SYSBOOSTER, application of Data Science to surveillance of systems for detection or anticipation of dysfunctions or failures of systems

Alain Celisse, Olivier Gauriau, Margot Corréard, Jean-François Bouin, Lennart Priester, Ronald Naumann, Emmanuel Arbaretier, Michel Kaczmarek, Uwe Schmietainski, Hagen Friedrich

Institut pour la Maîtrise des Risques
Sûreté de Fonctionnement - Management - Cindyniques

SYSBOOSTER, application of Data Science to surveillance of systems for detection or anticipation of dysfunctions or failures of systems

SYSBOOSTER, application de la Data Science à la surveillance de systèmes pour la détection ou l'anticipation de dysfonctions ou défaillances de ces systèmes

Alain Célisse
Olivier Gauriau
INRIA
Parc scientifique de la Haute-Borne
40, avenue Halley - Bât A - Park
Plaza, 59650 Villeneuve d'Ascq
alain.celisse@inria.fr

Margot Corréard, Jean-François Bouin
DIAGRAMS TECHNOLOGIES
Parc Euratechnologies
165 ave de Bretagne
59000 LILLE
mcorreard@diagrams-technologies.com
jfbouin@diagrams-technologies.com

Emmanuel Arbaretier
Michel Kaczmarek
APSYS
1 boulevard Jean Moulin
78996 Élancourt Cedex
emmanuel.arbaretier@apsys-airbus.com

Lennart Priester, Ronald Naumann, Dr
Uwe Schmietainski, Hagen Friedrich
NOKIA
Thurn-und-Taxis-Str. 10
90411 Nuremberg, Germany
lennart.priester@nokia.com

***Abstract*—This paper describes a pipeline designed by INRIA, DIAGRAMS TECHNOLOGIES, APSYS, and NOKIA for addressing questions related to root-cause analysis. These different tools, enabling multivariate multiple change-point detection as well as the automatic detection of future failures, are part of the achievements of the SYSBOOSTER project. For confidentiality reasons, many data and technical information have been changed or anonymized along the paper.**

***Résumé*—Cet article décrit un processus méthodologique supporté par une boîte à outils logiciels couplés conçus par INRIA, DIAGRAMS TECHNOLOGIES, APSYS, et NOKIA, afin d'adresser des questions relatives à l'analyse de cause. Ces différents outils, permettant la détection de phénomènes de dérives multi variables et multi points de vue, ouvrent également la porte à l'anticipation de défaillances futures, dans le cadre du projet européen SYSBOOSTER. Pour des raisons confidentielles, de nombreuses données et informations techniques ont été modifiées ou anonymisées tout au long du papier.**

***Keywords*—dysfunctions, failures, data science, predictive maintenance, diagnosis, troubleshooting, prognosis, HUMS**

I. INTRODUCTION

One strength of SYSBOOSTER is its very broad audience since it provides benefits to any industrial actor concerned with operating on or maintaining industrial assets. In particular it allows for maximizing the Service Continuity in terms of reliability or availability.

General description of the SYSBOOSTER product:
The SYSBOOSTER project is a **European EIT Digital project** which has led to a methodological process (pipeline) and a software toolbox. Firstly, the SYSBOOSTER methodology is a strong support in terms of qualification of failures, diagnosis / identification of the root cause(s), and troubleshooting of these failures. Secondly, the SYSBOOSTER software toolbox is composed of elementary bricks which turn out to be a powerful package for operational surveillance and maintenance achievement.

The first benefits of SYSBOOSTER have to do with failure detection, root cause analysis and identification, and troubleshooting process. One of the most striking asset of the SYSBOOSTER product is to warn against potential failures before any significant deviation is observed on the operational field.
For the SYSBOOSTER product to work properly on new data, it first requires a preliminary learning step based on *labeled data*, as any machine learning procedure would do. These *labeled data* have to be collected from the field and operational environment in the same way as the classical data that are to be dealt with usually. For every sample, it is important to collect and know:

- the recording of several descriptors (features) of the behavior along the time (multiple time-series, one for each recorded descriptor),
- if the "object" described by these features has failed or not,
- if the "object" has been already repaired or not (with success or not),
- if the original root cause (internal or external) has been already identified or not.

This amount of information constitutes the SYSBOOSTER input which will be integrated in final reporting concerning industrial use cases of end user.

Description of the problem addressed by SYSBOOSTER:
Our deliverable aim is to detect and characterize potential anomalies arising in the behavior of optical modules, those anomalies corresponding to early signs of future failures.
This early detection and characterization will help the Root Cause Analysis of failure. Roughly speaking, after pre-processing the end user's data set collected for learning purposes, several machine/statistical learning procedures (issued from Artificial Intelligence domain) are combined to achieve our goal. The different steps of our process (pipeline) are the following ones:

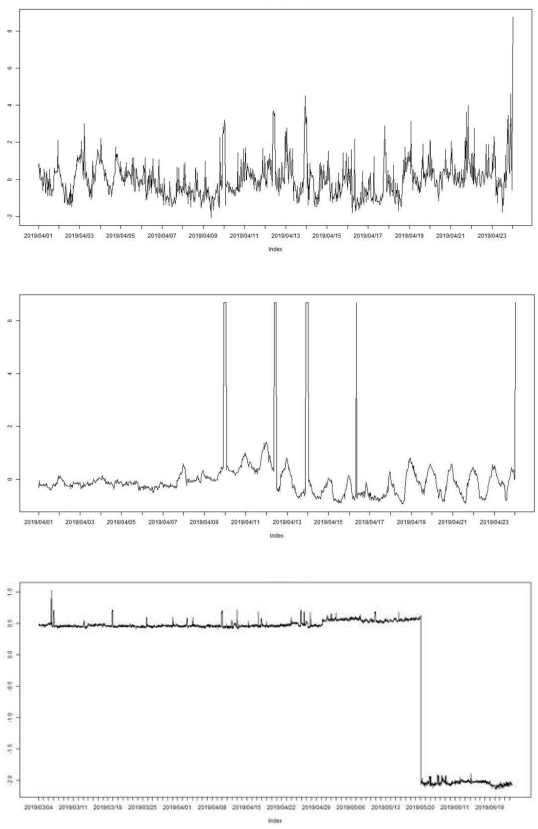
- Pre-processing of the (learning) dataset collected by the project end-user;
- *Simultaneously segmenting the multiple time-series* corresponding to the recorded features describing the behavior of the module under analysis along the time;
- Extracting meta-descriptors from each segment output by the previous simultaneous segmentation step;
- Clustering of "failure" segments, using descriptors of "failure" segments, into homogeneous classes (each class corresponding to a type of failure);
- Fitting classification models for learning the prediction rule of any potential failure. This is made from previously computed clusters combined with healthy segments.

Using the clustering and classification models, we are able to highlight which meta-descriptors and which signals (recorded variables from the original dataset) are the most influential ones regarding a particular failure type. Each step will be further described into more details in what follows.

II. LEARNING DATA DESCRIPTION

The end-user's equipment to be analyzed is composed of modules, which are parts of cards, which are put themselves into a specific shelf. The dataset was sorted shelf by shelf and in a chronological manner. These raw data are then gathered module per module.

The total number of modules included in this learning step is 3400. For each of them, 27 were categorical data and 20 descriptors (features) have been recorded along the time (20 time-series). The recording was 5-month long with one measurement every 30 minutes, which corresponds to time-series with around 7000 timestamps for each of the 20 descriptors.
Strong changes in the regime of each variable along the time have not been necessarily related to real failure occurrence according to end-user from the field: In the following, the term "failure" will therefore be used to distinguish modules with abnormal behavior from those that show expected behavior. The related criteria to be abnormal has been delivered by the data provider along with the data samples.
Abnormal modules have still been found operational in the field, only a few had really been sent for repair during sampling time. The data provider wanted to learn the reason for these abnormal (non-standard) indications in the sample data.
Such changes can be related to environmental modifications (temperature, intensity of the workflow…)



Figures 1-3: Examples of regime changes along time for 3 signals.

Therefore all recordings have been normalized so that the signals have zero mean and unit variance for each recorded variable on each module. Missing data are taken into account by imputation techniques, or removed from the dataset when imputation was not relevant.

III. MORE DETAILED DESCRIPTION OF THE PIPELINE STEPS

Just after normalizing the (learning) data, the first step consists in automatically detecting "homogeneous" regions along the time across the different recorded features for a given module. This is what we call the (simultaneous) *joint segmentation* of multiple time-series. This step is all the more relevant in the present context as the learning data reflects such homogeneous regimes across the recorded features, with sudden abrupt changes between successive regimes. Such abrupt changes simultaneously arising in (multiple) signal(s) are called changepoints in what follows. The output of this joint segmentation step is therefore a collection of (temporal) segments (simultaneously shared by several features), which will serve as a basis for the subsequent steps.

A. Segmentation

The R package used for the joint segmentation step of our pipeline is called KernSeg and has been developed by INRIA. It is not only computationally efficient (by saving both time and memory consumptions), but it also provides a great improvement upon ongoing segmentation strategies in several respects.

On the one hand, it takes advantage of the use of the so-called reproducing kernels for detecting changes that are not limited to the mean or the variance of a time-series. On the contrary, any appropriate choice of such a reproducing kernel allows for detecting any change arising in the process that has generated the observations, which is particularly relevant in the present context. On the other hand, these reproducing kernels are powerful enough for allowing the simultaneous segmentation of multiple time-series along the time, under the assumption that the changepoints occur simultaneously in most of these time-series. From a more general perspective, the segmentation procedure involves sophisticated model selection strategies, which reaches a trade-off fitting the data and avoiding too complex models.

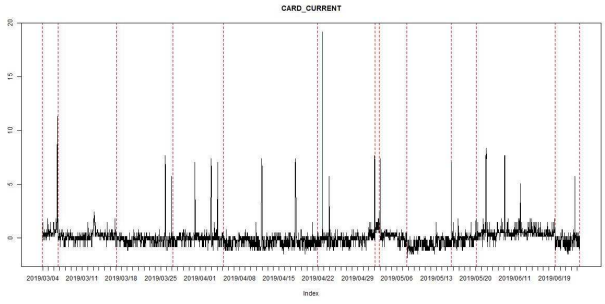
Two-stage learning strategy:

a- First stage: The kernel-based segmentation procedure is first applied to each recorded feature individually. The goal at this stage is checked if the corresponding feature exhibits (or not) any change in its behavior (characterized by a changepoint in the close neighborhood of a failure). This helps us identifying 6 relevant features among the 20 candidates that carry some information about the failure occurrences.

b- Second stage: As long as the 6 relevant features have been identified, the R package KernSeg is applied to the 6 corresponding time-series to perform their joint segmentation for each module. This outputs homogeneous segments that are shared across these 6 features, which correspond to different regimes of the corresponding module during the recording. Moreover, this joint segmentation avoids being too sensitive to small events related to only one particular feature (which should be interpreted as part of the noise for the present purpose). In particular, this provides us with a partition of signals that is smoother than the one we would have got from the individual segmentation of time-series.

Summary:

1. Input: Multiple normalized time-series
2. Use :
   a- Selection of the 6 most relevant variables,
   b- Automatic joint segmentation of these 6 time-series, module by module.
3. Results:
   a- Detection of the 6 most relevant variables
   c- Output one joint segmentation for each module



Figures 4: Preprocessed signal segmented.

Performances: automatic identification of 11000 segments labeled as "failed", and 120 000 segments labeled as "healthy", for a total of 131 000 segments over all the modules.

B. Meta-descriptors extraction for each segment

The ability of KernSeg for detecting changes that are not limited to the mean or the variance of a signal (time-series) is especially useful in the present context since it turns out that interesting features of the distribution regarding our root cause analysis problem are precisely not carried out by the mean. As a result, the segments cannot be only characterized by their respective means. For instance, considering the variance or the skewness within each segment could be relevant. Therefore our goal is to design descriptors of the segments that capture these different kinds of information that is likely to be relevant in our context.

Therefore, a total of around 90 meta-descriptors of each segment has been computed among which the mean, the variance, some Fourier coefficient….

Summary:

1. Input: jointly segmented times-series
2. Output: Around 90 meta-descriptors for each segment (either "healthy" or "failure").

C. Clustering the segments towards homogeneous classes

Once the "failure" and "healthy" segments have been labeled from the joint segmentation step, their respective 90 meta-descriptors are computed from the previous step.

However it turns out that the failure segments exhibit a strong heterogeneity which reflects that several different (abnormal) behaviors are likely to lead to a "failure".

A clustering strategy has been applied to "failure" segments for automatically defining homogeneous classes among them. This has been made possible by means of the 90 meta-descriptors of each segment and a clustering procedure

relying on the MixtComp package that has been developed by INRIA. The output of the MixtComp package is two-fold:

i. the automatic choice of the (a priori unknown) number of clusters,

ii. the gathering of the "failure" segments into homogeneous classes (clusters).

The MixtComp package relies on the mixture models technology for which efficient model selection techniques already exist.
A striking byproduct of this clustering step is the data-driven identification of the unknown number of different abnormal behaviors, each of them being potentially responsible for a particular type of failure.

First stage: Defining "failure" segments for learning

The "failure" segments are defined as the segments immediately preceeding a failure occurrence, which seems relevant since the goal is to detect early (weak) signs of future failure. Actually, computing the descriptors for each such segment (mean, variance, Fourier coefficients…) requires a minimum number of points. By contrast, we also define healthy segments (that will serve in the classification task) as segments that do not come before any failure, and do not share any points with a failure segment.

Second stage: Choose the number of clusters

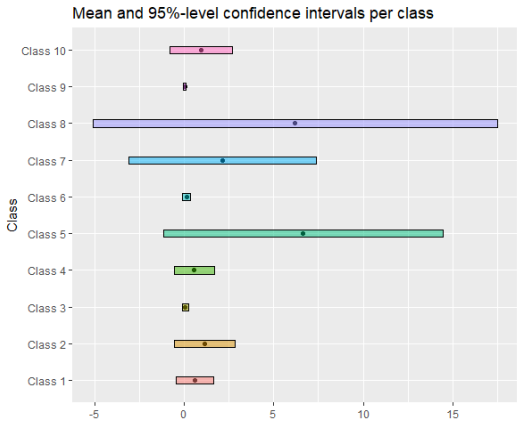
After some experiments, and following a decision criterion, we determined that the best trade-off was a mixture model with around 10 clusters, which makes a reasonable trade-off between interpretability and statistical performance. This model was reliable in term of coherence according to our criterion. By contrast, a higher number of clusters would have made the visualization of our classes impossible for the end-user.

Summary:

1. Input: Failure segments meta-descriptors;
2. Strategy: Clustering based on mixture models (MixtComp);
3. Output: Number of and classes constituted of homogeneous individuals. Each class is described by probability laws for each meta-descriptor (gaussian) with different parameters (mean & standard deviation).

   Performances: around 10 classes of failure segments, which can be interpreted as 10 typical abnormal behaviors that have been automatically detected.

As a remark, the picture below illustrates that using the variance of a given meta-descriptor (1 among the 90 meta-descriptors of each segment) for distinguishing between the different clusters would have been misleading. Actually most of the clusters share a similar mean (with widely overlapping 95 percents confidence intervals). By contrast, this illustrates the power of the present clustering step carried out by means of 90 meta-descriptors properly chosen.



Figures 5: Variance of one particular meta-descriptor along the 10 clusters.

D. Classification

First stage: The classification task requires the comparison between "healthy" and "failure" segments. The "healthy" segments have been already defined in Section 3.B and then left aside. Taking into account that the previous clustering step has output 10 clusters of "failure" segments, it is necessary to determine to which cluster each "healthy" segment is the closest. This is done by computing the distance between each "healthy" segment and the center of each of the 10 clusters previously defined. This distance is evaluated on the basis of the 90 meta-descriptors of each segment.

Second stage: For each of these clusters (which corresponds to a specific type of "failure"), the purpose of a classification procedure is to learn the rule which leads to predict the appropriate label ("healthy" or "failure") for any new segment. This learning task has to be made for each cluster individually, by comparing between "healthy" and "failure" segments within each cluster. Once such a rule has been learned for each of the 10 clusters, then it gives rise to an "identity card" for the 10 classes of potential "failures".

Third stage: From the learned "identity card" of the 10 "failure" types, the classification procedure can be applied to any new segment the label of which is unknown. The purpose is then to properly predict its label while voiding any mistake that is, avoiding false negatives (missing a "failure") and false positives (falsely predicting a "failure"). It is noticeable that such a classification rule can be applied to any new segment in an online framework (by contrast with the offline framework) where the data come sequentially and the label has to be predicted before any new observation has been made.

Learning the prediction rule with Random Forests:

Random Forests are a predictive model often used in machine learning due to their overall good prediction performances (and sensible underlying mechanism). From a wide comparison between several such predictive models, Random forests were identified as the best procedure in the present setting.

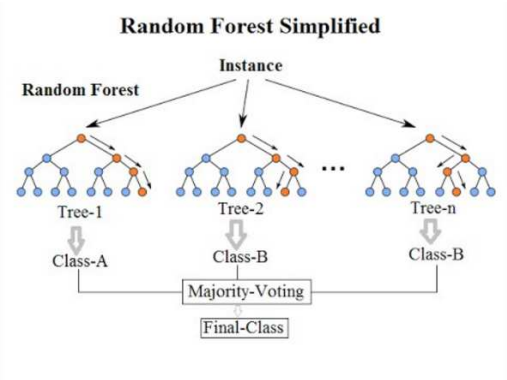


Figure 6 : Illustration of the underlying mechanism within the Random Forests.

Each random forest is made of several simple binary classification trees (CART) the outputs of which are combined in through a final majority-vote rule. At each node of a tree corresponds a variable and a threshold that have been learned from the training data.

In the present situation, 80% of the segments of each cluster were used for the training, while the remaining 20% have been used to assess the performance of each learned rule (testing). The rule that is finally learned ("identity card") is the one which achieves the best statistical performance one the testing data.

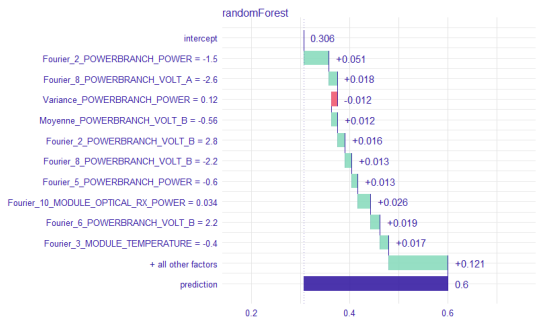
Summary:

1. Input: Labeled data from each cluster successively;
2. Goal: Learning the "id" of each cluster;
3. Output: Classification rule dedicated to each cluster;
4. Byproduct: Access to influential meta-descriptors in the classification rule of each cluster (type of "failure").

Performances: 75% of accurate classifications on the validation set.

Interpretation of the each classification rule:

Once the predictive model (Random Forests) has been learned appropriately, the influence of each meta-descriptor in the classification rule of each cluster can be inferred for interpretation purposes. Accessing to which meta-descriptors play the most important role, is a crucial information since it clearly helps in the root cause analysis for further technological hardware improvements for instance.



Figures 7: Importance of specific descriptors in the classification process of a new segment.

IV. DESCRIPTION OF THE SYSBOOSTER PLATFORM

Based on the end-user's dataset, INRIA has developed a methodology (supported by a software tool organized as a pipeline combining elementary bricks) and its corresponding prototype.

At the end of the project, the prototype consists of different software bricks (mainly R packages) that can be used either independently of each other, or rather sequentially applied carefully following the methodology (pipeline) earlier described.

The following description intends to illustrate how these bricks behave as well as the different possibilities they give access to, namely as a Root Cause Analysis (RCA) solution or an "Anomaly Detection" solution.

Note: The statistical and practical efficiency of the different elementary bricks on the end-user's dataset have been evaluated at the end of the project.

A. Brick 1: Data processing, Visualizing a module behavior

Characteristics of the dataset: the dataset consists of data from multiple modules, each module being described by: (i) multiple time-series (module temperature,…), and (ii) categorical variables (for instance describing if the module is working correctly or not). The time-series are normalized once uploaded on the platform.

1. *Joint segmentation brick*

This brick enables to perform the automatic detection of abrupt changes (changepoints) arising simultaneously across multiple time-series (features measured along the time) for each module. The resulting joint segmentation (shared by all recorded time-series) is not characteristic of any operating mode at this point, but only defined successive regimes. This joint segmentation brick also enables to reduce the quantity of information for the future analysis. It can be seen as a preliminary smoothing step.

Output of the brick: segments along the time during which most of the recorded variables exhibit a stationary (homogeneous) behavior.

2. *Analysis of the successive regimes for a module along the time and visualization*

The purpose of this step is to analyze (and visualize) the different regimes of a given module along the time to analyze how the module is operated.

To this end, from the joint segmentation of a given module at the previous step, the output segments of this module are divided into the 10 clusters already defined from the SYSBOOSTER pipeline. The corresponding module is labeled along the time according to the successive labels of the classes to which the segment belongs. From a visualization perspective, this gives access to the successive labels of a given module along the time. For instance, this could help the end-user identifying complex patterns which would be characteristic of abnormal behaviors.

In addition, for each label of a given module, the visualization allows for enumerating the most influential features defining the corresponding cluster (average temperature, …), which helps characterizing the current regime of the module along the time.

THIS BRICK 1 IS A STEP TOWARDS RCA GIVING THE POSSIBILITY TO ANALYSE EASILY HOW A MODULE IS OPERATED.

B. Brick 2: anomaly detection

The dataset consists of data from multiple modules, each module being described by multiple time-series without knowing if there are failures occurrences. From a new batch of data, the purpose is to predict/identify the periods of healthy or abnormal behaviors. For each module, the output is a list of time periods during which the operating mode is different from a healthy mode.

Main steps:

1. performing the joint segmentation of recorded features (time-series);
2. compute the meta-descriptors of each segment;
3. apply the classification rule already learned by means of random forests for classifying any new segments as “healthy” or “failure”.

On the existing dataset, this brick allows for identifying all abnormal segments of a new module. Details about what are the possible anomalies are provided as well, for maximizing the interpretability by the end-user and for RCA purposes.

THIS BRICK 2 CORRESPONDS TO ANOMALY DETECTION IN A BATCH OF NEW UNLABELED DATA AND IDENTIFICATION OF WEAK EVENTS RELATED TO FUTURE FAILURES.

C. Brick 3: Visualizing the module behavior before failure

The module is described by multiple time-series. There are two possibilities for applying this functionality: either failures have been observed by the end-user, or there is no recorded failure but the output of Brick 2 is available (among which segments labeled as “failure” or “abnormal behavior”).

The purpose of this functionality is mainly visualizing the possible symptoms related to a failure for a given module. It relies on the preliminary identification of such abnormal segments output by Brick 2 for instance.

1. This functionality allows the end-user for exploring the time series before any failure (in the area of any segment labeled as “failure”). The expert can not only visualize the time-series corresponding to the recorded features (that he can individually select or remove), but he can also access to some information regarding the most influential meta-descriptors in the classification rule of this segment as “failure”. This kind of visual exploration can be done on ONE MODULE at a time. This is a key step towards RCA.

THIS BRICK 3 CORRESPONDS TO RCA WORKS. Its use requires having identified potential failures (output of BRICK 2).

V. RESULTS

For monitoring the SYSBOOSTER performances, different KPIs have been defined related to both technical (statistical performance, computational efficiency) and business aspects.

A. Number of categories of abnormal behaviors

This KPI targets to measure how many groups (clusters) with an abnormal behavior can be identified from the end-user’s use case.

Objective: By discovering similarities in the apparent abnormal behavior of some segments compared to the others, the objective is to perform an automatic clustering of the “failure” modules. This is an important step in the SYSBOOSTER pipeline since it automatically identifies different categories of anomalies (which is not known a priori), and also allows for further exploration of the phenomenon that is responsible of this type of “failure”.

Targeted result: Since nothing was known on this aspect at the beginning, we expected to find around 2 categories of abnormal behaviors.

Obtained results : the SYSBOOSTER tools have automatically detected around 10 types of abnormal behaviors.

Detail & explanation:

We started by defining what is an abnormal behaviour for the end-user‘s experts. In the application domain of the end-user, the equipments are highly reliable (99,999%) and a true failure for this kind of technology does not mean that the module broke or that it is in an irremediable state (which almost never happens).

In a first step, we worked with the end-user to identify the abnormal behaviors for a module according to their field expertise (suboptimal behavior for instance). In a second step, we applied the SYSBOOSTER pipeline from these expert-based abnormal behaviors.

1. Abnormal behaviors based on experts

Works with the end-user enabled to identify a list of behaviors which have to be considered as "abnormal functioning" (or to simplify "failures"):

a- Fail message of a specific signal X : these failures are considered failures even if they do not correspond to an irremediable failure.
b- Exchange of a specific module Y in the dataset. When a module is exchanged it is very likely due to a failure even if not yet confirmed by the repair department.
c- Modules sent to repair and repaired (only one in the whole dataset).

2. Automatic abnormal behaviors analysis based on data

First achievement:
As modules are operated under multiple field conditions, which lead to multiple behaviors along the time for a single module, a first step was to identify "homogeneous" segments along the time during which a module exhibits a stable (stationary) behavior.
This step is done using an automatic joint segmentation method.

Output 1: The automatic joint segmentation method allowed to select 6 recorded variables (features with time-series) which are informative for anomaly detection (during an "abnormal functioning", the behavior of the module through the 6 selected variables seems different from the behavior before and after an "abnormal functioning").
>> This first output enables the end-user focusing on those specific features for future data collection, reducing its corresponding cost.

Second achievement:
The automatic segmentation, combined with classification, enables to define the time frame before a "failure" which has to be analyzed from the 6 parameters (for detecting a fail message of type X for instance).

Output 2: On these particular experiments, the length of the time frame was 85mn (median). This would strongly depend on the "objects" under consideration as well as on the tuning of environmental conditions.
>> This enables to focus the classification algorithm, in its prediction mode, on a reduced period of time ensuring a effective computation speed.

Third achievement:
After this first step, a clustering is applied on the segments extracted from modules (excluding period of time during abnormal behavior) to discover various types of "abnormal behaviors" within the dataset.
Considering, the fact that the length of every segment is different (and consequently non informative), the segments are summarized by meta-descriptors (mean, variance, Fourier coefficients,...) and the clustering is done from these features (around 90 on this particular example).

Output 3: The total number of clusters (that can be interpreted as the number of anomaly types) was estimated at 10 "abnormal behaviors". Each cluster has an identity card to ease the analysis.interpretability by end-user's experts of these modes of operation.

B. Number of detected unpredictable events

This KPI targets to measure how many events end-user's experts could not detect without data science for some specific module's use cases.

Objective: The goal was to find (a priori) unknown patterns in weak signals before failure occurrences (for a specific Function Y) which were unpredictable events for experts until now. Automatically identifying those patterns as soon as the signal starts to deviate gives rise to troubleshooting procedures before any strong failure occurs and also enables end-user's experts to understand the reasons of such an abnormal behavior.

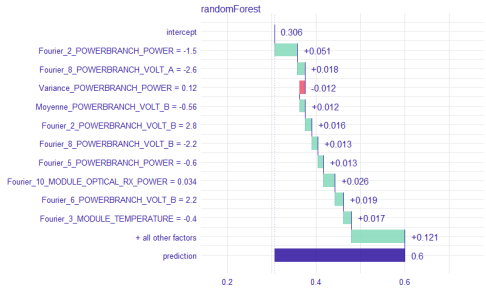
Targeted result: We took 500 failures that were not used in the learning process.

Obtained results: 375 abnormal events out of 500 have been automatically detected.

Detail & explanation:
In order to detect unpredictable events we learn a classification model based on random forests. It predicts the label (normal or abnormal) of any new segment for any given module.
The learning step has been made using 80% of end-user's dataset, while 20% has been kept aside for the performance evaluation.

Output 1: 75% abnormal behaviors (Function Y) classified accurately (375/500)

Output 2: Far more, the classification of "abnormal behavior" is explained by meta-descriptors extracted from each segment. These meta-descriptors can be considered as symptomatic of "abnormal behavior"



Figures 8: Importance of specific descriptors explaining abnormal behaviors

Thanks to these performances, it is then possible to:

- identify at any time if one module has a normal behavior
- identify any "abnormal behavior" for segments from multiple modules during a fixed period of time
- analyze into more details the symptoms (as described above) of any abnormal behavior.

Achievements:

- Helps the support services in their understanding of the customer problems.
- Helps the design department for analyzing the symptoms and possible causes of failures

C. Speed performance (% of Maximum Performance)

This KPI measures the speed performance of the method (computing speed and …).
Goal: Determine the speed performance of the pipeline.

Targeted result: We expected 85% of real time analysis.
Obtained results: We were able to reach 100%

Detail & explanation:
2 metrics are used to quantify the speed performance of the pipeline.

First achievement:
Speed necessary to identify if a given segment of one particular module has a "normal" or "abnormal behavior" (based on the preliminary learning-step): less than 1s.

Output: The method has consequently a run time lower than the acquisition time of a new point which allows for applying the classification rule on line.

Second achievement:
Speed necessary to identify, from a dataset of 24H-long & 100 modules, all the "abnormal behaviors" (based on the learning-step previously done): 1mn30s.

The *segmentation* and *classification* steps from the SYSBOOSTER pipeline have to be applied on the whole dataset (48 measurements for each recorded feature on each of the 100 modules). This explains the required time, which is longer than for the previous achievement, but still smaller than the acquisition time (every 30 minutes). Moreover in the present context, this process does not need to be fully real-time since its goal is only to help the offline detection and analysis of abnormal behaviors made by the design department.

D. Early detection of dysfunctions' leading failures

This KPI targets to measure how long before the end-user's experts the SYSBOOSTER tool is able to detect a failure for a specific module use-case.

Objective: Here, the objective is to detect a deviation (or dysfunction) leading to a failure. Until now, the end-user's experts identify the failure during the maintenance process or in the SAV department, that is to say, after the failure has occurred.

Targeted result: We expected to detect 80% of the failure before the end-user's action during the maintenance process or in the SAV department.

Obtained results: We were able to detect 75 % of these failure occurrences.

Detail & explanation:
In order to detect dysfunctions leading to failures, we learned a classification model based on random forests that will be able to predict if the behavior of a module during a period of time (segment) is normal or abnormal.

Output : **75% of abnormal behaviors (FunctionY) detection before the Function Y failure itself.**

VI. CONCLUSIONS

SYSBOOSTER has really been a very busy project where no contributor has spared his efforts to come to a high-value product in terms of methodology and final tools.
The end-user never stopped providing datasets almost until the very end of the project. Eventhough at the beginning there were difficulties to find datasets with failures, the end-user finally succeeded in collecting data corresponding to abnormal behaviors and failure cases.
INRIA and DIAGRAMS technologies carried out several improvements on the methodology as well as the development of possible supporting tools in the framework of the SYSBOOSTER platform.

The added values of the SYSBOOSTER product were numerous:

- Perform automatic anomaly detection. These anomalies were discovered by means of technologies involving AI.
- Disclose events otherwise usually not detected (detectable)
- Characterize abnormal behavior classes derived from telecommunication system operation data
- Compute and evaluate automatically most significant parameters
- Understand complex correlations and indications
- Ease the analysis of events as a relief for technical experts (engineers)
- Optimize data capture and reduce number of false positives

Moreover we can identify additional benefits:

- Data analysis with SYSBOOSTER on one module type also produces findings for other items/components
- The versatile SYSBOOSTER platform has the potential to handle a huge amount of data quickly. This offers the potential to apply the analysis to other very complex scenarios usually left unsolved.
- The SYSBOOSTER data analysis can lead to hints (trainings) for customer and system field operation

Therefore numerous may be the benefits in different perspectives: consultancy has really to accompany the SYSBOOSTER platform adaptation and methodological use to really fit best operational need and context of the client.

ACKNOWLEDGMENT

SYSBOOSTER is part of an activity that has received funding from the European Institute of Innovation and Technology (EIT). This body of the European Union receives support from the European Union's Horizon 2020 research and innovation program.

We would like to express all our acknowledgment to EIT Digital which has financed and supported us during the whole project, and which has helped us to specify our technical and commercial approach.

REFERENCES

[1] "Model selection via cross-validation in density estimation, regression, and change-points detection", thèse Paris 11, Alain Célisse, 2008.

[2] « Sélection de groupes de variables corrélées en grande dimension » par Quentin Grimonprez, thèse Lille 1, 2016

[3] «Gaussian models and kernel methods », Jérémie Kellner, Lille 1, 2016.

This activity has received funding from the European Institute of Innovation and Technology (EIT). This body of the European Union receives support from the European Union's Horizon 2020 research and innovation programme