances and their effects on the prediction are not unused. This is especially important to note with CDBSCAN, which is not limited to a certain number of clusters; here, even more data instances are unused. However these unused clusters are small with less than five data instances for 30% of constraints, with a lower amount of constraints these values increase, the same happens with the noise labeled by CDBSCAN. At 30% constraints, about 20 data instances are labelled as noise by CDBSCAN, and about 80 at 10% of the constraints. Fortunately,

Method	Group	Run	# Nan	# Instances	Gini	Accuracy per class	F score per class
RFperGroup	mde-OM3-11	1	120	5	0.30	1.0	1.0
RFperGroup	mde-OM3-11	2	94	4	0.25	1.0	1.0
SPH + CPI	mde-OM3-11	1	6492	270	0.29	0.67	0.80
SPH + CPI	mde-OM3-11	2	10434	435	0.19	0.0	0.0
RFperGroup	mde-OM3-9	1	770	32	0.52	1.0	1.0
RFperGroup	mde-OM3-9	2	688	29	0.53	1.0	0.98
SPH + CPI	mde-OM3-9	1	530	22	0.18	0.7	0.82
SPH + CPI	mde-OM3-9	2	594	25	0.06	0.62	0.81

Table 6.2: Possible SPH + CPI errors

the other constraint-based clustering algorithms must label all data instances, which explains why CDBSCAN has slightly lower accuracy. This also shows that **each data instance is important for the current ensemble solution**. Most of the lower accuracy compared to RF per group is due to the regrouping of the data instances. By sticking close to the product group the influence of sub-concepts is reduced. Constraint-based clustering is not able to recognize sub-concepts, since at most similar data instances are added to a cluster.

Behaviour SPH + CPI RF per Group uses the original product group partition without any modification. SPH + CPI modify this original product group partition to lessen the effect of the analytic challenges. Since RF per Group yields better results than SPH + CPI, it raises some questions. To find out what this might be causing, the behaviour of the SPH + CPI algorithm needs to be investigated in more detail. Therefore, each partition is analysed for both the SPH + CPI and RF per Group algorithms to see the differences. Table 6.2 contains two examples where SPH + CPI are likely to result in a loss of accuracy. The partition or product group mde-OM3-11 contains five data instances with two classes, where one class is represented once and the other four times. Therefore, the SPH step opted for a higher hierarchy level depending on the loss of information or representation of the classes, adding more data instances. This is later partitioned by the CPI step since the Gini coefficient is greater than 0.3. Through SPH there are more classes and consequently other classes are predicted that do not belong to the group, decreasing the accuracy. This is a trend for this product group in all further runs, as an example, the second run is also included here. In the next example: mde-OM3-9, only the CPI step is executed, but again there is a loss of accuracy. We can identify the CPI step, since the instance count is lower for mde-OM3-9 than RF per Group, both are using the same data set in the same run. It seams that the loss is higher when the SPH step is involved. These examples show that **both the SPH step and the CPI step can have an impact on accuracy, as positive or negative**. However, there are also examples in the data set where SPH + CPI work perfectly and do not result in a loss of accuracy.

6.2.5 Evaluation of different set of constraints

Table 6.3 provided below shows the results of the first four predictions for the different constraint sets introduced in Section 5.2.1. The left side shows the constraints by engine type and the right side shows the constraints by error class. Since the use of different sets of constraints has no impact on SPH + CPI or RF per group, only the constraint-based clustering algorithms and the Random

	Different engine types					Error classes				
Algorithm	R_e 1	R_e 2	R_e 3	R_e 4	ø10Δ	R_e 1	R_e 2	R_e 3	R_e 4	ø10Δ
RF Baseline	28.7	40.5	49.7	56.6	0.00	**28.7**	**40.5**	**49.7**	**56.6**	**0.00**
ΔCDBSCAN	**7.6**	**13.5**	**15.4**	**15.1**	**13.42**	-7.3	Nan	Nan	Nan	Nan
ΔMPCK-Means	4.7	10.0	10.8	11.1	10.67	Nan	Nan	Nan	Nan	Nan
ΔCOP-Kmeans	Nan	Nan	Nan	Nan	Nan	Nan	Nan	Nan	Nan	Nan

Table 6.3: Comparison of different set of constraints

Forest baseline are shown in Table 6.3. The constraint-based algorithms were run with 30% of the constraints, and a total of ten data sets were generated for evaluation to get a general direction of what happens when using the different constraint sets. A Nan in the table is an indicator that the pipeline failed at some point and no successful prediction was possible. The default cluster number for the K-Means-based algorithms is determined by the number of product groups for the engine type-based constraint and the number of error classes for the error class-based constraint.

Constraints based on different engine types The first four results are generated using the constraints based on different engine types. In this case, the goal is to group different product groups to merge together based on their similarity. The main difference to the constraints by product group is that the only cannot-link constraints are between the machine types MDE and HDE. The accuracy of CDBSCAN is still within the 1.25% of the upper quartiles that the algorithm normally has Section 6.2.3. MPCK-Means loses accuracy with an overall result of 10.67% over all ten runs, but is still an improvement over the Random Forest baseline. COP-Kmeans does not provide a partition and thus no prediction. This is due to either a hard-coded limit of 1000 attempts to find a cluster partition or the inability to find all expected cluster, both cases lead to exceptions. The solution to this problem seems simple: increase the amount of constraints. The results would still be similar to the CDBSCAN result presented in Table 6.3, as the 30% of constraints is already sufficient to achieve convergence. Additionally, it would improve the results of MPCK-Means and maybe COP-Kmeans would produce results. However, Section 6.2.1 explains that this ultimately leads to convergence on the original partition of product groups from which the constraints were derived.

Constraints based on existing error classes The second set of constraints is based on existing error classes of the data set. In this case, CDBSCAN is only able to deliver one prediction. However, CDBSCAN is not failing to deliver a partitioning. The reason for this situation is the following: **Since the partition only contains one error class, there is only one error class to predict**, and therefore only one prediction can be made, leaving R_e 2 and the others empty with Nans. For constraints by error class, more constraints would not help, since the issue occurs during the prediction. Furthermore, since no properties of the constraint-based clustering algorithms are used, in case of the error classes, it would be easier and with less overhead in computation to simply use the error classes as partitions. Therefore, using more constraints is not always the solution.

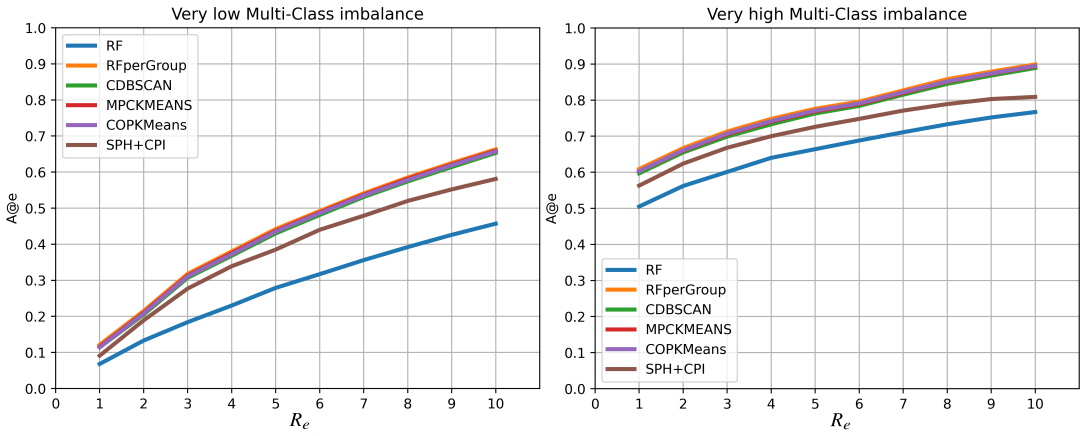


Figure 6.4: Different degrees of multi-class imbalance

6.3 Comparison of different degrees of multi-class imbalance

Figure 6.2 already shows the normal degree of multi-class imbalance. Figure 6.4 provide information about the behaviour of the partitioning algorithms at different degrees of multi-class imbalance. For this comparison, the settings for very low, normal and very high multi-class imbalance were chosen as they mark the middle, high and low ends of the available settings. There is no discernible difference in the order of the algorithms based on their accuracy results. However, **a change in the degree of imbalance results in a change in the overall performance of all algorithms**. Very low imbalance leads to very low overall accuracy, while high imbalance leads to higher accuracy. This is due to the fact that more data instances are labelled with the majority error classes and are therefore easier to predict. Looking at accuracy by error class, it shows that about 15 minority error classes simply can not be predicted at a normal degree of imbalance with the current amount of data instances. While six out of 84 error components are missing at a very low degree of imbalance, the accuracy for the majority classes also drops. This suggests that there are too few data instances for good prediction of so many error classes.

6.4 Hierarchy modifications

This section is used to evaluate different forms of hierarchies for the training set preparation, and to validate the behaviour observed in the previous approaches. This is done by modifying the original product hierarchy form the provided data set.

6.4.1 Flat hierarchy

The modified hierarchy can be seen in Figure 6.5. The first hierarchy form is a flat hierarchy, presented on the left in Table 6.4, a hierarchy containing only the root node and leaf nodes. The constraint-based clustering algorithms are not affected by this, as they use the product groups for constraint generation. SPH + CPI use the root node for the SPH step as there are no other nodes in between, one would expect this to affect accuracy largely, but **only about seven product groups**

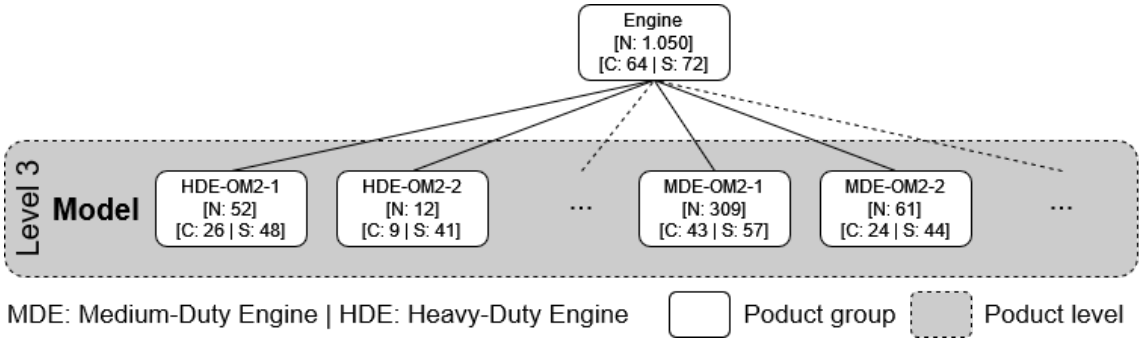


Figure 6.5: Flat hierarchy

	Flat hierarchy					Hierarchy without leave nodes				
Algorithm	R_e 1	R_e 2	R_e 3	R_e 4	ø10Δ	R_e 1	R_e 2	R_e 3	R_e 4	ø10Δ
RF Baseline	28.7	40.5	49.7	56.6	0.00	**28.7**	**40.5**	**49.7**	**56.6**	**0.00**
ΔSPH + CPI	3.9	8.8	10.4	10.3	7.63	-3.7	-4.1	-5.3	-6.1	-5.14
ΔCDBSCAN	7.5	13.8	15.4	15.1	13.71	-1.8	-1.2	-2.0	-3.0	-2.87
ΔMPCK-Means	**8.2**	14.2	15.9	15.6	14.25	-1.8	-1.2	-2.0	-3.0	-2.87
ΔCOP-Kmeans	8.1	14.2	15.9	15.6	14.22	-1.8	-1.2	-2.0	-3.0	-2.87
ΔRFperGroup	**8.2**	**14.3**	**16.0**	**15.7**	**14.34**	-1.8	-1.2	-1.9	-2.9	-2.75

Table 6.4: Evaluation of hierarchy modification

trigger the SPH step, so this only affects about a quarter of the product groups. Yet there is still a negative effect created by the SPH step since the new data instances increase the heterogeneous product portfolio challenge of the surrogate groups. Furthermore, when looking at accuracy by product group, it is noticeable that the seven surrogate groups perform worse on average, by about 30%.

6.4.2 Removed leaf nodes

The modified hierarchy can be seen in Figure 6.6. The second hierarchy form, shown on the right-hand side of Table 6.4, is done by removing all leaf nodes and only the level above the removed leaves is used to create constraints and partitions, for the training set preparation. With this change, all algorithms are less accurate than the Random Forest baseline. SPH + CPI are on average 5.14% worse than the plain Random Forest baseline. Constraint-based clustering algorithms are 2.87% behind the Random Forest baseline in accuracy. The RF per Group partitioning algorithm also loses to the Random Forest baseline by 2.75%. However it is at least slightly more accurate than the other algorithms. This shows that a careful take to clustering is necessary, since unspecific partitioning will result in a loss in accuracy. **The overall loss in accuracy results from less specialized Random Forest trees during the prediction**. With the original leaf nodes they had on average ten error classes per node, now they have 36 error classes per node. These are all error classes with different relevant features, degrees of imbalance and different value ranges for features that are sometimes the same. Furthermore, the value ranges of the different sub-concepts, are now mixed into each other, making the decision process of a Random Forest tree even more difficult.

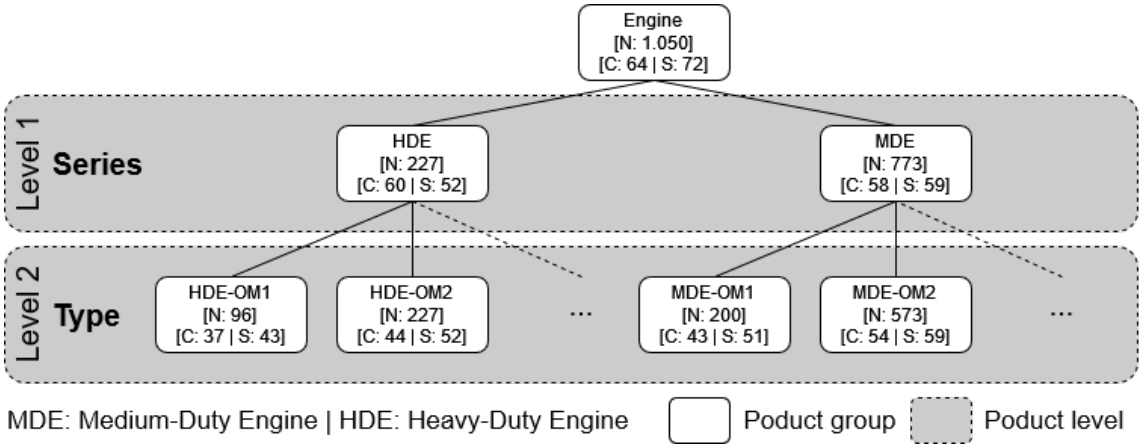


Figure 6.6: Removed leaves

6.5 No prior domain knowledge

In this section an evaluation of the different approaches from Section 5.1 is presented. The following section presents cases where there is little or no domain knowledge on the data set, especially no hierarchy or product group. Either it is possible to consult an domain expert or know certain properties of the data set that can be used for a more accurate prediction of the faulty component. Therefore, this section starts with active learning in Section 6.5.1. The second approach uses the missing features of the generated data set and use it for partitioning the data set in Section 6.5.2. Finally, some special cases of hierarchy forms are evaluated with the implemented algorithms in 6.4. The partitions and hierarchies created in this section of the pipeline can be used instead of the originals supplied with the data set.

6.5.1 Active learning and constraint generation

As described in Section 5.1.1, active Learning is used to obtain information about a data set from a domain expert. Specifically, the aim is to identify as many different product groups as possible with their corresponding data instances. Especially, in the context of multi-class imbalance and sub-concepts, data instances have to be selected efficiently for labeling. Active learning is only meant to label a small amount of the data set. With the labeled data instances different approaches can be used to partition the remaining data instances, for example: Classification with Random Forest and constraint-based clustering or with constraints generated from the labeled data instances. The results of these approaches are then used as partition for the other approaches in the training set preparation area of the pipeline. However, the results strongly depend on the number of allowed questions. For our data set with a normal degree of imbalance (Gini coefficient of 0.52), about 500 questions are required to find all but one group, by using a MinMax approach[VLB10] for the selection of data instances. Nevertheless, about 200 data instances could be identified and then used to label the remaining data instances by various means.

Classification A random forest classifier is trained to partition the remaining data instances into the identified product groups. The results are presented in Table 6.5 on the left side. All algorithms perform better compared to the Random Forest baseline in accuracy. SPH + CPI loses on accuracy compared to the original hierarchy, this is due to the usage of the root node as surrogate group by the SPH step. Instead of adding only a few more data instance, now all 750 data instances are used as surrogate groups, however the CPI step reduces them to about 300 to 450 data instances. The difference compared to original data set with a hierarchy is 1,33%, which is somewhat lower than the lower quartile of 1.25% from Section 6.2.3. Furthermore, this behaviour is analysed closer in Section 6.4. However, all algorithms show higher accuracy in the first prediction than those of the original product groups, and the further differences compared to the original hierarchy are minimal. These changes in accuracy are still within the range of variation, as presented in Section 6.2.3. The constraint-based clustering algorithms are within the lower quartile and therefore on par with the accuracy results of the provided data set. RF per group also loses accuracy compared to the original data set, but as the constraint-based clustering algorithms the difference is within the respective 1.25%.

Constraint-based clustering Here, the CDBSCAN algorithm was chosen because it does not require a predetermined number of clusters and is therefore more suitable for this exploration task. Other algorithms such as COP-Kmeans and MPCK-Means would require prior knowledge about the number of clusters present, which is usually not available in this situation and therefore was not evaluated. Constraint-based clustering is less successful in reconstructing the original product group, which is presented in the right side of Table 6.5. Although the constraints were created based on 200 data instances similar to the classification approach, they do not provide sufficient support for the density-based CDBSCAN algorithm. CDBSCAN recognises many of the data instances as noise and therefore does not assign them to a cluster or partition that won't be used for a prediction. On average, SPH + CPI loses 8.70% accuracy, compared to the Random Forest baseline. CDBSCAN with -11.1% is worse than the other constraint-based clustering algorithms, since there are even fewer groups to draw constraints from. The partition-based clustering algorithms perform better with -7.18 and -6.39% than CDBSCAN, but still worse than the Random Forest baseline. This is mainly because they aim to add all points to a cluster based on the available constraints. RF per group uses the 200 available data instances and creates the ensemble based on them. With an average gain of 0.24% compared to the Random Forest baseline, it performs better on average and on the first predictions, but loses on the later predictions compared to the Random Forest baseline.

Comparing the use of active learning with classification and active learning with constraint-based clustering, the classifier is clearly better for the existing data set with it's analytic challenges.

6.5.2 Partitioning by missing features

This subsection evaluates the results of the partitioning approach presented in Section 5.1.2. First, the parameters and the resulting hierarchy are presented, then the results are examined in more detail. For this evaluation, the partitioning was executed with $\epsilon = 0.20$ and $\theta = 5$. ϵ filters out the randomly missing values and θ should be at least 2, otherwise the group is unsuitable for the ensemble. In addition, the respective groups should not become too small, which is why the value 5 was chosen for θ.

	Classification					Constraint-based clustering				
Algorithm	R_e1	R_e 2	R_e 3	R_e 4	ø10Δ	R_e 1	R_e 2	R_e 3	R_e 4	ø10Δ
RF Baseline	28.7	40.5	49.7	56.6	0.00	28.7	40.5	49.7	56.6	0.00
ΔSPH + CPI	5.4	9.2	10.1	9.8	7.44	-4.9	-5.1	-7.2	-9.0	-8.70
ΔCDBSCAN	7.8	13.2	14.0	13.5	12.63	-6.7	-6.4	-8.9	-11.1	-10.69
ΔMPCK-Means	8.4	13.6	14.1	13.7	12.87	-6.6	-6.3	-6.4	-7.1	-7.18
ΔCOP-Kmeans	**8.5**	13.5	14.5	13.7	12.69	-7.7	-6.5	-5.2	-4.8	-6.39
ΔRFperGroup	8.3	**13.8**	**14.6**	**14.0**	**13.37**	**1.7**	**2.0**	**2.4**	**1.7**	**0.24**

Table 6.5: Evaluation of hierarchy modification

Algorithm	R_e1	R_e 2	R_e 3	R_e 4	ø10 Δ
RF Baseline	28.7	40.5	49.7	56.6	0.00
ΔSPH + CPI	**6.9**	**9.5**	**10.4**	10.1	8.51
ΔCDBSCAN	2.5	5.4	6.1	6.0	4.49
ΔMPCK-Means	2.7	5.3	6.4	6.7	5.86
ΔCOP-Kmeans	4.1	7.6	8.8	9.0	6.95
ΔRFperGroup	4.5	8.3	10.2	**11.3**	**10.56**

Table 6.6: Comparison of partitioning methods with hierarchy of missing features

The hierarchy itself now has 52 leaf nodes distributed over levels two to eight, therefore the hierarchy is no longer balanced. The largest leaf nodes are in the upper levels while the smallest are in the lowest. Furthermore, the largest leaf node has about 200 data instances varying during the different runs, all others are between 6 to 50 data instances. The imbalance of error classes is still present.

The different partitioning techniques give interesting results, presented in Table 6.6. For this partitioning, **SPH + CPI is better in the first three predictions compared to RF per Group**. Furthermore, **the first prediction is 2.8% better than the first prediction for the original hierarchy**. The eight levels of the hierarchy allow for the SPH step to move up the hierarchy and adding smaller amounts of data instances to the small partitions. Thereby, the later trained ensemble has to learn less decision-making boundaries compared to the original hierarchy. The overall accuracy of SPH + CPI is the same as the accuracy of the original partitioning. Constraint-based clustering still performs better than the random forest baseline by 4 to 7%. The loss in accuracy compared to the original partition is due to the fact that more clusters exist. In the original hierarchy the same amount of constraints draw, a more detailed picture of the grouping, since there are less product groups. RF per Group shows that the wide spread partitioning of 52 groups is not the best solution for this data set. This is the main reason why SPH + CPI is gaining in accuracy, since it has the ability to create a better partitioning, by taking surrogate groups that fit better.

Partitioning by missing features is designed to work with some information about the data set using the heterogeneous product portfolio and the missing features of the data set. With an increase in accuracy of 8.51% for SPH + CPI and 10.56% for RF per Group in addition to the baseline it is a useful tool for the current data set.

7 Conclusion & Future work

7.1 Conclusion

Quality control is an important tool for manufacturers to ensure the quality of their products, as it directly reduces the risk of refunds for defective or non-functioning products. Furthermore, if a faulty part can be isolated, it can be replaced or repaired, which is more profitable, and also more sustainable, than throwing the whole product away. However, the data collected in quality control brings some challenges, as Hirsch et al. [HRM20] states heterogeneous product portfolio, small sample size and multi-class imbalance are the most prominent examples. These analytical challenges can be solved individually, but are challenging in their totality, especially in the context of a classification problem, in this case the prediction of faulty components. Therefore, constraint-based clustering is evaluated in this work as it allows partitioning with the influence of domain knowledge through constraints. The existing product hierarchy can be translated into constraints and thus enrich the results of the clustering process. However, it is also possible to use the constraint-based clustering without any constraints and therefore domain knowledge. These partitions by clustering are evaluated and analysed in context of the analytic challenges.

In this thesis, constraint-based clustering was implemented in the form of CDBSCAN, COP-Kmeans and MPCK-Means. They were developed with different constraint sets derived from the data set, product groups or product hierarchy to work with each scenario of available domain knowledge. In addition, two approaches were developed and integrated for scenarios where no domain knowledge is available. An active learning algorithm was implemented to serve as an interface for domain experts. After some data instances were labelled, a clustering or classification algorithm can now label the remaining data instances. An algorithmic solution has also been developed to create a new product hierarchy with limited knowledge about the data set. This uses the analytical challenge of missing features to identify the product groups and create a similar partitioning. All individual components were integrated into the existing pipeline.

Constraint-based clustering algorithms tend to converge against the partitioning the constraints are derived from. This behavior was also evaluated to find good balance between the amount of constraints and the results. The different approaches were then evaluated against a Random Forest baseline and in most cases improve the accuracy of the baseline. Constraint-based clustering with sufficient constraints leads to an overall 14% improvement in accuracy compared to the Random Forest baseline and a 5% improvement over the existing SPH + CPI approach. Two approaches to create constraints were evaluated: the first by using the product hierarchy to either differentiate the product groups or the different engine types. The second approach is based around the error classes of the training data. However, the use of error class constraints was less accurate, the first constraint set yielded similar results to the constraint set by product groups. Furthermore, the behaviour of the constraint-based clustering algorithms was evaluated for different degrees of imbalance. The effect of the increase in imbalance effected all algorithms equally, there was no deviation in the

constraint-based clustering algorithms. In addition, the behaviour of the implemented algorithms as well as that of the existing SPH + CPI algorithm was investigated and explained as part of this thesis. Through active learning, a domain expert can now create a product grouping that is within the error margin of the original product group. The algorithmic approach to creating a new hierarchy is also better than the baseline Random Forest. However, the constraint-based clustering algorithms loss about 7% in accuracy for the newly created hierarchy. RF per Group loses about 3% compared to the original hierarchy. With the new hierarchy, SPH + CPI is performs 2.8% better in the first prediction than with the original hierarchy.

The evaluation shows that a partitioning by original product group is either a local maximum or close to the optimal solution for the data set. Moreover, the prediction lacks data, especially in minority groups, and the small sample size leaves little to no room for change. However, there might be possible improvements that will be discussed in the next section.

7.2 Future work

A minimum of one more data sets should be evaluated, to see if the behaviour of the different training set preparation methods influence the results for the better, or if a straight cut partitioning is the better all time solution.

Looking at the hierarchy without leaf nodes and the flat hierarchy, it becomes clear that classification requires partitioning. Further partitions are interesting to research: A partitioning or constraint set based on the used components of a product, which would lead to a partitioning with larger groups. This might also be useful if an engine has more than one faulty component. Another type of partitioning would be possible if the minority classes in each product group get additional data instances from neighbouring product groups with the same error class to counter the imbalance. This differs form the SPH step in that no unrelated data instances are added.

In the terms of clustering, a fuzzy clustering technique could be of interest, as it would create bigger partitions with the same amount of existing data, but could also lead to overfitting of the majority classes. Therefore, a selective fuzzy clustering would be of interest as to only allow minority classes to be used in multiple clusterings. A form of subspace clustering would also be interesting for the data set. There are matching clusters for different subspaces, the relevant dimensions could be identified by the identical component parts used in the different engines. Furthermore, the different value-ranges of the sub-concepts would separate the different product groups by their components.

Less specific but more general improvements are: A simple addition would be a kNN-impurer for noise in the form of not a numbers in the data set, as it is possible to identify and distinguish missing values of the noise from the missing features of heterogeneity. In addition, the data set can be further cleaned and mislabelled data instances could also be identified and reassigned as all products have different value ranges for features. While these are small steps, they would eventually add up to a small increase in accuracy and can be incorporated into the pipeline without much effort. Also, Random Forest randomly selects features for tree generation. Not all available features in the data set are relevant for the current product group or possible error class, a selected combination of these

two aspects could also lead to an increase in accuracy. One way or another, all error classes should be considered, currently only 69 of 84 are predicted. This also due to the lack of data instances. A solution to this problem would be oversampling.

Bibliography

[AHK01] C. C. Aggarwal, A. Hinneburg, D. A. Keim. “On the Surprising Behavior of Distance Metrics in High Dimensional Space”. In: *Database Theory — ICDT 2001*. Ed. by J. Van den Bussche, V. Vianu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 420–434. ISBN: 978-3-540-44503-6 (cit. on p. 30).

[BBM04] M. Bilenko, S. Basu, R. J. Mooney. “Integrating Constraints and Metric Learning in Semi-Supervised Clustering”. In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML ’04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 11. ISBN: 1581138385. DOI: 10.1145/1015330.1015360 (cit. on p. 26).

[BMG10] M. Bilgic, L. Mihalkova, L. Getoor. “Active Learning for Networked Data”. In: June 2010, pp. 79–86 (cit. on p. 28).

[DH08] S. Dasgupta, D. Hsu. “Hierarchical sampling for active learning”. In: *Twenty-Fifth International Conference on Machine Learning*. 2008 (cit. on p. 27).

[EGV+18] L. Ehrlinger, T. Grubinger, B. Varga, M. Pichler, T. Natschläger, J. Zeindl. “Treating Missing Data in Industrial Data Analytics”. In: *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*. 2018, pp. 148–155. DOI: 10.1109/ICDIM.2018.8846984 (cit. on p. 23).

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, X. Xu. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD’96. Portland, Oregon: AAAI Press, 1996, pp. 226–231 (cit. on p. 26).

[HBTK14] R. Houari, A. Bounceur, A. K. Tari, M. T. Kecha. “Handling Missing Data Problems with Sampling Methods”. In: *2014 International Conference on Advanced Networking Distributed Systems and Applications*. 2014, pp. 99–104. DOI: 10.1109/INDS.2014.25 (cit. on p. 23).

[HG09] H. He, E. A. Garcia. “Learning from Imbalanced Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284. DOI: 10.1109/TKDE.2008.239 (cit. on pp. 23, 24).

[HRM19] V. Hirsch, P. Reimann, B. Mitschang. “Data-Driven Fault Diagnosis in End-of-Line Testing of Complex Products”. In: *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 2019, pp. 492–503. DOI: 10.1109/DSAA.2019.00064 (cit. on pp. 13, 17, 18, 20, 24, 39, 43).

[HRM20] V. Hirsch, P. Reimann, B. Mitschang. “Exploiting Domain Knowledge to address Multi-Class Imbalance and a Heterogeneous Feature Space in Classification Tasks for Manufacturing Data”. In: 2020, pp. 492–503. DOI: 10.1109/DSAA.2019.00064 (cit. on pp. 13, 17, 18, 24, 33, 36, 39, 43, 53).

[KBT11] G. Köksal, İ. Batmaz, M. C. Testik. "A review of data mining applications for quality improvement in manufacturing industry". In: *Expert Systems with Applications* 38.10 (2011), pp. 13448–13467. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2011.04.063. URL: https://www.sciencedirect.com/science/article/pii/S0957417411005793 (cit. on p. 22).

[KG20] P. Kumar, A. Gupta. "Active Learning Query Strategies for Classification, Regression, and Clustering: A Survey". In: *Journal of Computer Science and Technology* 35.4 (2020), pp. 913–945. ISSN: 1860-4749. DOI: 10.1007/s11390-020-9487-4. URL: https://doi.org/10.1007/s11390-020-9487-4 (cit. on p. 27).

[Kra16] B. Krawczyk. "Learning from imbalanced data: open challenges and future directions". In: *Progress in Artificial Intelligence* 5.4 (2016), pp. 221–232. ISSN: 2192-6360. DOI: 10.1007/s13748-016-0094-0. URL: https://doi.org/10.1007/s13748-016-0094-0 (cit. on p. 22).

[LLE16] L. Leitner, A. Lagrange, C. Endisch. "End-of-line fault detection for combustion engines using one-class classification". In: *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. 2016, pp. 207–213. DOI: 10.1109/AIM.2016.7576768 (cit. on p. 21).

[Llo82] S. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489 (cit. on p. 25).

[RSM07] C. Ruiz, M. Spiliopoulou, E. Menasalvas. "C-DBSCAN: Density-Based Clustering with Constraints". In: *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*. Ed. by A. An, J. Stefanowski, S. Ramanna, C. J. Butz, W. Pedrycz, G. Wang. Berlin, Heidelberg: Springer Berlin Heidelberg, Jan. 2007, pp. 216–223. ISBN: 978-3-540-72530-5 (cit. on p. 26).

[RSM09] C. Ruiz, M. Spiliopoulou, E. Menasalvas. "Density-based semi-supervised clustering". In: *Data Mining and Knowledge Discovery* 21.3 (Nov. 2009), pp. 345–370. DOI: 10.1007/s10618-009-0157-y (cit. on p. 27).

[SC08] B. Settles, M. Craven. "An Analysis of Active Learning Strategies for Sequence Labeling Tasks". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '08. Honolulu, Hawaii: Association for Computational Linguistics, 2008, pp. 1070–1079 (cit. on p. 27).

[SGG18] H. Suresh, J. J. Gong, J. V. Guttag. "Learning Tasks for Multitask Learning: Heterogenous Patient Populations in the ICU". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*. KDD '18. London, United Kingdom: Association for Computing Machinery, 2018, pp. 802–810. ISBN: 9781450355520. DOI: 10.1145/3219819.3219930. URL: https://doi.org/10.1145/3219819.3219930 (cit. on p. 24).

[VLB10] V. Vu, N. Labroche, B. Bouchon-Meunier. "Active Learning for Semi-Supervised K-Means Clustering". In: *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*. Vol. 1. Oct. 2010, pp. 12–15. DOI: 10.1109/ICTAI.2010.11 (cit. on pp. 28, 49).

[WCRS01] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl. *Constrained K-means Clustering with Background Knowledge*. Jan. 2001, pp. 577–584 (cit. on p. 25).

[Wei04] G. M. Weiss. “Mining with Rarity: A Unifying Framework”. In: *SIGKDD Explor. Newsl.* 6.1 (June 2004), pp. 7–19. ISSN: 1931-0145. DOI: 10.1145/1007730.1007734. URL: https://doi.org/10.1145/1007730.1007734 (cit. on pp. 22, 23).

[WWIT16] T. Wuest, D. Weimer, C. Irgens, K.-D. Thoben. “Machine learning in manufacturing: Advantages, challenges, and applications”. In: *Production and Manufacturing Research* 4 (June 2016), pp. 23–45. DOI: 10.1080/21693277.2016.1192517 (cit. on p. 21).

[XZF13] L. Xuan, C. Zhigang, Y. Fan. “Exploring of clustering algorithm on class-imbalanced data”. In: *2013 8th International Conference on Computer Science Education*. 2013, pp. 89–93. DOI: 10.1109/ICCSE.2013.6553890 (cit. on p. 22).

[ZASA18] A. Zabihihesari, S. Ansari-Rad, F. Shirazi, M. Ayati. “Fault detection and diagnosis of a 12-cylinder trainset diesel engine based on vibration signature analysis and neural network”. In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 233 (June 2018), p. 095440621877831. DOI: 10.1177/0954406218778313 (cit. on p. 21).

[ZSWC19] J. Zhao, S. Sun, H. Wang, Z. Cao. “Promoting active learning with mixtures of Gaussian processes”. In: *Knowledge-Based Systems* 188 (Sept. 2019), p. 105044. DOI: 10.1016/j.knosys.2019.105044 (cit. on p. 27).

All links were last followed on October 20, 2021.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature