MIG FORENSIC – Report

T. BLONDEL – M. BOURDIN – A. CONTE – P. DHALLUIN – E. GARDIN – J. GATIN – B. LE CORFEC – Z. LIANG – N. LINE – L. MOENECLAEY – T. MONNIER – S. PARTARRIEU – P-A. PLESSIX – Y. TCHOUBOUKOFF

SUPERVISORS: Sébastien TRAVADEL, Enrico ZIO

TABLE OF CONTENTS

FOREWORD

The MIG ("Métiers de l'Ingénieur Généraliste") « Forensic » we attended had two main ambitions. The first ambition was educational and aimed to give us an overview of what forensic engineering is and teach us some rudimentary skills in machine learning and data science. The second ambition was more industrial: this MIG had been set up to meet Prüftechnik's demand, that is to say, offer it an automated way of monitoring its wind farms in a framework of condition based monitoring.

To this end, the MIG was divided into two parts. The first part, a week-long, aimed to define forensic engineering and its principal activities - safety regulation, monitoring, and investigation- through numerous visits to the premises of major actors in this industry. Thus, visiting the Commissariat à l'Energie Atomique and attending a lecture from the Bureau d'Enquêtes et d'Analyses pour la Sécurité de l'Aviation Civile illustrated the importance of safety standards and regulations in critical industries. Then, we discovered the role of investigation in improving such safety standards, through the visit of the Institut de recherche criminelle de la Gendarmerie Nationale's laboratories. We were finally introduced to condition-based monitoring by General Electric's engineers. The second part of the project lasted two weeks, these were dedicated to the implementation of algorithms in order to achieve automated monitoring of the wind turbines. First, researchers from laboratories in Sophia-Antipolis gave us some essential basic mathematical tools including an introduction to the Fourier transform and an introduction to machine learning with a presentation of the main algorithms we had to use in our work. We were then given raw data collected over 8 years of sampling two wind turbines monitored by the firm *Prüftechnik*. We dedicated the end of the second week to sorting out, analyzing, and treating this data, to get exhaustive and representative features of the dataset, whether it be statistical features or features from Fourier or wavelet transformations. Finally, we implemented our algorithms, trained them with training datasets we built and worked on optimizing our results.

To be more precise, the first three days of the second week were dedicated to understanding *Prüftechnik's* expectations and demand, learning the mathematical tools required to meet such demand, and discovering lean management methods in order to maximize our efficiency while working. We spent the last two days of the third week documenting our algorithms and finalizing a presentation to introduce the other students to our work. That is why we did most of our work in 7 or 8 days. This working week was very intense and it is important to notice we divided the team into groups, each group working on one or two algorithms: one must not think each student took part in all the things we have developed. On the first day, each group took some time to look at the data to appropriate it. Then, some groups worked on organizing this data to make it easier to manipulate, while the other groups were calculating its main statistical and Fourier features. Finally, for the last three or four days, each group worked on its algorithm, trying first to understand the mathematical theory on which it relies, to then implement and optimize it. During this week, we regularly organized group meetings to check each group's progress and difficulties, give each other some advice and introduce new ideas.

1. INTRODUCTION

Engineering can be defined as the modelling of objects to verify their efficiencies *a priori*. It encompasses a broad range of more specialized fields of engineering, for instance forensic engineering which aims to find out why a certain structure or machine failed and then take appropriate measures to limit the possibility of repeated failures. Forensic engineering is the application of engineering knowledge and practices to determine what went wrong or what could go wrong in this process, it uses reverse engineering to find out why a component, structure or machine failed to perform properly leading to possible injuries and economic losses.

Maintenance is an intrinsic part of forensic engineering. They are both deeply intertwined. On the one hand, forensic engineering tries to improve the performance and life of components, structures, or machines to avoid failures. On the other hand, the purpose of maintenance is to ensure the maximum efficiency and availability of equipment at optimal cost and under satisfactory conditions of quality and safety.

At first, unplanned maintenances were operated only to repair or replace failed units. Today, planned maintenances have replaced unplanned ones: first with scheduled maintenances and then, with condition-based or predictive ones. Condition-based and predictive maintenance interventions are both based on the current state of the industrial component to define whether it should be replaced, repaired or not. The quantification of the current state of the component depends either on the degradation level assessed or on the predicted Remaining Use Life and has to balance the production and safety benefits with the costs of performing maintenance.

For a long time, maintenance was synonymous with an inevitable waste of time and money. But today, due to the evolution of the field, industrials have become aware of its benefits, realizing that maintenance is no longer a chore but an indispensable requirement for optimizing. Among all forms of maintenance, condition based monitoring has proven to be very efficient in the Industry and gives the industrial manager the capability to perform maintenance before serious degradation occurs. This method is a breakdown-prevention technique, requiring no dismantling as it is based on inspection by auscultation of the components. It simply requires continuous observation of the components, done with multiple sensors, in order to monitor the health of the component and to decide whether repair actions are needed based on the degradation level assessed.

This condition based monitoring technique can be implemented to monitor the health of wind farms and more specifically, wind turbines. Indeed, as wind power aims at becoming one of the main renewable energy sources within the upcoming years, wind turbine system reliability is a critical factor in its success. Furthermore, the major issue with wind power systems is the relatively high cost of operation and maintenance. First, wind turbines are hard-to-access structures but they are also often located in remote areas, offshore, for example. That is why unplanned downtime can affect efficiency, productivity and profitability. So, how can operators avoid these machine problems and keep the turbines turning? In such a context, condition based monitoring and fault diagnosis of wind turbines, as described before, might, provided these are accurate, have great economic benefits.

Nowadays, many operators employ periodic inspections for condition assessment based on empirical and subjective measures. However, those inspections are still very expensive and require intrusive and undesired scheduled downtime to be performed.

That is why wind turbine companies are trying to develop condition based monitoring techniques and fault diagnostics in order to deliver quick and reliable information for service crews to act on. The case study has been provided by *Prüftechnik*, a company specialized in alignment system for rotating machinery and condition monitoring systems. It is considered as one of the leaders in maintenance solutions. At *Prüftechnik*, condition monitoring is realized by experts that analyse frequency spectrum of the acceleration and then judge the condition of the machine upon data collected manually by

getting directly next to the rotating machinery or by using a Supervisory control and data acquisition (SCADA) system. More specifically the case study is a wind farm a wind farm that has been facing expensive failures since several years. It deals with the improvement of the treatment and the numerical analyses of data in order to best predict potential failures of the wind turbines, using data-driven algorithms.

The acquisition and storage mechanisms of the SCADA system yields the monitoring signals values measured as N successive and disconnected time periods of equal lengths, $\Delta t_j = te_j - ts_j$, where ts_j is starting time of recording and te_j is the ending time of recording in the period j. Thus $\vec{x}(t) = \left[x_1(ts_1 : te_1), x_2(ts_2 : te_2), \dots, \dots, x_{j-1}\left(ts_{j-1} : te_{j-1}\right), x_j\left(ts_j : te_j\right), \dots, \dots, x_N(ts_N : te_N)\right]$ and $ts_j \gg te_{j-1}$.

On the other hand, the maintenance monthly reports indicated the health state of the generators is also available $\vec{y}(N) = [y_1, y_2, \dots, y_j, \dots, \dots, y_N]$. Thus, the health state of the generator during the $i-th$ time period ($\Delta t_j = te_j - ts_j$) is:

$$y(j) = \{0 \mid normal\ conditions, 1 \mid\ to\ monitor, 2 \mid\ abnormal\ conditions\ \}$$

Therefore, the objective of the present project becomes the development of the model f^c that predicts (detects or diagnoses) the generator health states based on the collected condition monitoring data:

$$\hat{y} = f^c\ (x_j)$$

2. RELATED WORK

How does one optimize wind turbine monitoring? Condition based monitoring relies on the surveillance of industrial plants thanks to multiple data sensors. With a solid database, it has become possible to prevent too serious degradation of a plant before it occurs. This requires detecting divergences in recordings compared to normal conditions.

2.1 Feature extraction and selection

The first steps of the process are feature extraction and selection. “Effective and efficient feature extraction techniques are critical for reliably diagnosing rotating machinery faults” [1]. “The focus of feature selection is to select a subset of variables from the input which can efficiently describe the input data while reducing effects from noise or irrelevant variables and still provide good prediction results” [2].

From accelerometers’ time series that record vibrations, one should build a database that depicts the machinery’s health evolution through time. Apart from the statistical approach, current methods capitalize on frequency analysis with Fourier transform or frequency-time analysis with Wavelets. Mainly used technologies are Fast Fourier Transform (FFT), Short-Time Fourier Transform (STFT), or Continuous Wavelet Transform (CWT). Thus, [3] highlights that “the adaptive, multi-resolution capability of the Wavelet Transform has made it a powerful mathematical tool for diagnostics of machine operation conditions in manufacturing”. The final form of this signal treatment is often a spectrogram (evolution in time of a frequency spectrum).

2.2 Fault detection

Afterwards, fault detection is implemented. Programs have been developed in order to tell whether a given time series has been recorded during a healthy period for the machinery or not, based on the comparison with features extracted in normal conditions. In short, it distinguishes anomalous operation from normal operation.

Fault detection is divided into two parts: the first one is to build a model that reproduces the plant’s behaviour in normal conditions, it is called regression. This model is fed by features extracted and selected as explained above. Mostly used methods are Principal Component Analysis (PCA) and Auto Associative Kernel Regression (AAKR). They are non-supervised algorithms. “PCA is used to define an orthogonal partition of the measurements space into two orthogonal subspaces: a principal component subspace and a residual sub-space” [4]. Regarding AAKR, it “performs a mapping from the space of the measurements of the signals to the space of the values of the signals expected in normal conditions” [5].

At the end, statistical tests are required to measure the deviation from normal conditions and alert the operator if it reaches a threshold. For example, Q-statistics is a quantification of the residual subspace of the PCA model of a dataset. Another approach is Sequential Probability Ratio Test (SPRT) “a statistical method for triggering an alarm with a controlled balance between false and missed alarms” [6].

2.3 Diagnostics

Finally, being able to perform accurate diagnostics is essential to improve maintenance efficiency. In other words, using extracted features labelled with the corresponding fault class, operators are notified when the plant is healthy, partially degraded, degraded, or faulty. We see that the result is more precise than the one given by fault detection and thus is key for accurate health management. Algorithms able to do so are supervised classifiers. Relevant techniques are Support Vector Machine (SVM) which aim is “to devise a computationally efficient way of learning ‘good’ separating hyperplanes in a high dimensional feature space” [7], Convolutional Neural Network (CNN), Autoencoders, K-Nearest Neighbours (KNN), or Random Forest. For example, [8] recently used Continuous Wavelet Transform to “extract the time–frequency image features, i.e., the wavelet power

spectrum. Then, the obtained image features are fed into a 2-D Convolutional Neural Network to construct" the Health Indicator of a machinery.

To conclude, technologies have been developed to monitor the health of the component and then decide whether repair actions are needed based on the degradation level assessed.

Figure 3.1 Summary of methods discussed

Feature extraction	Non supervised fault detection	Supervised fault detection
FFT STFT CWT	PCA with Q Stats AAKR with SPRT	SVM Neural Networks (CNN, Autoencoders) KNN Random Forest

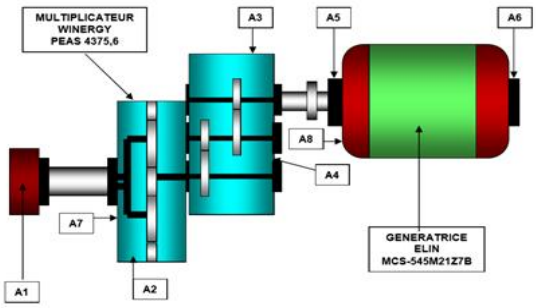
3. DATA

3.1 Raw data extraction

On the one hand, we received from *Prüftechnik* a bunch of data for each wind turbine. The data collected the measure of the acceleration over 5 seconds during 6 years from September 2013 to September 2019. It amounts to 2188 measures but they are not regular and long inactivity periods exist, notably due to failures in the transmission of data via Wi-Fi. Each measure lasts 5 seconds and the sample rate is 13 kHz, due to the Shannon theorem the Nyquist frequency I around 6.5 KHz. It also contains the value of the wind speed during the same period. However the acceleration and speed sensors have a different clock rate so the measures were not synchronized. The data points of a specific sensor were all gathered in one file, so the first task was to extract each measure into a new file. We also created a Python module that calculates the approximate wind speed at the time of the measure.

The acceleration measures were taken on 6 sensors named A1 to A6. At first sight, we looked into the condition of the Generator so we only considered the values of sensor A6. We also initially focused on the Wind Turbine 1380 out of the 8 Wind Turbines of the wind farm, because it is the one that has the highest number of measures.
On the other hand, we received a label file that registers the condition of the wind turbine during each month from March 2011 to October 2019. The labels were constructed according to the monthly reports from *Prüftechnik* of the wind farm. However some condition points were missing so after deleting the data where the condition of the wind turbine was absent, we had 1275 usable measures. We were then ready to extract the features of each measure and train our various algorithms.

Figure 3.3.1 Position of sensors on the wind turbine



3.2 Statistical features

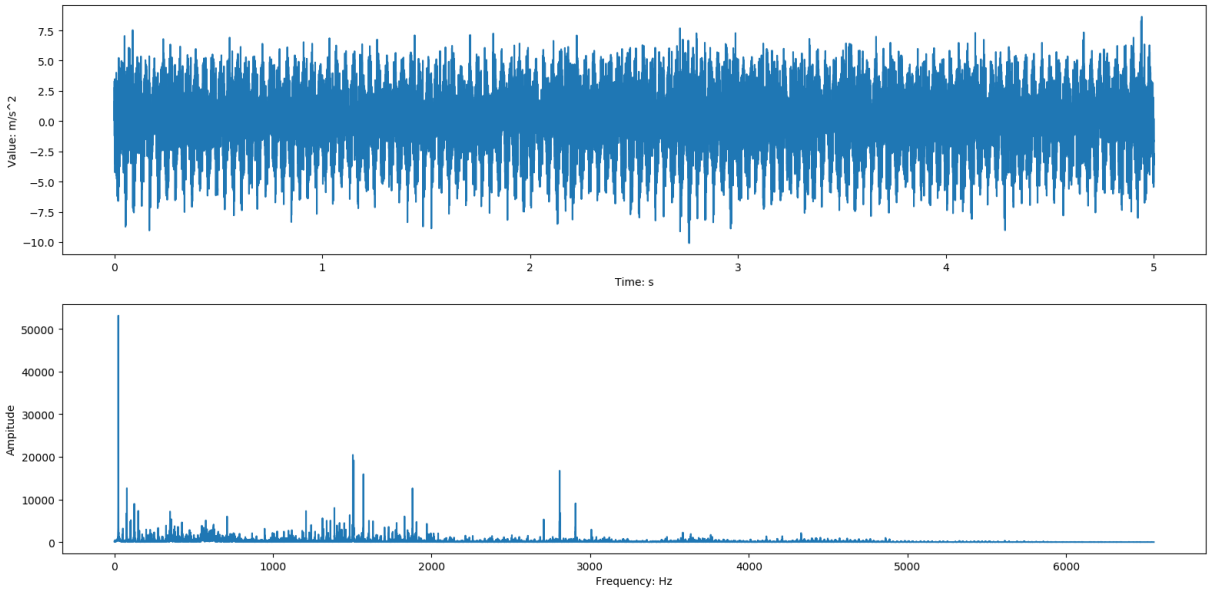
Seeing as the data we have isn't really intelligible, we wanted to extract the key features to represent it. We began with statistical features. As opposed to what we did later with Fourier and wavelet transformations, this step is quite straightforward.
Because the more features we extract, the more accurate we are, we want this step to yield as many statistical features as possible. Here is an exhaustive list of the features we extracted: mean, standard deviation, skewness, kurtosis, first quartile, median, third quartile, minimum, maximum.
Of course, some features are more important than others: for instance, in the case of a circular acceleration, we expect the mean to be far less significant than the standard deviation. This brings us 2

3.3 Fourier features

In addition to the statistical features of the data of acceleration, by considering the series of points in the data as a numerical signal with a certain sampling rate, we were able to apply the Fourier transform. The results will be utilized for dynamic analysis and further features extractions.
The Fourier transform (FT) decomposes a function of time (a signal) into its constituent frequencies. It is a complex-valued function of frequency, whose magnitude (modulus) represents the amount of that frequency present in the original function, and whose argument is the phase offset of the basic sinusoid in that frequency.

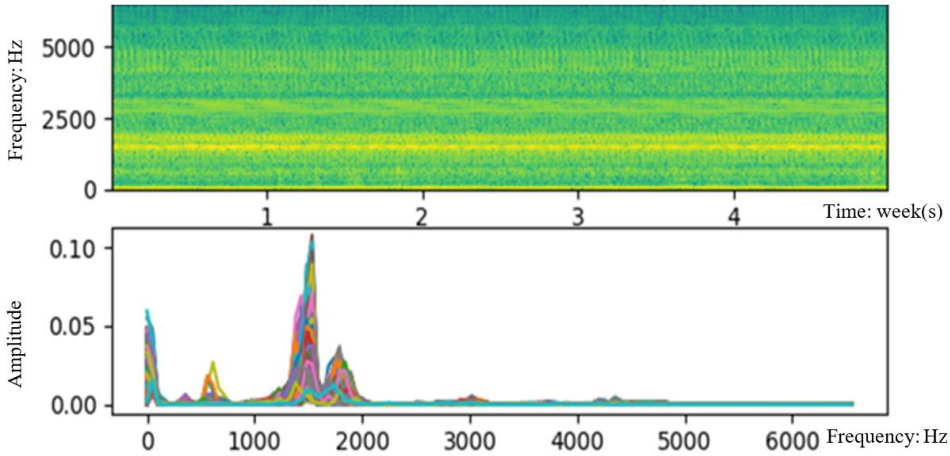
Here we have used the method called fast Fourier transform (FFT), which manages to significantly reduce the complexity of computing. Later we have also tried with the Sparse Fast Fourier Transform (SFFT). As the results calculated by the program are complex numbers, to visualize the spectrum, we take the module of the coefficients.

Figure 3.3.1 – One file of raw data and its spectrum of Fourier Transform



Using consecutive raw data of one whole month, we combined them by time order and visualized the general variation of the Fourier transform spectrum, which enabled us to use an evident and intuitive way to compare the Fourier transform features between different operating status of the wind turbine.

Figure 3.3.2 – Features extracted by SFFT from one month of recordings



From the figure above we observe that during a period when the wind turbine functions healthily, the spectrums are almost in a unanimous distribution pattern. To the contrast, during the month when the wind turbine was not in the perfect status, the pattern fluctuates. Therefore, the Fourier transform features can be considered as one of the usable dynamic features.

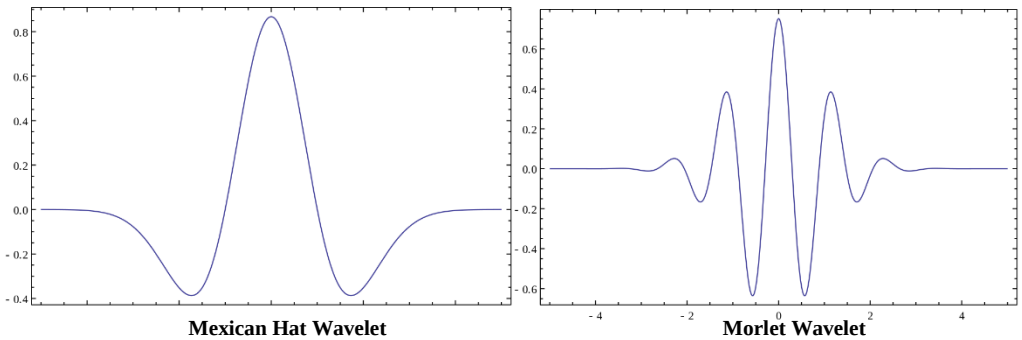
The usable features need to have a relatively low number of dimensions to feed the algorithms. That is why we did not use the Fourier Spectrum directly; however, we divide the spectrum into different intervals according to frequency, and in each frequency interval, we calculated the integral of the spectrum amplitude to access the energy in a specific bandwidth of frequency. The observation of a spectrum in normal and abnormal conditions told us that a division in 10 windows allows us to separate each relevant peak. Therefore, the corresponding features are named sample0 to sample9.

3.4 Wavelet features

3.4.1 Description of the wavelet

The FFT is only localized in frequency, for it is a projection of the signal on sinusoidal functions. Both SFFT and wavelet transform are localized in time and frequency, which enables to represent time-frequency diagrams. The wavelet transformation is based on the projection of the signal on a family of functions named wavelets. A wavelet is an oscillation that looks like a wave. It is what could be called a brief oscillation, close to the ones that can be observed by a heart monitor. The wavelet has, unlike the sinusoidal functions of the Fourier transform, a limited energy. Its amplitude begins and ends at zero. Several families of wavelets exist.

Figure 3.4.1.1 – Examples of wavelet families



Such families are created from what is called one mother wavelet ψ that is scaled by a strictly positive factor *a* (the scale) and translated by a factor *b* following the expression:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right)$$

3.4.2 Continuous Wavelet Transform

For the continuous wavelet transform, the subspace of scale *a* which is actually the frequency band [1/a,2/a] can be generated by such $\psi_{a,b}$. The projection of the signal x onto the subspace of scale *a* is then:

$$x_a(t) = \int_R \quad WT_\psi\{x\}(a,b) \cdot \psi_{a,b}(t)\, db$$

with the wavelet coefficients:

$$WT_\psi\{x\}(a,b) = \langle x, \psi_{a,b}\rangle = \int_R \quad x(t)\psi_{a,b}(t)\, dt$$

These coefficients can be assembled in a time-scale diagram, and so create a time-frequency analysis.

3.4.3 Discrete Wavelet Transform

For the discrete wavelet transform, given $a > 1$ the scale and $b > 0$ the factor of translation, the child wavelets are given as:

$$\psi_{m,n}(t) = \frac{1}{\sqrt{a^m}}\psi\left(\frac{t-nb}{a^m}\right)$$

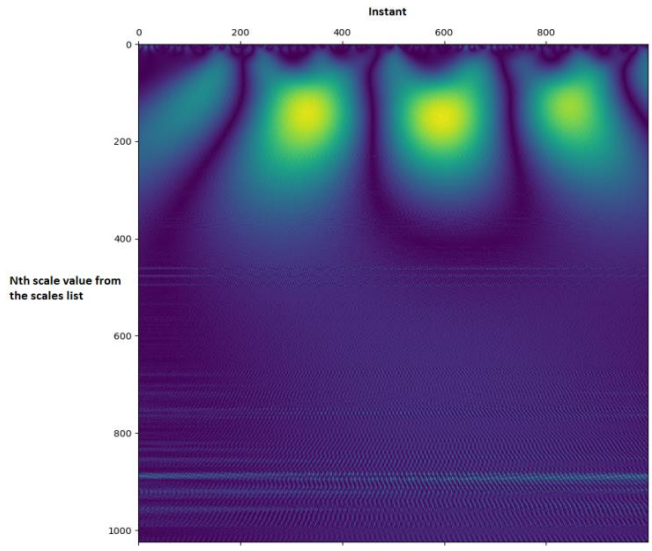
If $\{\psi_{m,n}, m \in Z, n \in Z\}$ forms an orthonormal basis of $L^2(R)$ (the square-integrable functions), then the reconstruction of the signal x can be expressed as follows:

$$x(t) = \sum_{m\epsilon Z} \quad \sum_{n\epsilon Z} \quad \langle x, \psi_{m,n}\rangle \cdot \psi_{m,n}(t)$$

3.4.4 Implementation

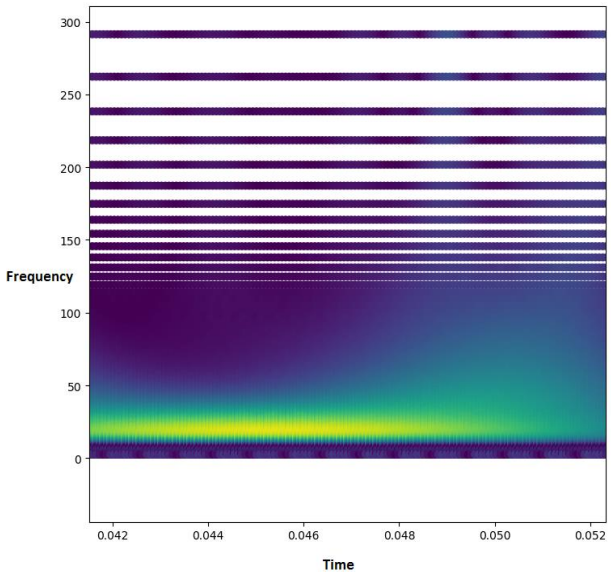
The wavelet transform used on the data was a continuous one, from the python module *PyWavelet*.
The function *PyWavelet.cwt* takes in argument the data which is in our case the acceleration, the sampling period, the family of wavelet that is used for the transformation and the different values of the scale that are going to be used. It returns a matrix time-scale of coefficients and a list of frequencies.

Figure 3.4.4.1 – Time-scale diagram for a set of acceleration data



As the scale is directly linked to the frequency, for each scale of the diagram corresponds one frequency of the list returned by the function. So, it is possible to create a spectrogram localized in time with this module. However, only a discrete number of frequencies is representable as a discrete number of scales is chosen.

Figure 3.4.4.2 – Time-frequency spectrogram for a set of acceleration data



The wavelet transform is therefore a powerful tool for time-frequency analysis. There are two ways of using the result:

- Analysing the spectrogram in frequencies, for example extracting features such as the most important frequency at a time.
- Using the time-scale diagram, which has the advantage to be treatable as a picture. This is interesting for methods like neural networks.

3.5 Dataset description

3.5.1. Dataset description

We gathered “wind speed”, statistical features, and Fourier features in one dataset, each line corresponding to an acceleration file. We added a last column named ‘Status’ which contained a label for each file, corresponding to the status of the equipment (see 3.2).

Figure 3.5.1.1 – Picture of the first rows of the dataset

	Date	Wind_speed	compteur	kurtosis	skew	count	\
0	2014-10-14 00:01:07	4.389751	288	0.584276	-0.100751	1638.0	
1	2014-10-14 00:01:07	4.389751	259	0.459943	0.027480	1638.0	
2	2014-10-14 00:01:07	4.389751	203	0.285982	0.100646	1638.0	
3	2014-10-14 00:01:07	4.389751	226	0.531725	0.105146	1638.0	
4	2014-10-14 00:01:07	4.389751	216	0.052788	0.077594	1638.0	

	mean	std	min	1q	...	sample1	sample2	sample3	\
0	0.002963	1.194432	-4.914753	-0.729746	...	0.069857	0.509968	0.265569	
1	0.014762	1.092781	-4.314058	-0.672441	...	0.071216	0.443540	0.206340	
2	0.003735	1.172515	-4.012893	-0.803280	...	0.080984	0.527040	0.210498	
3	0.006108	1.180338	-5.597373	-0.761878	...	0.077824	0.546460	0.265649	
4	0.017324	1.229622	-4.222382	-0.805004	...	0.073988	0.645914	0.200357	

	sample4	sample5	sample6	sample7	sample8	sample9	Status
0	0.148472	0.037507	0.242408	0.036956	0.002806	0.000244	2
1	0.125913	0.025536	0.147934	0.038242	0.004194	0.000406	2
2	0.144414	0.022464	0.181715	0.043266	0.004491	0.000250	2
3	0.172304	0.036126	0.145384	0.038205	0.003216	0.000256	2
4	0.162138	0.026829	0.221383	0.042478	0.003742	0.000785	2

We worked on two datasets. The first one is named “Milano” and is composed of 3 months of data. In this dataset, all labels are equally represented and this will influence our algorithms’ performances. The other one is named “main” and contains features’ evolutions during 6 years.

3.5.2 Dataset split

In order to use supervised classifiers, we had to split our dataset in a training set and a test set. Commonly used proportions are 75% of training and 25% of testing. Classifiers need to be trained on a set in which all labels are represented, to make sure these classifiers will be able to recognize each status of the wind turbine during the test phase. To ensure that, the dataset rows are shuffled before the split, we used [9]’s *train_test_split* function.

4. METHODS

4.1 Developing the methods

In the framework of prognostics health management and condition-based monitoring of our wind turbines, various algorithms were implemented and their corresponding results were compared in a fault detection scheme. Each being very rich either in terms of mathematical theory or numerical implementation, more detailed papers explaining each method individually have been provided at the end of the report.

At this point, it is useful to note that two types of machine learning algorithms were used, supervised, semi-supervised. For the supervised methods, we need a training dataset including labels with correct classification (whether the turbine is working or not working) and the algorithms “learn” different methods of separating our dataset. In the supervised one we find the SVM (Support Vector Machines), Random Forest, K-NN (K nearest neighbours), Convolutional Neural Network and Autoencoder. In the semi-supervised category we include the AAKR (Auto Associative Kernel Regression) and the

PCA as it only needs to "learn" on a healthy dataset whether the others need more balanced datasets in terms of classification to be able to function correctly. CNN and Autoencoder are based on different types of neural networks, and were both used on wavelets generated from our original data.

It is useful to note that these methods are often more effective when used together, for example PCA can be used for dimensionality reduction by selecting the most interesting features in the dataset, these selected features can in turn be taken for the other machine learning algorithms.

Our original development method was to first implement methods based on existing literature and use existing python modules when possible. Using the Milano Dataset, a first approach dataset. At the end, we ran our algorithms on the main dataset to study results in the hypothesis that the method had reached its full potential in terms of training.

4.2 Testing the algorithms

To compare the different algorithms, we needed to decide what tests would be relevant. *Prüftechnik* is interested in detecting or diagnosing the condition of wind turbines in a wind farm in order to reduce costs of maintenance. So on the one hand the company doesn't want to miss potential failures, because if they really occur, it will cost a lot to repair the wind turbine. On the other hand, as explained by *Prüftechnik's* expert, the maintenance of a wind turbine is uncommon, and usually needs many investments, so the expert should not be assaulted by a huge amount of fake failure alerts. That's why a compromise needed to be found between the Recall R (first situation) and the Precision P (second situation). A good indicator of this compromise is the f1-score, which is frequently used to characterize test efficiency. The closer to 1 the f1-score is, the better.

$$P = \frac{|True\ Positives|}{|Positives|}$$

$$R = \frac{|True\ Positives|}{|True\ Positives| + |False\ negatives|}$$

$$F_1 = 2.\frac{P.R}{P+R}$$

5. RESULTS AND DISCUSSIONS

5.1 Results

Here, we will focus on the results obtained with the final dataset sampled by sensor A6 for 8 years. These results are not as good as the results from the "Milano" dataset. These results are more reliable because they show how these algorithms would act in real conditions. For example, the Random Forest's F1-score on Orange labels steps down from 100% to 95%. As to the labels, we removed the Yellow labels. We noticed that the accuracy, and so the F1 score, is better by keeping only the Green labels and the Orange one. The Yellow labels seem to be too subjective to be trustworthy, the data points, which are labelled Yellow, may be Green or Orange, it only depends on the point of view of the one who classified the data. For example, the F1 score of the Random Forest algorithm gave:

- Green: 0.96 %
- Yellow: 0.89 %
- Orange: 0.93 %

These results are acceptable but don't reach the same level of accuracy as when only two labels were used. Here is the summary of the results.

Table 5.1.1 Accuracy, Recall and F1 Score of the different methods

	Orange			Green		
Method	**Accuracy**	**Recall**	**F1 Score**	**Accuracy**	**Recall**	**F1 Score**
Random Forest	0.97	0.92	0.95	0.97	0.99	0.98
PCA	0.46	0.19	0.27	0.74	0.91	0.81
KNN	0.55	0.65	0.67	0.92	0.72	0.81
SVM	0.55	0.65	0.67	0.92	0.72	0.81
AAKR + SPRT	0.77	0.91	0.84	0.62	0.34	0.44
CWT + CNN	0.58	0.87	0.70	0.96	0.84	0.90
AutoEncoder	0.21	0.92	0.34	0.99	0.91	0.95

We can notice that the results are very variable between these algorithms. Some are far better in Orange label detection than Green label detection such as the AAKR method. But most perform well in the Green label detection. All things considered, all these results are admissible, as told by a *Prüftechnik's* expert. In fact, we don't need a huge accuracy regarding fault detection; the time lapse needed to detect a default on a wind turbine is relatively wide, a couple weeks typically. We don't need to react as fast as we would have to with a disease or airplane. The dysfunction of a wind turbine is not too serious, and in the end these equipments are not regarded as "criticals" by maintenance experts, in this sense that a failure implies financial loss, but most probably not fatalities and does not jeopardize, as such, the whole business of the operator. The need for multiple measurements is not a problem if the result is reliable.

An algorithm takes the lead in every statistical measure: the Random Forest. This algorithm blows every other in our study. The AutoEncoder algorithm tends to have the same score in the Green label detection, the Random Forest beats it in the Orange Label detection. By taking into account the other advantages of this algorithm, it seems to be the most reasonable choice. It has a short training and prediction time and does not need a lot of features to be fed. And the features it used are understandable for an expert, that allows to consider a functional use of this algorithm by this expert.

5. 2 Limitations

Time was a limiting factor in our project. This section presents the points we would have liked to look further into, had we had the time.

First, the more accurate the labels are, the better our classifiers work. That is why the period of time over which the wind turbine is labelled as orange should be as precise as possible. As of right now, though, the wind-turbine is checked at most every month, which is not ideal. To solve this problem, we would have liked to entirely re-label our data using semi-supervised methods, such as PCA or AAKR, which would allow us to have a more precise idea of the moment the wind turbine started to malfunction.

A possibility could also have been to make a classifier englobing every single classifier we trained and having them vote, possibly with different weights: this would most likely result in a slightly better classifier.

Having a bigger set of data would also have helped, for our neural networks seem to have been a bit undertrained, even though that would have meant even more training time.

We want to insist on the fact that we assume in this study that the expert was always right. Results have to be understood as the capability to reach the same conclusions as the experts who also get interested into raw data. A more precise study could be made and the possibility to have better result than the expert is an eventuality.

6. RECOMMENDATIONS TO *PRÜFTECHNIK*

This last part aims at emphasizing some key elements we highlighted optimizing our results, and which can interest a wind farm-monitoring company.

To start with, all the methods we used have shown slightly good results, whether it be semi-supervised methods (AAKR, PCA) or supervised ones (the other ones). Indeed, we managed to obtain a minimum precision, which may be equivalent to the accuracy of an expert, even higher with some methods such as Random Forest. Furthermore, some of these methods (SVM, PCA, autoencoder and Random Forest) achieve feature selection, which means they are able to select a subset of features representative of the data they are trained with. Such a selection is interesting for a company, as it reduces the quantity of data it has to measure, store and process, and the quantity of sensors it has to set up. Moreover, even though the features outputted by SVMs, PCA and Autoencoders are not very understandable, the ones given by Random Forest (wind speed, different samples, FFT) are meaningful to an expert as they represent a physical scale. Indeed, this expert may be disturbed by the absence of Wind Speed for some algorithms like PCA or AAKR, providing the same results. Thus, an expert can confirm the diagnosis is likely to be true, and can even learn from the machine, which can detect patterns or focus on features the expert had not studied.

To sum this up, using these algorithms could bring a real value to the company. Indeed, they are initially complementary to the expert's work: as explained above they indicate the expert the features he has to focus on for the diagnosis. To go further, they can enable to reduce the number of expert necessary to the monitoring of a wind farm by simplifying the work of the remaining ones indicating them which windmills they have to focus on particularly: even if this depends on the precision of our algorithms, when a windmill is classified as orange, it is more likely to fail then one classified as green. Therefore, to maximize money savings, it is wiser that the expert check such windmill first, to take a decision and start or not a maintenance, and thus, an expert can be in charge of more windmills But, one must not forget some information can be lost with some algorithms, such as SVMs, PCA or autoencoders: the feature they select are not clearly understandable and can sometimes go against the traditional features an expert is used to study.

It is also important to notice both semi-supervised and supervised methods can be useful to *Prüftechnik*. Indeed, semi-supervised methods, PCA and AAKR, only require small amounts of data, representative of a healthy windmill. Therefore, they can be implemented after a few weeks of normal running of the wind turbine. Here, it is important to precise the different thresholds that are used in these methods can be modified and will influence the different precision, recall and f1-score of the algorithms. Depending on what is the most expensive for the company (repetitive and quite useless maintenance if healthy windmills have been diagnosed as unhealthy, or missed maintenance if damaged windmills are classified healthy), you can adapt the different thresholds to try and minimize your maintenance expenses. As for the supervised methods, they require a data set spanning all the weather conditions a windmill will face (wind speed varies along the year for example), and its different states of health. Therefore, they can't be implemented as soon as supervised methods, but most of them, especially Random Forest, were shown very efficient. Besides, we also recommend not to put aside CNN method. Indeed, neural networks are really efficient when they are trained with huge amounts of data. Thus, taking some time to develop a neural network perfectly shaped to your situation (thing we could not achieve for a lack of time) and training it with huge amounts of data could offer you a method surpassing the other ones.

Finally, we must precise the way the data, given to us, were labelled is one of the main problems we faced optimizing our algorithms. Indeed, the state of health of the windmill being labelled each month, it was really hard to characterize precisely the signs forewarning of a possible future failure, especially in the case of three-classes diagnosis, where the separation between green and yellow was not very clear. Therefore, in view of developing and implementing data analysis based on machine learning, we recommend the company to label the state of health of the windmill more regularly, each week for example. As PCA method is non-supervised and rather efficient in classifying the different states of health of the windmill, we recommend the company to relabel its data with PCA, before inputting them into the supervised algorithms. Finally, a study on the dependence time-Fourier spectrum on each 5 seconds sample showed us the Fourier spectrum does not change in 5 seconds. That is why making more numerous but shorter samples could be useful to detect variations in the state of health of the windmill, to improve the precision of our algorithms, to enable *Prüftechnik* to implement an efficient and automated condition-based monitoring.

If the company wants to go way further into developing new-tech condition monitoring solutions, the next step would be to use Artificial Intelligence to calculate the Remaining Useful Life. This would be the final solution to really optimize the maintenance organization. To conclude, here is a list of the different recommendations *Prüftechnik* should follow to implement an efficient and automated condition-based monitoring:

- Making shorter but more regular samples.
- Labelling the data on shorter periods to follow more precisely the evolution of the state of health of the windmill.
- Using first semi-supervised algorithms, to start monitoring the windmills quickly, and relabel properly the data. Then, using supervised algorithms with data labelled precisely.
- Implementing feature selection to focus on the features characterizing the state of health of the windmills to reduce the amount of data it has to process and store.
- Adapt the actual parameters to fit them to the level of optimization it wants to reach. At the end the goal will be not to maximize the accuracy of the algorithms, but to minimize economic loss.

Of course, we do not pretend that we have identified all the elements that could be improved, and that the advice we give above and our algorithms are optimized. It belongs to *Prüftechnik* to decide whether to follow them, depending on its policy and considering the benefits it could get generalizing the results of some eventual attempts on a small scale.

7. REFERENCES

[1] H. Yang, J. Mathew, et L. Ma, “Vibration feature extraction techniques for fault diagnosis of rotating machinery: a literature survey”, présenté à Asia-Pacific Vibration Conference, 2003.

[2] G. Chandrashekar et F. Sahin, “A survey on feature selection methods”, *Computers & Electrical Engineering*, vol. 40, nº 1, p. 16-28, janv. 2014.

[3] R. Yan, R. X. Gao, et X. Chen, “Wavelets for fault diagnosis of rotary machines: A review with applications”, *Signal Processing*, vol. 96, p. 1-15, mars 2014.

[4] B. Mnassri, E. M. E. Adel, B. Ananou, et M. Ouladsine, “Fault detection and diagnosis based on pca and a new contribution plot”, *IFAC Proceedings Volumes*, vol. 42, nº 8, p. 834-839, 2009.

[5] P. Baraldi, F. Di Maio, P. Turati, et E. Zio, “Robust signal reconstruction for condition monitoring of industrial components via a modified Auto Associative Kernel Regression method”, *Mechanical Systems and Signal Processing*, vol. 60-61, p. 29-44, août 2015.

[6] F. Di Maio, P. Baraldi, E. Zio, et R. Seraoui, “Fault detection in nuclear power plants components by a combination of statistical methods”, *IEEE Trans. Rel.*, vol. 62, nº 4, p. 833-845, Dec. 2013.

[7] N. Cristianini et J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge: Cambridge University Press, 2000.

[8] Y. Yoo et J.-G. Baek, “A novel image feature for the remaining useful lifetime prediction of bearings based on continuous wavelet transform and convolutional neural network”, *Applied Sciences*, vol. 8, nº 7, p. 1102, juill. 2018.

[9] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python”, *Journal of Machine Learning Research*, oct. 2011.

[10] Sharma, “Autoencoder as a Classifier” *DataCamp Community*, 20-Jul-2018. [Online]. Available: https://www.datacamp.com/community/tutorials/autoencoder-classifier-python. [Accessed: 18-Nov-2019].

[11] Ng, “Sparse autoencoder” from *CS294A Lecture notes*, 2011, vol. 72, no 2011, p. 1-19. [Online]. Available: https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf. [Accessed: Nov. 20, 2019]

[12] G. E. Dahl, T. N. Sainath, et G. E. Hinton. “Improving deep neural networks for LVCSR using rectified linear units and dropout”. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8609-13, 2013. https://doi.org/10.1109/ICASSP.2013.6639346.

[13] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2015 [Online], pp. 1440-1448.

[14] Alain Rakotomamonjy, *Variable Selection using SVM-based criteria,* Journal of Machine Learning Research, 3/03/2003

[15] "Karush–Kuhn–Tucker conditions," *Wikipedia*, 18-Nov-2019. [Online]. Available: https://en.wikipedia.org/wiki/Karush–Kuhn–Tucker_conditions. [Accessed: 19-Nov-2019].

[16] S. Wold, K. Esbensen, et P. Geladi, « Principal component analysis », *Chemometrics and Intelligent Laboratory Systems*, vol. 2, nº 1-3, p. 37-52, août 1987.

[17] E. Zio, X. Lu, "Part 2A: Fault Detection - Model That Reproduces Plant Behaviour in Normal Condition", *Statistical Methods of Risk Analysis*. [Lecture].

[18] E. Zio, X. Lu, "Part 2B: Fault Detection - Statistical Test", *Statistical Methods of Risk Analysis*. [Lecture].

[19] A. Cutler, D. R. Cutler, and J. R. Stevens, "Random Forests," in *Ensemble Machine Learning: Methods and Applications*, C. Zhang and Y. Ma, Eds. Boston, MA: Springer US, 2012, pp. 157–175.

[20] W. Koehrsen, "Random Forest Simple Explanation," *Medium*, 27-Dec-2017. [Online]. Available: https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d. [Accessed: 03-Dec-2019].

[21] Italo José, "KNN (K-Nearest Neighbors) #1", *towards data science*. [Online].[Accessed: 08-Dec-2019].
Available: https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d

METHOD 1

Autoencoder with fully connected ANN

Abstract: In this method we implement an Autoencoder in the form of a double-layers fully connected artificial neural network, which selects features through unsupervised learning while using different input datasets. After the construction and training of the encoder layer and decoder layer, we keep only the encoder section to extract features; at its output, we add a dense layer and a classifier layer in order to classify the data recordings into two classes: healthy (marked by green) and abnormal (marked by yellow) operation status. The whole network would serve to detect the operation status of a wind turbine according to the acceleration sensor data.

Keywords: Autoencoder, ANN, Classifier

1. INTRODUCTION

Aiming at implementing a numerical monitoring model for the operation status of a wind turbine, we consider the construction of an artificial neural network trained by classified data. One major issue is that we have no idea which data features are representative of the health of the wind turbine, so that we cannot determine or optimize the structure of such a network a priori. Therefore, instead of aimlessly testing different network structures, we use an autoencoder to extract the potential features hidden inside the data. This non-supervised machine learning process will simplify the construction of the network.

Inspired by the CNN-autoencoder-classifier structure given by *Sharma A.* [10], we add a classifier structure after the layer of extracted features, which will classify the records into 2 types of results. In this step, we use the labelled dataset to train the entire network while fixing the weight coefficients in the encoder structure. After the training of the second section, we will rerun the supervised machine learning process without limits on the coefficients for finest tuning.

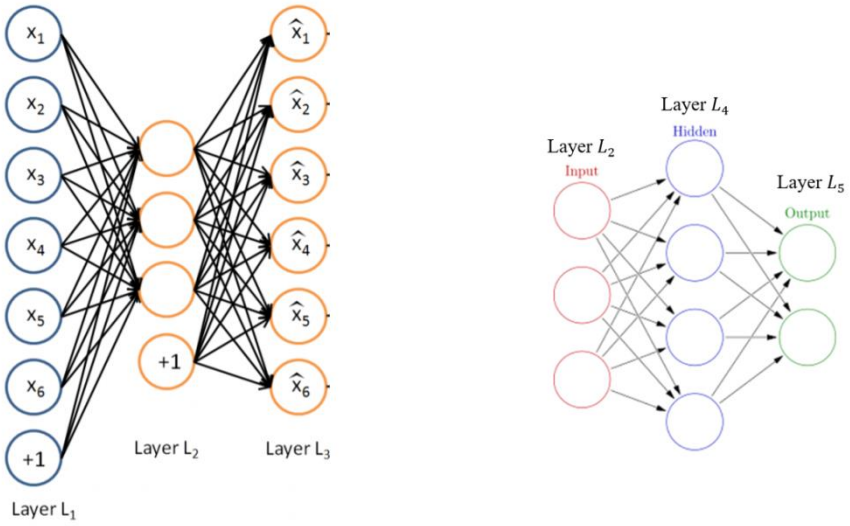
2. DESCRIPTION OF THE METHOD

2.1 Network construction

The autoencoder-learning algorithm is one approach to automatically learn features from unlabelled data. The general structure used here is the same as a fully connected artificial neural network. Each layer of the network is composed by a certain number of neurons, or called variables. Each "neuron" is a computational unit that takes as input x_1, x_2, $x_3, \ldots, x_N$ (and a +1 intercept term), and outputs $h_{W,b}(x) = f(W^T.x + b) = f\left(\sum_{i=1}^{N} W_i.x_i + b_i\right)$, where $f: R \to R$ is called the activation function[11].
The aim is to learn an approximation of the identity function, so that input and output are as similar as possible. The identity function seems to be a particularly trivial function to learn; but by placing constraints on the network, such as limiting the number of hidden units, we can force interesting data features to appear. Here is a general structure:

Layer L_1 is the input data. Layer L_3 is the decoder output layer, with same dimension than the input layer. Layer L_2 is the hidden layer, with an inferior number of neurons as a training constraint. With the same data at the input and output layers, the "compressed" information contained in the hidden layer L_2 can be considered as the features automatically selected by the network itself without any existing labels of the dataset.

Figure 2.1.1 – General structures of an autoencoder section (left) and of a classifier section (right)



Then, we delete the decoder layer L_3 and add the classifier structure. It is composed of 2 layers: the hidden dense layer L_4 and the output layer L_5. The dense layer is composed of a superior number of neurons than the layer L_2. As for the last layer, it has only two variables corresponding to the functioning status: normal or abnormal.

2.2 Training data

To discover the hidden features, we have tested different types of data. For example:

- Raw acceleration data, with 65536 input variables, resized into 256x256 pixels pseudo-images.
- Spectrum after Fourier transform, with 32768 input variables, resized into 128x256 pixels pseudo-images.
- Statistic features extracted, with 12 input variables
- Wavelet features, with 2714 input variables, resized into 46x59 pixels pseudo-images

The original datasets are monthly categorized into 3 types: green for normal function, yellow for warning status, orange for operation failure. But we found that for our method, the yellow labels do not seem to be separated enough from the other labels. We have therefore decided to use only the groups “green” and “orange” for training. We initially tried to train the autoencoder with different input data (raw, Fourier, statistics features, wavelet). Then we used the input that responded best to the autoencoder part to train the classifier.
In order to balance the number of the two types of data during the training, we will randomly select 20000 samples of each type for training. The remaining part of the data will be used as testing data. Due to the limit of access to powerful capability of calculation, we have tried only one set of (epochs, batch size, learning rate). We have set the number of epochs as 400; for each epoch, we used a batch of 50 samples for training. We fixed the learning rate to 10^{-5}. For all layers we used the ReLu [12] activation function, except the last layer for which we used SoftMax activation function [13].

3. IMPLEMENTATION

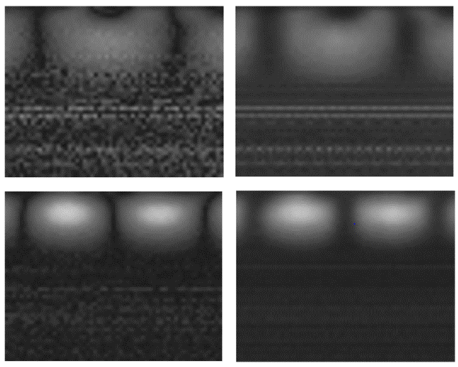
We have used the package “tensorflow.keras” (included in TensorFlow version higher than 2.0) of python to build and train the artificial neural network.

4. RESULTS

In the first step, our goal is to reconstruct the “pseudo-images” of input data through the autoencoder. Unfortunately, for the raw data and the Fourier spectrum, as the dimension of each recording is too large, we did not have enough data to train well the autoencoder. The reconstructed images are the same grey images. For the statistic features, there was no significant difference between each recording, thus the autoencoder ignores the difference between the input data and returns always the same image.

The only dataset that has worked is the wavelets features. With an appropriate dimension of input layer and a sufficient quantity of training data, the reconstruction succeeded. For these data, the dimension of the intermediate layer is 1357, half of the input dimension.

Figure 3.1.1 – Original data (left) and reconstructed data (right) by autoencoder



In the next step, we built the classifier with a dense layer of 1500 neurons and an output layer of 2 neurons. The statistical performance of the model is shown as follows, the testing dataset contains 44000 green recordings and 1150 orange recordings.

Figure 3.1.2 – Performance of the model

	Precision	**Recall**	**f1 score**
Green status	0.9977	0.9074	0.9504
Orange status	0.2059	0.9188	0.3364

We observe an acceptable accuracy of the model. For the green status, we have a surprisingly high precision and f1-score. However, for the orange cases, we have an extremely low precision and f1-score. This might due to the unbalanced size of 2 types of recording among testing data and the lack of precision of the data labels.

From here we can observe that this model is very strong at detecting a potential failure, but with a very high rate of false alarms, which means nearly 4 among 5 alarms will be in fact false.

METHOD 2

Fault diagnosis using Convolutional Neural Networks with Continuous Wavelet Transform

Abstract: This work focuses on the use of Convolutional Neural Networks (CNNs) to process the vibration signal of a wind turbine's generator and diagnose potential faults. To convert the one-dimensional vibration signals to a two-dimensional (2-D) image, the Continuous Wavelet Transform (CWT) extracts the time–frequency image features, i.e., the wavelet power spectrum. Then, the obtained image features are labelled according to the health of the wind turbine ("Orange", "Yellow", "Green"), and fed into a CNN. Once the CNN is trained, we test it on a different data set, and compare predicted values to the actual labels.

Keywords: Continuous Wavelet Transform (CWT), Convolutional Neural Networks (CNN), Convolution, Backpropagation

1. INTRODUCTION

The aim of our work is to create a tool for automatic diagnosis of a wind turbine. The data given by company *Prüftechnik* consist in acceleration values of a sensor placed on the generator of the wind turbine. The sensor makes daily measurements of 5 seconds and at sampling frequency of about 13kHz. Those daily measurements span from September 2013 to September 2019. Given the relatively large amount of given data, it is coherent to try using Convolutional Neural Networks to detect and diagnose faults. CNNs are indeed known to be very effective when trained over a large set of data. Since CNNs usually take images as inputs, using this method also enables us to work on inputs that can be easily understood and visualized by the expert.
To train the CNN, we chose to use Continuous Wavelet Transforms of the signal. This tool returns scalograms that vary noticeably between healthy and faulty signals.

2. DESCRIPTION OF THE METHOD

2.1 Continuous Wavelet Transform

Using different kinds of wavelets, we found that the First Derivative Gaussian Wavelet returned images that could be rather easily classified as "Greens", "Yellows" or "Oranges".

2.2 Convolutional Neural Networks

In deep learning, a Convolutional Neural Network is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. The structure of the CNN is determined by the number of 2D convolutional layers, of Max Pooling layers, and of Dense layers in the network:

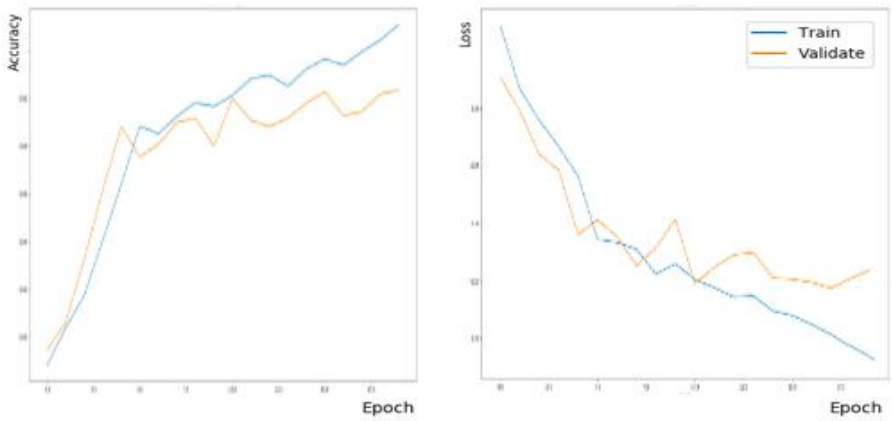
- Max Pooling: Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.
- 2D Convolution layer: 2D convolutional layers take a three-dimensional input, typically an image with three colour channels. They pass a filter, also called a convolution kernel, over the image, inspecting a small window of pixels at a time, for example 3×3 or 5×5 pixels in size, and moving the window until they have scanned the entire image. The convolution operation

calculates the dot product of the pixel values in the current filter window with the weights defined in the filter.

- Dense layers: A dense layer represents a matrix vector multiplication. The values in the matrix are the trainable parameters, which get updated during backpropagation.

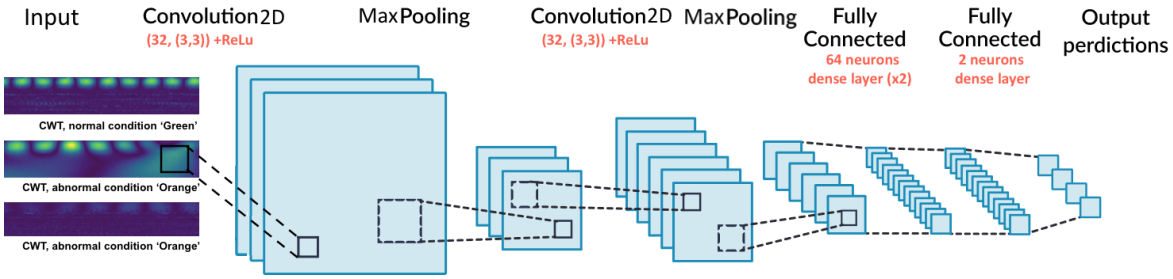
A number of 20 epochs were chosen to train our model. As the graph shows, one could have chosen to implement a higher number of epochs as the accuracy value keeps increasing and the loss value keeps decreasing:

Figure 2.2.1 - Training and validation accuracy (left) and loss (right)



By adjusting the hyperparameters of the CNN, we found that the most satisfactory configuration was as follows:

Figure 2.2.2 – CNN Architecture



Other hyper-parameters we could have modified are the learning rate -The amount that the weights are updated during training-, and the batch size -a hyperparameter of gradient descent that controls the number of training samples to work through before the model's internal parameters are updated-, but we did not have enough time to take a look at those hyper-parameters.

3. SENSITIVITY ANALYSIS

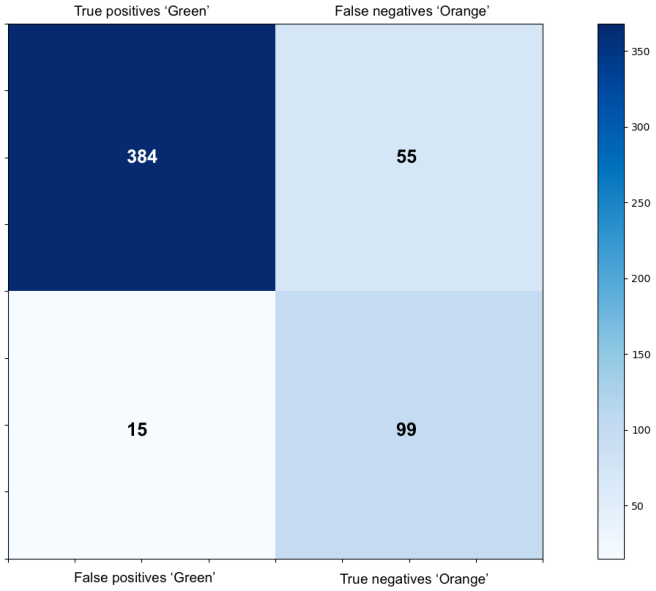
Increasing too much the resolution of the CWT leads to excessive calculation times, and diminishing it too much leads to low image quality, and lower quality data.

Regarding the CNN's structure, we found that adding too many Dense layers leads to a higher calculation time and a higher error rate. Two Dense layers of 64 neurons turned out to be the optimal number. It was the same for 2D Convolutional layers and Max Pooling layers.

4. RESULTS

Using the configuration quoted above, we managed to get a f1-score of 0.91 for Green detection and of 0.73 for Orange detection. As far as Green labels are concerned, we reached a precision of 0.96 and a recall of 0.87. When it comes to Orange labels, we have a precision of 0.64 due to the unbalanced batch test and a recall of 0.86. Those results are pretty encouraging, but we realized during our work that an even larger dataset would lead to noticeable improvements. In order to improve those results, one could also try other CNN configurations. It would also be great to have more Orange training data, in order to avoid Green over-training.

Figure 4.1 – Confusion Matrix, without normalization



METHOD 3

Support Vector Machine

Abstract: This paper analyses the possible use of support vector machines as an effective supervised machine learning tool for both fault detection and fault diagnostics. The industrial installation studied is a wind farm, within which two wind turbines were followed using the data produced by sensors placed on their respective generators. This study placed a particular emphasis upon feature extraction and the minimal amount of features required to have a stable result.

Keywords: SVM, Supervised Learning, Wind Turbine, Feature Extraction

1. INTRODUCTION

The support vector machine algorithms are based upon robust, demonstrated mathematical theory and are an example of a supervised machine learning algorithm [7]. SVMs can use virtually any type of feature, ranging from pure statistical features such as the mean or standard deviation of our signal to what we called "samples" (for explanation look to data section) making it a very versatile method and computationally efficient with regards to the number of features used (less so with the size of our training dataset as we will later explore). SVMs can also be used to reduce the number of features in a process of feature selection [14], making it perhaps more intelligible for an expert wishing to explain why certain features hold a predominant role. It is, however, important to note that the selected features are not necessarily easily understandable by an expert, they just provide a good discriminatory basis for our algorithms.

2. MATHEMATICAL CONCEPTS

2.1 Framing the problem

SVM mathematical theory is an interesting combination of multiple different domains, combining dual representations, feature spaces, learning theory and optimisation theory. This means that this paper has no pretension so as to present a rigorous view of SVMs but just to present a basic view of the notions needed to explain our results.

Let us consider our dataset S = $\{(x_i , y_i), i \in [1,n]\}$ of labelled examples, where the x_i are the feature vectors, each having p components, and the y_i are the true labels taken by an expert. If we consider a binary classification problem (fault detection, whether the turbine is healthy or not), we have $y_i \in \{-1, 1\}$, else we have $y_i \in \{2$ (green), 3(yellow), 4 (orange)$\}$ or any other set of classes if we wish to have even more precise fault diagnostics. We need to find a function that, given a new feature vector, outputs the correct classification of said vector.

2.2 Binary classification with an existing linear separation

Starting with the binary classification case, we wish to find a linear function that properly separates our data. This means that for any new feature vector with p coordinates (features) we will be able to discriminate upon which side of the boundary it lies and to classify it. This translates to:

$$f: \Re^p \rightarrow \Re, \forall x_i, y_i = 1, f(x) \geq 0 \; and \; \forall x_i, y_i = -1, f(x) \leq 0 \quad (1)$$

For any new vector x, the positivity of $f(x)$will give us the corresponding binary classification. The fact that $f$is a linear function gives us :

$$\exists (w, b) \in (\Re^p)^2, \forall x \in \Re^p, f(x) = < w, x > + b \quad (2)$$

Thus, training the machine aims to determine a couple (w,b) which separates efficiently the points from the training dataset and is robust enough not to fail if we test it on a new point. Mathematically, this means we have to maximise the distance between the hyperplane $< w, x > + b$=0 and the nearest

point from each class, which are called support vectors. This distance is called the margin and is the same for both classes. It is important to notice that this margin only depends on these few support vectors, and doesn't depend on extreme points of the data. Under the hypothesis:

$$\forall\, i \in \{1,..,n\},\ \text{sign}(f(x_i))y_i \geq 0 \qquad (3)$$

calculations we will not develop here show this margin is inversely proportional to $\|w\|$. Thus, our goal is to minimize $\|w\|$ respecting the hypothesis above. (this is an optimization problem, and the Karush-Kuhn-Tucker (KKT) [15] theorem states it is equivalent to minimizing the Lagrangian of a function which depends on the parameters of the problem).

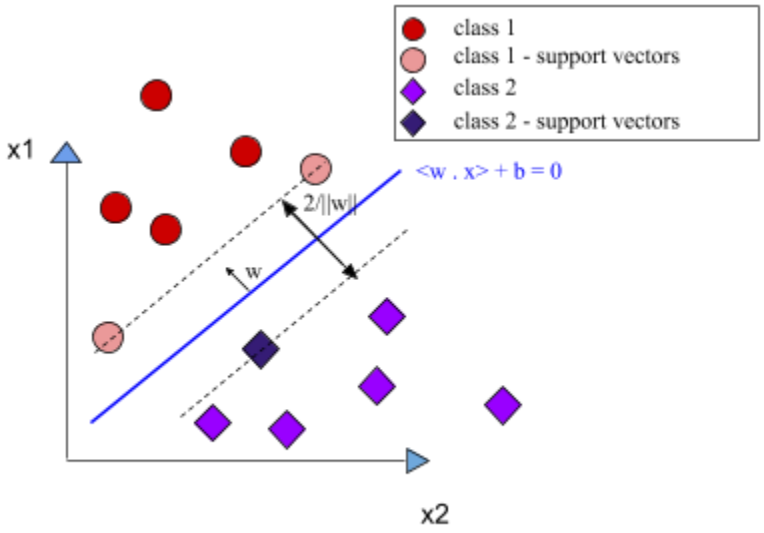


Figure 1 - Binary classification with linear separation

2.3 Binary classification with mapping into a new feature space

If such a linear separation does not exist, we can use a mapping function to plunge our feature vectors into a new space, the feature space, in which we might find a linear separation. This feature space can be of the same, lower or greater number of dimensions than our original space.

Finding the right feature space is not always an easy task and it can be very computationally expensive to explicitly map said feature space. Therefore, it is possible to use a kernel trick to avoid explicitly mapping the feature space and having to perform expensive operations. We introduce a kernel function K, which is as follows:

$$K: (x1, x2) \rightarrow < \phi(x1), \phi(x2) > \qquad (4)$$

Mercer's theorem provides sufficient conditions for a function K to be described this way: it states that it is enough that K be continuous, symmetrical and positive semidefinite[7].
To avoid having to explicitly calculate the dot product in a space of infinite dimensions (our feature map can indeed have an infinite number of dimensions), the kernel trick rests on having another formulation of this dot product based on our initial feature vectors. Take for example, an rbf (radial basis function) kernel:

$$K_{rbf} : (x1, x2) \rightarrow e^{-\gamma|x1-x2|^2}, \gamma > 0 \qquad (5)$$

This function avoids having to produce the actual feature vectors $\phi(x1), \phi(x2)$that actually have an infinite amount of components in this case.

3. IMPLEMENTATION OF SVMs

3.1 Python modules used and our approach

SVMs being a popular method regarding data classification, scikit-klearn (sklearn) has a module specifically dedicated to SVMs with different possibilities of implementation [9]. The most straightforward method is sklearn.svm.svc (svc for support vector classification). The parameter we needed to modify was mostly the kernel, however other parameters such as the regularization parameter and whether or not to enable probability estimates are interesting possibilities. This callable function does not scale well with the number of feature vectors we are dealing with (more than polynomial complexity) thus another method we could have explored is sklearn.svm.LinearSvc which scales better but with which we cannot choose the kernel function as it is automatically taken to be linear.

3.2 Results on different datasets

Different tests on varied datasets have shown that using balanced datasets (with as many pieces of data labelled 1 as labelled -1) to train the algorithm leads to better results. Indeed, if one class is overrepresented, the machine tends to detect this class very well but is more likely to be inaccurate in diagnosing the other class. Furthermore, big training datasets are not relevant, they tend to lead to overfitting : the model corresponding to the training dataset is so accurate that the machine can be wrong if a point distant from the training ones is inputted. The results are as follows:

Figure 3.2.1 - Table of results

Class\Indicator	Precision	Recall	f1-score
Orange	0.55	0.71	0.62
Green	0.87	0.76	0.81

3.3 Limitations of SVMs

The above table highlights some limitations of SVMs.
Even though the mathematical approach of the Support Vectors Machine method demonstrates the existence of a function which perfectly distinguishes both classes, it is hard to obtain perfect results in reality. First, computational time is more than a polynomial function of the size of the input datasets, making the training long when the dataset exceeds 10 000 vectors (which remains rather small in comparison with eight years of data from the windfarm). As the machine needs a sufficient amount of data to be correctly trained, one must try and optimize the relation size of the dataset/accuracy of the results. Moreover, the algorithm can be sensitive to inaccurate labelisations. As mentioned above, confusion matrices highlight difficulties in distinguishing classes 2 and 3 in this case. It is therefore necessary to pre-label as accurately as possible the data the company will train the machine with.

3.4 Next step: using several SVMs

In order to improve the precision of the algorithm, we have imagined an algorithm training simultaneously several SVMs on different random parts of the training dataset (but with intersections). This addresses the problem of training time for a training dataset nearing 50 000 feature vectors, we can train several SVMs on different, smaller parts of the dataset, reducing the training time significantly. Then, these SVMs would take a “vote” on the new data input, and the results would be the class with the higher score. Different ways of taking the vote are possible, ranging from a simple vote, in which each SVM has the same weight, to a vote weighting the importance of each SVM in

the vote depending on the distance of the input vector to the ones each SVM has been trained with (see AAKR for more details on this weighting method). This leads to slightly better results.

Here, we chose to give each SVM the same weight, which yielded to the following table of results:

Figure 3.4.1 Table of results using 20 SVMs, each trained on 1/10th of the training set

Class\Indicator	Precision	Recall	f1-score
Orange	0.55	0.75	0.63
Green	0.88	0.75	0.81

METHOD 4

Auto Associative Kernel Regression and Sequential probability ratio test

Abstract: In order to determine if a Wind Turbine Generator is working correctly or not, we used the combination between Auto Associative Kernel Regression (AAKR) and Sequential Probability Ratio Test (SPRT) methods. AAKR finds residuals that SPRT uses to say if a signal is abnormal or normal.

Keywords: Auto Associative Kernel Regression, Signal Reconstruction, Sequential Probability Ratio Test, Residual Analysis

1. INTRODUCTION

Condition monitoring aims at identifying problems on industrial components when experts are not able to spot them properly. Here, AAKR and SPRT, algorithms of fault detection are applied to signals coming from accelerometers that are installed on a Wind Turbine Generator.

AAKR tries to reconstruct the features of the signal that we want to label. We compare the reconstruction to the original signal and judge if it corresponds to normal or abnormal condition with SPRT. Those methods are semi-supervised ones because here we do not give abnormal condition data as training data but normal condition data instead. Besides, the algorithm is not really training itself with data, we start to calculate from scratch for each signal.

2. DESCRIPTION OF THE METHODS

2.1 AAKR

2.1.1 Database and signal reconstruction

AAKR method is based on a set of normal data, which is represented by a matrix. Each line represents the *n* features of the signal on a time *t* whereas each column represents, for just one feature, the *N* values corresponding to the *N* times. The robustness of the algorithm, ie his ability to reconstruct a normal condition vector when the given vector is an abnormal one, is proportional to the length of the matrix, the homogeneity of the signals and the number of relevant features.

We use the same model for the vector that we want to test:

$$\vec{x}^{obs} = (x_1^{obs},\ldots,x_n^{obs})$$

Then we calculate the reconstruction of the vector expected in normal condition as a weighted sum of the training patterns:

$$\vec{\hat{x}}^{nc} = (\hat{x}_1^{nc},\ldots,\hat{x}_n^{nc})$$

$$\hat{x}_j^{nc} = \frac{\sum_{k=1}^{N} w(k)\cdot x_{kj}^{obs-nc}}{\sum_{k=1}^{N} w(k)}$$

The different weights are based on the Euclidean distance *d(k)* between the input data and the training data:

$$w(k) = \frac{1}{\sqrt{2\pi}h} e^{-\frac{d^2(k)}{2h^2}}$$

h is a parameter to find by minimizing a function.

2.1.2 Calculation of the residuals

At this stage, the residuals may be calculated for each feature at each time but to simplify the treatments in SPRT, it was decided to measure the Euclidian distance between the two vectors: input one and output one.
The reconstruction (red star on Figure 6) is supposed to be in normal condition so if both vectors are close, we can conclude that the input vector (red point on Figure 6) is in normal condition as well:

Figure 2.1.2.1 – Normal Signal Representation for X_1 and X_2 as features [1]

However, if the input vector is far from the output one, the residuals will be abnormally high:

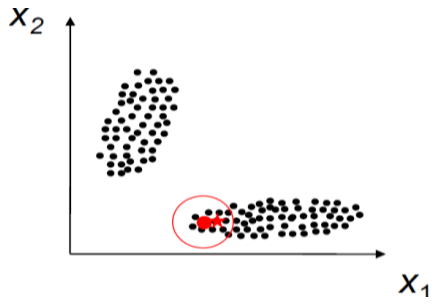
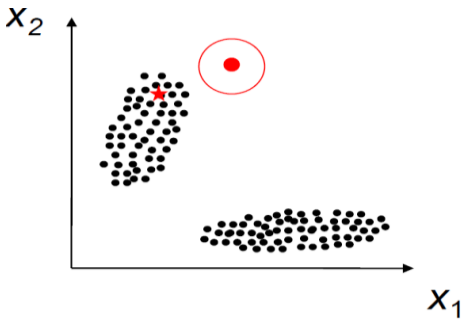


Figure 2.1.2.2 – Abnormal Signal Representation for X_1 and X_2 as features [1]



2.2 SPRT

Once the residuals given by the AAKR, a decision must be taken to say when residuals are too high or when they are normal. α and β are respectively the maximum of false alarm and of missing alarm that we want. They are helpful to calculate two thresholds. The residuals are not compared to them directly. It is more efficient to use a sequence of values calculated from the thresholds:

Figure 2.2.1.1 – Calculation of the sequence [1]

$$L_0 = 1 \rightarrow \ln(L_0) = 0$$

$$\ln(L_1) = \ln(L_0) + \frac{\mu_1}{\sigma^2}\left(r_1 - \frac{\mu_1}{2}\right)$$

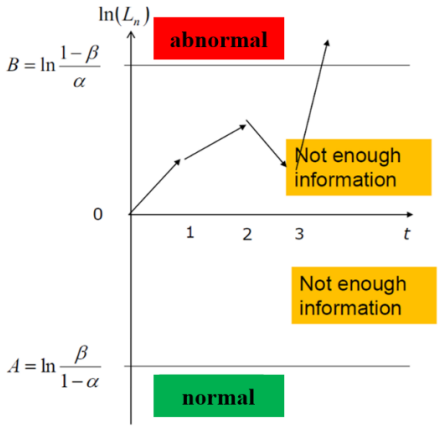
$$\ln(L_2) = \ln(L_1) + \frac{\mu_1}{\sigma^2}\left(r_2 - \frac{\mu_1}{2}\right)$$

...

- μ_1 is the mean of residuals in a dataset known as abnormal.
- σ is the standard deviation of residuals in a dataset known as normal.
- r_1 ... are the different residuals (Euclidian distance).

When a value goes over or under the thresholds, an abnormal or normal condition is detected. Then, the algorithm restarts from the beginning with the rest of the residuals:

Figure 2.2.1.2 – Operation of the SPRT [1]



At the end, a choice is made to know if the whole signal (several seconds) is normal or not by comparing the proportion of normal and abnormal alarms raised by SPRT to a limit named limit_yo in the results.

3. RESULTS

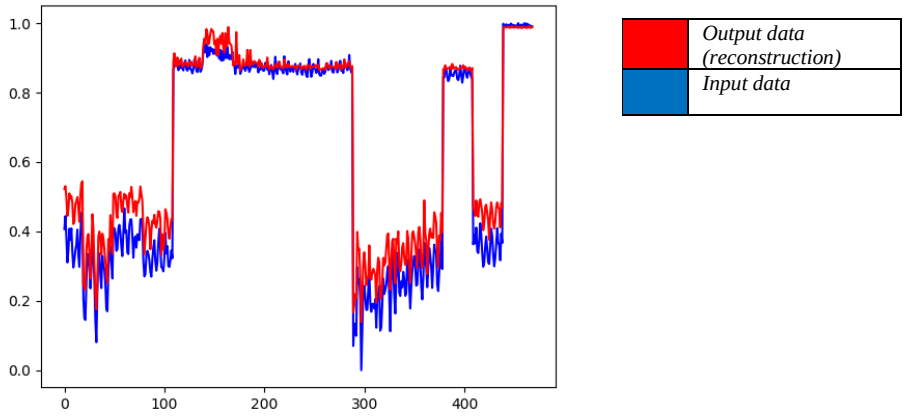
3.1. Implementation & Plots of the tests to find the good parameters

The implementation of the algorithms was almost made from scratch. Regarding the modules, Scikit-Learn was used to scale all the data not to have bigger impacts of some features on the others. We also used matplotlib, pandas, numpy, math, random and time.

To plot signals of 5 seconds, we split each of them in 40 *extracts* (0.125 sec) and then, we calculate the features on each of those signals. At the end, we execute the algorithms on 5 seconds so that we get 40 residuals, which is a minimum to obtain benefits from SPRT.

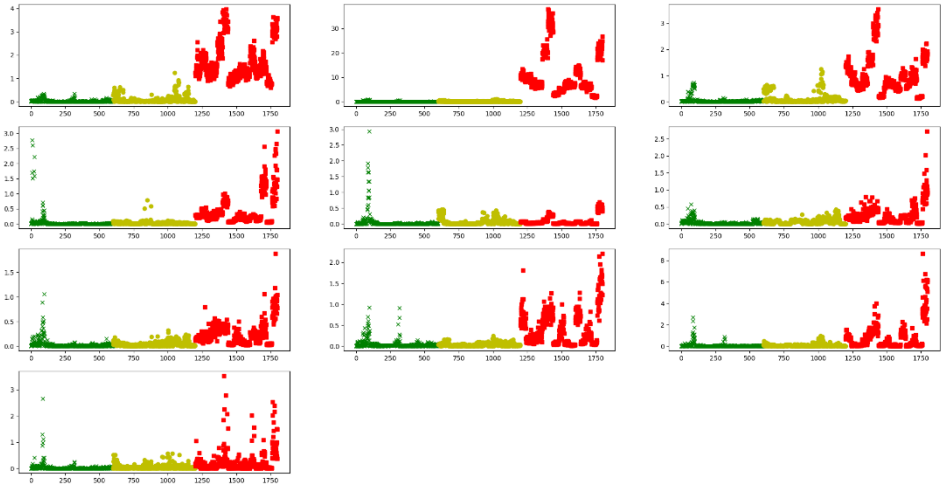
Here is the reconstruction of the feature 1q (first quartile) on 500 *extracts* (0.125 sec)*:*

Figure 3.1.1 – Signal reconstruction



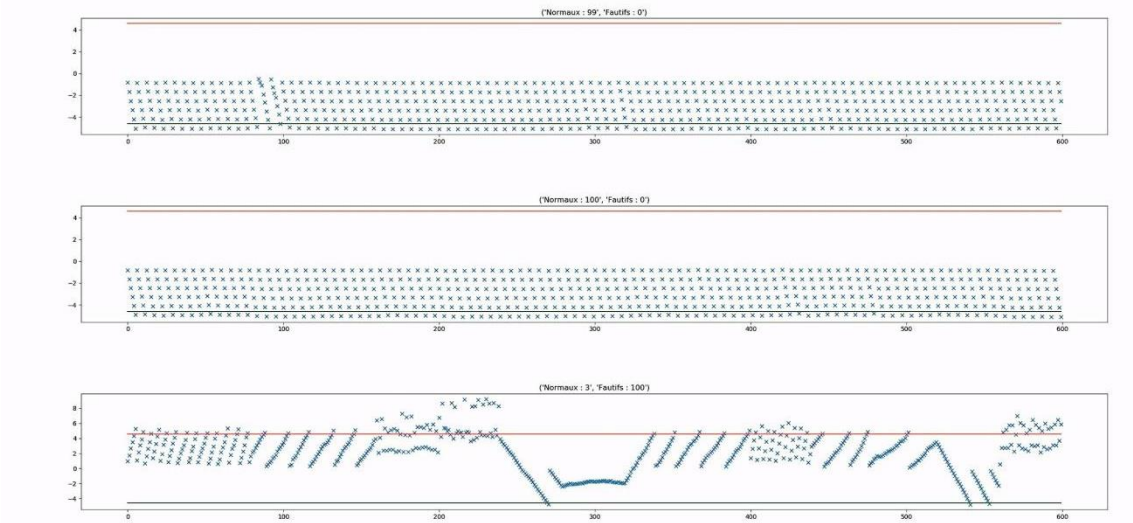
Then, we use data that were labelled in 3 categories: green, yellow and orange, and we plot residuals for each feature to select the good ones. Each point represents a residual for an *extract* (0.125 sec):

Figure 3.1.2 – Features coming from FFT



To finish, the plots of SPRT method show that SPRT does not make any difference between yellow (second plot) and green (first plot) signals like the PCA method but find orange signals (third and last plot). Each point represents one *extract* (0.125 sec) and here there are 600 extracts:

Figure 3.1.3 – SPRT Method for Green, Yellow and Orange data



3.2 Results on the "Milano" test dataset

When we first tried the algorithm with the "Milano" dataset, we reached very good results. We managed to retrieve all the predictions of the expert in a two-class model. Unfortunately, we did not succeed in splitting data in a three-class model.

Figure 3.2.1 – Results

Parameters	f1-Score GREEN	f1-Score Orange
limit_yo = 0.2	0.97	0.94
limit_yo = 0.5	0,93	0.97

3.3 Results on the final Dataset

We extracted the features from "main" dataset. And we trained the AAKR on "Milano" training dataset because we needed homogeneous data. We tested the model on 25% of the "main" dataset, which were used by other methods to test. We first optimized the set features to find the one that gave discrimination on the residuals.

Figure 3.3.1 – Results on the final Dataset

Parameters	f1-Score Green	f1-Score Orange	Used Features (after selection)
limit_yo = 0.5	0.44	0.84	['compteur', 'kurtosis', 'skew','mean','1q','Medi','max', 'sample0', 'sample1','sample2','sample7','sample8']

The results are not as good as before because here the data were labelled on a month and the AAKR was used on a lower scale. Maybe, by comparing these results to PCA's ones, we may find a new way to label the data day-by-day or week-by-week.

To get results, we used a big training dataset, which imposed a long time for the program to run. On "Milano" dataset, we needed about 40 seconds whereas Random Forest needed just one second.

METHOD 5

PCA & Q-Statistics

Abstract: In this paper, the PCA method combined with the Q-statistics are described and used to detect the faults in the studied dataset. The particularities and benefits of these methods are discussed, and the key parameter, the threshold of the percentage of information kept, is optimized in practice. Results, in particular the quite low f1-score for abnormal measures, are then debated.

Keywords: PCA, Q-statistics, space transformation, healthy training set

1. INTRODUCTION

PCA is a semi-supervised method, which can be combined with the Q-statistics in order to make a fault detection algorithm. The method is based on space reduction and principal component detection. It only requires data in normal conditions to be trained with, and it appears to be a very precise fault detection algorithm, making it a valuable method for our study. [16]

2. PCA & Q-STATS

2.1 Principal Component Analysis (PCA)

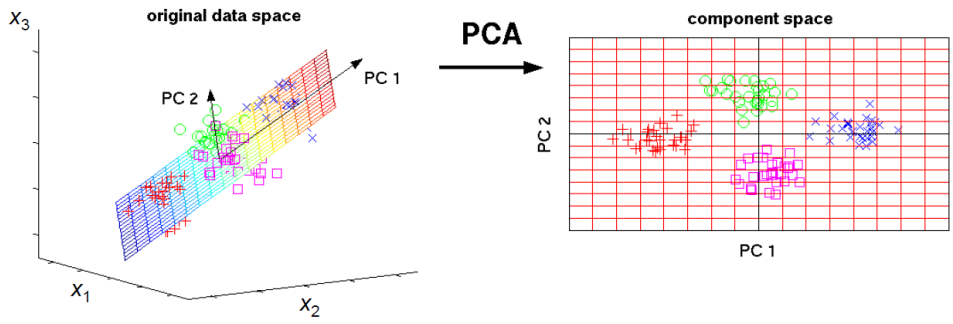
For this method, one measure is only represented by its statistical features in the data table (from kurtosis to max, without the wind speed), being thus a point in a 10-dimensional space.

2.1.1 Idea of the method

The PCA is trained with a set of normal condition measures, points in a n-dimensional space (n = 10 for our study). A threshold representing the minimum percentage of information kept β is defined. Using the covariance matrix of the training set, the algorithm finds l < n new principal components, that are the components of maximal variance of the set whose total information kept is greater than the threshold. The training phase thus provides the matrix to project from the n-dim space to the l-dim one.

A new given measure is transformed by being projected in the l-dimensional space of principal components, then projected back in the n-dimensional space. Some information is lost, but if the initial measure is a normal one, the training process guarantees that the distance between the two points will remain small. [17]

Figure 2.1.1.1 – Explanation of the PCA method,
in a 3-dimensional space with two principal components kept [2]



2.1.2 Implementation

We used PCA module from sklearn in Python. [9]

2.2 Q-statistics

The Q-statistics, also known as squared prediction error, is a statistical test which quantifies the residual subspace calculated by the PCA method and compares it with a threshold that depends on components not kept by the PCA model.

Thus, on the one hand, thanks to $\hat{x}^{(j)} = (\widehat{x_k}^{(j)})\ with\ k\ \in \{1; n\},\ \hat{x}^{(j)}$ being the residuals given by the PCA method at time j, the Q-stat is calculated at time j by the following formula:

$$Q^{(j)} = \sum_{i=1}^{n} \widehat{x_i}^{(j)^2}$$

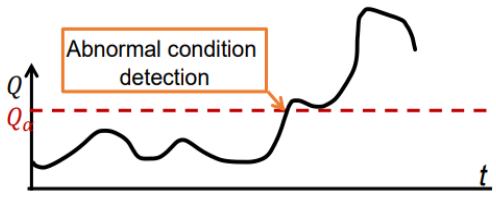
We obtain eventually a matrix $Q = [Q^{(1)}, \ldots, Q^{(m)}]$, m being the last time period.

On the other hand, a threshold Q_α is used to determine whether the wind turbine reacts normally or abnormally. Abnormal conditions can be detected when $Q^{(j)} > Q_\alpha$.
Q_α is calculated thanks to the eigenvalues of the components that have not being captured by the PCA method since they represent the normal behaviour of the wind turbine and thanks to a certain level of confidence αchosen. The αchosen (generally 0.05 or 0.01) is influencing the calculating with z_α which is the standard normal deviate corresponding to the upper 1-α percentile. [18]

$$Q_\alpha = \vartheta_1 \left(\frac{h_0 z_\alpha \sqrt{2\theta_2}}{\theta_1} + 1 + \frac{h_0 \theta_2 (h_0 - 1)}{\theta_1^2} \right)^{1/h_0}$$

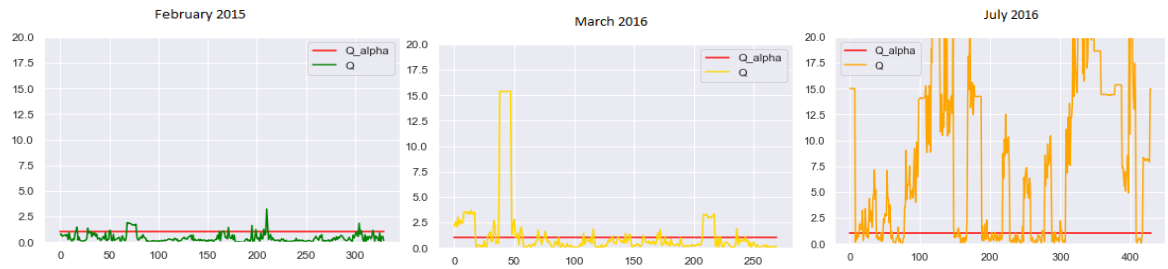
$$\vartheta_i = \sum_{k=l+1}^{n} \lambda_k^i, i = 1,2,3;\ h_0 = 1 - \frac{2\theta_1 \theta_3}{3\theta_2^2}$$

Figure 2.2.1 – Explanation of the Q-Statistics [4]



For instance, that is the graphs we obtained with for a month labelled 0 (normal conditions, in *green*), 1 (to monitor, in *yellow*) and 2 (abnormal conditions, in *orange*) with $\alpha = 0.05, \beta = 0.95$.

Figure 2.2.2 – Q-Statistics on three months



2.3 Legitimacy of the method

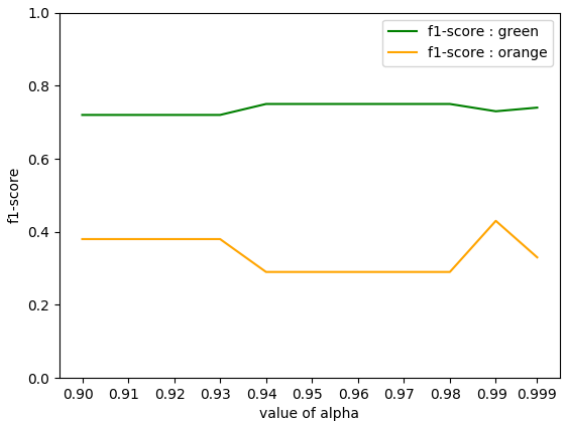
The PCA + Q-stat method requires only healthy measures to be trained with, thus it can be used on a wind turbine quite quickly after the start of the machine, even before the first failure.

However, the PCA method is based on finding almost-linear correlations between measures, and reducing the number of components thanks to these correlations. Thus, the method is ineffective for dataset characterized by highly non-linear relationship.
In our study, measures are represented by their statistical features, and we may assume that those features are not related by highly non-linear relationship, making the method legitimate to use.

2.4 Threshold sensitivity

The one parameter that can be changed to optimize our algorithm is the PCA threshold β. Keeping too little information (small β) increases the distance between the initial point and the transformed one, what could result in green measures detected as orange ones. However, keeping too much information ($\beta \approx 1$) will make it too hard for the algorithm to discriminate abnormal measures, resulting in orange measures detected as green ones.

Figure 2.4.1 – Evolution of the f1-score based on the value of α, for the last dataset



The constant f1-score on a period corresponds to a fixed number of principal components kept. Based on what is the most valuable between the green f1-score and the orange one, the practical analysis permits us to respectively consider β= 0.95 (l = 5 principal components) or β = 0.99 (l = 6).
As our study is about risk preventing, false positive is less bothering than missed errors: that is why we will favour the orange f1-score over the green one.

3. RESULTS

Figure 3.1 – Global results for the two datasets, with β = 0,99

Test type	Number of categories	Support	Precision	Recall	F1-score	Computational time
Milano dataset	2 y ➞ g	G: 1200	0.86	0.68	0.76	0.2 s
		O: 600	0.55	0.78	0.64	
Main dataset,	2	G: 5527	0.66	0.82	0.73	1.3 s

	No yellow	O: 3673	0.57	0.35	0.43	

Figure 3.2 – Global results for the two datasets, with β= 0,95

Test type	Number of categories	Support	Precision	Recall	F1-score	Computational time
Milano dataset	2 y ➞ g	G: 1200	0.90	0.84	0.87	0.2 s
		O: 600	0.72	0.81	0.76	
Main dataset,	2 No yellow	G: 5527	0.63	0.92	0.75	1 s
		O: 3673	0.61	0.19	0.29	

Analysis: The f1-score for the data labelled orange in the main dataset seems a bit disappointing, but it is actually all the contrary. What we should take into account is the fact that the pundit labelled an entire month orange, but in reality there could be only a few days or set of data of 5 seconds where the faults were detected and the rest of the data in a month had not captured any abnormal conditions, because they were not any! Therefore, the PCA combined with Q-statistics gives us more information about the data than the pundit does. It could be interesting to re-label the set of data with the ones given by this method, since its legitimacy can be proven by the more-than-honest results on the Milano dataset and the explanation above.

METHOD 6

RANDOM FOREST

Abstract: This paper develops the use of a Random Forest classifier to determine the current condition of a Wind Turbine Generator. Based on the notion of bagging, this supervised algorithm uses statistical measures in order to create decision trees that will help determine the status of the Wind Turbine. In this Study we have created an implementation of this Algorithm that has very good results in labelling the condition of the Wind Turbine. This Study placed emphasis on feature selection, and parameters optimization.

Keywords: Random Forest, Decision Trees, Condition Monitoring

1. INTRODUCTION

This supervised method of prediction relies on the use of decision trees built on training dataset. These trees are made to represent logical conditions that can lead a request to the corresponding label. They can be built from existing data using statistics methods. The random forest use a high number of trees made of random sample of Features. When a test request is made, all the trees vote following their constitution and the algorithm answers the label that has received the most votes. This type of algorithm seems very relevant in our case because the prediction made by the algorithm looks like the one made by an expert, who has many criteria and who give them precise importance in order to make his decision.

2. DESCRIPTION OF THE METHOD

2.1 The Method

The underlying method of random forest is the use of Decision Trees. The tree is a logical element that is used to go from observations about a condition to conclusions that are represented in the leaves, using logical test at each node to decide which branch to follow. However, trees can be very non-robust, they‘re easily subject to over-fitting as well that they are extremely depending on dataset constructions and shapes.

To get rid of this uncertainty, Random forest has been built to use the “wisdom of the crowd”[2] to get the correct answer. The algorithm grows *B* decision trees to become much more precise. The variance is lower than if one tree was used. The features need to be uncorrelated, although it will flaw the algorithm or give too much credit to a condition. To avoid such unwanted behaviours bootstrapping is used in the training phase.

2.2 Training Phase

As explained before the algorithm of a Random forest is a combination of decision tree learning algorithms and random process in order to reduce the weight of noise. The Bagging randomly selects samples in the dataset to grow decision trees from them. It then fits trees to these samples (noted x'). The prediction function $\hat{f}$ is given by [3]:

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} \quad f_b(x')$$

This bootstrapping procedure leads to better model performance because it decreases the variance of the model. While the predictions of a single tree are highly sensitive to the training set, bootstrap

sampling is a way of de-correlating the features. The number of trees B is a completely free variable that must be fixed in order to maximize the efficiency.

Moreover, in the construction of a random forest, the algorithm select random features at each split that avoid the trees to become too much correlated. This corresponds to the 'node size' parameter, in our case the inventors recommend a size of $K = \sqrt{p}$ where p is the initial number of features. A split creates two nodes (one left and one right) is based on the Gini impurity index Q (usually used in classification algorithms) on the K selected features in a node [19]:

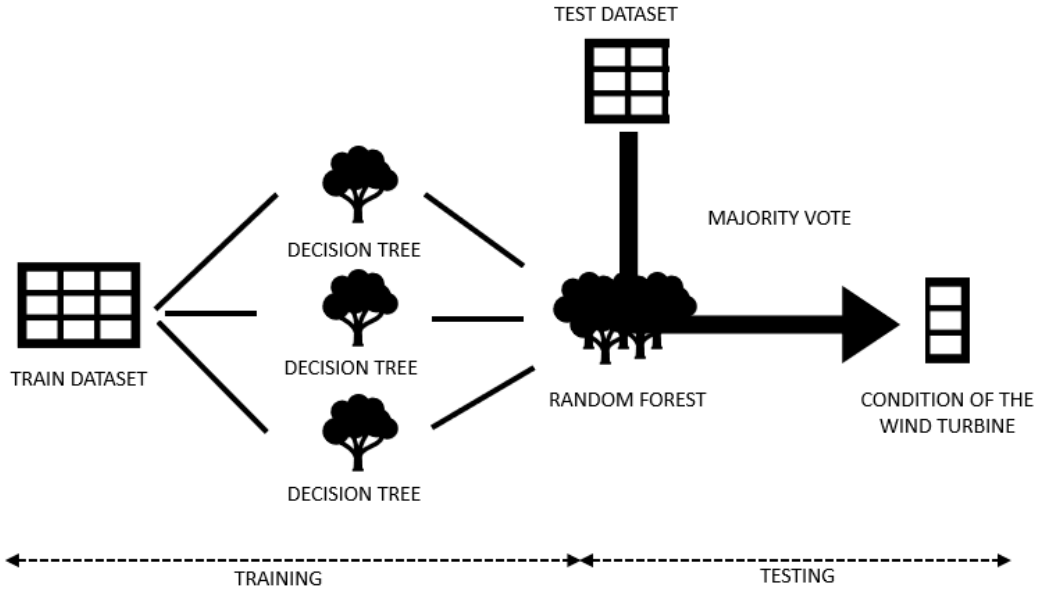
$$Q = \sum_{k \neq k'} \hat{p}_k \hat{p}_{k'}$$

Where $\hat{p}_k$ is the proportion of class k observations in the node. The split that is chosen is the one that minimizes $Q_{split} = n_L Q_L + n_R Q_R$, where n_R, n_L are the number of samples in each nodes.

2.3 The Vote

When a request to the random forest is made, each tree votes according to the real features, the answers of the algorithm is the most selected label.[20]

Figure 2.3.1 – Diagram of a random forest algorithm



3. IMPLEMENTATION

We have used the function RandomForestClassifier from the python package for data science SKlearn. [9]

3.1 Results on the "Milano" test dataset

When we first tried the algorithm with the Milano Dataset, we have reached very good results. We managed to retrieve all the predictions of the expert in a two classes model, and we have detected all the orange conditions in a 3 class model. And we even only considered the feature Sample1 and only one tree and we obtained the same results. The dataset was too idealistic because only the study of the Sample1 Feature was needed. We then decided to use a more general dataset.

Figure 3.1.1 – Results of a RF on the Milano Dataset

Parameters	F1-Score Green	F1-Score Orange	F1-Score Yellow
1 Tree / Sample1 / 2 class Model	1	1	
1 Tree / Sample1 / 3 class Model	0.85	1	0.86

3.2 Results on the final Dataset

We tested the algorithm on the Main dataset. To compare the efficiency we created a 2 class model and a 3 class model. We first optimized the set features to find the one that gave good results with a high number of trees, we then decreased the number of trees until the score dropped.

Figure 3.2.1 - Number of Trees optimization

Number of Trees	F1-Score Green	F1-Score Orange	Used Features
200	0.98	0.95	['Wind_speed', 'Ref_speed', 'kurtosis', 'std', 'sample0', 'sample1', 'sample4', 'sample7']
50	0.98	0.95	['Wind_speed', 'Ref_speed', 'kurtosis', 'std', 'sample0', 'sample1', 'sample4', 'sample7']
25	0.98	0.94	['Wind_speed', 'Ref_speed', 'kurtosis', 'std', 'sample0', 'sample1', 'sample4', 'sample7']

We reached very good results using the RF algorithm. We can observe that the RF uses expressive features to make its choice that is not usual in such implementation. But it can reveal the implicit data that the expert uses to make his judgment.

Using the same Algorithm and samples, we also found very good results for a three class model. However they are slightly worse than the previous one, but it really depends what kind of precision *Prüftechnik* is interested in.

METHOD 7

k-NN: k-Nearest Neighbours

Abstract: In this paper, the k-NN method is described and applied to classify the studied dataset. Thanks to a practical analysis, the key parameters are optimized in order to obtain the highest f1-score for the set. Some ideas for further optimization are given, and the results are briefly discussed.

Keywords: KNN, classifier, metric, lazy-learning

1. INTRODUCTION

The k-NN method is a lazy-learning, supervised classifier. A given object is classified by finding the most common class among its k-nearest neighbours, and assigning it to that class.
The algorithm is one of the simplest classifier, as it requires nothing but distance comparisons between points, but can be a quite effective first approach to classify our data.

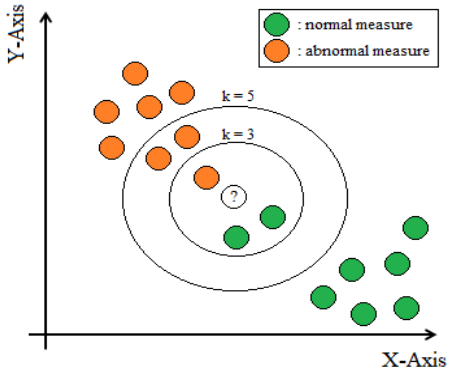
2. METHOD AND PARAMETERS

2.1 The k-NN method

For this method, one measure is represented in a 20-dimensional space by its different features in the data table (from kurtosis to sample9, without the wind speed). The training sample provides labelled points in this space that are used by the algorithm to classify the new data: a new measure is classified as the most common class among its k-nearest neighbours.

The classification of a given measure depends on both the number of neighbours considered and on the metric used, as the notion of *nearest neighbours* relies on it. Thus, those two parameters have to be optimised to reach the best possible results. [21]

Figure 2.1.1 – Explanation of the k-NN method, with 2-dimensional measures



2.2 Legitimacy of the method

k-NN relies on seeking neighbours of a point in a 20-dimensional vector space. It is expected that the statistical features used by the pundits reflect well enough the current state of the wind turbine, making it reasonable to assume that normal-condition measures are relatively clustered in that space. In addition, it requires no particular type of data, and can thus be used in our study.

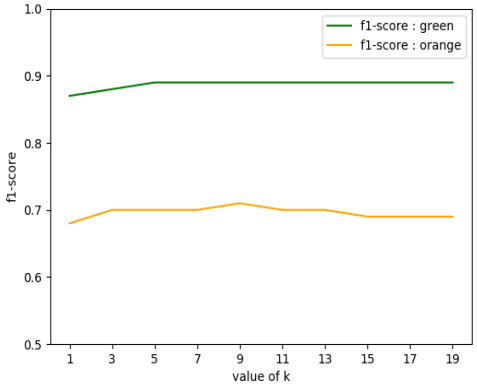
However, a possible drawback of the k-NN algorithm is that it doesn't work well with a biased labelled training set: if a class is highly represented compared to another one and k is large enough, a given measure to classify will almost always be classified as this one class, even if the measure is way nearer of a point from the other class. Nevertheless, it is not an issue for our data, as the labelled set used is composed of both normal and abnormal measures in significant amounts.

2.3 Parameters

Two parameters to fix have been identified: *k*, number of neighbours considered, and *d*, metric for comparison.

First of all, with two classes, k must be odd to avoid any equality in the class distribution of the neighbours. A small k (= 1, 3...) is not effective, as the goal is to assign the measure to a group, based on a global structure: it is too sensitive to the noise in the training set. On the other side, a large k (> 20 ...) is less bothering but it tends to merge the groups rather that discriminating against them.

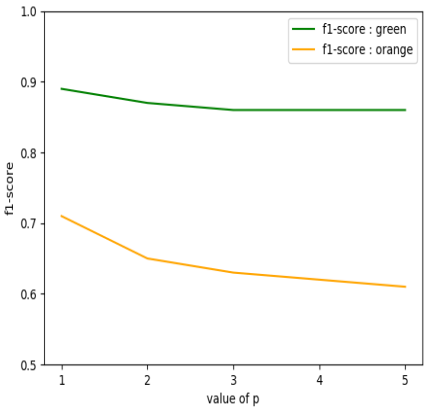
Figure 2.3.1 – Evolution of the f1-score based on the value of k, for the last dataset with 2 categories and the Manhattan metric.



With this practical test, k = 9 appears to be an optimal value. For the metric, p-norms are good candidates to try with:

$$For\ x \in R^n, ||x||_p = (\sum_{k=1}^{n} x_i{}^p)^{\frac{1}{p}}$$

Figure 2.3.2 – Evolution of the f1-score based on the value of p, for the last data set with 2 categories and k = 9



Practical tests give a more precise algorithm for p = 1, the *Manhattan metric*, and an accuracy decreasing with p increasing. For further optimization, metric learning can be used to train a new metric adapted to the training dataset. It hasn't been done in this work for lack of time.

Implementation: neighbors.KNeighborsClassifier from the sklearn module [9]

3. RESULTS

Figure 3.1 – Global results for the two datasets, with k = 9 and p = 1 (Manhattan metric)

Test type	Number of categories	Support	Precision	Recall	f1-score	Computational time
Milano dataset	3	G: 560	0,75	0,74	0,75	0,2 s
		Y: 560	0,75	0,76	0,75	
		O: 600	1	1	1	
Milano dataset	2 y → g	G: 1200	1	1	1	0,2 s
		O: 600	1	1	1	
main dataset	3	G: 5527	0,74	0,86	0,80	3,3 s
		Y: 3550	0,73	0,59	0,65	
		O: 3673	0,73	0,70	0,71	
main dataset	2 y → g	G: 9077	0,87	0,91	0,89	3,5 s
		O: 3673	0,74	0,67	0,71	

Analysis: the method seems quite effective, with a relatively high f1-score for the main dataset. One advantage is that this method works with more than 2 categories, while some other methods (PCA, AAKR....) don’t.

ACKNOLEDGEMENT

The whole team would like to extend its most sincere and utmost gratitude to Sébastien TRAVADEL and Frank GUARNIERI who accompanied us during our numerous visits and followed the entire project from start to finish. Special thanks are going to Frank FUGON and Prüftechnik for providing us the case study and all their advices on how to develop useful methods for the company. We would also like to thank Enrico ZIO and Philippe BLANC for responding to our numerous questions and giving us the mathematical basis needed to responds to this case study. We would also like to thank Xuefei, Eva, Ahmed, Georges-André and Samuel for helping us to face the sometimes complicated aspect of Machine Learning. We finally want to thank General Electric, The IRCGN, Remy JUTY for the BEA, the CEA, *the Conservatoire des arts et metiers* and all the people and companies that give us a first insight in Forensic Engineering. You all contributed to make this experience unforgettable.