CIplus
Band 1/2017

Conditional Inference Trees for the Knowledge Extraction from Motor Health Condition Data

Alexis Sardá-Espinosa, Subanatarajan Subbiaha, Thomas Bartz-Beielstein

Technology
Arts Sciences
TH Köln

Conditional Inference Trees for the Knowledge Extraction from Motor Health Condition Data[☆]

Alexis Sardá-Espinosa, Subanatarajan Subbiah[a], Thomas Bartz-Beielstein[b]

[a]*ABB AG*
German Research Center
Wallstadter Straße 59
68526 Ladenburg
[b]*Technische Hochschule Köln*
Faculty of Computer Science and Engineering Science
Steinmüllerallee 1
51643 Gummersbach

Abstract

As the amount of data gathered by monitoring systems increases, using computational tools to analyze it becomes a necessity. Machine learning algorithms can be used in both regression and classification problems, providing useful insights while avoiding the bias and proneness to errors of humans. In this paper, a specific kind of decision tree algorithm, called conditional inference tree, is used to extract relevant knowledge from data that pertains to electrical motors. The model is chosen due to its flexibility, strong statistical foundation, as well as great capabilities to generalize and cope with problems in the data. The obtained knowledge is organized in a structured way and then analyzed in the context of health condition monitoring. The final results illustrate how the approach can be used to gain insight into the system and present the results in an understandable, user-friendly manner.

Keywords: Decision tree, Conditional inference tree, Health condition monitoring, Machine learning, Knowledge extraction

1. Introduction

As technologies evolve, the amount of data generated and cataloged by monitoring systems increases. In many cases there is, hopefully, information contained in the data that can provide useful insight into the processes that generated it, so that a structured analysis can lead to the extraction and exploitation of specific knowledge. Learning from data is a diverse process, but when the quantity of available data is very large, it is necessary to use computational tools to be able to efficiently process it. This is where machine learning comes into play.

The question of what exactly constitutes a machine learning approach cannot be answered exactly. Learning, by itself, usually entails the acquisition of new knowledge or the refinement of an existing task, organizing and representing the results in an effective way. Machine learning, then, attempts to implant these capabilities into computers (Michalski et al., 2013).

Machine learning can be seen from several points of view. At the core, Michalski et al. (2013) differentiates between three research foci depending on the objective: task-oriented studies, cognitive simulation, and theoretical analysis. These are not mutually exclusive, and in most cases they complement each other. Subsequently, machine learning systems could be further classified on three bases: the underlying learning strategy, the representation of knowledge discovered, and the application domain. In the end, though, the goal is almost always the same: it is desired to develop a concise approach that will perform a certain task with consistent performance and that will be able to match or exceed human cognition. This usually has the additional advantage of being explicit and, to some extent, objective, which can then lead to an automated or semi-automated procedure that can be used in general situations.

The applications of machine learning are many. It is common to perform classification or regression in order to build models based on certain data and then apply the obtained models to new information to calculate predictions. Depending on the goal and scope of the task, classification or regression can be an intermediary step in the overall process of knowledge extraction.

Mazloumi et al. (2011) used neural networks to learn from travel time data in order to build prediction models, but then computed uncertainty in the model taking into account different sources of error, so that a prediction interval was obtained instead of a single point estimate.

Varga et al. (2009) applied decision trees to reactor run-

[☆]This document is the preprint version of the article accepted by the Journal of Engineering Applications of Artificial Intelligence: `http://dx.doi.org/10.1016/j.engappai.2017.03.008`

Email addresses: `alexis.sarda@gmail.com` (Alexis Sardá-Espinosa), `subanatarajan.subbiah@de.abb.com` (Subanatarajan Subbiah), `thomas.bartz-beielstein@th-koeln.de` (Thomas Bartz-Beielstein)

away data to organize and summarize the most important information from the process and then display it in a user-friendly format in an operator support system. These are good examples of how a relatively general machine learning algorithm can be fine-tuned to a specific application, so that relevant data can be extracted in their respective contexts.

In Gerdes (2014) the author proposed an approach based on decision trees to monitor aircraft systems and to forecast time series data of the condition monitoring system to reduce the number of unscheduled downtimes of the aircraft. The author also proposed embedding a genetic algorithm based optimization approach to improve performance of the decision trees thus making it in an automated manner.

Similar to the application case considered in this paper, in Marton et al. (2013) the authors considered the application case of generators using a data-driven approach. In contrast to decision trees, the authors used the data collected from a SCADA system to build a prediction model based on neural networks, principal component analysis, and partial least squares. The prediction performance of the model was measured using mean absolute percentage error, and the predicted data is revised to detect any abnormal behavior and possible degradation of the equipment to generate alarms.

Another contribution for motor diagnostics based on decision trees was proposed in Yang et al. (2000). The authors proposed a decision tree model based on the vibration analysis of motor to diagnose its status. The abnormal frequency components were classified and the causes for vibration were identified.

This work presents an approach to extract and exploit knowledge in an automated way by means of a decision tree algorithm. Section 2 describes the problem that was to be solved and its context. The foundations for the machine learning algorithm that was used, as well as a brief description of the algorithm itself, are presented in Section 3. Section 4 explains how the data was preprocessed in order to feed it to the algorithm, and how the cross-validation procedure was carried out. The analysis and interpretation of the obtained results are given in Section 5, and the final conclusions are summarized in Section 6.

2. Problem Definition

The data that will be considered pertains to a condition monitoring system which evaluates motors and generators. Based on voltage, current, and vibration measurements, a series of indicators that represent health condition of the motors is computed automatically and summarized in a report that is then given to the customers. In order to maintain confidentiality, the data will be anonymized throughout this report.

As with most condition monitoring systems, the main goal is to detect and quantify variation in the asset's health compared to its normal operating condition, which could lead to reduced performance or maintenance actions. In order to standardize the results, specific labels that reflect health status based on the measured parameters are assigned.

The workflow that concludes in the final report is partly interactive. The raw measurement files and the motor information must be loaded manually and an analyst must overlook the whole process to make sure that the output makes sense. Once all parameters have been obtained, the analyst must perform validation of the report and put forward, if necessary, a set of suggestions for the customer. This can be a time-consuming task, especially if the user is not acquainted with the different algorithms that are already being used.

The structure of the reports and the organization of the data therein suggests the possibility of using machine learning algorithms to extract meaningful information. Since the health labels take on a specific set of values, the task can be considered as a classification problem, although the main interest is to use the underlying models generated to assess the report generation workflow with respect to the results, and to try to identify areas of opportunity that can be improved in the future.

Eventually, it is desirable to analyze new data not only by itself, but also with respect to the knowledge that has already been acquired over the years from the whole fleet of devices currently in operation. This could help identify possible inconsistencies, such as input errors, illogical or non-standard results, among others. Additionally, it might be possible to simplify the validation task for the analyst so that time requirements are reduced and the explicitness of the reports increases. This last point is particularly important due to the fact that users are more prone to errors and biases, which could make it difficult to reproduce past results.

The overall approach is described in the diagram of Fig. 1. The data will consist of several input variables (sometimes also called attributes or covariates in the literature) as well as the output labels. Some important aspects of preprocessing will be considered due to possible correlation and missing values. The models will be obtained by using decision tree models, whose parameters will be tuned by using bootstrap cross-validation and 0.632+ error estimates. Finally, the evaluation of the models in the context of the data will be performed. The specifics of each step of the process will be explained in more detail in the following sections.

3. Decision Trees

3.1. Decision Trees

More than a specific algorithm, decision trees are a framework to create an explicit hierarchy of tests that result in a partitioning of the decision space. The general procedure allows for several strategies to be used at each step, which in turn results in a wide variety of tree models being available. They have very good comprehensibility and serve as the basis of most of the algorithms that will be used, so they will be briefly described following Kotsiantis (2007).

Decision trees form a structure made by nodes and branches, starting at a single root node and ending in the terminal nodes, also called the leaves of the tree. At each node, usually a single variable is considered, and one or more thresholds

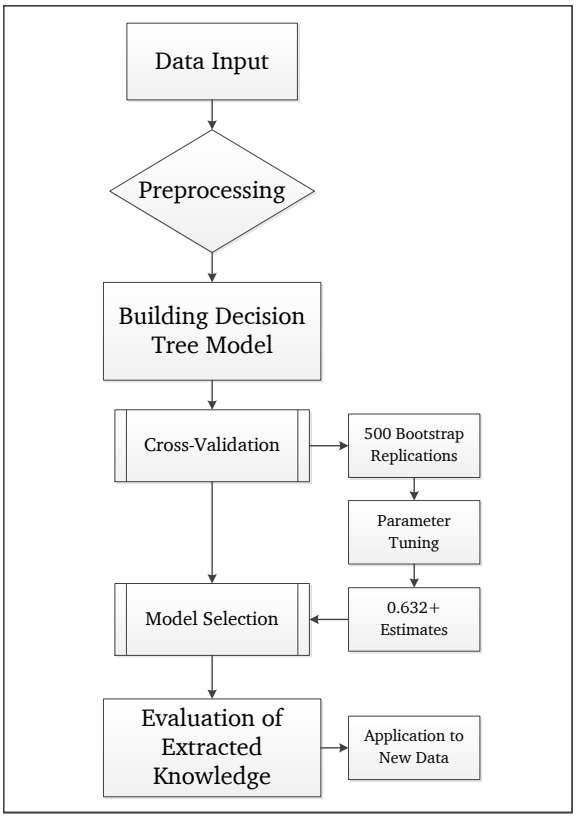

Figure 1: Diagram showing the different steps of the utilized approach.

are chosen by using a given measure of split quality or node impurity. These thresholds, depicted by the branches of the tree, divide the decision space for the considered variable. An instance can thus be classified by starting at the root node, analyzing the specified variable, following the appropriate branch and recursively repeating until a leaf is reached.

The general procedure to build decision trees is based on a *recursive partitioning procedure*, which can be expressed as follows: considering a specific dataset with m variables, the one that best splits the decision space with respect to a given measure is denoted with m^*. The root node is created by considering m^* and, assuming c is the threshold that achieves the best split, two branches are created: one where $m^* \leq c$ and another one for $m^* > c$. This must be recursively repeated on the sub-lists at each node until a stopping criterion is satisfied.

Note that the recursive partitioning procedure should be *unbiased*, i.e., under the assumption of independence of the response Y and the input variables $X_i, i = 1, \ldots, m$, the probability of selecting variable X_j is $1/m$ for all $j = 1, \ldots, m$ regardless of the measurement scales or number of missing values (Hothorn et al., 2006).

Depending on the methodology used at each step of the algorithm, and the selected criteria, a very different tree model can be obtained. As such, many variations of the general procedure were proposed, including versions with multiple splitting and regression trees. However, these procedures suffer from selection bias towards variables with many possible splits or with many missing values, and overfitting.

To avoid overfitting, several algorithms implement a pruning strategy after the tree is fully grown. Conditional inference trees go one step further by implementing a unified framework for handling both selection bias and overfitting.

3.2. Conditional Inference Trees

A conditional inference tree is one possible decision tree algorithm for recursive binary splitting, which tries to embed the framework in a well-defined statistical environment based on permutation tests, attempting to distinguish between significant and insignificant improvements.

As an improvement of the recursive partitioning procedure described in Section 3.1, conditional inference trees separate the variable selection from the splitting procedure. This results in basically three steps in the conditional inference tree procedure. The first one concerns variable selection, the second one chooses the splitting methodology, and the last one is the recursive application of the first two steps. The reader is referred to Hothorn et al. (2006) for a detailed descriptions of these steps. A vignette with more information and several practical examples is also available for the corresponding software package[1].

In addition to their basic capabilities of avoiding bias and overfitting, conditional inference trees possess other useful characteristics.

- Their maximum depth or the minimum amount of observations allowed at each node of the tree can be limited in order to prevent pathological splits.
- Conditional inference trees can deal with missing values on a split-by-split basis by setting weights to zero if a given variable from a considered observation is missing.
- They can be used for a broad variety of variables, e.g., nominal, ordinal, and multivariate response variables.
- Alternative modeling approaches such as neural networks or support vector machines have excellent prediction capabilities, but do not provide any insight into the underlying problem. Conditional inference trees can be used as tools for prediction and understanding.
- Conditional inference trees are better suited for diagnostic purposes than the standard (exhaustive) recursive partition procedures implemented in Classification and Regression Trees (CART).
- They use well-known, established statistical concepts for variable selection and stopping. The resulting tree models are easier to communicate to practitioners.

Because of these properties, conditional inference trees were chosen as the modeling tool for the analysis of health condition from motors and generators.

[1] `https://cran.r-project.org/web/packages/party/vignettes/party.pdf`

4. Methodology

4.1. Preprocessing

The condition monitoring reports are a way of structuring and summarizing the set of values that are computed for each motor after the raw electrical signals have been analyzed. Each report is organized into four main sections that are of interest to the analyst. They will be referred to as sections A, B, C and D, respectively. Each of these sections has a status label associated with it, which can take on three possible values: KR (Keep Running) < WW (Wait & Watch) < SI (Stop & Inspect); the order shown is increasing with respect to degradation level.

In many situations it is advantageous to transform the data in order to accentuate or attenuate certain characteristics (Cox and Jones, 1981). For example, some transformations like the logarithmic transform can help with regression when variability of a variable is not constant between different sub-populations. In other cases, data must be transformed so that the machine learning algorithms are capable of actually processing it. This may be necessary if there are missing values, or if dimensionality is so high that the problem becomes intractable. The scale of each feature is also relevant, since there are some algorithms that would naturally give more or less weight to features whose values are numerically larger or smaller.

There are several procedures that can be applied in order to preprocess data. What to use depends greatly on the algorithm to be used later for analysis, and also on the nature of the data. The algorithm under consideration is capable of dealing with variables measured with arbitrary scales, so normalization is not necessary in this case. Additionally, the algorithm is capable of dealing with missing values, so they do not need to be removed or imputed with a different procedure. Nevertheless, some of the variables present so many missing values that it is probably detrimental to keep them in the dataset. As such, if more than 75% of the observations have a variable missing, that variable will be removed.

There are advanced methods for dimensionality reduction, such as principal component analysis, that can reduce the number of variables by performing an orthogonal transformation on the data and keeping only those dimensions with the most information. Unfortunately, doing such a transformation modifies the decision space, and in our application it is critical to maintain as much interpretability as possible. Therefore, a simpler methodology employing the linear correlation between the variables of each report section will be used. Variables which are correlated to any other by 0.85 or more (absolute value), using Pearson's correlation coefficient and ignoring missing values on a pairwise basis, will be discarded.

4.2. Cross-Validation

There is no single method that can be optimally applied to every scenario. Therefore, it is necessary to measure performance in such a way that allows the evaluation of different algorithms as well as their sensitivity to their tuning parameters. Common measures include accuracy, speed, comprehensibility or interpretability, and time required to learn. It follows that the decision on which combination of algorithm and performance measure offers the best results heavily depends on the task at hand.

It is possible to utilize naive approaches as baselines, in order to ensure that the algorithms are indeed resulting in a measurable improvement. The simplest baseline is the no-data rule, which always assigns a given class regardless of the input values, and might be actually used if the cost of acquiring data is too high. Another approach is to always predict the most common class, which takes into consideration the prior probabilities given observed the data. In addition, costs or weights can be assigned to either classification or misclassification, in order to induce a higher priority on certain classes.

There are many problems to be taken into account when training machine learning algorithms. Once a performance metric has been chosen, knowing which algorithm, along with which combination of tuning parameters, yields the optimum results with respect to the metric is one of the main interests. Obtaining an estimate for the algorithm's general performance is not simple, the datasets used for training are always finite, so a way must be found to effectively use them in order to calculate estimates that have, ideally, low bias, low variance and that are not a result of overfitting.

It is well known that testing an algorithm with the same dataset with which it was trained leads to overly optimistic estimates (Arlot et al., 2010), which can be more or less biased depending on the learning procedure itself. Closely related to that is the fact that some algorithms can learn in such a way that they perfectly fit the training data but are not able to generalize to new observations, which is the problem of overfitting. Moreover, it is not uncommon for there to be one or more tuning parameters which can change the outcome depending on the algorithm's sensitivity. algorithm was used in conjunction with the data The set of procedures that are used to overcome these problems are called cross-validation, and they serve multiple purposes. First of all, they attempt to use the available data as efficiently as possible in order to yield valid estimates of performance. Furthermore, they also help evaluate one specific algorithm with different tuning parameters, as well as variability of the results when parameters are fixed but the data change. Even though they cannot alleviate overfit directly, they provide tools to identify it, so that measures can be taken in order to reduce or negate the effects. The approach presented here will focus on using bootstrap cross-validation.

The generic bootstrap methodology explained in Efron and Tibshirani (1993) is a computer approach that has many applications. In general, if there is a dataset with n observations, several new datasets are constructed by sampling *with replacement* from said dataset until n observations have been selected; this is repeated B times and these new datasets constitute the bootstrap samples.

In the nonparametric bootstrap, sampling is performed

based on a uniform distribution that places a probability of $1/n$ on each observation. This, along with the fact that sampling is done with replacement, means that the probability that a bootstrap samples does not contain an observation is $(1-1/n)^n \approx e^{-1} \approx 0.368$. Thus, on average, the number of observations in each bootstrap sample is $0.632n$.

In the simplest case, using bootstrap for cross-validation consists in training the algorithms with the bootstrap samples, then evaluating them with those observations that were not part of the sample and finally averaging all the bootstrap estimates.

A variation that is somewhat specific to classification tasks that use accuracy as metric is the so-called 0.632 estimator proposed in Efron (1983). Given a total of B bootstrap samples where ϵ_i is the error estimate for sample i and ϵ_0 is the error on the full training set (also called the apparent error), the 0.632 accuracy error can be calculated with Eq. (1). Note that the estimate is defined in terms of the error rate, which for classification tasks is simply 1 minus the accuracy.

$$\epsilon_{\text{boot632}} = \frac{1}{B}\sum_{i=1}^{B}(0.632 \cdot \epsilon_i + 0.368 \cdot \epsilon_0) \tag{1}$$

The 0.632 estimator has some shortcomings. It can fail if the classifier is a perfect memorizer or the dataset is completely random, where there is no relationship between outcome and predictors (Kohavi (1995)). In order to overcome these problems, the 0.632+ estimator was later introduced in Efron and Tibshirani (1997). It was intended to be a less biased compromise that depends on the amount of overfitting.

To compute it, first a no-information rate ξ must be estimated by permuting responses and predictors. Let $\delta_{i,j}$ denote the discrepancy between observation i and prediction j, then the estimate is given by Eq. (2a).

For multicategory classification, let $\hat{p}_l$ be the proportion of observed responses equal to level l and $\hat{q}_l$ be the corresponding proportion of predictions equal to l. Then, the no-information rate can be estimated with Eq. (2b).

$$\hat{\xi} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\delta_{i,j} \tag{2a}$$

$$\hat{\xi} = \sum_{l}\hat{p}_l(1-\hat{q}_l) \tag{2b}$$

Afterwards, a relative overfitting rate can be estimated with Eq. (3) and the final 0.632+ estimate is given by Eq. (4), where ϵ is the bootstrap estimate and ϵ' is defined as $\min\left(\epsilon, \hat{\xi}\right)$.

$$\hat{R}' = \begin{cases} (\epsilon - \epsilon_0)/\left(\hat{\xi} - \epsilon_0\right), & \text{if } \epsilon, \hat{\xi} > \epsilon_0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$\epsilon_{\text{boot632+}} = \epsilon_{\text{boot632}} + (\epsilon' - \epsilon)\frac{0.368 \cdot 0.632 \cdot \hat{R}'}{1 - 0.368 \cdot \hat{R}'} \tag{4}$$

4.3. Workflow

The workflow to analyze each report section will be essentially the same. First, each section will be evaluated independent of the remaining ones, and a stratified partitioning to create train and test sets, allocating 85% of the data to the training set, will be utilized. The training set will be further divided into new train and validation sets in concordance with the bootstrap cross-validation strategy. Afterwards, the models will be re-trained by utilizing all available variables, in order to see if the variables from other sections could provide valuable information to the model.

A baseline for each section will be established by employing the naive rule. This will provide a basic point of comparison to know how much of an improvement, if any, is the algorithm providing. Then, the accuracy estimates given by the 0.632+ bootstrap will be used to get an idea of approximate performance. 500 bootstrap samples will be used for each cross-validation run.

It is true that accuracy is not the best metric to assess classification performance, especially if the dataset is unbalanced. However, the current focus is doing data exploration by means of a machine learning algorithm to extract knowledge that can be useful in future analyses, so the interpretation of the results will be of greater importance. For this purpose, accuracy should yield satisfactory results.

All experiments and analyses were performed using the **R** programming language (R Core Team, 2016; RStudio Team, 2015) by leveraging the `caret` package for model training and validation (Kuhn, 2008). These are all open source software packages that support most operating systems and are freely available.

5. Experimental Results

5.1. Sample Training - Section A of the Reports

For section A of the reports, using the naive rule would result in an estimate of accuracy equal to 0.514. In the following, the improvements provided by the machine learning algorithm will be assessed.

Technically speaking, conditional inference trees can be tuned by modifying the minimum criterion (α), although there are other parameters that can also be controlled, such as the tree depth or the amount of observations allowed at each terminal node, also called the bucket size. The first step taken was to assess the influence of α by testing the common values of 0.9, 0.95 and 0.99, enforcing no restriction on the tree's depth. The obtained average accuracy and its standard deviation (SD) is reported in Table 1.

Judging by the cross-validation results, it can seen that α had virtually no influence on the overall algorithm's accuracy for this dataset. Having established the previous, the effects of the tree's depth on accuracy can also be evaluated. The previous tree had a depth of 3, so we can simply test values from 1 to 3 while keeping α constant at 0.99. The results for this run are shown in Table 2, where it is seen that limiting the tree depth to 2 marginally improves accuracy. Note that

Table 1: Train results for conditional inference trees in section A using minimum criterion as tuning parameter. A total of 500 bootstrap samples were used for cross-validation.

Min. Criterion (α)	Accuracy	Accuracy SD
0.900	0.758	0.030
0.950	0.759	0.030
0.990	0.760	0.029

a smaller depth implies that less splits are made throughout the tree, which means that, potentially, less variables would be needed in the final model.

Table 2: Train results for conditional inference trees in section A using maximum depth as tuning parameter.

Max. Depth	Accuracy	Accuracy SD
1	0.749	0.024
2	0.762	0.026
3	0.761	0.026

An added advantage of tree models is their ease of visualization and interpretation. For example, the tree model obtained in the second cross-validation run is shown in Fig. 2. Each node of the tree is depicted by a circle with the variable used for splitting and its associated p-value (see Section 3.2). The terminal nodes, also called the leaves, show a barplot of the output label distribution considering only the observations at each respective leaf, and denote with n the number of observations that were assigned to that leaf.

The tree in Fig. 2 provides a lot of information. First of all, it implies which predictors are the most relevant with respect to the output variable. The training set contains 20 predictors, so the fact that only two of them can provide an accuracy of 0.762 is noteworthy.

On the other hand, the barplots at the leaves depict the consistency of the results. The output given by the tree itself is a class label, but by analyzing the observations at the leaves, the posterior probabilities of each class label can be evaluated, conditioned on the splits given by the tree. For instance, the third barplot in Fig. 2 implies that, if an observation is assigned there, it would be extremely rare for it to have a class different than WW, whereas the last barplot implies that any observation assigned there would never have a KR class (given the data).

When doing classification, a popular method of displaying the results is by means of a confusion matrix. This matrix has a number of rows and columns equal to the number of levels in the output class, and each cell shows the correspondence between observed and predicted labels. During cross-validation, predictions are obtained at every step in order to evaluate model performance, but only considering the observations in the validation sets. Nonetheless, a confusion matrix with the average correspondence of each cell across the 500 replications can be constructed. This matrix is shown in Table 3, where the cell averages are expressed as percentage values of the total cell counts.

Table 3: Confusion matrix for section A using conditional inference trees. Each cell shows the average correspondence between predicted and observed values across the 500 replications performed during CV, but expressed as a percentage of the total counts.

	Reference		
Prediction	KR	WW	SI
KR	49.5	12	1.11
WW	1.86	18	3.09
SI	0.0386	6.75	7.61

It is evident the model is good at predicting the KR cases, which is expected for two reasons. On the one hand, the KR level was the most common one in the dataset, and on the other hand, from a practical point of view, it makes sense that discerning SI motors from the WW ones gets more difficult as the level of degradation increases.

So far the analysis has looked at section A independently from the other sections of the report. Additionally, some of the variables that were in the raw dataset were removed during cleaning on account of their correlation to other variables. By re-training the models with the ignored features, it can be checked whether the cleaning step was justified and if some of the variables from other sections can help with classification. The value of α was kept at 0.99, but maximum depth values from 0 to 8, where 0 signifies no restrictions, were tested. Also note that this is only a limitation of maximum depth, meaning that the algorithm can still decide to stop at a smaller depth if the conditions are satisfied (see Section 3.2).

The best model using all variables resulted, in an accuracy estimate of 0.758, which is essentially the same value obtained without the extra features. Interestingly, by inspecting the model obtained after using all variables, it was discovered that it was the exact same model obtained when using only the cleaned dataset from section A. This provides reassurance that the variables from other sections have no significant influence in the status labels assigned here.

As the final step in the learning process, the model from Fig. 2 can be used to classify the data that was left in the test set, which has been ignored so far. This will provide one final estimate of future performance of the specific algorithm that was selected.

The confusion matrix for the results with the final model and the test set is shown in Table 4. The results translate to an accuracy estimate of 0.753, which is very close to what was expected.

5.2. Overall Results

The experimental methodology followed for the remaining section was the same. The naive estimate was computed for each section of the report, since the proportion of the class labels changed. Different conditional inference trees were built by tuning the algorithm parameters and using only the respective section data. Then the models were trained again

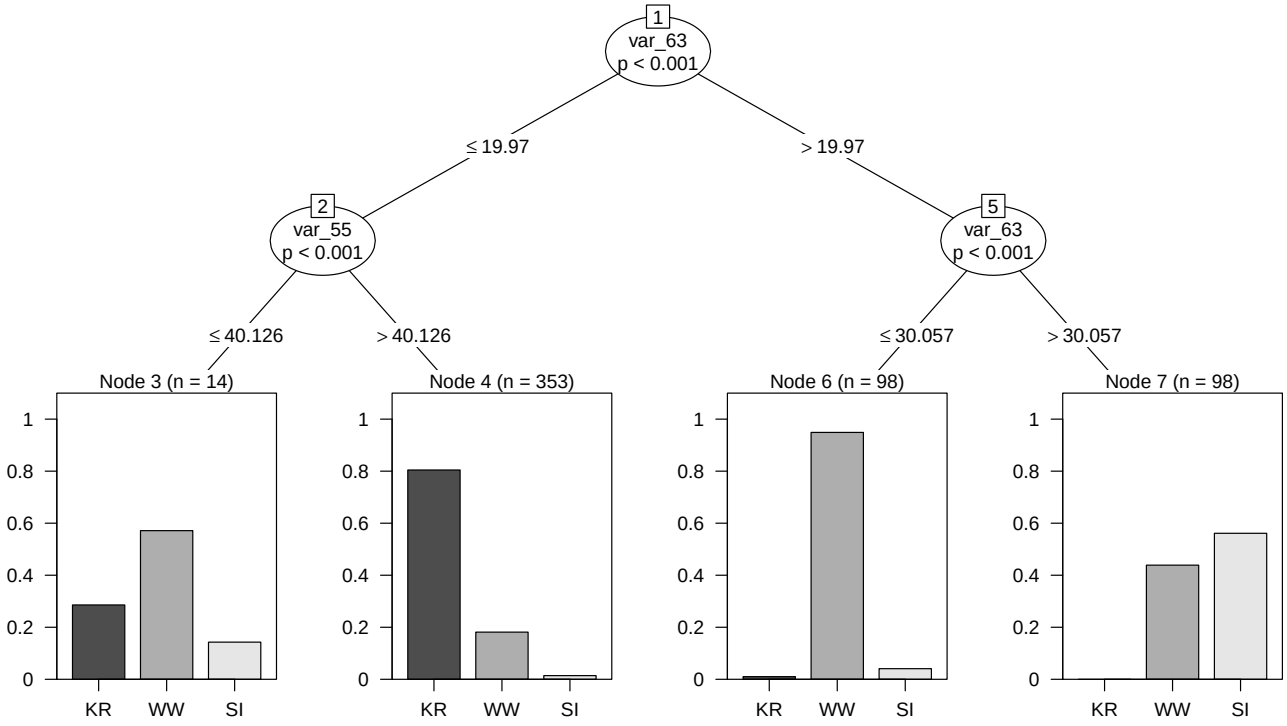

Figure 2: Visualization of the conditional inference tree for section A. Each oval contains a specific variable. Following the branches leads to specific binary partitions for the variables based on the shown threshold. The value of n at the leaves represents the total number of observations that fall in that terminal node.

Table 4: Confusion matrix for test data in section A. The final conditional inference tree was used. Each cell shows the raw correspondence counts between observed and predicted values.

	Reference		
Prediction	KR	WW	SI
KR	47	10	2
WW	2	17	0
SI	1	9	9

Table 5: Results after applying the machine learning workflow to all sections of the reports.

Section	Naive rule	Model accuracy	Accuracy with all variables	Test set accuracy
A	0.514	0.762	0.758	0.753
B	0.736	0.865	0.851	0.859
C	0.525	0.840	0.830	0.847
D	0.492	0.669	0.669	0.656

with all available variables to assess variations in the outcome. Once the final model was established, it was applied to the test set of the corresponding section to get one final estimate of performance. The summary of the results is shown in Table 5, including the previous results of section A for completeness. The graphical depiction of the trees for each section can be seen in Appendix A.

It is interesting to see that in most cases, including additional variables into the training procedure was actually detrimental for average accuracy, albeit slightly. This also means that the machine learning algorithm is good at dealing with irrelevant or redundant information contained in the data.

5.3. Evaluation

Now that there is an idea of the performance that could be expected from the conditional inference trees, the specific details that relate to the quality and the interpretation of the underlying mechanisms at play can be outlined. These interpretations will be focused on the considered dataset, and will reflect some insight that can only be obtained after careful analysis of all variables and their meaning.

In an ideal scenario, the variables in the data would contain all the necessary information regarding the motor's health, and a machine learning algorithm would be able to extract it and attain perfect accuracy on both known and future data; in reality, this is rarely the case. Some plausible reasons for this could be data input errors, some form of noise, inconsistent processing algorithms, human bias, etc. The accuracy estimates that were obtained in the previous sections provide clues about the data quality, or lack thereof: if accuracy is low, there is clearly some information missing, or the models were not able to uncover it. It is desired to provide possi-

ble directions which the data experts can take to find out the reason for the discrepancies, if they do not already know it. Otherwise, heir feedback could be used to improve on the models.

5.3.1. Evaluation of Section A

First of all, the tree in Fig. 2 for section A can be evaluated. This model is a nice starting point to show how the results of machine learning can be used. The tree itself is completely objective, it simply tries to organize the knowledge given by the data, so it cannot automatically decide whether some values are illogical, making them outliers. Nodes 4 and 6 show what would be desired: a large number of observations with the same class and very few (ideally none) of the other classes; this would mean that the cases are perfectly discernible by the given splits. On the other hand, nodes 3 and 7 show relatively more problems: the class distribution of the observations falling in those leaves are more balanced, so the boundaries between classes are not well defined. If only the accuracy and the confusion matrix of the model are analyzed, these discrepancies would go undiscovered. By delving into the internals of the model, more information and possibly errors and glitches in the actual data can be extracted.

The accuracy of the model was not high enough to warrant its use in an automated way, but is still considerably high even though it only used two variables, and it also helps gain insight into the data. It can be argued that `var_63` is the most important variable for section A, and that if its value is above 19.97, it would be extremely rare for a motor to have a KR label. It can be seen, however, that there is indeed at least one KR observation with a high value of `var_63` (in node 6), which begs the question: why is it that that motor has a KR label despite having such a high value of `var_63`?

Similarly, some guidelines for the evaluation of the reports suggest that if `var_55` is above a value of 54, the motor should be in good condition. Node 4 of that tree certainly has more KR cases, but the threshold chosen by the tree is different, and there are still many WW and SI observations in that node.

5.3.2. Evaluation of Section B

Naturally, this same type of evaluation can be applied to the models that were obtained for the other sections. The next tree would be the one in Fig. A.1 for section B. It is particularly interesting because it is remarkably simple, using only 5 predictors even though section B originally had more than 50. Its accuracy was also better. By itself, this already poses an important question: if good accuracy can be obtained with few predictors, what is the purpose of the other variables? Should they be calculated at all? It could be that they are required for intermediate calculations, so that the more relevant values can be obtained, but maybe it is not necessary to include them in the reports.

The variables that were chosen for the model appear to agree with what was expected. There are limits for variables `var_38` and `var_40`, given by specific standards, so it would be good to see if the splitting thresholds are in line with said standards.

Based on knowledge from the data, it was expected that any motor where `var_54` is larger than zero could never have a KR label, but there are a few cases where this is not true. By looking at the few KR cases falling on node 12, something worth noting was uncovered: some of those reports appear to have manual corrections made to them, whereby the values changed visually, but the numeric elements stayed the same. Due to the automatic parsing of the reports, such corrections cannot be detected automatically, and are a critical source of inconsistencies. Moreover, there is important knowledge that could be extracted here (and, ideally, standardized): if corrections were made, what was the reason for it? These reasons could be incorporated into the models, and they could also lead to improvements of the processing algorithms that use the initial raw data.

Predictor `var_2` is a categorical feature that specifies one of the inherent characteristics of the motor, so it is unexpected to see that it was actually used as a splitting variable. Due to the way it is shown in node 9 in Fig. A.1, it should be interpreted like this: any motor whose `var_2` is *not* `variable`, and whose `var_38` is larger than 5.4, has a higher chance of having a class different than KR.

5.3.3. Evaluation of Section C

Moving on to section C, the largest model was obtained, which is shown in Fig. A.2. It used almost all the available variables for that section and obtained decent accuracy, but it should still be improved. There is a good reason for the model to be so large: the variables in this section are for similar mechanical elements of the motor that are mounted in different places, and health status probably depends on the values of all the elements, so the model has to take into account all possibilities.

The appearance of `var_1` in nodes 25 and 10 seems rather redundant, because most observations at the left and right leaves are mostly WW. By contrast, using `var_2` in node 5 could have other implications with respect to `var_91`: it could be that `var_91` is more important if the motor's `var_2` is equal to `variable`.

Evidently, many of the variables are independent from each other: the right branches of nodes 2 through 7, except for node 5, have no further splits, and judging by the barplots at the leaves, it is clear that if any of those variables exceed their corresponding thresholds, there is a large probability that their labels will be WW.

Nodes 8 and 9 used variables that are equivalent (as noted before, they are calculated for the same mechanical elements, just mounted in different places of the device). However, it seems that the difference for `var_84` is not as obvious as the one for `var_76`; this only happens rarely (there are only 7 observations at node 14), but it would be better to define a clear threshold so that future analyses are more consistent.

It is also worth mentioning that, since section C had very few SI cases, it could be argued that the mechanical elements analyzed therein rarely fail, or it could also be that they are

changed more often by preventive maintenance actions. If that is the case, there is an area of opportunity that may be exploited here: discussing the few SI cases with the experts might shed light into how they were identified as such, and those rules could be incorporated into the classification algorithms.

5.3.4. Evaluation of Section D

Lastly is the tree for section D depicted in Fig. A.3. This model was the one with the worst performance, something that readily gives some information: there were many variables available for this section, but it seems that they are not able to capture the most important elements of variability. On the other hand, this could also mean that the differences between KR, WW and SI cases are not clearly defined, so the analysts may have a hard time identifying each new case.

There were 3 additional categorical variables in this section: `var_107`, `var_108` and `var_109`. A value of `yes` in any of them should mean that there are problems with the motor, but the model of Fig. A.3 tells a different story. Not only did `var_109` not appear in the model, following the right branches of nodes 2 and 5 shows that there were a considerable amount of observations with a KR label in the corresponding leaves. By looking at some of the corresponding reports, it was found out that some of them also presented manual corrections that change visually but not numerically. The reason for this inconsistency should definitely be clarified so that it is certain that the calculations that lead to the values reflected in those variables are not flawed.

Inconsistencies notwithstanding, it could still be argued that `var_106` and `var_102` are the most important for section D, and that higher values mean higher level of degradation. This behaviour was expected, but it is still advisable that the analysts establish clear boundaries for the values of those variables if there are currently none, in order to ensure consistency.

The fleet data provided is just one example of a monitoring system that collects large quantities of data. The gains that can be obtained by exploiting it are not always immediate, and the refinement process will need the collaboration of all parties involved. With this in mind, it is clear that, if used appropriately, machine learning can be a valuable set of tools with many applications.

6. Conclusions

In the present context, the data come from a fleet of motors whose health condition is monitored frequently, but in practical terms, the general methodology presented in this work can be applied to many other situations.

The number of available machine learning algorithms is staggering, and the line that separates them from other methods, such as statistical approaches and neural networks, is often blurry. Clearly, the first choice to be made is which algorithms are going to be tested, something that depends heavily on the scope, and is not always straightforward since no single algorithm can perform optimally in every situation.

Due to the nature of the data, classification algorithms were a natural choice. However, the main interest was obtaining deeper and more intuitive knowledge out of the analysis, so greater importance was given to the interpretability of the models. In this relatively early stage of the project, where understanding the data is critical, these models proved to be very helpful, being a very valuable tool. Nevertheless, as the project evolves, there will certainly be a need for refinement as new ways to automate the workflow as much as possible are explored, and more elaborate algorithms could be considered, such as neural networks and support vector machines.

In order to utilize the data in the best way possible, cross-validation was fundamental to the evaluation of the variability of the machine learning algorithm not only with respect to the data but also with respect to its tuning parameters.

The assessment of data quality and subsequent cleaning entails a series of preprocessing and, possibly, anonymization steps that can sometimes have the largest impact on the final results. After all, if the tidiness of the input data is lacking, not even the most powerful algorithm will deliver proper results. Unfortunately, there are also many options to choose from in order to obtain a tidy dataset, and it is not easy to say which ones are the most appropriate ones a priori. The cleaning that was performed, which depended mostly on linear correlation, was easy to justify by training with both datasets (the original and the tidy ones) and identifying the differences in results, or lack thereof. Commonly, this is not possible, since cleaning is often done in order to reduce dimensionality and carry out an analysis that would otherwise be unfeasible.

The experimental analysis marked the culmination of the analysis. Employing all of the aforementioned techniques on a real dataset showed how every element comes together and interacts with each other, emphasizing the utility of machine learning from a practical point of view. Conditional inference trees proved to be a robust algorithm, capable of effectively handling missing values and maintaining its results even in the presence of redundant or irrelevant data. They had no problem whatsoever when dealing with numeric and categorical variables, even if they were measured at different scales. Additionally, their visualization techniques are very intuitive, which makes their evaluation and communication of their results much easier.

Careful interpretation of the results led to finding glitches in the data and gain insight into the process, thus uncovering possible areas of opportunity in the report generation workflow. Even models with relatively poor performance helped by virtue of their inadequate results.

In the future, the classification analysis could also be further improved. For example, the levels of the output classes (KR, WW and SI) had the same weight during the experiments, and this could be modified by assigning different weights to each level, so that the correct classification of a given class has a higher priority with respect to the others. Similarly, other metrics could be evaluated to see if the final models change significantly.

References

Arlot, S., Celisse, A., et al., 2010. A survey of cross-validation procedures for model selection. Statistics surveys 4, 40–79.

Cox, N.J., Jones, K., 1981. Exploratory data analysis. Quantitative Geography, London: Routledge , 135–143.

Efron, B., 1983. Estimating the error rate of a prediction rule: improvement on cross-validation. Journal of the American Statistical Association 78, 316–331.

Efron, B., Tibshirani, R., 1997. Improvements on cross-validation: the 632+ bootstrap method. Journal of the American Statistical Association 92, 548–560.

Efron, B., Tibshirani, R.J., 1993. An introduction to the bootstrap. CRC press.

Gerdes, M., 2014. Predictive health monitoring for aircraft systems using decision trees.

Hothorn, T., Hornik, K., Zeileis, A., 2006. Unbiased recursive partitioning: A conditional inference framework. Journal of Computational and Graphical Statistics 15, 651–674.

Kohavi, R., 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection, in: IJCAI, pp. 1137–1145.

Kotsiantis, S.B., 2007. Supervised machine learning: A review of classification techniques. Informatica 31, 249–268.

Kuhn, M., 2008. Building predictive models in r using the caret package. Journal of Statistical Software 28, 1–26.

Marton, I., Sánchezb, A.I., Carlosa, S., Martorella, S., 2013. Application of data driven methods for condition monitoring maintenance. CHEMICAL ENGINEERING 33, 301–306.

Mazloumi, E., Rose, G., Currie, G., Moridpour, S., 2011. Prediction intervals to account for uncertainties in neural network predictions: Methodology and application in bus travel time prediction. Engineering Applications of Artificial Intelligence 24, 534–542.

Michalski, R.S., Carbonell, J.G., Mitchell, T.M., 2013. Machine Learning: An Artificial Intelligence Approach. Springer Publishing Company, Incorporated. chapter 1.

R Core Team, 2016. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. URL: `https://www.R-project.org/`.

RStudio Team, 2015. RStudio: Integrated Development Environment for R. RStudio, Inc.. Boston, MA. URL: `http://www.rstudio.com/`.

Varga, T., Szeifert, F., Abonyi, J., 2009. Decision tree and first-principles model-based approach for reactor runaway analysis and forecasting. Engineering Applications of Artificial Intelligence 22, 569–578.

Yang, B.S., Park, C.H., Kim, H.J., 2000. An efficient method of vibration diagnostics for rotating machinery using a decision tree. International Journal of Rotating Machinery 6, 19–27.

Appendix A.

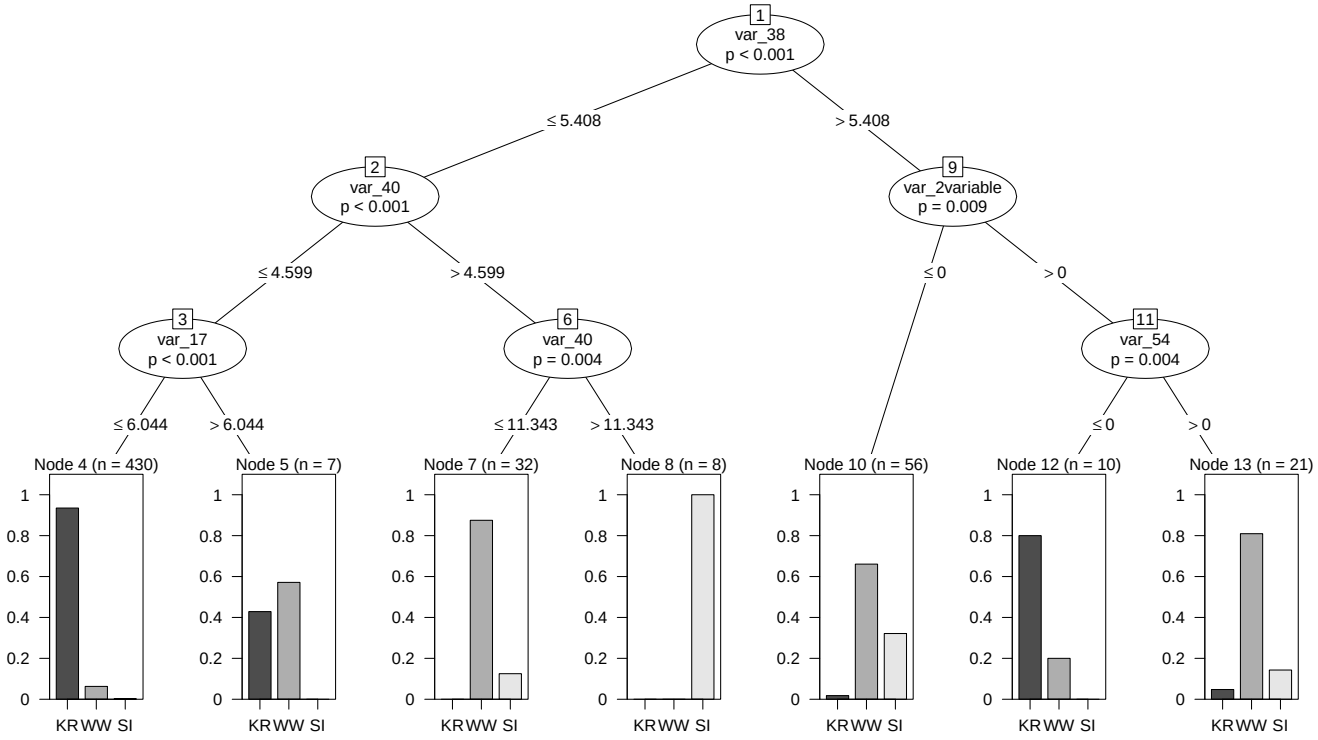

Figure A.1: Visualization of the conditional inference tree for section B of the data reports.

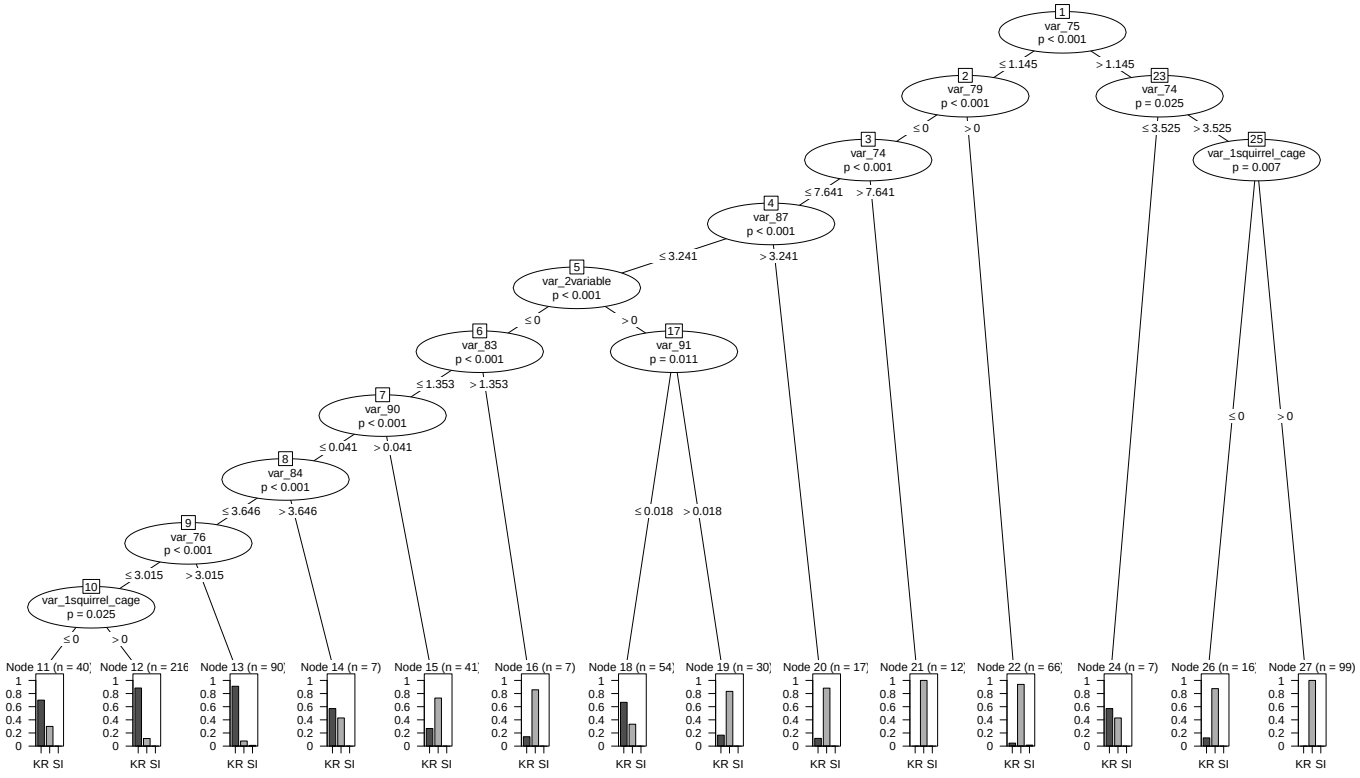

Figure A.2: Visualization of the conditional inference tree for section C of the data reports.

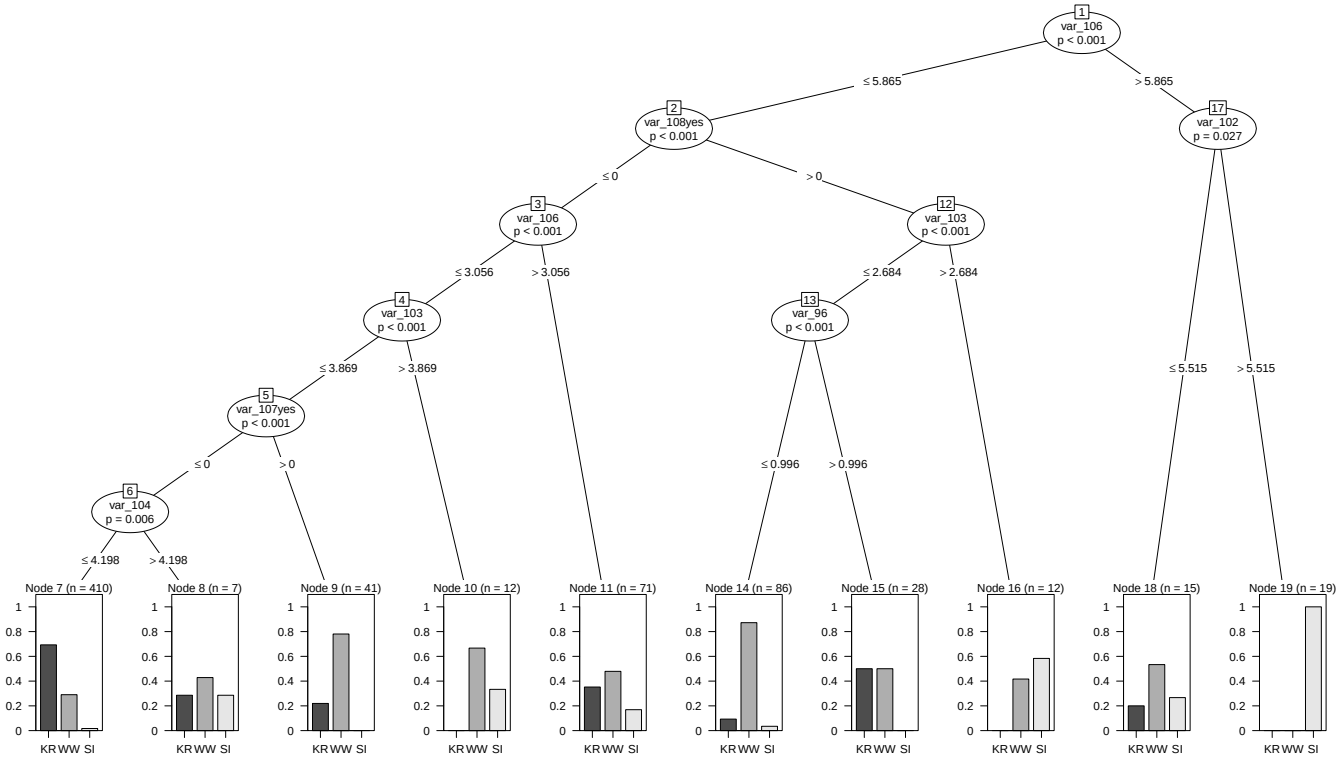

Figure A.3: Visualization of the conditional inference tree for section D of the data reports.

Kontakt/Impressum

Diese Veröffentlichungen erscheinen im Rahmen der Schriftenreihe "CIplus". Alle Veröffentlichungen dieser Reihe können unter
https://cos.bibl.th-koeln.de/home
abgerufen werden.

Köln, Januar 2012

Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.
Datum der Veröffentlichung: 01.04.2017

Herausgeber / Editorship

Prof. Dr. Thomas Bartz-Beielstein,
Prof. Dr. Wolfgang Konen,
Prof. Dr. Boris Naujoks,
Prof. Dr. Horst Stenzel
Institute of Computer Science,
Faculty of Computer Science and Engineering Science,
TH Köln,
Steinmüllerallee 1,
51643 Gummersbach
url: www.ciplus-research.de

Schriftleitung und Ansprechpartner/ Contact editor's office

Prof. Dr. Thomas Bartz-Beielstein,
Institute of Computer Science,
Faculty of Computer Science and Engineering Science,
TH Köln,
Steinmüllerallee 1, 51643 Gummersbach
phone: +49 2261 8196 6391
url: http://www.spotseven.de
eMail: thomas.bartz-beielstein@th-koeln.de

ISSN (online) 2194-2870

Technology
Arts Sciences
TH Köln

Technology
Arts Sciences
TH Köln