MASTER'S DEGREE THESIS

MSc INTERDISCIPLINARY AND INNOVATIVE ENGINEERING

WIND TURBINE GEARBOX FAULT PROGNOSIS BASED ONLY ON SCADA DATA

Report and annexes

Author:	MARC JOSEP BENNASAR ROMERO
Director:	YOLANDA VIDAL
Co-Director:	FRANCESC POZO
Call:	OCTOBER 2021

Abstract

Wind turbines often operate in environments with hostile and harmful agents. That is why such wind turbines require extensive and constant maintenance operations. To avoid cost overruns, good planning and organization of maintenance tasks is vital. The decisions to be made are normally based on the data that the Supervisory Control and Data Acquisition (SCADA) is capable of communicating, which is a system that monitors the status of the turbine and its components as long as they have sensors.

Turbine status monitoring detects failures when they occur and allows the maintenance team to know where they should act. In any case, when they arrive at the turbine, it is already damaged or requires a temporary suspension of its activities. One of the mechanisms that most often breaks down is the gearbox and its replacement is costly since it not only requires turbine shutdown time but also the disassembly and assembly of a large part of the wind power installation as well as the gearbox and that is not a cheap item.

The following project presents a method that aims to detect possible gearbox failures before they occur. This method is based on machine learning and on the data collected by the SCADA of the wind turbine. The early detection algorithm is based on the Relevance Vector Machine (RVM).

1.Preface

1.1 Motivation

Turbines are machines that need continuous maintenance to ensure their good operation. Working in harsh environments with many agents that damage them makes the cost of maintenance and operation of wind turbines high. In recent years there has been a large increase in wind turbines in operation, so there are many turbines with an advanced operating age in the world.

As one can imagine, as a turbine gets older, it becomes more likely that there is a failure in one of its components. One of the components that can fail is the gearbox. The gearbox is an element that, although it does not have high failure rates, can break down and cause critical economic losses as well as the non-operation of the wind turbine itself. The repair times of this component are very high compared to the rest and the cost is relatively high too.

To avoid having very high maintenance costs, a good planning of maintenance tasks and a good use of preventive techniques is necessary. Currently, there are sensors on the market specialized in reading signals that can be characterized to find faults in the gearbox of a wind turbine. These sensors are usually costly and require a high capital expenditure investment in qualified personnel and time. This whole situation becomes more extreme when talking about wind turbines in the sea (offshore wind turbines). When the wind turbines are offshore, a significant and complicated logistics component is added to the maintenance of them.

There are other methods that can help plan predictive maintenance tasks and lower the high maintenance costs that these machines require for turbines that do not have the specialised sensors or such a high investment in them is not justifiable. This method is based on the use of SCADA data, which are available and in great quantity in almost every industrial-sized wind turbine.

Thanks to the use of SCADA data to generate a fault detection model applied to the gearbox, operation and maintenance costs can be reduced, and logistics tasks can be carried out only when they have to be done. These models are usually more affordable and justifiable in turbines with an advanced operating age.

1.2 Objectives & Scope

The objectives and scope of the project consist of several points:

- First, this project seeks to be able to make prediction models of the gearbox oil temperature based on SBL. This aims to take advantage of probabilistic models over deterministic models.
- This project wants to carry out the aforementioned for any wind turbine of a given wind farm in an on-demand way. For this reason, the models are made around a framework that takes advantage of the generalization of the code to be able to be implemented in other turbines.
- Finally, the project also aims to make a general diagnosis of the turbines of the entire wind farm. In this diagnosis, it will be possible to visually see where the model flags that there may be indications of anomalies that lead to failures.

This project will focus on a specific wind farm and will use the data morphology of that farm to design the framework and the type of output data to be displayed.

2. Background

2.1 Wind energy context

Wind turbines have become a very significant factor in promoting sustainable development in society, and it has many advantages:

- It is a type of renewable energy since it has its origin in atmospheric processes due to the energy that reaches the Earth from the Sun.
- It is a clean energy as it does not require combustion, so it does not produce atmospheric emissions or polluting waste, thus avoiding an increase in the greenhouse effect and climate change.
- It can be installed in spaces not suitable for other purposes, for example in desert areas, close to the coast, on arid slopes or too steep to be cultivated and even in the sea.
- Its installation is fast, between 4 and 9 months.
- The physical situation of wind turbines in dispersed countries such as Spain makes it possible to compensate for the low production of some wind farms due to lack of wind with the high production in other areas.
- It is possible to build wind farms in the sea, where the wind is stronger, more constant and the social impact is lower, although installation and maintenance costs increase.

The cost of wind turbines has been falling very markedly from 1980 to 2000. These reductions in the cost of the turbines were accompanied by very large increases in the performance of the turbines, resulting in more advanced turbines and larger towers. Thus, the combined effects of lower cost and higher production caused the monetary cost per unit of energy produced to drop dramatically, making this a competitive power generation technology.

Various studies (U.S. Department of Energy, 2008) (European Wind Energy Association, 2009) suggest that the Leveraged Cost Of Energy (LCOE) will continue to drop in price on a global scale.

The following figure shows 18 cost scenarios which use historical trends of the main markets:

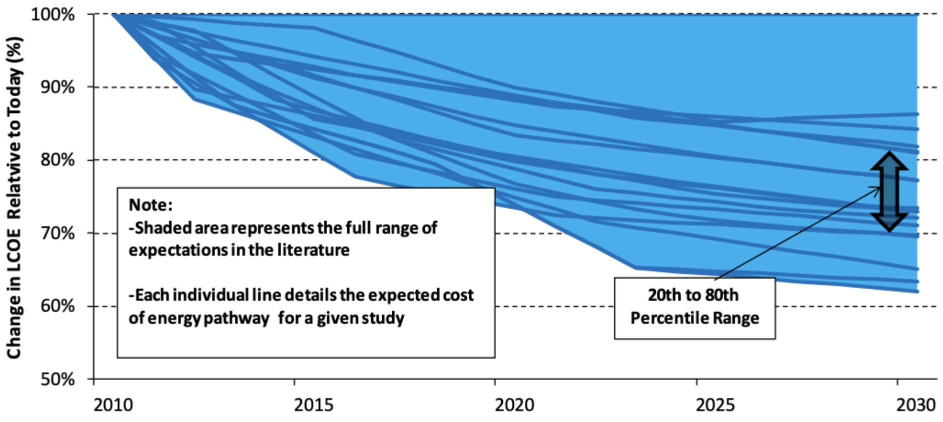


Figure 1. Estimated range of future wind LCOE. Source: NREL

All this has allowed wind energy to have achieved an important weight in the energy generation field around the world.
The presence of wind turbines has grown at 244% in the world and at 128% in the European Union (EU) in the last 10 years (IRENA 2020). In the EU, the most optimistic forecasts for the future of wind generation speak of a 300% increase by 2030 and a 900% increase by 2050 compared to 2018. Only offshore wind energy can accumulate an installed capacity of 1000 GW by 2050 (IRENA 2019).

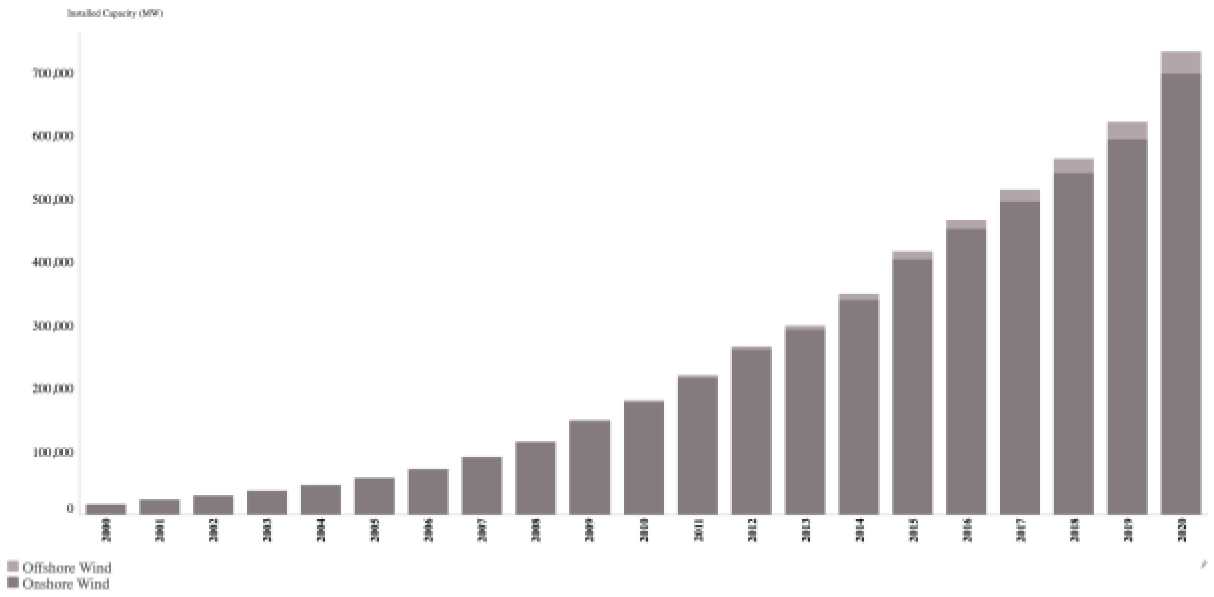


Figure 2. World wind turbine onshore and offshore trend from 2000 to 2020. Source: IRENA

However, future wind farm projects may encounter certain difficulties such as the lack of locations with high wind speeds near consumption points, restrictions in the electricity grid or the competitiveness of other projects. Social opposition can also be an important factor (Nadaï and Van der Horst, 2010).

2.2 Wind turbines context

It is clear that wind power is on the rise, but so far, we have only talked about the growth of turbines in the energy landscape. It is natural to focus on this, but one should not lose sight of the wind turbines already installed. In February 2020 Bloomberg published some shocking images of a landfill full of wind turbine blades. Turbines have been with us for many years, and they have an operative lifetime. It is important to know the current state of the turbines already installed as well.

Figure 3. Fragments of wind turbine blades in Wyoming. Source: C. Martin

Wind turbines will need to produce throughout their useful life to make the investment profitable. These machines typically have an operative lifetime between 20 and 25 years. If the components are renewed and the maintenance is adequate, it can be extended up to 30 years. As wind turbines are a technology that began to grow a few years ago, it can be seen a general aging of them. For example, in Spain:

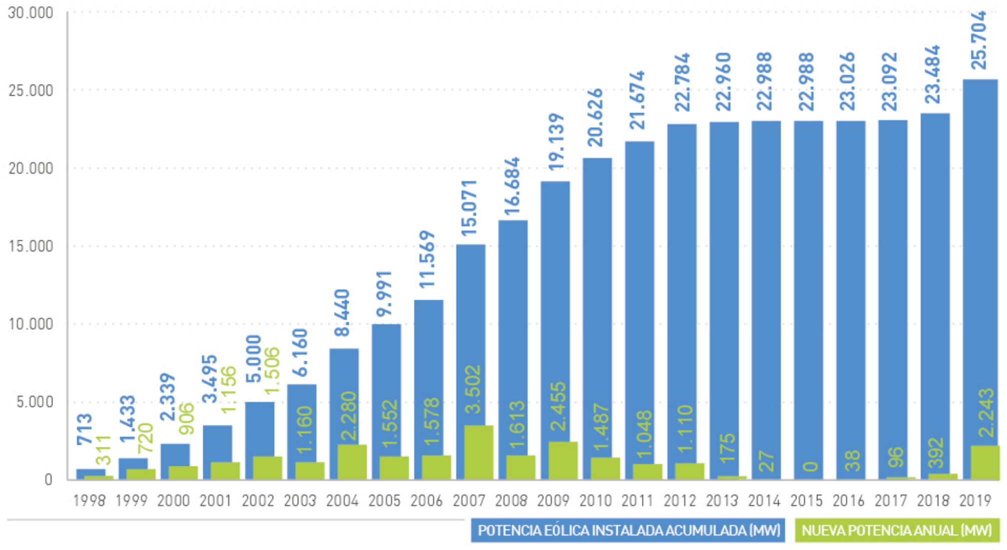


Figure 4. Installed power evolution, in green new installed power, in blue accumulated installed power (in Spanish). Source: AEE

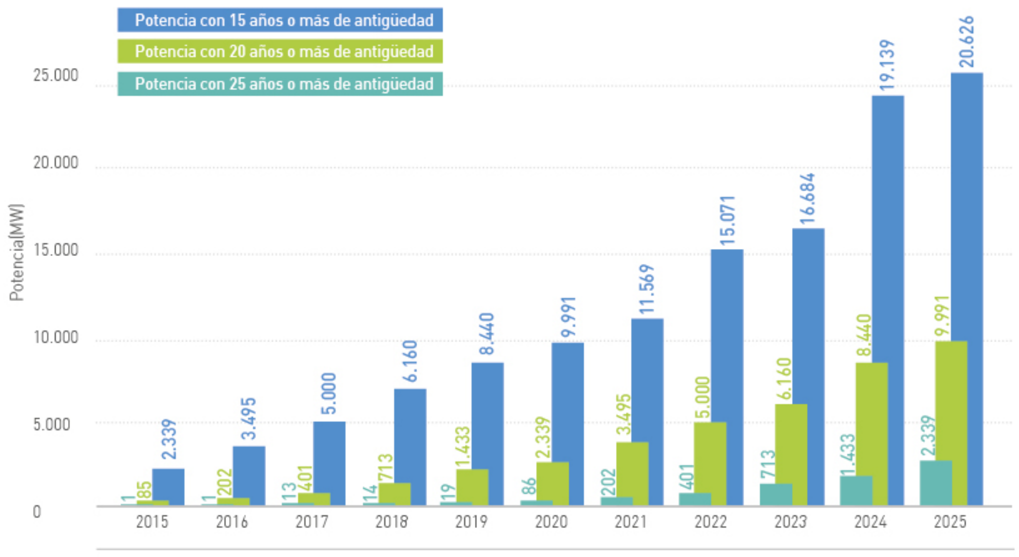


Figure 5. Installed power evolution, in green new installed power, in blue accumulated installed power. Source: AEE

It can be seen how half of the wind turbines installed in Spain will have reached an age equal to or greater than 15 years, but other countries are in the same situation. That is why it seems obvious to think that the operation and maintenance of wind farms will be a key element in the future of the global energy mix (Monforte, J. 2017).

Nevertheless, it is not only wind turbines that need maintenance because they are old. The environmental conditions of turbine operations are often hostile and damaging to the turbines. That is why wind turbines tend to have higher failure rates compared to other power generation technologies. Therefore, to avoid higher maintenance costs, it is necessary to plan well maintenance operations. These operations are expensive and can be even more expensive if they must be done at sea since not only operators are required but the means to get to the turbine such as ships and their crew.

2.3 Wind turbine maintenance context

Condition monitoring and early fault detection in wind turbines has become an essential practice in the industry. This practice helps improve the reliability of wind farms and their overall performance and productivity. If failures are not detected and rectified early, some of these failures can be catastrophic, leading to great economic and even environmental losses. A failure means that business is interrupted, and repair or replacement of affected parts can significantly reduce expected annual profits. A good fault detection system can play an important role in reducing operating and maintenance costs as well as increasing system reliability.

A wind turbine can be classified into several systems and subsystems. The most important are:

- Base: it is buried and is the base of the entire structure. In the case of being a wind turbine in the sea, its base is usually floating or with a tubular structure that reaches the seabed.

- Tower: The higher a turbine is, the more wind speed it will find. Therefore, the tower is a tubular structure that raises the generator to a suitable height.
- Rotor: The rotor is the part of the wind turbine that captures the kinetic energy of the wind. It is usually made up of 3 blades and a central part that may be able to rotate the angle of the blades to capture more or less kinetic energy.
- Nacelle: It is the structure that contains all the components that convert kinetic energy into electrical energy. It is worth highlighting the gearbox.
- Generator: It is in charge of converting the kinetic energy of rotation of an axis into electrical energy.

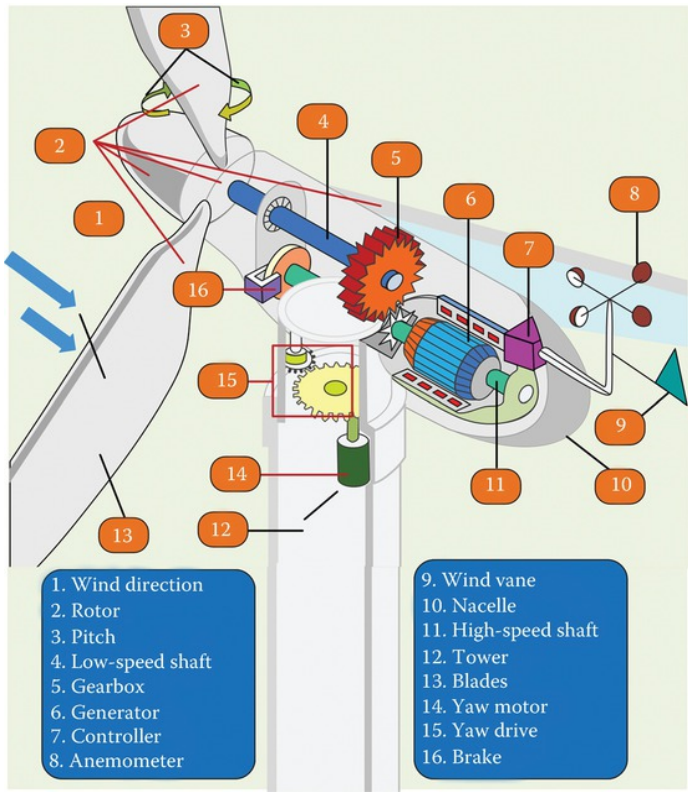


Figure 6. Principal components of a wind turbine. Source: Electrical Academia

In the diagram above, you can see the different components that normally exist in a conventional industrial wind turbine.

The gearbox is typically used in wind turbines as a means of increasing rotational speed from a slow spinning shaft (the spin caused by the blades) to a fast spinning shaft (the one that rotates inside the alternator and generates electricity). The conversion ratio is usually on the order of 90:1, that is, 16.7 rpm from the rotor to 1500 rpm to the generator.

Not all components of a wind turbine have the same failure rates, and the impact on the production of a failure in one component or another can be very different. One of the most crucial systems for the operation of a wind turbine are the electrical subsystem, the blades, the control system and the gearbox (Qiao and Lu, 2015). In the following image, you can see a more realistic aspect of the gearbox and its location between the slow axis and the fast axis:

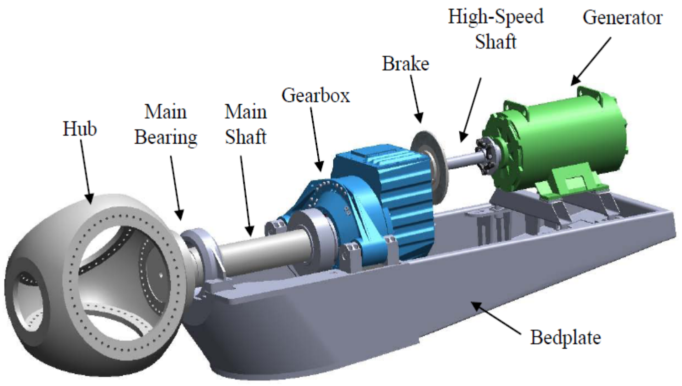


Figure 7. Nacelle components. Source: F.Oyague

In the following figure, it can be seen the internal components view of a standard gearbox:

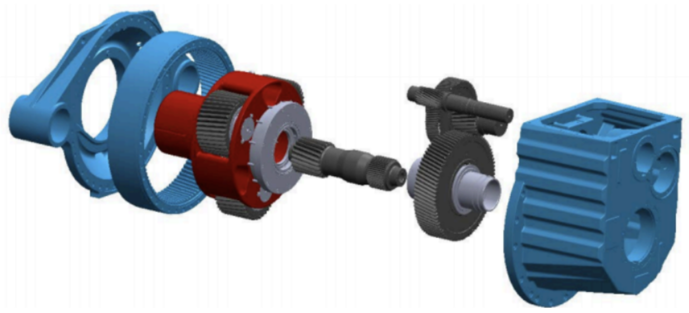


Figure 8. GRC gearbox internal components view. Source: F.Oyague

Although the latter does not have a failure rate as high as the other components, it entails a much higher downtime and replacement repair cost than the rest. In onshore wind turbines, it is estimated that the mean shutdown time when the gearbox fails is usually between 6 and 18 days (Faulstich et al., 2010). For offshore wind turbines, the downtime is multiplied considerably by having to face more severe logistical challenges. Therefore, predictive maintenance in the gearbox is desirable if what is wanted is to reduce the costs of maintenance and operation.

But what can be damaged in a gearbox? Several bearing types are employed in the gearbox, according to the loading conditions and gearbox life requirements. The planet carrier is supported by two full-complement cylindrical roller bearings, and each planet gear is supported by two identical cylindrical roller bearings (CRB). Each parallel shaft in the gearbox is supported by a CRB on the upwind side of the assembly, and by two back-to-back mounted, tapered roller bearings on the downwind side. The typical failures of a gearbox are gear bearing and gear tooth failures.

With the development of new communication technologies and extensive use of sensors in wind turbines, wind farms are monitoring numerous parameters and characteristics of wind turbines. Normally, wind turbines are monitored through a system called SCADA (Supervisory Control and Data Acquisition). SCADA data are those that every industrial-size wind turbine measures employing different sensors for its operation, such as regulating the power generated. This data can be used to maintain them, since if a parameter goes outside its usual range it can be an indicator that something is not working properly.

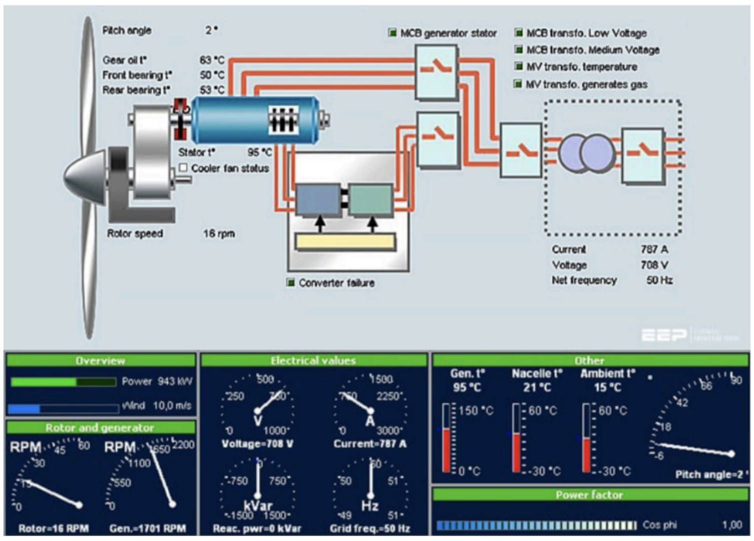


Figure 9. Design of SCADA for remote control and operation of wind power plants.
Source: sparesinmotion.com

SCADA data is typically collected into a file. In this file, it can be found all the data the sensors have generated through the operating time of the wind turbine. Often the data collected will have for every reading its mean, maximum value, minimum value and its standard deviation of a 10-minute interval. Typically, the frequency of the recorded data is 1 minute or 10 minutes.

As mentioned before, many wind turbines are already installed and their fault detection systems are old. Currently, there are specific methods to predict failures, but they require installing expensive sensors that are not found in all industrial wind turbines. The installation of the sensors, in addition to being expensive, would mean a prolonged production halt since its installation would require the dismantling of a large part of the turbine transmission system to be located in the right place.

2.4 Anomaly detection context

Anomaly detection devices are effective methods of preventing failures and discovering potential failures by analyzing the operational state of the system. Thanks to these systems, operators can make decisions before components deteriorate and act. These acts increase the stability and reliability of the system.

These systems have been used historically in fields such as petrochemicals, machinery production and electrical systems (Hossain et al., 2018). These anomaly detection systems can be classified into two categories: Model-based methods and Data-driven methods.

Model-based methods are dependent on precise mathematical models that simulate the dynamic behaviors of a system. These systems are complex and require a series of mathematical models that need to have a good tuning of their parameters and also be sufficiently descriptive of the system they are trying to replicate. These methods can be effective in the early detection of failures. However, these models are very difficult to get right and to perform accurately. Model-based methods are even more complicated to develop in wind turbines, since they need to replicate an electromechanical hybrid model. This hybrid model not only contains several subsystems, but also presents a high variability of operating conditions that can greatly hinder the success of a mathematical model. It is for all these reasons that mathematical models are not widely used in real wind turbine applications (Qiao and Lu, 2015).

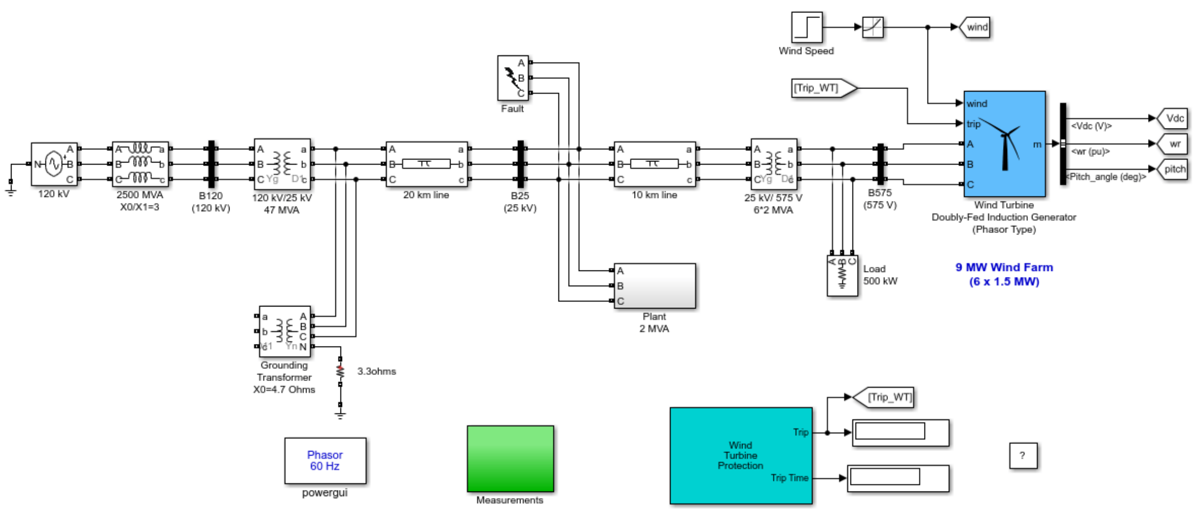


Figure 10. Model-based MATLAB-Simulink of a wind turbine. Source: Gagnon R., et al.

Conversely, we have data-driven methods. These methods imply that you are going to make decisions based on data collected from your area of interest. These methods are usually carried out using signal processing, statistical analysis, and machine learning. Compared with model-based methods, data-driven methods show great flexibility, reliability and a large field of application. Still, for a data-driven method to be successful, it is necessary to have a large amount of good quality data and a good source of computational power.

As previously discussed, thanks to the extensive use of sensors in wind turbines, there is a large amount of data available that opens the way to the use of data-based fault detection methods. In addition, the high cost, installation and adaptation price makes the implementation of specialized anomaly detection sensors unfeasible. This favors the choice of a method that allows to take advantage of existing large databases.

When someone is capable of making a series of technical decisions with the aim of detecting possible failures and defects of machinery in incipient stages, or before they occur, it is often said that predictive maintenance is carried out. Predictive maintenance prevents emergency downtime and non-productive stages. Catastrophic erosion of the machinery could lead to total failure and in the case of gearbox, this degradation could appear 6 or 7 months before its total failure. All this causes a positive productive and financial impact.

The data-driven methods need data that can come from two different approaches:

- Condition monitoring sensors
- SCADA historical data

Both methods will be seen in more detail in the state-of-the-art section of this project.

2.5 Project's use case

This project is based on a client's need to detect failures before they occur in their wind farms. As mentioned above, the gearbox is an element that when it fails causes great production and economic losses. Installing specialized sensors to detect gearbox anomalies is an expensive and complicated task, and that is why it is preferred to examine other alternatives.

The alternative chosen in this case is to make use of the data from the SCADA system of the wind turbines to design a model based on data that is capable of detecting failures before they occur.

These data-based models need to be trained, this training must be done with data from the turbine to be modelled. This data must be healthy for the model to learn to replicate the healthy behavior of the turbine. If there were failures in the training data, the model could understand that this type of behavior is normal. If the model replicates irregularities in the operation, we will not be able to effectively detect these irregularities.

The approach that will be followed in this project is to compare the predicted result with the actual result. The subtraction of these two values will give us something called the "residual". When the estimation error is large, the residual is also large. It is the high residuals or the large estimation errors that make the model capable of detecting irregularities. For all the above, that is why it is important that the data to be used is clean and characterized by being healthy.

This project has had the opportunity to access real turbine data, so it is a great opportunity to see how the model responds to this type of data. The wind farm for which we have data cannot be disclosed as there is a non-disclosure agreement:

Section	Operator	Wind turbine model	Number of wind turbines	Total nominal power
1	Cannot be disclosed	Cannot be disclosed	28-36	~50.000 kW
2	Cannot be disclosed	Cannot be disclosed	2	~6.000 kW
3	Cannot be disclosed	Cannot be disclosed	1	~3.000 kW
		Total	30-40	~60.000 kW

Table 1. Wind farm main characteristics. Source: Self-made

The wind farm is located in Europe.

The first section of the wind farm is the one that generates the most and the one that consists of the most number of turbines. The turbines of this first section have a unitary power of around 1500-2000 kW with a sweep diameter of the blades of 77 m and reach a height of up to 80 m. The model is relatively old and was put into operation in 2006, so it has more than 15 years of service. The gearbox of this model is made up of 4 levels and a 1:65 ratio.

In the second section there are 2 turbines that generate around 3,000 kW each with a blade sweep of 116 m in diameter and a maximum height of 120 m. It is a newer and notably larger model with a start-up in 2012. The gearbox has 3 levels and a transformation ratio of 1:89.

The third and last section are made up of a single wind turbine. This turbine also generates 3000 kW, but its sweep diameter is wider, 125 m. Its gearbox is identical to the previous section. Its commissioning is the most recent, in 2013.

All these wind turbines have a SCADA system that captures the signals from the sensors that the wind turbine itself has to guarantee its operation.

It is also important to bear in mind that the gearbox is one of the biggest and most difficult components of a wind turbine. Within a gearbox one can find numerous different types of faults, so not all faults are equal, have the same origin or cause the same consequences. This problem is as hard to solve as complex is the component.

3. State of the art

Before starting to talk about our use case, we will review where data-based decision-making technology is and how it is being applied in the world of wind turbines and specifically in the detection of anomalies in the gearbox.

There are generally two main methods for detecting anomalies with data-driven methods:

- Condition monitoring (CM) data-based methods
- Supervisory control and data acquisition (SCADA) data-based methods

Condition monitoring is the process that monitors a particular condition in a machine. This parameter can be either vibration, temperature, tension or any other measurable value. This method aims to extract signatures from the signals captured by monitoring. These parameters are captured using highly specialized sensors that operate at high sampling signals frequencies and that allow, after processing the signal obtained, the identification of changes that may end up developing into a failure

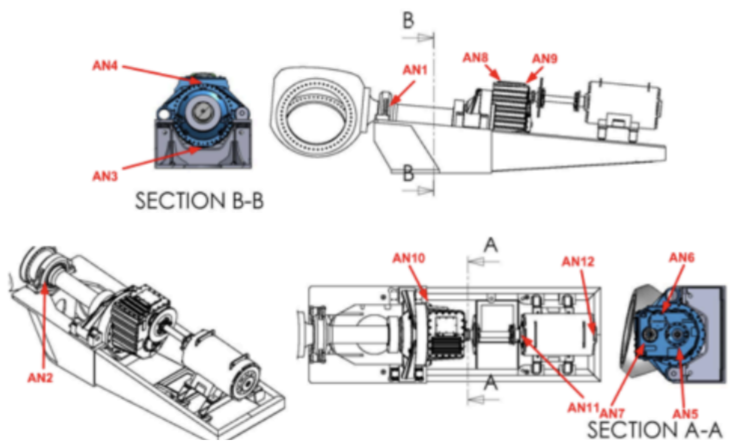


Figure 11. Vibration data acquisition system sensor locations. Source: Y. Vidal

Sensor Label	Description
AN1	Main bearing radial
AN2	Main bearing axial
AN3	Ring gear radial 6 o'clock
AN4	Ring gear radial 12 o'clock
AN5	LSS radial
AN6	ISS radial
AN7	HSS radial
AN8	HSS upwind bearing radial
AN9	HSS downwind bearing radial
AN10	Carrier downwind radial
AN11	Generator drive end radial
AN12	Generator non-drive end axial

Figure 12. Sensor notations and descriptions. Source: Y. Vidal

In the figures above, it can be observed a data acquisition system based on specialised vibratory sensors and the sensors it comprises. These sensors form the gearbox-specific condition monitoring.

One of the approaches used with this technique is the use of vibration sensors in the bearings of the gearbox to capture the signal for different load states. This method proposes the use of a wavelet energy transmissivity function (Zhang and Lang, 2018).

In another study, the approach of using vibration sensors was followed, but in this case, apart from vibrations in the bearing, oil debris was also used. This approach sought faults in the slow speed axis of the gearbox (Feng et al., 2013)

Continuing with the use of vibrational signals and wavelet functions, the following approach sought weak predictors at times of intensive generation (T Wei, 2016). In the study by (Wang et al., 2018) a multiscale reconstruction method is proposed that allows efficient filtering of vibratory signals and eliminating unnecessary noise in the sample, which allows better analysis of vibratory signals.

A slightly different approach was to use electric intensity sensors which are installed in the generator. These sensors detect electrical frequencies in the stator, which can identify faults since they can be characterized (Cheng et al., 2019).

Methods that rely on the use of sensor signal capture can undoubtedly increase the chances of detecting faults ahead of time. After all, these sensors are usually installed in well-studied and specific areas to capture the signals that can lead to early fault detection. However, this method usually requires costly sensors. They are not only expensive, but they need to be installed, as seen in previous applications, in the gearbox or in places that are difficult to access. These places usually require a partial dismantling of the turbine, which entails a certain period of non-production in addition to the expense in the installation. The purchase and installation of these sensors cannot always be justified for wind turbines that are already installed and that may be nearing the end of their lifetime. Currently, there are many turbines already installed in the world that such a large investment does not make sense if this turbine has already been in operation for most of its lifetime.

In contrast, the vast majority of industrial-size wind turbines are equipped with SCADA equipment, which generally monitors the operation and performance of the installation. It not only measures internal operating parameters such as oil pressure, generator temperature or blade tilt angle, but also external parameters such as wind speed or ambient temperature.

Often it does not make sense to install expensive sensing equipment for a turbine that has been installed for a long time due to its age. Nevertheless, it may have a large amount of monitoring data stored throughout its operational life. The availability of large amounts of data makes the SCADA a potential data source. Thus, designs of anomaly detection models fed with data from the SCADA can be performed. That is why this method usually has a cost-efficiency advantage over the method based on conditioning monitoring for turbines that have already been in operation for many years.

In the literature, we can find many application cases, some more complex than others:

The first use case it can be found is using data from the SCADA system to detect early failures of the gearbox. This is achieved through the detection of anomalies in the relationship between the fast shaft torque and the generator output power. This approach achieves this by clustering and applying principal component analysis (PCA) to the SCADA data (Parthasarathy et al., 2011).

In a second use case, we can find that failures are sought in the gearbox by using the variables of the energy produced and the temperature and angle of the blades through a classification algorithm (Leahy et al., 2016).

In other cases, we can see that the use of domain expertise can be very beneficial to achieve a better model. (Konstantakopoulos et al., 2016) uses this knowledge to expand the predictors that can help find faults and uses them together with an algorithm called Support Vector Machine (SVM). This method was shown to increase the accuracy of detecting anomalies in the gearbox.

Cases can also be found where deep neural networks are used to identify early failures in the gearbox by detecting anomalies in its oil pressure (W.L. et al., 2017).

(Alvarez and Ribaric, 2018) proposes the use of a torque histogram which, if combined with the damage model of the gearbox itself, allows calculating the useful lifetime that the component has left.

The methods discussed above seek to perform anomaly detection through variables that normally do not take temperature into account. When the mechanical components of the gearbox are degrading they produce extra heat to the system which can affect the useful life of the components. In a SCADA system of a wind turbine there are many temperature measurements such as the slow shaft temperature, the fast shaft temperature, the gearbox oil temperature, etc.

The methods specialized in finding anomalies in the temperature parameters of wind turbines are usually more convenient and efficient, which is why it is usually the parameter that is most used to detect failures before they occur (Sequeria et al., 2019). There are many works that have been done in this line.

For example, (Infield and Wang, 2013) built a non-linear model that estimates the gearbox oil temperature. Performing a Welch's t-test and a filtering method attempts to detect temperature anomalies.

In another study, SVM is used to estimate the gearbox oil temperature. The prediction of this temperature is later used to generate an alarm when the residual between the prediction and the actual reading crosses a certain threshold (Zhang and Qian, 2017).

You can find studies that apply Neural Networks such as (Huang et al., 2018) which uses an autoregressive Neural network model. This model predicts the gearbox oil temperature and once again the evolution of the residuals indicates whether there are signs of failure or not. An approach very similar to that is (Cui et al., 2018) which uses a neural network to estimate the gearbox oil temperature, but in this case, the Mahalanobis distance of residuals is used to determine if the gearbox operation is normal or abnormal.

Finally, the following study looks at the approach of (Xu et al., 2019) which uses a quantile regression neural network algorithm that estimates temperature. Giving certain weights to the residuals determines if the gearbox is in an unusual operation or not.

All of these methods use temperature in an attempt to perform deterministic anomaly detection. This method looks for anomalies by comparing the prediction of the real data, and it is in this residual that it is focused to determine if there is an anomaly or not. The downside of these methods is that they assume that the irregular values of the residuals are determined by anomalies in the oil temperature. They do not consider that these irregularities can be caused by other internal factors, such as oil pressure, as well as external factors like external temperature. Also, some irregularities in the residuals could come from simple estimation errors.

For all these reasons, deterministic estimation methods can lead to errors when detecting anomalies in wind turbines.

In order to face all these limitations, the use of probabilistic estimation models can be beneficial. These models can quantify the uncertainty of a gearbox oil temperature value. There are several methods that allow this type of estimation, such as:

- Kernel Density Estimation (KDE)
- Quantile Regression (QR)
- Least Squares Support Vector Machines (LSSVM)
- Sparse Bayesian Learning (SBL)

Among all of them, the SBL stands out, which is an algorithm with excellent generalization skills. This algorithm calculates the weight of the correlation vectors by maximizing the posterior probability, thus highlighting its performance with data not seen or not used for training. Furthermore, this algorithm proves to be efficient since it simplifies the complexity of the model by pushing some weights to 0 if they do not have any important role in the model.

It is a method also used in wind turbines, as in the study by (Yang et al., 2013) which uses SBL to predict the power generated by the turbines or the study by (Beck and Huang, 2015) which uses SBL to calculate the probability of structural damage to the turbines.

Considering the advantages offered by the SBL algorithm, this project will make use of it. Thus, a model based on SBL will be made by which a series of probabilistic estimates will be established for the prediction of the gearbox oil temperature.

4. Methodology

The objective of this project is to be able to visualize an analysis of the prognosis potential of a model based on data and on a sparse Bayesian method for the entire wind farm. For this, several modules have been created and executed in succession. The general scheme of the different modules is as follows:

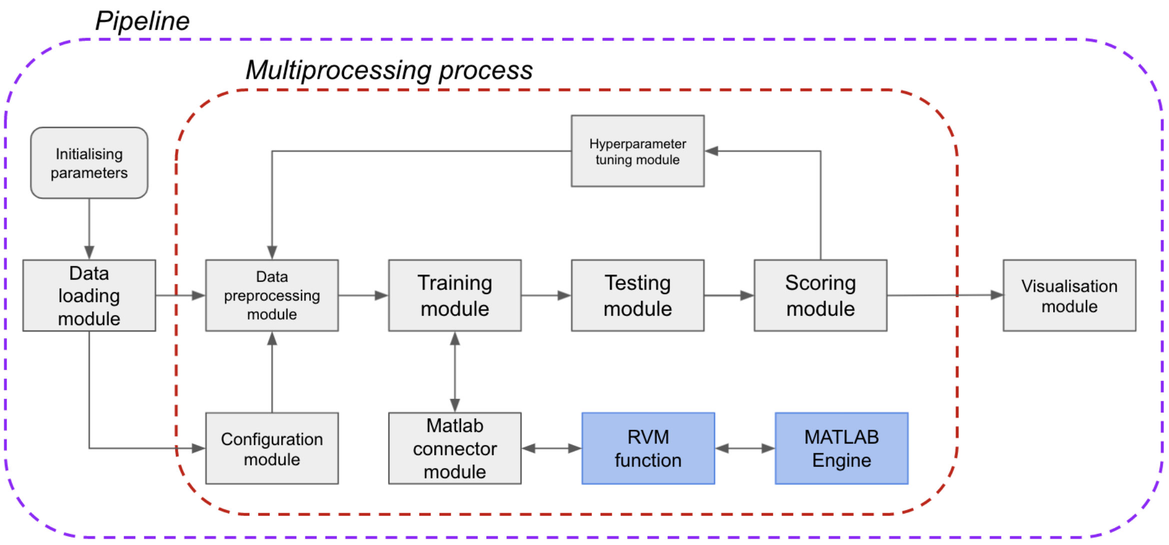


Figure 13. Schematic of the different modules and their workflow. Source: Self-made

As can be seen in the previous diagram, there are two large blocks; the pipeline and the multiprocessing process. The pipeline contains everything and contains the entire process from start to finish. The multiprocessing block is a loop that takes advantage of the different threads of the CPU to be able to execute up to 12 simultaneous executions in parallel. This way, the most suitable hyperparameter value for training the model can be found more quickly.

The most important initialization parameters are the following:

- Approach: It is in charge of defining which approach of predictors is used for the execution.
- Kernel: It is responsible for defining what type of kernel should be used to train the model.
- Target: It is responsible for defining which variable should be predicted: normally the gearbox oil temperature will be used in latter analysis.
- Anomaly factor: It is responsible for defining how demanding the model will be to detect values as abnormal or healthy.
- Random features: It is responsible for activating the use of predictors randomly with an exploratory purpose.
- Hyperparameter tuning: It is in charge of establishing the initialization data of the hyperparameter tuning module.

All modules, except the blue ones, have been developed in Python 3 (version 3.8). The modules in blue have been developed in MATLAB (version R2020b).

4.1 Database

As previously mentioned, this project starts from the data provided by the client. The customer has a need to find a method by which wind turbine failures are detected before they occur. One of the failures that the customer wants to avoid is the failure of the gearbox. That is why the client provides the data, to develop a data-based method, thus avoiding not only potential gearbox failures but spending on specialized vibration sensors.

The data is collected in .csv format files. Each file contains the historical data of a turbine, which is numbered in the filename itself. The files are as follows:

Filename	Size (MB)	Filename	Size (MB)
WT1	330,57	WT17	355,37
WT2	351,06	WT18	362,58
WT3	346,61	WT19	352,43
WT4	352,14	WT20	338,35
WT5	338,86	WT21	347,09
WT6	347,29	WT22	342,23
WT7	331,79	WT23	340,24
WT8	340,02	WT24	348,26
WT9	330,04	WT25	349,39
WT10	333,10	WT26	348,66
WT11	332,68	WT27	361,78
WT12	347,91	WT28	338,43
WT13	334,87	WT29	332,39
WT14	339,51	WT30	351,17
WT15	354,29	WT31	361,00
WT16	338,70	WT32	340,93

Table 2. Wind turbine monitored parameters by the SCADA. Source: Self-made

The files occupy a total disk space of 10.76 GB and total 32 files. The signals that we can find in each file are the following:

- Environment parameters:

Environment SCADA monitored parameters		
IndTurbulMean	TempAmbMean	VelVientoMean
IndTurbulSdev	TempAmbSdev	VelVientoSdev
IndTurbulMin	TempAmbMin	VelVientoMin
IndTurbulMax	TempAmbMax	VelVientoMax

Table 3. Wind turbine monitored environment parameters by the SCADA. Source: Self-made

- Electrical parameters:

Electrical SCADA monitored parameters				
ContEnerActivaMean	FrecRedMean	TensRedMean	TotPotReactMean	CosPhiMean
ContEnerActivaSdev	FrecRedSdev	TensRedSdev	TotPotReactSdev	CosPhiSdev
ContEnerActivaMin	FrecRedMin	TensRedMin	TotPotReactMin	CosPhiMin
ContEnerActivaMax	FrecRedMax	TensRedMax	TotPotReactMax	CosPhiMax
Diag1_R4Mean	PotMean			
Diag1_R4Sdev	PotSdev			
Diag1_R4Min	PotMin			
Diag1_R4Max	PotMax			

Table 4. Wind turbine monitored electrical parameters by the SCADA. Source: Self-made

- Temperature parameters:

Temperature SCADA monitored parameters				
TempAceiteGHMean	TempAceiteMultipMean	TempCojLOAMean	TempGenMean	TempGondMean
TempAceiteGHSdev	TempAceiteMultipSdev	TempCojLOASdev	TempGenSdev	TempGondSdev
TempAceiteGHMin	TempAceiteMultipMin	TempCojLOAMin	TempGenMin	TempGondMin
TempAceiteGHMax	TempAceiteMultipMax	TempCojLOAMax	TempGenMax	TempGondMax
TempRadInfMean	TempRadSupMean	TempRodamMultipMean	TempCojLAMean	TempRodamTraseroMean
TempRadInfSdev	TempRadSupSdev	TempRodamMultipSdev	TempCojLASdev	TempRodamTraseroSdev
TempRadInfMin	TempRadSupMin	TempRodamMultipMin	TempCojLAMin	TempRodamTraseroMin
TempRadInfMax	TempRadSupMax	TempRodamMultipMax	TempCojLAMax	TempRodamTraseroMax
TempRodTraseroMean	TempTrafo1Min	TempTrafo2Mean	TempTrafo3Mean	
TempRodTraseroSdev	TempTrafo1Max	TempTrafo2Sdev	TempTrafo3Sdev	
TempRodTraseroMin	TempTrafo1Sdev	TempTrafo2Min	TempTrafo3Min	
TempRodTraseroMax	TempTrafo1Mean	TempTrafo2Max	TempTrafo3Max	

Table 5. Wind turbine monitored temperature parameters by the SCADA. Source: Self-made

- Other wind turbine parameters:

Wind turbine wise SCADA monitored parameters				
AcumGralPala1Mean	AcumGralPala1Max	AcumGralPala3Mean	ConfirmContractGiroCCWMean	ConfirmContractGiroCWMean
AcumGralPala1Sdev	AcumGralPala2Mean	AcumGralPala3Sdev	ConfirmContractGiroCCWSdev	ConfirmContractGiroCWSdev
AcumGralPala1Min	AcumGralPala2Sdev	AcumGralPala3Min	ConfirmContractGiroCCWMax	ConfirmContractGiroCWMin
AcumGralPala1Max	AcumGralPala2Min	AcumGralPala3Max	ConfirmContractGiroCCWMin	ConfirmContractGiroCWMax
NivOscilMean	NivVibraMean	EstadoCalc	Pitch1Mean	Pitch2Mean
NivOscilSdev	NivVibraSdev	Flag2Mean	Pitch1Sdev	Pitch2Sdev
NivOscilMin	NivVibraMin	Flag2Min	Pitch1Min	Pitch2Min
NivOscilMax	NivVibraMax	Flag2Max	Pitch1Max	Pitch2Max
Pitch3Mean	PresFrenoMean	PresAcumGralSMean	PresGHMean	SPPitchMean
Pitch3Sdev	PresFrenoSdev	PresAcumGralSdev	PresGHSdev	SPPitchSdev
Pitch3Min	PresFrenoMin	PresAcumGralMin	PresGHMin	SPPitchMin
Pitch3Max	PresFrenoMax	PresAcumGralMax	PresGHMax	SPPitchMax
VelGenMean	VelRotorMean	YawMean		
VelGenSdev	VelRotorSdev	YawSdev		
VelGenMin	VelRotorMin	YawMin		
VelGenMax	VelRotorMax	YawMax		

Table 6. Wind turbine monitored parameters by the SCADA. Source: Self-made

As it can be observed, the parameters come in pairs of four variables since they are the mean, the minimum value, the maximum value and the standard deviation of a variable that has collected its readings for 10 minutes.

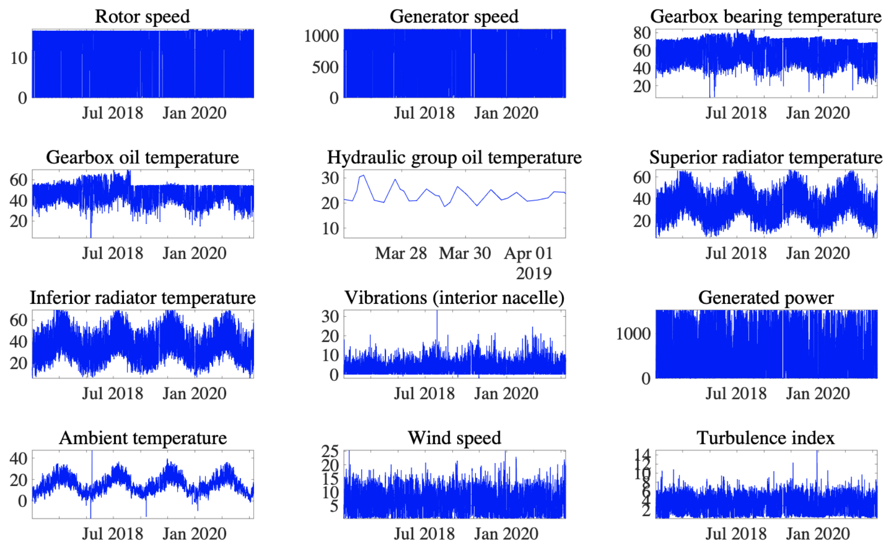


Figure 14. Preview of wind turbine 2353 signal variables. Source: Self-made

Above is a figure in which it can be seen what some of the most important variables look like. All variables are the average readings of each 10-minute signal.

The time series comprising these readings goes for all turbines from 2016 to 2021. Each week the customer updates their readings so that data scientists can work with the latest ones to provide more data to the models. In total for each signal there are 315,360 readings, considering that there are 170 signals this means 53,611,200 readings for each turbine that we have data.

As introduced in the "State of the art" section, special attention will be paid to temperature variables. In the previous image, you can see that these are:

- Gearbox bearing temperature
- Gearbox oil temperature
- Hydraulic group oil temperature
- Superior radiator temperature
- Inferior radiator temperature
- Ambient temperature

All of them present a certain pattern that coincides with the diurnal and nocturnal cycles of the day. That is to say, during the day a notable increase in temperature can be observed and at night one can observe that drops. We also find a similar pattern in the seasons of the year. In summer, the temperature increases, while in winter drops.

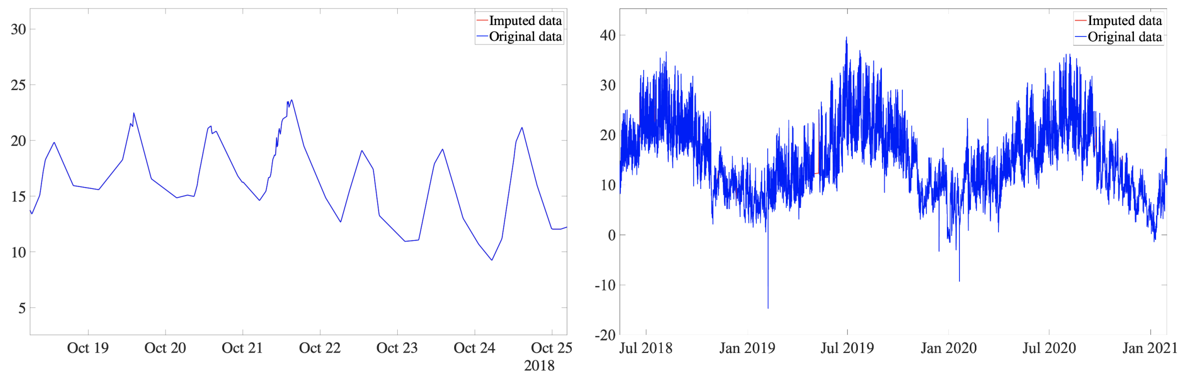


Figure 15. Daily (left) and seasonal (right) temperature patterns. Source: Self-made

The gearbox oil temperature, by contrast, suffers from a peculiarity and that is that the temperature is regulated. Understanding this is vital to understand the results of the model. Later, in the results section, it can be seen why.

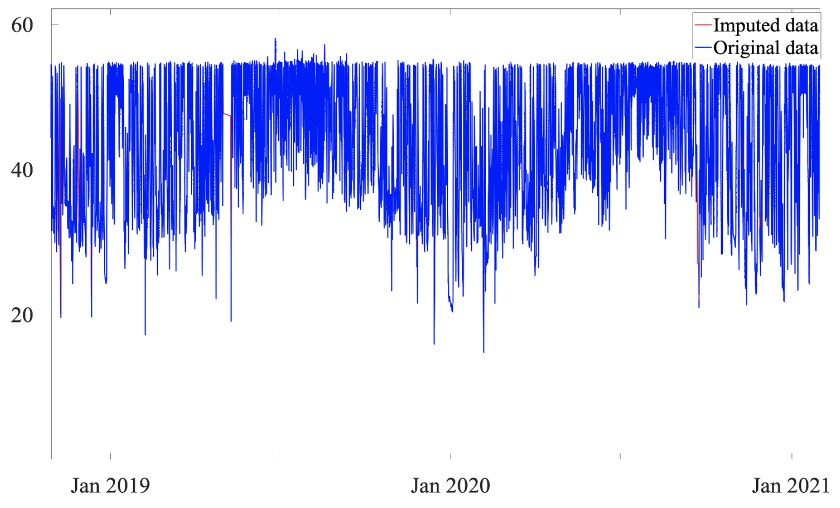


Figure 16. Gearbox oil temperature signal profile. Source: Self-made

In the previous figure, you can see the profile that shows the gearbox oil temperature. As it can be seen, this profile follows both a daily and seasonal temperature profile, but also has the particularity of being bounded superiorly. This temperature is regulated by a valve that opens the way to more or less refrigerant depending on the temperature of the fluid. When the oil temperature exceeds 55ºC, the valve acts. This is why we see almost a horizontal border across the top of the signal.

This type of knowledge is essential and is often called domain knowledge or "expertise". Thanks to this, one can better interpret both the input and output data, as well as calibrate the model to respond satisfactorily to these details. For this work, various combinations of signals will be used, but before making use of them, a good preprocessing is necessary so that the model does not have cracks.

Finally, the customer has provided a small database on turbine failures in recent years. In them, you can see the work order generated when the maintenance work was carried out, which turbine was affected, the date of the failure and its nature. The table is as follows:

WT ID	Date time	WT ID	Date time
WT29	10/29/17 11:40	WT10	5/15/19 9:30
WT26	5/9/18 11:47	WT11	8/28/19 8:00
WT19	7/9/18 10:01	WT5	5/19/20 8:30
WT2	8/20/18 18:50	WT23	9/15/20 8:45
WT13	11/14/18 9:00	WT1	9/21/20 9:00
WT16	1/25/19 18:14		

Table 7. Wind turbine's gearbox fault records. Source: Self-made

It is noted that the table contains failures since 2017 and in 11 different turbines. Although the data provided is very limited and without any detailed description, this can be used to analyse the model trained and evaluate its performance.

4.2 Preprocessing

The data provided by the turbines cannot be used raw. The signals require a process before their use, commonly called pre-processing.

In the same way that a cook needs to wash vegetables, peel potatoes, marinate the meat and measure the quantities well before starting to prepare a dish (among other things), a data scientist needs to perform a series of preliminary operations before starting to solve the problem, you are trying to make the data "edible". Due to the large amount of data that is generated, this first step is usually of vital importance to correct multiple deficiencies that we can find and to be able to really extract the relevant information for our problem. It is very common to hear data scientists comment that around 80% of the work time is focused on obtaining, cleaning and organizing the data, while only 3% of the time is dedicated to building machine learning models.

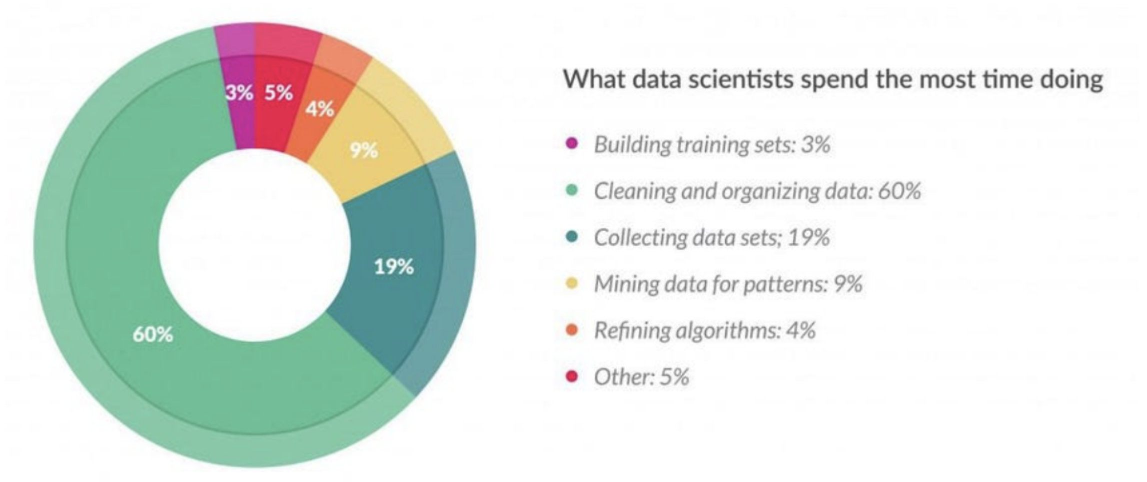


Figure 17. Survey results about what data scientists spend the most time doing. Source: G. Press

To do the cleaning of the data that will be used later, there are several techniques and steps that the vast majority of data-based projects carry out. The tasks done in this project are:

- Clean up of absence of values or NaNs
- Format adjustments (mainly dates when dealing with time series)
- Clean up of non-useful predictors (feature selection)
- Outlier detection or establishment of a range of values
- Training set and test set
- Standardization

Data loggers, whose function is to focus all the signals in a single point and process and store them, do not always capture all the data. Sometimes due to power failures or interferences, the time series is not captured. This causes that in the historical data some data does not exist for some dates, and they are stored with dummy values or "NaNs". Their presence can hinder or even make it impossible to use the data. That is why it is important to correct them.

There are several strategies to correct the lack of historical data at specific times. These strategies go through:

- Eliminate the signal containing NaNs, for example, if the vibration signal of a bearing contains NaNs, all its readings are eliminated. It is a strategy that penalizes a lot when you have little data or critical variables.
- Eliminate the timestamp that contains NaNs, in this case only the timestamp that contains the NaN would be eliminated but not only from the signal that contains the NaN but all the variables in that timestamp.
- Replace the NaN value with the immediately preceding value in the same signal.
- Replace the NaN value with the mean or median of the signal.

There are a multitude of strategies, but these are usually the most common and simple. In the case of this project, the strategy of eliminating the timestamp containing NaNs is used. As seen in the section where the data was explored, each signal has a maximum of 315,360 readings. Thus, removing the timestamp containing NaNs does not mean significant data loss. For the turbines analyzed, on average 4.39% of the available data is eliminated with this strategy. The use of this strategy sacrifices part of the available data in order not to affect the real performance profile of the turbine.

After cleaning the "NaN" values, the format of all variables is adapted. For the dates, they are forced to follow the following form:

```
YYYY-MM-DD HH:MM:SS'
```

So, any date will be, for example, like this: 08-22-2021 12:32:41. In this way, a variable that will be important in the model is standardized.

For the rest of the variables, they are simply forced to be of the float type, that is, they are numbers with the ability to be positive and negative in addition to containing decimal numbers.

The next process that needs to be done is to select only the predictors that can be beneficial to the model. This is called feature selection or predictor selection.

While it may seem reasonable that complex problems require complex models, many studies show that generally simpler models tend to reproduce more accurate predictions (A. Zellner, 2001). For this, a correlation study of the signals shown above has been carried out. The maximums, minimums and standard deviations of the available readings are set aside to use only the 10-minute mean value. The correlation method chosen is Pearson's. The Pearson correlation coefficient can be defined as an index that can be used to measure the degree of relationship of two variables, as long as both are quantitative and continuous. Its interpretation is simple, and a quick summary would be the following:

- Pearson's coefficients range from -1 to 1.
- If the coefficient is close to 1 it means that there is a positive correlation, that is, when one increases, the other increases proportionally.

- If the coefficient is close to -1, the correlation is negative, that is, when one increases, the other makes it inversely proportional.
- If the coefficient is close to 0 it means that there is no linear relationship.

The following table shows the Pearson correlation study for the WT1 turbine against the objective variable TempAceiteMultipMean (TemperatureOilGearboxMean):

Signal	Coefficient	Signal	Coefficient	Signal	Coefficient
TempRodamMultipMean	0,9341	Pitch1Mean	-0,6415	AcumGralPala2Mean	0,1299
TempCojLAMean	0,8477	TempTrafo1Mean	0,5793	AcumGralPala3Mean	0,1277
TempGenMean	0,8188	TempTrafo3Mean	0,5709	AcumGralPala1Mean	0,1273
TempCojLOAMean	0,8007	TempTrafo2Mean	0,5616	PresGHMean	0,1195
IndTurbulMean	0,7547	ContractGiroCWMean	0,5547	Flag2Mean	-0,1111
VelVientoMean	0,7505	TotPotReactMean	-0,5376	ContEnerActivaMean	-0,0836
TempRadInfMean	0,7461	CosPhiMean	-0,5329	PresFrenoMean	-0,0778
VelGenMean	0,7448	TempRodTraseroMean	0,5245	TempGondMean	0,0463
VelRotorMean	0,7433	NivOscilMean	0,4264	Diag1_R4Mean	0,0342
TempRadSupMean	0,7402	TempRodamTraseroMean	0,4101	FrecRedMean	0,0163
PotMean	0,7307	YawMean	0,3379	ContractGiroCCWMean	0,0161
NivVibraMean	0,6654	TempAceiteGHMean	0,2376	SPPitchMean	0,0024
Pitch2Mean	-0,6418	TempAmbMean	0,1738	TensRedMean	-0,0024
Pitch3Mean	-0,6418	PresAcumGralMean	0,1709	-	-

Table 8. Wind turbine WT1 Pearson's correlation coefficient against signal *TempAceiteMultipMean*.
Source: Self-made

As can be seen in the table above, there are many variables that stand out for both positive values close to 1 and negative values close to -1. These are the ones that interest us the most since they are variables that contain valuable information that can help the model to make the prediction it should make.

Not all correlations close to 1 or -1 can be useful to the study. Previously it was commented that knowledge in the domain of acting could be important and here it comes into play again. It is observed that the variable TempCojLAMean has a very high correlation, in the same way that TempCojLOAMean has it. Nevertheless, it is not beneficial for the model to choose any of them. They are variables that while monitoring the temperature of the bearings in different areas give very similar readings since they are located very close. Selecting them would be like repeating the information and could cause multicollinearity problems.

Multicollinearity is not always problematic, but it is important to keep it in mind. This happens when there are predictors correlated with other predictors in the model. It can cause an increase in the variance of the coefficients and make the model very sensitive to small changes. High multicollinearity can make the results unstable and difficult to interpret.

In this project, a fixed number of predictors has not been chosen, on the contrary, many of them have been experimented with to find the best model. Three main groups of predictors have been made:

Group A	**Group B**	**Group C**
VelRotorMean	TempAmbMean	PotMean
VelGenMean	VelVientoMean	TempAmbMean
TempRodamMultipMean	IndTurbulMean	TempCojLAMean
TempRadSupMean	TempAceiteMultipMean	TempCojLOAMean
TempRadInfMean		VelGenMean
NivVibraMean		VelRotorMean
PotMean		VelVientoMean
TempAmbMean		TempAceiteMultipMean
VelVientoMean		
IndTurbulMean		
TempAceiteMultipMean		

Table 9. Wind turbine WT1 Pearson's correlation coefficient against signal *TempAceiteMultipMean*. Source: Self-made

These are the main groups created to make the first models. Group A is a group characterized by being formed by predictors that have a high correlation with the target variable TempAceiteMultipMean. This group has not only been made thinking about the Pearson correlation coefficient, it has also considered the expertise in the field of wind turbines.

Group B is the group of exogenous variables. This would be the ideal scenario since it involves variables outside the installation. If they are observed, one can realize that they are variables inherent to the environment: ambient temperature, wind speed and wind turbulence. If this model worked with good precision, it would not be necessary to have SCADA data; data could be taken from a nearby meteorological station, making the models simpler and making this approach very affordable.

Group C is a group focused on temperature variables that maintain a high correlation. As previously mentioned, this group can present multicollinearity problems.

In addition to these groups, to broaden the analysis, a system for choosing random predictors has been designed. Given a maximum number of predictors N, N predictors are chosen randomly, and a model is made with them. This opens the door to a great deal of testing aimed at finding clusters of very inconspicuous predictors that can train a good model.

Finally, it is important to have knowledge of the Bias-Variance trade off to understand why the predictors need to be chosen instead of using all the available ones.

The Bias is the difference between the average prediction of our model and the real values that we are trying to predict. Models with a high bias tend to simplify the models a lot and tend to have high errors. The variance is the variability of the model's prediction, if a model has high variability it will have an excellent prediction for data that it has seen, but for data that it has not seen it will probably have a very high error.

As explained in the article (S. Singh, 2018) this can be explained mathematically as follows:

Since the bias is the difference between the prediction and the actual data, it can be expressed as the squared error:

$$Err(x) = E[(Y - f(x^2)]$$

So the error of a model can be expressed as:

$$Err(x) = bias^2 + variance + irreducible\ error$$

The irreducible error is the error that generates noise in our data and cannot be eliminated. These are the most common effects when the bias and / or variance is high in a model:

- Underfitting: Occurs when the model can not capture the underlying pattern of the data.
- Overfitting: Occurs when the model captures the underlying pattern of the data in a way that only responds well to training data but not to new data.

The following diagram summarizes very well the main aspects of the above effects of high bias or variance:

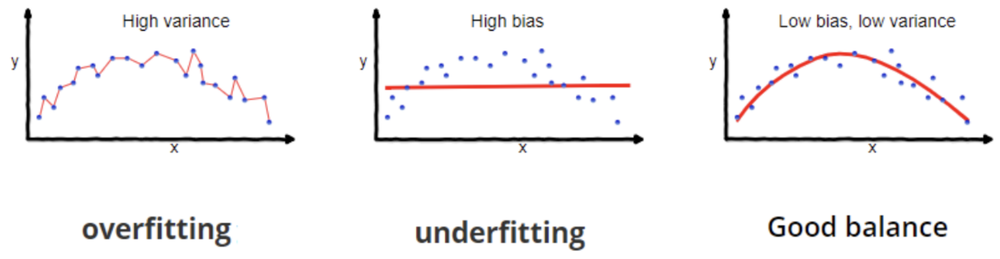


Figure 18. Bias vs variance trade-off. Source: TheMachineLearners

As can be seen in the figure above, the proper balance is found when both are as low as possible. It is straightforward to fall into overfitting when you have many predictors, and that is why not all of them are selected. A model should work well for both data that you have seen (the training ones) and data that you have not seen (testing or validation ones).

Below is a detection and elimination of outliers, but what are these?

There are many definitions that allow us to draw an idea of what we mean when we talk about outliers:

- "An outlier is an observation that deviates so much from the rest of the observations as to create the suspicion that it was created by a different generating mechanism" (Hawkins, 1980).
- "An outlier is an observation (or set of observations) that are inconsistent with the rest of the data" (Barnett and Lewis, 1996)
- "An outlier is an observation that is outside the general pattern of a distribution" (Moore and McCabe, 1999)
- "Outliers can be, depending on the circumstance, unwanted errors that can negatively affect the result or valuable nuggets of unexpected information" (Rousseeuw and Hubert, 2011).

Eliminating a data from a sample because it has been considered atypical can lead to the loss of relevant information due to a singularity of the generating mechanism, and in turn including an atypical data in a sample can confuse the results. Both cases alter subsequent analyzes and can lead to incorrect conclusions if the wrong decision is made. Therefore, the importance lies in properly identifying which data are atypical and which are not (Da Vila, 2020).

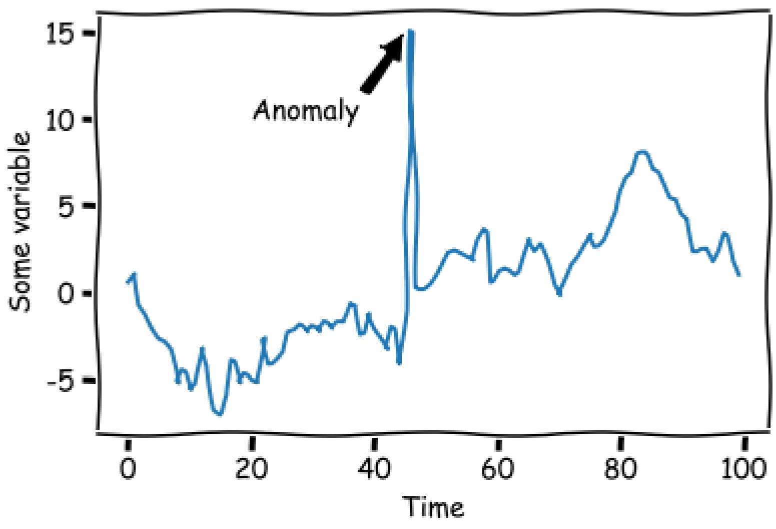


Figure 19. Anomaly detection example. Source: Ecomodeller

The term nuggets of unexpected information refers to data that is anomalous but does not necessarily need to be removed. In this case, we would speak of an outlier that will indicate a failure or a failure to occur.

A filter is applied that prevents the signals from containing values that are not physically coherent, such as ambient temperatures above 100 degrees Celsius or negative wind speeds.

The models need to be trained, for this the so-called training data is needed. That is why the process that follows is responsible for separating the data into training data and test data.

Training data is the data that we use to train a model. The quality of our machine learning model is going to be directly proportional to the quality of the data. As a result, clean up tasks consume a significant percentage of data scientists' time.

The test data is the data that is reserved to check if the model that has been generated from the training data works properly. That is, whether the responses predicted by the model for a completely new case are correct or not.

Normally, the separation of data in training and testing is done with a 70% - 30% rule. The idea behind this distribution is that many data is needed to train, but keeping in mind that a correct test to analyze the model is important. For this project, however, the rule changes slightly. Much more data is still used for training than for testing, but the training data is the oldest data in the dataset. This is done this way to ensure that the training data is as healthy as possible. For test data, however, all the remaining data is going to be used

The last preprocessing step that has been carried out is standardization. Standardization is the process by which data is converted from its original unit of measure to a common format with the rest of the variables. This process allows data of a different nature to be used with less trouble by learning algorithms. For example, this model can be found with temperature variables in degrees Celsius, speeds in meters per second, or vibrations in meters per squared second. Mixing units can be confusing and cause predictors to have larger or smaller weights just by presenting different scales. Standardization helps eliminate that effect. The formula to standardize is as follows:

$$z_i = \frac{x_i - \overline{X}}{S_X}$$

Given a sample x_1, x_2,. . . , x_n. Where S is the standard deviation and X is the mean. It must be done for each predictor and the target variable. The result is a new representation of the data without units. It is important to "save" the standardization parameters (mean and standard deviation) to later be able to undo the standardization and obtain the results in their original unit.

4.3 Algorithm and Model

The algorithm used in this project is called Relevance Vector Machine (RVM). It was developed by M.E. Tipping, and it has been since then available as an open-source and free software. RVM is very similar to Support Vector Machine (SVM).

The RVM algorithm is only implemented by its creator in MATLAB. This project is being carried out entirely in Python, so it has been an entry challenge. To overcome this challenge, a Python file has been developed whose sole purpose has been to be able to execute MATLAB files in Python through the MATLAB Engine API. For this, the necessary packages have been installed and a suitable connector developed. Instructions for its correct operation have been described in the project's readme file.

A brief overview of the sparse Bayesian model explained by (Tipping, 2001) is as follows: "For example, consider a predictive modelling task where there is a set of 'input' data samples $X = \{x_n\}$ with corresponding desired 'target' values $t = \{t_n\}$, $n = 1 . . . N$. The objective is to find an underlying functional model y(x) that "predicts" t well given x, and is not compromised by noisy, real-world, data.

One possible predictor y(x) would be a "generalized linear" one:

$$y(x) = \sum_{m=1}^{M} w_m \varphi_m(x)$$

where $w = (w_1, w_2, . . . , w_M)$ is the vector of adjustable parameters. This model is therefore said to be "linear in the parameters", which conveys a number of analytic advantages. At the same time, by choosing the "basis functions" $\phi_m(x)$ to be nonlinear, y(x;w) may be nonlinear too. If M is large, then this type of model is potentially very flexible, and providing the computational and statistical complexity of the model is appropriately managed, it can be very effectively applied to a wide range of problems.

The sparse Bayesian methodology has been designed to be a principled and practically effective mechanism for managing this computational and statistical complexity. The key to the approach is the definition of a hyper-parameterised prior over the model parameters of the form:

$$p(w|\alpha) \approx \prod_{m=1}^{M} \alpha_m^{\frac{1}{2}} exp\,(-\frac{\alpha_m}{2} w_m^2)$$

Analysis shows that this type of prior ultimately favours models that both fit the data perfectly and, in particular, are sparse (Tipping, 2001). That is, the prior locates most of its probability mass at $w_m = 0$.

Given the model and its prior, the algorithm computes the posterior distribution over the hyperparameters α_m given the data, and returns their most-probable values by maximising the marginal likelihood function. This leads to a posterior distribution over the parameters w_m where many are (mathematically) set to zero. As a result, the predictive models derived by the algorithm will, in most cases, be sparse. In practical terms, then, the generalised linear model predictor y(x) will typically comprise very few non-zero wm and will thereby incorporate a compact set of basis functions only."

Running the inference algorithm requires that an appropriate model basis matrix (or kernel function) for the problem at hand is set up in advance.
To construct a non-linear estimator, we must transform the input data non-linearly. The nonlinear transformation implies a correspondence towards a space of greater dimension, possibly infinite.

For example, given a polynomial transformation to a higher dimensional space, there is a scalar product in that space that can be expressed as a function of the input data. This dot product is called the Kernel, and the space with the highest dimensionality is called the Hilbert space. This transformation increases the possibility of linear separability.

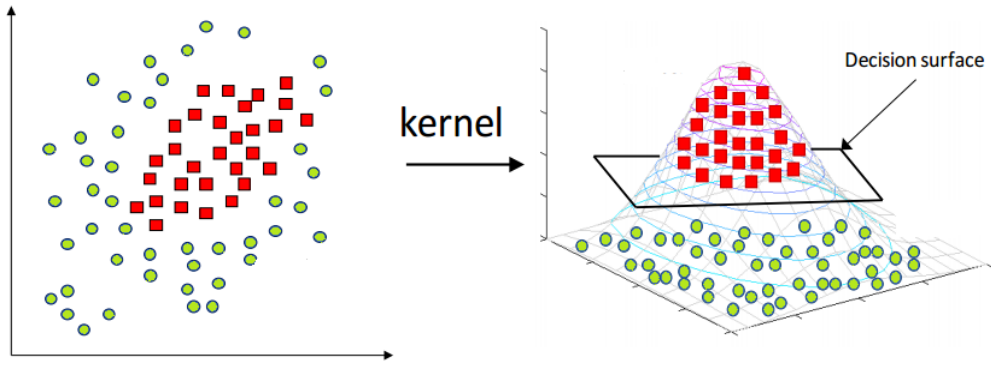


Figure 20. Kernel application of a SVM algorithm. Source: Grace Z.

In the image above, we can see a clear example of separability by increasing the size of the space. In the graph on the left, you can see two kinds of data. At first glance and in two dimensions, the separability of these data through a linear function is not possible. Instead, if we take the space to a larger dimension (right figure), we can use a linear function (in this case a plane) to effectively separate the data.

To do this, one might think that it is necessary to explicitly increase the dimension of all the points using a function. Increasing dimensions is computationally expensive and slow.

Now, to build the mapping function or the basis matrix, it is not necessary to actually build it. It is enough to be able to calculate the scalar products, and this is achieved through the kernel functions. This trick is called the "kernel trick" and allows you to bypass the explicit mapping that is necessary for linear learning algorithms to use nonlinear functions. This causes the computational load to be much less than what would be needed if the mapper had to be made (Y. Vidal, 2020).

The fact of needing only the scalar products allows us to reproduce any linear algorithm in a Hilbert space. So there is a non-linear version of any data-based linear algorithm. The basis functions or kernel functions used in this project are:

- Linear: The design matrix is effectively the data matrix:

$$\varphi_m(x_n) = x_{nm}$$

- Polynomial kernel:

$$\varphi_m(x_n) = (\gamma\,(xnm) + r)^d$$

- Gaussian kernel:

$$\varphi_m(x_n) = exp(-\gamma||x_n - x_m||^2)$$

In this project, one can choose any of the 3 kernels previously exposed to train the model.

Although the kernel trick helps the model to be faster, it does not mean that the process itself is still a slow process. After all, when the scalar product of the training points is calculated, a matrix of 100 observations by 5 predictors becomes a 100 x 100 matrix. The matrices that our model generates become 52,560 x 52,560. This matrix is very computationally expensive.

Hyperparameters are adjustable parameters that allow you to control the training process of a model. For example, with neural networks, you can decide the number of hidden layers and the number of nodes in each layer. These hyperparameters can be observed in the kernel functions described above. While the linear kernel does not have hyperparameters, they do have both the polynomial and the Gaussian. The polynomial has two hyperparameters γ and d. The Gaussian has a hyperparameter γ.

The tuning of hyperparameters is critical, since their correct optimization allows the algorithm to work effectively. It is also one of the most computationally expensive parts. To explain this, we will focus on the Gaussian kernel and what methodology has been followed to optimize the hyperparameters

A training with the data of 1 year of operation of a turbine with an RVM algorithm and Gaussian kernel takes about 2.5 hours in a 6-core Intel Core i7 processor at 2.6 GHz. Therefore, it means that if we wanted to find the more adequate hyperparameter, each iteration would take us approximately 2.5 hours. That is a long time and many iterations. To avoid this excessive amount of computing hours, two strategies have been used:

- Parallel Processing or multiprocessing
- A smarter way to find the optimal hyperparameter rather than brute computational force

For multiprocessing, what has been done is to take advantage of the fact that Python allows the developer to use the capacity of multiple processors on a machine. Therefore, in this project, 6 simultaneous calculations are performed for each hyperparameter tuning cycle, thus considerably reducing the execution time.

A simple method has been designed to avoid having to iterate through all the possible values of the optimal hyperparameter one by one. What has been done is to perform the search by intervals:

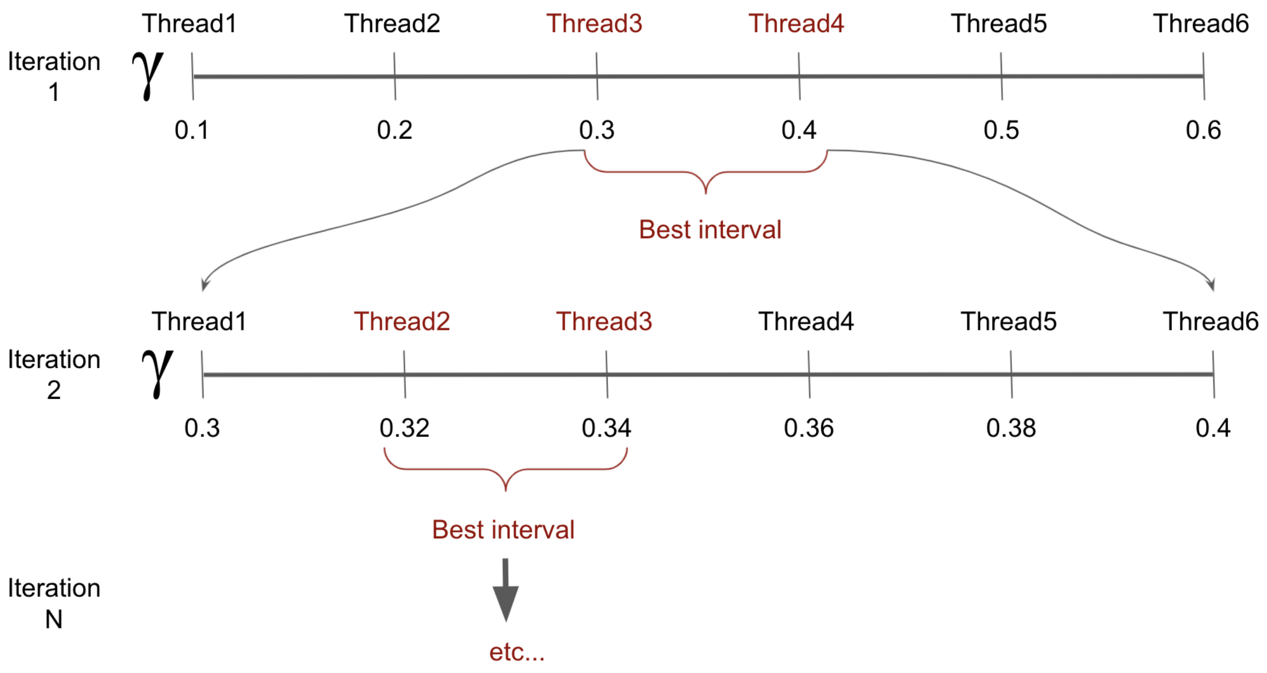


Figure 21. Schematic of the hyperparameter tuning iterative process. Source: Self-made

To avoid having to search for the optimal hyperparameter by brute force, an algorithm has been made that takes advantage of multiprocessing and makes a more intelligent selection of which interval is more attractive to focus efforts. This algorithm assumes that the optimal value is at an absolute point, and empirical facts have shown that it works.

The detailed explanation is as follows: each core is responsible for performing a calculation with an assigned hyperparameter. When the calculation is performed, a score is obtained and when all the processes finish, the interval where the two best results have been is chosen, as indicated in the previous figure. Once the best interval is known, the process is repeated up to a number N of specified iterations. When the last search iteration of the optimal hyperparameter is reached, the value that has given the best result is chosen within the last interval chosen. To make the decision of which interval is better, a scoring system is used, which will be discussed shortly later.

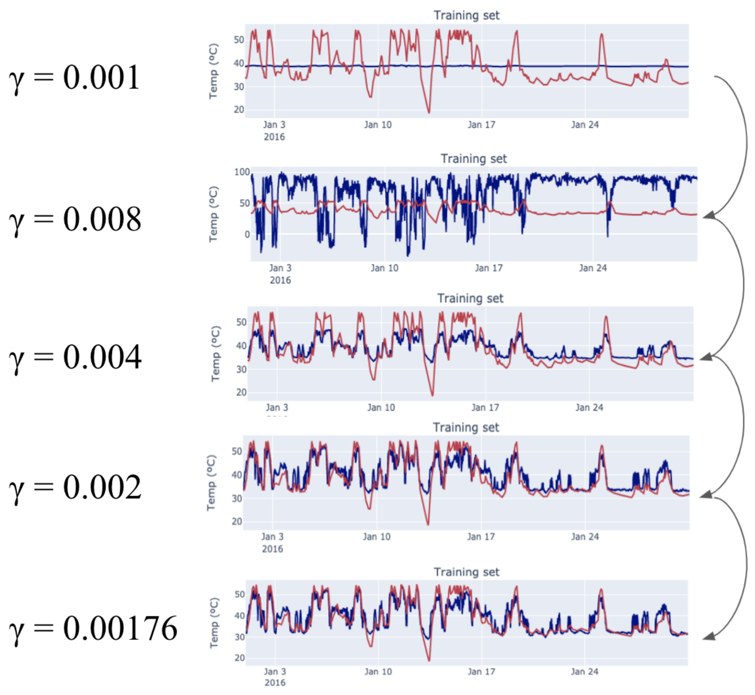


Figure 22. Example of the hyperparameter tuning iterative process. Source: Self-made

In the graph above, you can see how the different iterations in parallel try to find the best hyperparameter to culminate in the model that best fits the training parameters. The chosen parameter, in the example γ= 0.00176, is chosen by a combination of metrics that encapsulate the goodness of fit of the different models and their hyperparameters.

Once the best hyperparameter has been chosen, it is time to predict the gearbox oil temperature. This prediction will be an approximation of the temperature value that the gearbox oil is expected to have given some environment variables, the predictors. This prediction, when trained with healthy data, will be typical of a healthy turbine without failures, so any estimate well above or below the real data observations can indicate a failure in progress.

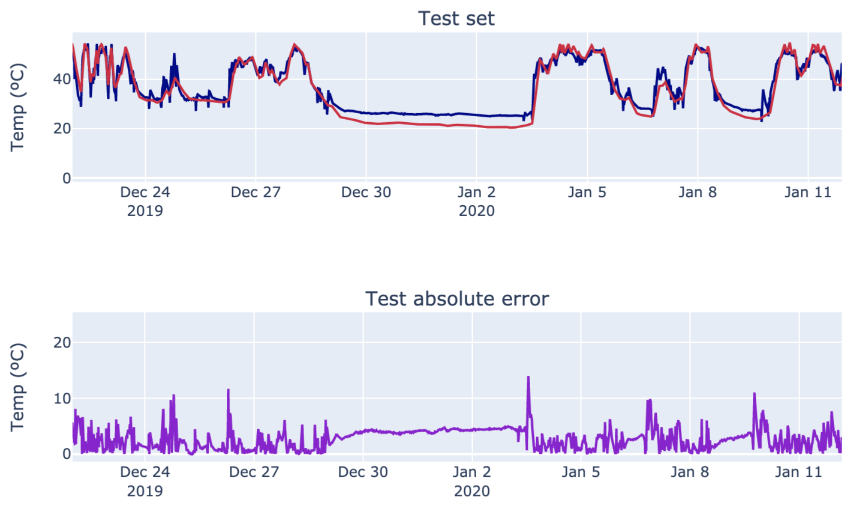


Figure 23. Test set and test absolute error plots example. Source: Self-made

In the graph above, you can see how over time, the red line (real or observed value) moves away from the blue line (predicted value). It could be an indication that something is not working properly and a potential clue that a major failure is about to happen in the future.

It is important to note that the model is intended to capture the behaviour of a healthy wind turbine. If the model is capable of capturing it, this model could be called a “normality model”. The normality model is a model which seeks to replicate the behaviour of a healthy system. Any deviation between the normality model and the real system could mean that something not healthy is happening. Normality models help to overcome the challenge of untagged data, which is this case.

At this point, the prediction has already been made. The model tried to emulate the healthy behavior of the gearbox oil temperature. The test data allows you to compare the prediction results with the actual results. Thanks to the fact that they can be compared, we can know if the models perform better or worse. One way to do this analysis is through the residuals, which is another way of calling the subtraction between the predicted value and the observed or real value. Residuals are important to validate the model and that is why they are included in the methodology.

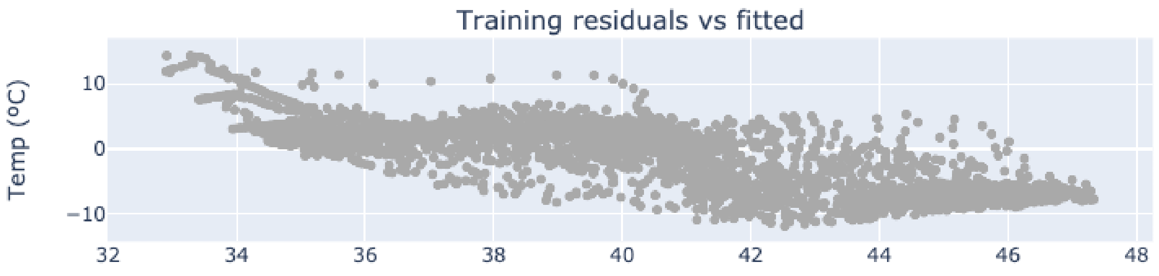


Figure 24. Training residuals vs fitted scatter plot. Source: Self-made

In the upper graph you can see the scatter plot of the residuals of a training set. The observed values appear on the X axis, while the error between the observed and predicted values appears on the Y axis for each of the observed values. The above plot sheds quite a bit of light on how a model performs. It can be seen how for low observed values (from 32ºC to 36ºC) the model tends to overestimate the resultin fact, it presents a certain proportional tendency; the smaller the observed value, the greater the error. It seems that the reverse is the case with the upper band, it tends to be underestimated. A scatter plot that indicates a good behavior of the model would be seen as points distributed on the plot randomly and with the same deviation in all points. If that were the case, the model would have successfully captured the relevant information from the training data and the estimation error would only be the irreducible error, caused by the signal noise.

To analyze them and take advantage of the information hidden in them, one should proceed to look at the following comment: “The bottom line is that randomness and unpredictability are crucial components of any regression model. If you don’t have those, your model is not valid.” (Minitab Blog, 2012)

There are two basic components to any valid regression model:

- Deterministic
- Stochastic

The deterministic component is one that can be explained by the predictors of the model. The expected value of the response is a function of a set of chosen variables. All the explanatory or predictive part of the model should be in the deterministic component.

The stochastic component is the unpredictable and random component. The error is the difference between the observed value and the predicted value. The differences between the observed and predicted value must be unpredictable. This component should not contain any kind of explanatory or predictive information.

In other words: "The idea is that the deterministic portion of your model is so good at explaining (or predicting) the response that only the inherent randomness of any real-world phenomenon remains leftover for the error portion. If you observe explanatory or predictive power in the error, you know that your predictors are missing some predictive information. Residual plots help you check this!". (Minitab Blog, 2012)

For all the above, a residual plot should look like a random scatter plot since the error of the predictions should not be able to be explained by anything. If the plot of residuals shows some kind of predictable behavior, for example a tendency to be wrong in high values compared to low values, it means that the model can be improved and that the deterministic component of the model is not capturing some predictive information. This is leaked in the residual plot. The general idea of how the residual plot should look is very well summarised here: "The residuals should not be either systematically high or low. So, the residuals should be centered on zero throughout the range of fitted values. In other words, the model is correct on average for all fitted values. Random errors are assumed to produce residuals that are normally distributed. Therefore, the residuals should fall in a symmetrical pattern and have a constant spread throughout the range.". (Minitab Blog, 2012)

4.4 Anomaly detection

Once the best model has been chosen and the test values have been predicted, we proceed to determine if there is potential for future failure in any part of the time series. To do this, the residual is first calculated:

$$Residual = |predicted\ value - observed\ value|$$

The mean and standard deviation are calculated below:

$$\mu = mean(Residual)$$

$$\sigma = std(Residual)$$

And finally, a threshold is calculated which, if exceeded, will be counted as signs of a potential anomaly:

$$Threshold = \mu + Anomaly\ factor * \sigma$$

The anomaly factor that appears in the above equation is a constant that can be configured and that regulates how sensitive the threshold is to anomalies. A high anomaly factor will cause the threshold to be also high, and therefore it will be much more difficult to consider a signal as anomalous. In contrast, if the anomaly factor is low, it will be easier for a signal to be flagged as an anomaly:

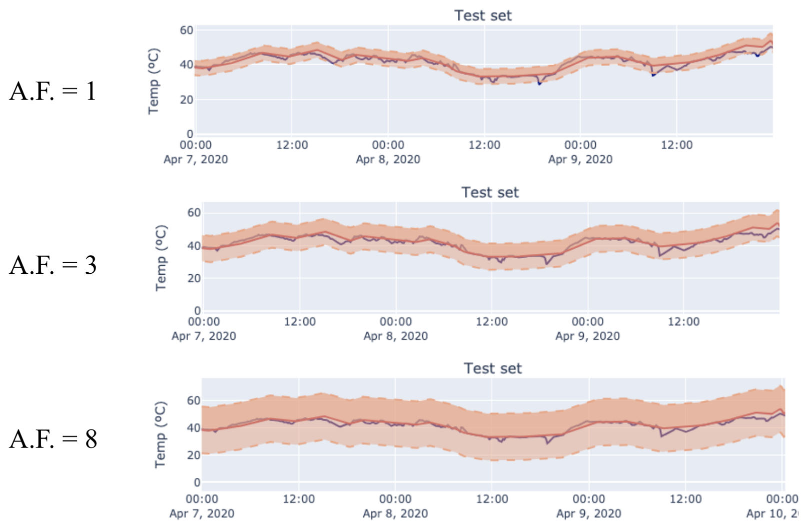


Figure 25. Anomaly factor effect on threshold. Source: Self-made

The anomaly factor is a value that the operator of the wind farm has to decide. If the operator desires a conservative approach, a small anomaly factor will be chosen to detect every minimal deviation. If the operator does not desire such sensitivity of the model, a big anomaly factor will be chosen. In this case, an anomaly factor of 3 is chosen to have a balanced model.

This methodology will allow to have a flag for each of the residuals that will indicate if the threshold has been exceeded, and therefore it is an anomaly. Note that when dealing with the absolute of the residuals, the threshold does not have to be a band with lower and upper limits. As there are no negative residuals, only an upper (positive) limit is necessary.

Once the flag that reveals whether a residual is classified as anomalous or not is obtained, an indicator is calculated that allows knowing with what probability it is, in fact, an anomaly. Not all abnormalities can be treated as true. Some are probably due to reading errors or interference with the actual observed value. An anomaly will be considered genuine if it is repeated enough for a short time. To achieve the above, a system of accumulation of possible anomalies is used. It works in the following way:

Imagine a temporary space of 20 residuals. In this temporal space, there are certain residuals that have exceeded the established threshold, so they are assigned the flag that allows us to know if it is considered an anomaly or not. T refers to the index of the timestamp and flag indicates whether it is an anomaly or not in red:

T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Flag		🟥						🟥					🟥	🟥	🟥	🟥			🟥	🟥

Table 9. Wind turbine WT1 Pearson's correlation coefficient against signal *TempAceiteMultipMean*. Source: Self-made

Next, a rolling sum is made, which deals with making a cumulative sum but with a window that moves one index at a time. In this example, a rolling sum (RS in the table) of 5 windows has been configured:

T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Flag		🟥						🟥					🟥	🟥	🟥	🟥			🟥	🟥
RS	0	1	1	1	1	0	0	1	0	0	0	0	1	2	3	4	4	3	3	3

Table 10. Wind turbine WT1 Pearson's correlation coefficient against signal *TempAceiteMultipMean*. Source: Self-made

As can be seen in the previous table, the RS row makes a sum by windows of 5 indexes. The maximum RS score is between indices 15 and 17 indicating that it is on those dates that there may be the greatest risk of a genuine anomaly. This anomaly potential is what allows visualising for all the turbines if the method behaves as desired. In the real model, this window is 2,016 indices, the equivalent of two weeks.

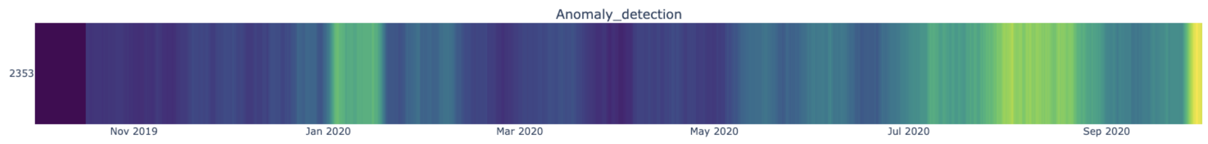


Figure 26. Wind turbine WT1 anomaly detection heatmap. Source: Self-made

In the previous figure you can see several points where the anomaly detection engine detects certain points in time that may be indicative that something future may happen. In this specific case, it can be seen how at the end of January 2020 there seems to be a recurrence in values that exceed the calculated threshold. This behavior intensifies in late summer 2020 shortly before the gearbox failure at the end of the time series.

4.5 Model score

To be able to judge whether a model is doing the job effectively and as desired, several metrics are collected for each model execution that help to analyse its behavior. These are:

- Mean Absolute Error (MAE): Considering two series of data, some calculated and others observed, related to the same phenomenon, the mean absolute error serves to quantify the precision between the predicted and observed values. In this case, the compared values are the prediction of the gearbox oil temperature and the actual value observed. That is why the MAE's unit of measurement in this case will be in degrees Celsius.

- Max error (ME): As its name indicates, this metric collects the maximum error between the prediction and the observed value. Again, this unit will be measured in degrees Celsius.
- Explained variance (EV): The explained variance is used to measure the discrepancy between a model and the actual data. High percentages of explained variance may indicate a high degree of association, therefore a model capable of making better predictions (Rosenthal and Rosenthal, 2011)
- Root Mean Square (RMS): RMS is a metric similar to MAE. The difference is that this metric penalises large errors more than small ones, thus having a metric that can help to know which models produce larger errors regarding the MAE.

These metrics end up forming the score module which allows both to analyze results and shed light on their effectiveness and to help the process of tuning hyperparameters. The hyperparameter tuning process takes these metrics into account and selects the best intervals and the best final iteration based on it. These metrics are collected for both training data and test data. For each execution you can find a table like this:

Score metrics	Training set	Test set
Explained Variance	0.80	0.75
Max error (ºC)	22.4	21.97
MAE (ºC)	2.62	3.21
RMS (ºC)	3.54	4.05

Table 11. Example of performance metric table of a model. Source: Self-made

It can be seen in the table above that normally the training set has better global metrics compared to the test set. This is because the models tend to do very well with data that they have seen and that they have trained with (training set) and they do not do so well with data that they have not seen (test set). The score metrics of the model are saved along with the hyperparameters' iteration search and each of the results. In the following table one can see the results for each iteration of gamma which sought to look for the best hyperparameter. Here it can be observed that **γ** = 0.0176 has better results in a combination of RMS, MAE and explained variance.

Gamma	RMS test	MAE test	Max error test	Explained variance
0.0016	4.064089	3.220664	0.775362	0.219734
0.00168	4.059973	3.215382	0.775511	0.219769
0.00176	4.056076	3.210273	0.775635	0.219759
0.00184	4.096731	3.211918	0.789090	0.229129
0.00192	4.092438	3.205717	0.789073	0.229179

Table 12. Example of hyperparameter tuning details table. Source: Self-made

As well as the execution times, important to consider:

Module	Time (s)
Data loading	5.63
Data preprocessing	0.44
Training	153.21
Testing	4.54
Total case runtime	163.83
Total pipeline runtime	512.18

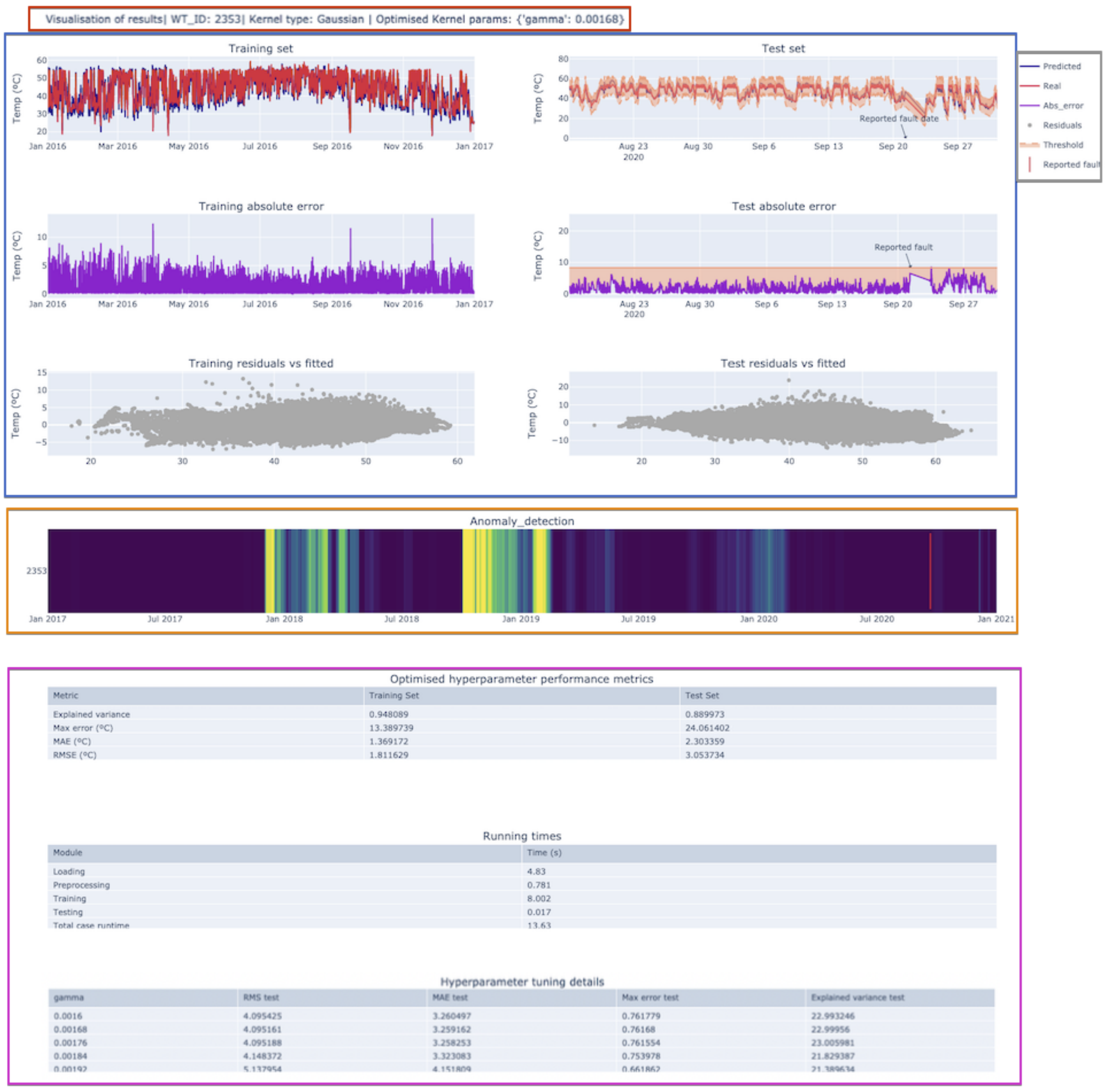
Table 13. Runtimes for an execution of a 45 days training set. Source: Self-made

As can be seen in the table above, the individual process that takes the longest is training. This is normal and expected, since it is in this process that the model is actually trained. Part of that training consists of creating the kernel function, which as discussed before is one of the most expensive processes for the CPU. This specific case is an execution with a training period of 1 month and a half. The target executions of this project usually have a minimum training period of 1 year. This can lead to runtimes bigger than 2 hours in the best cases. Sometimes the process has out of memory errors and can crash an execution in progress. The allocated available RAM is 4 MB in this project.

4.6 Visualisation

All these executions and processes end in a visualization process. To provide the user with an interface capable of giving a summary of the most important conclusions, a visualization has been developed such that:

- Detailed information about the analysed turbine, the kernel used, and the hyperparameters finally optimized can be read in the red box.
- In gray is the legend of the entire visualization.
- In the blue box, you can see various graphs that show the training and test data. In them, it can be seen the data in its temporal evolution, the errors and the residuals, as well as the different thresholds.
- The orange box shows a representation of the anomaly detection algorithm and its rolling window system.
- Finally, in magenta it can be observed more details of the execution as well as the evolution of the iterations, the execution times and the score metrics of the final iteration chosen.



Visualisation of results| WT_ID: 2353| Kernel type: Gaussian | Optimised Kernel params: {'gamma': 0.00168}

Optimised hyperparameter performance metrics

Metric	Training Set	Test Set
Explained variance	0.948089	0.889973
Max error (ºC)	13.389739	24.061402
MAE (ºC)	1.369172	2.303359
RMSE (ºC)	1.811629	3.053734

Running times

Module	Time (s)
Loading	4.83
Preprocessing	0.781
Training	8.002
Testing	0.017
Total case runtime	13.63

Hyperparameter tuning details

gamma	RMS test	MAE test	Max error test	Explained variance test
0.0016	4.095425	3.260497	0.761779	22.993246
0.00168	4.095161	3.259162	0.76168	22.99956
0.00176	4.095188	3.258253	0.761554	23.005981
0.00184	4.148372	3.323083	0.753978	21.829387
0.00192	5.137954	4.151809	0.661862	21.389634

Table 27. Final visualisation for a wind turbine prognosis model. Source: Self-made

This screen is shown whenever the process for a turbine of a wind farm has finished. In the previous image, you can see the different graphs and tables that were previously mentioned. Thanks to this visualisation one can get an idea of how the model works for each individual wind turbine.

To know whether the execution is good or not, it is interesting to have some context about the temporal location of the known failure. This can be seen in the orange box plots. For example:

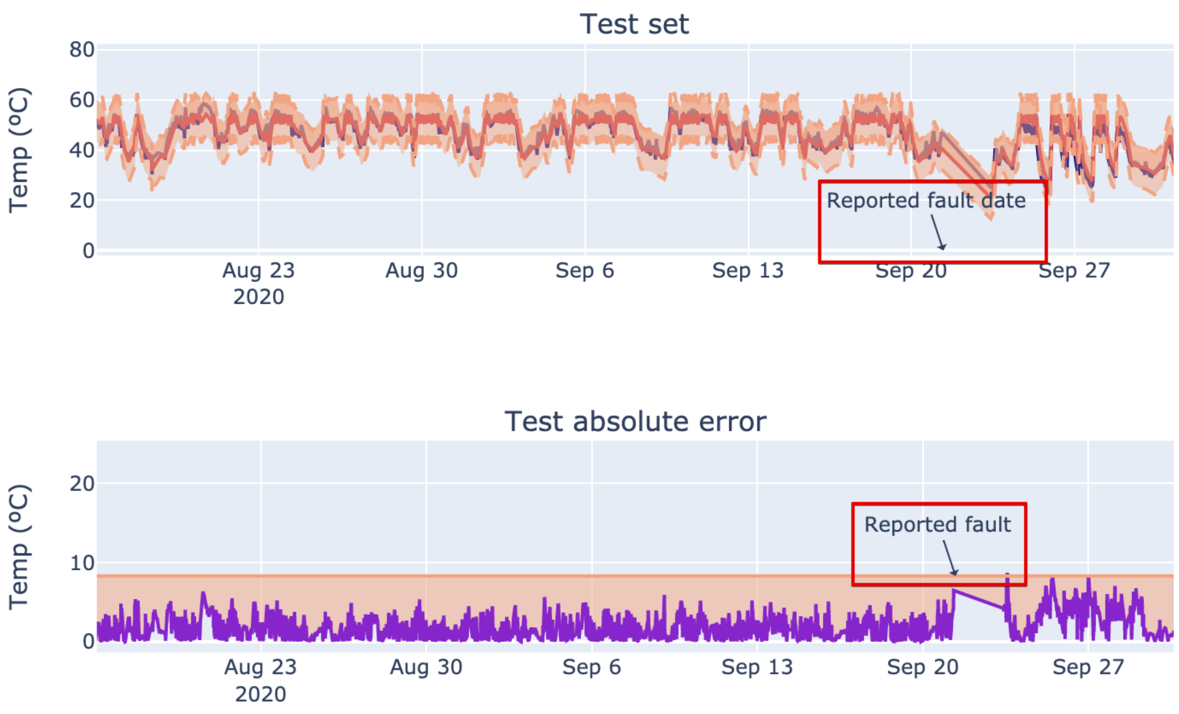


Figure 28. Anomaly date annotation for a turbine execution. Source: Self-made

As mentioned at the beginning of the project, this development aims to be able to analyse all the turbines in any wind farm. That is why this cycle is repeated for each of the turbines. That is why at the end of all cycles the following display appears:

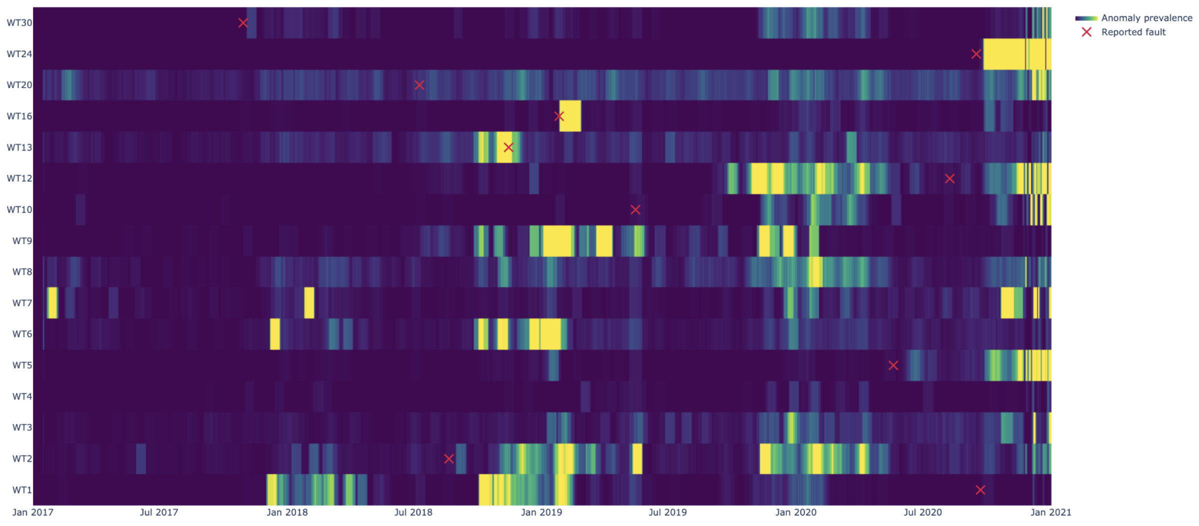


Figure 29. Anomaly heatmap plot for the different turbines. Source: Self-made

In the previous figure, you can see a heat map with all the executions carried out. Also in the upper part you can see the approach of predictors chosen for clarity purposes. Each run represents the anomaly detection heat map of a turbine. In red crosses it can be seen the reported faults for the turbines, they have this information available.

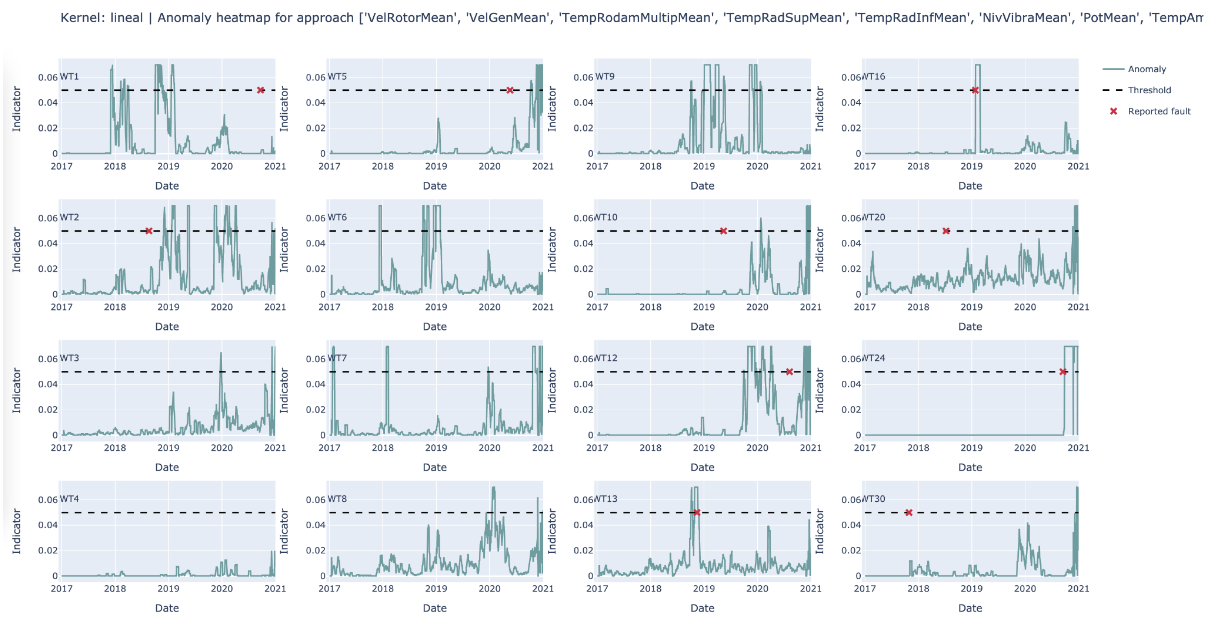


Figure 30. Anomaly line plot for the different turbines. Source: Self-made

The previous visualisation is also given in the last step of the entire pipeline to help the user understand the outputs. It can be seen for all the turbines tested its profile of anomaly detection, along an arbitrary threshold (dashed black line) where it is thought an anomaly should be seriously taken. In red, there is a cross that flags the reported faults for the turbines that had them. The threshold has to be chosen with the same criteria as the previously mentioned anomaly factor. If the operator of the wind farm wants a conservative approach and detect every minimal deviation, the threshold should be placed lower (e.g. at 0.02). If the operator just wants to know the very obvious deviation signals, a higher threshold should be placed (e.g. at 0.07).

4.7 Error handling, logs and tests

To establish good development practices, several concepts such as error handling, logs and tests have been added to the development.

Every part of the code is susceptible to error, and some sections more than others. That is why error handlers have been added to be able to quickly and efficiently debug the different errors resulting from development or from some improvement iterations. Each module has its own error management that reports both by command window and by logs the different errors that may occur and their origin. Most of these error handling fragments use Python's pattern try-except.

As mentioned in the previous paragraph, this development has a series of logs that are in charge of reporting the process of each execution as well as any faults that may occur. A log without failures usually has the following form:

```
31-08-2021 19:26:31: INFO - Wind Turbine ID: 1
31-08-2021 19:26:31: INFO - --- Starting pipeline ---
31-08-2021 19:26:31: INFO - Data loading module starting
31-08-2021 19:26:37: INFO - Data loading successful
31-08-2021 19:26:37: INFO - Data preprocessing module starting
31-08-2021 19:26:37: INFO - Cleaning data
31-08-2021 19:26:37: INFO - Starting to clean data
31-08-2021 19:26:37: INFO - Initial number of observations: 288970
31-08-2021 19:26:37: INFO - Number of observations after NaN filtering: 276085. 4.46% (12885) have been eliminated
31-08-2021 19:26:38: INFO - Number of observations after range filtering: 276085. 4.77% (221)  have been eliminated
31-08-2021 19:26:38: INFO - Preprocessing successful
31-08-2021 19:26:38: INFO - Splitting training and test data
31-08-2021 19:26:38: INFO - Data is being split by the following configured dates:
 Train: ('2016-01-01 00:00:00', '2016-12-31 23:59:00'),
 Test: ('2019-10-01 00:00:00', '2020-09-30 23:59:00')
31-08-2021 19:26:38: INFO - Scaling data
31-08-2021 19:26:38: INFO - Calculating Radial Basis Function kernel (gaussian kernel)
31-08-2021 19:26:51: INFO - Radial Basis Function kernel (gaussian kernel) calculated
31-08-2021 19:26:51: INFO - RVM function starting
31-08-2021 19:26:51: INFO - Starting MATLAB engine
31-08-2021 19:49:24: INFO - Launching RVM function through MATLAB engine
31-08-2021 20:38:49: INFO - Shutting down MATLAB engine
31-08-2021 20:39:50: INFO - RVM model attributes written.
31-08-2021 20:39:50: INFO - RVM function done
31-08-2021 20:39:50: INFO - Testing trained model
31-08-2021 20:39:50: INFO - Getting predictions
31-08-2021 20:39:54: INFO - Calculating Radial Basis Function kernel (gaussian kernel)
31-08-2021 20:41:19: INFO - Radial Basis Function kernel (gaussian kernel) calculated
31-08-2021 20:41:21: INFO - Getting score
31-08-2021 20:41:22: INFO - Getting individual visualisation
31-08-2021 20:41:30: INFO - Getting global visualisation
31-08-2021 20:41:42: INFO - --- Ending log ---
```

All logs are permanently saved in the execution directory in .txt or .log format

Finally, a series of tests developed with the pytest module has been carried out to ensure the proper functioning of the development in the event of any update. For example, the matlab connector module has its own test:

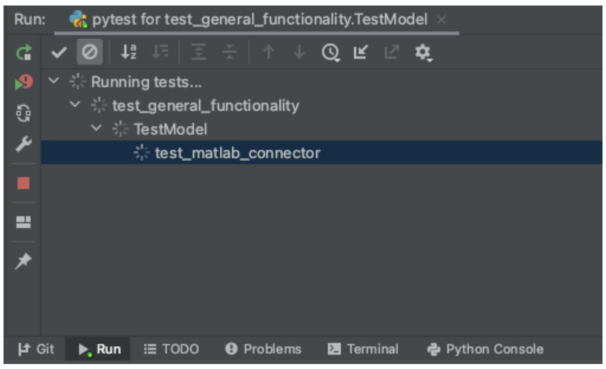


Figure 31. Matlab connector module test run as seen in PyCharm's IDE. Source: Self-made

This test is responsible to ensure that whenever there is a modification of the code, the connection with MATLAB can be tested. If the test fails, the developer will know that something is wrong and will fix it before uploading any new update.

5. Results and discussion

The results of this project will be presented in the following order:

- First a comparison of the different kernels implemented in the tool, both the results and the scores
- Second, a comparison of the different initial approaches that have been considered. Once again both the results and the scores
- Once the kernel and the most favorable approach have been chosen, we will analyze in detail the results it provides for several turbines.
- Finally, we will see a general image of the model for all the turbines that we know that have a fault reported.
- To take advantage of the potential of the tool, global results of different approaches will be given.

5.1 Kernel comparative

This section compares the results of the 2353 turbine for Gaussian and linear kernels. The polynomial kernel will be avoided in this section to not concur in redundancies.

The following execution has been made with the linear kernel:

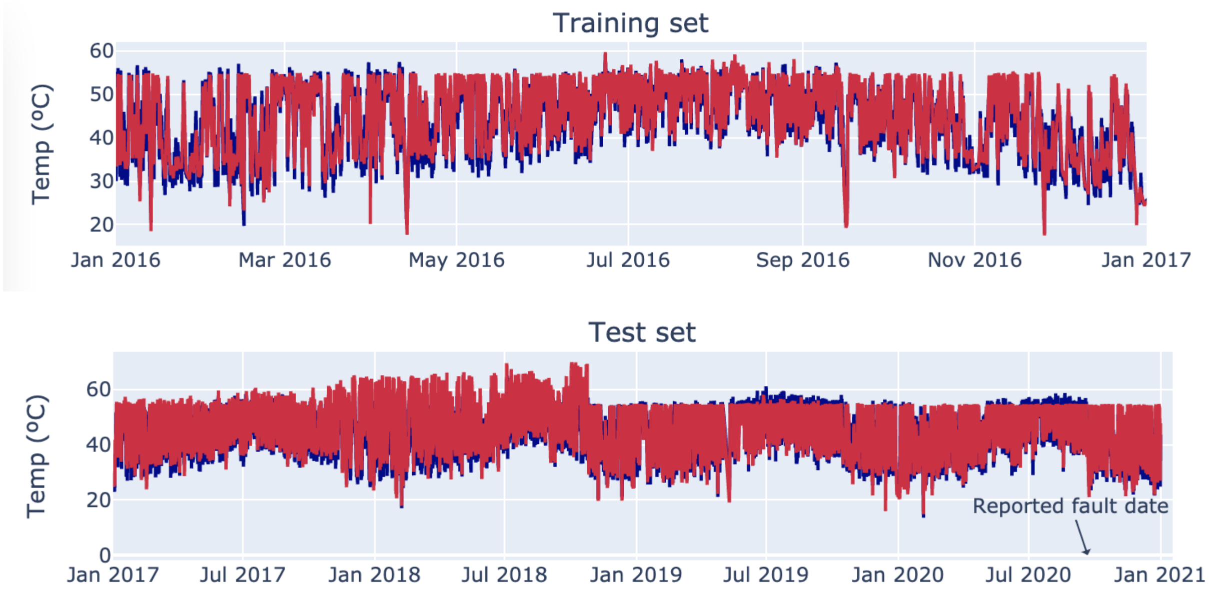


Figure 32. Kernel comparative WT1 lineal kernel training and test set plots. Source: Self-made

As can be seen in the previous figure, the actual training and test data (blue) are very similar to the predicted data (red). It seems that the general seasonal and daily trends are respected, as well as the areas most characterised by rises or falls. It can be seen in the test set that the gearbox failure date is around the end of September. On that same date, a sharp drop in temperature can be observed. This drop was probably caused by a turbine shutdown after the fault occurred.

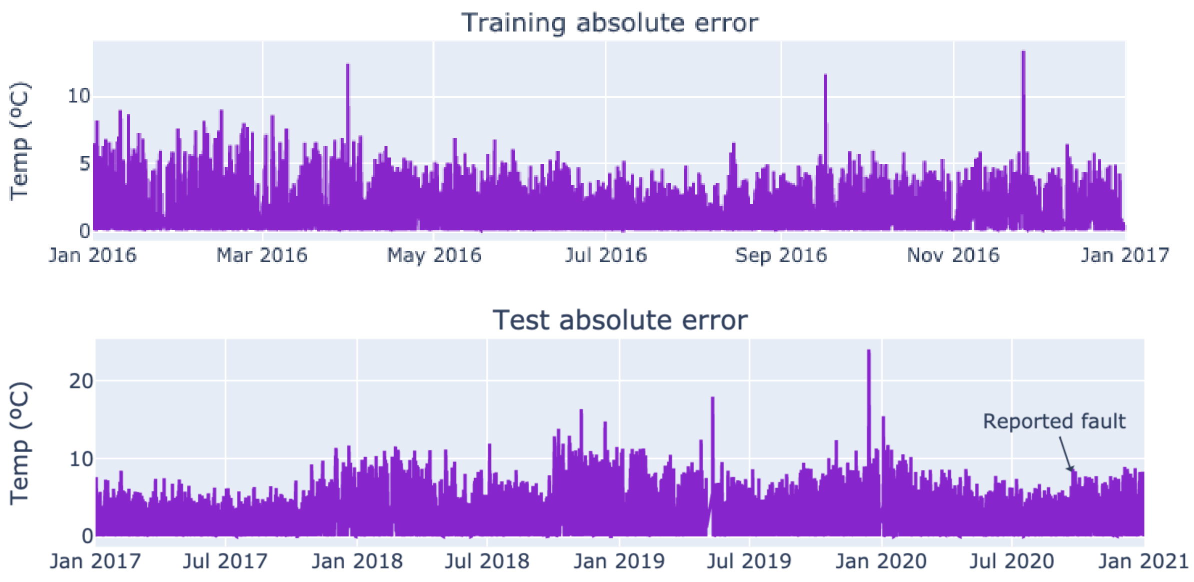


Figure 33. Kernel comparative WT1 lineal kernel training and test error plots. Source: Self-made

In the graph above, you can see the absolute error plots of the predicted curve and the observed curve. In purple is the absolute of the subtraction between the two mentioned curves. It can be seen that the differences do not usually exceed 10ºC in temperature except on specific occasions. No major behavior pattern or error appears to be observed before the anomaly noted in the test graph.

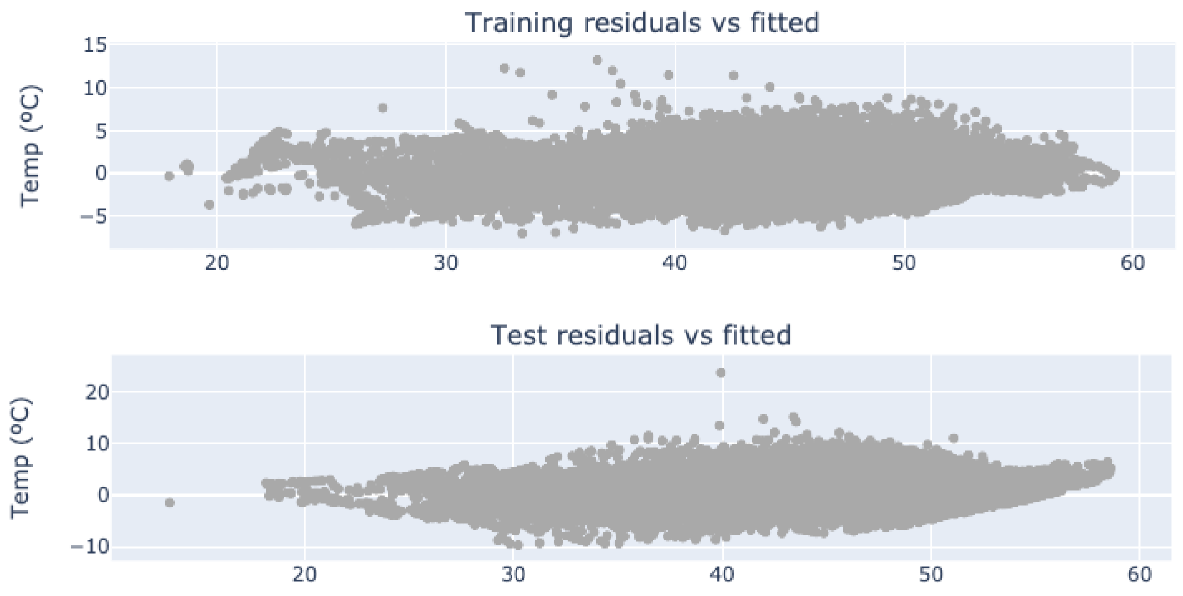


Figure 34. Kernel comparative WT1 lineal kernel training and test residuals vs fitted plots. Source: Self-made

In the previous graph, it can be seen the plots of training and test residuals vs fitted. These plots are very useful to determine the quality of the model. As explained in the methodology, this plot should be seen as a graph of random points, equally distributed and without any visible trend. This plot looks good in training, but not so good at testing. It can be observed how in the test the errors decrease in the extremes, while in more central values they grow both superiorly and inferiorly.

Score metric	Training set	Test set
Explained Variance	0.95	0.93
Max error (ºC)	13.38	24.06
MAE (ºC)	1.37	1.94
RMS (ºC)	1.81	2.53

Table 14. Kernel comparative WT1 lineal kernel score table. Source: Self-made

The results' table shows some excellent metrics considering that the model is linear, that it has trained with a full year of data and has predicted a year of data as well. We can see that the explained variance is very high, which means that the model is closely related to the data and has a good predictive ability. We can also observe excellent MAE and RMS values, less than 2ºC. Being wrong on average by 2ºC is good news, since it indicates that the model tends to predict on average below 2ºC of error.

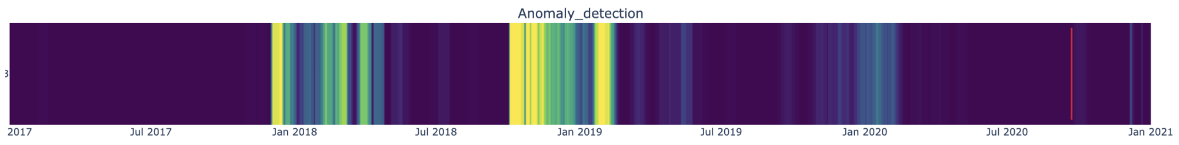


Figure 35. Kernel comparative WT1 lineal kernel anomaly detection plot. Source: Self-made

In the heat map above, a large presence of anomalous values can be observed at the end of 2019 and the beginning of 2020. After these dates, there are very few indications of error until the end of September 2020 where the registered fault is located.

Time is money and that is why the times of this execution are shown:

Process	Runtime (s)
Loading	5.96
Preprocessing	0.77
Training	10.09
Testing	0.01
Case	16.82
Pipeline	17.09

Table 15. Kernel comparative WT1 lineal kernel runtime table. Source: Self-made

It can be seen in the table above that the execution times are very fast with a total of 17 seconds invested for the 2353 turbine execution.

The following execution this section will focus on is the Gaussian kernel (or Radial Basis Function).

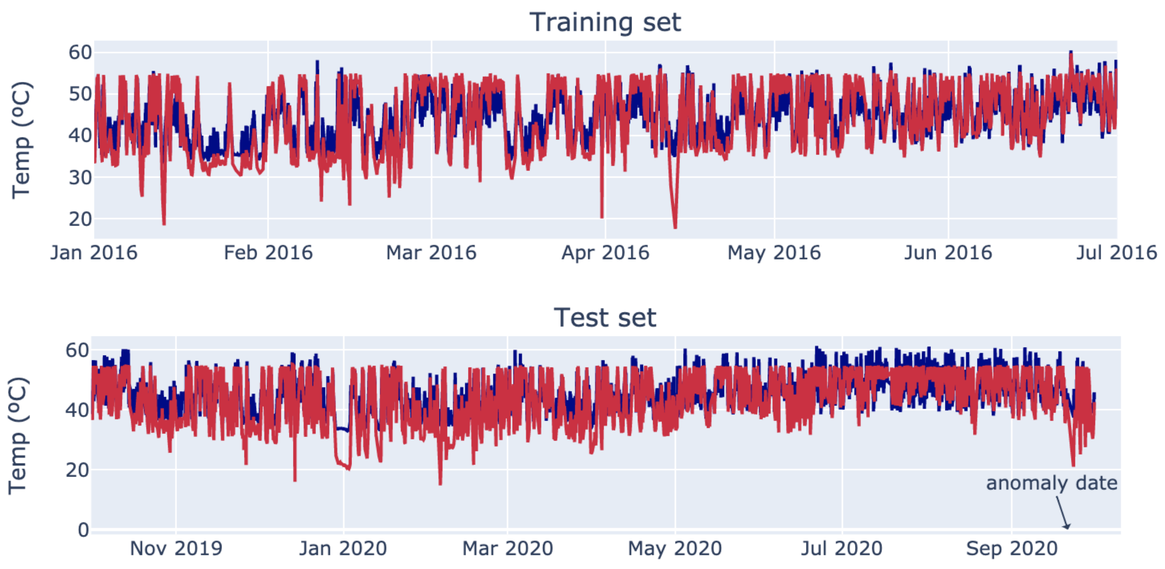


Figure 36. Kernel comparative WT1 Gaussian kernel training and test set plots. Source: Self-made

As in the previous graph, it can be seen how, in general, the observed curve (in blue) quite coincides with the predicted curve (in red). Even so, a certain vertical offset between the two can also be observed in the test set.

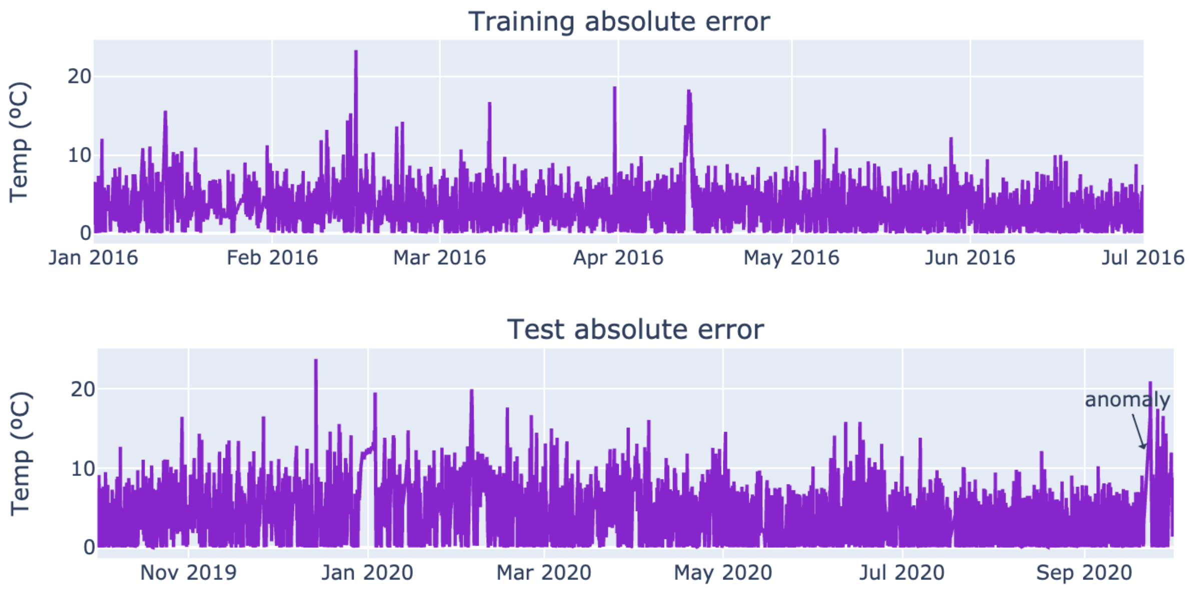


Figure 37. Kernel comparative WT1 Gaussian kernel training and test error plots. Source: Self-made

Observing the previous figure, which allows visualising the absolute error values between the real and predicted values, a larger general error is seen, especially in the test. In this run, the error values relatively easily exceed 10ºC and there does not seem to be a strong indication of error before the fault. The fault is the same as the previous comparison, so it continues to be at the end of September 2020.

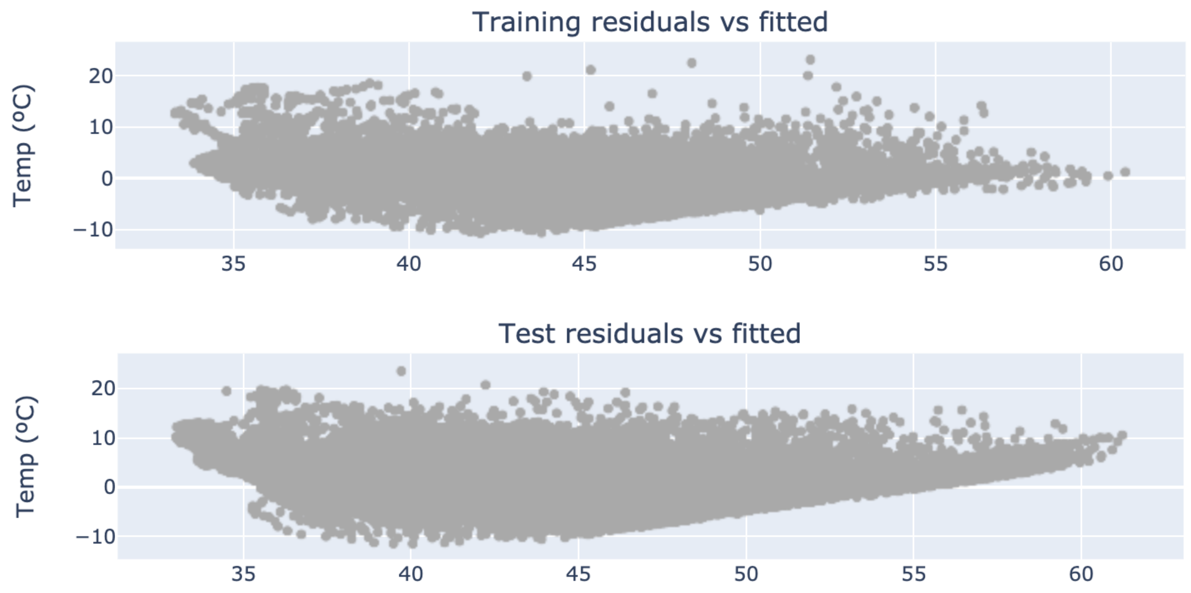


Figure 38. Kernel comparative WT1 Gaussian kernel training and test residuals vs fitted plots. Source: Self-made

Residual plots look acceptable if the errors above are observed. In any case, it can be observed that the residuals do not have a random appearance, in fact a diagonal cut is observed from the value 45 (on the horizontal axis) that rises steadily, limiting the lower or negative errors.

Score metric	Training set	Test set
Explained Variance	0.75	0.71
Max error (ºC)	23.39	23.80
MAE (ºC)	3.26	3.87
RMS (ºC)	3.95	4.79

Table 16. Kernel comparative WT1 Gaussian kernel score table. Source: Self-made

With the score metrics, we can more objectively determine the goodness of fit of the model under discussion. All metrics are worse for the Gaussian kernel than for the linear model. Mean errors higher than 3ºC are found in addition to a much lower explained variance, 71% in the test set.

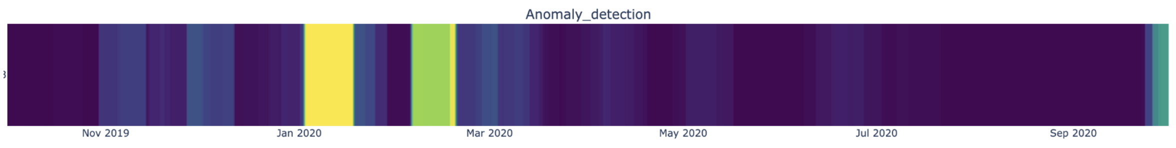


Figure 39. Kernel comparative WT1 Gaussian kernel anomaly detection plot. Source: Self-made

The anomaly detection heat map is quite similar, but with more obvious colour cuts where there are anomalous errors. As the errors are larger and in greater quantity, the threshold is exceeded more frequently, and that is why it is marked with a more visible color.

Process	Runtime (s)
Loading	5.71
Preprocessing	0.57
Training	4391.82 (~73 min)
Testing	90.81 (~1.5 min)
Case	4488.91 (~75 min)
Pipeline	4489.71 (~75 min)

Table 17. Kernel comparative WT1 Gaussian kernel runtime table. Source: Self-made

Lastly, the runtime data speaks for itself. About 1 hour and 15 minutes of total execution for training and subsequent prediction of the test data. Only the training process lasted 73 minutes, which comprises 97% of the time invested in the execution. As mentioned above, the kernel trick allows the processor to do its job in a plausible way, but this does not mean that the process is fast. The hyperparameters were calculated with the aforementioned iterations in the project. If this run had to include the various iterations to find an optimal hyperparameter, the runtime could easily have reached more than 14 hours.

In conclusion, the Gaussian kernel has a worse precision than the linear kernel. Both are quite close to the observed values, but undoubtedly, the linear kernel is a better model that is capable of better capturing the information on the predictors. The latter can be verified by looking at the plots of the residuals. In addition, the execution times are enormously different. The linear kernel method takes 17 seconds while the Gaussian kernel method takes 1 hour and a quarter; about 263 times slower, or put another way: the linear kernel uses 0.3% of the time that the Gaussian kernel needs. Moreover, Gaussian models require hyperparameter optimisation. If this had to be considered, the difference between runtimes would be tremendously different (17 seconds vs 14 hours). After this data, it is evident that the linear kernel offers better technical performance than the Gaussian.

5.2 Approach comparative

Next, a comparison will be made between the different groups of predictors selected and discussed in previous sections. The previous executions were with approach A, which can be consulted in Table 9. That is why in this section the graphs will not be repeated and will only be commented on in the conclusion. The following execution has been carried out with approach B:

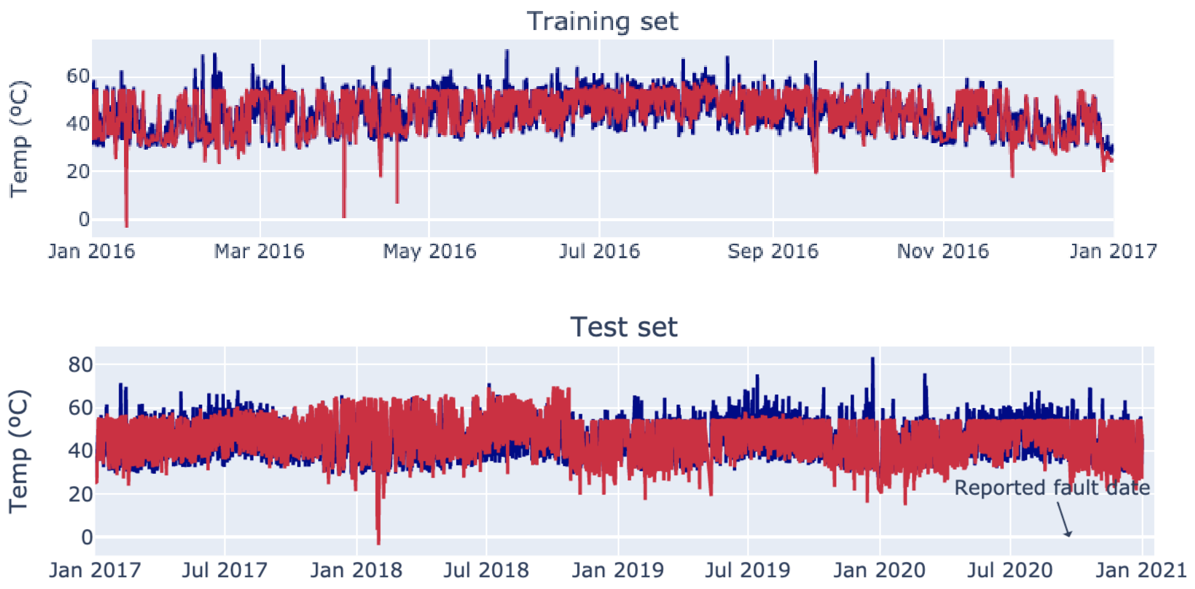


Figure 40. Approach comparative WT1 group "B" training and test set plots. Source: Self-made

It can be seen in the previous figure that the red line (prediction) fits in the same range as the blue line (observation). While the range of values is similar, it can be seen that the model does not fit as well as in the "A" approach. It seems that both for the upper band and the lower band, the prediction falls short. In the test set, the aforementioned behavior can be seen in a more accentuated way.

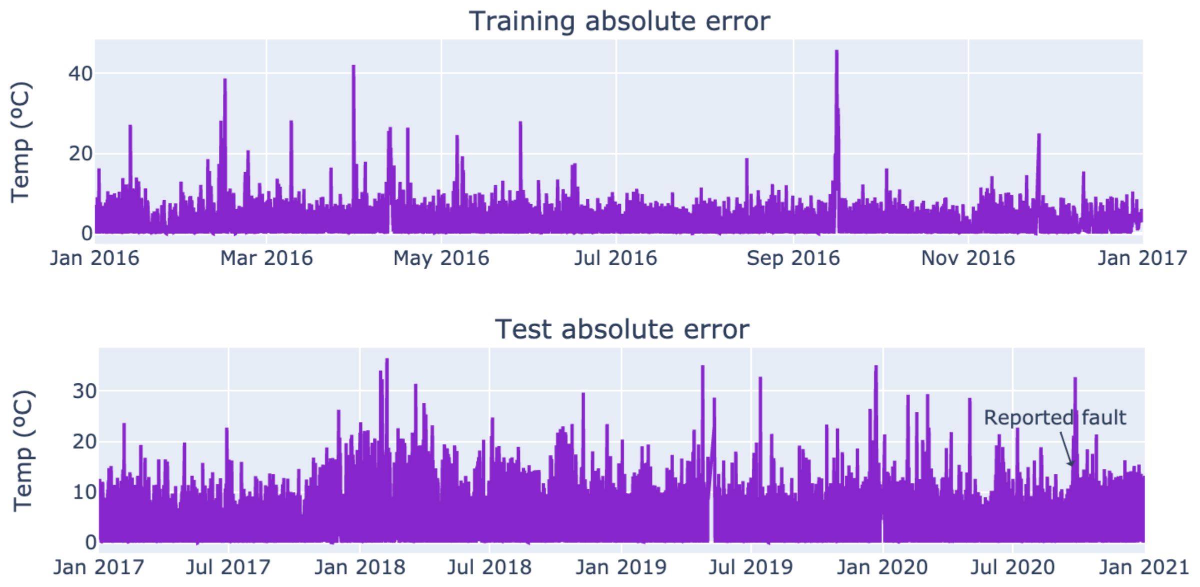


Figure 41. Approach comparative WT1 group "B" training and test error plots. Source: Self-made

The plot of residuals shows a higher intensity and higher values. This indicates that the model tends to fail more in frequency and in order of magnitude. It can be seen that for the test set, the error is much greater than the training set. As with the previous approach, the residual is triggered when the registered failure occurs.

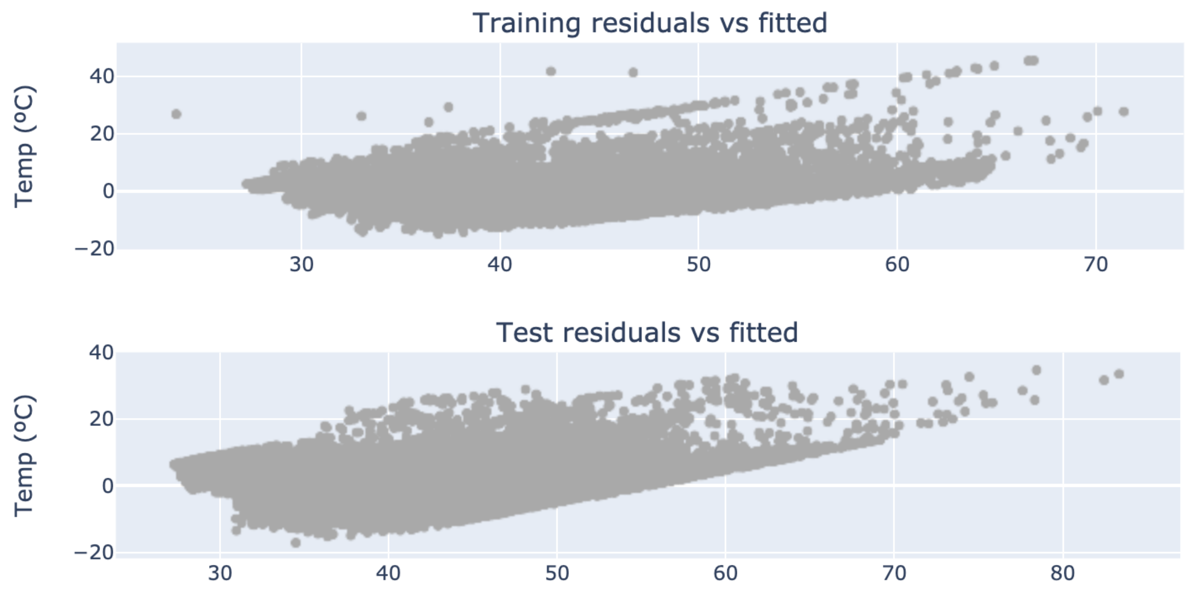


Figure 42. Approach comparative WT1 group "B" training and test residuals vs fitted plots. Source: Self-made

In the training residuals vs fitted profile, it can be seen how the data is quite skewed for high values. An almost perfect line can be observed that cuts the errors from 40ºC. This plot indicates that we will never have high negative errors from 40ºC. While the latter seems to be something positive, it is really telling us that the model is incomplete and is incapable of capturing the information on the predictors well.

Score metric	Training set	Test set
Explained Variance	0.71	0.62
Max error (ºC)	45.91	35.08
MAE (ºC)	3.12	4.04
RMS (ºC)	4.29	5.23

Table 18. Approach comparative WT1 group "B" score table. Source: Self-made

In the table above, it can be seen immediately how the results are worse than with the "A" approach. Much lower explained variance can be observed, indicating that the model does not represent the data with which it has been trained as well. In short, it is a model that is not as good as the previous one.

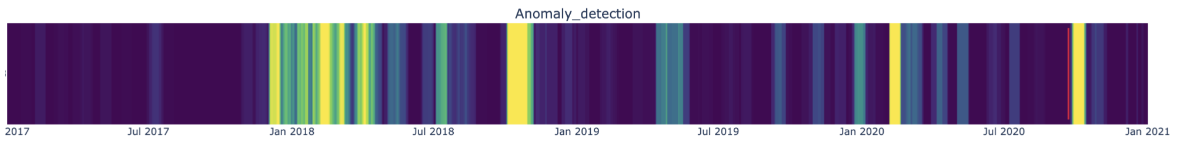


Figure 43. Approach comparative WT1 group "B" anomaly detection plot. Source: Self-made

In the previous anomaly detection plot, a higher number of detected anomalies is observed, but with a progressively lower incidence on the date of the registered failure. It is not a conclusive result.

Process	Runtime (s)
Loading	5.89
Preprocessing	0.44
Training	6.09
Testing	0.01
Case	12.42
Pipeline	12.71

Table 19. Approach comparative WT1 group "B" runtime table. Source: Self-made

The times are very similar to the "A" approach, but the training is notably lower. This can be explained given that the number of predictors used in this model is notably less than those used in approach "A".

The following execution has been carried out with approach C, whose predictors can be found in Table 9:

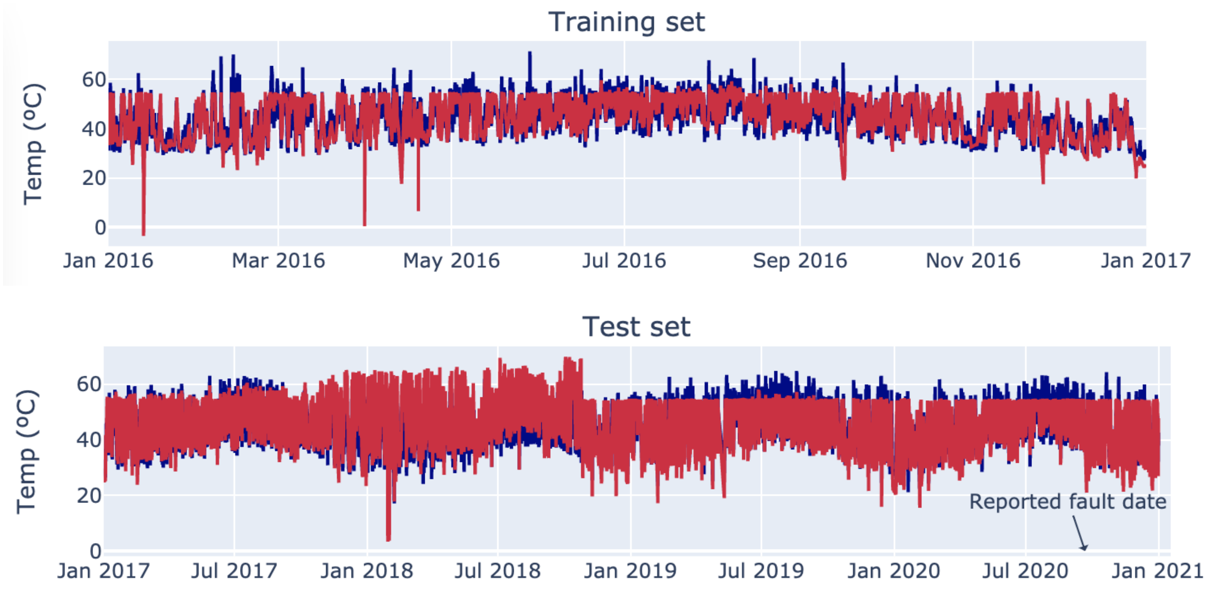


Figure 44. Approach comparative WT1 group "C" training and test set plots. Source: Self-made

In this plot, it can be observed a greater fit than approach "B" and quite similar to approach "A". For the three approaches, it is observed that the prediction is cut off at a value close to 55ºC in the upper band.

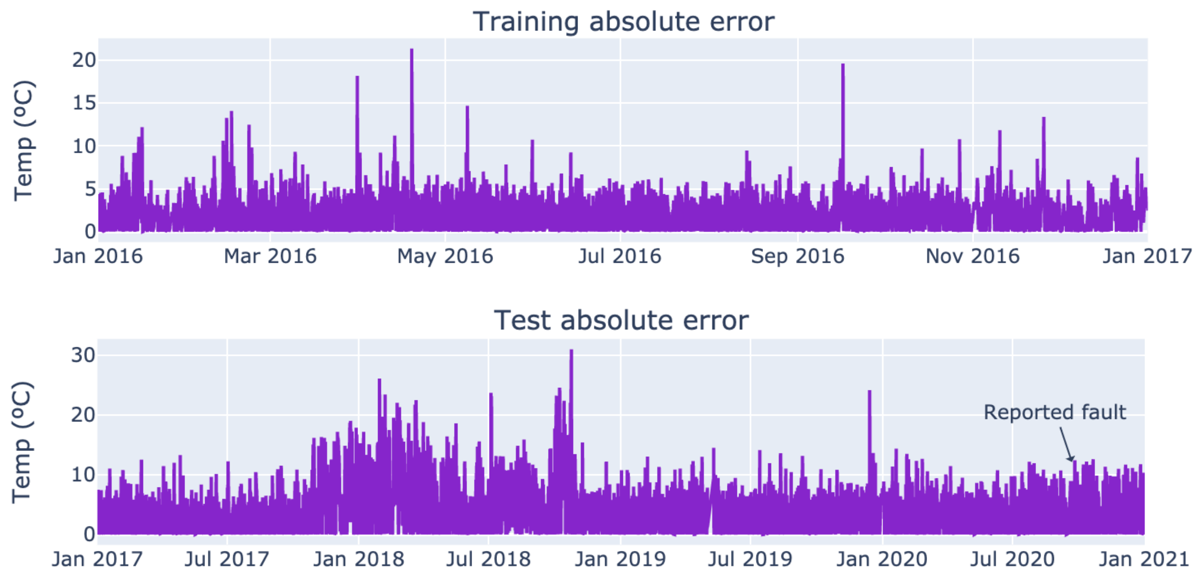


Figure 45. Approach comparative WT1 group "C" training and test error plots. Source: Self-made

The residual plots show a more relaxed behavior towards estimation errors with smaller intensities. Thus, the "C" approach at the residuals level is quite similar to the "A" approach.

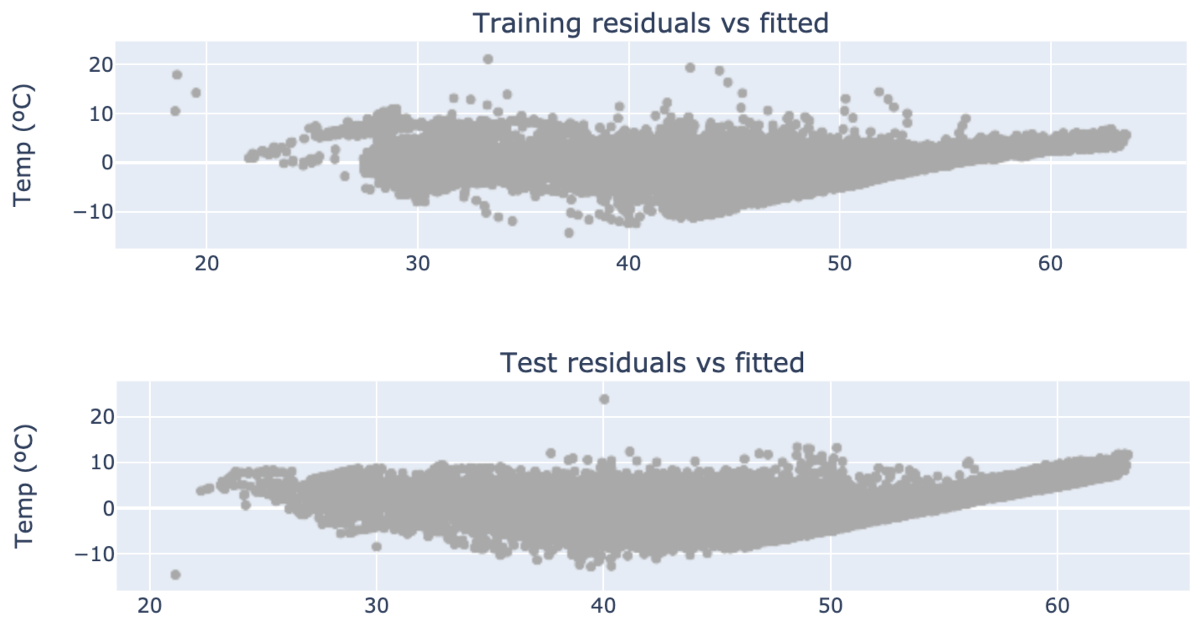


Figure 46. Approach comparative WT1 group "C" training and test residuals vs fitted plots. Source: Self-made

The residuals vs fitted plot has a much more pronounced shape than the "A" approach about cutting residual values at high temperatures. This approach seems to be between the behaviors of approach "A" and "B". It is not as accentuated as the second, but it is certainly more noticeable than for the first approach. It seems that this model fails to capture the information present in the predictors completely as well as approach "A".

Score metric	Training set	Test set
Explained Variance	0.90	0.85
Max error (ºC)	21.33	24.15
MAE (ºC)	2.00	2.74
RMS (ºC)	2.51	3.33

Table 20. Approach comparative WT2353 group "C" score table. Source: Self-made

In the metrics, an improvement of the model can be observed compared to approach "B", although it does not manage to be at the same level as approach "A".

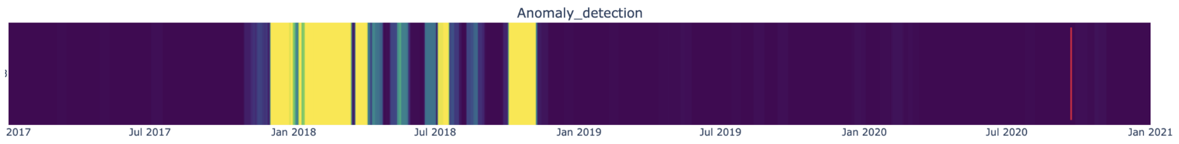


Figure 47. Approach comparative WT1 group "C" anomaly detection plot. Source: Self-made

It can be observed how the approach "B" marks more intensely the anomalies present in the time series. In any case, it can be observed that those close to the date of the registered failure are not "highlighted" as much.

Process	Runtime (s)
Loading	7.00
Preprocessing	0.64
Training	8.36
Testing	0.01
Case	16.01
Pipeline	16.31

Table 21. Approach comparative WT2353 group "C" runtime table. Source: Self-made

Finally, it can be seen in the table above how the runtimes of approach "C" are very similar to the runtimes of approach "A". It can also be observed how they are greater than the "B" approach, but this is explained by the number of predictors used. Approach "A" uses 11 predictors, approach "B" 4 and finally "C" uses 8. The number of predictors determines the speed with which the training phase will take place. The more predictors, the greater the matrix that the kernel has to calculate and therefore the greater the execution time.

5.3 Global results

The following section will show the global results for the turbines of the wind farm for the different approaches studied with the linear kernel. It will focus on the last plot of the pipeline since they are the most user-friendly. The objective is to analyze the behavior of the model in face of future failures. The first result to be analyzed will be approach "A". In the following graph, you can see 16 profiles of anomalies detected in 16 wind turbines. Each has its own model trained from the first year of data, which is presumed to have been healthy. What is displayed in the following graphs is the test data, which is the data that has not been used to train. First, the predicted values have been calculated, compared with those actually observed, and according to the estimation error, a metric of anomaly intensity is calculated last. If the anomaly persists in time, it will accumulate a high value and vice versa.

The WT1, WT2, WT5, WT10, WT12, WT13, WT16, WT20, WT24, WT30 wind turbines have registered faults while the WT3, WT4, WT6, WT7, WT8 and WT9 turbines have no faults. In this way, the behavior of the model with turbines with and without registered faults can be observed.

In this project, a prediction will be considered successful when the indicator surpasses the threshold at most one year before the fault registered happens. If there is no fault or the fault happens more than 1 year later will be considered as a false positive. If there is no indicator 1 year before the fault, this will mean that a false negative in the prediction has happened.

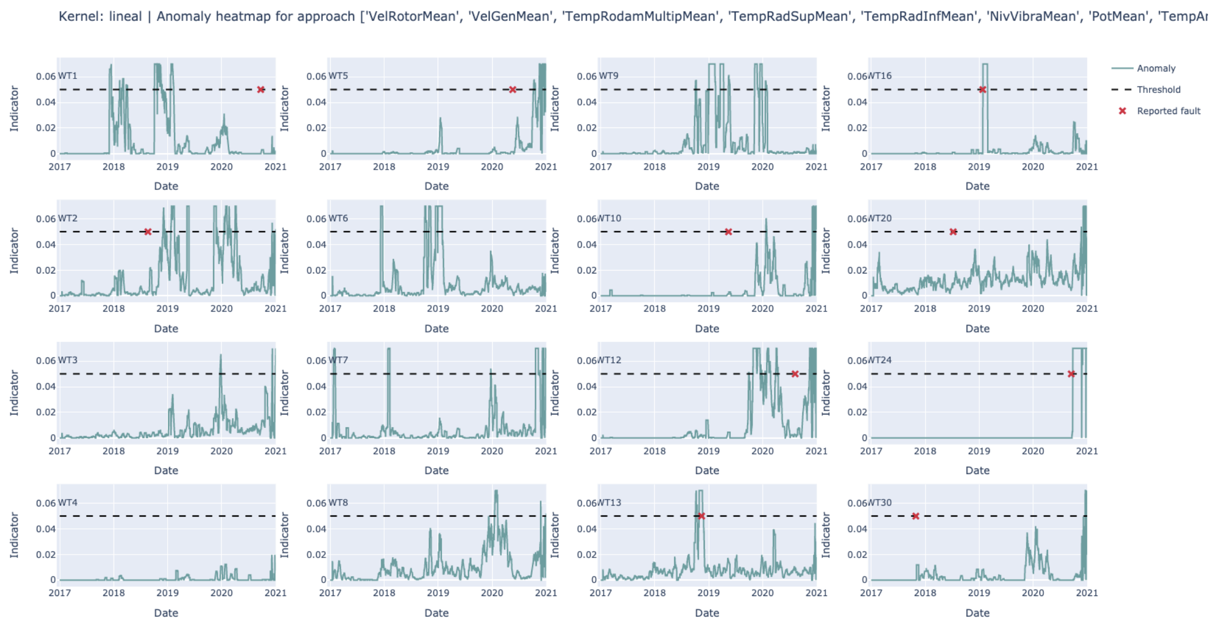


Figure 48. Final results for approach "A". Source: Self-made

The results of 16 turbines for approach "A" can be seen in the previous graph.

It can be observed for this approach that it is possible to capture a signal that exceeds the threshold weeks before the failure in the WT13 turbine. For the WT12 turbine, it can be seen how the prediction is correct a few months before failure. For the WT1, WT2, WT5, WT10, WT16, WT20, WT24 and

WT30 turbines it can be observed that the threshold is only exceeded after or just at the moment the fault is registered, so they were unable to capture signals that indicate that a failure was yet to come. For the WT4 turbine it can be seen how no signal is detected that exceeds the threshold since the turbine seems to be healthy during the entire time series. For most turbines, cases can be found where an anomaly is detected that exceeds the threshold but, in contrast, there is no fault registered one year after the signal.

The results for approach "B" will be analyzed below with the same methodology as for approach "A":

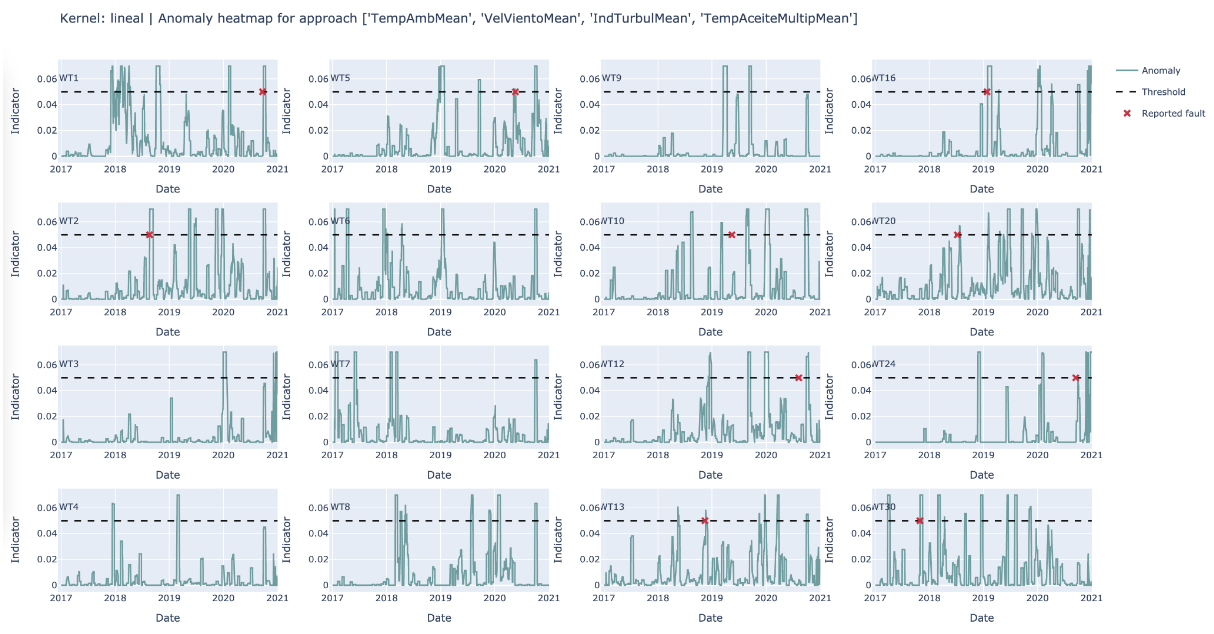


Figure 49. Final results for approach "B". Source: Self-made

For approach "B" it can be seen that the model detects signals that indicate that something unhealthy is happening one year before the failure recorded in the WT1, WT5, WT19, WT12, WT13, WT24 and WT30 turbines. This could indicate that this model performs better than the previous one, but a much higher number of false negatives can also be observed in all the turbines in this sample. In contrast to the previous model, the model fails to obtain a true negative on the WT4 turbine. Only for the WT2, WT16 and WT20 turbines, no anomalies are registered one year before the registered failure. In conclusion, the model is more sensitive to variations between the prediction and the values observed in the real system and causes many false negatives. Finally, the approach "C" will be analysed:

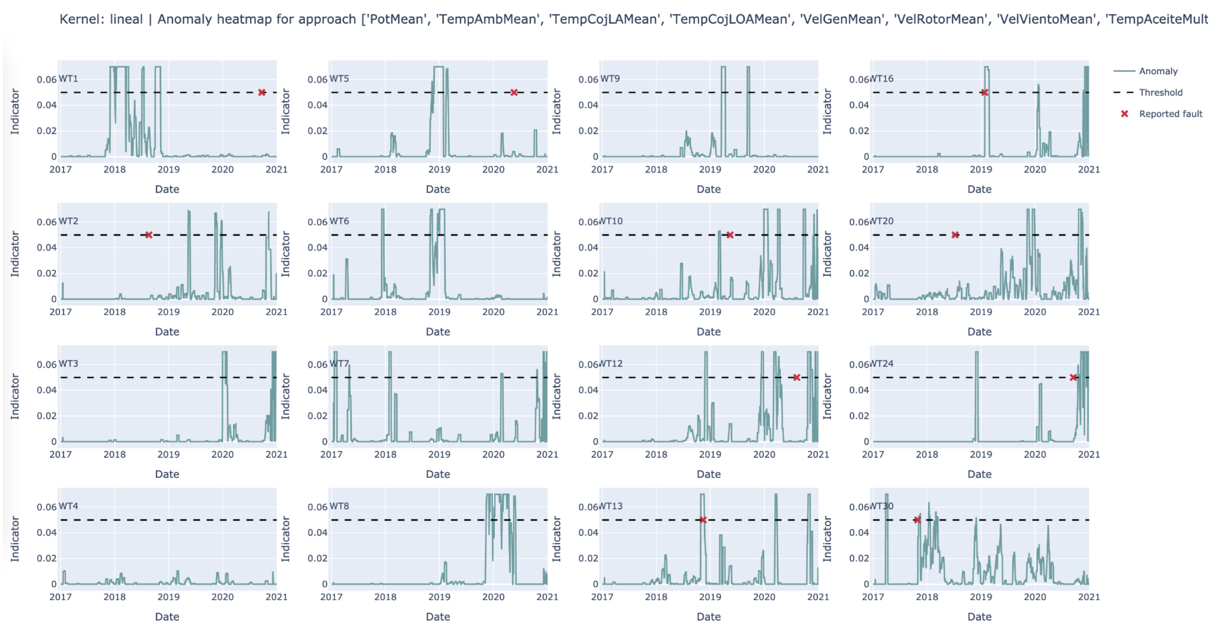


Figure 50. Final results for approach "C". Source: Self-made

The approach "C" has a similar behavior to approach "A", this makes sense since the models share a similar number of predictors and some of these predictors are used in both models. It can be seen that for the WT10, WT12, WT13 and WT20 wind turbines, signals indicating unhealthy behavior of the turbine are detected at most one year before the recorded failure occurs. For the WT13 turbine, it can be seen how the signal triggers only a few days before. A true positive can be seen repeated for the WT4 turbine. It can also be observed that for the WT1, WT2, WT5, WT16, WT20 and WT24 turbines there are no signals that indicate that a failure is going to occur or the signals just appear at the time of failure. False positives can be observed for most turbines, but in much less quantity than in the case of approach "B" and similar in number to approach "A".

Looking at the behavior of the three approaches, it could be concluded that the "C" approach seems to be the approach that works best, considering the number of registered failures that are detected at most one year before they occur and the amount of false negatives that are detected. It seems that the "C" approach has a better trade-off than the rest of the approaches considering these criteria.

Both the threshold and the anomaly factor are parameters adjustable by the wind farm operator. These parameters could make the models able to capture more future failures at the cost of a more significant number of false positives. This is part of a maintenance strategy decision-making that a team should decide to follow.

Prediction failures can be classified into type I and type II. Not all failures have the same impact. For example, if a false positive occurs, that is, the signal detects a major anomaly, but the turbine is healthy, this implies that a maintenance team travels to the turbine without any real need for inspection. This entails an economic loss in transport and personnel. If a false negative occurs, that is, no signal is detected in the event of a future failure, it can cause a productive and economic major loss of the wind farm. Lack of preventive maintenance will turn an avoidable failure into a costly repair and loss of production of the turbine itself. It could be said that false negatives penalize the wind farm feasibility more than false positives.

For all the above, the tool must be configured to penalize false negatives over false positives in order not to incur serious economic and productive losses. That is why, apart from the configurable threshold or anomaly factor parameters, the "C" approach is preferable as it is capable of detecting more failures before they occur, as well as producing fewer false negatives.

6. Conclusions and future work

The entirety of this work has witnessed the effort involved in developing a tool that is capable of detecting anomalies and predicting failures before they occur. The tool has been developed with a dynamic mindset, intending to being able to be used with any turbine, kernel or predictors and with hyperparameter optimization and predictor search functionalities. Thus, it is not a custom-made tool for a specific dataset, but rather made to be used with high flexibility and in a multitude of use cases.

The tool can clean and adapt the input data, use it to train highly customizable normality models, identify anomalies, quantify them and display the results in various graphical interfaces rich of information, thus trying to take advantage of all the content of the tool for the novice or experienced user.

Thanks to the developed tool, it has been possible to experiment with different types of models and different data inputs. These experiments have allowed us to analyze the changes in the behavior of the model and see the importance of the different parameters.

Data preprocessing has proven to be essential in making the models. Input data is from what the model feeds on, so higher quality at this stage translates into a higher quality model. A detailed methodology of data preprocessing has been explained.

Various kernels have been experimented with; the linear, the polynomial and the Gaussian. A brief introduction to kernels and the kernel trick has been made. All kernels have a series of advantages and disadvantages, the main ones being the speed of calculation and the precision of the regression. The linear kernel has shown to have the best trade-off between the speed of calculation and the good-of-fitness of the model, thus being used in the final results.

Finally, anomalies have been identified between the predicted result and the observed one, based on the hypothesis that any observed value sufficiently far from the predicted value may be indicative that something is wrong and the turbine may be potentially damaged. An analysis of the different results thanks to the analysis screens has been carried out and a conclusion has been reached.

From the beginning, this project has encountered several challenges.

In the first place, an attempt has been made to find a solution to a very complex problem. This case shows a high complexity compared to other predictive maintenance models, such as those that attack components such as the main bearing of a wind turbine. The gearbox is a large and complex component which can be broken down into several parts, and failures usually do not share origin or manifest themselves in the same way. In addition, this project has been carried out with unlabeled data, which complicates a correct evaluation of the effectiveness of the models. This has made it necessary to work from normality models that are not assured that they are based on completely healthy data.

The lack of information in the recorded failures also complicates the identification of actual failures. The failures registered in this project could be periodic reviews and therefore giving us false clues, in any case there has been no way to clarify these informational gaps. Finally, a higher level of understanding in wind systems could have helped the choice of predictors to increase the reliability of the model. An extensive analysis of the chosen predictors has been carried out, but probably thanks to experts there would be room for improvement.

Some of these challenges were technological, as this project was trying to find a solution to a real problem with a huge amount of data and limited computational power. The models have been calculated with CPU instead of GPU, and the CPU is still a domestic one. An improvement in computing power would allow better optimization of the hyperparameters and consequently the use of more computationally expensive kernels.

The models produce many unsatisfactory cases where errors can lead to serious economic and productive losses. That is why, although promising, the model still needs to be more fine-tuned to be able to be used in a production environment. It has been observed how the models, although they have not been good enough to be a completely reliable source, can be used as support in decision-making in the day-to-day management of a maintenance team.

Given the results obtained and the potential room for improvement in the methodology, future lines of work could be:

- Improve the choice of predictors thanks to experts in wind generation systems.
- Make several models for different seasons
- Use this model as one more within an ensemble method. This model may prove to be effective in the event of a certain gearbox failure and can help complement a predictive model based on various models.
- Make several models by operating regimes as shown in the following figure:

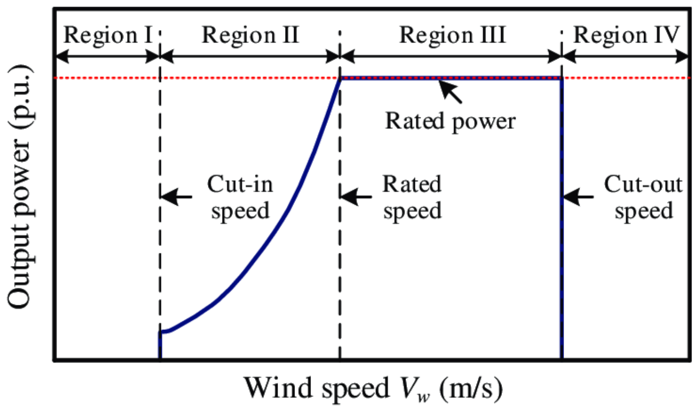


Figure 51. Different operational regimes of a wind turbine. Source: Yang et al.

7. Acknowledgements

The author of this project wishes to thank the Smartive company for providing the data used in this project. The delivery of it has provided the opportunity to work with real data from a wind farm that is currently in production. Thus, the data seen in this project are not synthetic or acquired in a laboratory, but from a real case.

The author is also deeply grateful to the two directors of this project, Yolanda Vidal and Francesc Pozo, for their constant guidance and mentoring in the key points of this project as well as in the preparation of this report. His experience and way of working have given this thesis the structure and seriousness that any student could wish for.

Without the help of Smartive, Yolanda V. and Francesc P. this project would not have been half as enriching as it has been both academically and personally.

8. Bibliography

Alvarez, EJ., and AP. Ribaric. "An improved-accuracy method for fatigue load analysis of wind turbine gearbox based on SCADA." *Renewable Energy*, vol. 115, 2018, pp. 391-399. https://doi.org/10.1016/j.renene.2017.08.040.

Asociación Empresarial Eólica. "Potencia instalada y generación." *AEE*, AEE, 2020, https://www.aeeolica.org/sobre-la-eolica/la-eolica-espana/potencia-instalada-y-generacion.

Beck, JL., and Y. Huang. "Hierarchical sparse bayesian learning for strucutral health monitoring with incomplete modal data." *International Journal for Uncertainty Quantification*, 2015, pp. 139-169. 10.1615/Int.J.UncertaintyQuantification.2015011808.

C., Martin. "Wind Turbine Blades Can't Be Recycled, So They're Piling Up in Landfills." *Bloomberg Green*, Bloomberg, 05 02 2020, https://www.bloomberg.com/news/features/2020-02-05/wind-turbine-blades-can-t-be-recycled-so-they-re-piling-up-in-landfills.

Cheng, F., et al. "Fault Diagnosis of Wind Turbine Gearboxes Based on DFIG Stator Current Envelope Analysis." *IEEE Transactions on Sustainable Energy*, vol. 10, no. 3, 2019, pp. 1044 - 1053. 10.1109/TSTE.2018.2859764.

Cui, Y., et al. "An Anomaly Detection Approach Based on Machine Learning and SCADA Data for Condition Monitoring of Wind Turbines." *IEEE International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, 2018. 10.1109/PMAPS.2018.8440525.

Da Vila, S. *Detección de outliers en grandes bases de datos*. Santiago de Compostela, María José Lombardía Cortiña, 2020, http://eio.usc.es/pub/mte/descargas/ProyectosFinMaster/Proyecto_1780.pdf.

"Design of SCADA for remote control and operation of wind power plants." *Sparesinmotion*, 2020, https://electrical-engineering-portal.com/download-center/books-and-guides/electricity-generation-t-d/scada-wind-power-plants.

Ecomodeller. "tsod: Anomaly Detection for time series data." *GitHub*, https://github.com/DHI/tsod.

Electrical Academia. "Horizontal-Axis Wind Turbine (HAWT) Working Principle | Single Blade, Two Blade, Three-Blade Wind Turbine." *Electrical Academia*, 2020, https://electricalacademia.com/renewable-energy/horizontal-axis-wind-turbine-hawt-working-principle-single-blade-two-blade-three-blade-wind-turbine/.

European Wind Energy Association (EWEA). *Economics of Wind Energy*. Brussels, Belgium, 2009.

Faulstich, S., et al. "Wind turbine downtime and its importance for offshore deployment." *Wind Energy*, no. 14, 2010, 327.337, https://onlinelibrary.wiley.com/doi/epdf/10.1002/we.421.

Feng, Y., et al. *Monitoring wind turbine gearboxes*. Wind Energy, 2013.

Gagnon R., et al. "Wind Farm (DFIG Phasor Model)." *Mathworks*, https://uk.mathworks.com/help/physmod/sps/ug/wind-farm-dfig-phasor-model.html.

Grace Z. "What is the kernel trick? Why is it important?" *Medium*, 11 11 2018, https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d.

Hossain, M. Liton, et al. "Methods for Advanced Wind Turbine Condition Monitoring and Early Diagnosis: A Literature Review." *Energies*, 2018, https://doi.org/10.3390/en11051309.

Huang, Z., et al. "Prediction of oil temperature variations in a wind turbine gearbox based on pca and an spc-dynamic neural network hybrid." *Tsinghua University*, 2018. 2018;58(6):539–46.

Infield, D., and Y. Wang. "Supervisory control and data acquisition data-based non-linear state estimation technique for wind turbine gearbox condition monitoring." *European Wind Energy Association*, 2013. https://doi.org/10.1049/iet-rpg.2012.0215.

IRENA 2019. "Future of wind: Deployment, investment, technology, grid integration and socio-economic aspects." *International Renewable Energy Agency*, 2019, https://periscope-network.eu/analyst/future-wind-deployment-investment-technology-grid-integration-and-socio-economic-aspects. Accessed 15 05 2021.

IRENA 2020. "Renewable capacity statistics 2020 International Renewable Energy Agency." *Renewable Capacity Statistics 2020*, https://irena.org/publications/2020/Mar/Renewable-Capacity-Statistics-2020. Accessed 15 05 2021.

Konstantakopoulos, IC., et al. "Using domain knowledge features for wind turbine diagnostics." *15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2016.

Leahy, K., et al. "Diagnosing wind turbine faults using machine learning techniques applied to operational data." *IEEE international conference on prognostics and health management*, 2016.

Minitab Blog Editor. "Why You Need to Check Your Residual Plots for Regression Analysis: Or, To Err is Human, To Err Randomly is Statistically Divine." *Minitab Blog*, 05 04 2012, https://blog.minitab.com/en/adventures-in-statistics-2/why-you-need-to-check-your-residual-plots-for-regression-analysis.

Monforte, Javier. "La edad de los parques eólicos abre nuevas oportunidades al sector del mantenimiento de aerogeneradores." *energética21*, 29 11 2017, https://www.energetica21.com/noticia/la-edad-de-los-parques-elicos-abre-nuevas-oportunidades-al-sector-del-mantenimiento-de-aerogeneradores.

Nadaï, A., and D. Van der Horst. "Wind power planning, landscapes and publics." *Land Use Policy*, no. 27, 2010, pp. 181-184.

NREL. "2019 Cost of Wind Energy Review." NREL, 12 2020, https://www.nrel.gov/docs/fy21osti/78471.pdf.

Oyague, F. *Estimation of Blade and Tower Properties for the Gearbox Research Collaborative Wind Turbine*. Technical Report NREL/EL-500-42250 ed., 2007. *National Renewable Energy Laboratory (NREL)*.

Parthasarathy, G., et al. *Use of scada data for failure detection in wind turbines*. 2011.

Press, Gil. "Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says." *Forbes*, 23 03 2016, https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=2127cdb86f63. Accessed 23 06 2021.

Python3. “multiprocessing — Process-based parallelism.” *Python Documentation*, https://docs.python.org/3/library/multiprocessing.html.

Qiao, W., and D. Lu. “A survey on wind turbine condition monitoring and fault diagnosis—Part II: Signals and signal processing methods.” *IEEE Transactions on Industrial Electronics*, 2015.

Qiao, W., and DG Lu. “A Survey on Wind Turbine Condition Monitoring and Fault Diagnosis—Part II: Signals and Signal Processing Methods.” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 10, 2015, pp. 6546 - 6557.

Rosenmai, P. “Using the Median Absolute Deviation to Find Outliers.” *Eureka Statistics*, 25 11 2013, https://eurekastatistics.com/using-the-median-absolute-deviation-to-find-outliers/. Accessed 05 02 2020.

Rosenthal, G., and J. Rosenthal. *Statistics and Data Interpretation for Social Work*. Springer, 2011.

Sequeria, C., et al. “Analysis of the efficiency of wind turbine gearboxes using the temperature variable.” *Renewable Energy*, vol. 135, 2019, pp. 465-472. https://doi.org/10.1016/j.renene.2018.12.040.

Singh, S. “Understanding the Bias-Variance Tradeoff.” *Towards Data Scinece*, 21 05 2018, https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229.

TheMachineLearners. “Entiende de una vez qué es el Tradeoff Bias-Variance.” *The Machine Learners*, https://themachinelearners.com/tradeoff-bias-variance/.

Tipping, M.E. “Sparse Bayesian learning and the relevance vector machine.” *Journal of Machine Learning Research,*, vol. 1, 2001, pp. 211 - 244.

U.S. Department of Energy (U.S. DOE). *Wind Energy by 2030: Increasing Wind Energy's Contribution to U.S. Electricity Supply*. DOE/GO-102008- 2567, 2008, Washington, DC.

Vidal, Y. *SCADA Data for Wind Turbine Gearbox Fault Prognosis*. 2021.

Vidal, Yolanda. "Máquinas de Soporte Vectorial." *YouTube*, 23 04 2020, https://www.youtube.com/watch?v=7Q9eXrB1ub0.

Wang, J., et al. "Multiscale filtering reconstruction for wind turbine gearbox fault diagnosis under varying-speed and noisy conditions." *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, 2018, pp. 4268 - 4278. 10.1109/TIE.2017.2767520.

Wei, T. *Multi-fault detection and failure analysis of wind turbine gearbox using complex wavelet transform*. no. 93-8, Renewable Energy, 2016, p. 591.

W.L., et al. "Wind turbine gearbox failure identification with deep neural networks." *IEEE Trans Ind Inf*, 2017.

Xu, Q., et al. "Quantile regression neural network-based fault detection scheme for wind turbines with application to monitoring a bearing." *Wind Energy Open Access* [Hefei], 15 July 2019.

Yang, M., et al. "Probabilistic Short-Term Wind Power Forecast Using Componential Sparse Bayesian Learning." *IEEE Transactions on Industry Applications*, vol. 49, no. 6, 2013, pp. 2783 - 2792. 10.1109/TIA.2013.2265292.

Yang, Z., et al. "LPV Model Based Sensor Fault Diagnosis and Isolation for Permanent Magnet Synchronous Generator in Wind Energy Conversion Systems." *Applied Sciences*, vol. 8, no. 10, 2018. 10.3390/app8101816.

Zellner, A. "Simplicity, Inference, and Modelling: Keeping it Sophisticatedly Simple." *Cambridge University Press* [Cambridge], 2001.

Zeng X.J., et al. "Gearbox oil temperature anomaly detection for wind turbine based on sparse Bayesian probability estimation." *Electrical Power and Energy Systems*, 2020.

Zhang, D., and Z. Qian. "Probability warning for wind turbine gearbox incipient faults based on SCADA data." *Chinese Automation Congress (CAC)*, 2017. 10.1109/CAC.2017.8243420.

Zhang, L., and ZQ Lang. *Wavelet energy transmissibility function and its application to wind turbine bearing condition monitoring*. IEEE Transactions on Sustainable Energy ed., 2018.

9. Annex

9.1 Readme file

What is this tool?
This is a tool created to explore different approaches and kernels within the RVM algorithm. In this tool you can analyse for a wind farm the capabilities of a model to identify faults before they happen.

How can I use this tool?
There are some user defined general parameters one can define in order to make this tool run:

- WT_ID_BACH: Here you will put a list of the different wind turbine ids you want to analyse.
- WT_ID: Gaussian kernel just allows you to do an execution per wind turbine, if you are using the Gaussian kernel you will put here the id of the wind turbine you want to analyse.
- APPROACH: Here you will put a string with the desired approach of predictors you want to use. Currently, there are 3 different approaches defined: 'A', 'B' and 'C'.
- KERNEL: Here you will define a string with the desired kernel. There are three kernels configured: 'lineal', 'polynomial' and 'gaussian'.
- TARGET: Here you will define a string with the desired target variable.
- ANOMALY_FACTOR: Here you will define an integer with the desired anomaly factor.
- SPLIT_THRESHOLD: If you want to overtake the training and testing dates defined in the configuration, you can put this on True and a standard split threshold of 75% training and 25% testing will be used.
- RANDOM_FEATURES: This tool is capable of selecting random predictors for exploratory purposes. If set to True you will activate this feature.
- N_APPROACHES: Number of approaches the tool will create if RANDOM_FEATURES is True
- N_FEATURES: Number of predictors each approach will have if RANDOM_FEATURES is True
- HYPER_PARAMETER_TUNING: Only available for Gaussian kernels. If set to True an hyper parameter tuning will be done. If True, you can configure a dictionary with the number of desired iterations to find the best hyperparameter and the parameter range to search for it.

Further specific turbine definition can be done in wt_configuration.py file

How to install MATLAB engine in Python
Source:
https://uk.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html

<u>Note: You must have installed MATLAB</u>

At a macOS or Linux operating system prompt (you might need administrator privileges to execute these commands) —

Open the terminal
Go to your matlab root directory and go into "yourmatlabroot/extern/engines/python" and write:

cd "yourmatlabroot/extern/engines/python"

> A: Install setup.py into your current working python directory. If you just use the default one you can write:
>
> *sudo python setup.py install*
>
> B: If you are using a Python that is used in your local venv instead of "python" you have to specify your current python working directory like:
>
> *sudo /Users/PycharmProjects/MSc_Thesis/venv/bin/python setup.py install*

And that's it!