# Divide, Propagate and Conquer: Splitting a Complex Diagnosis Problem for Early Detection of Faults in a Manufacturing Production Line

Kerman López de Calle - Etxabe [1], Meritxell Gómez - Omella [2] and Eider Garate - Perez [3]

[1,2,3] *Tekniker, Basque Research and Technology Alliance (BRTA), Iñaki Goeanaga, 5, 20600, Eibar, Spain*
{*kerman.lopezdecalle, meritxell.gomez, eider.garate*}*@tekniker.es*

[2] *Faculty of Informatics, University of the Basque Country (UPV/EHU), Manuel Lardizabal Pasealekua, 20018, Donostia, Spain*

[3] *Faculty of Science and Technology, University of the Basque Country (UPV/EHU), Barrio Sarriena s/n, 48940, Leioa, Spain*

**ABSTRACT**

This work elaborates the procedure followed by the HIRUTEK team to solve the data challenge proposed by the PHM 2021 organisation. This challenge deals with a manufacturing line that continuously tests fuses and suffers from several malfunctions. The solution addresses the diagnosis of the faults; the efficiency of the diagnosis; the identification of the signals related to each fault type; and, the identification of different operation settings that occur during the non-faulty conditions. This problem presents some difficulties that are common to machine fault diagnosis or manufacturing line monitoring; such as the class imbalance; the high amount of missing values; multicollinearity and high dimensionality; and, experimental noise. Additionally, the evaluation criteria presents further challenges such as the consideration of chronology and the detection of operation states (also referred in the literature as context awareness). The consideration of all these factors turns this exercise in a very representative and challenging problem. The solution here proposed, that obtained the highest score in the contest, relies on the combination of decision tree algorithms and a propagation system. The trees provide observation-wise diagnoses while the propagation system deals with chronology by adding a Kalman style filter that updates the probabilities, resulting in a more reliable result.

## 1. PROBLEM DESCRIPTION

### 1.1. Condition monitoring in manufacturing lines

The pursue of a more competitive manufacturing has led the production equipment to be more flexible, sustainable and require of less human supervision for its operation. Recent advances as the inclusion of sensors in the line have allowed the use of operator inputs together with the sensor measurements to determine the state of the manufacturing process (Stavropoulos, Chantzis, Doukas, Papacharalampopoulos, & Chryssolouris, 2013). This state or condition detection that is provided by decision systems allows line operators to take corrective actions which, in turn, improves the responsiveness in terms of the machine downtime reduction. However, as the decisions depend on the data received, providing reliable data is critical in order to optimize the manufacturing system (Assad et al., 2021)

Considering the previous, exploring decision-making algorithms for manufacturing systems at testbed level is of great interest, as it provides a better insight of the kind of problems that can be faced at industrial level and fosters the adoption of reliable decision-making algorithms in real industrial setups.

### 1.2. Experimental rig

The problem proposed by the 6th European Conference of Prognostics and Health Management Society 2021 (PHM) resembles a typical component of a large-scale quality-control pipeline of a production line. The experimental bed, courtesy of the Swiss Centre for Electronics and Microtechnology (CSEM), generates data similar to a real-world industrial manufacturing line.

The line consists of a 4-axis SCARA-robot with a vacuum gripper that picks up fuses from a feeder to a fuse-test-bench. On the test-bench, fuse current conductivity is measured, and, if the conductivity is appropriate, heating is applied to the fuse while a thermal camera measures during the heating process. Once the fuse is tested, it is sent back to the feeder with two conveyor belts.

### 1.3. Data

The testbed had different sensors installed that captured 50 signals measuring different physical properties of the system, such as pressure, vacuum, humidity, etc. Those signals were continuously measured during the experiments. However, some statistical descriptors were computed every 10 seconds instead of storing raw measurements. Those signal features were: counts (*Cnt*), frequency (*Freq*), maximum (*Max*), minimum (*Min*), standard deviation (*Std*), *Trend* and *value*, which were not computed for every signal.

During each experiment, lasting between 1 and 3 hours, disturbances were applied to some of them, simulating 8 different system failure conditions, labelled 2, 3, 4, 5, 7, 9, 11 and 12. With a total of 9 different classes, including the healthy class, labelled 0, in which no disturbances were introduced.

During the challenge the data was provided in two stages. Firstly, 50 healthy experiments and 4 experiments with each of the labels 2, 3, 5, 7 and 9 were released. Algorithm development began with those data. Later, another 20 data sets without failures and 3 of each of the 4, 11 and 12 faults were made available.

### 1.4. Evaluation

The data challenge organising committee proposed the following 4 criteria in order to assess the goodness of each approach to the problem.

- Identification and classification of faults: Determining which experiments are faulty and identifying which type of fault it is. That is, a fault diagnosis system that provides the class when given an experiment. This is tested by providing unseen experiments to the solution and checking how many of them are classified correctly.
- Root cause analysis: The solution had to provide a ranking of the signals in descending order of importance that could be causing each of the 8 failures. The appropriate signal should be among the top four signals, assigning a proportional score to the position of that signal in the ranking.
- Prediction in the shortest time: Algorithms are forced to consider the chronology of the experiments when providing the class. The solution is run twice to validate this assumption. In each run, the time required to reach a definitive diagnosis is provided. Later, this time is used to cut the experiment, repeat the process and ensure, this way, that the diagnostic system is robust (it returns the same diagnostic in the second run).
- System operation parameter identification: Apparently, the test rig can operate with two different operation parameter configurations that provide different sensors readings but are not causing faulty operation of the line. The solution should be able to identify the point where taken in one or the other condition, without having any labelling related to the different conditions as they are all labelled as healthy experiments.

## 2. SOLUTION

### 2.1. Evaluation Related Inference

- The fourth evaluation criterion implied the existence of two data groups within the same label 0 (healthy). Hence, developing algorithms without considering this fact would lead to a worse diagnosis capability. For that reason, instead of considering this aspect as an optional bonus, it was decided to start tackling this point first.
- The challenge had an interesting system that validated the time allegedly spent by the algorithm to reach a solution. For that validation, the experiment was cat following the indicated time and it was fed back to the algorithm, ensuring that the class provided by the algorithm in the second attempt coincided with the one previously obtained. This evaluation system was designed to avoid data leaking from one run to the next, as the algorithm could not know whether it was being ran in the first (with the full experiment) or the second (with the reduced experiment) time. However, it was not perfect, as during the first run the algorithm could see the full experiment.

### 2.2. Issues

On the top of the previous observations, the first exploratory analyses unveiled some additional aspects that needed to be considered to properly tackle the problem:

1. Missing Values: Data values that are not properly stored or are missing can have a significant impact on the analysis and further conclusions. The complete data set contained 10% of missing values not identically distributed by the variables.
2. Class imbalance: Class imbalance occurs in classification predictive modelling when there is an unequal distribution of classes in the training set. Typically this kind of problem hinders the obtaining of reliable diagnosis algorithms since the traditional models and performance metrics (such as accuracy) assume a balanced class distribution. In a first analysis of the available data, clear imbalance was identified as there were only 4 experiments for each 2, 3, 5, 7 and 9 classes; 3 experiments for classes 4, 11 and 12; whereas there were 70 experiments belonging to class 0 (healthy).
3. Multicolinearity: Multicolinearity occurs typically in highly sensed environments where the same signal is described with various statistical descriptors. As different descriptors belonged to the same signal, and some signals were measuring similar effects, many of the resulting variables were highly correlated, which can be very harmful, particularly in linear models.

4. High dimensionality: A total of 248 variables were obtained in each test, with a number of observations ranging from 357 to 1081 per experiment. Comparatively, this amount of variables was high, making us aware of potential downsides of employing distance based techniques (due to the curse of dimensionality) or a high chance of facing overfitting if the algorithms used noisy signals when no relevant signal was found.
5. Bias in experiments/Experimental noise: In relation to the previous point, an additional source of noise in condition monitoring is the experimental noise. For any reason, exact replications of experiments in the same testbeds lead to have different signal values. In that regard, considering each observation as purely independent (as in most machine learning problems) is risky, as the inertia the systems have tends to be used by the algorithms to be capable of detecting the experiment in contrast to generalising the class value. For that reason, using standard randomly created train/test splits needed to be avoided.
6. Chronology in diagnosis: Considering chronology can be beneficial and detrimental at the same time. On the one hand, considering chronology could ease the identification of some faults that were not present on the complete signal due to their intermittent behaviour. On the other, considering chronology needed of tools that were not of-the-shelf, hence requiring to build ad-hoc designed algorithms to benefit from it.

### 2.3. Algorithm Development

Bearing in mind all the previously presented issues, an algorithm was developed by combining the techniques explained throughout this section. On the one hand, the statistical software R was used for the exploratory analyses. The libraries used to preprocess the data were `dplyr` (Wickham, François, Henry, & Müller, 2019), `imputeTS` (Moritz & Bartz-Beielstein, 2017) and `cluster` (Maechler, Rousseeuw, Struyf, Hubert, & Hornik, 2021). On the other hand, the development of the final algorithm was programmed in Python. `sklearn` library was used to train and validate the models and also to perform Principal component Analysis (Pedregosa et al., 2011) and `imblearn` to manage the imbalanced data (Lemaître, Nogueira, & Aridas, 2017).

#### 2.3.1. Missing Value Handling

After a completeness analysis was carried out variable-wise, a relationship was identified between the appearance of missing values in some features. In many cases, when the feature *Cnt* (Counter) of a variable took a value of 0, the *Freq* feature is also 0 and the rest of the features (*Max*, *Min*, *Std*, *Trend*, *value*) did not have any value. For that reason, it was decided to eliminate from the study those variables that contained more than 80% of missing values in some tests. For the remaining variables, in each test the missing values were imputed using the LOCF (Last Observation Carried Forward) (Barnes, Lindborg, & Seaman, 2006) method followed by a backward fill. This way the amount of information brought from "the future" was minimised as most of the values were imputed by the first forward imputation pass.

#### 2.3.2. Validation of the Performance in Classifications

The effectiveness of the classification models was validated by comparing the results of metrics extracted from the confusion matrix. It was decided to use Recall because this metric penalises false positives (Ting, 2010). Therefore, Recall values close to 1 were desired, minimising the cases in which failed experiments were classified as healthy.

#### 2.3.3. Identification of the Two System Parameter Configurations

For the identification of the two parameters sets under the healthy cases CLARA clustering algorithm was used forcing the algorithm to identify two clusters (Kaufman & Rousseeuw, 1986). This algorithm applies the Partition Around Medoids (PAM) in different samples of the dataset to obtain an optimal set of medoids. It was implemented using Manhattan distance for 50 samples with 500 observations each. The high dimensionality of the problem hindered the obtaining clear clusters, as, inside the same experiment, CLARA assigned very different proportions of class values. This was assumed to be incorrect, because according to the problem statement, each experiment could only contain a single set of configuration parameters.

Therefore, the experiments that were clustered clearly (with all the observations belonging to either cluster 0 or 1) were taken and a supervised decision tree was used to identify which rules were critical in their identification. This same model was applied to the rest of experiments (the ambiguous ones in the clustering) and the supervised model proved to be a perfect cluster classifier with only a single feature.

As a consequence, considering the inherent difficulties of training a model that had sub-classes inside a class, an additional label was created. From this point on, experiments with 0 label were split into two different classes according to the results of the decision tree.

#### 2.3.4. Root Cause Analysis

The detection of important signals related to each fault was carried out with a PCA (Principal Component Analysis), as it is well suited for high dimensionality and multicolinearity scenarios. From the whole dataset (with all the available experiments), subsets containing the observations of the healthy class and the observations of each faulty class were created. In each subset, PCA was applied separately for the features coming from each signal, thus, as many PCAs as the number

of signals were computed. For each PCA, the number of components to be kept was optimised according to f1-score and a Classification Tree, in that way, the information of signal (from three to seven features) was condensed in a reduced set of principal components. This procedure was done for each data subset containing the observations of the healthy class and the observations of each faulty class. Once the signals were reduced with PCA, the decreasing impurity of a Classification Tree was used to identify the signals which might have caused the faults. The approach here was validated using LeaveOneGroupOut cross validation for each data subset with the healthy observations and faulty observations, leaving out a different faulty experiment in each iteration. Note that the results obtained in this process were used for the sole purpose of answering the root cause analysis task of the challenge. The knowledge gathered at this stage was used only as a notion of which features could be of interest for the diagnosis algorithm, as using the same signals was expected.

#### 2.3.5. Diagnosis models

Fault identification process was split into two layers. In the first layer, algorithms were developed to provide the probability of a single observation (data from 10 second window) of belonging to the different classes. In the second layer or the propagation, the observation-wise probabilities were used by another algorithm to identify the underlying signal, the true class of the experiment, that was obtained by considering the chronological information.

The following key aspects were considered during the development of the diagnosis algorithms:

- In order to avoid experimental noise, instead of carrying out random training/testing partitions, LeaveOneGroupOut cross validation paradigms were favoured, as they did not leak data from the same experiment to the testing set resulting in a more reliable estimation of the error. Data from one experiment of each class was left out of the training phase and all of them were used to validate at each iteration.
- Imbalance was tackled by combining the SMOTE (Synthetic Minority Oversampling Technique) method (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) followed by the Tomek Links method for undersampling (Tomek, 1976). This way the number of observations belonging to each of the classes was equalized.
- Tree-like algorithms were preferred due to their simplicity (explainability) and capability to handle multiconlinearity. Additionally, seeking for robustness in the development of the trees, the tree depth was set to 5 to avoid overfitting and make the tree use only those signals that were significant. The minimum number of observations per leaf was set at 450. This decision was made to force the trees to contain more than one complete experiment on the final leaf, since the shortest experiment contains 360 observations. These values were established in order to generalise the solution as much as possible. However, other values were tested for these parameters, obtaining similar results. *Gini* impurity function was used for the measure of the quality of a split.

As some faults required of less effort than others to be diagnosed, several diagnosis models were stacked. Each time some faults were identified as similar and difficult to distinguish from each other, another model was created for those specific faults trying additional and more complex approaches.

#### 2.3.6. Propagation of probabilities

The diagnosis models explained in the previous section provided a observation-wise or instant-wise class probability array as shown in Figure 1 Left). In order to improve the overall accuracy of the model, a Kalman filter like algorithm was used (Kalman, 1960). Starting from a scenario of equally probable class state, the algorithm kept updating the states (probability of having a certain class fault) with each new observation (a vector of probabilities provided by the diagnosis model). This way, the algorithm filtered the intermittence of the diagnosis layer by providing clearer trends as in the example of Figure 1 Rigth).

#### 2.3.7. Feature engineering

As detecting some classes was non trivial from the raw data, feature engineering was used to create more meaningfull features that could help to disambiguate. This feature engineering consisted on the use of certain thresholds to detect the amount of data surpassing this value from the total number of measurements at that time, that is, creating a ratio. This ratio variables were done in an on-line trend without violating any time series constraints. The meaningfull features detected were *VacuumValveClosedvStd* for class 5 and *DurationPickToPickvStd* for class 7. If $V$ is the ordered set containing chronologically ordered observations of VacuumValveClosedvStd, and $D$ the ordered set containing chronologically ordered observations of *DurationPickToPickvStd*, the new variables *ratioV* and *ratioD* are defined for $i$-th observation as follow:

$$\begin{aligned} ratioV_i &= \frac{|N_{V,i}|}{i}, \\ ratioD_i &= \frac{|M_{D,i}|}{i}, \end{aligned} \tag{1}$$

where the sets $N_{V,i}$ and $M_{D,i}$ are defined by,

$$\begin{aligned} N_{V,i} &= \{v \in \{v_k\}_{k=1}^{i} \subseteq V |\ v > 0.2\}, \\ M_{D,i} &= \{d \in \{d_k\}_{k=1}^{i} \subseteq D |\ 0.26 < d < 0.45\}. \end{aligned} \tag{2}$$

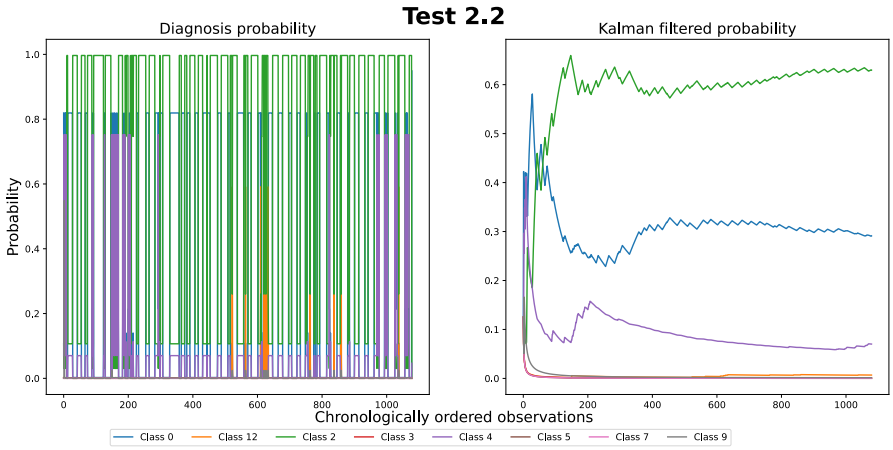



Figure 1. Diagnosis and propagated diagnosis of test 2 of class 2. Left) Diagnosis probabilities as provided by the diagnosis model. Right) Kalman filtered diagnosis probabilities.

Table 1. Glimpse of feature creation on test 5.0

| Observation | V | ratioV |
|---|---|---|
| ⋮ | ⋮ | ⋮ |
| 2 | 0.0498 | $0/2 = 0$ |
| 3 | 0.2519 | $1/3 = 0.3333$ |
| 4 | 0.0946 | $1/4 = 0.25$ |
| 5 | 0.0944 | $1/5 = 0.2$ |
| 6 | 0.2045 | $2/6 = 0.3333$ |

Table 2. Glimpse of ratio feature creation for test 7.0

| Observation | D | ratioD |
|---|---|---|
| ⋮ | ⋮ | ⋮ |
| 11 | 0.2135 | $0/11 = 0$ |
| 12 | 0.4209 | $1/12 = 0.0833$ |
| 13 | 0.0169 | $1/13 = 0.0769$ |
| 14 | 0.4373 | $2/14 = 0.1429$ |
| 15 | 0.0290 | $2/15 = 0.1333$ |

Examples of the creation of the these new features for tests 5.0 and 7.0 are shown in Table 1 and Table 2, respectively.

#### 2.3.8. Shortest answer time

There where several aspects to consider in order to determine the shortest required time:

Firstly, as some imputation was carried out, it was necessary to consider that the within the elapsed time, at least a single non-empty observation of the employed variables should exists. This instant was named $T_{NA}$.

Secondly, the minimum time necessary to classify the experiment in the two system parameter classifications (SPC1 and SPC2) needed to be computed, which was named as $T_O$.

Finally, for the diagnosis, as mentioned in Section 2.1, the evaluation method could not avoid the algorithm having a full picture of the experiment on the first run. However, there was no a direct way to transfer information regarding the run to the algorithm. Consequently, it was decided to assume that, regardless of the run, the most probable real class was identified at the end of the experiment. By doing so, the cutting point/instant became the first point of time where the most probable class was the same as the most probable class at the end of the input test. This assumption ensured consistent results and allowed the algorithm to benefit from the full picture it had on the first run, this time was called $T_D$. Note that, as many diagnostic models are present in the algorithm, $T_D$ represents the longest time required by the diagnostic algorithms of the current test. Therefore, $T_D = \max(T_{D'}, T_{D''})$ and $T_{D''} = 0$ in case the class was different to 0, 5 and 7.

In summary, the minimum number of time windows required by algorithm to classify the input experiment, was calculated as $T_c = \max\{T_{NA}, T_O, T_D\}$.

## 3. Results

The final algorithm consisted of a combination of sub-models which were assembled as shown in the diagram in Figure 2.

The divide, propagate, and conquer strategy allowed to split the initial big problem (classification of 9 classes) into smaller classification problems, in which robustness of the final algorithm was always sought. One of the major worries during the development of the algorithm was incurring in overfitting, as the classes were really imbalanced and not considering that could lead to bad results on the testing. At the same time, due to the small amount of the faulty class tests, it was not possible to totally exclude some data to validate later our methods. Efforts were made to withdraw strong rules that fitted well the data and did not provide false positives nor overfit the

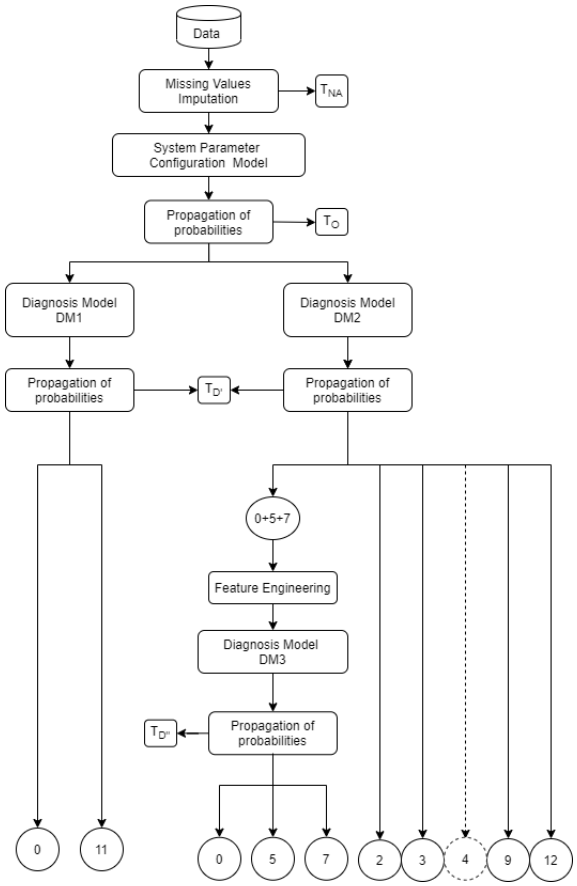



Figure 2. Execution Flow of the Classification Algorithm.

data and provide overoptimistic results.

Initially, regarding the identification of operating conditions in healthy tests, a single rule was required for their correct identification as Figure 3 shows. This rule was capable of detecting the different operating conditions without ambiguity, except for a couple of outliers that appeared on the beginning of some tests.

Due to the peculiarities of the dataset, more than a single diagnosis models were required to perform a full identification of the faults. The first decision tree (DM1) separated the healthy experiments that were done with the first parameter configuration and class 11 failures. The remaining healthy cases and the 8 faulty classes were disambiguated by another decision tree (DM2) since those experiments were carried out with the second parameter configuration. In addition, due to the difficulties to discern between classes 0, 5 and 7, an additional tree (DM3), with extra features (ratio variables for VacuumValveClosed and DurationPickToPick) was needed. Using this new approach, a diagnostic model was achieved that correctly classified the two faults 5 and 7, obtaining an average Recall in the training step of 0.9825 for class 5 and 0.9806 for class 7. In the validation step of the DM3 model,

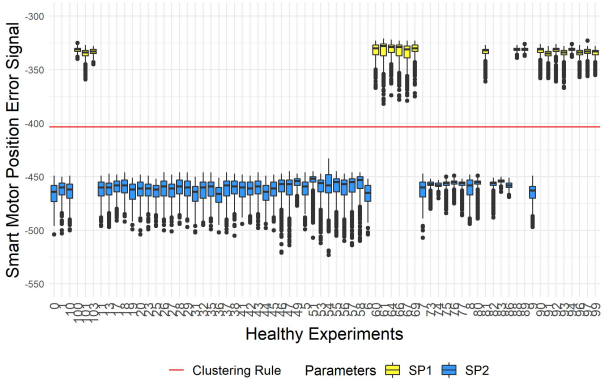



Figure 3. Boxplots of healthy tests, red line represents the operation configuration decision rule.

a mean value for Recall was obtained in the corresponding iterations to leave out each of the experiments of 0.9625 for class 5 and 0.94875 for class 7. Those results were considered successful and the DM3 diagnosis model was added to the final algorithm.

Finally, regarding fault class 4, differences between the three available tests of this type were observed. Some class 4 tests had the same behaviour as some healthy tests whereas the remaining one had a similar behaviour as other healthy classes, as shown in Figure 4. Hence, depending on the tests which were selected for training and testing, the models were very different without giving a chance to obtain robust solution for the diagnosis of this class. Thus, although the mean value of Recall metric in the training phase was 0.714, the diagnosis model was not able to correctly classify the testing experiments and the Recall value in the validation was 0 in all iterations. Furthermore, the most important signals were different in the three tests. For these reasons, it was assumed that the testing set would have a similar distribution of the cardinality as the one on the training, and the final algorithm labelled 4 class predictions as healthy. Additionally, to compare the results obtained for the diagnosis of faulty classes 5 and 7 with the results obtained with faulty class 4, LeaveOneGroupOut cross validation is used once again. Figure 5 shows the confusion matrix of one of these iterations.

All in all, the final algorithm only required 10 features for the diagnosis and propagation, which are shown in Table 3 .

Regarding the Root-cause-analysis, the signals that were identified as most important in relation to each fault class are shown in Table 4. Finally, the propagation thought the Kalman algorithm provided very interesting results. In the tests where the class value was not clearly predominant according to the diagnosis algorithm, the filter allowed to visualise a clearer trend, as the Figure 1 shows, which improved the final diagnosis.

The solution dealt properly with the identification of the tests

Table 3. Set of features used by the final algorithm.

| **Features** | |
|---|---|
| SmartMotorPositionErrorvMin | VacuumValveClosedvStd |
| DurationPickToPickvStd | DurationRobotFromFeederToTestBenchvalue |
| SharpnessImagevalue | NumberFuseDetectedvMin |
| TotalCpuLoadNormalizedvStd | SmartMotorSpeedvStd |
| Temperaturevalue | DurationRobotFromTestBenchToFeedervalue |

Table 4. Feature Importance for each class

| **Class** | **Features** |
|---|---|
| Class 2 | FeederAction2<br>Humidity<br>NumberFuseDetected |
| Class 3 | SharpnessImage |
| Class 5 | VacuumValveClosed |
| Class 7 | DurationPickToPick |
| Class 9 | SmartMotorSpeed |
| Class 11 | SmartMotorPositionError<br>DurationRobotFromFeederToTestBench<br>DurationRobotFromTestBenchToFeeder<br>DurationTestBenchClosed |
| Class 12 | DurationRobotFromFeederToTestBench<br>DurationRobotFromTestBenchToFeeder |

that were provided for the training, allowing a totally accurate diagnosis of faults except for class 4 which was predicted as 0 on purpose (assuming the testing set would follow a similar distribution on the number of tests per class). Additionally, the time required to determine the class was most of the times very short, since 2 observations were enough to detect the class in many of the tests. In average 18 observations were required for all the tests in the train and only in 5 cases more than 100 observations were necessary to classify the experiment correctly. Finally, the features related to each class (or root cause analysis) were visually validated.

In the context of the challenge, the solution has been proved to be robust. It has obtained the highest Final score, obtaining

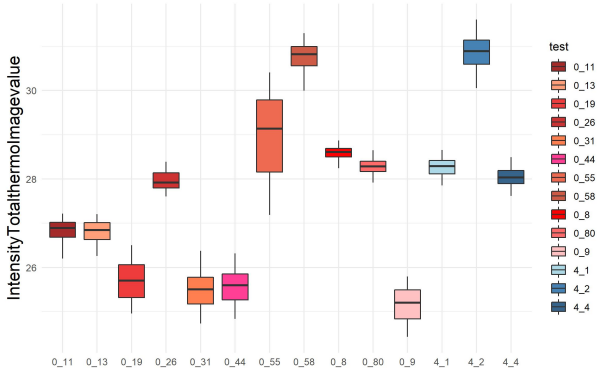



Figure 4. IntensityTotalThermoImagevalue boxplots per test for class 0 and 4.

the highest scores amongst the participants for accuracy and required time, a full score in clustering and a more modest score in root cause identification.

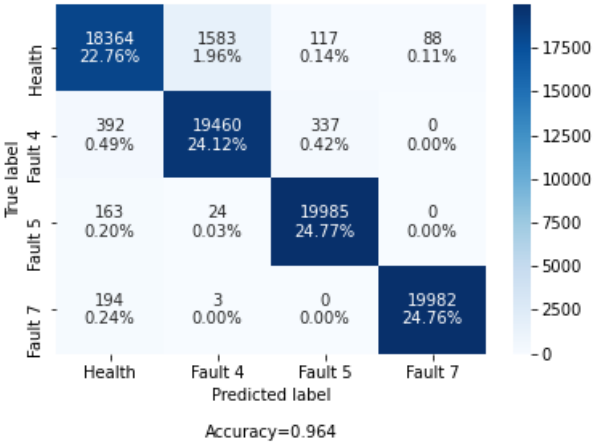
These results demonstrate that the algorithm has been able to generalise well, in sight of the accuracy score; and, that the policy developed for minimising the time required has also behaved correctly according to the timing score. In addition, the clustering rule has shown very robust results.

Regarding the root-cause-analysis, the main differences between our solution and the official solution are found in fault classes 7 and 11. For fault class 7, the important features according to the official solution are FusePicked and VacuumFusePicked, instead of DurationPickToPick, which is our choice. In a scenario with highly colinear features, it is probable that all the mentioned features could be valid. Furthermore, for faulty class 11, the official solution has only considered the features corresponding to the separation between the two operation configurations, but features that separate the faulty class 11 with its operation's healthy class are also considered in this work.

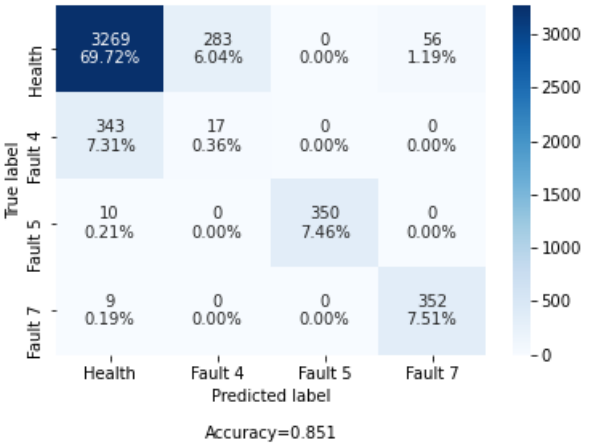
## 4. Final Remarks

The solution presented in this paper must be considered in the context of a challenge, with limited access to relevant information and, at the same time, a considerably short time to develop a proposal. This approach is based on the need to split a big problem into smaller and simpler parts that could be solved independently from the each other. This was achieved by: employing decision trees and similar structures that allowed to identify certain classes at a time and leaving less uncertainty to the next layer of the algorithm; a separate feature selection methodology that helped in the development of diagnosis algorithms; the propagation of the observation-wise diagnosis of the algorithms; and, a backwards identification of the required computation time.

Overall, the solution has yielded satisfactory results, obtaining the highest score in the competition and showing great generalisation capability by only failing to detect the class 4 tests, as expected by design. This has been achieved by stacking a rater simple algorithm: The decision tree, which has the added advantage of being understandable. Additional interesting findings are the appropriateness of feature engineering approaches, that have helped in the diagnosis of difficult




(a) Train Confusion Matrix




(b) Test Confusion Matrix

Figure 5. Confusion Matrix of an iteration of LeaveOneGroupOut Classes 0,4,5 & 7. Test used for validation: 10 random tests of class 0, and test 4.1, 5.4 and 7.5 for other classes.

classes; the propagation with Kalman filter, that allows the identification of trends in noisy tests; and the strategy used for the detection of required time, which helps the solution to make very precise adjustment of the required time.

This approach has, however, some limitations. Firstly, regarding the strategy used to detect the shortest time required for the classification, it needs to be mentioned that this approach fits well inside the challenge context, but it is not applicable in a real scenario, as there is no first and second run for each test. Secondly, it has not being possible to detect all the faults, which makes this solution incomplete.

Team HIRUTEK is aware of the limitations of this work, but, at the same time, some interesting techniques are presented and could be further studied. In that sense, we consider of great interest working on the identification of fault 4, which has been left aside in this work, but that we believe that could be tackled by using frequency or other time/frequency techniques. Also, it might be interesting to study the missing values and their correct imputation, as they could be a source of interesting information even if they have been overlooked in this work. Finally, the use of Kalman filters to visualise trends of probabilities could be further extended and studied including more realistic strategies to identify at which point is the class probability stable enough to provide a label for the test.

## ACKNOWLEDGMENT

This work is partly supported by the project 3KIA (KK-2020 / 00049), financed by the SPRI-Basque Government through the ELKARTEK program and AI-PROFICIENT, which has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 957391.

## REFERENCES

Assad, F., Konstantinov, S., Nureldin, H., Waseem, M., Rushforth, E., Ahmad, B., & Harrison, R. (2021). Maintenance and digital health control in smart manufacturing based on condition monitoring. *Procedia CIRP*, *97*, 142–147. doi: 10.1016/j.procir.2020.05.216

Barnes, S. A., Lindborg, S. R., & Seaman, J. W. (2006). Multiple imputation techniques in small sample clinical trials. *Statistics in Medicine*. doi: 10.1002/sim.2231

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*. doi: 10.1613/jair.953

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering, Transactions of the ASME*. doi: 10.1115/1.3662552

Kaufman, L., & Rousseeuw, P. J. (1986). CLUSTERING LARGE DATA SETS. In *Pattern recognition in practice.* doi: 10.1016/ b978-0-444-87877-9.50039-x

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, *18*(17), 1-5. Retrieved from `http://jmlr.org/papers/v18/16-365`

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., & Hornik, K. (2021). cluster: Cluster analysis basics and extensions [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=cluster`

Moritz, S., & Bartz-Beielstein, T. (2017). imputeTS: Time series missing value imputation in R. *R Journal*. doi: 10.32614/rj-2017-009

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V.,

Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Stavropoulos, P., Chantzis, D., Doukas, C., Papacharalampopoulos, A., & Chryssolouris, G. (2013). Monitoring and Control of Manufacturing Processes: A Review. *Procedia CIRP*, *8*, 421–425. doi: 10.1016/j.procir.2013.06.127

Ting, K. M. (2010). Precision and recall. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of machine learning* (pp. 781–781). Boston, MA: Springer US. Retrieved from https://doi.org/10.1007/978-0-387-30164-8$_6$52 doi: 10.1007/ 978-0-387-30164-8$_6$52

Tomek, I. (1976). Two modifications of cnn. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-6*(11), 769-772. doi: 10.1109/TSMC.1976.4309452

Wickham, H., François, R., Henry, L., & Müller, K. (2019). *dplyr: A Grammar of Data Manipulation. R package version.*

## BIOGRAPHIES

**Kerman López de Calle - Etxabe** obtained his bachelor in Renewable Energies Engineering from the University of the Basque Country (UPV/EHU) in 2016. In 2016-2017 he obtained his master's in Computational Engineering and Intelligent Systems from the same university, after he developed his master's thesis in collaboration with the Research Center Tekniker, where he carries out his research currently. From 2017 to 2020 he pursued a PhD in Information Engineering related to the development of condition monitoring algorithms. To date, he has published various works in high impact journals and has also contributed to diverse conferences in the field of condition monitoring and data analysis.

**Meritxell Gómez - Omella** got her BSc in Mathematics from the Universitat Autònoma de Barcelona (UAB) in 2018. In 2019 she obtained her MSc degree in Modeling and Mathematical Research, Statistical and Computing from the University of the Basque Country (UPV/EHU), with a master's thesis related to data quality and missing data imputation in collaboration with the Research Centre Tekniker. Since September 2019 she is pursuing a PhD in Time Series Analysis and Forecasting. Currently, she is a Junior Data Science of the Intelligent Systems Unit in Tekniker and she has been working in different national and European projects.

**Eider Garate - Perez** acquired her Bachelor in Mathematics from the University of the Basque Country (UPV/EHU) in 2020. Presently, she is studying a MSc degree in Modeling and Mathematical Research, Statistical and Computing from the same university, and is doing her master's thesis related to missing values imputation methods in time series in collaboration with the Research Center Tekniker.