Máster Interuniversitario
en
Ciencia de Datos

Curso académico 2020-2021

MANTENIMINTO PREDICTIVO Y ANÁLISIS ESPECTRAL: DE FOURIER AL APRENDIZAJE AUTOMÁTICO
(Predictive Maintenance and Spectral Analysis: From Fourier to Machine Learning)

Trabajo de Fin de Máster
para acceder al
Máster en Ciencia de Datos

Autora: Judith Sáinz-Pardo Díaz
Director: Steven Van Vaerenbergh
Co-Director: Carlos Alberto Meneses Agudo
Junio - 2021

Abstract

Predictive maintenance is a set of techniques that analyzes physical parameters to prevent equipment failure. The present study explores state-of-the-art techniques to address the main problems involved in predictive maintenance, namely the detection and classification of failures, and the time-to-failure prediction. These techniques are applied to time series obtained from sensors placed on bearings.

First, the classical techniques for fault detection are reviewed, based on spectral analysis techniques using the Fourier and the Hilbert-Huang transforms. A review is performed of contemporary machine-learning techniques for failure detection, and experimental results are presented on time series obtained from several open-source databases of bearing data.

Next, an overview is given of the time-to-failure prediction problem, which is approached through the context of anomaly detection. Several machine-learning techniques are presented to tackle this problem, and a comparison of the obtained classifiers is included based on a wide range of experimental results.

The present study lays the research groundwork for an industrial project that aims to operate on live data. All developed code was written in Python and distributed through an open-source repository.

Key words: predictive maintenance, bearing, anomaly detection, fault detection, machine learning.

Resumen

El mantenimiento predictivo es un conjunto de técnicas que analizan parámetros físicos para prevenir fallos en los equipos. El presente estudio explora las técnicas más avanzadas para abordar los principales problemas que plantea el mantenimiento predictivo, a saber, la detección y clasificación de los fallos, y la predicción del tiempo hasta el fallo. Estas técnicas se aplican a series temporales obtenidas a partir de sensores colocados en los rodamientos.

En primer lugar, se revisan las técnicas clásicas de detección de fallos, basadas en técnicas de análisis espectral mediante las transformadas de Fourier y de Hilbert-Huang. Se realiza una revisión de las técnicas contemporáneas de aprendizaje automático para la detección de fallos, y se presentan resultados experimentales sobre series temporales obtenidas a partir de varias bases de datos de código abierto de rodamientos.

A continuación, se ofrece una visión general del problema de la predicción del tiempo hasta el fallo, que se aborda a través del contexto de la detección de anomalías. Se presentan varias técnicas de aprendizaje automático para abordar este problema, y se incluye una comparación de los clasificadores obtenidos basada en una amplia gama de resultados experimentales.

El presente estudio sienta las bases de investigación para un proyecto industrial que pretende operar con datos reales. Todo el código desarrollado se ha escrito en Python y se ha distribuido a través de un repositorio de código abierto.

Palabras clave: mantenimiento predictivo, rodamiento, detección de anomalías, detección de fallos, aprendizaje automático.

Contents

Acronyms

BPFI: Ball Pass Frequency Inner (internal race failure frequency).

BPFO: Ball Pass Frequency Outer (failure frequency of the external race).

DFT: Discrete Fourier Transform.

EMD: Empirical Mode Decompositon.

FCF: Fault Characteristic Frequency.

FFT: Fast Fourier Transform algorithm.

fr: Rotational frequency.

Fs. Sampling frequency.

FTF: Fundamental Train Frequency.

GMM: Gaussian Mixture Model.

HHT: Hilbert-Huang Transform.

IMF: Intrinsic Mode Function.

kNN: k-Nearest Neighbors.

OCSVM: One Class SVM.

PCA: Principal Component Analysis.

SVM: Support Vector Machine.

Chapter 1

Introduction

1.1 Context, motivation

Predictive maintenance of machinery consists of a set of techniques and actions to detect failures of the equipment under study in incipient stages, avoiding that they affect the operation in advanced stages of a process. In addition, vibration analysis is a method used to identify failures in rotating machinery, such as bearing or gear failure, misalignment, unbalance, etc. Recently, numerous studies can be found about the diagnosis of bearing failures, as these are the main components of a rotating machine (see, for instance, [11]). Furthermore, the diagnosis of faults in machinery subjected to a rotation speed that varies over time, has become a new line of research for many authors, as the more classic techniques do not solve this problem.

The complexity of modern machinery equipment requires the use of new diagnostic techniques and tools for monitoring and automating results. Although there are classical techniques widely studied for the diagnosis of failures in rotating machinery, many methods that are more effective and accurate in early detection have yet to be studied.

In order to illustrate the motivation of the study and predictive maintenance, we present the following real case: The wind turbines that produce energy, consist of a braking system for cases in which the blades rotate at excessive speed. However this automatic braking system can fail and, in very windy situations, the high speed rotation of the blades can cause them and the turbine to break. This could be avoided either by regular maintenance, which would be costly and dangerous for those who perform it, or by enhancing automatic and predictive maintenance.

1.2 Scope, objectives and context

As we have commented in the previous section, numerous problems can be avoided and savings increased by carrying out effective machinery maintenance. In addition, the automation of maintenance instead of carrying out regular maintenance allows for a reduction in costs and risks to the personnel who carry it out. This can be achieved by monitoring the condition of the machinery so that the state of the equipment can be checked and the time until failure can be predicted.

Our aim will be to illustrate predictive maintenance and spectral analysis techniques and apply them to different cases of signal processing and early detection of machinery failure. We will focus on bearing failure, as this is one of the main reasons for machinery breakdowns.

The following figure shows a schematic diagram of the bearing components

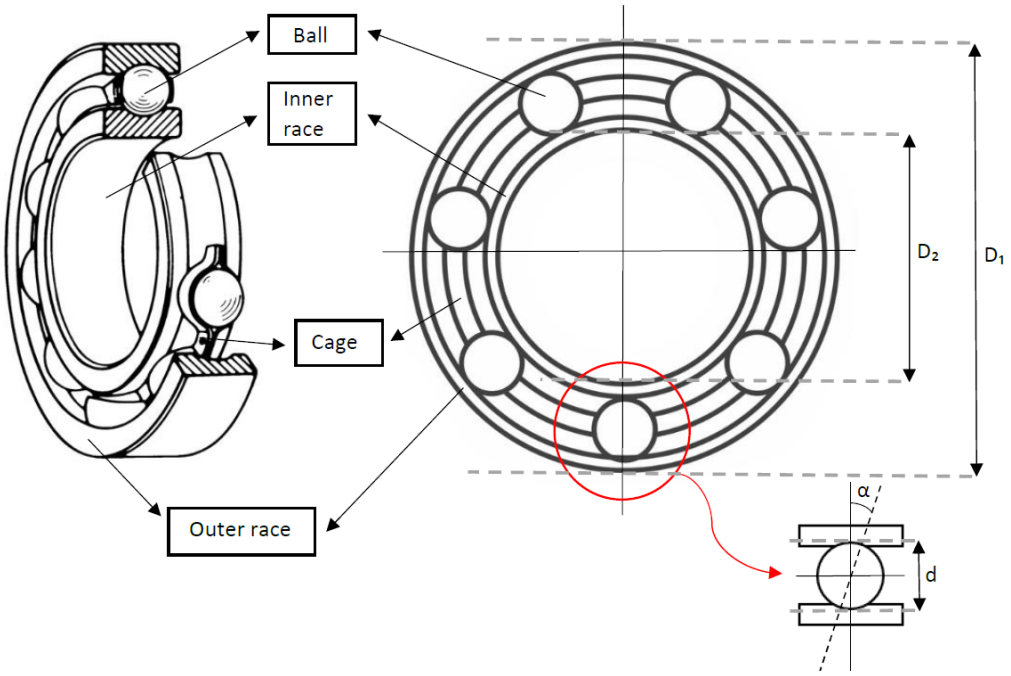


Figure 1.1: Diagram showing the parts of a rolling bearing.

Let us briefly introduce some key concepts: Be FTF (Fundamental Train Frequency) the number of turns the bearing cage makes each time the shaft makes a full turn, we will especially focus on two types of failure, which we present together with their Fault Characteristic Frequency (FCF) (see [20]):

- BPFO (Ball Pass Frequency Outer, failure frequency of the external race). It physically corresponds to the number of balls or rollers that pass through a given point in the outer race each time the shaft makes a complete turn. Its FCF:

$$BPFO : f_o = \frac{N_B f_r}{2}\left(1 - \frac{d}{D}\cos(\alpha)\right),$$

  where N_B is the number of balls, f_r the shaft rotational frequency, α is the contact angle, d the diameter of the ball and D is the pitch diameter (in Figure 1.1, $D = \frac{D_1+D_2}{2}$).

- BPFI (Ball Pass Frequency Inner, internal race failure frequency). It physically corresponds to the number of balls or rollers that pass through a given point on the inner race each time the shaft makes a complete turn.

$$BPFI : f_i = \frac{N_B f_r}{2}\left(1 + \frac{d}{D}\cos(\alpha)\right)$$

Our final objective will be to present techniques that can later be applied to other cases of machinery, in order to be able to anticipate and solve this type of failure. This baseline study will be carried out using open datasets that will allow us to study the different techniques that can be applied once other data collected under similar conditions are available.

1.3 Structure of the work

This study is structured as follows: in this first chapter we have set out the background, the motivation for the scope of the study, as well as the context and some key concepts about the machinery on which the study is based. In Chapter 2 we present some aspects of the fundamental theory of signal

processing and spectral analysis, such as the Fourier transform, the Hilbert transform and certain techniques for signal processing and noise removal. We conclude this chapter by explaining the state of the art.

In Chapter 3 we present the two main problems of predictive maintenance: the detection of failures and the prediction of time to failure. The first problem will be divided into two cases, depending on whether the signals are captured under constant or variable rotational velocity conditions. Different techniques and their development to solve these problems are presented. Regarding time-to-failure prediction, different techniques will be presented to detect anomalies to anticipate failure.

In the fourth chapter we study in detail the results obtained in the cases studied in the previous chapter, and finally in Chapter 5 we conclude the study with some general conclusions and suggest how the study will be carried out when the other kind of data are obtained.

Chapter 2

Theoretical basis and state of the art

2.1 Literature and algorithms

Let us present some classical techniques and the advances in the field of predictive maintenance, starting as the title of this work indicates, by Fourier analysis:

2.1.1 Fourier analysis

Jean-Baptiste Joseph Fourier was a French mathematician and physicist known for stating that *"any arbitrary function $f(x)$ can be expressed as an infinite series of sinuses"*. He also devoted much of his work to studying the decomposition of periodic functions into converging trigonometric series called *Fourier series*. With the Fourier series method many problems associated with partial derivative equations, such as the heat equation or the wave equation, can be solved.

We are now going to define the mathematic tool called "Fourier transform", which is used to transform a function defined in the time domain (as in our case, a signal) into the frequency domain. We will see throughout this work that this is fundamental for studying signals and performing predictive maintenance.

Definition 2.1.1. *Given a function $f(x)$ defined for each $x \in \mathbb{R}$, its Fourier Transform is another function, called $F(f)$, which, for each $\xi \in \mathbb{R}$ is given by:*

$$F(f)(\xi) = \int_{-\infty}^{+\infty} f(x)e^{-i\xi x}dx,$$

if the integral exists. [7].

Although this tool has many uses in the field of differential equations, we are going to use it to transform a signal that does not give us clear information in the time domain, into the frequency domain.

We will illustrate the use of this key tool in the theoretical basis of signal processing with an example. To do so, we consider the following function:

$$f(x) = \begin{cases} \cos(2x) + 2 \cdot \sin(x) & \text{if} \quad |x| < 2\pi \\ 0 & \text{in other case} \end{cases} \tag{2.1}$$

We calculate its Fourier transform as follows:

$$F(f)(\xi) = \int_{-\infty}^{+\infty} f(x)e^{-i\xi x}dx =$$

$$\int_{-2\pi}^{2\pi} (\cos(2x) + 2 \cdot \sin(x))e^{-i\xi x}dx = \int_{-2\pi}^{2\pi} \cos(2x)e^{-i\xi x}dx + 2\int_{-2\pi}^{2\pi} \sin(x)e^{-i\xi x}dx$$

We solve these two integrals separately, for which we use that $sin(x) = \frac{e^{ix}-e^{-ix}}{2i}$ and $cos(x) = \frac{e^{ix}+e^{-ix}}{2}$ (which can be proved using Euler's identity and trigonometric properties):

$$\int_{-2\pi}^{2\pi} \cos(2x)e^{-i\xi x}dx = \int_{-2\pi}^{2\pi} \left(\frac{e^{2xi}+e^{-2xi}}{2}\right) e^{-i\xi x}dx = \frac{1}{2}\left[\int_{-2\pi}^{2\pi} e^{ix(2-\xi)}dx + \int_{-2\pi}^{2\pi} e^{-ix(2+\xi)}dx\right] =$$
$$= \frac{1}{2}\left[\frac{e^{2\pi i(2-\xi)} - e^{-2\pi i(2-\xi)}}{i(2-\xi)} + \frac{e^{2\pi i(2+\xi)} - e^{-2\pi i(2+\xi)}}{i(2+\xi)}\right] = \frac{1}{2}\left[\frac{2i\sin(2\pi(2-\xi))}{i(2-\xi)} + \frac{2i\sin(2\pi(2+\xi))}{i(2+\xi)}\right] =$$
$$= -\frac{\sin(2\pi\xi)}{2-\xi} + \frac{\sin(2\pi\xi)}{2+\xi} = \boxed{-\frac{\xi\sin(2\pi\xi)}{4-\xi^2}}$$

$$\int_{-2\pi}^{2\pi} \sin(x)e^{-i\xi x}dx = \int_{-2\pi}^{2\pi} \left(\frac{e^{xi}-e^{-xi}}{2i}\right) e^{-i\xi x}dx = \frac{1}{2i}\left[\int_{-2\pi}^{2\pi} e^{ix(1-\xi)}dx - \int_{-2\pi}^{2\pi} e^{-ix(1+\xi)}dx\right] =$$
$$= -\frac{1}{2}\left[\frac{e^{2\pi i(1-\xi)} - e^{-2\pi i(1-\xi)}}{1-\xi} - \frac{e^{2\pi i(1+\xi)} - e^{-2\pi i(1+\xi)}}{1+\xi}\right] = -\frac{1}{2}\left[\frac{2i\sin(2\pi(1-\xi))}{1-\xi} - \frac{2i\sin(2\pi(1+\xi))}{1+\xi}\right] =$$
$$= i\left[-\frac{\sin(2\pi\xi(1-\xi))}{1-\xi} + \frac{\sin(2\pi\xi(1+\xi))}{1+\xi}\right] = i\left[\frac{\sin(2\pi\xi)}{1-\xi} + \frac{\sin(2\pi\xi)}{1+\xi}\right] = \boxed{\frac{2i\sin(2\pi\xi)}{1-\xi^2}}$$

Thus, for this particular case, we have that the Fourier transform of the function $f(x)$ defined in Equation (2.1), is:

$$F(f)(\xi) = -\frac{\xi\sin(2\pi\xi)}{4-\xi^2} + \frac{4i\sin(2\pi\xi)}{1-\xi^2}.$$

In signal processing, Discrete Fourier Transform (DFT) is frequently used. This mathematical tool also allows a signal to be transformed from the time domain to the frequency domain, and, in addition, DFT requires that the input function be a discrete sequence of finite duration. Its use in the field of digital signal processing is due to the presence of sampled signals.

Definition 2.1.2. *Be $x(n)$ a discrete aperiodic signal in time. The discrete Fourier transform of this signal is defined as:*

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi k\frac{n}{N}}, \quad k = 0, \dots, N-1.$$

Note that $X(k)$, $k = 0, \dots, N-1$ is a set of complex numbers. (See [2]).

The most widely used algorithm for calculating the Discrete Fourier Transform is, due to its efficiency, the Fast Fourier Transform algorithm (FFT). Although there are different implementations of the FFT algorithm, in practice we will use those based on the Cooley–Tukey FFT algorithm, which are implemented in MATLAB and Python.

The Cooley–Tukey FFT algorithm it is a recursive algorithm with a computation time of $O(N\log(N))$. The recursive idea consists of: on the one hand, find the DFTs of the entries with an even index $(x_2, x_4, \dots)$, on the other hand those with an odd index, and combine the two results to obtain the DFT of the whole sequence. [9].

Be x a signal and N its length, Figure 2.1 shows an example made in Python where the signal is observed in the time spectrum, and Figure 2.2 its transformation using the FFT algorithm to the frequency domain:

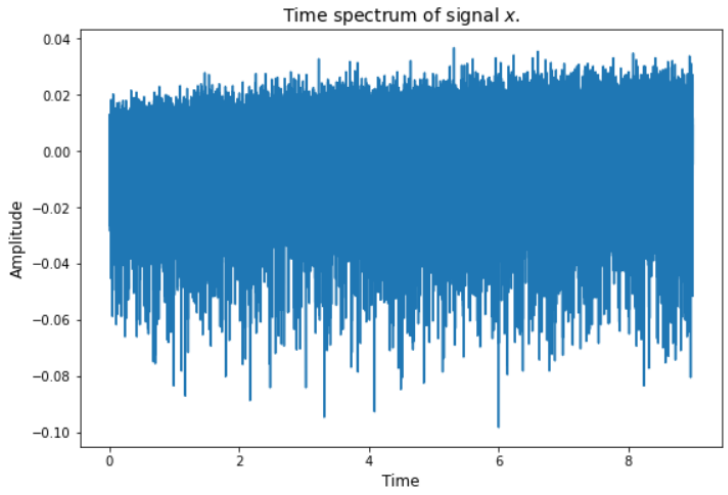


Figure 2.1: Example of the spectrum of a signal in the time domain.

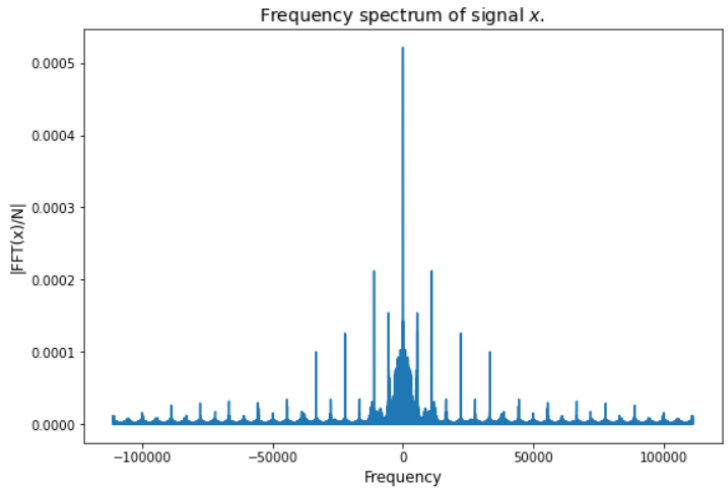


Figure 2.2: Example of the spectrum of a signal in the frequency domain.

In the previous figures we can note that the frequency spectrum gives us more information and is more visual than the time spectrum.

2.1.2 Hilbert-Huang transform

Hilbert's transform is named after the mathematician David Hilbert who introduced it in 1905 in order to solve a specific case of the Riemann-Hilbert problem.

Definition 2.1.3. *In mathematics and signal processing, the* ***Hilbert transform*** $\mathcal{H}$ *of a real function,* $s(t)$*, is obtained by the convolution of the signals* $s(t)$ *and* $1/(\pi t)$*, obtaining* $\widehat{s}(t)$*. This is:*

$$\widehat{s}(t) = \mathcal{H}\{s\}(t) = (h * s)(t) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{s(\tau)}{t - \tau}\, d\tau.$$

considering the integral of Lebesque (see, for instance, [15]). Be $s_a(t)$ *the analytical signal, it's reconstructed as:*

$$s_a(t) = s(t) + i\widehat{s}(t)$$

Now, let us introduce some new concepts in order to present the Hilbert-Huang Transform:

Definition 2.1.4. *A **mode** is a signal whose number of local extrema and zero-crossings differ at most by one. [6].*

Definition 2.1.5. *An **Intrinsic Mode Function** (see, for instance [6]) is a function verifying:*

1. *The number of extrema and the number of zero-crossings must either be equal or differ at most by one.*
2. *At all points, the mean value of the envelope defined by the local maxima and minima is zero.*

Definition 2.1.6. ***Empirical Mode Decompositon (EMD)** is an algorithm used to decompose a signal into principal "modes". Using this method any data set can be decomposed into a small number of components, which can be described as IMFs. [6], [14].*

The Hilbert-Huang Transform (HHT) analyzes vibration using the IMFs extracted using EMD. It is one of the most effective techniques for bearing failure diagnosis (see [13]). The spectrum of IMFs obtained from the HHT process can capture some defects in vibration that are not appreciated using FFT. Hilbert-Huang Transform is the simplest way to analyse changes in frequency and amplitude. It can only be used on single component signals. Using the EMD method we can break a signal down into single components and thus apply HHT. [14]

On the one hand, we have the FFT, which causes deficiencies when used directly on a damaged bearing signal. In addition, it does not correctly detect defects in inner race. On the other hand, the HHT adapts to the nature of the signal and breaks it down based on its frequencies and variations.

In Python, we can use the *PyHHT* library to aply the EMD and Hilbert Spectral Analysis tools.

Later we will see that in our case studies instead of applying these techniques, which can be computationally expensive, we will resort to machine learning techniques. However, we want to show here an example of the use of the EMD technique, showing how would be the result of decomposing a signal into its IMFs. Then, let us look at the EMD decomposition of the signal shown in Figure 2.1, for which we have used the Python library *PyHHT*:

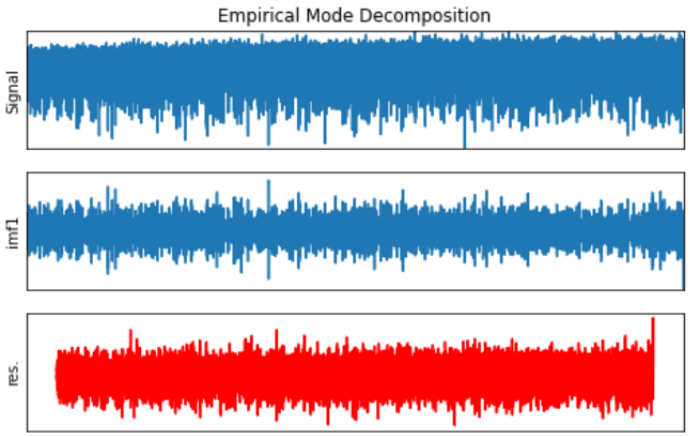


Figure 2.3: Empirical Mode Decomposition of a signal captured under variable rotational speed conditions.

Highlight that the analyzed signal was captured under variable speed conditions, and in view of the previous figure, we see that this algorithm decomposes the signal into a single IMF and the residue, which may not be very useful. Moreover, the execution time of such decomposition was 2893.07s. Let us see the decomposition of a signal captured in constant velocity conditions:

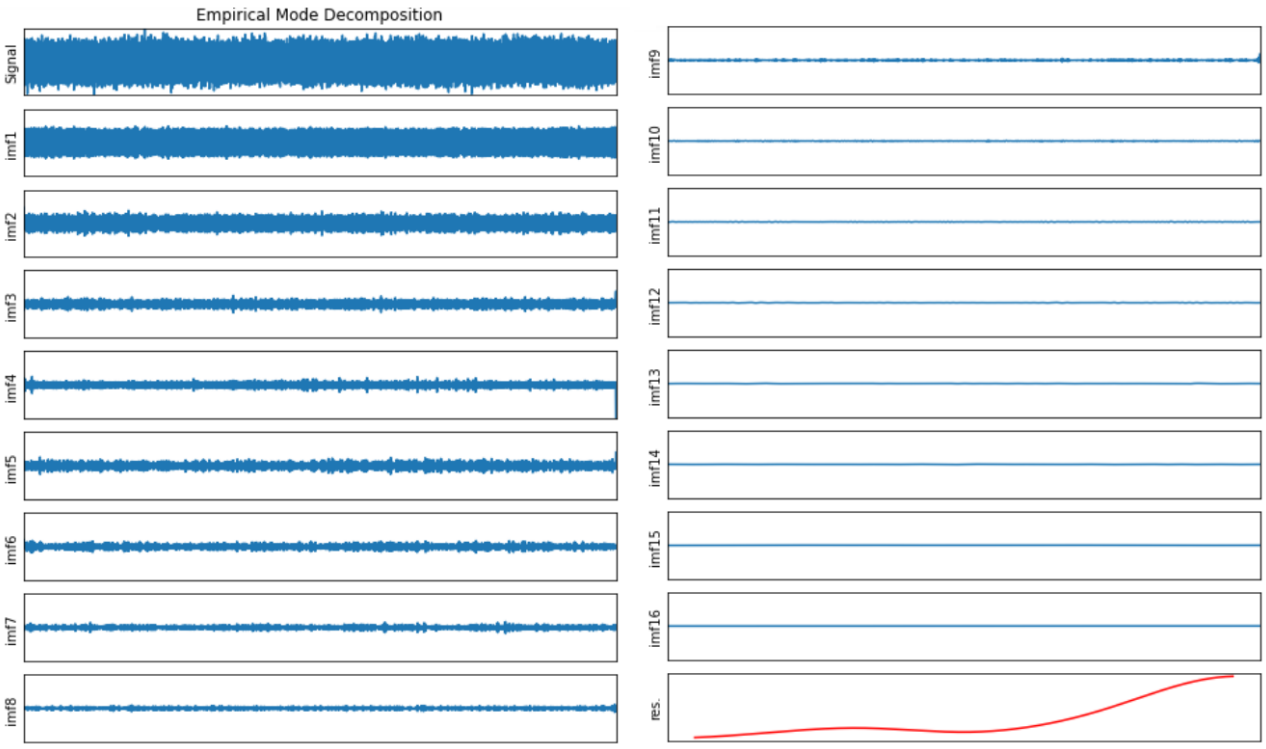


Figure 2.4: Empirical Mode Decomposition of a signal captured under constant rotational speed conditions.

In this case it is observed that the empirical mode decomposition can be more useful for the analysis, since we have a larger number of IMFs to analyze, which show increasingly smoother fluctuations. However, the execution time to find this decomposition is 314.17s, so we will not use it in the next chapter, but we will resort to different machine learning models.

2.1.3 Signal pre-processing

It is important to highlight the importance of pre-filtering the signal. By making a finite Fourier transformation, we are applying it to a signal that is repeated in infinity. In this way, if the beginning and end of our finite sample do not match, then this will be seen as an unwanted discontinuity in the signal. This is why we introduce the window functions, by which we ensure that the ends of the signal match, keeping the signal reasonably smooth:

Definition 2.1.7. ***Window functions*** *(see, for instance, [28]) are mathematical functions used in signal analysis and processing to avoid discontinuities at the beginning and end of the analyzed blocks. Be $f(x)$ a window function of size N, with $0 \leq x \leq N-1$, some common windows used in spectral analysis are:*

- *Hann:*

$$f(x) = 0.5 - 0.5\, cos\left(\frac{2\pi x}{N-1}\right)$$

- *Hamming:*

$$f(x) = 0.54 - 0.46\, cos\left(\frac{2\pi x}{N-1}\right)$$

- *Flattop:*

$$f(x) = 1 - 1.93\cos\left(\frac{2\pi x}{N-1}\right) + 1.29\cos\left(\frac{4\pi x}{N-1}\right) - 0.388\cos\left(\frac{6\pi x}{N-1}\right) + 0.032\cos\left(\frac{8\pi x}{N-1}\right)$$

- *Blackman:*

$$f(x) = 0.42 - 0.5\cos\left(\frac{2\pi x}{N-1}\right) + 0.08\cos\left(\frac{4\pi x}{N-1}\right)$$

The following figure shows a comparison of the 4 window functions exposed above, each of them applied on the same signal, that shown in Figure 2.1.

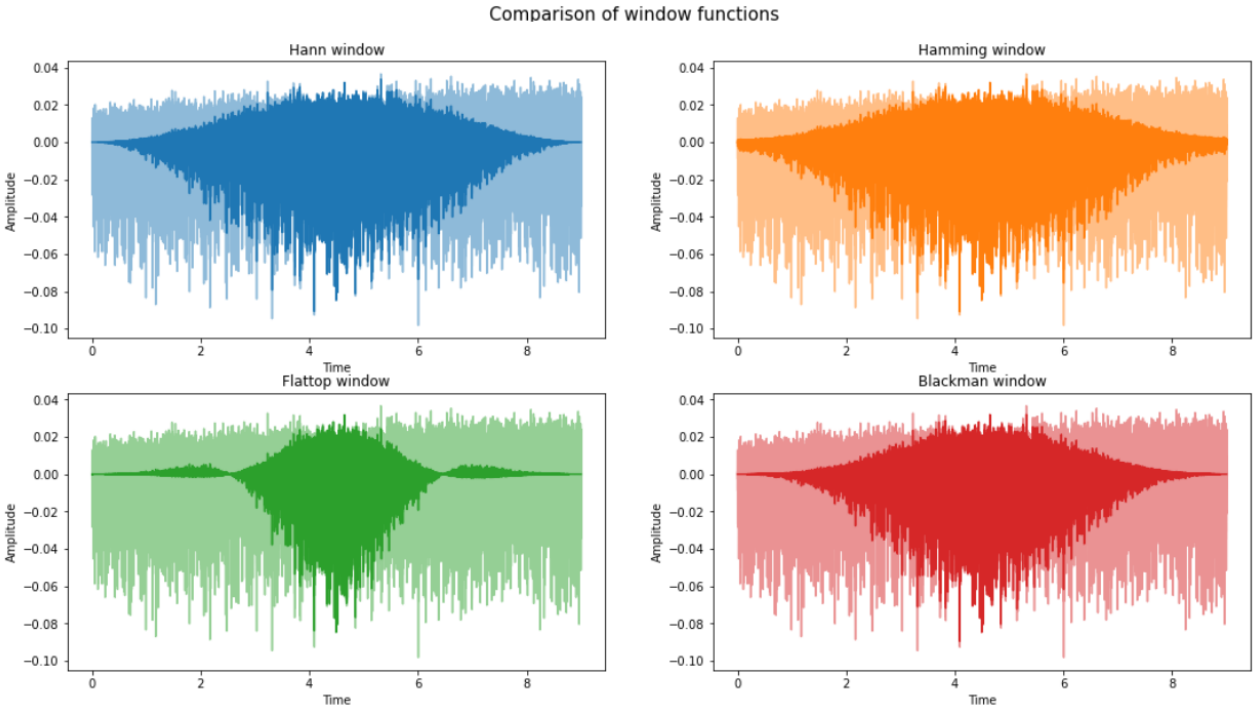


Figure 2.5: Comparison of different window functions applied on the same signal.

In addition, the signals are usually subjected to several filters before being analysed. The main purpose of filtering is usually to eliminate noise caused during the measurement, but also to eliminate phase delay or distortion. Here are some of the most commonly used in general signal processing studies (see, for example, [27]):

- According to the part of the spectrum in which they operate:
  - Highpass-filter.
  - Lowpass-filter.
- Wiener filter: is a filter proposed in the 1940s which purpose is, using statistical methods, to reduce the noise present in the input signal so that the filter output signal is as close as possible (in the mean square sense) to a desired (noise-free) signal.
- Savitzky-Golay filter: softens the input data.

These filters are implemented either in MATLAB's Signal processing Toolbox, or in Python's Scipy library, so during our study their use will be simple and we will be able to compare them to choose the optimal ones in each case. In the following figure we show a comparison of these filters, performed using Python:

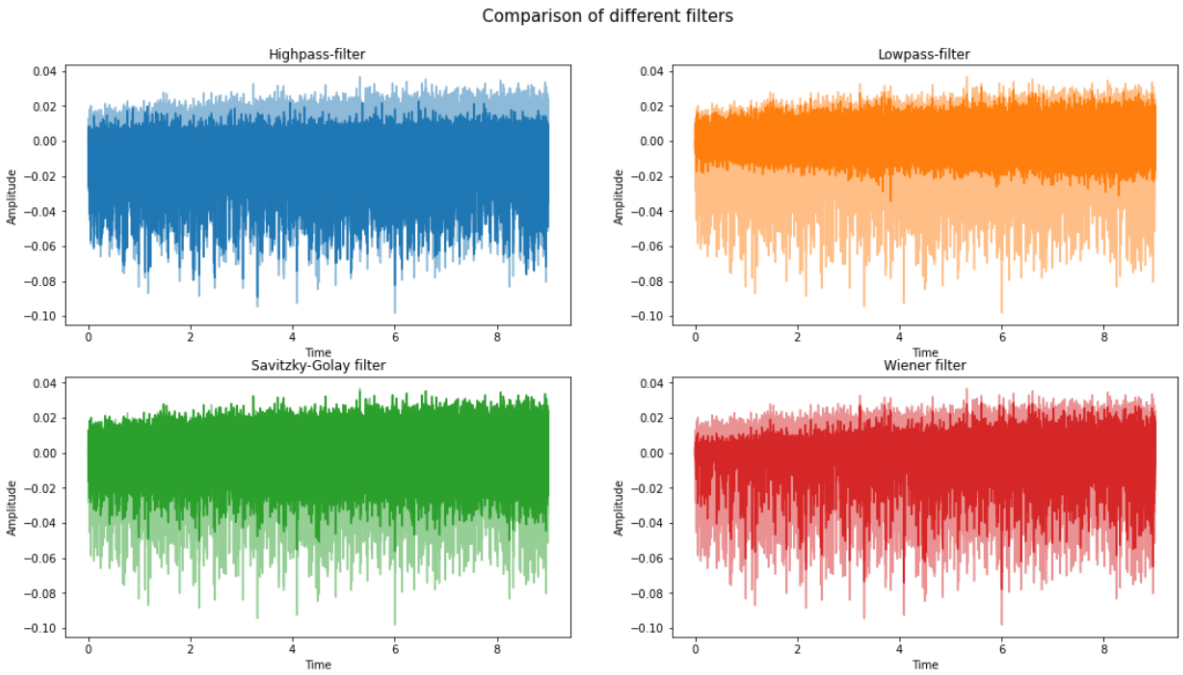


Figure 2.6: Comparison of different filters applied on the same signal.

2.2 State of the art: machine learning techniques

With the current boom in machine and deep learning techniques, their application to new sectors is becoming increasingly common. In addition, real-time analysis of the operation of machinery is very important for companies. This leads to the need to use more sophisticated techniques than the basic ones based on Fourier analysis, such as real-time pattern recognition on signals. On the other hand, the fact that around 41% of serious machine failures are caused by bearing failures (see [1]), makes it necessary to detect these failures as quickly and reliably as possible, automating the processes.

Most of the commonly proposed algorithms for detecting failure frequencies in the spectrum require that the speed of rotation of the machinery shafts be constant. In order to solve the difficulty of diagnosing failures in variable speed conditions, machine learning techniques (such as artificial neural networks) can be used. [5].

One use of artificial neural networks applied to this problem is, for example, the detection of local or global peaks in the spectrum as is exposed in [10], in order to find anomalies that may be due to machinery failure.

As we will see in the next chapter, classical machine learning models, such as random forest, SVM or kNN, can be used together with certain parameters and/or statistics to predict the type of signal. Similarly, other types of supervised and unsupervised learning techniques can be studied to try to predict the time to failure.

Chapter 3

Methodology and development

In this chapter we will study the different phases of development carried out to solve the two main problems of predictive maintenance: the detection of failures and the prediction of time to failure.

3.1 Predictive maintenance: fault detection

First of all, we are going to focus on the problem of classifying a vibration signal as a healthy or faulty signal. We are interested in being able to classify whether the failure is in inner race or outer race. Within this problem, we distinguish two cases that we will deal with separately: the rotational velocity under which the vibration signal was captured is constant or variable. We will study some different types of techniques which can be carried out in each case.

Be x the signal which we want to analyse, $\overline{x}$ its mean, σ its standard deviation and N the number of measurements. Furthermore, be $||x||_\infty$ the infinity norm[1] of x and $||x||_2$ its euclidean norm. Let us present some of the statistical parameters that we will use in the fault classification, and which are widely used in the usual techniques for detecting faults in machinery by means of machine learning techniques (see, for instance [8]):

Kurtosis:	$\frac{1}{N}\sum_{i=1}^{N}\frac{(x_i-\overline{x})^4}{\sigma^4}$	**Impulse factor:**	$\frac{\|\|x\|\|_\infty}{\frac{1}{N}\sum_{i=1}^{N}\|x_i\|}$
RMS:	$\|\|x\|\|_2$	**Margin factor:**	$\frac{\|\|x\|\|_\infty}{RMS^2}$
Skewness:	$\frac{1}{N}\sum_{i=1}^{N}\frac{(x_i-\overline{x})^3}{\sigma^3}$	**Shape factor:**	$\frac{RMS}{\frac{1}{N}\sum_{i=1}^{N}\|x_i\|}$
Peak to peak:	$x_{max}-x_{min}$	**Crest factor:**	$\frac{\|\|x\|\|_\infty}{RMS}$

Table 3.1: Parameters on the captured signals.

On the one hand, if the rotational speed of the machinery shafts is constant, in most cases the problem can be solved using classical techniques such as those described in Chapter 2: Fourier transform, Hilbert transform, etc. On the other hand, this type of techniques will not be useful for the case in which the rotational speed is variable, and to solve that problem we will have to resort to machine learning techniques.

[1] We define the infinity norm of a vector $x = (x_1, \ldots, x_N)$ as $||x||_\infty = \max_{i\in\{1,\ldots,N\}} |x_i|$.

3.1.1 Problem 1: the rotational speed of the machinery shafts is constant

In order to study the case where the signal is captured under conditions of constant rotational velocity, we will consider the case presented by Case Western Reserve University Bearing Data Center Website (see [24]).

We are going to start working by applying some of the classical mathematical techniques exposed in Chapter 2. Specifically, we are going to find the envelope sprectrum of the signal, for which it will be necessary to use the FFT and the Hilbert transform. We will start the analysis of 12 signals (4 healthy, 4 with failure in outer race and 4 with failure in inner race), captured in the following conditions:

Samples per second	**RPM**
12000	1797, 1772, 1750 or 1730
BPFO	**BPFI**
$5.4152 \cdot \frac{RPM}{60}$	$3.5848 \cdot \frac{RPM}{60}$
Diameter (inches)	**Bearing**
0.007	Drive end

Table 3.2: Parameters of the first 12 signal analyzed from CWRU.

As we have just mentioned, we are going to start applying the envelope spectrum technique to these signals, to try to identify the peaks in the spectrum with the corresponding harmonic. That is, we can expect that if the signal is healthy, the most significant peak of the envelope spectrum will correspond to RPM/60, if it has an outer fault, to BPFO, and if it has an inner fault, to BPFI. Let us present how we have implemented this technique in Python, for which the main reference has been the MATLAB function *envspectrum* (see [25]):

1. Given the original signal, we subtract its mean and find the Hilbert transform. For this we can use the function *hilbert* of *scipy.signal*.

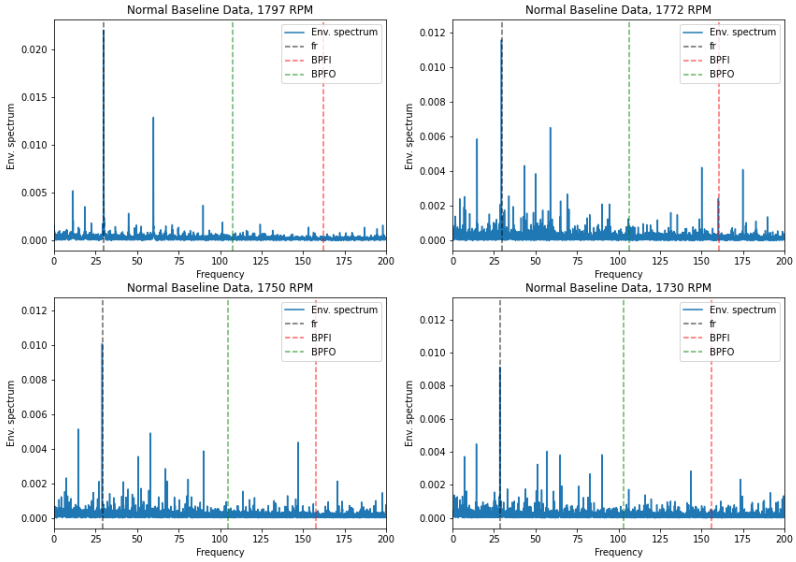
2. We find the absolute value of the previous result and again we subtract its mean (we will call it *amplitude_envelope*).

3. Given Fs the sampling frequency or number of samples per second, we calculate the frequencies corresponding to the signal, which will be as many as the length of the signal, and will go from 0 to Fs (we will use it for the graphical representation).

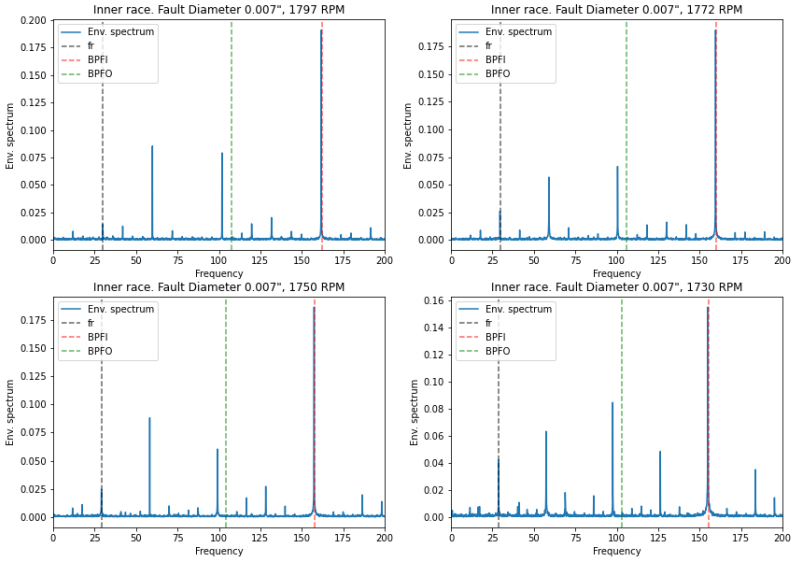
4. We find the spectrum of the signal as the absolute value of the FFT of *amplitude_envelope* (using the function *fft* of *scipy.fft*) divided by the length of the signal.

5. Finally, we compute the one-sided spectrum: compensate the amplitude for a two-sided spectrum by doubling all points except the first and the last one.

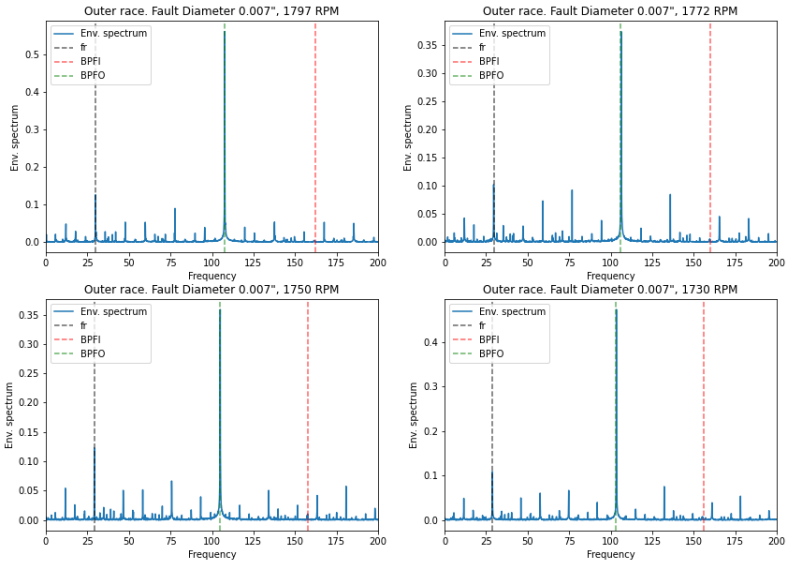
In the following figure we show the envelope spectrum of the 12 signals presented above, considering only the frequencies between 0 and 200Hz, and together with the frequency f_r (RPM/60) and the harmonics BPFO and BPFI according to their expression in Table 3.2.



(a) Case: healthy signal.



(b) Case: inner race fault.



(c) Case: outer race fault.

Figure 3.1: Envelope spectrum and harmonics of the different signals.

Figure 3.1 shows that in the case of healthy signals, the frequency where the most significant peak of the envelope spectrum is located corresponds to the value of the shaft rotational frequency ($f_r = RPM/60$) and in the case of signals with failure in outer race or inner race corresponds to the values of BPFO and BPFI given in Table 3.2 respectively.

Once we have the envelope spectrum calculated, it is easy to automate the classification by looking for the frequency where these peaks are found and comparing with the 3 harmonics mentioned above. However, it is found that in certain cases this starts to work worse as there are a greater number of significant peaks of similar values. For example, in the following figure we show the same graphical representation as in the previous case, but now varying the value of the diameter of Table 3.2, being 0.014 or 0.021 inches (note that there are no measurements of healthy signals in these conditions):

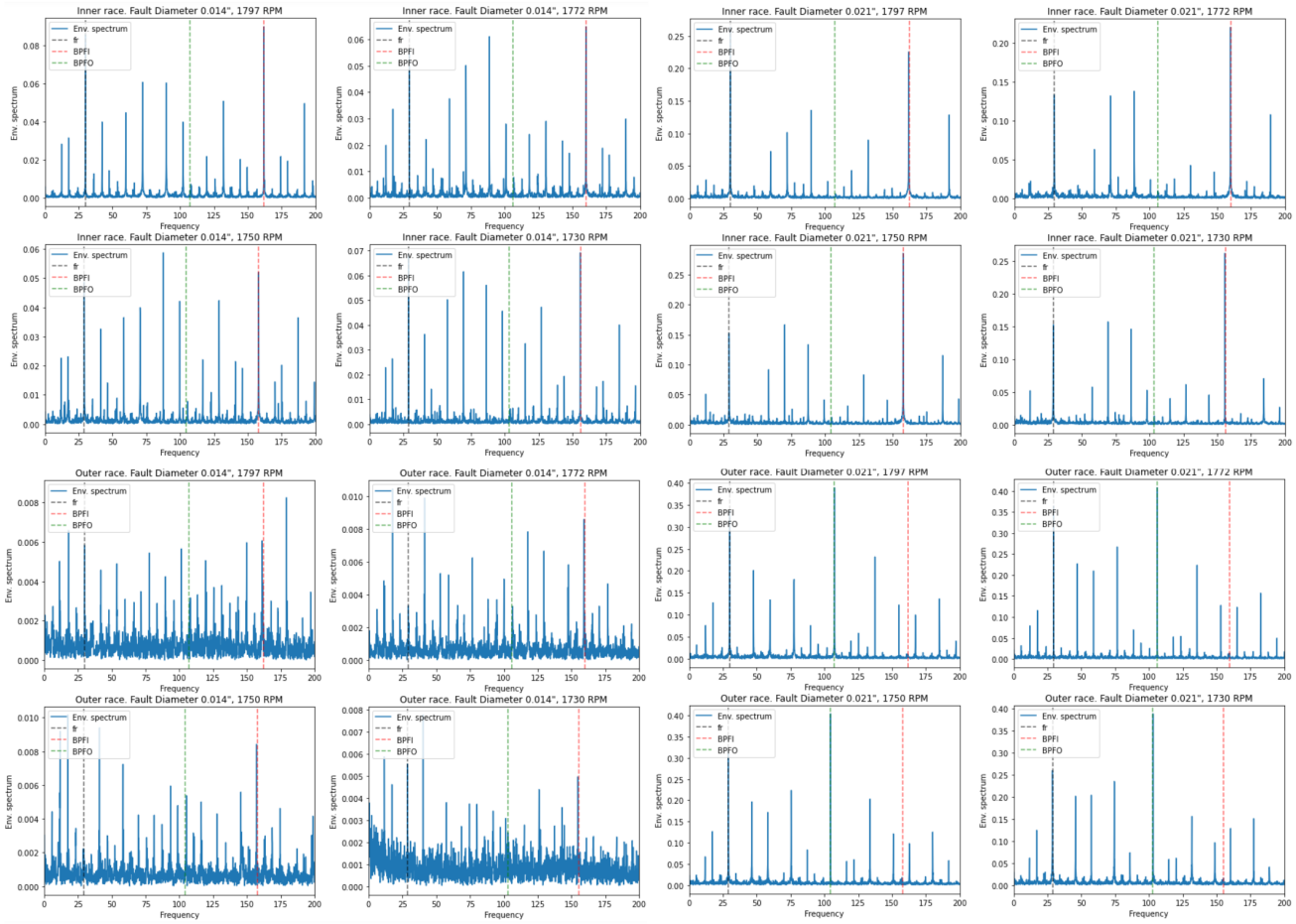


Figure 3.2: Envelope spectrum and harmonics of signals with outer race and inner race fault and diameter 0.014 or 0.021 inches.

In Figure 3.2 we can see how the envelope spectrum does not work as we would like and to classify this type of signals it would be necessary to resort previously, for example, to EMD decomposition in IMFs or to other types of processing. As this would significantly increase the computational cost, we will try to classify these kind of signals using machine learning models.

Let us note that we have 4 healthy signals, and in total 12 signals with inner race failure and 12 with outer race failure (we only consider the signals with outer race failure captured centered and we do not use the signals captured with diameter value of 0.028 inches because there are only data of signals with inner race failure in these conditions). In this way, we only have 28 signals, so we are going to resort to data augmentation techniques to obtain more data to be used in train and

test: First, for each of the 28 signals, we will find the statistics of the Table 3.1 and store them in a dataframe together with the type of signal. Now, from each of the 28 rows of this dataframe we will generate 10 new ones (which will therefore be of the same type as the initial one) by adding random Gaussian numbers to each statistic. We test this technique to adjust the selected data augmentation technique using some machine learning models. For example, we will show the results obtained in these conditions with the Random Forest and k-Nearest Neighbors models.

- **K-nearest neighbors (kNN):** We train the model by varying the value of k, and show the evolution of the accuracy for train and test:

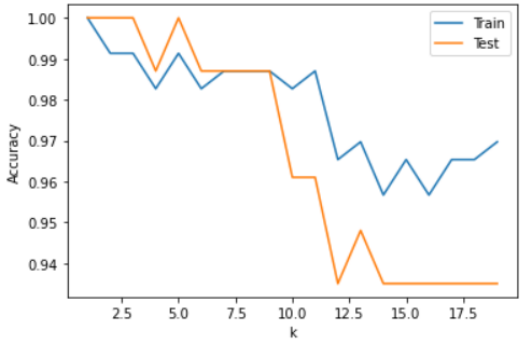


Figure 3.3: Accuracy of train and test in function of k value.

In view of the above figure, we take $k = 2$ and predict in train and in test. We obtain the following classification report:

	precision	recall	f1-score	support
Inner	1.00	1.00	1.00	33
Outer	1.00	1.00	1.00	33
Healthy	1.00	1.00	1.00	11
accuracy			1.00	77
macro avg	1.00	1.00	1.00	77
weighted avg	1.00	1.00	1.00	77

(a) Classification report for train.

	precision	recall	f1-score	support
Inner	0.98	1.00	0.99	99
Outer	1.00	0.98	0.99	99
Healthy	1.00	1.00	1.00	33
accuracy			0.99	231
macro avg	0.99	0.99	0.99	231
weighted avg	0.99	0.99	0.99	231

(b) Classification report for test.

Figure 3.4: Classification report for train and test with kNN.

- **Random forest:** We train the model by varying the number of trees, and show the evolution of the accuracy for train and test:

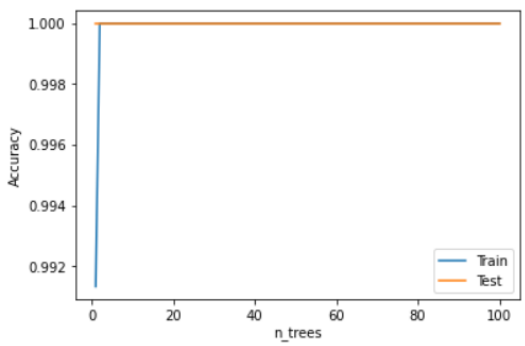


Figure 3.5: Accuracy of train and test as a function of the number of trees.

In view of the above graph, we take 20 trees and predict in train and in test. We obtain the following classification report:

	precision	recall	f1-score	support
Inner	1.00	1.00	1.00	99
Outer	1.00	1.00	1.00	99
Healthy	1.00	1.00	1.00	33
accuracy			1.00	231
macro avg	1.00	1.00	1.00	231
weighted avg	1.00	1.00	1.00	231

(a) Classification report for train.

	precision	recall	f1-score	support
Inner	1.00	1.00	1.00	33
Outer	1.00	1.00	1.00	33
Healthy	1.00	1.00	1.00	11
accuracy			1.00	77
macro avg	1.00	1.00	1.00	77
weighted avg	1.00	1.00	1.00	77

(b) Classification report for test.

Figure 3.6: Classification report for train and test with random forest.

In view of the previous results, we can conclude that with simple machine learning models, without the need for complex processing, a good classification of the signals can be obtained. When there is other data with similar characteristics, the strategy to follow would be the following:

1. Check the effectiveness of the envelope spectrum technique. If clearly identifiable peaks are observed as in Figure 3.1, use a *Python* module for classification to identify to which harmonic the frequency of the most significant peak corresponds.

2. If the above does not give good results, apply simple machine learning models such as random forest and/or kNN. Note that in a predictive maintenance project it is expected to have a large volume of data that allows to carry out these techniques without using data augmentation.

3.1.2 Problem 2: the rotational speed of the machinery shafts is variable

To study this case, the data we are going to use are obtained from [23]. The sampling time is 10 seconds, and at a frequency of 200kHz. First of all, note that we only have 12 signals of each type. However, having a frequency of 200kHz, in one second we will have 200000 samples, enough to detect the possible faults in inner and outer race. Therefore, we will decompose each of the 36 signals (of duration 10 seconds), into 10 signals of duration one second. Thus, we will have 120 healthy signals, 120 with inner race failure, and 120 with outer race failure. The following table explains how the 36 original signals have been captured:

Bearing Health condition	**Increasing speed**	**Decreasing speed**	**Increasing then decreasing speed**	**Decreasing then incressing speed**
Healthy	H-A-1, H-A-2, H-A-3	H-B-1, H-B-2, H-B-3	H-C-1, H-C-2, H-C-3	H-D-1, H-D-2, H-D-3
Inner race	I-A-1, I-A-2, I-A-3	I-B-1, I-B-2, I-B-3	I-C-1, I-C-2, I-C-3	I-D-1, I-D-2, I-D-3
Outer race	O-A-1, O-A-2, O-A-3	O-B-1, O-B-2, O-B-3	O-C-1, O-C-2, O-C-3	O-D-1, O-D-2, O-D-3

Table 3.3: Table with the codes of the signals used in this case and explaining how they have been captured. Source: [23].

Each of these 360 signals has to be processed. After finding results with certain machine learning models using the window functions and the filters exposed in Chapter 2, we are left with the Hamming window and the low pass filter as preprocessing. Let us see a pairplot with the comparison of the 8 statistics of Table 3.1 according to the type of signal

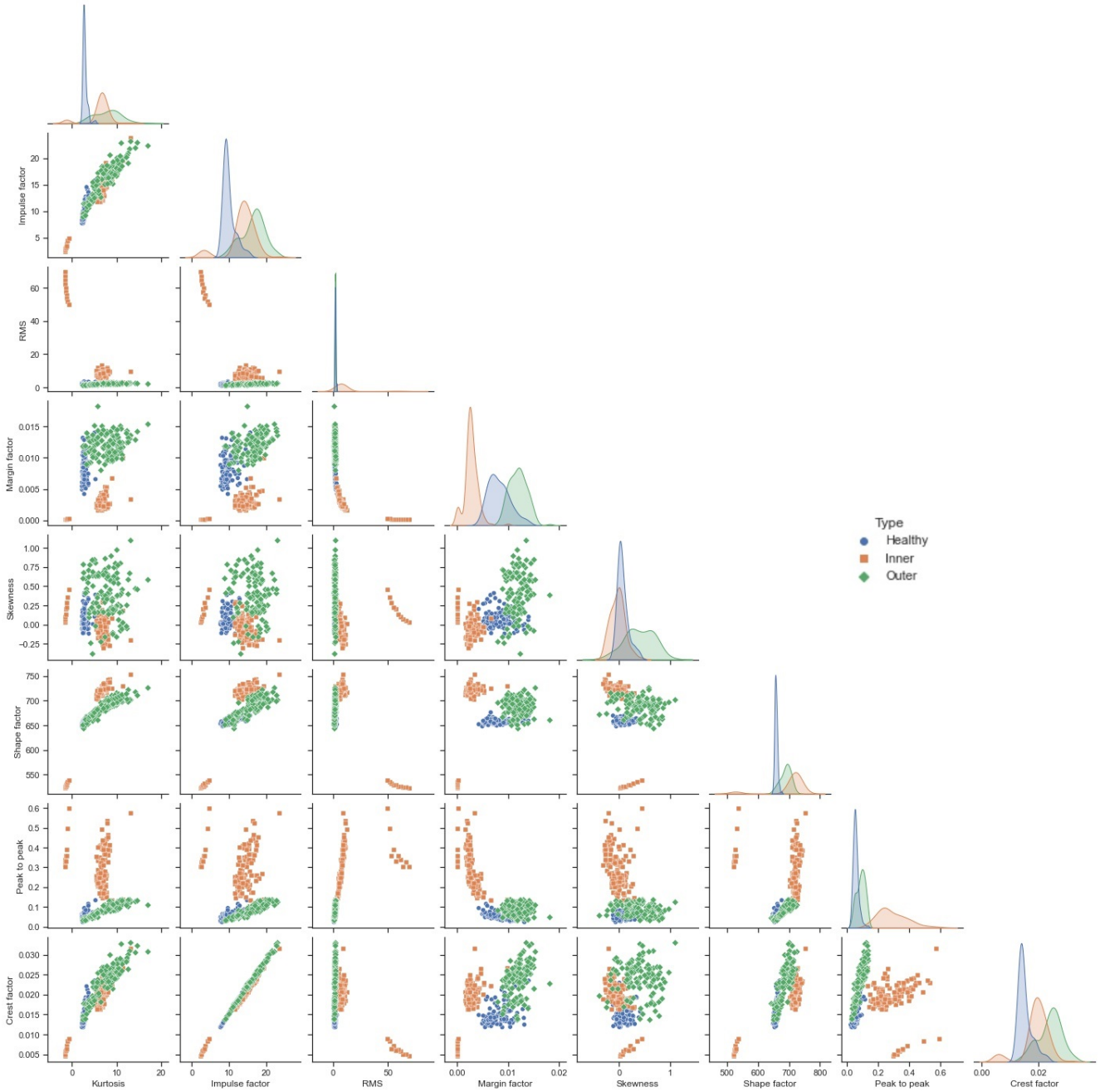


Figure 3.7: Comparison of the 8 statistics.

In the previous figure we can see how in some of the scatter plots the difference between the three types of signals is clearly visible, distinguishing the three colors in an unequivocal way. This is the case of comparing, for example, kurtosis and margin factor. In addition, the orange color, which corresponds to signals with inner race failure, is perfectly distinguishable from all the others in all the comparisons. On the other hand the most conflicting case is in differentiating whether a signal is healthy or has outer race failure (this is clearly seen in the comparison of kurtosis and shape factor, where the green and blue marks overlap). Therefore, at first glance it seems that it will not be difficult to discern signals with inner race failure from other, but it will be to differentiate healthy signals from those with outer race failure. Also note that in all the plots there are 10 orange marks that are notably different from the others. These correspond to the decomposition of a particular signal (I-A-1), for which in its time domain plot we could observe that the vibration was centered in a position different from the other signals (see Figure 3.8). It may be a signal captured erroneously, but a priori we will

work with it because the anomaly may also be caused by the failure in inner race.

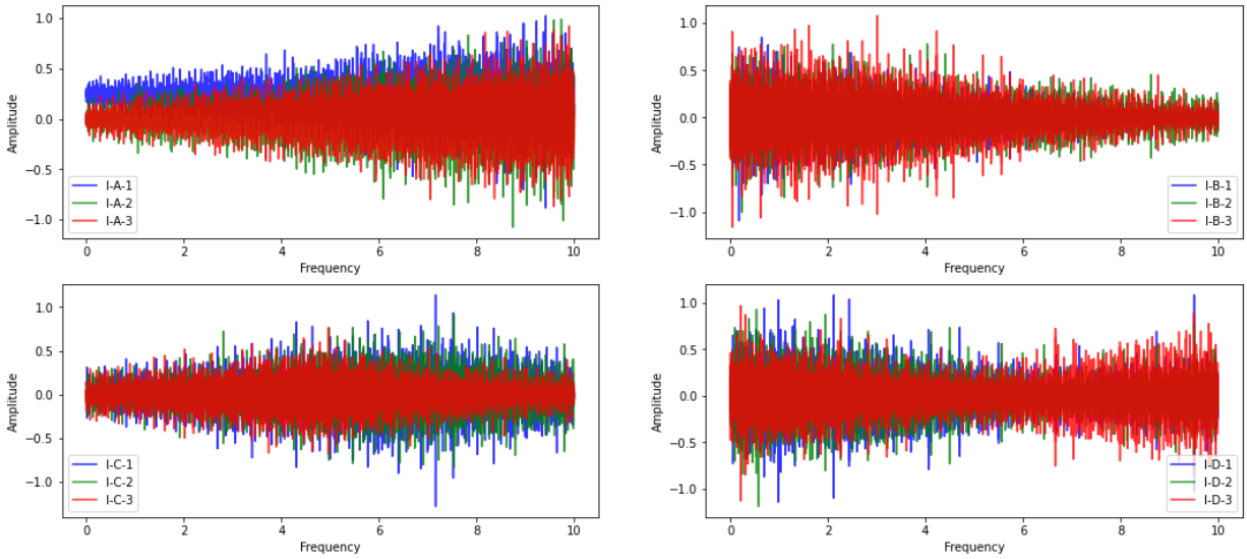


Figure 3.8: Inner race fault signals.

We are going to explain a peculiarity that we have taken into account when dividing our data into train and test sets: Let us note that given a 10-second signal we are going to decompose it into 10 one-second signals, and it could lead to overfitting to use a one-second signal in train, and the signal corresponding to the immediately following second in test. Therefore, we will do the following: given 36 initial signals, we divide them into two sets, so that 75% corresponds to the train and 25% to the test. Once we have this division, we can decompose each of these signals in 10, apply the Hamming window function, the low pass filter, and calculate the statistics of Table 3.1. In this way we will have 270 signals to train and we will use 90 as test (the input of the models will be the statistics calculated for each signal), and we will be avoiding the problem of training with a signal and that the second following this one is used to predict.

With these train and test sets we are going to train and predict using the following machine learning models:

- **Decision trees:** We will train decision trees using *DecisionTreeClassifier* from *scikit-learn* library. Let us study the accuracy graph for train and test as a function of the maximum tree depth value:

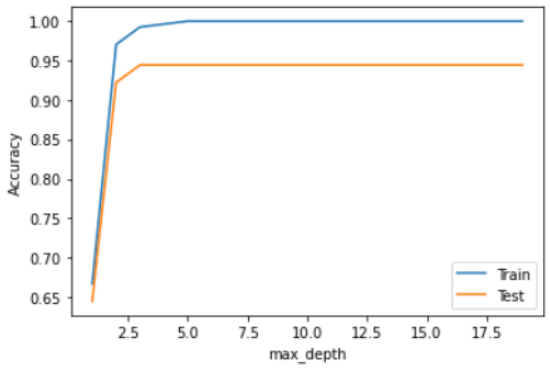


Figure 3.9: Accuracy of train and test according to the value of *max_depth*.

In view of the above figure, we will select 7 as the maximum depth, as the graph begins to

stabilize.

	precision	recall	f1-score	support
Inner	1.00	1.00	1.00	90
Outer	1.00	1.00	1.00	90
Healthy	1.00	1.00	1.00	90
accuracy			1.00	270
macro avg	1.00	1.00	1.00	270
weighted avg	1.00	1.00	1.00	270

(a) Classification report for train.

	precision	recall	f1-score	support
Inner	1.00	0.93	0.97	30
Outer	0.88	1.00	0.94	30
Healthy	0.96	0.90	0.93	30
accuracy			0.94	90
macro avg	0.95	0.94	0.94	90
weighted avg	0.95	0.94	0.94	90

(b) Classification report for test.

Figure 3.10: Classification report for train and test with decision trees.

- **Support Vector Machines (SVM):** In this case we are going to train an SVM on the training data and we will use the *rbf* kernel (Gaussian). In addition, using *GridSearchCV* we will search for the optimal parameters of C and γ, getting the following values: $C = 500$ and $\gamma = 0.01$. Note that the adjustment time of these parameters is less than 0.2s.

  The following is the classification report for train and test:

	precision	recall	f1-score	support
Inner	1.00	1.00	1.00	90
Outer	1.00	1.00	1.00	90
Healthy	1.00	1.00	1.00	90
accuracy			1.00	270
macro avg	1.00	1.00	1.00	270
weighted avg	1.00	1.00	1.00	270

(a) Classification report for train.

	precision	recall	f1-score	support
Inner	0.97	1.00	0.98	30
Outer	0.93	0.93	0.93	30
Healthy	0.97	0.93	0.95	30
accuracy			0.96	90
macro avg	0.96	0.96	0.96	90
weighted avg	0.96	0.96	0.96	90

(b) Classification report for test.

Figure 3.11: Classification report for train and test with SVM.

- **K-nearest neighbors (kNN):** We train the model by varying the value of k and show the evolution of the accuracy for train and test (note that the variables have been scaled previously):

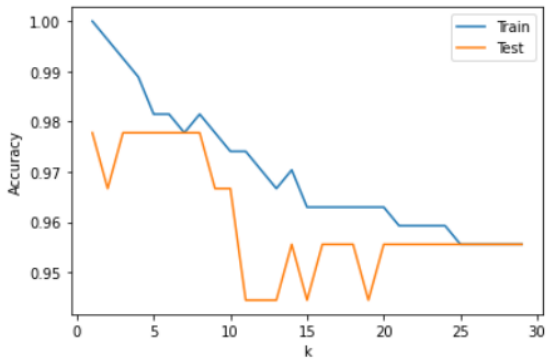


Figure 3.12: Accuracy of train and test according to k value.

In view of the above graph, we take $k = 5$ and predict in test. We obtain the following classification report for train and for test:

	precision	recall	f1-score	support
Inner	1.00	1.00	1.00	90
Outer	0.99	0.96	0.97	90
Healthy	0.96	0.99	0.97	90
accuracy			0.98	270
macro avg	0.98	0.98	0.98	270
weighted avg	0.98	0.98	0.98	270

(a) Classification report for train.

	precision	recall	f1-score	support
Inner	1.00	0.97	0.98	30
Outer	0.94	1.00	0.97	30
Healthy	1.00	0.97	0.98	30
accuracy			0.98	90
macro avg	0.98	0.98	0.98	90
weighted avg	0.98	0.98	0.98	90

(b) Classification report for test.

Figure 3.13: Classification report for train and test with kNN.

- **Random forest:** We train the model by varying the number of trees, and show the evolution of the accuracy for train and test:

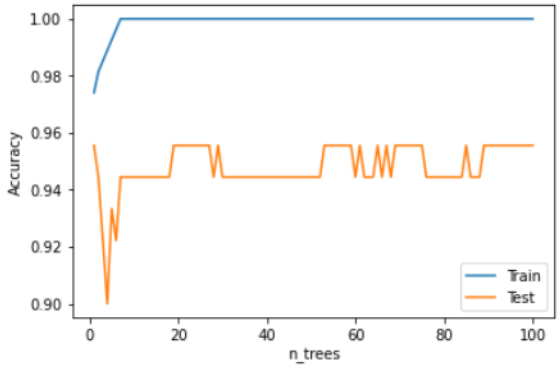


Figure 3.14: Accuracy of train and test according to the number of trees.

In view of the above graph, we take 90 trees and predict in test. We obtain the following classification report for train and for test:

	precision	recall	f1-score	support
Inner	1.00	1.00	1.00	90
Outer	1.00	1.00	1.00	90
Healthy	1.00	1.00	1.00	90
accuracy			1.00	270
macro avg	1.00	1.00	1.00	270
weighted avg	1.00	1.00	1.00	270

(a) Classification report for train.

	precision	recall	f1-score	support
Inner	1.00	0.93	0.97	30
Outer	0.88	1.00	0.94	30
Healthy	1.00	0.93	0.97	30
accuracy			0.96	90
macro avg	0.96	0.96	0.96	90
weighted avg	0.96	0.96	0.96	90

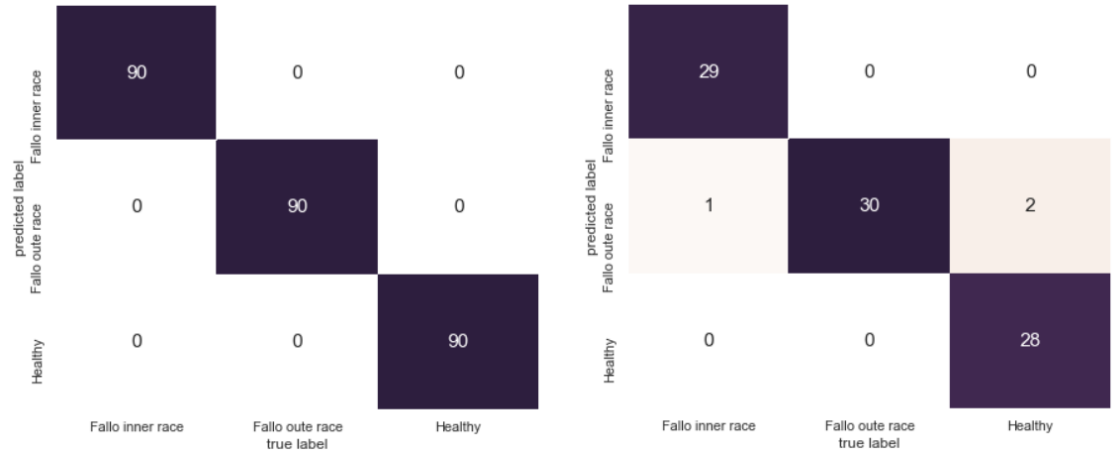
(b) Classification report for test.

Figure 3.15: Classification report for train and test with random forest.

We have studied four machine learning models and we have obtain the optimum parameters in each case. Now we are going to use the power of each model with an ensemble method. For this we have carried out two techniques:

1. **Majority voting:** Using the Python library *joblib* we save the 4 previous trained models and the scaler of the variables to be applied to the input signal in the case of kNN algorithm. Given a signal, we pass it through a function implemented in *Python* that applies the hamming window function and a low pass filter to it, and calculates the statistics of Table 3.1. The signal state

is predicted using the 4 models and the majority prediction is returned. In case of a 2-2 tie, we keep the prediction given by the random forest. Although kNN gives better results for the test set individually, we will select the prediction given by random forest because in the accuracy graph we observe more stability in this case than in the kNN case (see Figures 3.12 and 3.14). Below we show the results obtained in train and test with this method, showing the confusion matrix:

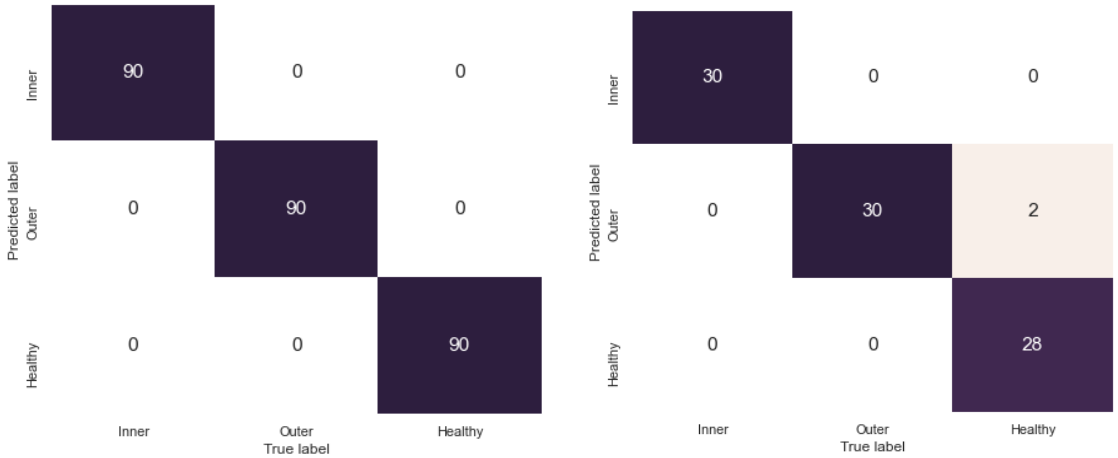


(a) Confusion matrix for the train set.

(b) Confusion matrix for the test set.

Figure 3.16: Confusion matrices for train and test sets with the method of *majority voting*.

2. **Stack of estimators:** The power of this technique comes from the fact that we use the output of each individual estimator as input of a final estimator. To do this we are going to use the function *StackingClassifier* from *scikit-learn* library considering the four machine learning models exposed previously. We obtain the following confusion matrices for the prediction on train and test sets:



(a) Confusion matrix for the train set.

(b) Confusion matrix for the test set.

Figure 3.17: Confusion matrices for train and test sets with the method of *stack of estimators*.

Finally, we would like to conclude this section by mentioning two other classical ensemble methods that we have tested in order to have a broader study for when other data with similar characteristics

are used:

- AdaBoost (adaptive boosting): we have use the method *AdaBoostClassifier* from the library *sklearn*. After testing several baseline estimators and optimizing the parameter values, we are left with *RandomForestClassifier* as the base estimator and 125 as the maximum number of estimators at which boosting is terminated. All the data in the train set and 85 out of 90 of the test data are correctly classified.

- Gradient Boosting for classification: we have use the method *GradientBoostingClassifier* from the library *sklearn*. After finding the optimal values for the number of boosting stages to perform and the lerning rate, we get that all the data in the train set and 87 out of 90 of the test data are correctly classified.

  We would like to highlight that in this case we have also studied the use of *LightGBM* library, which is a gradient boosting framework that uses tree-based learning algorithms. The improvements that lead us to consider the use of this library with respect to *sklearn* are among others, higher accuracy, increased training speed and efficiency and less memory usage. Specifically, we have used the method *LGBMClassifier* and *hyperopt* library to find the optimal parameters. However, in this particular case, the results obtained are slightly worse than those obtained with *GradientBoostingClassifier* from the library *sklearn*: all the data in the train set and 86 out of 90 of the test data are correctly classified.

3.2 Predictive maintenance: time to failure

In this particular predictive maintenance problem, the objective will be to predict the time remaining until a bearing will show a failure. To carry out our baseline study, we will use a data set called "Bearing Data Set" which is provided by the Center for Intelligent Maintenance Systems (IMS), University of Cincinnati, and is available in the NASA Data Repository ([26]).

Note that each data set consists of individual files each one with 20480 points and the sampling rate set at 20 kHz. The rotation speed was kept constant at 2000 RPM. In the *README* file of this data, we obtain the following information on the three test sets included on this data set:

- ***Test 1:*** consists of 2156 files, in each one we have data captured during 10 minutes, except in the first 43, where they were captured every 5 minutes.
  - **Recording Duration:** October 22, 2003 12:06:24 to November 25, 2003 23:39:56.
  - **Channel Arrangement:** Bearing 1: channels 1 and 2. Bearing 2: channels 3 and 4. Bearing 3: channels 5 and 6. Bearing 4: channels 7 and 8.
  - **Description:** At the end of the test-to-failure experiment, inner race defect occurred in bearing 3 and roller element defect in bearing 4.

  In our case we will not use bearing 4, since the type of faults we are interested in detecting are inner race and outer race faults.

- ***Test 2:*** consists of 984 files and in each one we have data captured during 10 minutes.
  - **Duración:** February 12, 2004 10:32:39 to February 19, 2004 06:22:39.
  - **:** Bearing 1: channel 1. Bearing 2: channel 2. Bearing 3: channel 3. Bearing 4: channel 4.
  - **Descripcción:** At the end of the test-to-failure experiment, outer race failure occurred in bearing 1.

- ***Test 3:*** consists of 6324 files and in each one we have data captured during 10 minutes.
  - **Recording Duration:** March 4, 2004 09:27:46 to April 18, 2004 02:42:55.
  - **Channel Arrangement:** Bearing 1: channel 1. Bearing 2: channel 2. Bearing 3: channel 3. Bearing 4: channel 4.
  - **Description:** At the end of the test-to-failure experiment, outer race failure occurred in bearing 3.

  Note that in this case in the data set there is the number of files previously commented (6324), while in the *README* file it was indicated that there were 4448, and that data was taken until April 4, 2004 19:01:57. Surely this variation is due to the fact that more data was added after creating the *README* file of the data set.

Figure 3.18 shows the time evolution of the signals of the six channels of test 1 (remember that we are not going to use channels 7 and 8), Figure 3.19 shows that of the four channels of test 2 and Figure 3.20 that of the four channels of test 3:

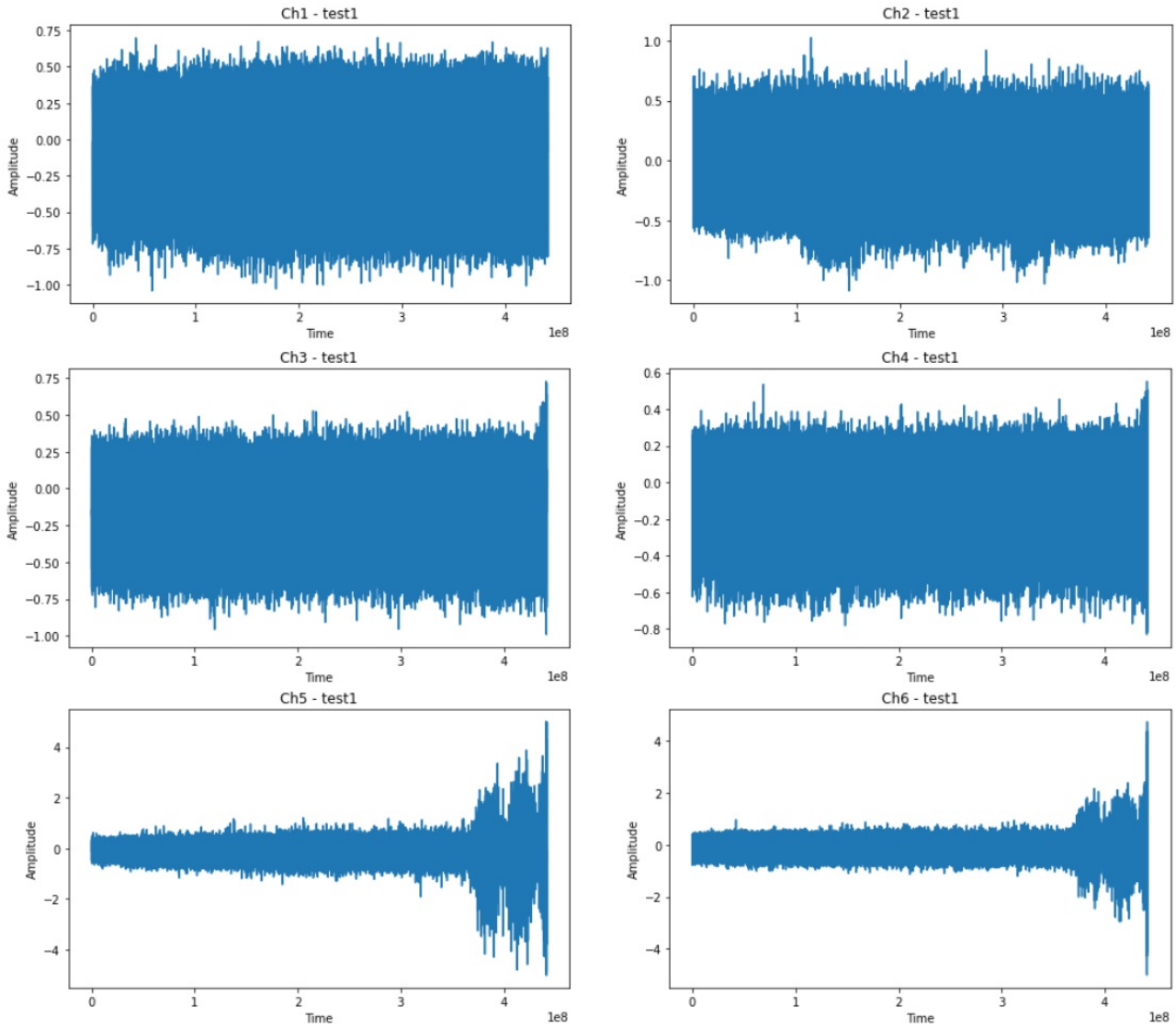


Figure 3.18: Signals from channels 1 to 6 of test 1.

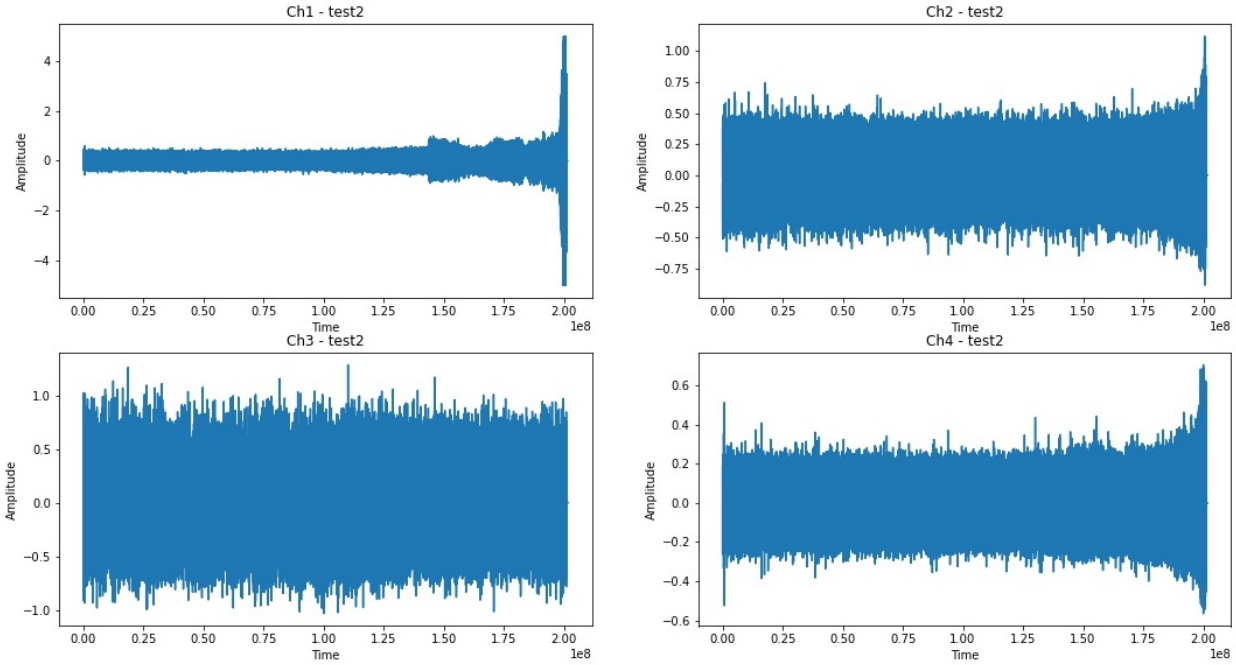


Figure 3.19: Signals from channels 1 to 4 of test 2.

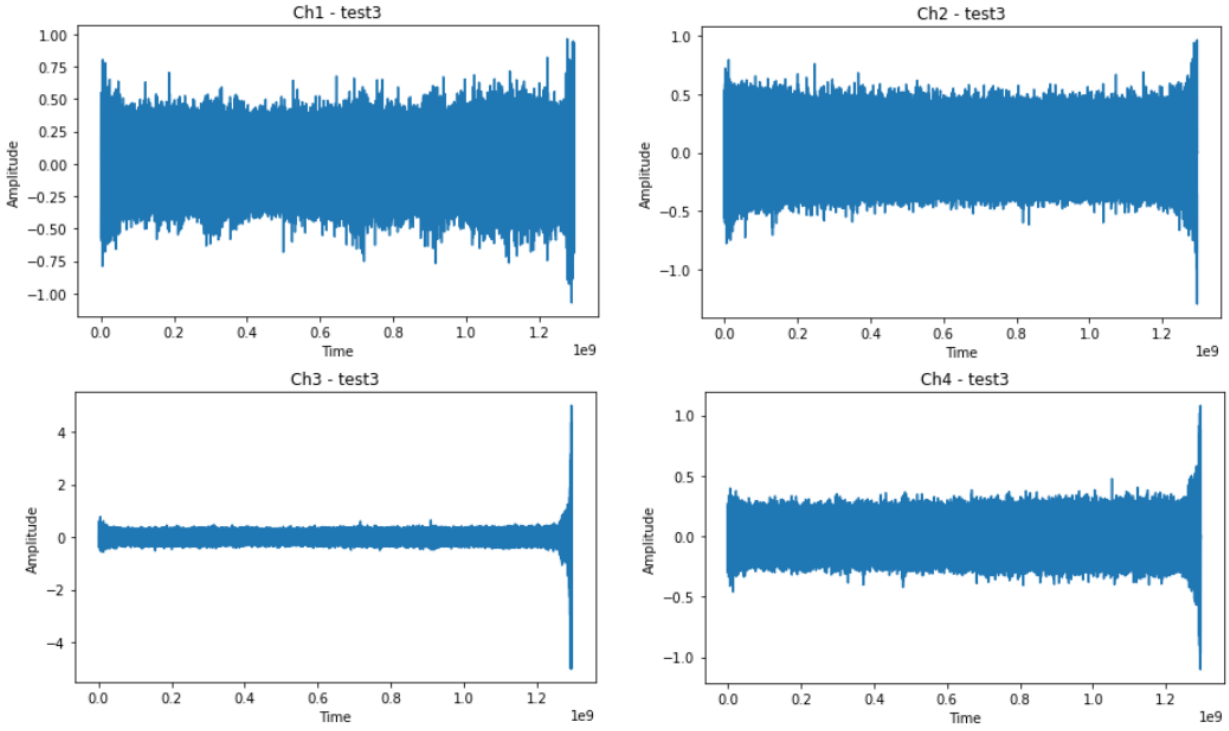


Figure 3.20: Signals from channels 1 to 4 of test 3.

Visually, we can detect the onset of anomalies in the figure which shows the signal of channel 1 of test 2 with respect to the signals of the rest of the channels of the same test (see Figure 3.19), the same applies to Channel 3, test 3 (see Figure 3.20) and to Channels 5 and 6, test 1 (see Figure 3.18). However, we are interested in being able to perform this detection automatically and without using our visual interpretation and the bias that this entails.

We therefore have signals of 14 different channels, which in 4 cases end up presenting a failure at the end of the capture. We are interested in being able to detect early the beginning of anomalies that could indicate that a failure is going to occur. To do this, first of all, for each file of each channel, which contains either 10-minute or 5-minute intervals, we will calculate the minimum, the maximum and the 8 statistics of the table 3.1.

We will focus on the four signals that we know (thanks to expert knowledge) that end up failing, and we consider all the others to be healthy signals at all times. Let us present some techniques to perform the task of anomaly detection:

3.2.1 Anomaly detection using Principal Component Analysis (PCA)

Numerous papers in the literature use dimensionality reduction techniques such as PCAs for anomaly detection (see, for instance, [3] and [4]), so let us begin by briefly recalling the concept of PCA:

Definition 3.2.1. ***Principal Component Analysis (PCA)*** *is a dimensionality reduction technique which was developed by Pearson (1901) and Hotelling (1933). PCA transforms the initial data into a new coordinate system such that the new set of variables are linear functions of the original variables, uncorrelated, and the largest variance of any projection of the data is in the first coordinate. This is achieved by calculating the covariance matrix for the entire data set. The eigenvectors and eigenvalues of the covariance matrix are then computed and ordered according to decreasing eigenvalue. [19].*

PCA obtains a set of r orthogonal directions $Pr = [p_1, \ldots, p_r]$ that maximize the variance of the projected data $y_i = P_r \cdot x_i$.

Anomaly detection is a highly developed area in recent years, as it can encompass problems of different types. While in our case an anomaly could be caused by an incipient bearing failure, in other areas, such as cybersecurity, an anomaly could be related to a cyber-attack attempt.

We are going to use PCA to detect anomalies in the signals that we know that at the end of the test present some kind of failure. The main idea that we are going to apply is the following:

1. Given an initial set of features from a healthy part of the signal, we will apply PCA to identify the subspace in which the healthy data lies. Some studies show the importance of training this type of techniques on a healthy part of the signal, where we assume that there are no anomalies. In our case we will assume that the first quarter signal is healthy.

2. Use inverse transform on the PCA-transformed data to reconstruct the original data, which allows to measure the reconstruction error. The assumption is that, if PCA is capable of correctly modeling the linear subspace in which the healthy data lies, that the reconstruction of healthy data will incur in a low reconstruction error while the anomalous data will lead to larger reconstruction errors.

3. Once the reconstruction error has been found, we will present certain criteria in order to classify these into normal or anomalies.

Since the search for the principal components will be carried out using the observations of the healthy part of the signal, it will be the observations closest to the average that will be better projected and consequently better reconstructed. The anomalous observations, on the contrary, will be badly projected and their reconstruction will be worse. In this way, we can use the reconstruction error to locate the beginning of the anomalies in our signals.

We are going to perform this process for the 4 signals of the channels in which we know that at the end of the test a failure occurs: channel 1 of test 2, channel 3 of test 3, and channels 5 and 6 of test 1. In the four cases we are going to fit PCA using the first quarter signal. The results obtained are shown below:

- **Channel 1, test 2.** Let us remember again that we apply PCAs on the first quarter of the signal since we assume that in this part there will be no anomalies yet. In view of the following figure, we note that it is enough to take 6 main components to have 99% of the variance explained. We will set this value to be used in the four cases studied.

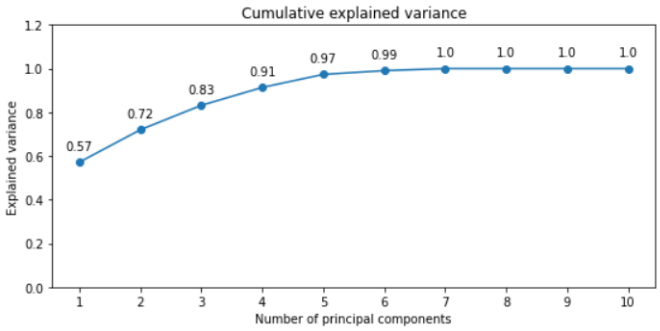


Figure 3.21: Cumulative explained variance using PCA in channel 1, test 2.

In the following figure we can see the distribution of the reconstruction errors for the part we used for the adjustment (the first quarter of the signal, our train), and for the complete signal:

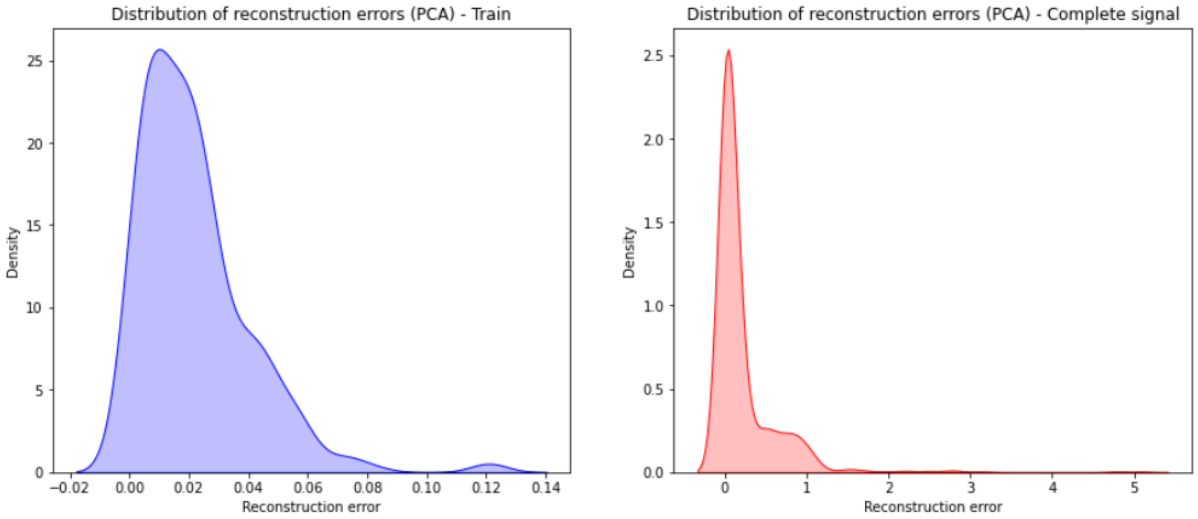


Figure 3.22: Distribution of the reconstruction errors: train and complete signal (channel 1, test 2).

In this case, specifically in the distribution of the reconstruction errors of the complete signal, a clear bend is located that separates a Gaussian clearly differentiated from the rest. Intuitively, the Gaussian closest to zero will correspond to the healthy part, since it is the area where the reconstruction error is lower, while we can intuitively consider that the part after the bend of the histogram may correspond to anomalous parts of the signal, since the reconstruction error is higher. Therefore, as in this case there is a clear bend in the histogram (see Figure 3.22), we can model the error distribution as a mixture of Gaussians (GMM[2]) to automatically detect the

[2]A Gaussian mixture model is a probabilistic model in which observations are considered to follow a probability distribution formed by the combination of multiple normal distributions. (See, for instance, [16]).

beginning of the anomalies. To accomplish this task we will use *GaussianMixture* from *sklearn*.

We fit the model *GaussianMixture* for all the reconstruction errors except those corresponding to the first quarter signal, since we have assumed that these will not present any anomalies. We considered two mixture components, and we also predict for all the reconstruction errors except those corresponding to the first quarter signal (these are assumed to be normal data). As we have considered two components to fit the model, we will consider as anomalous the data that are classified as the Gaussian with mean farther from 0. The average of each of the two Gaussians obtained by the model is 0.06547723 and 0.73872314. Let us see in the following figure how the reconstruction errors of the complete signal except the first quarter are distributed, and the values of the means given by *GaussianMixture*:

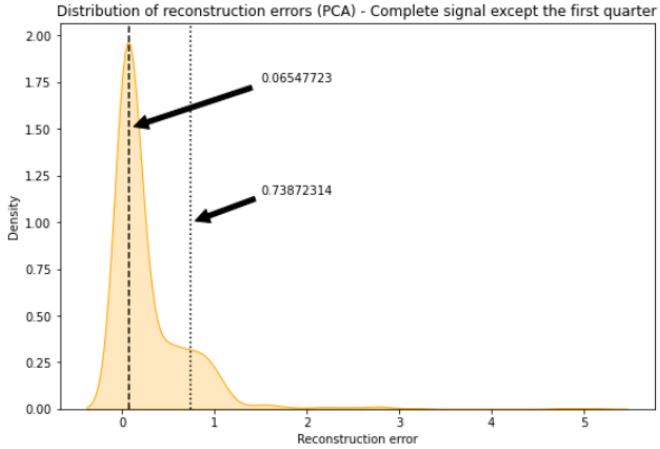


Figure 3.23: Distribution of the reconstruction errors (PCA) of the complete signal except the first quarter (channel 1, test 2).

In the following figure we show, in red the reconstruction errors that we consider anomalous with the previous criterion, and in green those that we consider normal.

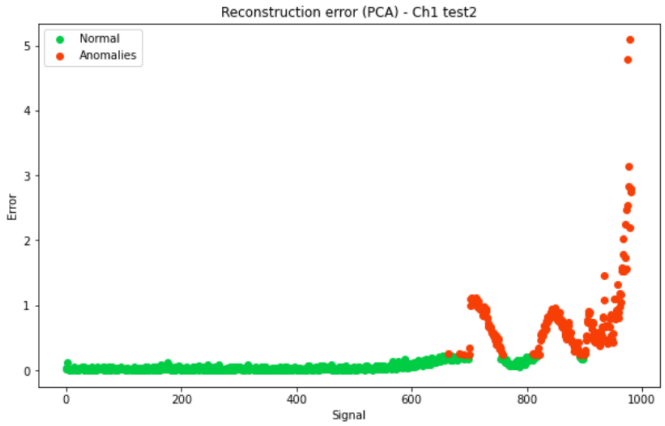


Figure 3.24: Reconstruction error and classification of anomalies (PCA - GMM), channel 1, test 2.

However, this approximation may be too strict in some cases, taking into account that each reconstruction corresponds to 10-minute intervals. A criterion that could be more reasonable is to consider that the anomalies begin when 6 consecutive reconstructions (one hour of signal caption) are classified as anomalous with the previous criterion. Once the beginning of anomalies is located, all subsequent data will be considered anomalies until the end of the signal. We take

this criterion because in periods of 10 minutes there may be a punctual problem that does not continue in time, but if during a full hour anomalies are detected, it may be due to a more serious error. Using this approximation we obtain the following error classification for each 10-minute signal:

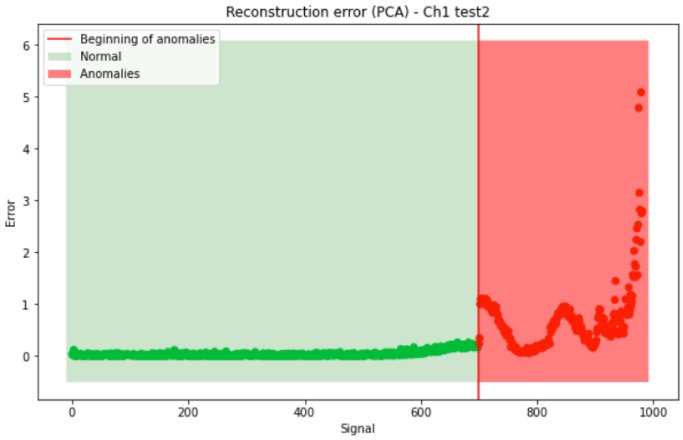


Figure 3.25: Reconstruction error and beginning of anomalies (PCA - GMM), channel 1, test 2.

Finally, another criterion that we are going to use in this case to detect outliers will be to use the Z-Scores. We define **Z-Score** as the number of standard deviations away from the mean that a certain data point is. Be X be the reconstruction errors, $\bar{X}_{train}$ the mean of the train errors, and σ_{train} the standard deviation of the train errors, then:

$$z = \frac{X - \bar{X}_{train}}{\sigma_{train}} \tag{3.1}$$

Thus, the further the value of z_i ($i \in \{1, \ldots, N\}$, with N the length of X) is from 0, the greater the probability that it is an anomalous value. A commonly used threshold is 3, that is, if the value of z is greater than 3 or less than -3, we will consider it anomalous. Using this criterion and 3 as a threshold, we obtain the classification of the reconstruction errors given in Figure 3.26, and in Figure 3.27 the classification considering that the anomalies begin when the errors of 6 consecutive reconstructions exceed the established threshold:

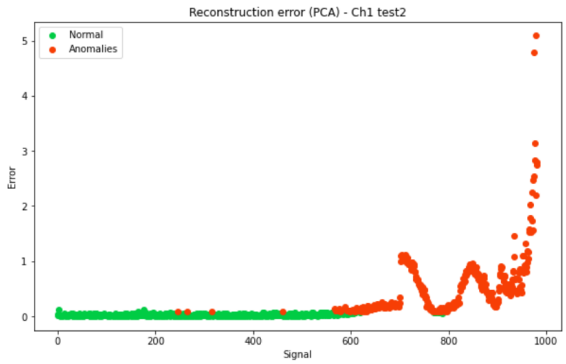


Figure 3.26: Reconstruction error and classification of anomalies (PCA - Z-Scores), channel 1, test 2.

Figure 3.27: Reconstruction error and beginning of anomalies (PCA - Z-Scores), channel 1, test 2.

- **Channel 3, test 3.** As in the previous case, when we apply PCA on the first quarter of the signal, taking 6 main components we get 99% of the variance explained. In Figure 3.28 we can see

the distribution of the reconstruction errors of the complete signal except the first quarter, and the mean of each of the two Gaussians obtained with the *GaussianMixture* model (0.01833045 and 0.9274821 respectively).

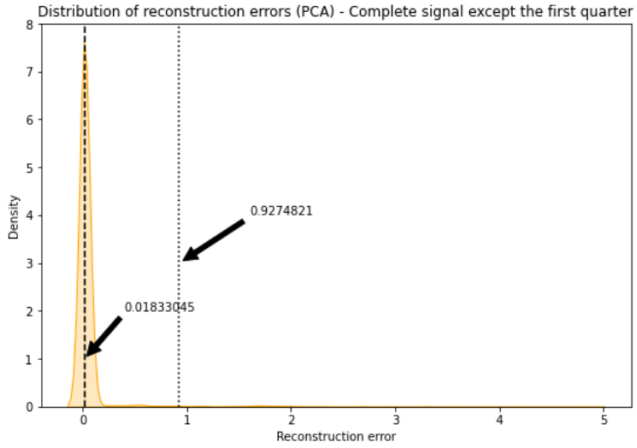


Figure 3.28: Distribution of the reconstruction errors (PCA) of the complete signal except the first quarter (channel 3, test 3).

Again, as in the previous case, we consider anomalous the data classified as those following the Gaussian distribution farthest from 0. Using this criterion, we obtain the classification of the reconstruction errors given in Figure 3.29. On the other hand, if we locate the beginning of anomalies as the first moment when 6 errors in a row are classified as anomalous, we get the classification of Figure 3.30.

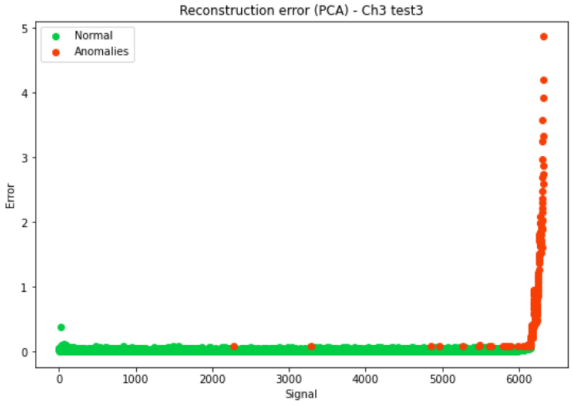


Figure 3.29: Reconstruction error and classification of anomalies (PCA - GMM), channel 3, test 3.

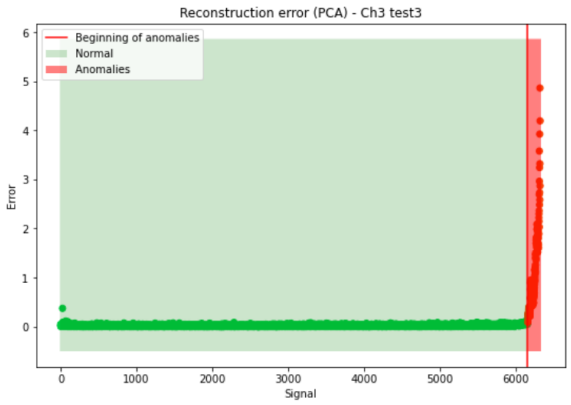


Figure 3.30: Reconstruction error and beginning of anomalies (PCA - GMM), channel 3, test 3.

Finally, we will consider again the Z-Scores with a threshold of 3. In Figure 3.31 we show the classification of the errors into normal and anomalous, and in Figure 3.32, the classification after locating the onset of anomalies as the time at which 6 consecutive reconstruction errors are classified as outliers with this criterion:

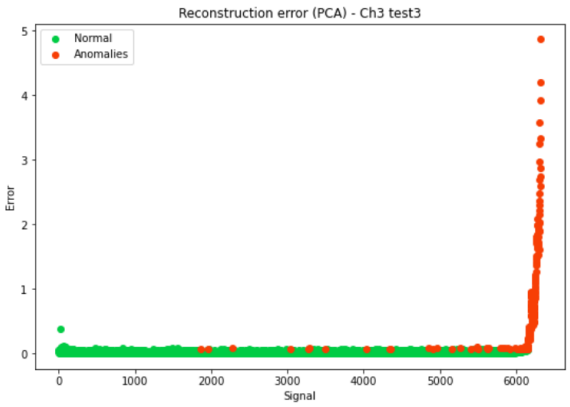


Figure 3.31: Reconstruction error and classification of anomalies (PCA - Z-Scores), channel 3, test 3.

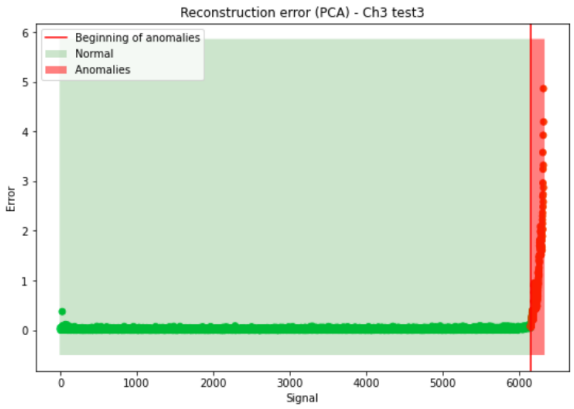


Figure 3.32: Reconstruction error and beginning of anomalies (PCA - Z-Scores), channel 3, test 3.

- **Channel 5, test 1.** In the same way, we are going to repeat the previous analysis for channel 5 of test 1, which ended up presenting a failure in outer race. Let us apply PCA on the first quarter of the signal. Note that in this case the percentage of variance explained is slightly different, and taking 6 main components we get 100% of the variance explained:

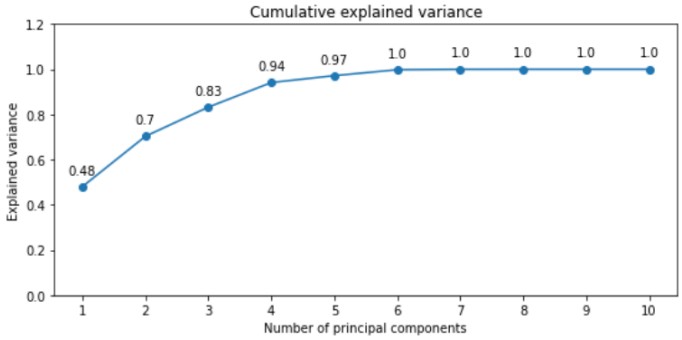


Figure 3.33: Cumulative explained variance using PCA in channel 5, test 1.

The following figure shows the distribution of the reconstruction errors for the complete signal except the first quarter, and the average of each of the two Gaussians obtained by the Gaussian Mixture model (0.039993 and 0.58153622 respectively).

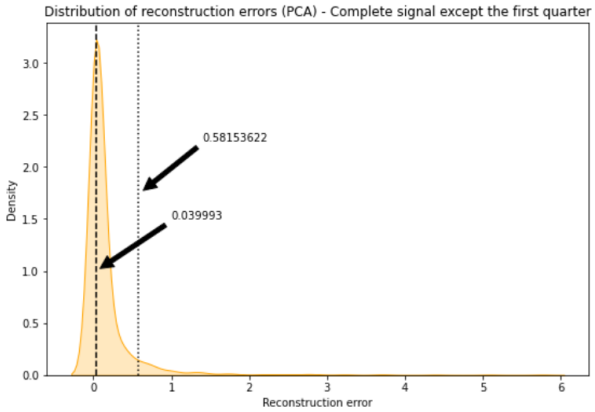


Figure 3.34: Distribution of the reconstruction errors (PCA) of the complete signal except the first quarter (channel 5, test 1).

Again, we consider anomalous the data classified as those following the Gaussian distribution farthest from 0. Then we obtain the classification of the reconstruction errors given in Figure 3.35. If we locate the beginning of anomalies as the first moment when 6 errors in a row are classified as anomalous, we get the classification of Figure 3.36.

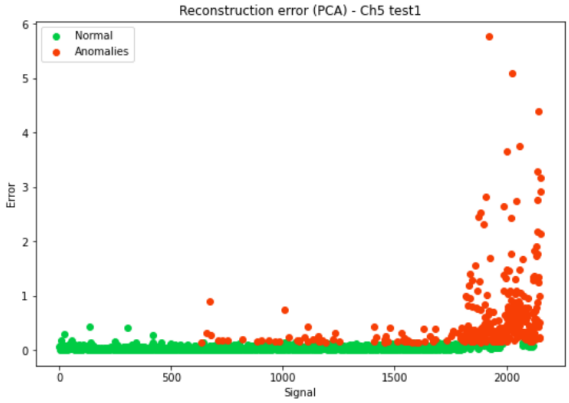


Figure 3.35: Reconstruction error and classification of anomalies (PCA - GMM), channel 5, test 1.

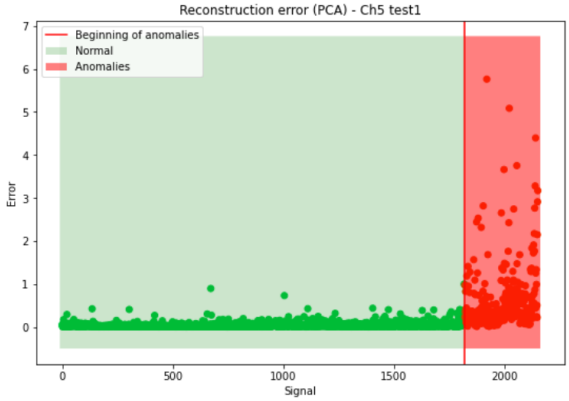


Figure 3.36: Reconstruction error and beginning of anomalies (PCA - GMM), channel 5, test 1.

Finally, let us use the Z-Scores with a threshold of 3. In Figure 3.37 we show the classification of the errors into normal and anomalous, and in Figure 3.38, the classification after locating the beginning of anomalies as the first time at which 6 consecutive reconstruction errors are classified as outliers:

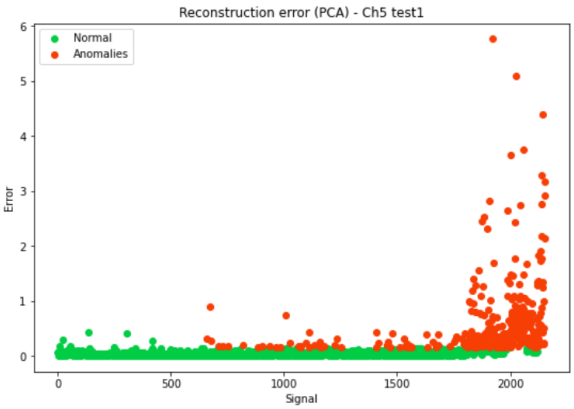


Figure 3.37: Reconstruction error and classification of anomalies (PCA - Z-Scores), channel 5, test 1.

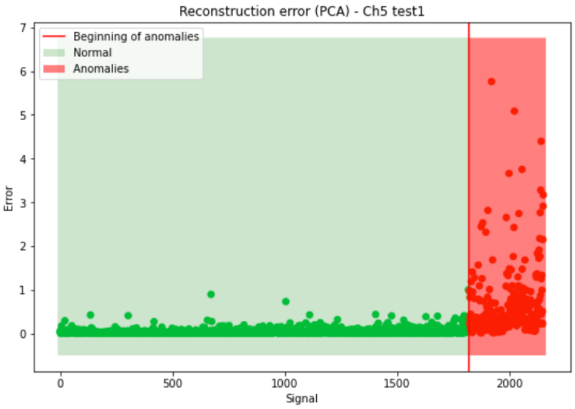


Figure 3.38: Reconstruction error and beginning of anomalies (PCA - Z-Scores), channel 5, test 1.

- **Channel 6, test 1.** We conclude this section on outlier detection using PCA with channel 6 of test 1. As in the case of channel 5 of this same test, this channel ends up presenting a failure in outer race. In this case, applying PCAs on the first quarter of the signal, taking 6 main components we get 100% of the variance explained.

In this case, the mean of the two Gaussians obtained by the Gaussian Mixture Model are 0.02115857 and 0.417237842 recpectively. Once again we consider anomalous the data classified as those following the Gaussian distribution farthest from 0, in this case, that which is centered in 0.41724. With this criterion we obtain the classification of the reconstruction errors given in

Figure 3.39. In Figure 3.40 we show the classification after locating the beginning of anomalies as the first time at which 6 consecutive reconstruction errors are classified as outliers.

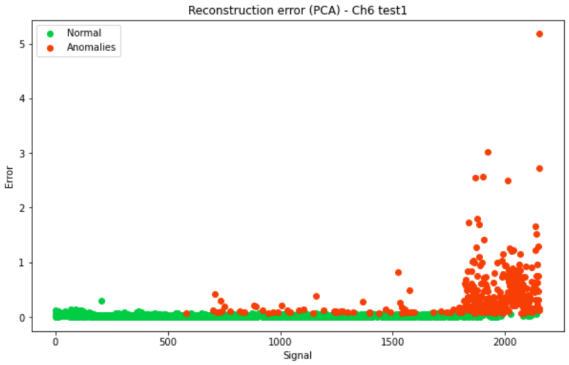


Figure 3.39: Reconstruction error and classification of anomalies (PCA - GMM), channel 6, test 1.

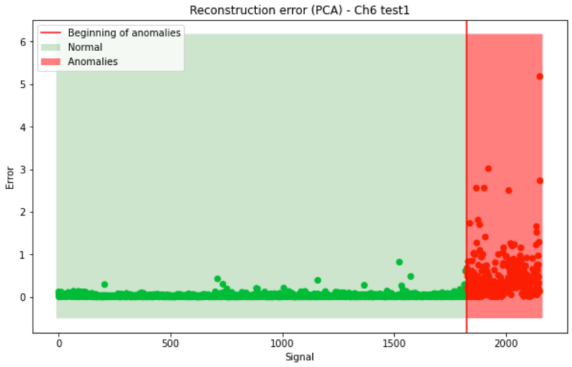


Figure 3.40: Reconstruction error and beginning of anomalies (PCA - GMM), channel 6, test 1.

Finally, let us use the Z-Scores with a threshold of 3. Figure 3.41 shows the classification of the errors into normal and anomalous with this criterion, and Figure 3.42 the classification after locating the beginning of anomalies as the first time at which 6 consecutive reconstruction errors are classified as anomalies:

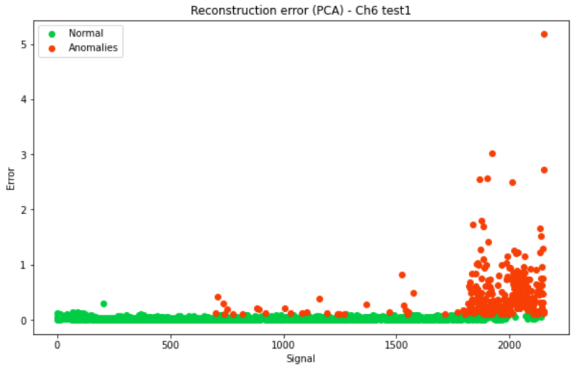


Figure 3.41: Reconstruction error and classification of anomalies (PCA - Z-Scores), channel 6, test 1.

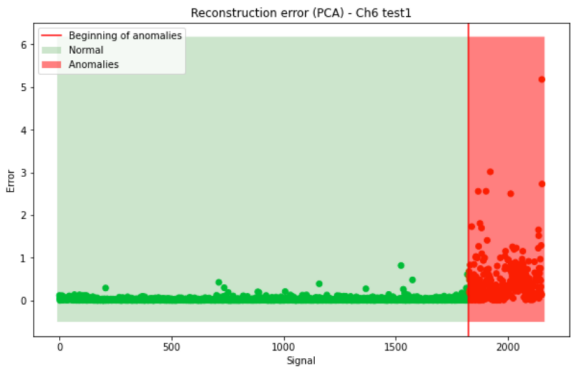


Figure 3.42: Reconstruction error and beginning of anomalies (PCA - Z-Scores), channel 6, test 1.

3.2.2 Anomaly detection using Autoenconders

The analysis which we are going to carry out in this subsection could be viewed as a nonlinear generalization of that made using PCA. Remember that, as mentioned above, we have calculated for each file of each channel (which contains either 10-minute or 5-minute intervals) the minimum, the maximum and the eight statistics of the table 3.1. Let us start with a brief reminder of what an autoencoder is:

Definition 3.2.2. *An **autoencoder** is an unsupervised artificial neural network wich learns to reproduce input vectors* $\{x_1, x_2, \ldots, x_n\}$ *as output vectors* $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\}$*. That is, the objective is that the output of the autoencoder is the input itself. [18].*

Anomaly detection based on machine learning models, tries to capture the normal behavior of the data in the training period, and then test if the the rest of the data can fit the trained model or not. [18]. If the test data does not match the trained model, we consider it an anomaly. For this purpose,

we will use the reconstruction error. With the encoder we transform the input into its compressed representation (in a lower dimensional subspace), and with the decoder stage we try to reconstruct the original input from the lower dimensional representation, trying to obtain a small reconstruction error. [17], [22]. In the four cases under study, we will use the following encoder-decoder structure:

ENCODER:

- Dense layer. 10 neurons. Activation: *ReLU*.
- Dense layer. 10 neurons. Activation: *ReLU*.
- Dense layer. 5 neurons. Activation: *ReLU*.

DECODER:

- Dense layer. 5 neurons. Activation: *ReLU*.
- Dense layer. 10 neurons. Activation: *ReLU*.
- Dense layer. 10 neurons. Activation: *tanh*.

The process to be followed in this case will be as follows:

1. We apply the previous autoencoder structure to the ten statics of the first quarter of each channel. The output of the autoencoder will reconstruct the initial features.
2. Once we have the reconstruction of each feature of the complete signal, obtained by the autoencoder, we calculate the RMSE, and then we have the reconstruction error.
3. Once the reconstruction error has been calculated, we establish two criteria to classify them as normal or anomalies:
   - Use GMM to find two Gaussians in the reconstruction errors of the complete signal except the first quarter (which we assume to be healthy).
   - Find the Z-Scores using the mean and variance of the data from the first quarter of the channel. We will set a threshold of 3 to detect anomalies.

Let us present the analysis carried out for the four channels that end up presenting a failure, following a scheme analogous to that performed with PCAs:

- **Channel 1, test 2.** As in the case of the analysis performed with PCA, we will train the autoencoder on the first quarter of the signal (train). We find the reconstruction errors on the whole signal and the RSME. Then we apply *GMM* with two Gaussians to the reconstruction errors of whole signal except the first quarter. In the following figure we see the reconstruction errors of the complete signal except the first quarter, and the average of the two Gaussians obtained with *GaussianMixtureModel*:

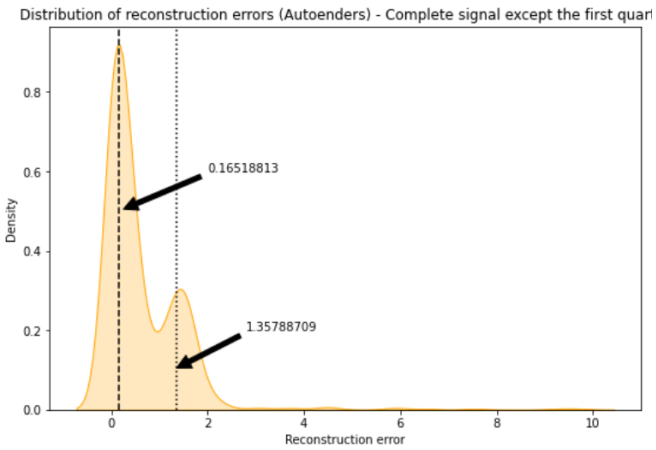


Figure 3.43: Distribution of the reconstruction errors (Autoencoder) of the complete signal except the first quarter (channel 1, test 2).

We consider as anomalous the data with reconstruction errors classified as the Gaussian with mean farthest from 0, and display the results in Figure 3.44. In addition, if we locate the beginning of anomalies as the first moment when 6 errors in a row are classified as anomalous, we get the classification of Figure 3.45.

Figure 3.44: Reconstruction error and classification of anomalies (Autoencoder - GMM), channel 1, test 2.

Figure 3.45: Reconstruction error and beginning of anomalies (Autoencoder - GMM), channel 1, test 2.

To detect anomalies, we now consider the Z-Scores, with a threshold of 3. In Figure 3.46, we show the reconstruction errors classified as anomalous or normal with this criterion, and in Figure 3.47, the classification after locating the beginning of anomalies as the first time at which 6 consecutive reconstruction errors are classified as outliers:

Figure 3.46: Reconstruction error and classification of anomalies (Autoencoder - Z-Scores), channel 1, test 2.

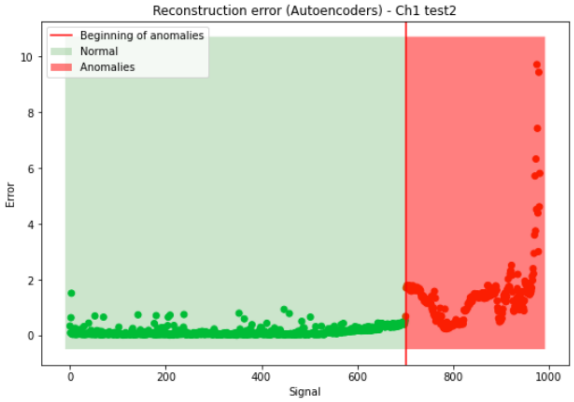


Figure 3.47: Reconstruction error and beginning of anomalies (Autoencoder - Z-Scores), channel 1, test 2.

- **Channel 3, test 3.** As in the previous case, we will train the autoencoder on the first quarter of the signal. Let us begin by showing the results with the first approach used: apply GMM to the reconstruction errors of whole signal except the first quarter. In Figure 3.48 we show the classification of the reconstruction errors into normal or anomalies using GMM. If we locate the beginning of anomalies as the first moment when 6 errors in a row are classified as anomalous, we get the classification of Figure 3.49.

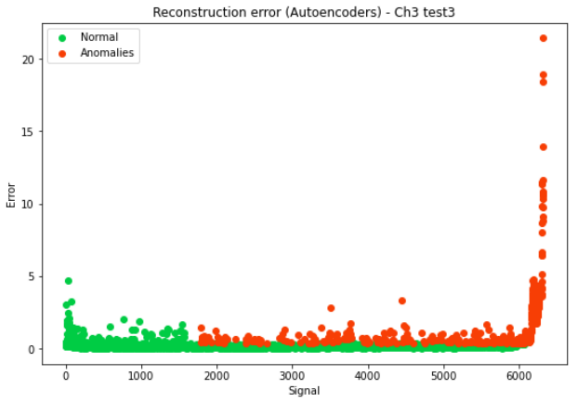


Figure 3.48: Reconstruction error and classification of anomalies (Autoencoder - GMM), channel 3, test 3.

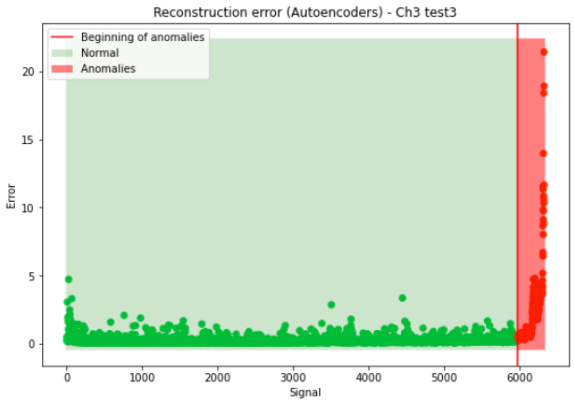


Figure 3.49: Reconstruction error and beginning of anomalies (Autoencoder - GMM), channel 3, test 3.

In Figures 3.50 and 3.51 we show the reconstruction errors and their classifications, performed analogously to the previous case, but using Z-Scores in place of GMM.

Figure 3.50: Reconstruction error and classification of anomalies (Autoencoder - Z-Scores), channel 3, test 3.

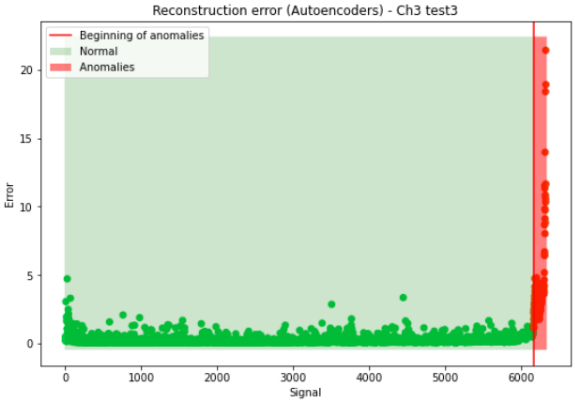


Figure 3.51: Reconstruction error and beginning of anomalies (Autoencoder - Z-Scores), channel 3, test 3.

- **Channel 5, test 1.** As in the previous cases, let us see the results obtained in this case using GMM and Z-Scores:

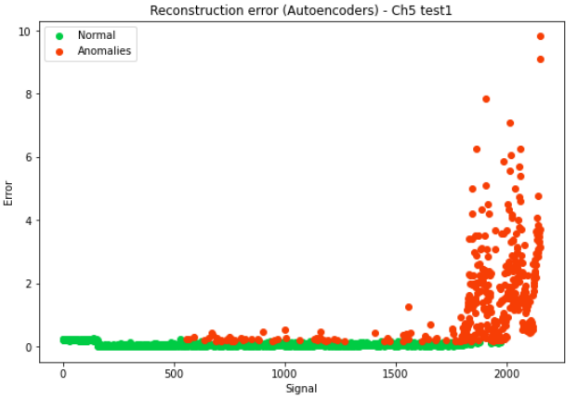


Figure 3.52: Reconstruction error and classification of anomalies (Autoencoder - GMM), channel 5, test 1.

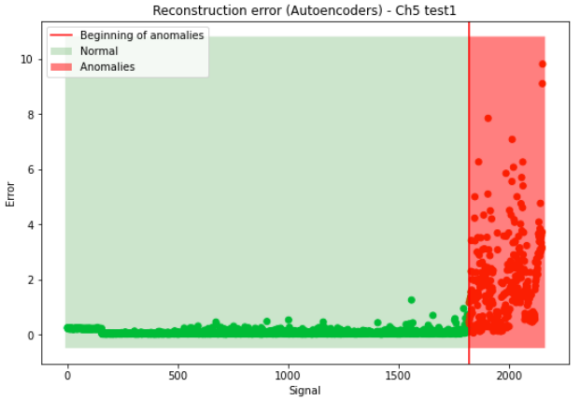


Figure 3.53: Reconstruction error and beginning of anomalies (Autoencoder - GMM), channel 5, test 1.

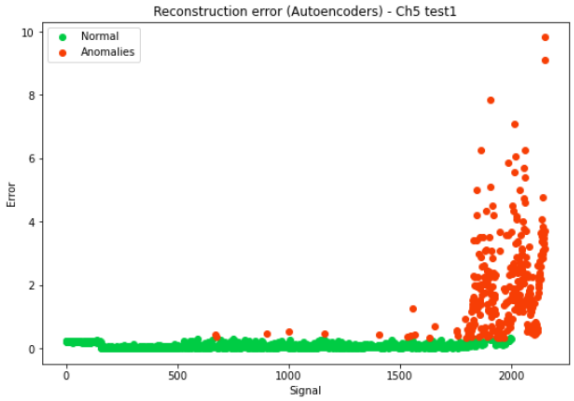


Figure 3.54: Reconstruction error and classification of anomalies (Autoencoder - Z-Scores), channel 5, test 1.

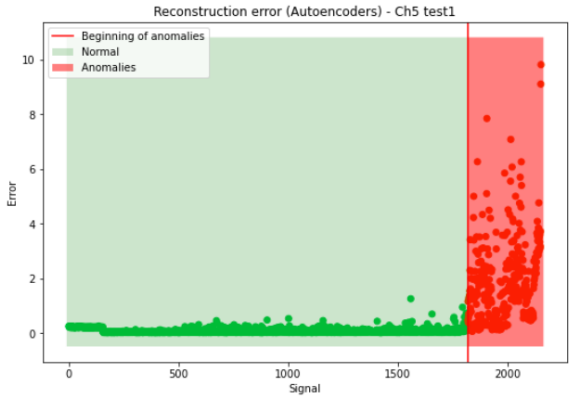


Figure 3.55: Reconstruction error and beginning of anomalies (Autoencoder - Z-Scores), channel 5, test 1.

- **Channel 6, test 1.** We conclude this subsection by showing the classifications obtained using again GMM and Z-Scores:

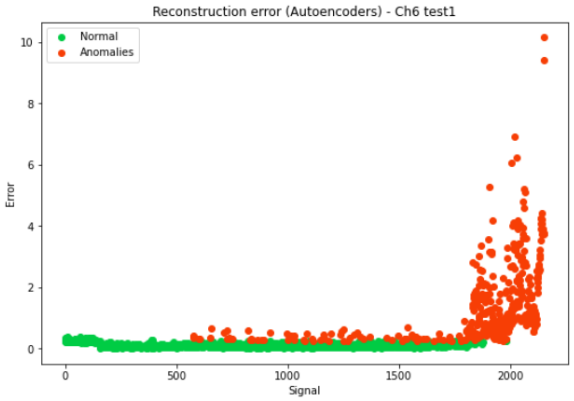


Figure 3.56: Reconstruction error and classification of anomalies (Autoencoder - GMM), channel 6, test 1.

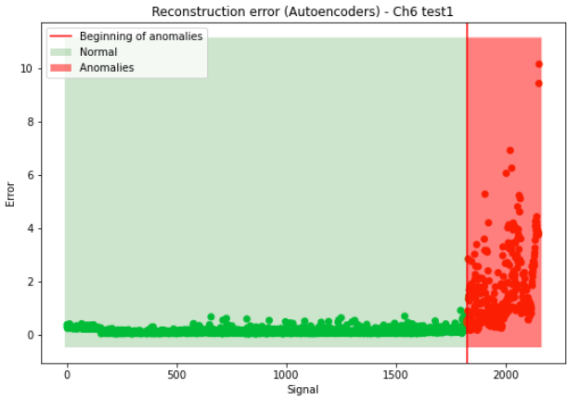


Figure 3.57: Reconstruction error and beginning of anomalies (Autoencoder - GMM), channel 6, test 1.

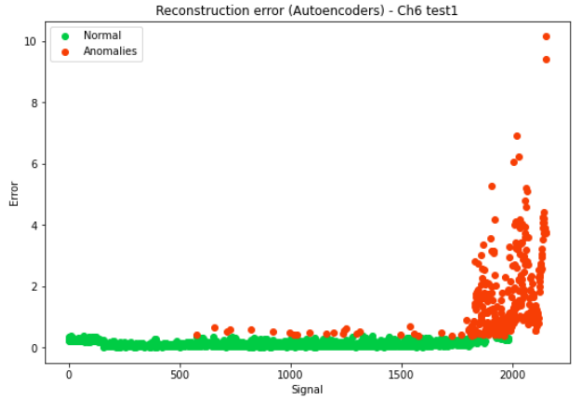


Figure 3.58: Reconstruction error and classification of anomalies (Autoencoder - Z-Scores), channel 6, test 1.

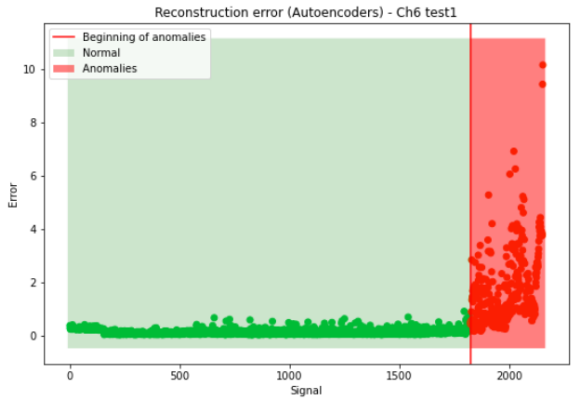


Figure 3.59: Reconstruction error and beginning of anomalies (Autoencoder - Z-Scores), channel 6, test 1.

3.2.3 Anomaly detection using One Class SVM

We will conclude this section by discussing the application of a classic technique that can be used to detect anomalies: One Class SVM.

First of all, the basic idea of SVM is, given a set of non-linearly separable data, to transform it to a higher dimensional space so that in this space these data can be linearly separated. [12]. On the second hand, with the One Class SVM algorithm (OCSVM) we try to distinguish certain samples from the learned ones in a training set, which only contains data of one class. In this case, the main idea is that the algorithm maps the data into a new feature space, using an appropriate kernel function, and then tries to find the hyperplane or hypersphere that separates normal data from outliers. [21].

In our case, we will pass as training the first quarter of the signal (which we assume to be healthy), and given the rest of the signal, we will try to identify the samples that are of the same type as the trained data (and therefore will be identified as healthy), and those that differ from the trained data class will be identified as anomalous.

To apply this technique in Python, we will use *OneClassSVM* from *sklearn.svm*. Specifically, we will use the Gaussian kernel, and we will apply this technique with the 10 statistics used in this section. We will locate the onset of anomalies as the first moment when 6 consecutive data are classified as anomalous. As an example of the results, we show the kurtosis of each 10-minute signal, showing in this plot the moment when the anomalies begin to be detected. Note that the onset of anomalies has been calculated using the 10 statistics, and that we only show kurtosis as an example.

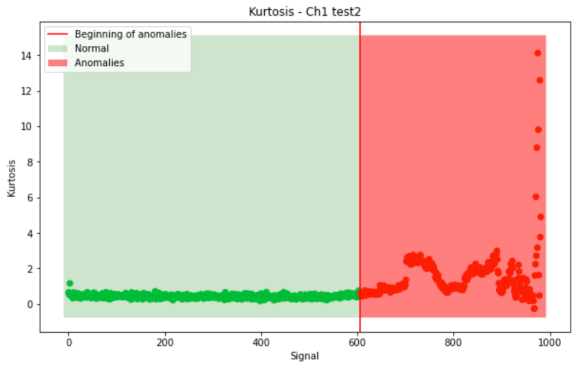


Figure 3.60: Kurtosis and beginning of anomalies (OCSVM), channel 1, test 2.

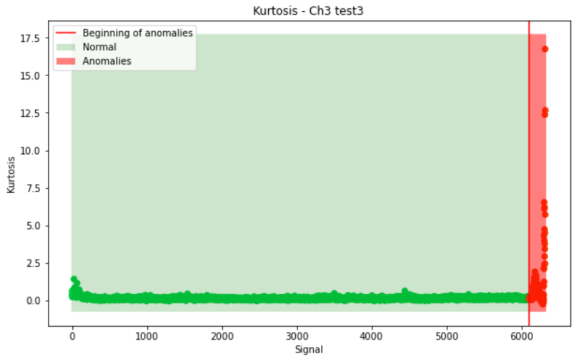


Figure 3.61: Kurtosis and beginning of anomalies (OCSVM), channel 3, test 3.

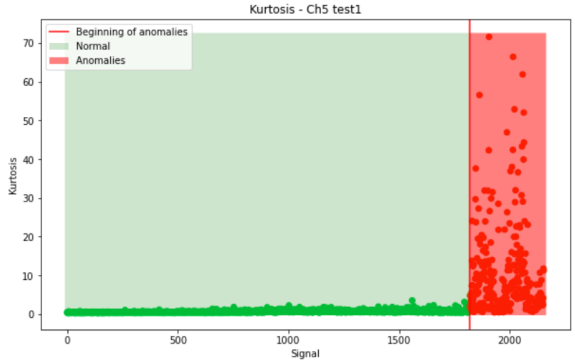


Figure 3.62: Kurtosis and beginning of anomalies (OCSVM), channel 5, test 1.

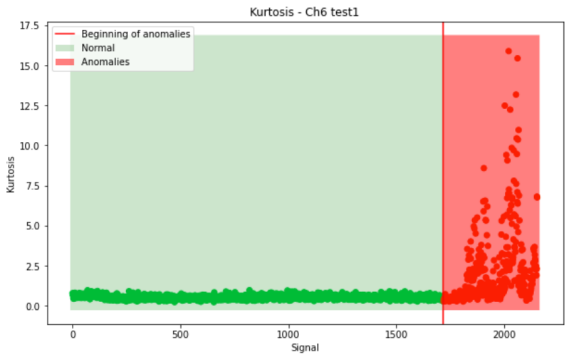


Figure 3.63: Kurtosis and beginning of anomalies (OCSVM), channel 6, test 1.

Before proceeding to discuss the results of the outlined methods on the different bearing data sets in the next chapter, we first provide some brief notes on the development and tools used in this study.

3.3 Development

The present study was carried out using Python, specifically version 3.8. In addition, the following libraries have been used:

- **NumPy:** Library specialized in numerical analysis. Support for the creation of vectors and matrices, as well as for the use of different mathematical functions. Version 1.19.3.
- **Pandas:** Tool for data analysis and manipulation. Written as an extension of NumPy. Version 1.2.4.
- **Matplotlib:** Library for creating static, animated and interactive visualizations in Python. Version 3.4.1.
- **Seaborn:** High-level interface for creating visualizations. Version 0.11.1.
- **SciPy:** Library with different mathematical tools and algorithms. Used for the calculation of statistics, in the preprocessing of signals to obtain different filters and window functions, and to use the FFT algorithm (for envelope spectrum). Version 1.6.1.
- **PyHHT:** Module that implements the Hilbert-Huang Transform (HHT). Used in the function which calculates the envelope spectrum. Version 0.1.0.
- **Scikit-learn:** Library wich contains tools for data analysis, especially used in our case to predict with different machine learning models, such as *kNN, Random Forest...* Version 0.24.2.
- **LightGBM:** Gradient boosting framework which uses tree-based learning algorithms. For fault detection, *LGBMClassifier* from this library is used. Version 3.2.1.
- **Keras:** Open Source Neural Networks library written in Python. Used for the anomaly detection using Autoencoders. Version 2.4.3.
- **Hyperopt:** Hyper-parameter optimization. Used in this study to optimize the parameters of *LightGBMClassifier*. Version 0.2.5.
- **Joblib:** In our case, library used to export models for forecasting. Version 1.0.1.

All the development and methodology presented has been collected in the following GitHub repository: https://github.com/judithspd/predictive-maintenance. In addition, the README file of this repository shows how to reproduce the results, using a *requirements.txt* file with the necessary libraries and their dependencies. This repository is licensed under the MIT license (see https://github.com/judithspd/predictive-maintenance/blob/master/LICENSE), and the following Zenodo DOI has been created for it: https://zenodo.org/record/4922623.

Chapter 4

Results and discussion

During the previous chapter we have presented different results that have guided the methodology to be followed. We will now proceed to present a summary of the main results derived from this study and the corresponding discussion.

4.1 Fault detection

To approach this problem we have distinguished two cases:

- *Constant rotational speed.* This case has been discussed in the subsection 3.1.1. The first approach to this problem will be to find the envelope spectrum. We try to locate the most significant peaks of the envelope and see with which harmonic they coincide: if they do it with f_r, we will consider that the signal is healthy, if they coincide with BPFO, that it has failure in outer race, and if it is with BFPI, failure in inner race.

  If no significant peaks are found, or if there are several and they coincide with different harmonics, we will have to resort to other types of techniques. In our study we have presented tests using classical machine learning models such as kNN or random forest. Using kNN, the accuracy obtained in train has been 1, and 0.99 in test. In addition, using Random Forest, both in train and test the accuracy obtained has been 1.

- *Variable rotational speed.* This case has been discussed in the subsection 3.1.2. In this case, we do not obtain good results using classical techniques such as envelope spectrum. That is why we have resorted to machine learning models. Concretely, we are going to present some of the models studied and the accuracy obtained for train and test:

Machine learning model	**Accuracy train**	**Accuracy test**
Decision tree	1	0.94
SVM	1	0.96
kNN	0.98	**0.98**
Random forest	1	0.96
Majority voting	1	0.97
Stacking classifier	1	0.97
Ada Boosting	1	0.95
Gradient Boosting	1	0.97
LGBMClassifier	1	0.95

Table 4.1: Table with accuracy for train and test with different machine learning models.

The table above shows that all the Machine Learning models tested give very good results in terms of accuracy in both train and test. However, as expected, it is the ensemble methods that give the best results as they combine the power of different models. In view of the table above, the models that give the best results are *kNN*, *majority voting*, *stacking classifier* and *gradient boosting*. In terms of accuracy *majority voting*, *stacking classifier* and *gradient boosting* give the same results, although *stacking classifier* presents slightly better results, since in test it showed 2 failures out of 90, while the other 2 models, 3 out of 90. In addition, *kNN* is the model that gives the best results in test, although while the rest of the models obtain an accuracy of 1 in train, in this case it is moderately lower, 0.98.

4.2 Time to failure

The objective in this problem will be to detect the time to failure in a signal. Specifically, we have used 4 signals that end up presenting a failure: channel 1 test 2, channel 3 test 3, channel 5 test 1 and channel 6 test 1 (see Section 3.2).

In this case we present three techniques to detect anomalies: based on the reconstruction error of PCAs, on that of autoencoders, and One Class SVM (OCSVM). In all three cases, we trained on the first quarter of each signal, which we assumed to be healthy. Both models were applied on 10 input features, which were statistics obtained for each signal fragment of duration 5 or 10 minutes. In addition, for the two first techniques we established two ways to identify a reconstruction error as anomalous: using GMM or Z-Scores. Finally, with each of these two techniques we have used two criteria: to directly put all the data indicated by the established criteria (GMM or Z-Scores), or to find the onset of anomalies as the first moment when 6 consecutive errors are classified as outliers. This second criterion seems to be the most robust, since the other way could lead to a very premature onset of anomalies.

For the four signals that we have studied, which end up presenting a failure, we are going to show in the following table the minutes in advance with which the failure is detected, taking as a criterion that the moment when the failure is detected is the first time that 6 consecutive errors are classified as outliers. First let us look at the duration in minutes of the signal:

	Channel 1, test 2	Channel 3, test 3	Channel 5, test 1	Channel 6, test 1
Signal duration	63240 min	9840 min	21345 min	21345 min

Table 4.2: Table with the duration of each signal.

	TIME TO FAILURE			
Method	Channel 1, test 2	Channel 3, test 3	Channel 5, test 1	Channel 6, test 1
PCA - GMM	2820 min	1650 min	3520 min	3280 min
PCA - Z-Scores	3720 min	1720 min	3320 min	3280 min
Autoenconders - GMM	2820 min	3550 min	3350 min	3310 min
Autoenconders - Z-Scores	2810 min	1650 min	3350 min	3310 min
OCSVM	3760 min	2200 min	3350 min	4350 min

Table 4.3: Table with the time-to-failure prediction using different methods.

Let us begin to analyze the results for channel 1 of test 2. We obtain the following lead times:

2820, 3720, 2820, 2810 and 3760 min. The prediction given by OCSVM of 3760 min is probably over-anticipated, as well as that of PCA-Z-Scores (3720 min), since they are much earlier than the other 3 predictions. Graphically we can see that in the case of OCSVM and PCA-Z-Scores, the anomalies began to be detected in an area that, visually, did not seem anomalous (see Figure 4.1). Let us also note that with the autoencoder, similar predictions are obtained with both criteria.

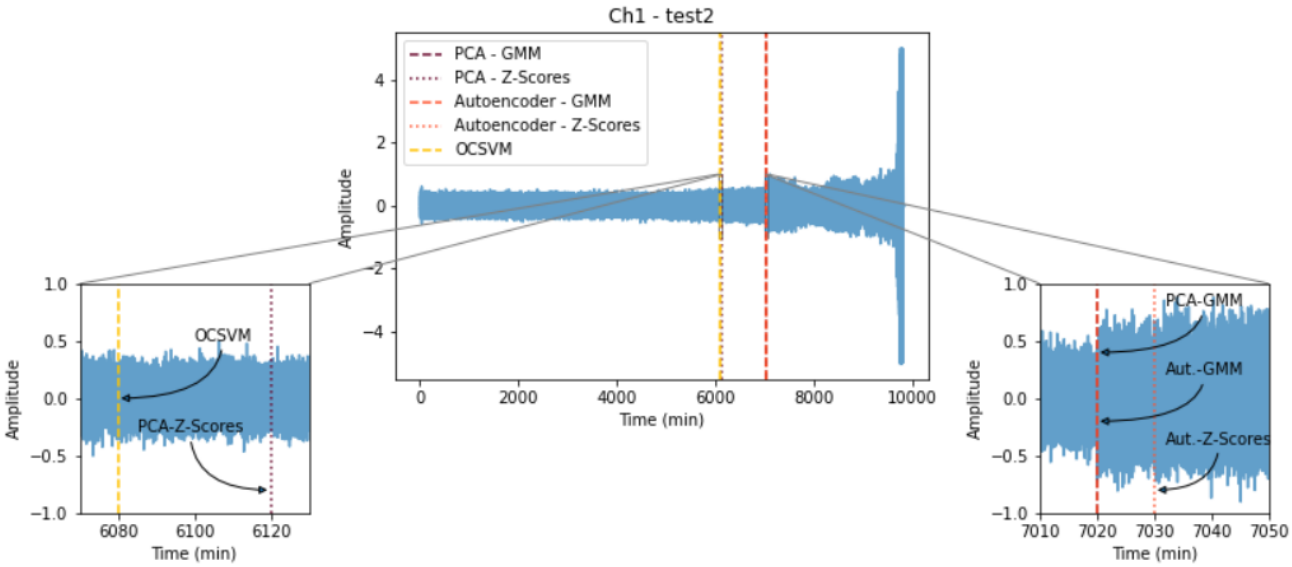


Figure 4.1: Time to failure prediction, channel 1, test 2.

With regard to channel 3 of test 3, note that the time to failure given with the autoencoder and GMM, is somewhat excessive compared to the other results (this is greater than 3000 minutes, while in the rest of the cases, it is less than 2200). The anticipation given by GMM with the autoencoder is likely to be excessive and probably a serious problem is not really occurring in that moment. Again, the prediction given by OCSVM also anticipates more than the others (except for autoencoder-GMM). The following figure shows the time-to-failure prediction for this channel and the different models studied.

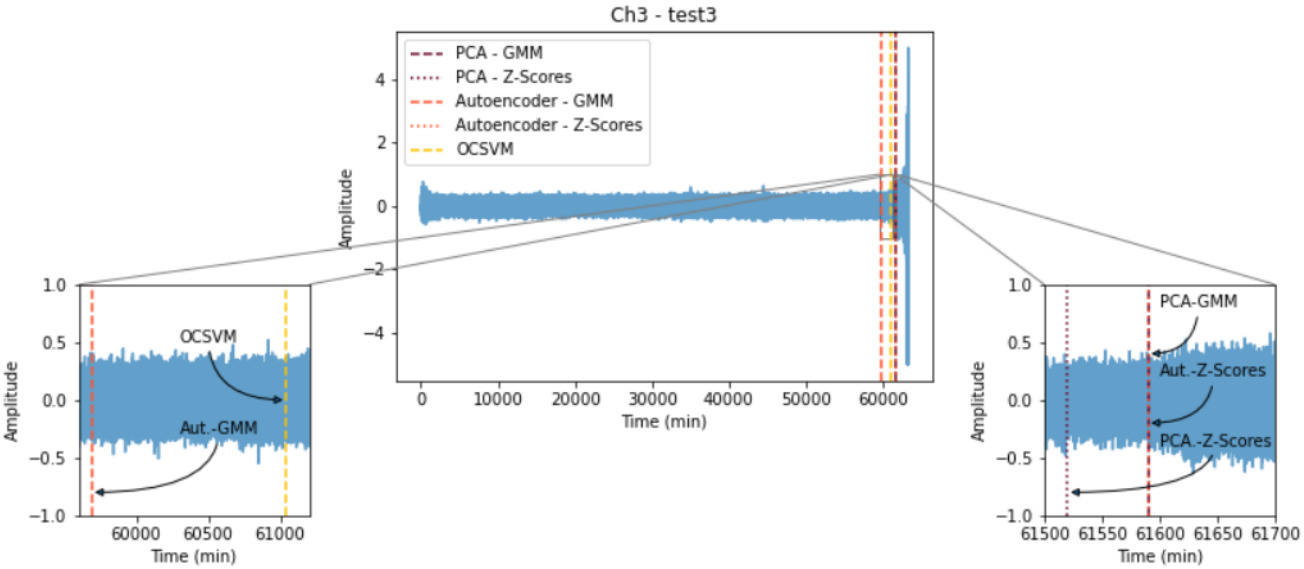


Figure 4.2: Time to failure prediction, channel 3, test 3.

For channel 5 of test 1, with both criteria for PCA, the same value is obtained (3320 min), and

with both with the autoencoder, also the same (3350 min), the difference between them being only half an hour. Therefore, these criteria appear to be more robust in this case. Moreover, with OCSVM the same prediction is obtained as with autoencoder, so they seem very reasonable predictions. Figure 4.3 shows the prediction given by the 5 models for this signal.

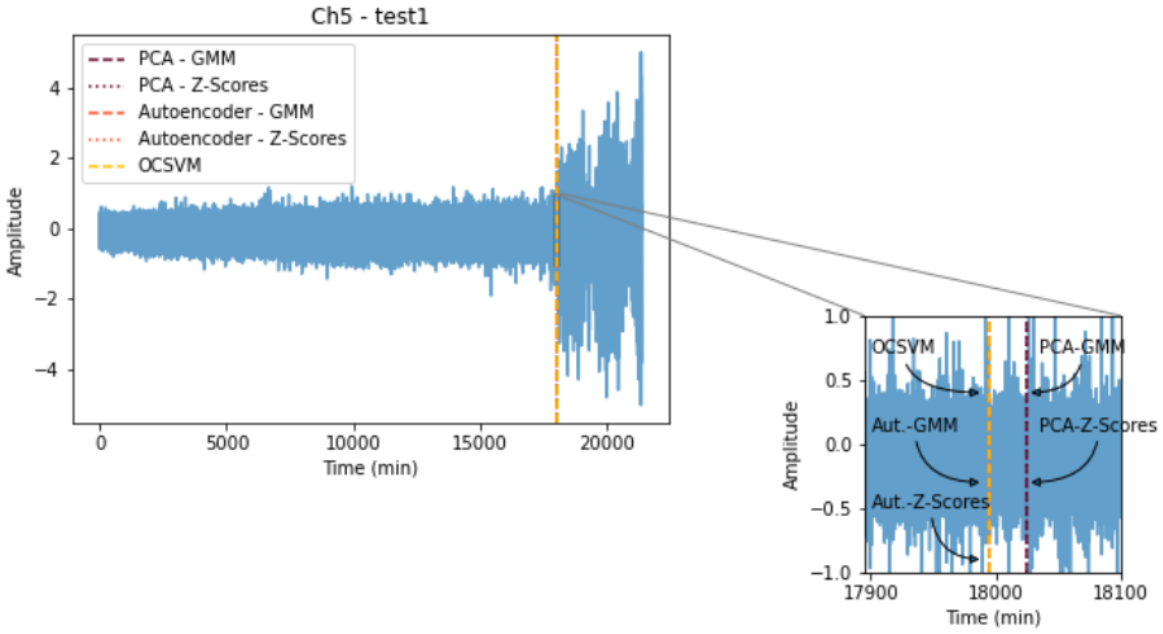


Figure 4.3: Time to failure prediction, channel 5, test 1.

In the same way, in the channel 6 of test 1, with the two PCA criteria we obtain the same value, 3280 min, very similar to that obtained with the autoencoder with GMM or Z-Scores, which in both cases is 3310 min. On the other hand, again with OCSVM, the highest anticipation is obtained, being surely an excessively early anticipation in view of the other predictions with the other models. The following figure shows the prediction for this channel with each studied model.

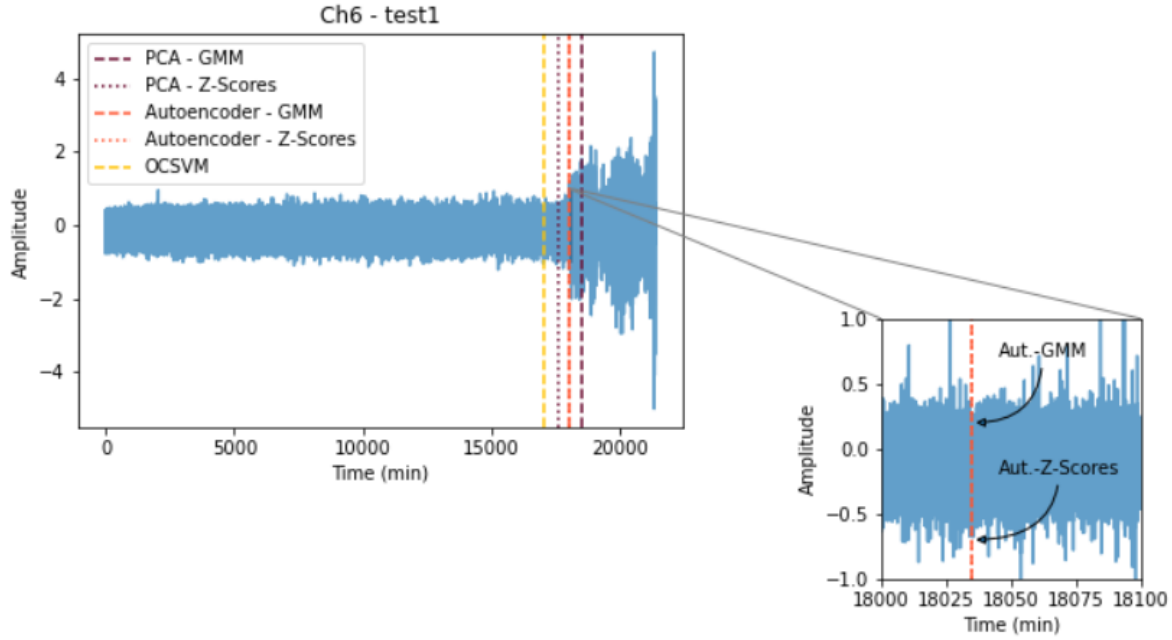


Figure 4.4: Time to failure prediction, channel 6, test 1.

Although we cannot select one criterion as the best or the worst, since we do not know the actual time of onset of anomalies, we can observe that in 3 out of 4 cases, the earliest prediction is given by OCSVM. This has a positive aspect, which is greater prevention, but if it tends to anticipate failure too far in advance, some cases it may be negative because it requires excessively early attention by the maintenance personnel. Furthermore, we should note that OCSVM operates on the data in the original space, while PCA and autoencoder are based on approximating the data. Then, it is possible that PCA and autoencoder lose some accuracy as a result.

Regarding the comparison between Autoencoders and PCA, note that in most cases, the predictions given with Autoencoders are less anticipated than those given by PCA. We can interpret this as being due to the fact that the anomalies fall more clearly outside the linear subspace found by PCA to model the healthy data. While in the case of the autoencoder the anomalous data are still close to the non-linear subspace found for the healthy data.

Finally, in view of Figures 4.1, 4.2, 4.3 and 4.4, it can be seen that the predictions with all models move in similar ranges. It also should be noted that in most cases certain predictions overlap, so we can conclude that the criteria studied appear to be robust.

Chapter 5

Conclusions

In the present study we have started by exposing the key mathematical tools that are the basis for spectral analysis and signal processing. As the title of this paper indicates, we have started by studying the traditional spectral analysis techniques based on Fourier analysis: Fourier transform, DFT, FFT, etc. Then we discuss the Hilbert transform, and a tool that uses the Hilbert-Huang transform as a basis: the empirical mode decomposition (EMD). After describing these tools and the state of the art of predictive maintenance, we detailed the development and methodology carried out for the two main problems of predictive maintenance: failure detection, time-to-failure prediction.

First of all, regarding the problem of detecting the failure, we were interested, given a signal, to be able to say if this signal presents a failure, and in this case, if this failure occurs in the inner race or in the outer race. For this, we have distinguished two cases: signals captured in conditions of constant or variable speed. For the first of these, we have seen that we can apply tools based on FFT and HHT, such as the envelope spectrum, but that not in all cases they present good results. That is why we decided to apply machine learning models, training on different statistics obtained for each signal. Specifically, the learning performed is supervised, since it is known which signals fail and what type of failure they present. Thus, for signals captured in conditions of variable rotational speed, we have resorted directly to this type of models, and some of the most successful techniques are *kNN*, *majority voting*, *stacking classifier* and *gradient boosting*. Thanks to the study carried out, when data captured under similar conditions are obtained, it will be possible to decide which of the above techniques to apply in each case.

As for the prediction of the time to failure, different techniques have been presented to detect the moment when a signal that ends up presenting a failure starts to show anomalies (with a certain amount of time in advance). Specifically, we have used PCA, Autoencoders and OCSVM to detect anomalies, and all of them do the job correctly and give similar results. In our case studies we have not been able to apply supervised learning techniques, since we knew which signals presented a failure, but not at what time it started to be detected. Thus, when other signals are received, we will be able to apply this type of techniques, and if for any signal we know the moment when the failure starts to be detected, we will be able to decide which method to use. In addition, in the case of having a larger number of signals all of them labeled in a way that indicates the beginning of anomalies, machine learning techniques similar to those of the previous problem could be applied. It should be noted that for the data used in this case, the studies in this regard are limited to using expert knowledge about the time at which anomalies would begin, and directly apply supervised learning. In our case we have resorted instead to different anomaly detection techniques.

As future lines for further research, it would be interesting to obtain a larger number of data, so that it is not necessary to apply data augmentation. In addition, as already mentioned, it would be

of great interest to be able to use signal data where the exact time of onset of anomalies is known in order to use supervised learning techniques in the time-to-failure prediction problem. Finally, as a future line of action, an industrial project in a private company environment, based on the techniques studied in this master's thesis, should be highlighted.

Although we have not had a large number of data, being all the data used obtained from open repositories, we have managed to carry out a comprehensive study of different techniques for predictive maintenance, obtaining very satisfactory results. It should also be noted that the studies that focus on predictive maintenance are centered on the use of classical techniques (FFT, HHT, envelope spectrum, etc.), instead of resorting to machine learning techniques as we have done in our study.

Finally, it is worth mentioning that the key parts of this study can be seen in the following GitHub repository: https://github.com/judithspd/predictive-maintenance, for which a Zenodo DOI has also been created: https://zenodo.org/record/4922623.

Bibliography

[1] P. Albrecht, J. Appiarius, E. Cornell, D. Houghtaling, R. McCoy, E. Owen and D. Sharma. Reliability of Motors in Utility Applications. *IEEE Transactions on Energy Conversion* 3 (1987) 396-406.

[2] L. Bluestein. A linear filtering approach to the computation of discrete Fourier transform. *IEEE Transactions* 18(4) (1970) 451-455.

[3] D. Brauckhoff, K. Salamatian and M. May. Applying PCA for Traffic Anomaly Detection: Problems and Solutions. *IEEE INFOCOM* (2009) 2866-2870.

[4] J. Camacho, A. Pérez-Villegas, P. García-Teodoro and G. Maciá-Fernández PCA-based multivariate statistical network monitoring for anomaly detection. *Computers & Security* 59 (2026) 118-137.

[5] M. Cocconcelli, and R. Rubini. Diagnostics of ball bearings in varying-speed motors by means of Artificial Neural Networks. *The Eighth International Conference on Condition Monitoring and Machinery Failure Prevention Technologies* (2011).

[6] K. Dragomiretskiy and D. Zosso. Variational Mode Decomposition. *IEEE Transactions on Signal Processing* 62 (2014).

[7] L.A. Fernández. Introducción a las Ecuaciones en Derivadas Parciales. *Departamento de Matemáticas, Estadística y Computación, Universidad de Cantabria* (2020).

[8] K.H. Hui, C.S. Ooi, M.H. Lim, M.S. Leong and SM. Al-Obaidi. An improved wrapper-based feature selection method for machinery fault diagnosis. *PLoS One* 12 (2017).

[9] S. L. Johnsson and R.L. Krawitz. Cooley-Tukey FFT on the Connection Machine. *Parallel Computing* 18(11) (1992) 1201–1221.

[10] H. Kamata N. Matsumoto, Y. Ishida. A learning algorithm of neural networks for spectrum envelope estimation. *Proceedings of Digital Processing Applications* 1 (1996) 77-82.

[11] A. Khadersab and S. Shivakumar. Vibration Analysis Techniques for Rotating Machinery and its effect on Bearing Faults. *Procedia Manufacturing* 20 (2018) 247-252.

[12] K.L. Li, H.K. Huang, S.F. Tian and W. Xu. Improving one-class SVM for anomaly detection. *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, 5 (2003) 3077-3081.

[13] Z.K. Peng, P.W. Tse and F.L. Chu. A comparison study of improved Hilbert–Huang transform and wavelet transform: Application to fault diagnosis for rolling bearing. *Mechanical Systems and Signal Processing* 19 (2005) 974-988.

[14] V.K. Rai, A.R. Mohanty. Bearing fault diagnosis using FFT of intrinsic mode functions in Hilbert–Huang transform. *Mechanical Systems and Signal Processing* 21 (2007) 2607–2615.

[15] R.I. Ramírez-Castro, J. Montejo. Transformada de Hilbert, descomposición modal empírica y sus aplicaciones en el análisis de vibraciones libres. *Revista Internacional de Desastres Naturales, Accidentes e Infraestructura Civil* 11 (2011).

[16] D. A. Reynolds. Gaussian Mixture Models. *Encyclopedia of biometrics* 741 (2009) 659-663.

[17] S. Russo, A. Disch, F. Blumensaat and K. Villez. Anomaly Detection using Deep Autoencoders for in-situ Wastewater Systems Monitoring Data. *arXiv* (2020).

[18] M. Sakurada and T. Yairi. Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction. *Association for Computing Machinery* (2014) 4–11.

[19] M. Sewell. Principal Component Analysis. Department of Computer Science. University College London. (2007).

[20] W.A. Smith, R.B. Randall. Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study. *Mechanical Systems and Signal Processing* 64-65 (2015) 100-131.

[21] Y. Xiao, H. Wang and W. Xu. Parameter Selection of Gaussian Kernel for One-Class SVM. *IEEE Transactions on Cybernetics* 45 (2015) 941-953.

[22] C. Zhou and R. C. Paffenroth. Anomaly Detection with Robust Deep Autoencoders. *Association for Computing Machinery* (2017).

[23] Bearing vibration data collected under time-varying rotational speed conditions. *Data in Brief* 21 (2018) 2352-3409. Available at: https://www.sciencedirect.com/science/article/pii/S2352340918314124. Consultation date: 23/02/2021.

[24] Case Western Reserve University Bearing Data Center Website. Available at: https://csegroups.case.edu/bearingdatacenter. Consultation date: 23/02/2021.

[25] MATLAB *envspectrum* function. Envelope spectrum for machinery diagnosis. Available at: https://www.mathworks.com/help/signal/ref/envspectrum.html. Consultation date: 26/02/2021.

[26] NASA Data Repository. Available at: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/. Consultation date: 07/04/2021.

[27] Scipy reference guide of signal processing. Available at: https://docs.scipy.org/doc/scipy/reference/signal.html. Consultation date: 16/02/2021.

[28] Scipy reference guide of window functions for filtering and spectral estimation. Available at: https://docs.scipy.org/doc/scipy/reference/signal.windows.html. Consultation date: 16/02/2021.