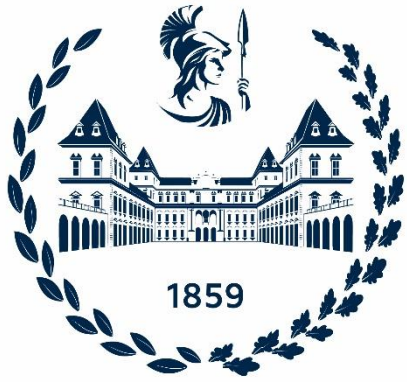
Politecnico di Torino

Master's Degree in Engineering and Management



Politecnico di Torino

Predictive Maintenance based on Vibrational Analysis and Machine Learning in RDM Group

Supervisor:
Prof.ssa Giulia Bruno

Candidate:
Riccardo Toso

Academic Year 2023-2024

TABLE OF CONTENTS

INTRODUCTION **7**

CHAPTER 1 - Vibrational Analysis **9**

1.1 Diagnosis Methods *9*

1.2 Vibrational Analysis *12*

1.2.1 Data Acquisition Methods 14

1.2.1 Signal Processing and Feature Extraction Techniques 17

CHAPTER 2 – Machine Learning approach **35**

2.1 Decision Tree (DT) *36*

2.2 Random Forest (RF) *37*

2.3 Support Vector Machine (SVM) *39*

2.4 K-Nearest Neighbors (k-NN) *41*

2.5 Fuzzy Logics *44*

2.6 Artificial Neural Networks (ANNs) *44*

2.7 Performance Evaluation Metrics of Machine Learning Algorithms *51*

CHAPTER 3 - Case Study and Methodology **55**

3.1 The Company's Necessity: A Predictive Maintenance Approach *60*

3.3 Cilindro Aspirante Copertina BM1 ..... *64*

3.4 Sensors *68*

3.5 The failure *72*

3.6 The Approach to Requirement *74*

3.6.1 Goal of the project 75

3.6.2 The methodology 76

CHAPTER 4 - Implementation and Python code **83**

4.1 Python Libraries Used *84*

4.2 Code Implementation *98*

CHAPTER 5 – Results and Discussion **117**

5.1 Signal Processing Outputs *117*

5.2 Results of the Machine Learning algorithms application in the Fault Recognition phase *127*

5.3 Application and evaluation of the model to latest data *137*

5.4 Training and evaluation of a new model using the latest data *142*

CONCLUSIONS **149**

BIBLIOGRAPHY **153**

ABSTRACT

The advent of Industry 4.0 has significantly transformed the manufacturing industry landscape with the introduction of interconnected systems, advanced automation and real-time data acquisition. In this context, Predictive Maintenance (PdM) has gained increasing relevance as a key component to ensure business continuity and competitiveness of companies. Maintenance is now at the heart of business strategy.

An unexpected failure in an industrial environment results into significant losses for companies as it affects production. In fact, an unexpected failure leads to the sudden interruption of production and can therefore have a negative impact on the product supply to customers, customer satisfaction and the company's reputation. In addition, a failure leads to significant damage to the machine or plant, increasing maintenance costs and can seriously compromise the safety of operators. Predictive Maintenance aims to prevent these failures by anticipating them. The benefits of adopting a PdM approach are numerous and include reduced machine downtime, significant savings in maintenance costs, improved safety in the working environment, improved reliability and production quality, and more precise planning of maintenance activities.

One of the main challenges of PdM is to design and develop an embedded intelligent system to monitor and predict the health status of the machine. This project aims to examine the implementation of a predictive maintenance model based on the concrete application of vibration analysis and Machine Learning techniques, through a case study conducted at the RDM Group company. Intelligent predictive monitoring and smart decision-making processes have become a crucial requirement for the company to safeguard industrial assets from damages that would compromise the achievement of business objectives and that would result in a loss of competitiveness.

The main goal of this research is to address the challenges inherent in maintenance in an environment where early recognition of potential failures is critical. Vibrational analysis represents a powerful and effective diagnostic methodology for monitoring the condition of industrial machines, enabling the early identification of anomalies and signals of imminent faults. The combined use of Machine Learning algorithms offers the opportunity to obtain accurate predictions and become an important and supportive tool for decision-making processes. Through a detailed analysis of machine vibrations and the use of Machine

Learning algorithms, this research intends to develop a fault recognition model capable of anticipating imminent faults and optimizing maintenance.

The case study within the RDM Group provides a concrete and illustrative overview of the challenges and opportunities arising from the implementation of a Predictive Maintenance strategy in the context of Industry 4.0.

INTRODUCTION

The "Fourth Industrial Revolution", or Industry 4.0, has profoundly changed the industrial sector as a result of the continuous development of new digital technologies. In particular, companies are focusing more and more on increasing the integration and interconnection between physical and digital systems. In this context, data is becoming progressively more important. Indeed, this digital transformation in the production environment allows the acquisition and processing of large amounts of data. In this context, Predictive Maintenance plays a crucial role today, increasing its popularity in manufacturing companies.

Predictive Maintenance is a strategic approach that exploits the collection and analysis of real-time data through the use of Machine Learning algorithms to accurately and timely predict when an industrial component will require maintenance, thus enabling targeted and efficient intervention to optimize plant performance.

The main purpose of maintenance is to ensure the correct operation of the plant, intervening where a fault is likely to occur, extending the useful life of a machine. Today, in the industrial sector, there are three types of approach to maintenance: corrective maintenance, preventive maintenance and Predictive Maintenance. The former is based on carrying out maintenance only after a fault has occurred compromising the operation of the component or machine. Preventive Maintenance, on the other hand, involves the planning of maintenance activities carried out at regular intervals with the aim of preventing a potential failure regardless of the real status of the system. On the other hand, Predictive Maintenance, by means of advanced algorithms, aims at predicting failure in advance, so that maintenance activity is only carried out when strictly necessary avoiding potential damage to the plant and production.

A manufacturing company that implements a Predictive Maintenance approach can benefit from numerous advantages. Scheduling maintenance activities only when really necessary and not on a preventive basis allows a strong reduction in costs associated with maintenance. Being able to predict a potential failure helps to reduce operating costs and maximize production efficiency. Knowing when and where to intervene reduces the risk of workplace injuries by improving operator safety and optimizing stock management.

In a business context such as that of the RDM Group, where a large part of production is non-stop, the uninterrupted preservation of operational efficiency becomes a critical goal. In this scenario, the Predictive Maintenance approach assumes a role of primary importance as

it allows to avoid both planned shutdowns to perform preventive maintenance and unplanned interruptions due to sudden machine failures.

In this study, the design and effective development of a successful Predictive Maintenance model applied to a real business context is presented. In particular, the focus was on a machine in an RDM Group's plant.

The methodology adopted in this study includes the application of vibrational analysis to monitor machine health combined with the application of Machine Learning algorithms for the Fault Recognition phase. The primary objective of this research was to address and overcome the inherent challenges associated with Predictive Maintenance. In particular, the central focus of this investigation was to develop a model that would exploit the synergies between vibration analysis and Machine Learning. The fundamental goal was to explore and identify the most effective Signal Processing techniques, as well as to identify the Machine Learning algorithms that could offer the best performance in recognizing the status of the machine.

Raw vibrational signals are recorded using mono-axial accelerometers, which provides valuable data for analysis. The processing of the data and the creation of various datasets is a key step in the model development phase. Finally, the outputs of the Signal Processing phase are used to train and evaluate various Machine Learning algorithms, including k-NN, Decision Tree, Random Forest and SVM, which are used for the machine status recognition phase.

Chapter 1 provides a clear overview of vibrational analysis, presenting the main techniques in the time, frequency and time & frequency domains. Chapter 2 outlines the main Machine Learning algorithms suitable for the case under consideration and the rationale behind them. Chapter 3 presents the case study, illustrating the methodology implemented to meet the business requirement. Chapter 4 shows the Python code implemented to develop the model and the main libraries used. Finally, the results and discussion are presented in Chapter 5 followed by the conclusions obtained from this research.

CHAPTER 1 - Vibrational Analysis

With the advent of Industry 4.0, the topic of production optimization assumes a crucial role. In this context, Predictive Maintenance (PdM) has become of interest to many companies. Indeed, an approach based on Predictive Maintenance brings numerous benefits to the company:

- Increasing revenues
- Reduction in lost production time
- Reduction in machinery costs
- Enhanced safety
- Enhanced efficiency of maintenance personnel
- Reduction in labor costs

Today, thanks to recent digital innovations, we are able to measure and monitor production parameters in real time using state-of-the-art sensors. In addition, we have a significant amount of data such that we can implement prediction techniques efficiently. By keeping the machine monitored, potential failures such as unbalance, mechanical wearing, faulty bearings, misalignment and cracking teeth in gears can be predicted, and as a result, it is possible to safely intervene in advance by performing maintenance, thus avoiding an unexpected production breakdown.

1.1 Diagnosis Methods

Diagnosing a machine provides useful information about its status and the condition of its components. A running machine provides many different indicators that, when processed, communicate signals about operating conditions. Thus, there are a multitude of diagnosis methods depending on what kind of information is processed.

1.1.1 Acoustic diagnosis

Acoustic diagnosis of a machine refers to the analysis of the acoustic signals produced by an industrial machine in order to detect any anomalies or problems.

Acoustic signals are sound waves and are picked up by microphones. The signal must then be processed and analyzed through sound algorithms and techniques. This diagnosis technique makes it possible to identify specific patterns or characteristics in the signal that indicate the presence of a problem.

The main advantage is that microphones are easily installed anywhere and, unlike accelerometers or other sensors, do not need a particular surface. In addition, many sensors placed directly on the machine do not work in high temperatures, which is a very common condition when a fault occurs in a machine. On the other hand, the main disadvantage of this method is that the microphones are very sensitive to noise and background reverberations that might cover the information signal. Due to the inherent limitations, acoustics-based diagnostics has not received due attention and the studies that have been conducted are generally carried out in a controlled environment. [1]

1.1.2 Thermography

Another widely used technique in predictive maintenance is thermography.

This method is based on the analysis of the thermal radiation emitted by objects, making it possible to detect an anomaly where the temperature increases in relation to normal working conditions. All objects in fact give emit energy as long as the temperature is above absolute zero (-273.15°C).

This technique is based on the analysis and capture of infrared waves. Thus, looking at the electromagnetic wave spectrum, the infrared wave band extends between the wavelength range of approximately 700 nm to 1 mm.

The sensors that are used to apply thermography are infrared sensors for temperature measurement, line scanners and infrared imaging methods.

The main advantages of thermography are that it is a non-destructive and non-invasive technique and that it can be used to detect anomalies even in hard-to-reach places at a distance. However, it is dependent on environmental conditions (temperature and humidity) and the cost of the equipment is high. [2]

1.1.3 Oil Analysis

Oil analysis is a predictive maintenance technique used to monitor the operating state of industrial machines and plants by analyzing the lubricant (oil) used in such equipment. The condition of the lubricant can be used to detect anomalies, wear, contamination and other conditions that could indicate impending mechanical problems.

The oil analysis process usually consists of the following steps:

1) Sampling
2) Preparation and chemical-physical analysis of the sample in the laboratory
3) Spectroscopic analysis if necessary to identify the presence of specific chemical elements and metal particles suspended in the oil
4) Comparison with reference data
5) Report generation and data interpretation
6) Prediction and maintenance actions

However, this technique requires very high initial costs and requires highly specialized analysis skills. In addition, oil analysis may not be able to detect all types of faults or mechanical anomalies.

1.1.4 Visual Inspection

Visual Inspection (VI) is one of the oldest and simplest diagnosis methods in the field of maintenance. It consists of directly observing the plant or individual component in order to examine its condition.

It can be applied in different ways, according to the specific needs of the application:

- Direct visual inspection: an operator intervenes physically to examine components and machines for evidence of deterioration, damage, wear or other anomalies
- Inspection with auxiliary tools
- Real-time visual monitoring through the use of fixed or mobile cameras

Visual inspection is particularly useful for identifying visible problems. However, despite its wide use, VI has certain limitations. It is often unable to detect problems that are not visible from the external environment, and it is also a technique that depends heavily on the subjectivity of the operator.

To conclude, nowadays, with the technology constantly evolving, Visual Inspection is often functional when used simultaneously with another diagnostic technique.

In addition to those described above, there are many other diagnostic methods for determining faults or potential faults in a machine, such as vibrational analysis, particle analysis, magnetic resonance testing and eddy currents. However, among all these methods, vibration analysis emerges as a popular approach in manufacturing companies, especially when they operate continuous rotating machines, because of its accessible costs and because many faults can be identified without stopping production.

1.2 Vibrational Analysis

Mechanical systems with reciprocating or rotating components generate vibrations as a result of mechanical disturbances originating from various sources such as the engine, sound, and noise. In the field of mechanical engineering, the term "vibration" typically refers to systems that can oscillate freely without external forces. Uncontrolled vibration in mechanical machinery can lead to performance, operational, or safety issues, making it crucial to acquire, analyze, and quantify this parameter for the purpose of improving reliability, lifespan, quality control, productivity, and safety against catastrophic failure. Vibration signatures of the machine can provide early warnings to operators for timely maintenance or to make critical decisions before serious problems or unscheduled downtime occurs. The amplitude of the vibration signature indicates the severity of the problem, while the frequency can help identify the source of the defect.

Extracting these signals serves as a valuable diagnostic tool for predicting run-up failures of machine components. However, extracting features from acquired signals is a challenging task due to interference caused by noise. This necessitates the development of high-level data processing and analysis systems to ensure accurate extraction of machine health data. Various feature extraction techniques, such as statistical domain, frequency domain, time domain, and time-frequency domain, are employed to obtain diagnostic information.

Vibration analysis has the advantage of being highly accurate and effective. However, the sensor technology used to acquire the signal is influenced by noise and the environment, and it is also quite expensive economically. Anyway, vibration analysis is a powerful and reliable

technique for monitoring the operating conditions of machines. Its non-destructive nature and ability to enable sustainable monitoring without interfering with the process have contributed to its increasing popularity and widespread use in the industry. [3]

The vibration analysis for machine monitoring and diagnosis typically consists of three main steps, which are data acquisition, signal processing, and fault recognition [4].

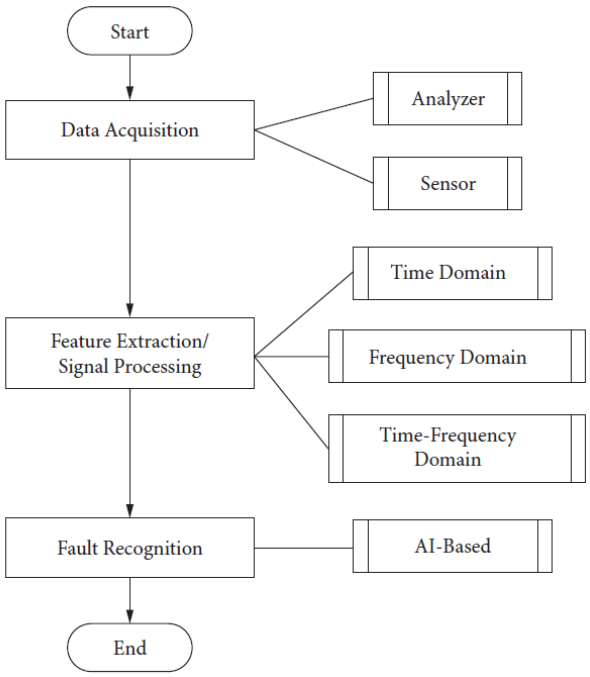


Figure 1: Main vibrational analysis stages [4]

Nowadays, for each stage, briefly described above, there are different techniques and tools that are used. Understanding which techniques and how to use them is challenging because each of them has different advantages and disadvantages depending on the application case.

These approaches can be divided into three groups:

- Data-driven approach: construction of a system behavior model based on historical data
- Model-based approach: building an analytical model of the system
- Hybrid approach

Because it is very difficult to model a faulty system, data-driven methods are widely applied in machine diagnosis and monitoring compared to model-based methods [4].

1.2.1 Data Acquisition Methods

The first stage of vibrational analysis is Data Acquisition. Vibration measurement is and effective, non-intrusive method to monitor machine condition during start-ups, shutdowns and normal operation [5].

Vibration sensors are crucial because, as well as acquiring an accurate signal, they are needed to be able to proceed to the next step of signal processing. The sensors transform the vibrational signal into an electrical signal containing the same information. From this electrical signal we are able to apply signal processing algorithms to extract interesting features to be monitored.

Referring to Figure 2, there are mainly two tools that are used in the first stage of Data Acquisition: analyzer and sensor.

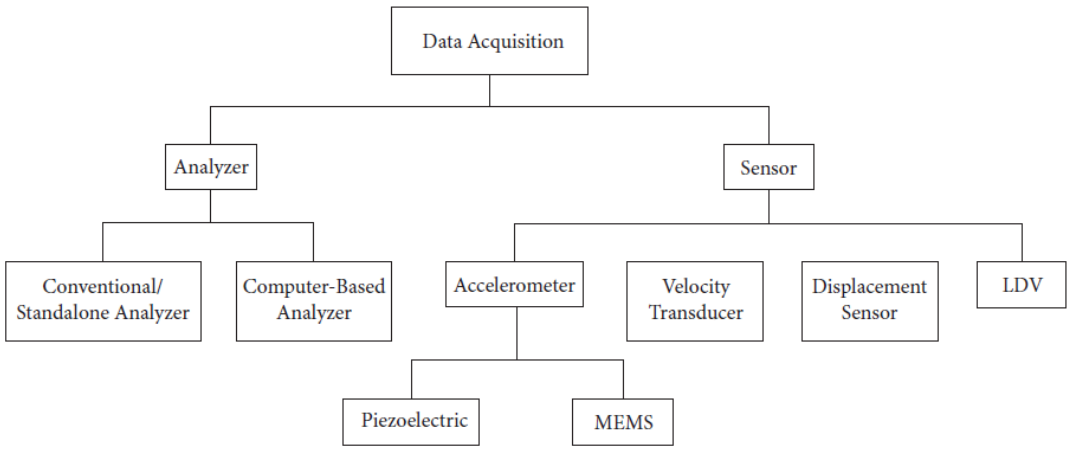


Figure 2: Methods for the Data Acquisition stage [4]

1.2.1.1 Analyzer

An analyzer is an instrument used for the examination of vibration data generated by machinery. It typically consists of several components, including a sensor, amplifier, filter, and Analog to Digital (A/D) Converter. The vibration signal collected by the sensor first undergoes amplification to enhance its resolution and signal-to-noise ratio. Subsequently, the amplified signal is subjected to filtering to prevent aliasing during the digitization phase. The signal is then converted into digital format by the A/D converter before entering the processing unit, where it can be presented as a time waveform or further processed to generate a frequency spectrum. [4]

There are mainly two types of analyzers: conventional (or standalone) and computer-based analyzer. The former is a stand-alone tool that is very expensive and challenging to use.

Usually in this case, companies contact experts from outside to visit and check the condition of the machines. The second is a recent instrument where the raw signal is automatically processed and analyzed using specific software. Computer-based analyzer has achieved popularity due to the development of technology and because it is an easy-to-use tool and can perform the same functions as the conventional analyzer without the need of an expert's presence. However, nowadays its effectiveness and accuracy are lower than that of the conventional analyzer.

1.2.1.2 Sensors

Like analyzers, sensors are instruments that take the raw vibrational signal as input and return as output an electrical signal that is easier to be processed.

Referring to Figure 2, sensors are divided into: Accelerometers, Velocity and Displacement Transducers and Laser Doppler Vibrometers (LDV).

1.2.1.3 Accelerometers

Accelerometers find widespread use in vibrational analysis. These devices are employed to measure the oscillations or changes in velocity of a structure, and they use the SI unit of measurement, 'g' (meters per second squared). Their fundamental operation involves the use of piezoelectric material, which generates an electric charge when subjected to a force. This generated charge is directly proportional to the applied force's magnitude, and any alteration in this relationship results in a corresponding change in the charge, which is then amplified. Accelerometers are available in both single-axis and three-axis configurations. A single-axis accelerometer can detect motion in a single plane, whereas a three-axis accelerometer can capture movement in all three spatial dimensions. Although three-axis accelerometers offer greater data storage capacity compared to their one-axis counterparts, they also come with a higher cost. [4]

Accelerometers can be divided into piezoelectric and MEMS accelerometers. Piezoelectric accelerometers exploit the piezoelectric principle, which is based on the ability of certain materials to generate an electrical charge when subjected to mechanical stress or deformation. MEMS accelerometers are based on Micro-Electro-Mechanical Systems technology, which combines microscopic mechanical and electronic components within a

single device. A typical MEMS accelerometer contains a small mass or suspended structure that can be moved in response to acceleration. This movement is detected using capacitive, piezoresistive or other electronic principles. The movement of the mass causes changes in the electrical properties of the sensor, which are then converted into acceleration signals.

1.2.1.4 Velocity Transducer

These transducers are employed for measuring vibrations in the frequency range of 10–1000 Hz and for performing balance adjustments on rotating machinery [3]. They function by detecting voltage generated due to the relative movement of an object, typically measured in units like meters per second or centimeters per second. The operating principle relies on electromagnetic induction and doesn't necessitate external devices. When the surface beneath the sensor undergoes motion, the motion of a magnet within a coil generates a voltage directly proportional to the vibration's speed. This voltage signal serves as a representation of the vibration itself. [4]

Speed transducers have several advantages such as low cost and very simple installation. However, they are very big and heavy sensors and are also limiting because they are not suitable for high-speed machines.

1.2.1.5 Displacement Transducer

A displacement transducer device can measure both the relative vibration and the position of a shaft. The measurement unit for displacement can be specified in meters, centimeters, or millimeters. Typically, it's employed to gauge vibrations at low frequencies below 10 Hz, but it can also be utilized for measuring vibrations up to 300 Hz. Irregularities like imbalance and misalignment are examples of issues that the motion sensor can identify. [4]

Despite its accuracy and precision, it is seldom used due to its low bandwidth.

1.2.1.6 Laser Doppler Vibrometers (LDV)

The Laser Doppler Vibrometer (LDV) is an optical measurement device that operates without physical contact and enables the measurement of the absolute velocity of a vibrating object. This is accomplished by detecting the Doppler shift of the laser beam that is scattered by the target. The operating principle relies on the Doppler phenomenon associated with

lasers: a coherent laser beam with a modulated frequency is directed at a vibrating surface, and the change in the frequency of the reflected beam (known as the Doppler effect) is compared with a reference beam. With the assistance of a Doppler signal processor, the signal detected by the photodetector is analyzed and converted into the instantaneous velocity of the vibrating object as a function of time. [3]

This technique is very accurate and allows one to change the measuring point at will. However, it is not widely used due to the high cost of instrumentation.

1.2.1 Signal Processing and Feature Extraction Techniques

Referring to Figure 1, signal processing refers to the second stage of vibrational analysis. This stage is fundamental because it represents the heart of the analysis. There are numerous techniques for extracting features from the signal of interest, however, choosing the right techniques is tricky and challenging. One must first understand and be very familiar with the context and environment in which the machine operates. Secondly, one must clarify what one is looking for: the presence of damage in the structure, or the specific location of the damage, or the degree of severity of the anomaly. Depending on the purpose of the analysis and the environment in which the machine operates, there are techniques that are more or less effective.

Signal processing as already mentioned covers many techniques, and these can be classified into three macro-categories. Feature extraction techniques can be divided into analysis in the domain of: time, frequency and time-frequency.

1.2.1.1 Time Domain Analysis

A signal can be categorized as either deterministic or non-deterministic, which is often referred to as random. Deterministic signals are those that can be precisely described using mathematical relationships. In contrast, random signals are not expressible through mathematical formulas. Due to the inherent uncertainty and variability in non-deterministic signals, it is impossible to establish or precisely observe an instantaneous value of the signal. Instead, it is only feasible to estimate the value with a certain degree of probability. Deterministic signals can be further classified as periodic or non-periodic. Periodic signals

exhibit regular repetition and can take the form of sinusoidal waves or more complex periodic patterns. Non-periodic signals, conversely, can be quasi-periodic or transient. [6]

A sinusoidal signal, in the time domain, can be represented by the following expression:

$$x(t) = A \cdot sen(\omega t + \phi)$$

Where A is the amplitude, ω is the radian frequency (equal to $2\pi f$) and ϕ is the phase.

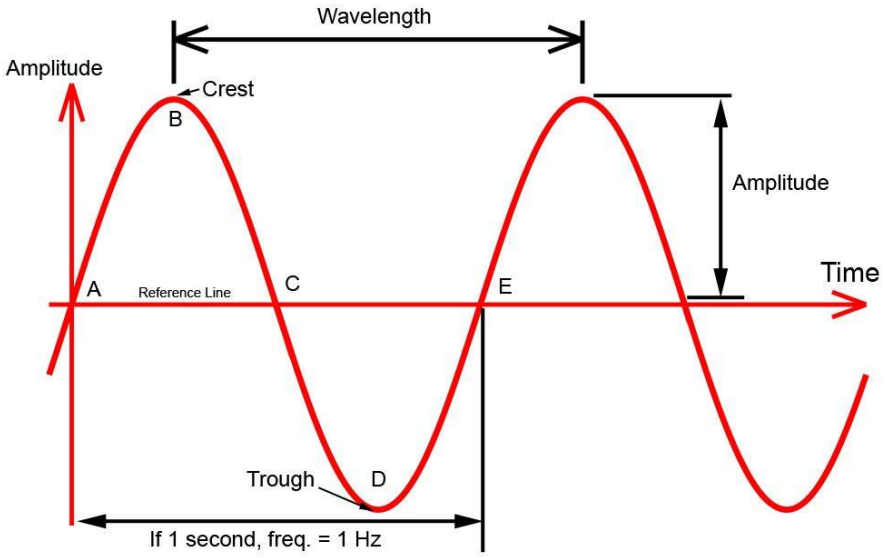


Figure 3: Sinusoidal wave components [7]

A complex periodic signal can be expressed with the following time function:

$$x(t) = x(t \pm n \cdot T) \qquad n = 1,2, \ldots \text{ [6]}$$

Complex periodic signal repeats itself identically after a fixed period of time T. This type of signal can also be written as:

$$x(t) = X_0 + \sum_{n=1}^{\infty} X_n \cdot \cos(\omega t - \vartheta_n)$$

Where: $X_0 = \frac{B_0}{2}$, $X_n = \sqrt{A_n^2 + B_n^2}$ and $\vartheta_n = tan^{-1}(\frac{A_n}{B_n})$.

With

$$A_n = \frac{2}{T}\int_0^T x(t) \cdot sen(n\omega t)dt \qquad n = 1,2, \ldots$$

$$B_n = \frac{2}{T}\int_0^T x(t) \cdot cos(n\omega t)dt \qquad n = 1,2,\dots$$

$$B_0 = \frac{2}{T}\int_0^T x(t)dt$$

This is equivalent to saying that a complex, but periodic phenomenon can be decomposed into a static (zero-frequency) component X_0, and an infinite number of sinusoidal components, the harmonics, each with its own amplitude and phase [6].

It can be shown that if the frequencies forming the signal, divided two by two, give a rational number, the signal is periodic, otherwise the signal is quasi-periodic. While transient signals are all those deterministic signals that cannot be represented with periodicity or quasi-periodicity relations [6].

Time domain analysis is a straightforward and widely used signal processing technique. In the context of machine diagnosis, it entails the examination of the vibration signal recorded over a period of time. Vibration signals, representing factors like proximity, velocity, and acceleration, consist of a series of data points. In time domain analysis, these data points' amplitudes are plotted against time. While more advanced temporal analysis methods exist, it's important not to overlook the visual approach, which involves inspecting the temporal waveform. This visual examination can reveal various insights, such as the presence of amplitude fluctuations, shaft imbalances, transient components, and high-intensity frequencies. [4]

The time-domain signal shows the history of the energy content of the signal and has been applied successfully to numerous complex problems [8].

The main statistical indicators in the time domain are: mean and standard deviation, Root Mean Square (RMS), peak value, peak-to-peak value, crest factor, Kurtosis and Energy Index (EI).

1.2.1.1.1 Mean and Standard Deviation

Signals can be related to the time axis using the mean and standard deviation. Given a set of values, the mean represents the ratio of the sum of the values and the total number of values in the set.

Standard deviation is used to measure the dispersion of the signal and it is calculated as the square root of the variance.

This means that given a set of values, the mean represents the central value of the set, while the standard deviation measures the average dispersion or deviation of the values from their mean.

1.2.1.1.2 Peak value

When examining a signal across a time period, the peak denotes the highest or lowest point in the waveform, representing its maximum value. To calculate this peak, the signal must be discretely sampled over time intervals. In the presence of impacts or anomalies, the peak values in the vibration signal will exhibit variations. In cases of faults, the peak value tends to increase. By assessing the amplitudes of these peaks, one can determine the severity and nature of the fault. [4]

However, it is a measurement that can detect a fault but not the type of fault.

1.2.1.1.3 Root Mean Square (RMS)

RMS value presents the power content in vibration and it is useful in detecting an imbalance in rotating machinery [4].

The initial peaks in the signal at the beginning stages of the fault cannot be detected using RMS, furthermore, RMS is not suitable for identifying the location of the fault. However, RMS is suitable for measuring the severity of the fault. It is also worth noting that RMS is not sensitive to transient changes that can last for only milliseconds [8]. The RMS returns a more realistic and less distorted average signal value than the simple mean.

$$RMS = \sqrt{\frac{A^2}{2}} \approx 0.707A$$

Where A represents the amplitude of the vibration signal.

1.2.1.1.4 Crest Factor (CF)

Crest Factor is an indicator frequently used in vibrational analysis and is calculated as the ratio of peak to RMS.

$$Crest\ Factor = \frac{peak}{RMS}$$

Normally, during production activity under normal conditions, CF assumes values between 2 and 6 [8]. Values above this threshold are usually associated with bearing or machine failures.

Compared to peak and RMS values, the crest factor is usually used when measurements are conducted at different rotational speeds because it is independent of speed [4]. However, like the majority of indicators in the time domain, CF is not accurate in recognizing a fault when it is in its early stages.

1.2.1.1.5 Peak-to-Peak value

The concept of peak-to-peak indicates the variation between the highest and lowest values of a periodic signal. In substance, it represents the vertical distance between the highest and lowest point of the signal, taking into account both its positive and negative oscillations. It is a similar indicator to peak, in both disadvantages and advantages.

1.2.1.1.6 Kurtosis

Kurtosis is a nondimensional statistical measurement of the number of outliers in distribution and in vibration analysis, it corresponds to the number of transient peaks. A high number of transient peaks and a high kurtosis value may be indicative of wear [4].

$$Kurtosis = \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu)^4}{\sigma^4}$$

Where y_i represents the instant amplitude, μ is the mean and σ represents the standard deviation of a sample of n data.

The Kurtosis technique is sensitive to impulsiveness and can detect vibration signals associated with faults in their early stages. The main disadvantage of Kurtosis is that it cannot detect the fault at the later stages [8].

1.2.1.1.7 Energy index

The main disadvantage of Kurtosis is that as the magnitude of the damage progresses, the Kurtosis value decreases to normal values. The Energy Index overcomes this difficulty.

Energy index represents the ratio of the root mean square of the segment of the signal (RMS segment) to the overall root mean square (RMS signal) of the same signal [9].

$$Energy\ index = (\frac{RMS_{segment}}{RMS_{\ signal}})$$

In general, time domain analysis is a powerful and useful tool because it is easy to compute and because it summarizes the signal characteristics of a time interval in a few easy-to-monitor parameters. The most important advantage is that it gives information regarding the signal and its characteristics over time, thus providing an overview of the dynamics and evolution of the signal over time.

On the other hand, time domain analysis also has disadvantages. First of all, it is very sensitive to noise. Due to the inevitable noise or interferences of a real environment, time domain indicators are often misleading and inaccurate. To avoid these problems, numerous denoising techniques exist. A further drawback of time domain analysis is the lack of signal frequency information. In many applications, vibrational analysis aims to study the frequencies of the signal that are most stressed. The frequency composition of the signal often makes it possible to identify the location of faults.

1.2.1.2 Frequency Domain Analysis

In the frequency domain, the main objective is to analyze the frequency content of the vibrations of a system. This makes it possible to obtain detailed information on the dynamic characteristics of the system while completely omitting information on the signal's behavior over time.

According to Fourier, it is possible to state that any complex signal can be decomposed into the sinusoidal waves, each at a certain frequency, that compose it. Unlike the time domain where amplitudes are plotted against a time interval, in the frequency domain the output will be a spectrum where amplitudes are plotted against frequencies. Vibration analysis in the frequency domain thus allows an analysis of the frequency content of a signal by verifying which frequencies are stressed.

One of the most important applications of time domain analysis is the detection of bearing faults. Defects on bearings are divided into two types: distributed and localized defects. Distributed defects are mainly caused by manufacturing error, inadequate installation or mounting and abrasive wear, then these defects include surface roughness, waviness, misaligned races and unequal diameter of rolling elements [11]. Defects such as these usually generate a broad spectrum of machine vibrations, and the raw data cannot locate the defect [8]. Localized defects include cracks, pits and spalls on rolling surfaces caused by fatigue [11]. These defects accelerate when there is an increase in production speed or when engines are subjected to higher loads.

Thanks to the geometry of the bearings, its characteristic frequencies can be monitored. In particular, specific bearing elements generate specific failure frequencies. The most important of these are: Fundamental Train Frequency (FTF) which is the frequency of the defective cage, Ball Pass Frequency of the Inner race (BPFI) which is the frequency of the rotating elements passing over a defect in the inner race, Ball Pass Frequency of the Outer race (BPFO) which is the frequency that the rotating elements produce when passing over a defect in the outer race and Ball Spin Frequency (BSF) which is the frequency of a defective rotating element. These failure frequencies can be calculated with the equations below:

- $FTF\ [Hz] = \frac{1}{2}S\left(1 - \frac{B_d}{P_d}\cos\beta\right)$
- $BPFI\ [Hz] = \frac{n}{2}S\left(1 + \frac{B_d}{P_d}\cos\beta\right)$
- $BPFO\ [Hz] = \frac{n}{2}S\left(1 - \frac{B_d}{P_d}\cos\beta\right)$
- $BSF\ [Hz] = \frac{P_d}{B_d}S\left[1 - \left(\frac{B_d}{P_d}\cos\beta\right)^2\right]$

Where:

- S is the rotational speed (rotations per second)
- n is the number of rolling elements of the bearing
- β is the contact angle
- B_d is the diameter of rolling element
- P_d is the pitch diameter calculated as: $P_d = \frac{D_1 + D_2}{2}$ with D_1, D_2 the diameters of inner and outer race

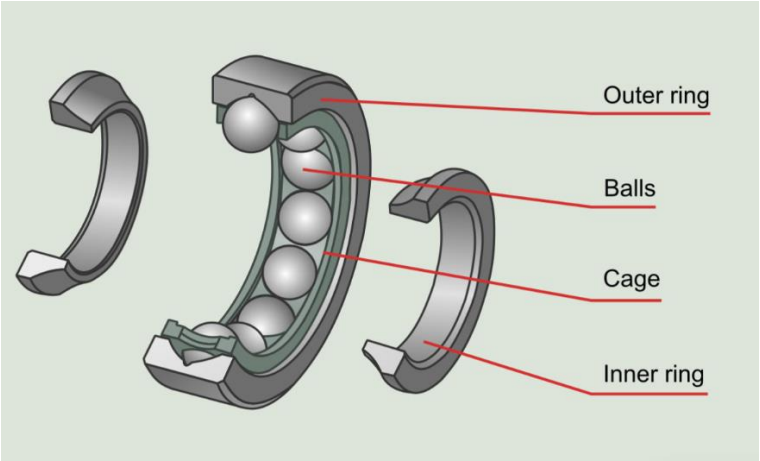


Figure 4: Rolling Element Bearing Components [12]

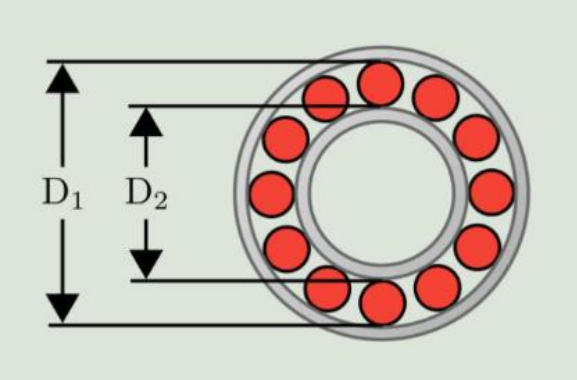


Figure 6: Rolling Bearing Geometry [12]

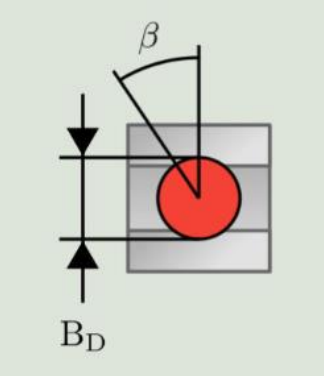


Figure 5: Rolling Element Bearing Geometry & Contact Angle [12]

Vibrational analysis in the frequency domain thus allows to breakdown the signal into the sinusoids of which it is composed and at the same time to highlight the amplitudes of the most stressed frequencies. It is a very powerful tool because it allows us to understand given a stressed frequency whether this is a characteristic frequency, a harmonic of interest or better still whether it is a bearing failure frequency.

The main techniques used in the frequency domain are: the Fourier Transform, Cepstrum Analysis and Envelope Analysis.

1.2.1.2.1 Fast Fourier Transform

The Fourier Transform is a powerful technique to decompose any signal in the time domain into all its sinusoidal components having different frequencies. Fourier states that any time-varying signal can be obtained by summing harmonics. The Fourier Transform is fundamental to understanding the frequency structure of a signal, revealing dominant frequencies, harmonics and other spectral characteristics. This process reveals which frequencies contribute to the signal composition and with which amplitude.

The Fourier Transform (FT) formula for a continuous function $f(t)$ is given by:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt$$

Where:

- $F(\omega)$ is the Fourier Transform (or amplitude of the sinusoidal wave) of the continuous signal $f(t)$ analyzed at the frequency ω
- ω is the frequency
- $e^{-i\omega t}$ is a complex exponential function with a real part and an imaginary part that depend on the frequency ω and the time instant t. Meaning it is the mathematical representation of a sinusoidal wave with frequency ω that changes with time t

When one integrates $f(t)e^{-i\omega t}$ over the full time interval, we are basically calculating how much the sine wave with frequency ω contributes to the overall signal $f(t)$. Thus, $F(\omega)$ tells us the amplitude of this sine wave component at frequency ω in the original signal.

Each ω represents a different frequency. Therefore, when we apply this integration to all possible frequencies, we obtain a series of amplitudes for the different sinusoidal components. This series of amplitudes represents the frequency spectrum of the signal, in other words, how many and which frequencies are present in the signal.

There is also the Inverse Fourier Transform (IFT). This technique allows the FT to be reproduced in reverse. The IFT allows a signal to be reconstructed in the time domain from its frequency spectrum. Its formula is as follows:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{i\omega t}d\omega$$

Nowadays, the Fast Fourier Transform (FFT) is mainly used compared to the FT. It is an algorithm for obtaining the FT of discrete time signals with a very high computational speed compared to simple FT. In general, the FFT is used because it is the discrete function of the FT, in reality we will never have a continuous signal but will have discrete sample observations.

1.2.1.2.2 Cepstrum Analysis

Cepstrum Analysis can be defined as the power spectrum of the logarithm of the power spectrum [4].

This instrument is used to detect periodicities in a frequency spectrum. These periodicities are usually given by the fundamental frequencies, the harmonics of the fundamental frequency and the presence of sidebands spaced at equal distances around the fundamental frequency. In gearboxes, for example, these sidebands represent gear or pinion faults (if the logarithm of the amplitude is seen to increase at these frequencies) and are equally spaced around the gear meshing frequency. In any spectrum, these periodicities are difficult to detect because they are disturbed by the presence of other elements and noise. Cepstum Analysis facilitates the detection of periodicities through noise cancellation.

In Cepstrum Analysis, the horizontal axis represents frequencies, similar to the power spectrum, but the vertical axis represents a distance in the frequency domain.

Cepstrum Analysis has advantages such as much better detection of periodicities in the signal and the detection of transient events. However, it is a noise-sensitive technique and complex to apply. The complexity is given by the proper selection of parameters and the interpretation of the spectrum.

1.2.1.2.3 Envelope Analysis

Envelope Analysis, also referred to as amplitude demodulation or demodulated resonance analysis, isolates the low-frequency signal from surrounding background noise. This technique finds extensive application in the diagnosis of rolling element bearings and low-speed machinery. One of its notable advantages is its capability for early detection of bearing issues [4]. Envelope Analysis is a technique used in signal analysis to extract the dominant amplitude information and its variations from a complex signal. This technique is particularly useful for identifying amplitude variations over time. The basic idea is to extract the low-frequency part of the signal to simplify the identification of amplitude variations. Graphically, it represents a curve that follows the amplitude variations of the signal, isolating them from the original signal and highlighting transient events.

Envelope Analysis can be performed using three different approaches based on the application case:

1) Demodulation: often also called amplitude demodulation, it should not be confused with the demodulation known from carrier wave modulation techniques. In this context, it refers to a process of extracting dominant amplitude variations. A constant-frequency oscillator is often used to demodulate the amplitude of the original signal, capturing low-frequency amplitude variations.
2) Hilbert Transform: The Hilbert Transform is a linear transformation that works on a real function and produces a complex function. The output of the transform represents the original signal together with its imaginary component. The amplitude of the transform output represents the envelope of the original signal.
3) Band-pass filter: A bandpass filter allows a portion of the frequencies of a signal to be isolated. Since envelope analysis focuses on highlighting amplitude variations at low frequencies, it is sufficient to apply a low-pass filter. In this way isolating the signal components at low frequencies while eliminating those at high frequencies. After that, the amplitude variations are extracted and analyzed.

It is important to remember that one of these approaches does not exclude the others. It is appropriate to use these methods in combination. A good choice would be to apply a low-pass filter and then apply demodulation. This would isolate the lowest frequencies, after which the amplitude variations would be extracted with demodulation.

Envelope Analysis is a useful technique, especially for identifying transient events that may be synonymous with faults or failures. However, it is a very noise-sensitive technique and sensitive to the choice of the frequency band to be isolated. Wrong filter bandwidths can exclude frequencies that contain significant information. Furthermore, the interpretation of amplitude variations can sometimes be complex and require the intervention of an expert.

In general, there is no one method that works better than another, everything depends on the application case.

It can be deduced that analyzing the signal in the frequency domain has numerous advantages. Primary among these is the fact of being able to reconstruct the spectrum of the signal in all its frequencies. By being able to analyze individual frequencies, one is able to detect which frequencies are abnormal and associate them with a localized fault. In addition, the frequency domain allows for very accurate signal filtering techniques, isolating the band of frequencies in which one is interested.

However, this type of analysis also has disadvantages. First, analysis in the frequency domain completely loses the temporal information of the signal. Furthermore, analysis in this domain often involves complex mathematical calculations. This fact consequently affects the need for a high computational capacity.

The techniques discussed earlier for vibration analysis, whether in the time domain or frequency domain, primarily rely on the assumption of stationarity. However, this assumption prevents the simultaneous identification of localized features in both the time and frequency domains. As a result, these methods are inadequate for the analysis of signals that change over time. This constitutes the primary drawback of analyses conducted in the time and frequency domains. [4]

1.2.1.3 Time & Frequency Domain Analysis

Time-frequency domain techniques can handle both stationary and non-stationary vibration signals [11]. This is the main advantage over time-frequency domains.

With this type of analysis, one has the three information of time, frequency and amplitude plotted on a single graph. Generally, you have on the x-axis the time, on the y-axis the signal frequencies and on the z-axis the various amplitudes. It is a very important tool in vibrational analysis because it simultaneously maintains information about the time and frequency of a signal. It combines the two analyses described above at the same time.

The main techniques in the time & frequency domain are: Wavelet Transform (WT), Hilbert-Huang Transform (HHT), Short Time Fourier Transform (STFT) and Wigner-Ville Distribution (WVD).

1.2.1.3.1 Wavelet Transform (WT)

The signals are often characterized by abrupt and sudden changes called 'transients'. The Fourier Transform is a powerful tool, however the spectrum is not localized in time and frequency at the same time. The Wavelet Transform is a tool capable of analyzing these transients in a localized manner in both time and frequency.

WT consists of a family of elementary functions (as a function of time) that can be independently dilated and shifted, known as wavelets [3]. There are two key concepts of

wavelets, namely: scaling and shifting. Scaling refers to being able to shrink or relax the wavelet in order to capture the various peaks in the signal. Clearly, the more one shrinks it, the higher the frequency to be captured and vice versa. Shifting refers to the fact that one can shift the wavelet in time by sliding it over the signal in order to analyze any portion of interest. Since these elementary functions can be scaled and translated, the basic idea is to use these wavelets adapted to the shape of the signal concerned and use them to capture all the information we are interested in.

Based on how wavelets are scaled and translated and how they are used, wavelets can be distinguished into Discrete Wavelet Transform (DWT) and Continuous Wavelet Transform (CWT). The CWT involves a continuous scale of wavelets covering the entire frequency spectrum. While DWT means dividing the signal into discrete time intervals and analyzing each of them by scaling and shifting the elementary functions. The difference between the two resides in the fact that CWT allows better frequency resolution, while DWT shows better time resolution. Furthermore, DWT is computationally more efficient because it allows the signal segmentation without guaranteeing continuous coverage of the entire frequency spectrum.

1.2.1.3.2 Hilbert-Huang Transform (HHT)

HHT consists of empirical mode decomposition (EMD) of signals and Hilbert transform, and by combining these two methods, a Hilbert spectrum can be obtained, where the faults in a running machine can be diagnosed [4].

First of all, EMD must be applied to the signal. Empirical Mode Decomposition (EMD) is an adaptive time-frequency technique for analyzing nonlinear and nonstationary signals, which can decompose the time domain signal into a set of oscillatory functions in the time-domain, called intrinsic mode functions (IMF) [13]. Any complex and real signal can thus be decomposed into its IMFs. The process of generating an IMF repeats itself iteratively until a stop criterion is satisfied. Each intrinsic mode has its own associated time and frequency scale. This decomposition allows a more accurate analysis of signal variations at different scales. Each intrinsic mode represents a characteristic frequency and can be interpreted as a local mode function. The sum of all IMFs represents the original signal that has been decomposed.

EMD analysis is very sensitive to noise and inaccurate in the low-frequency region, therefore Ensemble EMD (EEMD) was introduced, which smooths out the noise by introducing random variability into the data [4].

After decomposition of a signal through the EMD, the Hilbert Transform is applied to each IMF. The Hilbert Transform is a linear transformation that works on a real function and produces a complex function. The Hilbert Transform allows the phase and amplitude components to be obtained for each intrinsic mode. This combined approach allows the temporal and spectral characteristics of each mode to be analyzed separately.

To summarize, the EMD allows the signal to be decomposed at different time and frequency scales (mainly in time), after that the Hilbert Transform at each IMF provides the phase and amplitude information, overall allowing the frequency content and its time evolution to be examined. Breaking down the signal on different time and frequency scales makes it possible to analyze non-stationary signals that also show abrupt local variations.

This method requires considerable computational capacity, especially for the application of EMD.

1.2.1.3.3 Short Time Fourier Transform (STFT)

STFT has the ability to counter the limitations of FFT and is mostly applied to extract the narrowband frequency content in nonstationary or noisy signals [4].

The concept behind STFT is to examine the segmented signal in distinct portions, one by one, using a type of moving 'window'. This window moves through the signal examined and divides it into sections or segments of the same length. The signal within each section is assumed to be stationary. Then, the Fourier transform is applied to each segment or window to reveal the frequencies within that specific part of the signal. [13]

In this way, one is able to keep track of both the temporal content by windowing at a given length and the frequency content by applying the Fourier Transform to each window. There are many types of windows that can be used, from rectangular to Hamming (similar to a Gaussian). The choice of window is very important, especially because they affect the signal queues in the signal segment to which the window is applied.

Figure 7: Short Time Fourier Transform [13]

For a signal $x(t)$, its STFT can be calculated as:

$$STFT(f,\tau) = \int_{-\infty}^{\infty} x(t) \cdot w(t-\tau) \cdot e^{-j2\pi ft}\, dt$$

Where:

- W(t) is the window centered at τ and T long
- τ is the location parameter of the window
- $e^{-j2\pi ft}$ is the phase term used in the Fourier Transform

What is important in STFT is the presence of the tradeoff between temporal and frequency resolution. The temporal and frequency resolution depend on the choice of the window, a narrower window (which can be of the order of a millisecond for example) guarantees a high temporal resolution but a poorer frequency resolution. A wide window, on the other hand, has a high frequency resolution and poor temporal resolution.

STFT is a very powerful tool because it allows you to obtain information in both the time domain and the frequency domain simultaneously. In addition, this technique is easy to implement and does not require great computational capacity. However, it has some disadvantages. The main drawback of this approach is that it cannot achieve high resolution in the time and frequency domain simultaneously [4]. The trade-off between resolutions cannot be avoided. The choice of time window affects the characteristics of the STFT, especially affecting the queues of the signal segment. Different windows may be suitable for different situations, but the optimal choice is not always clear. In addition, the frequency resolution is constant throughout the analysis. Once chosen, the length of the window remains the same throughout the analysis, thus ensuring constant resolutions in time and frequency throughout the signal.

1.2.1.3.4 Wigner-Ville Distribution (WVD)

This technique is used to analyze complex, non-stationary signals and was introduced to overcome the STFT limit for the trade-off between temporal and frequency resolutions.

Wigner-Ville Distribution is a specific case of the Cohen class distributions which yields a time-frequency energy density computed by correlating the signal with a time and frequency translation of itself [4]. WVD is based on the Wigner transform, which involves calculating the convolution between the original signal and a delayed, advanced version of itself. The basic idea is that WVD calculates the energy distribution of the signal based on time delays (translations) and frequencies. So, this technique keeps track of temporal information as it considers phase changes between the original signal and advanced, delayed versions. At the same time, it also tracks frequency content as it examines how different frequencies contribute to signal energy at different time delays.

The Wigner-Ville Distribution (WVD) formula for a signal $x(t)$ is given by:

$$W_x(t,f) = \int_{-\infty}^{\infty} x^*\left(t + {}^{\tau}/_{2}\right) x\left(t - {}^{\tau}/_{2}\right) e^{-j2\pi f t} d\tau$$

Where:

- $x(t)$ is the signal in the time domain
- $W_x(t,f)$ represents the Wigner-Ville distribution of the signal $x(t)$ at time t and frequency f
- $x^*(t)$ represents the complex conjugate of $x(t)$
- τ is the temporal delay

This technique achieves excellent performance in terms of resolution. However, it suffers greatly from the effects of cross-term, which can cause significant interference in the output of the WVD. This effect is extended especially when the signal contains multiple frequency components and when the environment is noisy.

In conclusion, vibration analysis proves to be an innovative and effective diagnostic tool for a wide range of applications from industrial maintenance to medicine. Through an accurate vibration analysis, it is possible to diagnose and predict potential faults on machines, thus enabling the entire plant to operate more efficiently in the long term. This type of analysis is

enjoying success in all its applications due to its non-destructive nature and because the sensors used do not disturb the normal functioning of the examined machine.

The main signal processing techniques all demonstrate to be very versatile and effective in extracting meaningful information from vibrational signals. The use or non-use of a technique in reality depends only on the application case.

The time domain techniques only can indicate the fault(s) present in the bearing but it can't identify the location [11]. However, time domain techniques are the computationally fastest and also have the characteristic of being able to summarize the behavior of a signal over time in only one or a few indicators.

Frequency domain techniques have ability to identify the location of fault(s) in bearing. Vibration peaks generates in spectrum at the bearing characteristics frequencies, from that we can easily understand which bearing element is defected [11]. However, the identification of the bearing faults by using frequency analysis is difficult because, it is not suitable for non-stationary signal analysis [13].

Usually, nonstationary signals contain abundant information on machine faults and time and frequency domain techniques for machine monitoring are based on the assumption of stationary signals [4]. Time & frequency domain techniques can analyze and processing even non-stationary signals, which is why their use has been so popular. However, these methods require high computing power even for modest signals and samplings.

Each technique has its advantages and disadvantages. The combined use of several signal processing methods is recommended to obtain a better understanding of the signal.

In the next Chapter, referring to Figure 1, the last stage of vibrational analysis will be analyzed. The Fault Recognition step is essential to achieve useful and profitable predictive maintenance. It will also illustrate why Artificial Intelligence is needed to support this stage and which Machine Learning algorithms are most successful in doing this.

CHAPTER 2 – Machine Learning approach

Referring to Figure 1, the last step of vibration analysis for a predictive maintenance approach is Fault Recognition. This step consists in recognizing some kind of premonitory signal of a future fault on the machine. Basically, the signal processing stage produces numbers or graphics summarizing the behavior or certain characteristics of the signal. The signal taken into consideration is initially the raw signal generated by the vibrations of the machine. Consequently, the signal processing phase takes this signal and processes it in the time and frequency domain in order to deduce the operating state of the machine. Thus, the Fault Recognition phase takes as input what is the output of the signal processing phase and, by analyzing the correlations between the numbers and signal characteristics, is able to realize what the machine's operating state is. If there is an anomaly on the bearings, gearbox or any other machine gear that generates a small increase in vibration, this stage makes it possible to recognize the anomaly and thus predict a deterioration in the machine state. Sometimes these anomalies, in their initial phase, generate vibrations that are not perceptible to the human ear, at least until the anomaly turns into a critical fault such that unorganized maintenance has to be carried out, stopping production. This makes it possible to intervene promptly and safely with targeted, organized and effective maintenance.

Machine learning has evolved into a powerful tool that implements high quality intelligent algorithms for prediction. A feature of ML is the ability to process big data and therefore effectively find hidden correlations between them. [14]

The use of artificial intelligence in vibration analysis for machine monitoring and diagnosis is gaining increasing popularity. This phenomenon is explained by the fact that the previously mentioned methodologies require considerable expertise to be effectively implemented and interpreted, thus making them unsuitable for use by common users. Moreover, the presence of an expert in the field is not always available. In this scenario, methods based on artificial intelligence come into play, as they allow a non-specialist user to take reliable decisions without necessarily depending on the presence of an expert in the field of machine diagnosis. [4]

Machine learning is classified into three types: in supervised learning, where the predictors and response variables are known for building the model, in unsupervised learning, only

response variables are known, and in reinforced learning, where the agent learns actions and consequences by interacting with the environment [15].

Any machine learning algorithm must be trained and tested before it can be used in any application. In particular, there are common steps in the training of any algorithm. First, there is the collection of data and its examination to understand its nature and characteristics. After processing and elaborating the data to prepare it for the training, we move on to choosing the algorithm and defining its parameters. At this point, it is time to start training the model. To train a model, it is essential to have a set of data for training and a set of data to test the model's performance. During the training process, the model will learn to recognize correlations and patterns between features by changing its parameters and minimizing the prediction or classification error. At this point, the test data set is used to obtain the model's performance. Finally, if the model's performance is not satisfactory, optimizations are applied by moving backwards and modifying the model's architecture and parameters. A common problem with all Machine Learning models is overfitting. Overfitting occurs when the model fits the training data too tightly by learning the inherent randomness and variability of the data. As a result, the model when processing real data that does not belong to the training set generates poor performance compared to the test phase. To work around this problem, one can use a larger dataset, apply cross-validation to select the best model parameters or one can split the dataset into three parts (training, validation and testing) using the validation step to optimize the parameters.

The most used algorithms in the Fault Recognition phase are: Random Forest, Support Vector Machine, Artificial Neural Networks, K-means, Fuzzy Logics and Decision Trees.

2.1 Decision Tree (DT)

The decision tree algorithm represents a supervised learning approach used to investigate classification and regression problems. In the context of data mining, the commonly employed procedures involve classification techniques. Classification algorithms are capable of handling considerable amounts of data. These tools are used to formulate assumptions about label categories, to divide knowledge based on training data and classes, and to classify newly obtained data. [16]

A decision tree represents a structured technique, where any path from the root is delineated by a sequence of data separation operations, until a Boolean outcome is achieved in the leaves of the tree. Each tree is composed of nodes and branches. Each node represents the characteristics of a category to be classified, with each subset defining a value that can be associated with that node. The leaves of the tree correspond to the output categories. DT is a model that unites a series of the basic test efficiently and cohesively where a numeric feature is compared to a threshold value in each test [17]. The algorithm begins by choosing the characteristic (attribute) that best distinguishes the different output categories, with the goal of selecting the one that maximizes the separation between the classes. Once the optimal characteristic has been identified, a decision node related to it is created, while the different values assumed by the characteristic are used to create the branches of the tree. Subsequently, the dataset is divided into subsets based on the different values of the chosen feature. This process of selecting the best feature and splitting the data is repeated recursively for each decision node, until the condition in which all instances in a node are in the same class and the maximum depth of the tree has been reached. [16]

To summarize, decision trees construct a sequence of logical decisions to classify or predict data. Each decision is based on the characteristic that maximizes the separation between categories. The objective is to develop a tree structure that effectively represents the decision-making process in the dataset.

The use of the decision tree has become extremely popular in predictive maintenance due to its numerous benefits. These benefits include the short training period required, its intuitive comprehensibility, the absence of initial assumptions required, and its ability to handle both categorical and numerical data, even though it requires the generated attributes to be categorical in nature. However, it is crucial to be cautious when interpreting the results, as the logic behind the decisions made may be biased and incorrect choices may occur. [16]

2.2 Random Forest (RF)

Random Forest is a supervised learning algorithm widely used to solve classification and regression problems. It is essentially a decision tree-based classifier that selects the best classification tree through a voting process. Concretely, Random Forest operates through a set of decision trees, each of which is trained on a random sample of the training data. The main objective of a Random Forest is to reduce variance and prevent overfitting that might

occur using a single decision tree. This feature makes it one of the most powerful classification algorithms currently available. [18]

Random Forests are basically composed of multiple Decision Trees, which are the basic classifiers that makeup Random Forests. In particular, RF is a classifier consisting of a set of tree-structured classifiers with identically distributed independent random vectors and each tree casting a unit vote at input x for the most popular class [19]. A new independent random vector is generated, with a distribution similar to the previous ones, and a tree is created using the training set. Basically, when the sample to be classified is reached, the final outcome of the classification is decided by a vote on the Decision Tree. It takes the test characteristics and uses the rules of each randomly generated Decision Tree to forecast the result and store the expected result (target). Determine the votes for each predicted goal. Consider the predicted high-voted goal as the final prediction from the Random Forest algorithm.

In the Random Forest algorithm, there are two steps, one is Random Forest formation, and the other is to make a guess from the first step of the Random Forest classifier. Basically:

- Select "K" features at random from the complete "m" features, where k << m.
- Calculate the node "d" among the "K" features using the best split point.
- Using the best division to divide the network into daughter nodes.
- Repeat measures from 1 to 3 until the number of nodes 'n' has been reached.
- Develop a forest to build the "n" number of trees by repeating steps 1 to 4 for "n" number of times.

Following this step, the prediction process can be initiated. It involves taking the test features and employing the rules from each randomly generated Decision Tree to make predictions and record the expected outcome (target). For each prediction target, the algorithm counts the votes. The final prediction produced by the Random Forest algorithm is determined by giving more weight to the predictions that received the highest number of votes. [18]
The decision formula is:

$$H(x) = argmax_y \sum_{i=1}^{k} I(h_i(x) = Y)$$

Where:

- x is the test sample,

- h_i is a Decision Tree,
- Y is the output variable, or the class,
- I is the indicator function.

In other words, the classification result of each test tree for the test sample is summarized and the final classification result is the class with the highest number of votes [18].

Random Forests are widely used in various applications. In the area of maintenance, they can be used in machine status recognition for example. They bring the same advantages as Decision Trees but with an improvement over them, as they include the reduction of overfitting. However, being based on multiple decision trees, Random Forests require more computational capacity.

2.3 Support Vector Machine (SVM)

SVM is a commonly used supervised machine learning model and it is used for classification and regression [20].

This approach involves taking the dataset or sample space and mapping it into a higher-dimensional feature space through a non-linear transformation induced by a kernel function. Subsequently, it identifies the optimal hyperplane, which is defined as the hyperplane with the maximum margin. The data points from both classes that are in close proximity to this hyperplane and have an impact on its position and orientation are referred to as support vectors. The primary objective of Support Vector Machines (SVM) is to maximize this margin while ensuring that classification errors remain below a specified threshold. [4]

In the case where the SVM is used for regression problems, the idea is similar. However, instead of finding a hyperplane that separates the classes, the algorithm attempts to find a hyperplane that best fits the data to enable prediction. In both cases, the model is an optimization model.

Given two classes and if the input data can be separated linearly, the separation hyper plane can be shown by:

$$f(X) = w^T x + b$$

Where w is n-dimensional weight vector and b is the scalar multiplier or bias value.

The hyper plane that satisfies maximum margin between the two classes can be found by solving:

$$\begin{cases} Minimize\ \frac{1}{2}\|w\|^2 \\ Subject\ to\ y_i(w_i x_i + b) \geq 0 \end{cases}$$

SVM training is performed by solving the optimization problem in:

$$\begin{cases} Maximize\ \ L(\alpha) = \sum_{i=1}^{k} \alpha_i - \frac{1}{2} \sum_{i,j=0}^{k} \alpha_i \alpha_j y_i y_j k(x_i x_j) \\ Subject\ to \sum_{i=1}^{k} y_i \alpha_i = 0 \\ \alpha_i \geq 0\ for\ i = 1, \dots, k \end{cases}$$

Where:

- $k(x_i x_j)$ is kernel function,
- α_i are Lagrange multipliers.

On the other hand, if the input data could not be separated linearly, the kernel function changes in accordance with:

$$k(x_i x_j) = k(x_i x_j) + \frac{1}{C}\delta_{ij}$$

Where:

C is penalize parameter which affects the performance of the algorithm,

δ_{ij} is Kronecker symbol. [21]

One of the reasons why SVM is widely applied in vibration analysis for machine diagnosis is due to its compatibility with large and complex datasets such as data collected in the manufacturing industry [4]. SVM is very useful as the number of features of classified entities will not affect the performance of SVM. This means that for the base of the diagnosis system, there is no limited number of attributes that can be selected. there is no requirement for experts' knowledge in SVM, as is the case with fuzzy logic, and no layers are involved in SVM structure, compared to NN [4]. Furthermore, the usage of kernels allows SVMs to handle non-linearly separated data, thereby enabling the algorithm to be used for a wide range of more complex classification problems. On the other hand, the SVM requires high

computational capacity, especially in the case of very large datasets or complex kernels. The SVM may have difficulty handling datasets in which one class is strongly unbalanced compared to another one.

2.4 K-Nearest Neighbors (k-NN)

K-NN is an algorithm that is a non-parametric way for classification or regression. Specifically, in this method, the class of an object is the output which is shown by a majority vote of the nearest k-neighbor [8]. For example, if it acts like a dog, emits a dog-like sound, and looks like a dog, then the algorithm will classify it as a dog.

In the k - nearest neighbor rule, a test sample is assigned the class most frequently represented among the k nearest training samples. In other words this method is very easy to enforce for instance if an example "x" has k nearest examples where feature space and majority of them are having the same label "y", then "x" belongs to "y". If two or more such classes exist, then the test sample is assigned the class with minimum average distance to it [22]. When a new data item is presented for classification or prediction, the algorithm calculates the distance between the input data item and all instances of the training set. The algorithm selects the k points in the training set that are closest to the new input data based on the calculated distance. The size of k varies according to the case of interest. It is important to choose k carefully because it affects the performance of the algorithm. Choosing a k that is too small can mean greater sensitivity to noise or outliers, while a k value that is too large can make the algorithm less sensitive to variations in the data. For classification, the k-NN allocates the new data to the class that appears most often among the classes of the k nearest elements.

The distances used by the k-NN to find the nearest elements can be various. The most commonly used distance is the Euclidean Distance. The Euclidean distance between points p and q is the length of the line segment connecting them ($\overline{pq}$). In Cartesian coordinates, if $p = (p_1, p_2, \dots, p_n)$ and $q = (q_1, q_2, \dots, q_n)$ are two points in Euclidean n-space, then the distance from p to q, or from q to p is given by:

$$d(p, q) = d(q, p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

The position of a point in a Euclidean n-space is a Euclidean vector. So, p and q are Euclidean vectors, starting from the origin of the space, and their tips indicate two points. The

Euclidean norm, or Euclidean length, or magnitude of a vector measures the length of the vector:

$$|p| = p_1^2 + p_2^2 + p_3^2 + \cdots + p_n^2 = \sqrt{p \cdot p}$$

Where the last equation involves the dot product. A vector can be described as a directed line segment from the origin of the Euclidean space (vector tail) to a point in that space (vector tip). If we consider that its length is the distance from its tail to its tip, it becomes clear that the Euclidean norm of a vector is just a special case of Euclidean distance: the Euclidean distance between its tail and its tip. The distance between points p and q may have a direction, so it may be represented by another vector, given by:

$$q - p = (q_1 - p_1, q_2 - p_2, \dots, q_n - p_n)$$

In a three-dimensional space (where n=3), this can be visualized as an arrow extending from point p to point q. It can also be seen as indicating the position of q in relation to p. In certain contexts, especially when p and q represent the positions of the same point at two consecutive moments in time, it can also be referred to as a displacement vector. The Euclidean distance between p and q is just the Euclidean length of this distance (or displacement) vector:

$$|q - p| = \sqrt{(q - p) \cdot (q - p)}$$

Which is equivalent to the first equation. [22]

Another commonly employed distance metric is the Manhattan distance, also known as the Taxicab metric. In an n-dimensional real vector space with a fixed Cartesian coordinate system, the Manhattan distance, denoted as d_1, between two vectors p and q is calculated as the sum of the lengths of their projections onto the coordinate axes along the line segment connecting the two points .More formally:

$$d_1(p, q) = \|p - q\|_1 = \sum_{i=1}^{n} |p_i - q_i|$$

Where $p = (p_1, p_2, \dots, p_n)$ and $q = (q_1, q_2, \dots, q_n)$ are vectors. [22] [21]

Alternatively, the cosine distance is also used. The cosine of two vectors can be derived by using the Euclidean dot product formula:

$$a \cdot b = \|a\|\|b\|cos\varphi$$

Given two vectors of attributes, A and B, the cosine similarity, $cos\varphi$, is represented using a dot product and magnitude as:

$$Similarity = cos\varphi = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2} \sqrt{\sum_{i=1}^{n} {B_i}^2}}$$

The resulting similarity ranges from −1 meaning exactly opposite, to 1 meaning exactly the same, with 0 usually indicating independence, and in-between values indicating intermediate similarity or dissimilarity. [22]

Correlation between two variables is another method that can be used by the K-NN as a distance between elements. The distance correlation of two random variables is obtained by dividing their distance covariance by the product of their distance standard deviations. The distance correlation is [22]:

$$dCor(X,Y) = \frac{dCov(X,Y)}{\sqrt{dVar(X) \cdot dVar(Y)}}$$

K-NN offers various benefits, including its rapid computational speed and straightforward interpretability. It is also robust when dealing with noisy data, as it takes into account multiple nearby points rather than relying solely on an individual data point. Additionally, K-NN can address issues related to imbalanced datasets, as it focuses on a limited number of neighboring points rather than a rigid decision boundary between classes. Consequently, K-NN is well-suited for classifying scenarios involving a mixture of overlapping data points and cases where the examples have intersecting boundaries [22].

Nevertheless, K-NN does have its limitations. While it is a powerful and straightforward algorithm to implement, one of its primary shortcomings is its inefficiency when dealing with large-scale and high-dimensional datasets. This inefficiency stems from its 'lazy' learning approach, which lacks a dedicated learning phase and consequently results in a significant computational cost during classification. Additional drawbacks include its complete reliance on the training dataset and the absence of weighted distinctions between different classes. [22]

A new methodology called Weighted K Nearest Neighbor (WKNN) was developed to overcome the problem that there are no weights to differentiate classes. It is a squared inverse feature weighting technique that improves the performance of the k-NN classifier and can

optimize the computation complexity and classification accuracy [8]. Instead of simply considering the k closest data points uniformly, this method assigns different weights to each of them according to a certain function.

2.5 Fuzzy Logics

Compared to conventional logic, fuzzy logic aims at modeling the imprecise modes of reasoning to make rational decisions in an environment of uncertainty and imprecision [4].

The fuzzy logic system consists of four primary stages: fuzzification, inference mechanism, rule-base, and defuzzification component. In the fuzzification stage, input data, which can be numerical, is transformed into fuzzy sets. Subsequently, the fuzzy inference stage generates a dependable conclusion by utilizing rules established in the rule-base stage. Finally, the defuzzification stage yields measurable outcomes. Fuzzy logic is closely linked to membership functions, which play a crucial role in mapping non-fuzzy input values to fuzzy linguistic terms and vice versa. [4]

To create a diagnostic system with exceptional sensitivity, adjustments can be made to the rules and membership functions. Nonetheless, accurately defining the fuzzy rules and optimizing the membership functions pose significant challenges in the realm of fuzzy logic. In comparison to Support Vector Machines (SVM) and Neural Networks (NN), fuzzy logic is a simpler implementation. Moreover, unlike other AI methods like SVM and NN, it operates without the need for datasets, as it lacks training or testing phases. In specific scenarios, fuzzy logic may offer only a broad diagnosis, as it may not consistently identify the precise fault symptoms of a machine. Nevertheless, it serves as the only viable alternative when collecting fault data is infeasible. [4]

2.6 Artificial Neural Networks (ANNs)

Artificial neural networks (ANNs), often abbreviated to NN or neural networks, are a type of machine learning model inspired by the functioning of the human brain. They are widely used in predictive maintenance and many other applications for fault detection (fault recognition) and data analysis.

Artificial Neural Networks (ANNs) are computational models inspired by the neural architecture of the human brain. ANNs consist of interconnected processing units referred to as neurons. Typically, these neurons are organized into a sequence of layers, including an input layer, one or more intermediate layers, and an output layer. The input layer receives data input but doesn't perform computations itself. The output layer produces the network's response to the given input. The intermediate layers, also known as hidden layers, typically connect the input and output layers. Neurons in the hidden and output layers receive signals from all neurons in the layer above them, then perform a weighted summation and apply a transfer function to the inputs. Each connection between neurons is assigned a weight that signifies its significance. During the training process, data is passed through the network, and the connection weights are adjusted iteratively to minimize the error between the network's predictions and the training data. [23]

Neural network models are several and differ from each other on the basis of their architecture. The main differences between these architectures concern their structure, the way the neurons are connected, the activation functions used and the applications for which they are best suited. The choice of neural network architecture depends on the type of problem to be solved and the characteristics of the data. The most common and widely used architectures are: Single-Layer Perceptron, Multi-Layer Perceptron (MLP), Radial Basis Function Network (RBFN) and Recurrent Neural Network (RNN).

2.6.1 Single-Layer Perceptron Neural Network

The Single-Layer Perceptron is one of the simplest architectures of artificial neural networks (ANNs). It is also known as the "Single-Layer Perceptron" or "Single-Layer Feedforward Neural Network". The Single-Layer Perceptron consists of a single layer of input neurons, and a single output neuron. Each connection between an input neuron and the output neuron has an associated weight. It typically uses a binary activation function. Its major limitation is that Single-Layer Perceptron cannot learn or represent non-linear relationships in the data. It is a very simple and limited neural network model, suitable only for linear classification problems. Despite its simplicity and limitations, it is historically important because it was one of the first ANN models developed.

2.6.2 Multi-Layer Perceptron Neural Network (MLP)

Since single layer feedforward networks are limited to learning linearly separately patterns, nonlinear layers between the input and output are added to separate the data with enough training to model any well-defined function to arbitrary precision [24]. This model is known as a Multi-Layer Perceptron. The MLP architecture consists of one or more hidden layers of neurons connected to an input layer and an output layer. Each neuron in an MLP is completely connected to all neurons in the previous and next layer. This structure is called 'feedforward' because the data flow only moves in one direction, from the input layer to the output layer. The performance function of the forward network in the research is the mean square error, which is the average square error between the output of the final layer and the desired output.

Multilayer Perceptron (MLP) networks are trained using the Backpropagation (BP) algorithm. Backpropagation is a systematic approach for training multilayered feedforward neural networks. Its primary strength lies in its ability to find nonlinear solutions to complex problems. Backpropagation algorithms can be categorized into standard backpropagation and variants designed for faster training. Standard backpropagation utilizes a gradient descent technique, where weights are adjusted in the opposite direction of the gradient of the performance function. The gradient of the performance function is updated and propagated backward through the network layers until it reaches the input layer. A single iteration of this algorithm can be represented as follows:

$$x_{k+1} = x_k - r_k g_k$$

where x_k is a vector of current weights and biases, g_k is the current gradient, and r_k is the learning rate. Such traditional gradient-based training algorithms do not necessarily produce fast converges, although the performance function decreases most rapidly along the steepest descent direction. [23]

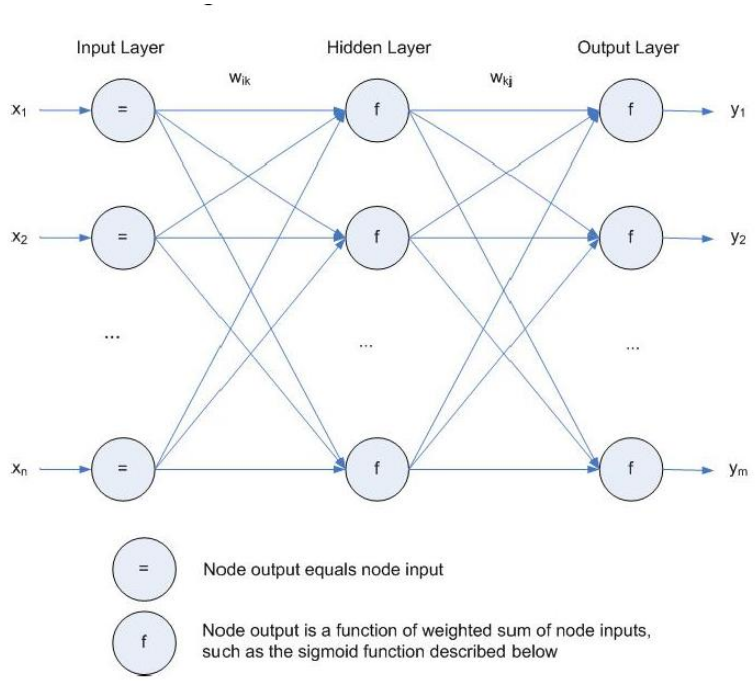


Figure 8: One-hidden layered MLP Network Structure [24]

2.6.3 Radial Basis Function Network (RBS)

As additional hidden nodes and layers are introduced, along with the nonlinear nature of the sigmoid activation function in each hidden and output node within Multilayer Perceptrons (MLPs), the Backpropagation algorithm can generate complex error surfaces containing numerous minima. Some of these minima are deeper than others, potentially causing the algorithm to become trapped in local minima rather than finding a global minimum. Local minima represent suboptimal solutions. In contrast, Radial Basis Function (RBF) networks were introduced with the use of data clustering to determine centers, which allowed for a

linear solution in determining connection weights directly to a unique local minimum, the global minimum. This approach significantly streamlines the network training process.

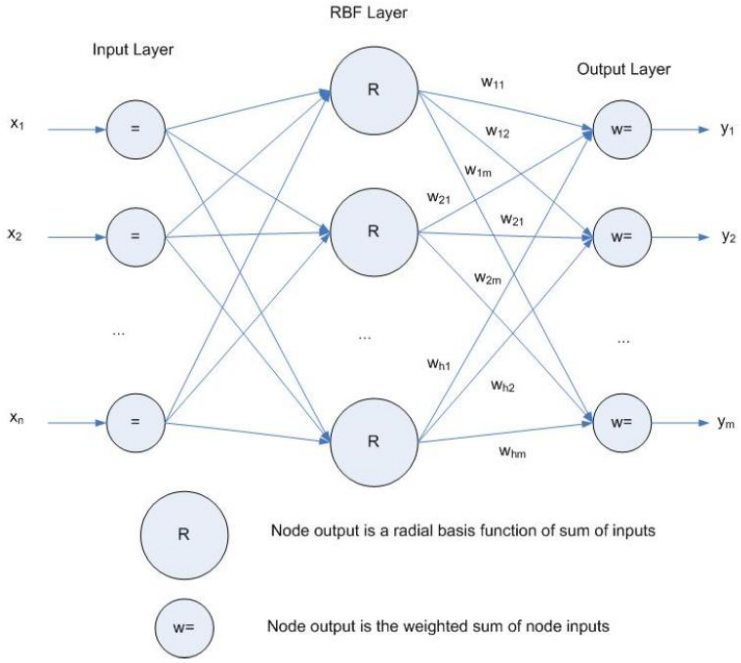


Figure 9: Radial Basis Function Network Structure [24]

The figure is a diagram of a one-hidden-layered RBF network structure.

The RBF network training also is supervised, involving determination of data clustering centers, c_k $(k = 1,2, \dots, h_c)$, where h_c also is the number of the centers to be determined, and network connection weights from the hidden layer to the network outputs, w_{kj} $(k = 1,2, \dots, h_c; j = 1,2, \dots, m)$. Similar to Backpropagation algorithm training, the RBF training is to establish the map between process inputs and desired outputs by minimizing the errors between the desired outputs and the calculated outputs driven from the inputs and network learning. The objective function of error minimization is defined as:

$$E = \sum_{t=1}^{N} \sum_{j=1}^{m} (d_{jt} - y_{jt})^2$$

Where this objective function obtains global minima if a linear solution of the output layer weights is found in the equation:

$$y_{jt} = \sum_{k=1}^{h_c} w_{kj} R(\left\| \sum_{i=1}^{n} x_{it} - c_k \right\|)$$

Where $R(...)$ is a Radial Basis Function.

A RBF function is any function R that satisfies $R(x)= R(\|x\|)$. It is a real-valued function that depends on the distance from the origin as $R(x)= R(\|x\|)$ or from other point or center, c, as $R(x-c)= R(\|x-c\|)$. The norm $\|...\|$ is usually Euclidean distance.

Radial Basis Function (RBF) networks incorporate nonlinear RBF activation functions within the hidden layer and utilize a linear combination of weights in the output layer. When it comes to clustering the input data, various techniques are available, including methods like k-means clustering and fuzzy c-means. [24]

2.6.4 Recurrent Neural Network (RNN)

A recurrent neural network (RNN) is a subtype of artificial neural networks (ANNs) where network nodes can have directed connections. In this type of network, either the activation values of hidden nodes or the output values of the network are fed back into the input nodes, creating a cyclic structure. RNNs excel in modeling temporal dynamics in processes.

On the other hand, Multi-Layer Perceptron Neural Networks (MLP NNs) are effective at capturing the static relationships between input and output in a process (from x to y). However, they require a large number of input nodes, which can lead to extended computation times and susceptibility to external noise. In contrast, recurrent neural networks have garnered attention in the field of dynamic system identification because they offer solutions to the issues faced by MLP networks.

In time series analysis, ANNs can establish AR (AutoRegressive) function map between process input and output: $\{y(t-i)|i=1,2,\dots,p\} \rightarrow y(t)$. In process modeling and control, ANNs can help establish ARX (AutoRegressive with exogenous input) function map of process input and output: $\{y(t-i), x(t-j)|i=1,2,\dots,p; j=1,2,\dots,q\} \rightarrow y(t)$where p and q are the orders of time lags of y and x, respectively. [24]

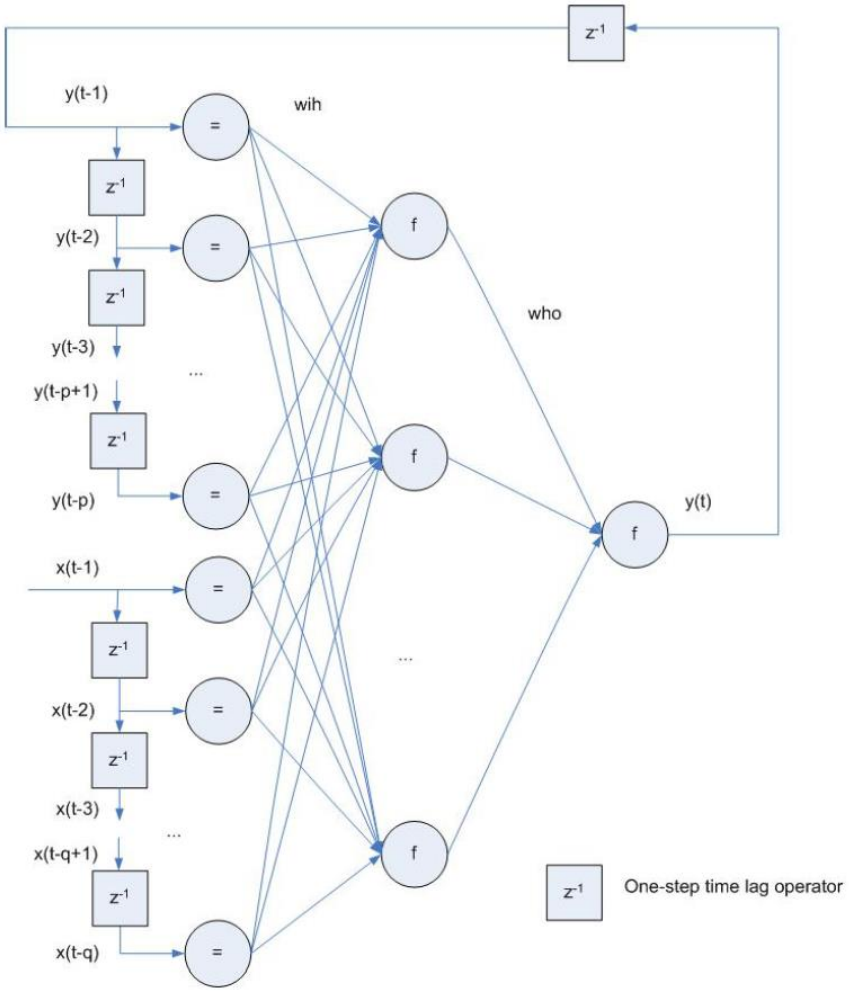


Figure 10: Recurrent Neural Network Structure [24]

The figure shows a network structure to map the dynamic process input/output relationship with the ARX function map. Internally, this network is identical to an MLP. Externally, it feeds back the network output back to the network input with a lagged time shift.

In summary, Recurrent Neural Networks (RNNs) belong to a category of artificial neural networks specifically designed for handling sequential data, making them well-suited for tasks involving time series data. Unlike feedforward neural networks, RNNs incorporate cyclic connections within the network, allowing them to retain a 'memory' of prior information as data is processed. Essentially, an RNN is comprised of a sequence of units, each equipped with two inputs: one from external sources and another from its own output during the preceding time step. This cyclic connection enables the network to capture information from past iterations. During the current time step 't,' the RNN combines the external input with the input from the prior time step 't-1' to generate an output. This output

can serve as the basis for computing the current time step's output or be utilized as input in the subsequent time step.

However, the RNN has limitations as, due to its sequential nature, calculations must be performed one by one, which can result in slower training. In addition, the error gradient can decrease significantly as it is back-propagated through the time steps, which makes learning long-term interdependencies sometimes impossible.

2.7 Performance Evaluation Metrics of Machine Learning Algorithms

Once the architecture has been defined and a machine learning model has been applied, it must be evaluated. Evaluating the performance of ML algorithms is a critical step in the design and implementation of predictive maintenance solutions.

There are many evaluation metrics. The main evaluation methods are accuracy, precision, TPR, TNR, FPR, FNR, F1 score, specificity, and Kappa values.

Accuracy is the ratio of the number of correct estimates to the total number of input observations:

$$Accuracy\ (ACC) = \frac{TP + TN}{TP + FN + TN + FP}$$

Where:

- $TP = True\ Positive$
- $TN = True\ Negative$
- $FP = False\ Positive$
- $FN = false\ Negative$

TP indicates the cases in which was predicted “Healthy” and the actual output was also “Healthy”. TN indicates the cases in which was predicted “Faulty” and the actual output was “Faulty”. FP indicates the cases in which was predicted “Healthy” and the actual output was “Faulty”. FN indicates the cases in which was predicted “Faulty” and the actual output was “Healthy”.

Kappa statistics not only show the performance of a single classifier, but also provide direct comparison for other models used for the same classification task. Kappa Metric can be calculated as:

$$Kappa = \frac{ACC - EA}{1 - EA}$$

Where Expected Accuracy (EA) is:

$$EA = \frac{(TP + FP)(TP + FN) + (FN + TN)(TN + FP)}{N^2}$$

In which N represents the Total Number of observation and can be calculated as:

$$N = TP + FN + FP + TN$$

Precision is obtained by dividing the number of correct positive results by the number of positive results predicted by the classifier:

$$Precision = \frac{TP}{TP + FP}$$

TPR is the number of correct positive results divided by the number of all samples identified as positive:

$$True\ Positive\ Rate\ (TPR\ or\ Recall) = \frac{TP}{TP + TN}$$

True Negative Rate (TNR) is:

$$True\ Negative\ Rate\ (TNR\ or\ Specificity) = \frac{TN}{TN + FP}$$

False Positive Rate (FRP) can be represented by:

$$False\ Positive\ Rate\ (FPR) = \frac{FP}{FP + TP}$$

While False Negative Rate as:

$$False\ Negative\ Rate\ (FNR) = \frac{FN}{FN + TP}$$

F1 score is the harmonic mean of TPR and precision:

$$F1\ Score = \frac{2}{\frac{1}{Precision} + \frac{1}{TPR}}$$

Accuracy, precision, specificity, True Positive Rate (TPR), True Negative Rate (TNR), and the F1 score are performance metrics used to evaluate classifiers. Ideally, these metrics should approach a value of one, indicating high performance, while False Negative Rate (FNR) and False Positive Rate (FPR) should approach zero. The Kappa statistic, on the other hand, can range from -1 to +1. A Kappa value of -1 suggests perfect disagreement between two observers, while +1 signifies perfect agreement. A Kappa value of 0 indicates that the level of agreement between the two observers is no different from what might occur by chance. [25]

In the ML classification problem, the most common method to signify classification accuracy is to prepare the Confusion Matrix, also known as the error matrix, since it describes the complete performance of the model [25]. Its main function is to show how good a model is able to classify the different classes of an application case. In practice, the Confusion Matrix contains all the metrics such as TP, FP, FN and FP needed to calculate the metrics outlined above.

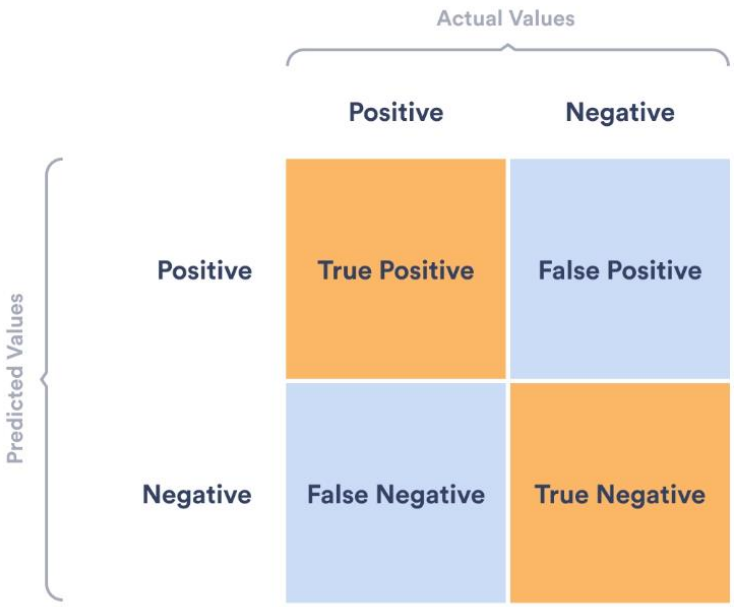


Figure 11: Confusion Matrix [26]

In this Chapter, the crucial role that machine learning algorithms play in predictive maintenance was explored. In particular, it was highlighted that an approach based on AI methods is crucial in the Fault Recognition step. Through the analysis of the main

algorithms, including Decision Trees (DT), Random Forests (RF), k-Nearest Neighbors (K-NN), Artificial Neural Networks (ANN) and Support Vector Machines (SVM), the different approaches and their applications were highlighted.

Machine learning algorithms offer significant potential for improving predictive maintenance in various industries, allowing imminent faults or hidden defects to be identified in a timely manner. However, it is important to recognize that the choice of the right algorithm depends on the specific application and the nature of the available data. In some cases, an approach based on neural networks might be more suitable, while in others, an algorithm based on decision trees might be the best choice. The ability to adapt the machine learning approach to the specific requirements of the problem is crucial to achieve meaningful results. Furthermore, it was emphasized that the successful implementation of machine learning models in fault recognition does not only depend on the choice of algorithm, but also on the quality of the data, their preparation and the management of the training process. Indeed, to achieve a good result, Data Acquisition and Signal Processing are of crucial importance. The first must ensure clean, real-time sampling of vibrational signals without interfering with production. The second Signal Processing phase is necessary to transform the raw vibrational data into a signal cleaned from noise and to clearly highlight features and characteristics of the signal. It was found that the application of the time domain parameters is directly proportional to the applications of AI methods. This is because time domain features can improve the performance of AI methods and have a low computational cost, which would not put much computational burden on AI methods [4]. It is often appropriate to optimize the learning model by focusing on a combination of algorithms in order to mitigate the limitations of a single algorithm.

Data-driven methods such as Neural Networks, Support Vector Machines (SVM), and other techniques discussed in this Chapter rely on training with historical datasets provided to the algorithm. However, when entirely new datasets are introduced, there can be challenges related to generalization. Hence, it is crucial to train AI algorithms with appropriate, diverse, and well-optimized datasets. Additionally, it is advisable to test these AI techniques under various operational or environmental conditions, beyond what the training datasets encompass, to address the generalization issue. Even though these techniques have demonstrated strong performance in monitoring and diagnosing machine conditions, expertise in signal processing remains essential. [4]

CHAPTER 3 - Case Study and Methodology

This Chapter presents a case study applied to an established manufacturing reality such as RdM Group.

RdM Group is a leading manufacturer of recycled cartonboard and the largest producer in Italy, France, the Netherlands and the Iberian Peninsula. While coated board is mainly supplied to the European market, solid board is representing a leading segment worldwide including US and Asia. The Group has a global presence with three production mills, three of which are in Italy, one in France, one in Germany, two in Spain, two in the Netherlands and one in Sweden, and four sheeting centers, two of which are in Italy, one in Spain and one in the United States.

RdM Group cartonboard is mainly used as packaging for consumer goods like foodstuffs, pharmaceuticals, households' appliances, cosmetics and personal care products. The Group has two main business areas: White Lined Chipboard (WLC) and Solidboard. The former which is a coated cartonboard for packaging made of recycled fibers. The second which is cartonboard in high grammages well suited for specialty products, luxury packaging and the publishing market.

The production process consists of a sequence of stages:

- Raw materials: obtaining raw materials to be recycled
- Pulper preparation: at this stage the raw materials are treated to make them suitable for use
- Board Machine: continuous machine that handles the main steps, from the orientation of the paper fibers mixed with water to the coating of the cartonboard.
- Finishing and sheeting: where cartonboard can be rolled into smaller reels for direct dispatch or cut into sheets
- Shipping: shipping the product to the customer

Figure 12: RdM Group's Production Process

RDM Group business model replaces the idea of "waste" with "resource", aiming to reduce raw material consumption and increase efficiency in the use of materials. RDM Group mainly uses paper for recycling, coming from both the industrial and commercial sectors as well as from municipal collection systems. Indeed, 92% of fiber material used in the Group production is made of paper and cartonboard for recycling. Overall, the 84% of the total material used by RDM is made up of renewable materials. Sometimes, the input of virgin fibers in the production of recycled cartonboard is necessary to give the product the physical and mechanical characteristics necessary for becoming packaging.

Figure 13: Storage of raw materials

In the first stage, raw material is mixed with water to create a pulp of fibers. At this stage the raw materials are treated to make them suitable for use. Raw material bales are sent to the pulper in order to be mixed in water to make them suitable for cleaning. In the pulper, the recycled fibres are mixed, disentangled and suspended into water. The output of this process is 95% water and only about 5% fibres.

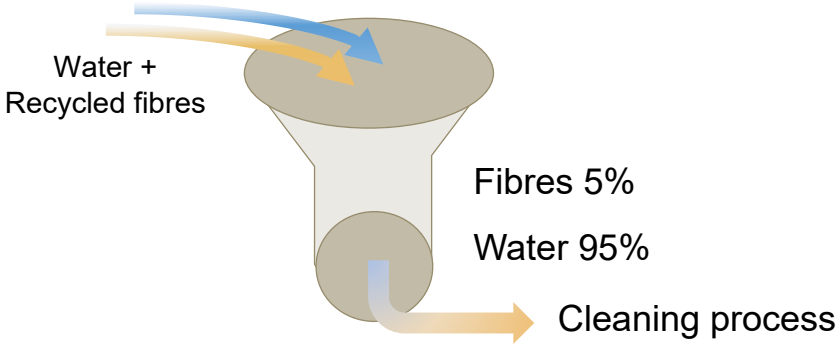


Figure 14: Pulper output

Figure 15: Pulper

Then, through a complex cleaning system, unwanted elements (e.g. sand, metals and plastics) present in the raw materials are eliminated. The fiber/water mixture coming out from the pulper has to be cleaned. At this stage, using the centrifugal principle and other filtering techniques, unwanted impurities are removed.

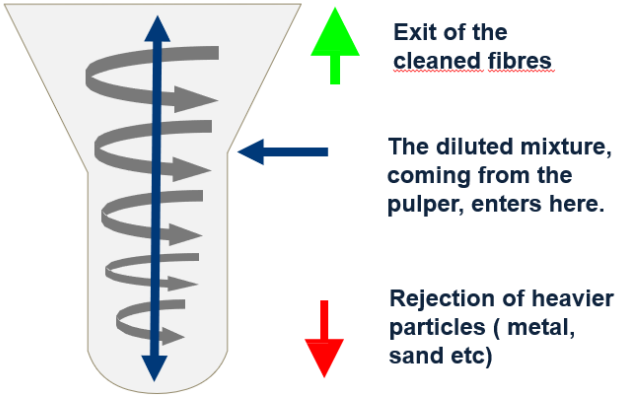


Figure 16: Centrifugal Principle

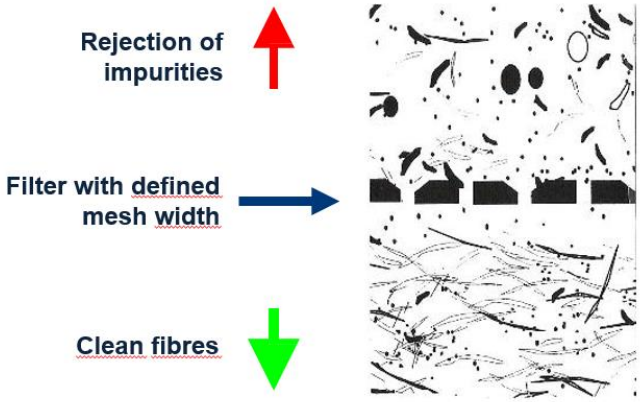


Figure 17: Filtration Principle

The board machine uses a sheet forming system, then dewaters the sheet by pressing and evaporating the water in a drying section. Depending on the type of cartonboard, a coating is applied. More specifically, the board machine step consists of four stages:

1) Wet or Forming Section: the fiber/water mix is pumped to the headbox, which homogenously spread the mix on to a continuously moving wire. Here the fibers orient, bind to each other and forms the fiber sheet while water drains to the below area.

Figure 18: RdM Group mill Forming Section

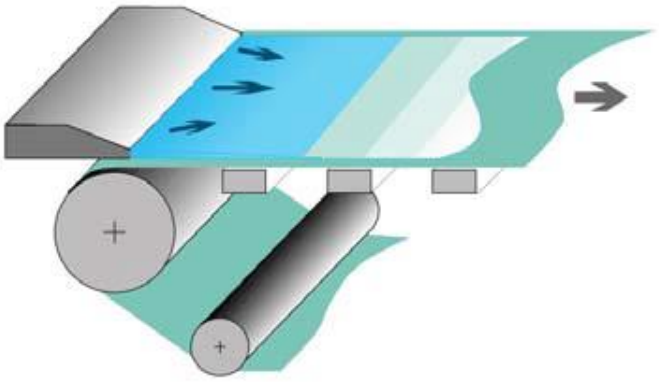


Figure 19: Forming Section

2) Press Section: the sheet is taken through the Press Section, pressed between rotating rolls for further water removal. The water is been removed by felts and the felts are again cleaned by uhle boxes (vacuum boxes).
3) Drying Section: the sheet is further dried with steam in the Drying Section, composed by several high temperature rolls in which the paper sheet passes accompanied by two felts.

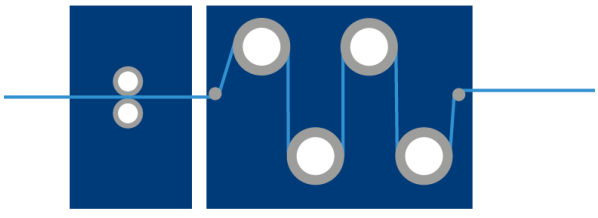


Figure 20: Press & Drying Sections

Figure 21: RdM mill Press Section

4) Coating Section: cartonboard can be supplemented with the coating treatment, which is a surface treatment laid down by deposition of CaCO3 / clay / starch, on one or both sides, to allow a better surface, appearance and printability.

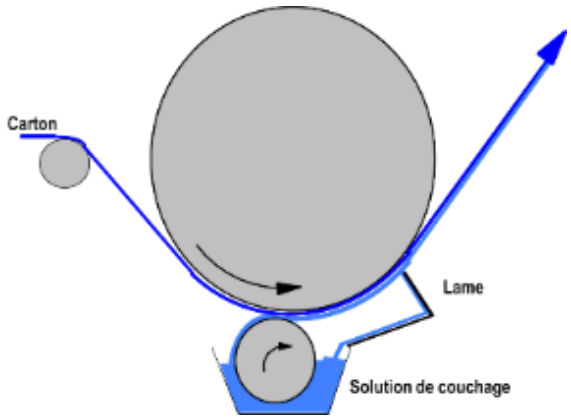


Figure 22: Coating Section

The output of the board machine is recycled cartonboard ready to be cut and shipped to the customer. More specifically, at the finishing and sheeting phase cartonboard can be rolled into smaller reels for direct dispatch or cut into sheets and packaged on pallets in the desired sheet formats to be supplied to the customer.

Figure 24: Cartonboard Reels

Figure 23: Cartonboard Sheets

Quality tests are carried out throughout the production process.

3.1 The Company's Necessity: A Predictive Maintenance Approach

One of the main targets of the company is minimizing waste and optimizing production efficiencies.

Accordingly, the Group has created a portfolio of projects called 'Operational Excellence'. Operational Excellence is a corporate approach that aims to continuously improve operations and business processes to achieve greater results in terms of efficiency, quality, customer satisfaction and overall value. This results in a series of target outcomes such as improving operational efficiency and enhancing product and service quality while satisfying the customer. Improving operational efficiency means reducing waste, reducing operating costs and improving the efficiency of business processes. Essentially, an Operational Excellence approach includes initiatives to improve production, procurement, inventory management, quality management, supply chain optimization, process automation and many other aspects. Operational Excellence is an evolving process, which is why it is also referred to as continuous improvement. Organizations are constantly looking to improve their processes This involves collecting and analyzing data to make well-informed improvement decisions. The overall goal is to increase the company's competitiveness, improve its resilience and ensure its long-term sustainability.

Among these projects, the company intends to change its approach to maintenance policy. In the coming years, RDM expects to introduce predictive maintenance methodology in all its mills. This is part of the need to improve production processes and production efficiency to create added value.

With the fourth industrial revolution, which has become increasingly widespread in the manufacturing industry, traditional maintenance has been replaced by the industrial internet of things (IIoT) based on condition monitoring system (CMS). The IIoT concept provides easier and reliable maintenance. Unlike traditional maintenance, IIoT systems that perform real-time monitoring can provide great advantages to the company. It is very important to detect faulty bearings before they reach the critical level during the rotation. The costs associated with maintenance represent a significant portion of a company's operating costs. This is particularly relevant in a context where global competition is growing steadily. One of the main objectives of manufacturers is to offer the best possible product at a lower cost. Factory operations using predictive maintenance have been shown to reduce maintenance activities by 25-30% and breakdowns by 35-45%, while increasing production by 20-25% and generating a return on investment. Therefore, it becomes essential for companies to constantly monitor the machinery and equipment used in production processes through condition-based maintenance and early detection of any faults. [25]

To the present day, the company has always adopted a policy of preventive maintenance. Preventive maintenance is an approach to the management of plant, equipment or systems that aims to prevent failures or malfunctions by scheduling regular and planned maintenance activities. The main objective of preventive maintenance is to extend the useful life of plant or machinery and improve its reliability. Predictive maintenance, also known as condition-based maintenance, is the type of maintenance in which the time of maintenance is determined by monitoring the system. Predictive maintenance is the newest type of maintenance that provides the longest equipment life and reliability, the least environmentally friendly and cost effective solutions. [25]

Cartonboard production for RDM is based on continuous production. In this context, it is essential to avoid downtime as much as possible. Plant shutdowns, whether due to machine failures or due to maintenance organized to prevent breakdowns even where the threat does not exist, wear down the company's business. Predictive Maintenance can therefore be a real solution for RDM. By predicting when the machine will break down, it is possible to intervene safely and minimize machine standstills.

The choice of adopting a predictive maintenance approach would therefore lead to numerous advantages for RDM:

- Reduced Maintenance Costs: Predictive maintenance allows organizations to plan and conduct maintenance operations only when these are really needed. With preventive maintenance, on the other hand, you do the intervention regardless if it is needed or not.
- Minimization of Downtime: identifying a critical issue before it occurs allows timely intervention, increasing the time available for production.
- Safety Enhancement: this approach helps to prevent a failure before accidents can occur and also helps the maintenance team to intervene in an organized and coordinated manner.
- Optimization of Spare Parts Stocks
- Improved Remote Monitoring
- Extension of Asset Useful Life
- Improved Operational Efficiency

Summarizing the need also arises from the desire to be more competitive in the market. Organizations that successfully implement predictive maintenance can become more

competitive as they are able to offer more reliable services, reduce costs and improve asset management.

3.2 RDM Ovaro

For the reasons stated, this project deals with a real application of a predictive maintenance approach. The project began by first focusing on a machine in one of the Group's mills. More specifically RDM. Ovaro. Ovaro mill is located at the foot of the Carnic Alps in northeastern Italy, close to the Austrian border. It has a production capacity of about 110,000 tons and provides employment for about 160 people, but the number multiplies if we consider outsourcing and downstream activities. Today, the plant has three production lines.

The cartonboard produced in Ovaro, usually made with high grammage, laminated and coated, is well suited for specialties, luxury packaging and publishing.

Ovaro mill was selected because it has been implementing sensors that measure engine vibrations in real time since a few years and therefore has a good historical data set.

Figure 25: RDM Ovaro mill

Electric motors are used in many areas from daily life to industrial environments. People use them in white goods, escalators, travel vehicles, power plants, production facilities, etc. Due to their critical duties in the workplace, electric motors need to be operated properly and maintained on time. Electric motor failures are basically divided into two groups: electrical

and mechanical failures. As vulnerable machine parts, bearings are among the most crucial components of electric motors as they connect the stationary motor (stator) with the rotating motor (rotor). In small and medium-sized motors, bearing damage constitutes the significant majority of potential faults. It is of paramount importance to prevent failures, minimize economic losses and ensure the safe operation of industrial plants. This is why reliable condition monitoring of electric motors is essential and has received considerable attention through condition monitoring. [25]

A rotating machine is any device or mechanism that turns or rotates to perform a specific function. An electric motor is a specific type of rotating machine that converts electrical energy into mechanical energy by rotating a shaft or drive shaft. Bearings are crucial components in rotating machines and are used to reduce friction and enable smoother rotation of the shaft or drive shaft.

The manufacturing process that was exhibited earlier highlighted that most of the machines in a paper mill are rotating machines. Therefore, a predictive maintenance approach becomes even more meaningful and important in this industry.

Ovaro mill recently encountered failures and critical issues on some bearings in an engine. Specifically, the Cilindro Aspirante Copertina of Board Machine 1 (BM1).

Since a significant amount of historical data representing the operating condition under both normal and critical (failure) conditions is available, the Cilindro Aspirante Copertina engine of Ovaro's BM1 was therefore chosen as the engine on which to perform the analysis for this project.

3.3 Cilindro Aspirante Copertina BM1

Board Machine 1 is one of Ovaro's three production lines. The forming section consist of 3 wires: top, middle and back. The BM1 produces, mainly, 9 board grades from 100% recycled paper.

Figure 26: Board Machine 1

The Cilindro Aspirante Copertina is a rotary machine. It is composed of the motor-gearbox-cylinder assembly. It is used to vacuum and collect water that drips from felts in order to obtain only recycled fibers.

The motor is a SIEMENS motor, number 127031/1993 and model 315L. It has an output of 132 kW and has a shaft of 80 mm diameter.

The motor shaft goes directly into a gearbox to allow the cylinder to rotate at a lower speed but to have higher torque. The gearbox is left orthogonal, CARCANO brand and model N-MAG8. It has a reduction ratio of 1/8.33. This indicates how much the rotational speed of the output shaft is reduced compared to the speed of the input shaft. Thus, the output shaft of the gearbox rotates 8.33 times slower than the input shaft.

The Cilindro Aspirante Copertina, on the other hand, presents a cylinder with a diameter of 840 mm.

Figure 28: Engine-Gearbox assembly

Figure 27: Cilindro Aspirante Copertina

The Cilindro Aspirante Copertina uses radial ball bearings. In particular, SKF model 6319/C3. Single row deep groove ball bearings are particularly versatile, have low friction and are optimized for low noise and low vibration, which enables high rotational speeds. They accommodate radial and axial loads in both directions, are easy to mount, and require less maintenance than many other bearing types [27].

The bearing in consideration has the following technical characteristics:

- Dimensions:
  - Bore diameter: 95 mm
  - Outside diameter: 200 mm
  - Width: 45 mm
  - Ball diameter: 33,34 mm
  - Number of balls: 8
  - Contact angle: 0°
- Properties:
  - Filling slots: Without
  - Number of rows: 1
  - Locating feature, bearing outer ring: None

- Bore type: Cylindrical
- Cage: Sheet metal
- Matched arrangement: No
- Radial internal clearance: C3
- Material, bearing: Bearing steel
- Coating: Without
- Sealing: Without
- Lubricant: None
- Relubrication feature: Without

- Performance:
  - Basic dynamic load rating: 159 kN
  - Basic static load rating: 118 kN
  - Reference speed: 7 000 r/min
  - Limiting speed: 4 500 r/min

The values for the diameter and number of balls and the contact angle were taken with autocad directly from SKF's website.

Figure 29: SKF 6319/C3 bearing

Bearing failures are one of the main reasons for failure in rotating machines. These failures are mainly caused by phenomena such as pitting, spalling, electro erosion and wear. Most of these problems occur due to surface damage to the bearing raceways or the rotating elements themselves [28].

As explained in Chapter 1, such surface defects can occur in two forms: localized defects and distributed defects. Using a predictive maintenance approach based on vibration analysis, in which raw data is acquired in real time from a sensor, it is possible to predict these bearing failures in time.

3.4 Sensors

RDM Group started a monitoring project in 2020. Several real-time monitoring systems have been installed to monitor product quality and production. In particular, sensors were installed to acquire and monitor the status of the machine. These sensors collect vibrational data from the motor depending on where they are positioned.

The sensors in consideration are Schaeffler OPTIME. The Schaeffler Group has been driving forward groundbreaking inventions and developments in the field of motion technology for over 75 years. With innovative technologies, products, and services for electric mobility, CO_2-efficient drives, chassis solutions, Industry 4.0, digitalization, and renewable energies, the company is a reliable partner for making motion more efficient, intelligent, and sustainable over the entire life cycle.

OPTIME sensors are non-intrusive uniaxial accelerometers. Uniaxial accelerometers are devices used to measure acceleration along a single specific direction, thereby enabling the vibration behavior of machines or structures to be analyzed.

The potential of these tools is that they are wireless and share information over a network. In this way, an operator can monitor the condition of the machine wherever he is and even through a simple smartphone. The sensors are activated via near field communication technology or (NFC). NFC is a form of short-range wireless communication that enables the transfer of data between compatible devices. All sensors connect automatically, both with each other and with the gateway, and together they form a mesh network, one of the most reliable and energy efficient IoT networks. Since the condition monitoring of machines in large industrial plants requires simultaneously bridging large distances and reaching

machines that are difficult to access, mesh technology was chosen. The actively managed mesh network is able to establish contact with sensors up to 100 m away, ensuring reliable communications.

The entire OPTIME Condition Monitoring System consists of the following components:

- OPTIME Gateway
- OPTIME Sensors
- OPTIME App

The gateway is a device that acts as a collection point. Its main role is to act as an intermediary between remote sensors and the main communication network, allowing data to be collected, processed and transmitted efficiently and reliably. The gateway is housed in a robust protective casing suitable for wall or surface mounting. The gateway should be mounted in such a way that the gateway-sensor passage is not obscured by any object that could obstruct data transmission.

The sensors are uniaxial accelerometers. The system allows two types of sensors:

- OPTIME 3 sensor: it has a bandwidth ranging from 2 Hz to 3 kHz. According to Nyquist's theorem, this means that the sensor can sample at a maximum of 3 kHz.
- OPTIME 5 sensor: has a bandwidth of 2 Hz to 5 kHz, therefore it can sample a signal up to a maximum of 10 kHz.

The Shannon-Nyquist theorem states that any signal can be reconstructed correctly if sampling on it is done at a frequency which is at least twice the maximum frequency (of the sine wave with the highest frequency) [29].

Figure 30: OPTIME Sensors

Due to their bandwidths, it can be claimed that the OPTIME 3 sensor is suitable for monitoring motors, generators and fans. OPTIME 5 sensors, on the other hand, can also be applied to pumps, geared motors, gearboxes and compressors. Clearly, the application depends on the characteristic frequencies of a system.

As Schaeffler states, the OPTIME system is suitable for machines that are operated continuously or partially continuously. Machines that are only run for short periods during the day are less suitable for monitoring with OPTIME. This criterion meets RDM's needs very well as its production is based on a non-stop line.

The sensors are mounted in proximity to the machine's contact surfaces in a radial direction to the load zone. During vibration monitoring, the structural noise of machines is measured, so there should be a stable connection to the contact points. This means that machine covering parts are unsuitable as a mounting location. However, if the bearing area is not accessible, the sensor can be mounted on a flat surface of the motor housing. The sensors are attached using special adhesives.

Some of the machines in the Ovaro plant are very large, so it was decided to use at least two sensors per machine.

Figure 31: Application of the sensor on the motor's fast shaft output

Figure 32: Motor-gearbox assembly with application of the sensors

In addition to vibration, the sensors also measure temperature. They also determine some useful parameters for understanding the condition of the electric engine. In particular, they calculate Root Mean Square, Kurtosis and the envelope curve.

The sensors sample the signal every day at 2 p.m. for a different duration depending on the sensor. OPTIME 3 sensors sample the signal for the duration of 1000 ms, while OPTIME 5 sensors sample for a duration of 3000 ms.

The raw data coming to the system depends on the bandwidth of the sensor. For example, if we consider a machine being monitored by an OPTIME 3 sensor, which has a maximum bandwidth of 3 kHz, more than six thousand observations will arrive at the system (again due to Nyquist's theorem). In fact, OPTIME 3 samples the signal for 1 second with a frequency of at least 6 kHz. This is a consequence, since it has a bandwidth of 3 kHz it means that digitally it can represent signal frequencies up to about 3 kHz, which in turn means that it will sample with a frequency of about 6 kHz and thus observe about six thousand observations per second. In the other hand, the OPTIME 5 sensor will send about thirty-five or forty thousand observations. This is because having 5 kHz bandwidth, it will sample with a frequency of at least 10 kHz. Thus, it will collect more than ten thousand observations per second for a duration of three seconds.

These sensors are powered by batteries, so it is good to check periodically that they have not run out of power. Battery life depends on various parameters and operating conditions such as ambient temperature, connection quality, and frequency of measurement intervals.

3.5 The failure

As mentioned previously, the Cilindro Aspirante Copertina BM1 machine recently exhibited a bearing failure.

To get to the goal of developing a system based on vibration analysis that can perform predictive maintenance, it is essential to meet certain requirements. First and foremost, having accurate real-time data acquisition. This requirement is met by OPTIME sensors. Secondly, it is necessary to elaborate and process the data in a functional and efficient manner. And finally apply Machine Learning algorithms for fault recognition.

A Machine Learning algorithm needs a training phase and a testing phase to evaluate the learning process itself.

In a real-world case such as this one where the aim is to achieve the implementation of a system that can predict a failure on the machine weeks in advance, the training phase of the algorithm becomes even more important. Indeed, it is essential that the training dataset, used to train the algorithm, presents all the states of the machine that one wants to predict. In practice, if one wanted to predict a fault, in order for the algorithm to recognize the fault state, it is necessary to present the algorithm a dataset showing the parameters of the machine state under normal, alarm (pre-fault) and fault conditions. Otherwise, the algorithm will never be able to classify and/or predict a failure.

To sum up, in real case like this, for the ML algorithm to recognize and predict a failure on the engine, it will necessarily have to be trained with a training set that presents the operating condition of the machine under both normal and failure conditions.

For this main reason the Clindro Aspirante Copertina machine was chosen, because it recently presented a failure on a bearing.

The defect occurred from mid-December 2022. Operators did not notice the defect, so production continued for another two months before maintenance was done. It is normal that the operators had not noticed anything since this defect did not involve an increase in vibration that could be captured by the human ear.

However, looking at the signal demodulation in Figure 33 one can notice that from mid-December to mid-February there was an abrupt increase in the signal captured by the sensors. Although this was undetectable through human visual inspection, this shows how the sensor had picked up the fault and therefore a predictive maintenance approach would have helped the plant operators.

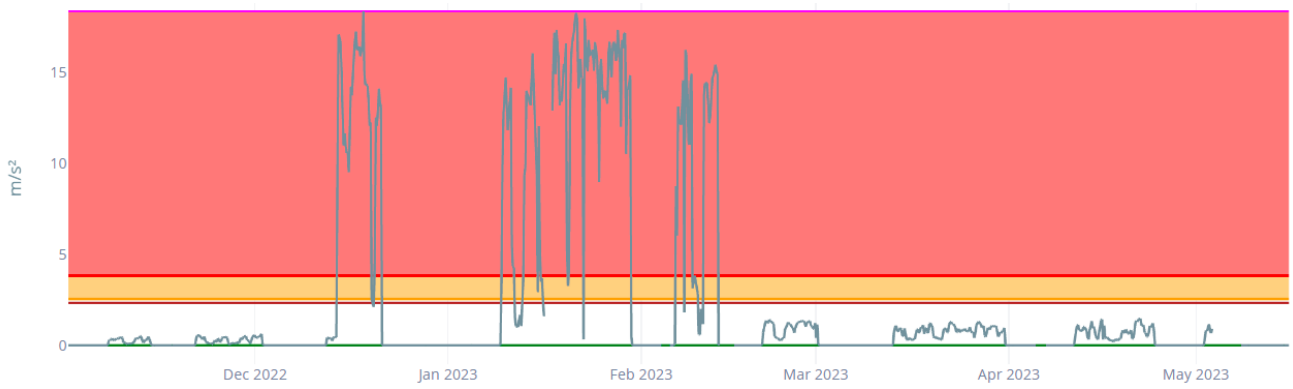


Figure 33: Demodulation of the signal highlighting the fault

The failure affecting the bearing in this case was a deformation of the bearing outer race. Specifically, small 'notches' were evident on the outer race of the bearing. These most likely appeared due to excessive loads or cyclic stresses repeated over time. Under these conditions the bearing is unable to work optimally, causing increased temperature and wear.

In February 2023, routine maintenance was done and upon disassembling the bearing they realized the notches on the bearing and the bearing was replaced with a new one.

Figure 34 shows the replaced failed bearing and the presence of the notches on the outer race.

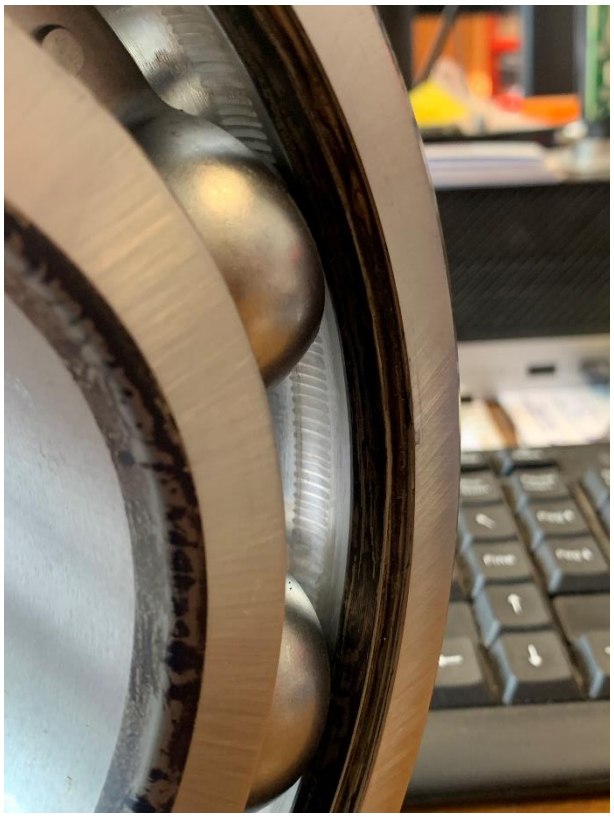


Figure 34: Faulty bearing

3.6 The Approach to Requirement

Thus, the company's need is to implement a predictive maintenance approach based on vibration analysis and monitoring given the technology the company currently possesses.

First, the goal and scope of the project must be defined.

3.6.1 Goal of the project

The objective was to develop a model which, through the application of vibration analysis and Machine Learning algorithms, could identify and classify the state of the machine.

As already mentioned, the machine taken into consideration for the development of the project was the BM1 Cover Suction Cylinder of the Ovaro mill.

In order to achieve the best possible result, it was studied which Dataset-ML algorithm combination was most efficient for the case study. The aim was to investigate which combination of Signal Processing technique and Machine Learning algorithm for Fault Recognition was the most efficient.

This helps to determine which Signal Processing technique is better in this application case. It should be recalled that there are different techniques that belong to three macro categories: time domain, frequency domain and time & frequency domain methods.

For this, several time domain techniques are applied, such as:

- Peak
- Peak-to-Peak
- Mean
- Root Mean Square (RMS)
- Crest Factor

Fast Fourier Transform (FTT) was applied as a technique in the frequency domain.

While Short Time Fourier Transform and Empirical Mode Decomposition (EMD) were applied as time & frequency domain methods.

Once all these Signal Processing techniques were applied, four Datasets were created:

1) A Dataset with only methods in the time domain.
2) A Dataset containing methods in the time domain and the output of the FFT as method in the frequency domain.
3) A Dataset containing time domain metrics and the outputs of the Short Time Fourier Transform as a method in the time & frequency domain.
4) A Dataset containing time domain metrics and the outputs of the FFT but subjected to a Denoising technique to reduce distortions caused by noise.

It is also evaluated which algorithm is better at recognizing the state of the machine. For this purpose, the following algorithms were implemented: Decision Tree, Random Forest, k-NN and Support Vector Machine (SVM).

Therefore, several Datasets were created because in this way it is evaluated which methods are more accurate to make a prediction of failure in this application. In fact, each Dataset was divided into a training set and a test set to train the ML algorithms. In this way, by evaluating the performance via the accuracy metric and error matrices, it is possible to determine which technical combination of Signal Processing and ML algorithm for Fault Recognition is best, thus establishing which approach will be used in the future to monitor and predict faults on the machine.

The whole project was developed using Python as the programming language to define the architecture and structure of the model. For codes that required high computing power, the Google Collaboratory platform was exploited. Google Colab is a cloud-based platform offered by Google that provides a free interactive development environment for running Python code. It is particularly popular among developers and researchers working on machine learning, data science and data analysis projects. This is because Google Collaboratory allows you to exploit the powerful computing resources offered by Google for free.

3.6.2 The methodology

The approach that was taken in order to achieve the project goal consists of several steps:

1. Analysis and study of the literature
2. Massive extraction of raw data
3. Signal Processing: processing and cleaning of the raw data
4. Dataset Design
5. Fault Recognition: selection and training of Machine Learning algorithms
6. Performance Evaluation of the Dataset - ML algorithm combinations for Fault Recognition

3.6.2.1 Analysis and study of the literature

The first step in implementing a predictive maintenance approach was to research and study the literature on the topic. Artificial Intelligence is now more than ever evolving; new techniques and algorithms are being developed every day. Therefore, it is essential to remain updated with the development of Machine Learning to be familiar with the latest approaches and methods used in the field of predictive maintenance.

The study was started from simple signal analysis to fully understand the nature of waves generated by vibration.

After that, several academic papers regarding vibrational analysis and its techniques were analyzed.

Finally, the focus turned to Machine Learning techniques for fault detection and machine state recognition.

Overall, more than 80 academic papers and articles were analyzed. The keywords of the research were 'Predictive Maintenance' and 'Vibrational Analysis'. All the literature taken into analysis belongs to the new millennium, specifically more than 50 percent of the literature was published from 2018 to the present.

3.6.2.2 Massive extraction of raw data

The first real practical stage is to collect the historical data.

The importance of historical data in the training of an ML algorithm has already been explained. Historical data must be reliable and must reflect the real evolution of the machine state.

By connecting to the OPTIME platform, it was possible to download the raw data.

First, we have to select the mill and the machine we are interested in. After that, you can download the raw data for one or more days in a csv file.

In this case study, the time frame from 01/01/2022 to 11/06/2023 was considered. The extraction required a lot of time as the instrument downloaded the raw data sampled by the OPTIME 3 sensor day by day.

After collecting the raw data, it had to be sorted and cleaned so that it can be processed by Signal Processing techniques. The cleaning and reading of the raw data will be explained in the next Chapter.

3.6.2.3 Signal Processing

Once the raw data has been collected and sorted, Signal Processing techniques must be applied.

Using these techniques, it is possible to obtain meaningful information about the condition of the engine or machine under observation. This step, in vibration analysis, is essential to obtain high-quality data and useful information from the measured vibration, preparing it for advanced analysis and identification of abnormal behavior or impending faults in the monitored engine or machine.

In this case study, many techniques of a different nature were applied to see which best suited the company's needs. Thus, raw data was processed by time domain, frequency domain and time & frequency domain methods.

In addition, a denoising technique was also applied to reduce noise. Being a real and not ideal case, it is natural that the signal collected by the sensors was interfered with by background noise in the environment. This could have been caused by operators passing by the machine or by the passage of machines transporting components. All these conditions of deviation from an ideal system cause interference during signal collection that could make the analysis misleading. For this purpose, a dataset subjected to a noise reduction technique was also created.

3.6.2.4 Datasets Design

One of the main goals of the project is to understand which of the signal processing techniques is best suited to train the ML algorithm to recognize the fault.

For this reason, four different datasets were constructed. In this way, it is possible to evaluate which one is most suitable for training the ML algorithms.

In constructing the dataset, not only the information gained from the applications of signal processing techniques on the raw data was considered. In fact, two other parameters

considered important in order to develop a predictive maintenance model were considered. Data on engine temperature and drive shaft rotation speed were also collected.

The temperature is measured directly by the OPTIME sensor. Indeed, it was expected that an increase in temperature may be related to the presence of an anomaly or defect during production.

Rotational speed was another input that was considered critical. Looking only at vibration by isolating it, it could be that an abrupt increase in the amplitude of the vibrational signal is correlated with the presence of a defect or anomaly. However, it could also be that an abrupt increase in the amplitude of the vibrational signal is related to an increase in production speed.

The production speed of a paper machine in a paper mill is varied several times during the day. The production speed at which Board Machine 1 is set depends on the machine load and the grammage of paper being produced. High grammages require lower production speeds and vice versa. It could be misleading for an ML algorithm to have only the vibrational signal as input without knowing the rotational speed of the drive shaft. For this reason, it was a parameter that was chosen to be used to populate the datasets.

The Ovaro plant does not have a sensor that measures the rotational speed of the drive shaft. Therefore, the latter was derived from the tangential velocity at which the felt flows on the cylinders of Board Machine 1. Knowing what the tangential velocity is, one is able to derive the rotational speed of the Cilindro Aspirante Copertina and through the reduction ratio of the gearbox one is able to derive the rotational speed of the drive shaft.

The formula used that makes it possible to derive the rotational speed of the motor shaft given the tangential speed of the output cylinder is as follows:

$$\omega_a = \left(\frac{(v_{c,o} * k)}{r_{c,o}}\right)\left(\frac{1}{2\pi * 60}\right)$$

Where:

- ω_a is the rotational speed of motor shaft in RPM.
- $v_{c,o}$ is the tangential speed of the output cylinder in meter per min.
- k is the gear ratio (here 8,33).
- $r_{c,o}$ is the radius of the output cylinder in meter.

In addition, all information regarding maintenance activities on the machine over the last year and a half was collected in order to obtain an accurate partitioning of the dataset. In fact, once maintenance has been carried out (which could also have been a simple lubrication) the average vibration levels drop and decrease. This factor was taken into account in the subdivision of the dataset.

A 'Status' column was created and populated based on the subdivision of the dataset. This reflects the status of the machine and motor. The column was populated for each date. In fact, in general, the date column was used to subdivide the datasets. The 'Status' column therefore contains all the classes of the machine we want to predict. In particular, the classes will be: 'Normal' if the conditions under which the machine was operating were normal, 'Warning' to represent the pre-fault situation, 'Critic' when the fault occurs and 'Not Classifiable' because for a limited period of time the sensor had disconnected from the surface of the motor and was therefore no longer able to collect the vibrations generated by the motor accurately.

The machine state column was used in the training set to train the algorithm. In the test set, however, the content of the column was hidden and populated by the predictions of the algorithm. After that, the predictions were compared with the true labels to evaluate the performance of the algorithm.

3.6.2.5 Fault Recognition

In this phase, the architecture of the ML algorithms is defined in order to predict failure.

In particular, it will be a classification problem. The algorithm, based on the dataset and vibrational characteristics, will have to be able to recognize the state of the machine.

The training of the ML algorithms is central to the whole project. The datasets must be divided into a training and a test set. The test set is constructed so that it has a size of 20-25 per cent of the starting dataset.

There were two splits of the datasets into training and test set in order to mitigate the problem of overfitting.

Overfitting, in the context of Machine Learning (ML), occurs when a trained model has learnt to perform very well on training data, but performs poorly on test data or new data not previously seen.

For this purpose, therefore, the datasets were first carefully divided on the basis of prior knowledge of the machine state. Care was then taken to place all classes in both the training and test set without regard to temporal sorting. After that, these training sets were used to train the algorithms and were then evaluated.

After that, another subdivision was tried. That is, the datasets were subdivided respecting the temporal flow. Thus, the training set was composed of the portion of the starting dataset from 01/01/2022 to 31/01/2023. While the remaining dates were considered to populate the test set.

For this classification problem, based on the evidence from the literature, the following ML algorithms were used: Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF) and k-Nearest Neighbors (k-NN). All these algorithms and their functioning were discussed in Chapter 2.

These algorithms were trained with each dataset and the parameters of each of them were set to define their characteristics and structure. The implementation of each algorithm will be explained in the next Chapter.

3.6.2.6 Performance Evaluation

Finally, the performance of each dataset-algorithm combination was evaluated.

The evaluation was carried out by considering several metrics.

Primarily, the accuracy of each algorithm application was taken into account. Accuracy intended as the ratio of correct predictions to the total number of inputs.

In addition, the Confusion Matrix for each algorithm application was also calculated as a supporting tool. This matrix is crucial for understanding where the algorithm makes misclassifications. In this way, the nature of the error can be understood, and the model can be corrected or optimized.

However, classical performance evaluation metrics were not the only indicator used. In fact, the training time of an algorithm was also considered. For codes that required high computing power, the Google Collaboratory platform was exploited.

CHAPTER 4 - Implementation and Python code

In the following section of this research, the synergies between vibration analysis and Machine Learning techniques applied to Predictive Maintenance are examined in the context of the investigation of concrete problems in a corporate environment. In the previous Chapter, the company's need to adopt a Predictive Maintenance approach was presented and the theoretical foundations on which the approach is based and the strategy to overcome failure prediction challenges were outlined.

The following Chapter constitutes a crucial phase of the project as it allows the theoretical concepts and approach proposed previously to be translated into practice. This section reveals the technical architecture behind the implementation of the Predictive Maintenance model, explaining the technical details and tools that support the functioning of the system. Through the presentation of the Python code associated with the implementation, as well as the design choices made, the implementation process that led to the success of the entire project will be illustrated.

The in-depth analysis of the technological infrastructure and the libraries used will be a fundamental aspect of this section, enabling a full understanding of the context in which the model operates. In addition, the analysis of architectural decisions and design strategies will provide a comprehensive overview of the choices made and their impact on the achievement of the set goals.

However, the Chapter is not limited to an illustration of the code. A clear understanding of the development processes will also be provided, highlighting how the results of vibration analysis are integrated within the Machine Learning system and how the algorithms are structured and set up to achieve machine state recognition. This practical implementation represents the core of the contribution, as it transforms the information extracted from the data into suggestions to facilitate maintenance-related decision-making processes.

In particular, the Chapter will be subdivided on the basis of the structure of the approach to the problem adopted meanwhile highlighting the implementation of the code and the rationale behind it. The first part of the Chapter will present the libraries used during the implementation, after which it will follow the below logical flow by presenting the code:

1) Raw data extraction and cleaning

2) Signal Processing
3) Datasets Creation
4) Training of Machine Learning algorithms
5) Performance Evaluation

In conclusion, this Chapter represents the point of convergence between theory and practical application, demonstrating the effectiveness of the multidisciplinary approach based on vibration analysis and Artificial Intelligence in the field of Predictive Maintenance. The detailed analysis of the Python implementation and code will highlight the applicability of the methodologies developed for the specific business needs that are under study.

4.1 Python Libraries Used

During the course of this section, the libraries used in the project's Python code will be exposed and analyzed in detail. Each library will be presented, emphasizing its specific functionality and importance in the context of the implementation. In addition, the rationale behind the choice of each library will be provided, explaining why they were integrated into the development environment and where in the code they found application.

The aim of this section is to provide a comprehensive overview of the software resources used in the implementation of Predictive Maintenance, ensuring a deep comprehension of the technological infrastructure at the basis of the model. Each library, carefully selected to meet specific needs, contributes significantly to the overall system performance, enabling vibration data analysis and fault detection in an accurate and efficient manner.

4.1.1 NumPy

NumPy is an open-source project that enables numerical calculation with Python. It was created in 2005 based on the initial work of the Numeric and Numarray libraries. It is a fundamental Python library for manipulating and analyzing numerical data. Its architecture provides support for the efficient handling of multidimensional arrays and matrices, making it a first choice for developers and scientists working with complex data and functions or requiring high numerical computation.

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more. [30]

This library is therefore based on a key concept: NumPy arrays. These arrays are similar to Python lists, but they offer significant advantages in terms of computational speed and functionality. NumPy arrays enable the simultaneous processing of large-scale data. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists [31].

Therefore, the main benefit of NumPy resides in its ability to perform mathematical and statistical operations on arrays quickly and efficiently. This is particularly relevant in the area of vibration signal analysis, where operations on large datasets can require considerable computing power. NumPy offers integrated functions for statistical calculation, data analysis and signal processing, thereby significantly simplifying the process of analyzing vibrational data.

This library contributed significantly to the definition of this model. NumPy was used in much of the Signal Processing phase. In particular, it was used during the application of:

- Techniques in the Time domain techniques: to calculate the RMS and Crest Factor indicators of the signal.
- Techniques in the Frequency domain: to calculate the Fourier Transform of the signal.

In addition, it was used in the initial cleaning and sorting of the raw data.

As explained before, the decision to use the library derives from its high computational speed, which made it possible to simultaneously process large-scale data in a short time.

4.1.2 Pandas

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. is a Python package providing fast, flexible, and expressive data structures designed to make working

with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis/manipulation tool available in any language. It is well suited for many different kinds of data like tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet, ordered and unordered (not necessarily fixed-frequency) time series data, arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels and many other data structures. [32]

Pandas is a widely used library for the manipulation and analysis of tabular data. Its versatility and advanced functionality make it an indispensable tool for anyone involved in data analysis, including data scientists and engineers involved in Predictive Maintenance.

At the heart of Pandas are two key data structures: the DataFrame and the Series. The DataFrame is a two-dimensional table-like object in which data can be organized in columns and rows. The Series, on the other hand, is a one-dimensional structure that can be viewed as a column or row in the DataFrame. These data structures are ideal for representing and analyzing complex data, such as signals collected by vibration analysis sensors. In this case study in particular, the DataFrame was a crucial and widely used concept for representing data and creating Datasets.

In addition, pandas offers a wide range of functions for data cleaning, selection of relevant information, aggregation and data analysis. This is particularly useful in the context of Predictive Maintenance, where data quality and preparation play a crucial role. Using the library, one can easily load data from various sources, including CSV, Excel or database files, and then one can always use Pandas to explore, process and prepare them for further analysis.

One of the main advantages of Pandas over other Python libraries or data structures is its user friendliness and ability to handle heterogeneous data. Pandas allows handling of missing data in a robust manner, performing data merge and aggregation operations without problems, and conducting detailed analyses through its extensive set of statistical functions. In addition, Pandas integrates perfectly with other data analysis libraries, such as NumPy and Matplotlib, enabling the creation of comprehensive and customized analysis workflows. This approach has also been implemented in the code of the project in question.

Thanks to its ability to manipulate data and thus exploiting the characteristics of DataFrames, this library can be encountered in almost every phase of Python code. In particular, its application can be found in the following areas:

- Reading raw data: The raw data extracted from the OPTIME system are in CSV format. Thanks to the pandas' functions, it was possible to load the documents into the working environment.
- Raw Data Cleaning: In this step, pandas was used for a variety of applications such as concatenating data, converting data into numeric formats, DataFrame merging and many other applications.
- Signal Processing: Using the properties of DataFrames to manage and organize data which are subject to the various signal processing techniques and methods.
- Creating Datasets: taking advantage of the functionality that the library offers.
- Training of Machine Learning Algorithms: pandas was used to load and read the various training and test sets required for training the algorithms.

Pandas is a very useful library and the methods and attributes of DataFrames are crucial for data management. Examples of the DataFrames attributes most used in this case study are:

- '.columns': attribute that returns the list of column labels in the DataFrame. This is essential because it allows specific columns to be accessed in a clear and readable manner.
- '.shape': returns the number of columns and rows in a DataFrame. Important for understanding the size of data structures.
- head()': displays the first n rows of the DataFrame. Very useful for visualising and monitoring the evolution of data cleaning.

There are many other attributes that are functional to a specific application.

Some of the DataFrames methods most often implemented to manage and manipulate and data are:

- '.loc[]': this method allows the selection of content based on the indicated row and column labels.
- '.iloc[]': allows selections based on integer positions as indices instead of labels.

- '.groupby()': this method allows you to group data based on the values of one or more columns and then apply aggregation functions. The groupby() method has been widely used to apply Signal Processing techniques in the time domain.
- '.merge()': allows two DataFrames to be joined based on one or more common columns. It is similar to the 'join' operation in Java. This method has been used extensively in the creation of Datasets.
- '.drop()': allows specific rows or columns to be removed from the DataFrame. The drop() method was mainly used in the initial part of the code where it was necessary to read and sort the CSV file containing the extraction of the raw data from the OPTIME system.

Of course, any DataFrame object allows a multitude of very useful operations for data manipulation. Combined with the computational speed of the library, it offers a perfect solution for handling large amounts of data such as data analysis during vibration analysis.

In conclusion, Pandas is an inestimable resource in the field of Predictive Maintenance, providing the necessary means for managing and analyzing complex vibrational data. Its flexibility, usability and advanced functionality make it an ideal choice for those seeking to explore, prepare and analyze data effectively and efficiently in the area of fault prediction.

4.1.3 CSV

The handling of CSV (Comma-Separated Values) files is a fundamental aspect of data analysis and Predictive Maintenance. Python offers a built-in library for manipulating CSV files, which makes it easy to read, write and manipulate structured data in table format.

This library was clearly used in the initial phase of the code. In particular, it was used when it was necessary to load the raw data massive extraction file.

4.1.4 Regular Expression (re)

A regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression (or if a given regular expression matches a particular string, which comes down to the same thing) [33].

A regular expression is a sequence of characters that defines a searching pattern. These patterns can be used to search, extract or replace text within a text string. For example, a regular expression can be created to find all occurrences of a certain text pattern, such as a simple character, punctuation mark, telephone number or e-mail address, within a document. This library allows users to perform very flexible search and manipulation operations.

The use of regular expressions is widespread in data analysis and text manipulation, including data used in Predictive Maintenance. Some of the advantages of implementing this package are the precise searching and filtering of regular expressions, the replacement of expressions in a text, and the processing and extraction of data from a text.

In the application in consideration, regex was very useful for extracting dates from the massive extraction of raw data.

By doing a massive signal extraction, the returned csv contained the sampled values for each day from 01/01/2020 until June 2023. Specifically, the dates were stored in a text in the first lines of the csv file. The text, however, contained not only the dates, but also other details such as the machine, the reference sensor and other information. The intention was to sort the vibrational signals collected on the basis of the matching date, as this would help to define the classification of the machine state on the basis of dates. However, the fact that the dates were immersed in a text represented a problem.

Regular Expression (re) has proved to be a reliable and effective solution. In fact, using the re.findall() function, it was possible to find all dates within the text and extract them into a Python list. 're.findall()' returns all non-overlapping matches of pattern in string, as a list of strings or tuples [33]. The regular expression used to identify dates in the text was 'dd/dd/dddd' or preferably 'dd/mm/yyyy'. Since all dates immersed in the text showed the format 'dd/mm/yyyy', the re.findall() function was used to locate all dates and save them in a list.

This library was therefore very useful in the initial Data Cleaning phase of the code to prepare the data for the Signal Processing phase.

4.1.5 SciPy

SciPy is an open-source library, and it is a collection of mathematical algorithms and convenience functions built on NumPy. It adds significant power to Python by providing the user with high-level commands and classes for manipulating and visualizing data [34].

SciPy offers a wide range of specialized modules and sub-packages, each of which addresses specific mathematical and scientific problems. The following are some of the sub-packages and their functionalities in SciPy:

- clusters: offers the possibility of applying clustering algorithms.
- interpolate: allows the application of functions to interpolate data.
- linalg: this SciPy package offers the opportunity to apply linear algebra operations.
- stats: allows the application of statistical functions and the search for unknown distributions of data.
- signal: this sub-package of SciPy is very important in a Predictive Maintenance application. In fact, it contains numerous useful functions in the field of signal analysis. Some of the functions of this sub-package are signal convolution, filtering applications, applications of methods in the time, frequency and time & frequency domains, windows functions and functions for applying spectral analysis.

The use of SciPy has several significant advantages in the field of Predictive Maintenance and data analysis:

- High computational speed: It makes it possible to perform complex calculations and operations even on large datasets.
- Large Community and Documentation: The library has a large user community and extensive documentation, which facilitates learning and problem solving.
- Versatility: SciPy is able to tackle a wide range of mathematical problems thanks to its sub-packages.
- Compatibility with NumPy and Pandas: Besides being able to apply signal analysis techniques quickly, this is certainly one of SciPy's main advantages. Its compatibility with NumPy and Pandas enables the creation of complete and efficient data analysis workflows without work-arounds. This is undoubtedly a feature that makes the three libraries present in most Predictive Maintenance applications.

In the case study presented in this research, the library was adopted mainly in the Signal Processing phase. In particular, for calculating the Fast Fourier Transform, but especially for applying the Short Time Fourier Transform (STFT) technique in the time & frequency domain.

scipy.signal.stft()' allows the calculation of the STFT of a signal. This function of the 'signal' module of SciPy requires certain parameters as input, which in the case under consideration were:

- x: an array in the form of a time series of the signal. The signal for each date sorted and cleaned after the data cleaning phase was given as input.
- fs: representing the sampling frequency. In our case this depends on whether the extraction made belongs to the OPTIME 3 or 5 sensor as they have different bandwidths. By default this parameter is set to 1.
- window: it should be specified which window to apply to analyze the signal. It depends on the application case.
- nperseg: length of analysis window. Indicates how many signal samples are included in each window. By default it is set to 256.
- noverlap: indicates the number of observations to be considered in the overlap between windows of analysis. It must always be less than nperseg by definition.
- nfft: indicates the length of the FFT used to calculate signal frequencies. It must always be greater than nperseg.

The last three parameters are very important for defining the desired trade-off between time resolution and frequency resolution. In the case under consideration, based on multiple tested combinations, the three parameters were set as follows: nperseg = 1000, noverlap = 300 and nfft = 2000.

In conclusion, SciPy is a fundamental library in the field of scientific and data analysis, providing a wide arsenal of tools and algorithms for the advanced analysis of vibrational data and the design of effective Predictive Maintenance models. Its versatility, efficiency and compatibility with other libraries make it a valuable choice for data analysts and engineers involved in similar projects.

4.1.6 PyWavelets (pywt)

PyWavelets is open-source wavelet transform software for Python [35].

PyWavelets is a library mainly used to apply Wavelet Transforms. The Wavelet Transform is a Signal Processing technique in the time & frequency domain that has been discussed in previous Chapters.

By making the wavelet transform easily implementable, it is a library that finds its application primarily in image processing and analysis. However, it has also been applied in this practical case to try to analyze the signal by finding sudden peaks.

In addition, the pywt library is a powerful and flexible tool for denoising signals, including vibrational signals used in the context of Predictive Maintenance. pywt offers a wide range of functionality for applying wavelet transforms to signals in order to eliminate noise.

In fact, pywt played a crucial role in this practice, particularly in the creation of the Denoising Dataset with which the Machine Learning algorithms were then trained.

The function pywt.threshold() was used to reduce the noise in a signal. It was used because it is suitable for the application of a dynamic filter. Initially, fixed denoising was implemented, setting a standard thresholding for any signal. However, it was realized that this was unsuitable because it greatly diminished the characteristics of the signals when the machine was operating under normal conditions, however it only slightly diminished the noise when the machine was operating under critical conditions (pre-fault or fault) resulting in an almost useless application.

This dynamic denoising technique, on the other hand, overcomes this problem because the threshold varies from day to day based on the signal thanks to a 'for loop' that operates for each date. The amount of threshold in this way is equal to a fixed amount multiplied by the maximum signal of the day being analyzed.

pywt.threshold() requires parameters to be set as input. In particular:

- data: a numeric array, in this case the signal.
- value: a scalar value representing the threshold value. Which in our case we have made dynamic.
- mode: it requires the type of thresholding to be applied to the data. There are two types of modes: hard and soft.

In hard mode (or hard threshold), all signal coefficients below the threshold are set to zero. Coefficients above the threshold are retained without modification. In the soft mode (or soft threshold), the signal coefficients below the threshold are set to zero, while the coefficients above the threshold are reduced in proportion to the difference between the coefficient and the threshold. In the research case under discussion, the 'soft' mode was chosen because this mode preserves the significant coefficients and attenuates the noise coefficients, allowing greater preservation of details in the signal.

In conclusion, the pywt library represents a valuable tool in the field of Predictive Maintenance, allowing the quality of vibrational data to be improved and anomalous patterns to be identified more effectively, and enabling the application of dynamic denoising techniques. The multi-scale approach of wavelet transforms offers a unique perspective for advanced signal analysis, and pywt's flexibility makes it an ideal choice to address specific challenges in the field of Predictive Maintenance.

4.1.7 Scikit-learn (sklearn)

Scikit-learn is a Python module for machine learning built on top of SciPy [36].

Scikit-learn, commonly abbreviated as sklearn, is an open-source library in Python that is widely recognized and used in the field of machine learning. Scikit-learn offers a wide range of machine learning algorithms, data pre-processing tools and performance evaluation functions, making it an essential resource for anyone wishing to develop effective machine learning models.

Some of the main features of this library are:

The ability to easily implement Machine Learning algorithms. It offers classification algorithms for recognizing and predicting classes, regression algorithms for predicting values, clustering algorithms for finding patterns in the data, dimension reduction algorithms such as PCA and many other algorithms.

It is easy and intuitive to implement, even if one does not have a great experience with ML algorithms.

The library offers tools for data pre-processing, including standardization, normalization and handling of missing data. This step is critical to ensure that the data are ready for model training.

Sklearn also allows the division of data into training and test sets and the evaluation of model performance using the metrics we have discussed in previous Chapters, such as accuracy, precision and F1 score.

The library has a high computational efficiency, allowing models to be trained on large datasets efficiently.

In addition, it has high compatibility with other Python libraries. Scikit-learn in fact integrates easily with other Python libraries widely used in this model, such as NumPy, SciPy and pandas.

Scikit-learn has applications in a wide range of fields, including: classification of product reviews, classification of medical images or object detection, machine translation or chatbots, for the processing and preparation of complex datasets to enable algorithms to be learnt.

In conclusion, scikit-learn is a versatile machine learning library with applications in a wide range of fields. But above all, the library plays a key role in the field of Predictive Maintenance because it enables raw data to be transformed into crucial information for failure prevention, thus helping to keep industrial equipment running efficiently and cost-effectively.

In the RDM project, sklearn found numerous applications and proved to be fundamental, especially in the final phase where the various algorithms for the Fault Recognition phase were trained and evaluated.

In the following, some of the modules of this library that have been implemented are presented, quickly describing them and explaining why they have been used.

- Sklearn.neighbors()
    - sklearn.neighbors is a module within the scikit-learn library that contains classes and functions for the application of neighbor-based algorithms in machine learning. Specifically in the code, which will be presented in the second part of the Chapter, the function KNeighborsClassifier() is called. This class enables the implementation of the k-Nearest Neighbors (k-NN) algorithm for classification.

The module was used to implement the k-NN algorithm, which was then trained from the different datasets and evaluated according to its use with each dataset.

- Sklearn.tree()
  - sklearn.tree is a module of the scikit-learn library that contains classes and functions for the creation, training and use of decision trees in machine learning. In particular, the DecisionTreeClassifier() function was used. This class implements the Decision Tree (DT) algorithm for classification problems. This function allows decision trees to be created and trained to classify data into distinct classes. It was used to train the Decision Tree algorithm.
- Sklearn.ensemble()
  - sklearn.ensemble is a module that offers a wide range of ensemble methods for machine learning. Ensemble methods are techniques that combine the predictions of several machine learning models to improve the overall performance of the model. The module was used to implement the RandomForestClassifier() function. This class was implemented in order to use the Random Forest algorithm. RF is an ensemble algorithm because, as explained in the ML Chapter, it creates an ensemble of decision trees.
- Sklearn.svm()
  - This module was used because it allows the Support Vector Machine (SVM) algorithm to be implemented. The class of this module that was called is the SVC (Support Vector Classification): this class implements the SVM algorithm for classification problems. The SVC tries to find a hyperplane that separates the different classes in the training data as optimally as possible.
- Sklearn.metrics()
  - sklearn.metrics is a module that offers a number of functions and classes to calculate and evaluate various performance metrics of machine learning models. In this case study, it was a widely used module because it was implemented in all the ML algorithms that were trained. In fact, it allowed the functions accuracy_score(), classification_report() and confusion_matrix() to be called, where the former was used to calculate the accuracy of the model, the latter metric to plot the confusion matrix, and the classification report on the other hand represents a complete overview of the classification metrics.
- Sklearn.impute()

- sklearn.impute is a module within the scikit-learn library that contains classes and functions to deal with missing data. In fact, ML algorithms cannot be trained using datasets or DataFrames with missing values (NaN). To handle these exceptions in the model, the SimpleImputer() function was called. This class allowed missing values to be handled by replacing them with average values.

4.1.8 Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python [37].

Matplotlib is a data visualization library in Python that is widely used in data analysis and machine learning. Its versatility and wide range of customization options make it an essential tool for the graphical representation of data.

Matplotlib offers a wide variety of graphs, including histograms, scatter plots, line plots, bar plots, pie charts and many others. This flexibility allows users to create effective visual representations for any type of data and to adapt the graphs to their specific needs.

The operation of Matplotlib is object-based. Users create a 'figure' object that represents the main graph display, within which one or more 'subplots', which are the axes of the actual graphs, can be created. The axes can be customized in terms of scale, labels, colors and more.

It was mainly used to visualize the output of evaluation metrics of Machine Learning algorithms, such as the Confusion Matrix.

In particular, a module of this library was used, namely matplotlib.pyplot(). More specifically, pyplot provides a high-level interface for creating and customizing graphs in a simple and effective manner. This module was chosen because it provides a high-level interface for creating and customizing graphs in a simple and effective way. This module greatly simplifies the process of creating charts without having to deal with complicated details. In addition, it supports a wide range of chart types and it is easy to export and save the various chart types to the local computer.

In conclusion, matplotlib.pyplot is used for its high-level interface and flexibility make it a valuable tool for exploring, communicating and sharing visual results from data analysis. In this specific case, it was used to understand the results obtained from the application of ML

algorithms on the test set. Its potential and functionality make this library and module an important resource for Predictive Maintenance projects.

4.1.9 Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics [38].

This library is designed to simplify the creation of attractive and informative statistical graphs, and is widely used in data analysis, data visualization and data exploration in environments such as data science, academic research and business analysis.

Being based on matplotlib, seaborn is well integrated with the popular graphing and visualization library. It is also well integrated with Pandas and its DataFrames, which also made it usable in the project that is the topic of this research. Furthermore, Seaborn has a simple syntax and automatically handles missing data in graphs.

In the project under consideration, the library was used to visualize the results of the confusion matrices of the outputs of the Machine Learning algorithms in the form of heatmaps.

In the context of developing a Predictive Maintenance model, the use of Python libraries is a crucial element for the success of the project. During the implementation of the code, the need emerged to exploit the different capabilities offered by the libraries discussed in the first part of this Chapter. Each library contributed in its own unique way to the realization of an effective and efficient system.

Among the libraries used, Pandas, NumPy, SciPy and scikit-learn, and Matplotlib emerged as fundamental pillars for data management, numerical processing, statistical analysis and the application of Machine Learning algorithms. Pandas provided a flexible and powerful environment for data manipulation, enabling efficient data preparation and cleaning. NumPy was essential for performing large-scale numerical operations and time domain techniques, facilitating the handling of complex vibrational data. SciPy provided tools for advanced vibrational signal analysis and extraction of critical information especially in the time & frequency domain. Scikit-learn enabled the implementation of fault recognition models based on Machine Learning. While matplotlib was crucial for visualizing and interpreting the results obtained from the ML algorithms.

This section has highlighted the crucial importance of these Python libraries in this Predictive Maintenance and Fault Recognition project. In the next part of this Chapter, the main sections of the implemented Python code will be presented, highlighting how these libraries have been integrated to create a complete Predictive Maintenance system.

4.2 Code Implementation

This section will present the significant sections to the realization of a system to predict the machine state of the Python code that has been implemented.

It will therefore be presented the solution adopted in a real case to overcome the challenges inherent in taking a Predictive Maintenance approach.

The logical flow that will follow this section will be the same of the approach to the RDM need presented in the previous Chapter related to the case study. Therefore, in sequence, the code will be analyzed in the following phases:

1) Data Cleaning
2) Signal Processing
3) Datasets Creation
4) Training of ML algorithms
5) Evaluation of the Dataset-Algorithm combination

4.2.1 Data Cleaning

The first phase of Data Cleaning aims to make the raw data extracted from the OPTIME system readable and ready to be processed by the various Signal Processing techniques.

This section illustrates the main parts of the code adopted to obtain a DataFrame in which it is possible to operate on the vibrational data for each date. In the figure below, an example of raw data extraction in CSV format is shown:

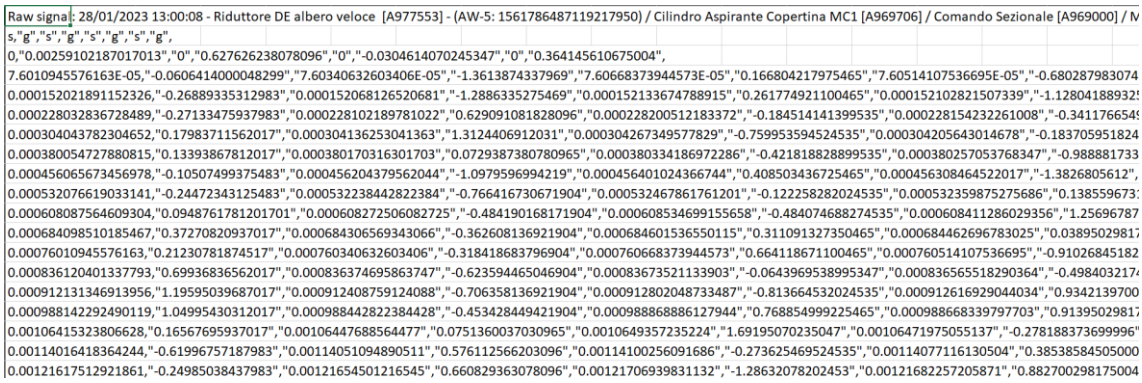


Figure 35: Example of a data extraction

The extraction shown is an example of a 4-day extraction in which the signal was sampled.

It can be observed that the first line is a text containing characteristics of the extraction. Among other information, this text contains the dates that must be extracted in order to be able to perform an efficient data cleaning.

The first part of the code is in fact focused on the extraction of these dates. First of all, the data extraction is opened using the 'pandas' library and some rows and columns are removed because they are insignificant to the purpose of the project. At this point, the resulting table is saved in a DataFrame 'df'.

After that, using the 'Regular Expression' library (re) and the findall() method, all dates within the text are searched for and appended to a 'dates' list. The library in these cases makes it possible to find a certain, repeated regular expression in a text. Using a for loop, the regular expression 'dd/mm/yyyy' is searched within the text.

At this point, one has all the dates of the extraction in a list. However, as can be seen from the extraction example, there are two measurements taken by the sensor for each date: 's' and 'g'. 's' refers to the time instant of the single measurement, 'g' refers to the acceleration measured by the accelerometer. Therefore, each date found with Regular Expression must be duplicated and then used as the first row of the DataFrame, in this way there will be two columns for each date with the corresponding values of the instants of time 's' and vibration 'g'.

```python
import numpy as numpy
import matplotlib.pyplot as matplot
import re
import csv
from scipy.signal import stft
import pandas as panda
import pywt

df = panda.read_csv("C:/Users/riccardo.toso/esempio_tesi.csv", header=None ,sep=',')#lettura estrazione
#separa ogni riga in base alla virgola e crea le colonne corrispondenti
df = panda.DataFrame([row.split(",") for row in df[0]])
df = df.drop(df.index[:5])
df = df.drop(df.columns[-1], axis=1)
#estrazione e duplicazione delle date dal testo
with open("C:/Users/riccardo.toso/esempio_tesi.csv", newline='') as csvfile:
    reader = csv.reader(csvfile)
    for row in reader: #voglio estrarre dalla prima riga le date dal testo della descrizione
        #Estrai il testo dalla prima cella
        text = row[0]#Estrai il testo dalla prima cella
        dates = re.findall(r'\d{2}/\d{2}/\d{4}', text) #Trova tutte le date nel testo con regex
        dates_list = []
        for date in dates: #ho una data per due misure ('s' e 'g')->duplico ciascuna data volta per volta.
            dates_list.extend([date, date])
        break
```

Figure 36: Extraction of dates

After the concatenation of the two DataFrames 'df' and the one containing the dates, a few steps were implemented to clean and order the DataFrame such as removing characters that impede the ability to apply Signal Processing techniques and the subsequent transformation from strings to float numeric values.

```python
#Per ciascunda data, le colonne pari si riferiscono al tempo, le dispari all'accelerazione captata dal sensore
new_names = {}
for i, col in enumerate(final_df.columns):
 if i % 2 == 0:
  new_names[col] = 's-' + final_df.iloc[0][col]
 elif i % 2 == 1:
  new_names[col] = 'g-' + final_df.iloc[0][col]
final_df.rename(columns=new_names, inplace=True)
#Ora cancello la riga 0 (che contiene solo le date) e cancello anche la riga 1 che contiene i vari 's' e 'g'
final_df = final_df.drop([0,1])
#Pulizia dati: tolgo le "" e converto in float
cols_accelerations = [c for c in final_df.columns if c.startswith('g-')]
for c in cols_accelerations:
 final_df[c] = final_df[c].str.strip('"')
for c in cols_accelerations:
    final_df[c] = panda.to_numeric(final_df[c], errors='coerce')

cols_time = [c for c in final_df.columns if c.startswith('s-')]
for c in cols_time:
 final_df[c] = final_df[c].str.strip('"')
for c in cols_time:
    final_df[c] = panda.to_numeric(final_df[c], errors='coerce')

```

Figure 37: DataFrame cleaning

By means of a simple print, it was realized that the various columns of accelerations and time instants had different dimensions and also contained null values NaN.

The current DataFrame, in fact, is in 'wide' format, which means that it is spread out in width: I have two columns for each date. This makes the DataFrame inefficient because a df by definition wants the columns with the same length, so it has automatically inserted null values to the columns to arrive at the maximum column length. The fact that a DataFrame wants equal column lengths means that null values cannot be removed.

To solve this problem, it was decided to change the format of the DataFrame from 'wide' to 'long' (thus developed in length). This means that the long_df will only have 3 columns: date, time and signal. Where the first column will contain all dates. In this way it will be possible to remove NaN values using the dropna() function.

```
#Ho un problema perchè ho valori NaN in ogni colonna e non posso eliminarli perchè il df vuole colonne lunghe uguali
#quindi modifico il mio dataframe da formato wide a formato long, in questo modo mantengo solo 3 colonne date, time e signal
time_df = panda.melt(final_df, value_vars=[col for col in final_df.columns if 's-' in col], var_name='date', value_name='time')
time_df['date'] = time_df['date'].str.replace('s-', '') #tengo solo la data
signal_df = panda.melt(final_df, value_vars=[col for col in final_df.columns if 'g-' in col], var_name='date', value_name='signal')
signal_df['date'] = signal_df['date'].str.replace('g-', '') #tengo solo la vibrazione

long_df = panda.concat([time_df, signal_df], axis=1)

long_df = long_df.loc[:, ~long_df.columns.duplicated()] #rimuovo duplicati

long_df.dropna(inplace=True) #Rimuovo i NaN
long_df['date'] = panda.to_datetime(long_df['date'], format='%d/%m/%Y')
```

Figure 38: DataFrame format transformation

4.2.2 Signal Processing

The Signal Processing phase is a fundamental step in developing a Predictive Maintenance model. Data processing in this case is aimed at extracting relevant information from the raw signal. To summarize, the Signal Processing phase in Predictive Maintenance is used to analyze the acquired signals to extract relevant information for fault recognition.

There are various Signal Processing techniques that can be grouped into three categories: time domain, frequency domain and time & frequency domain techniques. Their application and related advantages and disadvantages have been described in Chapter 1.

Firstly, five different indicators were applied in the time domain. Specifically, peak, peak-to-peak, RMS, Crest Factor and signal average were calculated.

Then, a dictionary was initialized in which the various aggregation functions were defined. In order to apply the functions for each date, the groupby() method was used, which works

like a for loop, grouping the DataFrame by date and thus applying the methods in the time domain to the corresponding signals.

```
#TIME DOMAIN: Statistical Measurements

#Prima di tutto creo un dizionario che definisce 5 diverse funzioni (di aggregazione) da applicare alla colonna signal del dataframe.
agg_func = {'signal': ['max', lambda x: x.max() - x.min(), 'mean',
                       lambda x: numpy.sqrt(numpy.mean(numpy.square(x))), lambda x: (x.max())/numpy.sqrt(numpy.mean(numpy.square(x)))]}

#questo dizionario con le 4 funzioni chiaramente lavora su una colonna, ma io sulla colonna signal ho tante date quindi:
agg_df = long_df.groupby('date').agg(agg_func) #Per ogni data applico gli indicatori nel dominio del tempo

agg_df.columns = ['peak', 'peak-to-peak', 'mean', 'rms', 'Crest Factor'] #Rinomino
```

Figure 39: Time Domain methods application

After time domain techniques, the Fast Fourier Transform was applied as a frequency domain method.

Any signal, including vibrational signals, can be decomposed into its sinusoidal components. This application is fundamental for analyzing the frequency component of a signal. The Fourier Transform provides detailed information on the frequencies present in a signal, which is crucial for vibration analysis and fault detection in machines. In fact, bearings and gearboxes emit specific frequencies that depend on the rotation of the motor shaft. If there is a fault in a bearing, for example, its characteristic frequencies are amplified. The detection of a fault through frequency analysis is enabled by the Fourier Transform.

In order to be able to apply the Fourier transform, a function was defined that wants as input the time instants, the signal measured by the sensor and the sampling rate.

By grouping the DataFrame by date, the time and signal arrays are easily obtained. However, the question for the sampling rate becomes a little more complicated. The OPTIME sensor samples for one second. Therefore, its sampling rate will be equal to the size of the 'signal' column (i.e. the number of measurements it took in one second). The OPTIME 5 sensor, on the other hand, samples for three seconds, so its sampling rate will be equal to the number of observations collected divided by three. For this reason, a for loop was implemented to calculate the sampling rate.

The function defined to calculate the Fourier Transform returns as output the information required to obtain a frequency spectrum, i.e. the frequencies and corresponding amplitudes.

To apply the transform, a method from the NumPy library, 'numpy.fft.fftfreq()', was invoked, which is used to calculate the frequencies corresponding to the coefficients of the discrete Fourier Transform (DFT) of a signal. This method is part of the NumPy module dedicated

to Fourier transforms. This method allows a signal to be converted from the time domain to the frequency domain by decomposing it into a series of sinusoidal components at different frequencies.

```python
#FREQUENCY DOMAIN
#Definiamo una funzione per calcolare la FFT
def calc_fft(time, signal, sampling_rate):
    n_samples = len(time)
    #t_end = time.max()
    freq = numpy.fft.fftfreq(n_samples, d=1 / sampling_rate)
    #mask = freq >= 0
    fft_vals = numpy.fft.fft(signal) #sono i valori che andranno sull'asse della x (frequenze in Hz)
    fft_power = numpy.abs(fft_vals) #sono i valori che andranno sull'asse della y (ampiezza in frequenza)
    return freq, fft_power
    #return freq[mask], fft_power[mask] #se volessi tornare solo le freq positive


#ciclo for per data per calcolare l'FFT
for date, group in long_df.groupby('date'):
    time = group['time'].values
    signal = group['signal'].values
    #alcuni sensori che campionano per 1 s ed altri per 3s. Quindi per calcolare il sampling rate ho 2 modi:
    if (time > 2).any():
        sampling_rate = len(time) / 3 #per i riduttori
    else:
        sampling_rate = len(time) #per i motori
    freq, fft_power = calc_fft(time, signal, sampling_rate) #applico la funzione definita e mi da in output frequenze e moduli della fft
```

Figure 40: Fast Fourier Transform definition

After techniques in the time domain and those in the frequency domain, time and frequency domain methods were applied.

Signal processing techniques in the time domain are suitable for analyzing time events and signals that change rapidly over time. On the other hand, frequency domain techniques reveal the spectral composition of signals and are useful for identifying periodic components. However, many applications require a combined approach encompassing both domains, especially when dealing with complex or time-varying signals. Techniques in the time & frequency domain offer this opportunity by obtaining both temporal and frequency information of the signal simultaneously. However, these techniques require a trade-off between temporal and frequency resolution. If one is more interested in temporal information, one optimizes temporal resolution by foregoing frequency resolution, and vice versa. In this case, a standard approach was taken by giving the both resolutions more or less equal weight.

The first application was the Short Time Fourier Transform (STFT). STFT is a signal-processing technique that decomposes a signal into a series of discrete Fourier Transforms (DFTs) calculated over superimposed short time windows of the signal. This application allows both temporal and frequency information to be obtained by passing time windows over which Fourier transforms are applied.

A function ('stft') of the 'signal' module of the SciPy library was used to apply the STFT.

A function to calculate STFT was then defined. It wants as input:

- the signal (observations measured by the sensor), taken from the main DataFrame.
- sampling rate, which was handled as in the case of the FFT.
- the window, i.e. the type of window to run over the signal. In this case, various window types were tried out and their application evaluated. Finally, the Hanning window was chosen.
- the three characteristic values of the STFT, nperseg, noverlap and nfft. In this case, various types of combinations were tried out and evaluated. Finally, the combination nperseg=1000 - noverlap=300 - nfft=2000 was the one that guaranteed the best resolutions for the model under consideration.

The outputs of the defined function will be: time instants, which are plotted on the x-axis, captured frequencies, which are plotted on the y-axis, and 'Zxx', which represents the spectrogram of a signal. Each element Zxx[t, f] of the matrix indicates the amplitude (or power) of the spectral component at frequency f at time t. This value is plotted on the z-axis using a color scale.

```
#STFT
def calc_stft(time, signal, sampling_rate, window, nperseg, noverlap, nfft):
    f, t, Zxx = stft(signal, fs=sampling_rate, window=window, nperseg=nperseg, noverlap=noverlap, nfft=nfft)
    return f, t, Zxx

df_stft = panda.DataFrame(columns=['date', 'STFT'])
for date, group in long_df.groupby('date'):
    time = group['time'].values
    signal = group['signal'].values
    #print(group)
    #Qui ho un problema: ho alcuni sensori che campionano per 1 s ed altri per 3s. Quindi per calcolare il sampling rate dico: se nella c
    if (time > 2).any():
        sampling_rate = len(time) / 3 #per i riduttori
    else:
        sampling_rate = len(time) #per i motori
    window = 'hann'
    nperseg = 1000
    noverlap = 300
    nfft = 2000
```

Figure 41: Fast Fourier Transform definition

In addition to STFT, the Wavelet Transform was also applied. In particular, the Continuous Wavelet Transform (CWT) was applied. A wavelet is a mathematical function that is localized in time and frequencies. The CWT analyses the signal as it runs along the time axis while detecting the frequencies.

The CWT was applied by calling up the PyWaveletes library. Wavelets can have different shapes, e.g. symmetrical, asymmetrical or complex wavelets. The shape of the wavelet determines how it responds to variations in signals.

In this case, the 'morlet' wavelet was chosen. This is one of the most widely used wavelets in Predictive Maintenance models. It is a complex wavelet with a sinusoidal-like oscillating component and a symmetrical shape.

However, the application of this type of analysis has achieved poor success proving to be unsuitable for this case study. It may be that the signal is too 'clean' or too uniform in frequency to be analyzed effectively with CWT. Moreover, it requires a high computational effort.

```
#Estraggo dati da dataframe
time = new_Data_Frame['Time'].values # estrai la colonna del tempo dal DataFrame
Acceleration = new_Data_Frame['Acceleration'].values # estrai la colonna dell'accelerazione dal DataFrame

# Impostare i parametri della CWT
wavelet = 'morl'  #wavelet morlet
scales = numpy.arange(1, 128)

#Applicazione della CWT
cwtmatr, _ = pywt.cwt(Acceleration, scales, wavelet)

matplot.imshow(cwtmatr, extent=[0, len(Acceleration), scales[-1], scales[0]], cmap='PRGn', aspect='auto')
#matplot.imshow(cwtmatr, extent=[0, len(Acceleration)/2 , scales[-1], scales[0]], cmap='PRGn', aspect='aut
matplot.colorbar()
matplot.xlabel('Tempo (ms)')
matplot.ylabel('Scala')
matplot.show()
```

Figure 42: CWT application

Another type of approach that has been applied is the combination of Empirical Mode Decomposition (EMD) and the Hilbert Transform, this type of signal processing is also known as the Hilbert-Huang Transform (HHT). This approach is used in signal analysis to achieve a better understanding of spectral components and temporal characteristics.

EMD is a signal decomposition technique that is based on the decomposition of a complex signal into a set of empirical mode functions (EMDs). The Hilbert Transform, on the other hand, is a mathematical function to identify the phase and amplitude information of the components of a signal. The sum of all empirical functions returns the original signal, so the aim of this approach is to apply the HT to each empirical function to obtain a better understanding of the signal.

This approach has proven to be poorly suited to the RDM case study due to its high computational requirements. The problem is the overly large Dataset and the application of EMD alone takes hours to process all the data.

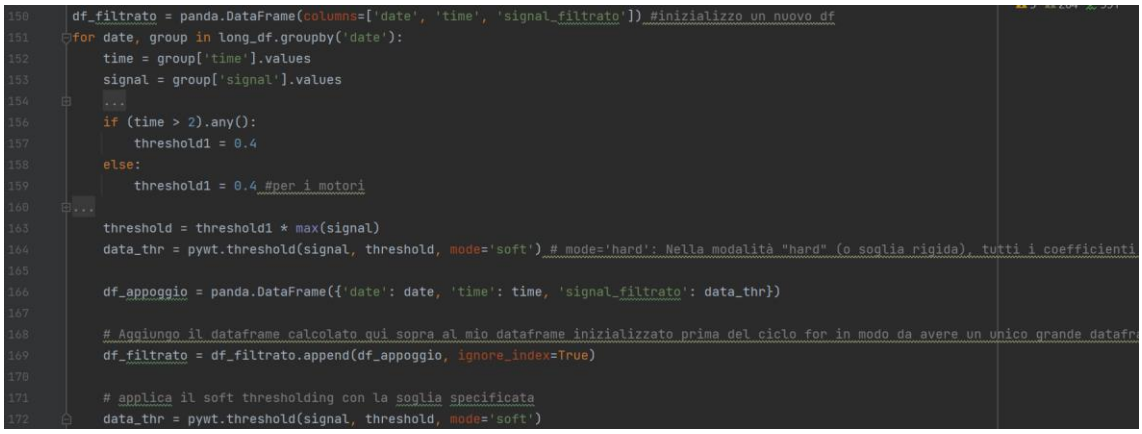
```
for date, group in grouped:                                                    ⚠ 6  ⚠ 112  ✓ 507
    time = group['time'].values
    signal = group['signal'].values

    #Applica l'EMD al segnale corrente
    emd = EMD()
    imfs = emd(signal, time)
    residual = emd.get_imfs_and_residue()[1]  #In posizione [0] mi ridà le IMFs, in posizione [1] mi rid

    #Plot delle IMFs e della funzione residua per la data corrente
    num_imfs = len(imfs)
    fig, axes = plt.subplots(num_imfs + 1, 1, figsize=(8, 2 * (num_imfs + 1)))
    fig.suptitle(f"IMFs for date: {date}", fontsize=12, fontweight='bold')

    axes[0].plot(time, signal, color='blue')
    axes[0].set_title('Segnale originale')

    for i, imf in enumerate(imfs):
        axes[i + 1].plot(time, imf, color='red')
        axes[i + 1].set_title(f'IMF {i + 1}')

    axes[-1].plot(time, residual, color='green')
    axes[-1].set_title('Funzione residua')
```

Figure 43: EMD analysis

Finally, a denoising technique was applied to reduce the influence of noise, thus facilitating the highlighting of the most significant signal information. In particular, a 'soft' thresholding technique was applied. This application implies a dynamic threshold, which varies as the signal sampled by the sensor varies. The threshold is in fact calculated by multiplying a fixed value for the maximum amplitude of the signal at each date.

This soft thresholding technique was implemented using the PyWavelets library.

Based on the coefficients reduced by the technique just presented, time domain techniques and the FFT will be applied to develop a Dataset different from the others. The latter will be one of the Datasets with which the Machine Learning algorithms will be trained.



```
df_filtrato = panda.DataFrame(columns=['date', 'time', 'signal_filtrato']) #inizializzo un nuovo df
for date, group in long_df.groupby('date'):
    time = group['time'].values
    signal = group['signal'].values
    ...
    if (time > 2).any():
        threshold1 = 0.4
    else:
        threshold1 = 0.4 #per i motori
...
    threshold = threshold1 * max(signal)
    data_thr = pywt.threshold(signal, threshold, mode='soft') # mode='hard': Nella modalità "hard" (o soglia rigida), tutti i coefficienti

    df_appoggio = panda.DataFrame({'date': date, 'time': time, 'signal_filtrato': data_thr})

    # Aggiungo il dataframe calcolato qui sopra al mio dataframe inizializzato prima del ciclo for in modo da avere un unico grande datafr
    df_filtrato = df_filtrato.append(df_appoggio, ignore_index=True)

    # applica il soft thresholding con la soglia specificata
    data_thr = pywt.threshold(signal, threshold, mode='soft')
```

Figure 44: Denoising application

4.2.3 Datasets Creation

In this section, the creation of the various Datasets will be illustrated. In this case study, since it is not clear which Signal Processing technique performs more efficiently, four types of Datasets were created:

- A dataset containing the results obtained only from the methods in the time domain.
- A dataset containing FFT results and indicators in the time domain.
- A dataset containing the outputs of the STFT and indicators in the time domain.
- A dataset containing the FFT outputs and indicators in the time domain once a denoising technique has been applied to reduce the effect of noise.

In addition to the above information, all datasets contain the additional information of temperature and drive shaft speed.

In addition to vibrations, the sensors also measure temperatures. Therefore, this data was also extracted from the OPTIME portal. Unlike the vibration measurement, temperatures are measured continuously. It was therefore necessary to filter out the temperature dataset by only keeping those from 14:00. Only those at two o'clock in the afternoon were kept because vibrations are also measured at that time.

The dates in this dataset are crucial to allow later merging with the other DataFrames containing the time domain, time & frequency and Denoising information.

Once a clean and ordered DataFrame of temperatures has been obtained, it will be saved on the PC so that it can be easily recalled when the datasets are created.

```
df['date'] = df['date'].astype(float)
df['date'] = df['date'].apply(lambda x: datetime(1899, 12, 30) + timedelta(days=int(x), milliseconds=int((x % 1) * 86400000)))

df['temperature °C'] = panda.to_numeric(df['temperature °C']) #conversione to float
#filtro il dataframe tenendomi solo le temperature riferite alle '12' perchè in realtà sono quelle che si riferiscono alle 14.
df_filtrato = df[df['date'].dt.hour == 12]
#rimuovo la parte relativa all'orario tenendo solamente la data intera come formato
df_filtrato['date'] = df_filtrato['date'].dt.date
#rimuovo la riga 2 e la riga 27 perchè sono dei duplicati
df_filtrato = df_filtrato.drop([2, 27])
```

Figure 45: Temperature DataFrame

The rotational speed of the motor shaft, on the other hand, is not measured directly by the OPTIME sensors. It is set directly by the operator during the plant set-up phase. However, the speed that is obtained is the running speed of the felts, given in meters per minute. The production speed varies according to the product mix and the grammage of the cartonboard to be obtained.

This information is considered crucial to the analysis because as the speed increases, the vibration level also increases. Therefore, it is not guaranteed that if the vibration level increases, there will be a failure. If this were the case, it would be sufficient to set a simple threshold above which a failure could be predicted. However, the situation is more complex.

The tangential speed data of the cylinder must be converted to obtain the rotational speed of the drive shaft. To do this, it is necessary to have the data for the reduction ratio and the cylinder radius.

Once the rotational velocities have been obtained, the DataFrame again needs to be filtered by keeping only the velocities relating to the 2pm. Dates are also indispensable to make merge with other DataFrames. Finally, the rotational velocities are saved on the PC in CSV format.

```
#definisco la funzione che mi calcola la velocità angolare
def calc_Hz_albero(vel_tang):
    riduzione = 8.33
    raggio_cil = 0.42 #0,42 metri perchè ha un diamentro di 840 mm
    #NB: divido per 2 pigreco per trovare i giri e non i radianti (RPM), dividendo anche per 60 (secondi) trovo la frequenza di rotazione d
    velocità_albero = ((vel_tang * riduzione) / raggio_cil) / ((2 * math.pi)*60) #divido per 2pigreco per trovare le rotazioni al minuto (R

    return velocità_albero

#creo un nuovo df che andrò a popolare alla fine del mio ciclo for con le date e le velocità angolari che calcolerò per ogni data applicand
df_veloc_albero = panda.DataFrame(columns=['date', 'Hz albero'])

for date, group in velocità.groupby('date'):
    vel_tang = group['velocità'].values #fisso la velocità tangenziale per la data

    velocità_albero = calc_Hz_albero(vel_tang) #applico la funzione appena definita calcolando la frequenza dell'albero motore

    #appendimi i valori di data e di velocità angolare che hai appena iterato nel dataframe inizializzato all'inizio del ciclo
    df_veloc_albero = df_veloc_albero.append({'date': date, 'Hz albero': velocità_albero}, ignore_index=True) #ignore index=true significa
```

Figure 46: Rotational velocities calculation

In order to create the datasets, it is necessary to take the DataFrames obtained from the Signal Processing techniques and merge them with the processed information of temperature and production speed.

In order to obtain a complete dataset with all the information, three different merges must be performed.

Firstly, as each dataset will contain the outputs of the techniques in the time domain, it will be necessary to merge them. After that, a second merge will be performed with the temperature information. Finally, a third merge will be done with the csv file for the rotational velocities of the motor shaft.

Each merge was done according to the date column, this column in fact functioning as a join key.

```
#MERGE DI TUTTE LE INFORMAZIONI IN UN UNICO DATASET

#Prima incollo sul df della FFT i valori del dominio del tempo:
primo_merge = FFT_DataFrame.merge(df_time, on='date') #la colonna 'date' funge da chiave di join, quindi per ogni riga del df dell'FFT og

#Ora prendo il file con le temperature e faccio il merge anche con quello
df_temp = panda.read_csv("C:/Users/riccardo.toso/temperature.csv", header=None, sep=',')
df_temp.columns = df_temp.iloc[0] #imposto la prima riga come intestazione di colonna
df_temp = df_temp[1:] #elimino la prima riga
df_temp['date'] = panda.to_datetime(df_temp['date'])
df_temp['temperature °C'] = df_temp['temperature °C'].astype(float)

secondo_merge = primo_merge.merge(df_temp, on='date', how='inner') #how = inner indica che verranno tenute solo le righe che hanno corris

#Ora faccio il merge col il file delle velocità
df_vel = panda.read_csv("C:/Users/riccardo.toso/veloc_rotazionali.csv", header=None, sep=',')
df_vel.columns = df_vel.iloc[0] #prima riga come intestazione colonna
df_vel = df_vel[1:] #elimino la prima riga 'date' e 'Hz albero'
df_vel['date'] = panda.to_datetime(df_vel['date'])
df_vel['Hz albero'] = df_vel['Hz albero'].astype(float)
df_vel['Hz albero'] = df_vel['Hz albero'].round(3)

fft_dataset = secondo_merge.merge(df_vel, on='date', how='left') #how = left mi dice invece laddove trovi corrispondenza sulla data incol
```

Figure 47: Merge of DataFrames

At this point, one has Datasets complete with all the elements that are known a priori or that have been obtained from Signal Processing techniques.

The missing thing from these Datasets is the variable to be predicted. The ultimate goal of this model is the early recognition of the machine status in order to prevent the occurrence of a fault in the future. For this reason, a 'Status' column has been initialized, which will then be populated with possible classes. The population of this column is based on the operators' knowledge of the machine's status.

Using the date column, it was possible to populate for each date the condition in which the machine was operating. Four classes were identified regarding the state of the motor:

Normal, Not Classifiable, Warning and Critical. Clearly, since these are historical data, the state of the machine is known a priori.

```
#Aggiungo le due colonne da predire: Stato Macchina
fft_dataset['Status'] = ''

#print(fft_dataset)

#Ora andiamo a popolare le colonne Status
fft_dataset.loc[(fft_dataset['date'] >= '2022-01-01') & (fft_dataset['date'] <= '2022-03-04'), 'Status'] = 'Normal'
fft_dataset.loc[(fft_dataset['date'] >= '2022-03-05') & (fft_dataset['date'] <= '2022-08-07'), 'Status'] = 'Warning'
fft_dataset.loc[(fft_dataset['date'] >= '2022-08-08') & (fft_dataset['date'] <= '2022-12-13'), 'Status'] = 'Not Classifiable'
fft_dataset.loc[(fft_dataset['date'] >= '2022-12-14') & (fft_dataset['date'] <= '2023-02-13'), 'Status'] = 'Critic'
fft_dataset.loc[(fft_dataset['date'] >= '2023-02-14') & (fft_dataset['date'] <= '2023-07-02'), 'Status'] = 'Normal'
```

Figure 48: Defining and populating the variable to be predicted

It is now necessary to split the dataset creating a training set and a test set.

This phase is crucial for implementing correct learning of Machine Learning algorithms. In particular, the training set is used to train the prediction algorithm. Based on the information in the training set, the algorithm learns to recognize the state of the machine by setting correlations between existing features. After that, the test set is used to evaluate the functioning of the algorithm.

The test set must always have a size of 20-30% of the original dataset. This weight was respected in the project in consideration.

In this phase, two different approaches were used to divide the original dataset into training and test sets.

The first, creates the training set and the test set on the basis of a decomposition made ad hoc so that all the classes to be predicted are included in the test set. The decomposition in this case therefore does not follow the time axis but is done in such a way that all information are represented, in a good proportion, in both sets. Furthermore, in this case, maintenance interventions were also taken into account. In this way, any post-maintenance effects are mitigated.

After that, some numerical conversions are necessary and finally both the training set and the test are saved in CSV format.

```
# Crea una lista contenente gli intervalli di date desiderati per il dataset di training
training_intervals = [('2022-01-01', '2022-01-20'), #Status Normal
                      ('2022-02-11', '2022-06-22'),
                      ('2022-06-28', '2022-07-10'),
                      ('2022-07-31', '2022-11-07'),
                      ('2022-11-20', '2023-01-04'),
                      ('2023-01-10', '2023-01-25'),
                      ('2023-01-31', '2023-02-04'),
                      ('2023-02-08', '2023-03-06'),
                      ('2023-03-14', '2023-04-18'),
                      ('2023-04-21', '2023-05-10'),
                      ('2023-06-12', '2023-07-02')]
# Crea una lista contenente gli intervalli di date desiderati per il dataset di test
test_intervals = [('2022-01-21', '2022-02-10'),#Status Normal - Lubrification No
                  ('2022-06-23', '2022-06-27'), #Status Warning - Lubrification Yes il 23/03
                  ('2022-07-11', '2022-07-30'), #Status Warning - Lubrification No
                  ('2022-11-08', '2022-11-19'), #Status Not Classifiable - Lubrification No
                  ('2023-01-05', '2023-01-09'), #Status Critic - Lubrification No
                  ('2023-01-26', '2023-01-30'), #Status  Critic - Lubrification No
                  ('2023-02-05', '2023-02-07'), #Status Critic - Lubrification No
                  ('2023-03-07', '2023-03-13'), #Status Normal - Lubrification yes
                  ('2023-04-19', '2023-04-20'), #Status Normal - Lubrification Yes
                  ('2023-05-11', '2023-06-11')] #Status Normal - Lubrificatio No
```

Figure 49: Subdivision of the Dataset

The second approach of splitting the original dataset instead aims to be consistent with the time axis.

Thus, the training set will contain information from the first historical date until the date for which the test set is 24% of the size of the original dataset. However, by proceeding in this way, the test set will not contain all the classes to be predicted.

```
# Crea una lista contenente gli intervalli di date desiderati per il dataset di training
training_intervals = [('2022-01-01', '2023-01-31')] #Primi 13 mesi nel training

# Crea una lista contenente gli intervalli di date desiderati per il dataset di test
test_intervals = [('2023-02-01', '2023-06-11')] #intervallo per il test
```

Figure 50: Second subdivision strategy

Both types of approaches will be evaluated later.

This process of defining the variable to be predicted and splitting the dataset into training and test sets will be repeated for each Dataset defined at the beginning of the section. After that, several Machine Learning algorithms will be trained and evaluated with each output of this phase.

4.2.4 Training of ML algorithms

In this phase, the Python code implemented to train the various Machine Learning algorithms for predicting the machine state will be presented.

In the context of Predictive Maintenance, the training of algorithms plays a crucial role in the entire model development process as it is at this stage that the model learns from the hidden relationships between features to predict and recognize a fault.

Every decision made at this stage, from algorithm selection to feature management, has a direct impact on the accuracy and reliability of the model, which is why careful attention to detail is crucial to the success of the entire Predictive Maintenance project.

In summary, four different datasets were created which differ from each other on the basis of the Signal Processing techniques applied.

In addition, two different approaches to splitting the single dataset were used to create the training and test set.

For each approach and for each dataset, four different classification algorithms were applied: the Decision Tree (DT), the Support Vector Machine (SVM), the k-Nearest Neighbors (K-NN) and the Random Forest (RF).

The applications of these algorithms differ from each other only in the moment of creation of the instance (initialization of the prediction algorithm) and in the configuration of its parameters.

All algorithms, in fact, follow a common implementation approach:

- The date and time information of both the training set and the test set must be removed at this stage. Otherwise, the algorithm, when tested, will associate the prediction of the machine state with the date and the performance would be, erroneously, very high. The algorithm must learn to correctly predict the class on the basis of vibrational and production features, not on the basis of the time axis.
- Independent features must be separated from dependent features in the two sets. In this case, the column relating to the status of the motor. In this way, the ML algorithm instance will be trained on the two DataFrames obtained from the training set. The first contains the training data, i.e. all the features (independent variables). The second will contain the class or labels to be predicted (dependent variables). The other two DataFrames obtained from the test set will instead be used to test the learning of the algorithm and to evaluate its performance.
- Data processing before the training phase. The presence of missing data causes problems when training the model as many machine learning algorithms require

complete data to make accurate predictions. In all applications, the 'SimpleImputer' module of the scikit-learn library was chosen. 'SimpleImputer' allows one to specify an imputation strategy to handle missing data. In particular, the strategy chosen in this project was to use averaging. Whenever a missing value, or NaN, is encountered in the sets of independent variables, or features, it will be replaced by an average column value.

- Once the algorithm instance has been created, it will be trained on the features and dependent variables of the training set.
- Finally, the execution of the models is implemented to make predictions about the test set to evaluate the models.

```
#elimino le prime due colonne 'date' e 'time' perchè magari l'algoritmo apprende sulla base di queste due colonne e sare
training_set = training_set.drop(['date', 'time'], axis=1) #axis=1 vuol dire che elimino le colonne e non le righe
test_set = test_set.drop(['date', 'time'], axis=1)

#Separo le variabili in variabili indipendenti (features) e variabili dipendenti (quelle che voglio predire) - step nece
X_variables = training_set.drop(['Status'], axis=1) #features
Y_Status = training_set['Status'] #prima variabile da predire

#Faccio la stessa cosa nel test set
X_test = test_set.drop(['Status'], axis=1) #features
Y_test_Status = test_set['Status'] #prima variabile da predire

#siccome gli algoritmi non funzionano se ho valori NaN utilizzo la tecnica di imputazione 'imputer con strategy=mean. Ov
imputer = SimpleImputer(strategy='mean')

X_variables_imputed = imputer.fit_transform(X_variables)
X_test_imputed = imputer.fit_transform(X_test)

dt_Status = DecisionTreeClassifier() #le istanze per il Decision Tree

dt_Status.fit(X_variables_imputed, Y_Status)#alleno i modelli usando il comando '.fit()' utilizzando i training set

Y_predizione_Status =dt_Status.predict(X_test_imputed)#Esecuzione dei due modelli per fare predizioni sul test set
```

Figure 51: Separation of variables, handling of missing data and training of the ML algorithm

The initialization of the various algorithms is shown in below. The parameters were set and configured after numerous training attempts.

The criterion by which the parameters were chosen is the achievement of good accuracy combined with a training time considered acceptable.

```
#Decision Tree
dt_Status = DecisionTreeClassifier() #le istanze per il Decision Tree
#K-NN
knn_status = KNeighborsClassifier(n_neighbors=8) #Imposto il numero di vicini da considerare (k) per 'Stato'
#Random Forest
rf_status = RandomForestClassifier(n_estimators=100, random_state=42)
#SVM
svm_status = SVC(kernel='linear', C=1.0, random_state=42) #Il regressore SVC si usa per problemi di classificazione multicla
```

Figure 52: Creation of ML algorithm instances

4.2.5 Evaluation of the Dataset-Algorithm combination

The accuracy, reliability and robustness of the algorithms are essential to ensure the success of Predictive Maintenance, as they determine the ability to anticipate failures and malfunctions in complex industrial plants. The performance evaluation phase is where one tests the effectiveness of the models and algorithms, comparing predictions with real data and measuring the accuracy of the results. Only through an in-depth evaluation is it possible to identify areas for improvement and optimize the performance of the algorithms to ensure efficient Predictive Maintenance.

In this section, the performance evaluation phase of the various Dataset-Algorithm combinations will be investigated.

For all applications under analysis, accuracy was calculated as the main performance evaluation metric. However, the training time of each application was also an indicator taken into account. The latter was tracked manually as the codes processed the outputs.

In addition to the two metrics just described, the Confusion Matrix and the importance of each feature in terms of its contribution to prediction were also calculated for each application.

The Confusion Matrix is very useful for understanding where and why the algorithm makes mistakes in prediction, specifically identifying the metrics of True Positive, False Positive, True Negative and False Negative. From the confusion matrix, accuracy can be derived.

Feature importance measures how much a particular feature contributes to the prediction of the model. The higher the importance, the more informative the feature is for the model. Feature importance is calculated based on how well the model uses each feature to make decisions during classification. In simple terms, if a feature is often used by decision nodes in the model and improves the purity of the nodes, then it will have a higher importance. Some models, e.g. Decision Tree and Random Forest directly present the method within their library.

Both the Confusion Matrix and the importance of each feature are plotted and visualized in graphical form to facilitate interpretation of the trained model. Visualization is made possible by exploiting the Matplotlib library.

```
dt_Status.fit(X_variables_imputed, Y_Status)#alleno i modelli usando il comando '.fit()' utilizzando i training set

Y_predizione_Status =dt_Status.predict(X_test_imputed)#Esecuzione dei due modelli per fare predizioni sul test set

#Valutazione delle prestazioni dei due modelli
accuracy_status = accuracy_score(Y_test_Status, Y_predizione_Status)

#Confusion Matrix
conf_matrix_Status = confusion_matrix(Y_test_Status, Y_predizione_Status)

# Calcola l'importanza delle feature dai modelli Decision Tree
feature_importance_status = dt_Status.feature_importances_

#Associa le feature all'importanza
feature_names = X_variables.columns  # Sostituisci con i nomi delle tue feature
```

Figure 53: Implementation of the main performance evaluation metrics

```
plt.figure(figsize=(10, 6))
plt.barh(range(len(feature_importance_status)), feature_importance_status, align='center')
plt.yticks(range(len(feature_names)), feature_names)
plt.xlabel('Importanza delle Feature')
plt.ylabel('Feature')
plt.title('Grafico delle Feature Importance per lo Status')
plt.show()
#printo le accuracy da 0 a 100%
print(f"Accuracy for 'Status': {accuracy_status:.2f}")
#printo il classification report che contiene delle metriche come precision, recall, f1-scorse, etc.
print("Classification Report for 'Status':")
print(classification_report(Y_predizione_Status, Y_predizione_Status))
print("Confusion Matrix for 'Status: ")
print(conf_matrix_Status)
# Visualizza la matrice di confusione per 'status' come heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_Status, annot=True, fmt="d", cmap="Blues",
            xticklabels=dt_Status.classes_, yticklabels=dt_Status.classes_)
plt.title("Matrice di Confusione - Status")
plt.xlabel('Etichetta Predetta')
plt.ylabel('Etichetta Vera')
plt.show()
```

Figure 54: Visualization of the main performance evaluation metrics

In this Chapter, it has been outlined the basic foundations on which the practical implementation of the Predictive Maintenance model based on vibrational analysis is constructed to meet the requirements of RDM.

Libraries such as NumPy, Pandas, SciPy, scikit-learn and Matplotlib were carefully integrated into the development process to take full advantage of their specialized capabilities. In fact, the five libraries represent the backbone of the model by appearing similarly in all applications of the datasets and ML algorithms. The synergic interaction of these libraries has allowed the creation of a resilient and practical system for Predictive Maintenance.

While the first part of the Chapter highlights the functionalities and implementations of the various libraries, the second part demonstrates in practice an effective approach through which the challenges of Predictive Maintenance can be overcome.

In the next Chapter, concrete results obtained from the application of these models will be explored, highlighting which Dataset-Algorithm combinations are most suitable for the company's case study. It will be shown how the integration of these libraries has led to practical solutions for fault prediction and the realization of an advanced predictive approach in industrial asset management.

CHAPTER 5 – Results and Discussion

This Chapter represents the heart of the entire research, as it focuses on the in-depth analysis of the results obtained through the application of the Predictive Maintenance model developed. The purpose of this Chapter is to present in detail the results of the practical implementation of the model in a real business context, such as RDM, and to conduct a critical discussion aimed at assessing the effectiveness and efficiency of the model itself and to understand the rationale behind these results.

The results presented in this Chapter are the outcome of the model designed, developed, and applied as described in the previous Chapters. The results are analyzed and compared with the initial research objectives, providing a comprehensive overview of the adequacy of the model in approaching the problem of Predictive Maintenance.

During this Chapter, some of the results obtained from the Signal Processing phase will be displayed. These were fundamental to understanding whether the techniques used had been implemented correctly. After that, the details of the models' performance in terms of accuracy in predicting faults and classifying machine states will be examined. The implemented models are presented and evaluated on the basis of different dataset partitioning strategies and on the basis of different training and test sets. Several attempts and analyses were explored in order to find the most efficient Dataset-Prediction algorithm combination.

In this section of the project, we follow the logic maintained and the reasoning and interpretations that were made during the development of the models.

In conclusion, this Chapter will allow to provide meaningful conclusions on the success and potential use of the proposed Predictive Maintenance approach, highlighting the practical implications for the company and its maintenance strategy.

5.1 Signal Processing Outputs

In this section, we will analyze the results obtained from the application of several techniques in the time, frequency and time & frequency domains. The visualization of the results was

crucial at this stage because it made it possible to optimize and better configure the parameters of the various Signal Processing techniques.

In general, for each application of a Signal Processing technique, the aim was to analyze how the output varies with the condition in which the machine was operating. Therefore, methods were applied to the raw signal of when the machine was operating under normal and fault conditions. Clearly, the purpose of this tests was to understand whether the implemented techniques behaved as expected and to understand whether they were able to "recognize" the state of the machine. The analysis was crucial to understand a priori which methods were appropriate for training the Machine Learning algorithms in the Fault Recognition phase. A simple increase in production speed leads to an increase in vibrational signal amplitudes, so the rotational speed of the drive shaft was also considered. Not only the production speed, but also the motor temperature was considered.

The first vibrational Signal Processing methods concerned the time domain. In the time domain, the amplitude of the signal is plotted against time. Monitoring the vibrational signal can be useful to understand the condition of the motor. However, due to noise, it is easy to misinterpret the signal and come to wrong conclusions. Therefore, time domain techniques intervene to facilitate fault prediction by applying comprehensive and appropriate statistical parameters to the raw data.

In the implementation of the Predictive Maintenance model presented in this research, five metrics were applied in the time domain, namely peak, peak-to-peak, average, RMS and Crest Factor. These metrics are presented and explored in Chapter 1.

In the following figure, the results of the application of these techniques will be presented for the cases in which the machine operated under Normal and Critical conditions. The data referring to June 2023 represent the statistical indicators in the time domain where the machine was operating under Normal conditions, while those of January refer to when the machine was operating under Critical conditions.

	Peak	Peak-to-peak	Mean	RMS	Crest factor
date					
2023-01-18	8.521	16.554	-0.000	2.660	3.204
2023-01-19	7.291	14.838	0.000	2.463	2.960
2023-06-07	0.736	1.414	-0.000	0.186	3.959
2023-06-08	0.670	1.241	-0.000	0.151	4.442

Figure 55: Time domain techniques outputs under fault and normal conditions

As can be seen, almost all of the metrics showed higher values where the machine operates under fault conditions. These techniques showed the ability to highlight the status of the plant and were consequently considered for the creation of the model.

Thanks to the NumPy library, the Fast Fourier Transform to signal (FFT) was implemented. This algorithm is used to analyze complex signals or data in the frequency domain. The FFT takes as input the time domain signal sampled by the OPTIME sensor and decomposes it into a combination of sinusoids of different frequencies. The result of the FFT is called the 'frequency spectrum' or 'power spectrum', which shows the amplitude of each frequency component in the original signal. In other words, it shows which frequencies are contained in the signal and how powerful they are. In the dataset constructed to train the ML algorithms, the FFT results are represented by the columns 'freq', containing the frequencies of the signal, and 'fft_power', which contains the amplitude for each frequency.

The figure below represents the application of the Fast Fourier Transform to the same two days sampled when the machine was operating under Normal and Critical conditions.

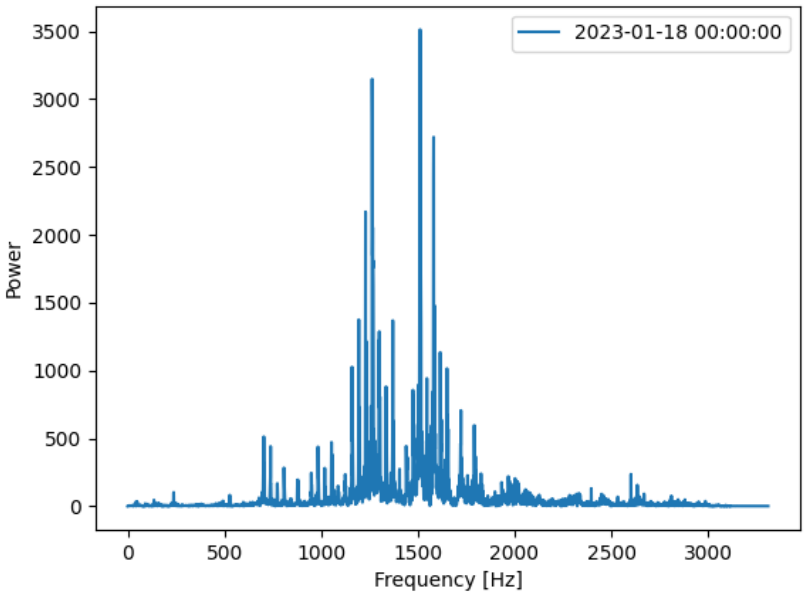


Figure 56: FFT spectrum under fault conditions

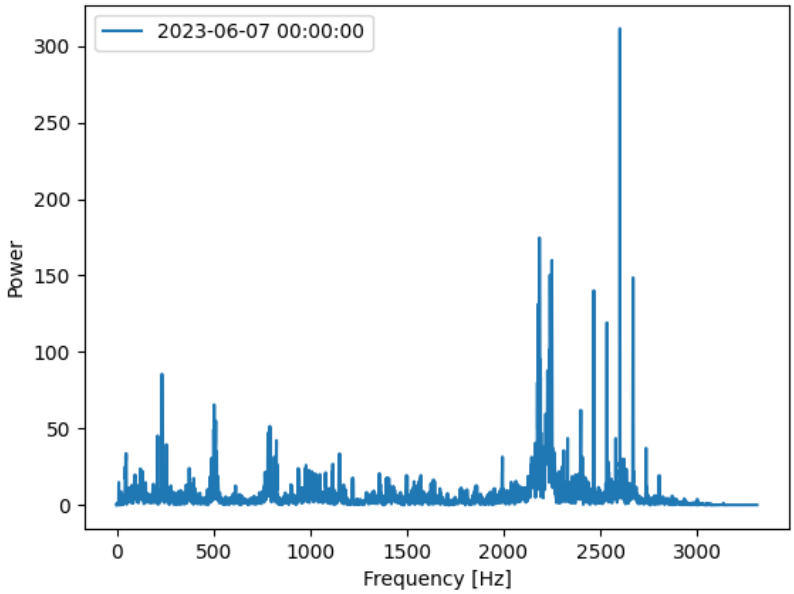


Figure 57: FFT spectrum under normal conditions

The frequency spectrum shows how the FFT is potentially able to recognize a fault. In fact, the frequencies that are stressed under the two conditions are different. In particular, the amplitudes, or more precisely the absolute values of the complex values obtained by the FFT, assume higher values when the machine operates under critical conditions. The spectrum of 7 June (normal conditions) shows how the powers reach values of 300, while the maximum power of 18 January (critical conditions) reaches 3500.

Thus, this analysis of the output powers can reveal whether the system is in a normal state or whether it is manifesting critical indications or anomalies. This concept is fundamental in the construction of the Predictive Maintenance model and, most importantly, is crucial for training Machine Learning algorithms. In conclusion, the Fast Fourier Transform was implemented in the model under investigation for its ability to reveal the state of the motor.

By implementing techniques in the time domain, the focus is on pure temporal behavior without having any information about the frequency composition of the signal. With the FFT, on the other hand, the frequency composition of the signal is examined without considering the signal's temporal behavior. For this reason, it was decided to implement certain techniques in the time-frequency domain. Through these techniques, it is possible to explore how frequencies vary over time.

The techniques that have been implemented are the Wavelet Transform, the Hilbert-Huang Transform (HHT) and the Short Time Fourier Transform (STFT). All these techniques have been analyzed and explored in Chapter 1.

The Continuous Wavelet Transform (CWT), is a signal analysis technique used to examine non-stationary signals, detect patterns or transient events, and study how frequencies vary over time. CWT is based on a family of mathematical functions called 'wavelets' that are used to analyze the signal at different frequency scales and at different positions in time. Its functioning is based on translating the wavelet onto the signal at different points in time and at different frequency scales while evaluating the similarity between the analyzed signal and the wavelet. The result of CWT is a continuous representation of the signal in time-scale planes.

The application of CWT, however, has achieved little success. The output of the wavelet transform, in fact, did not provide nothing of significance to the model. The reason is that the input signal to the algorithm is too 'clean' or too uniform in frequency to be analyzed effectively with the application of the wavelet transform.

The Hilbert-Huang Transform (HHT) is also a time-frequency domain technique used for analyzing complex signals that vary in time and frequency. The HHT is based on empirical mode decomposition (EMD), which is an iterative process for decomposing a signal into a series of 'intrinsic mode functions' (IMFs). Each IMF represents an oscillating component of the signal at a given frequency and amplitude. The Hilbert Transform is applied to each IMF

to obtain frequency component information and phase and amplitude information about the signal.

The application of the HHT has experienced limited success. The main obstacle encountered in the application of HHT was its high computational capacity requirement. The operations of empirical mode decomposition (EMD) and calculation of the Hilbert transform require a significant amount of computational resources, and this demand increases linearly with the size of the dataset, which in this case is large. Consequently, the implemented code required an extremely long execution time, and for this reason it was decided not to implement the technique in the constructed Predictive Maintenance model.

The following figure shows the application of EMD to a sampled signal. What can be seen in the visualization are the 'intrinsic mode functions' (IMFs) of the signal.

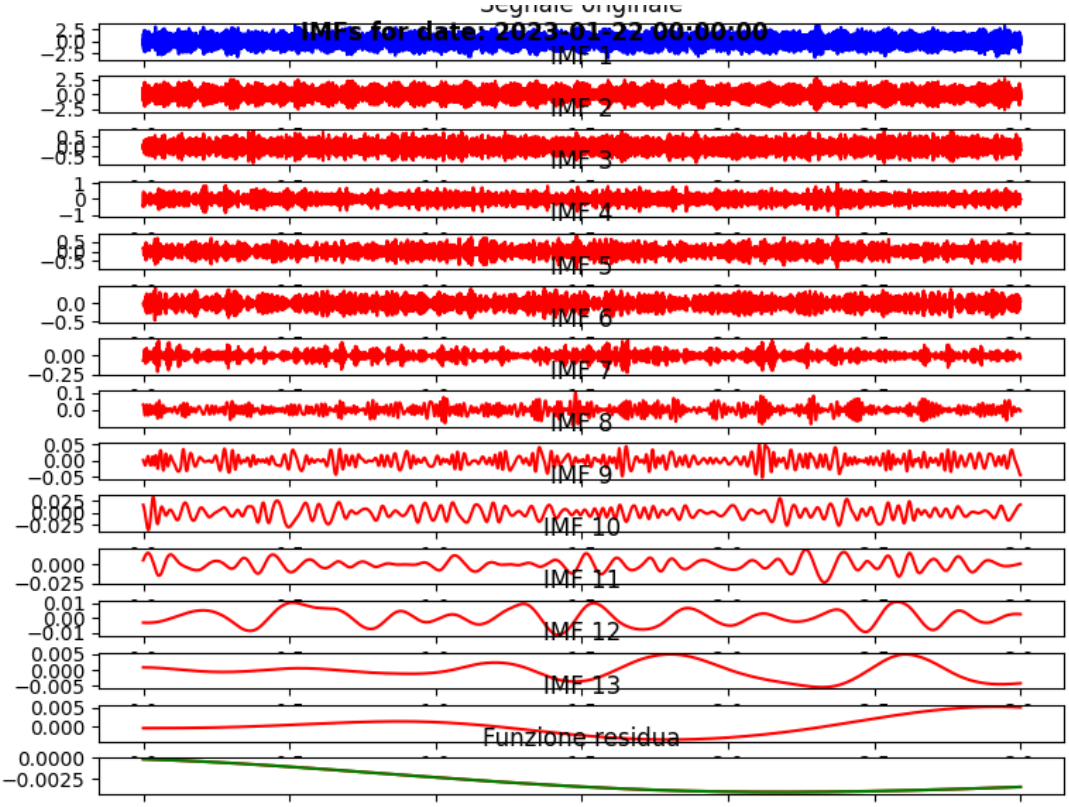


Figure 58: EMD application - Intrinsic Mode Functions (IMFs)

The Short Time Fourier Transform (STFT) is another time & frequency domain method capable of obtaining both frequency and time information of a non-stationary signal. Contrary to the Fourier Transform, instead of analyzing the entire signal in one step, the STFT divides the signal into smaller segments, known as "windows". For each window, the STFT calculates the FFT, thus obtaining a frequency spectrum for that specific window. This process is then repeated by moving the window along the signal over time. The end result is a representation of the frequencies over time, showing how the frequencies change as the signal evolves over time.

The two figures below show the output of the STFT. This technique makes it possible to analyze how frequencies vary with time. On the x-axis there are the sampling time instants and on the y-axis there are the frequencies. The color or intensity on the spectrogram indicates the power of the frequency components at a given time and frequency. As the following two figures show, the STFT can potentially recognize when the motor is operating under fault conditions. In fact, the scale of powers assumed by the various frequencies has very different values depending on the state of the motor. The frequency components in the case where the machine operates under Critical conditions assume values of 1, while the maximum power in the case where the machine operates under Normal conditions are 0.07.

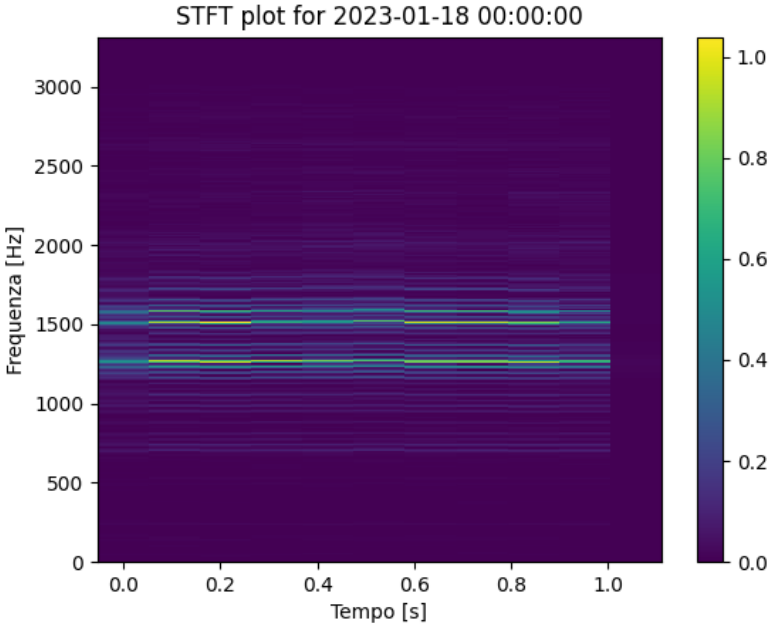


Figure 59: Short Time Fourier Transform application under fault condition

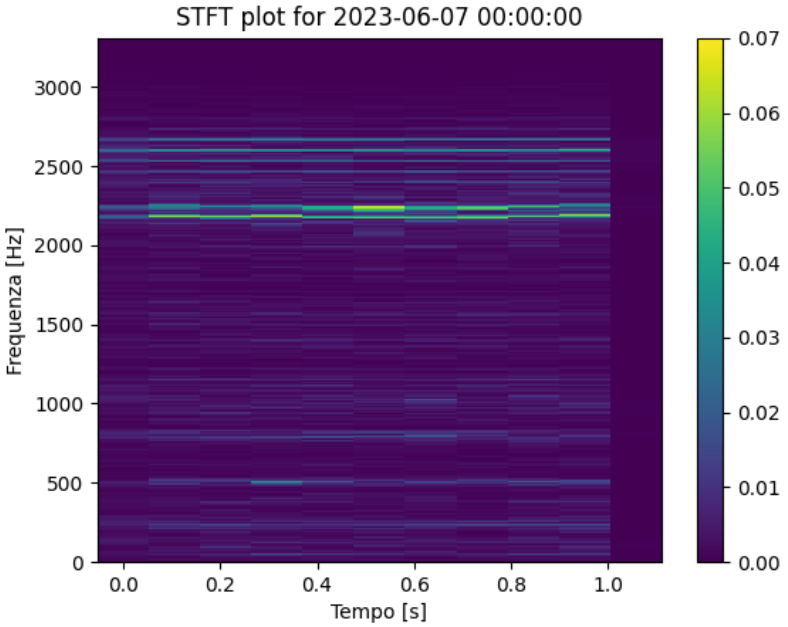


Figure 60: Short Time Fourier Transform application under normal condition

Due to the STFT's ability to highlight the state in which the machine operates, it was decided to implement this technique in the model.

After that, it was decided to apply a Denoising technique to the signal to reduce the noise that could disturb the application of the various techniques in the time domain. In general, Denoising techniques play a crucial role in many Signal Processing applications by improving signal quality and making it easier to analyze and extract useful information from the data.

The implemented Denoising technique was illustrated in the previous Chapter. Dynamic thresholding was used based on the maximum value assumed by each sampled signal. As can be seen from the figure below, the result was a “cleaner” raw signal.

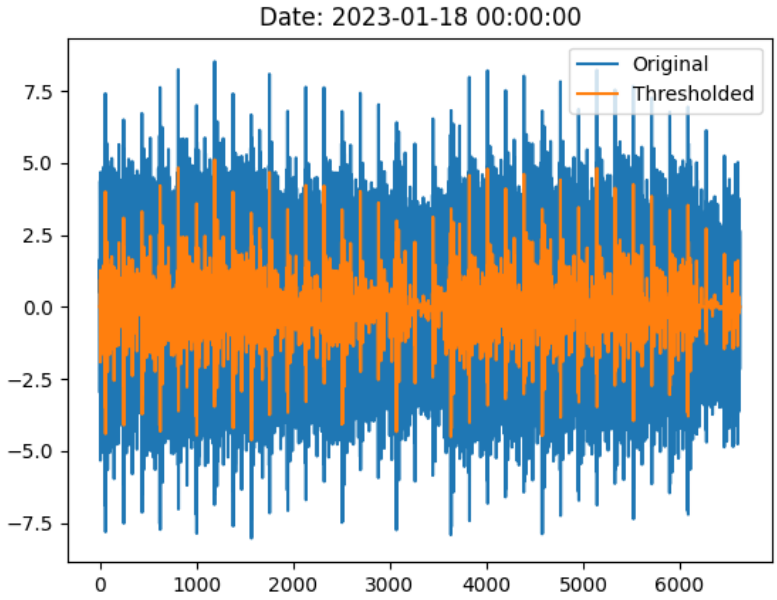


Figure 61: Visualization of raw data before and after application of the denoising technique

All the Signal Processing techniques presented were reapplied to the "cleaned" signal obtained by the Denoising technique. With the results of these, a Dataset was constructed with which the ML algorithms will be trained.

The following two figures show the application of the FFT to the same signals shown in the previous figures after these have been reduced and "cleaned" by the Denoising technique. As can be seen, the powers of the frequency components in this case assume reduced values compared to the original signal. Furthermore, both spectra in this case better highlight the most stressed frequencies by reducing the powers of the less significant frequencies.

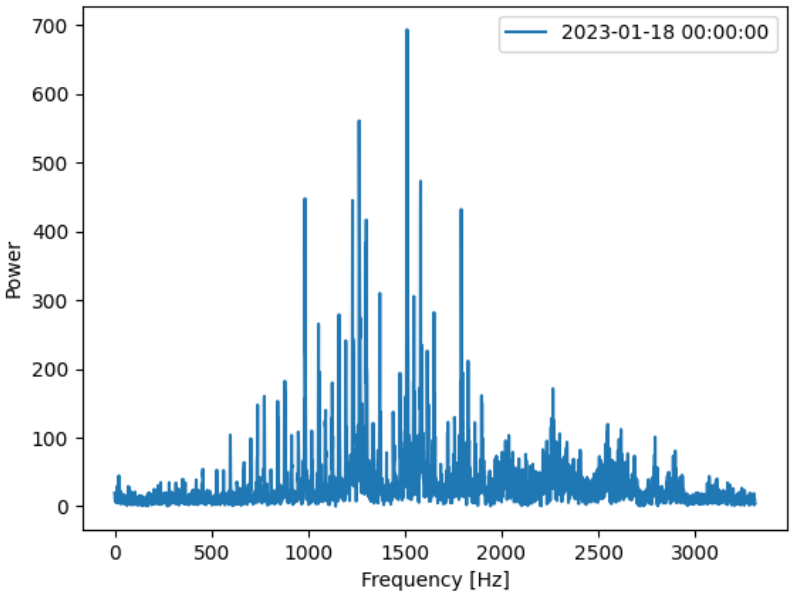


Figure 62: FFT spectrum under fault conditions after the application of the denoising technique

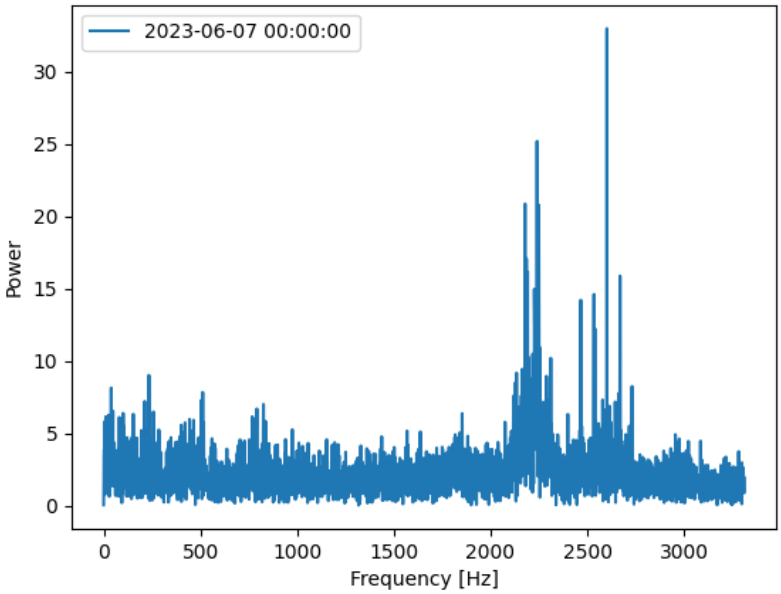


Figure 63: FFT spectrum under normal conditions after the application of the denoising technique

In this section, the results obtained through the application of various Signal Processing techniques to the vibrational analysis of the data acquired from the engine under consideration were presented. The main objective of this phase was to acquire an in-depth understanding of the behavior of the vibrational signal under different machine operating conditions and to identify any significant variations that could be indicative of imminent

faults. In this way, it is possible to establish which techniques are capable of showing considerable variations depending on the operating state of the machine.

The results obtained demonstrate clearly the ability of Signal Processing techniques to detect and capture variations in the vibrational signal when the machine is subjected to critical or fault conditions. These variations provide a clear indication of the presence of problems or anomalies, allowing early detection of fault conditions. Time domain analyses, the application of the Fourier Transform and the Short-Time Fourier Transform (STFT) provided a complete picture of the dynamic characteristics of the vibration signal. The implementation of the Denoising technique further improved the ability to detect variations depending on the operating condition of the plant, removing noise from the raw data and highlighting relevant variations.

The final purpose was to evaluate which Signal Processing techniques could be effective and strategic for the development of the model. After identifying Signal Processing techniques, various Datasets were structured, depending on the techniques implemented, with which different Machine Learning algorithms will be trained. In this way, it is possible to determine which Signal Processing techniques and which ML algorithm are most effective in recognizing an engine failure in advance in order to proceed with efficient and safe maintenance.

5.2 Results of the Machine Learning algorithms application in the Fault Recognition phase

In the previous section, the Signal Processing phase was presented in detail. This stage allowed the acquisition of valuable information from the vibrational signal of the machine under study. This processed data represents the fundamental basis for the development of an effective Predictive Maintenance model.

The next step following the Signal Processing phase of our approach was the Fault Recognition stage, which involved the training of various Machine Learning algorithms in order to develop a machine state recognition model. Such a model is crucial for the early identification of critical conditions or impending faults, thus enabling predictive and timely maintenance action.

In this Chapter, the results of training four Machine Learning algorithms will be presented: the Decision Tree, the Random Forest, the K-NN (K-Nearest Neighbors) and the Support

Vector Machine (SVM). Each algorithm was trained using different datasets, each containing distinct information obtained from the Signal Processing phase.

The goal of this approach is to evaluate the effectiveness of the different Signal Processing techniques and to identify the most suitable prediction algorithm for our specific case. Different Datasets were used to train the prediction algorithms, including those combining results in the time domain, frequency domain and the application of signal Denoising techniques. The objective is to evaluate which combination of Dataset and ML algorithm offers the best performance for our Predictive Maintenance application.

However, it is important to note that the dataset containing the Short Time Fourier Transform (STFT) was not included in the training phase, since the output of the STFT consists of complex numbers, and the selected algorithms require real data in the training phase. Furthermore, the SVM (Support Vector Machine) algorithm was not implemented due to the extremely long training times, which were impractical on both local hardware (more than 48 hours of training) and cloud resources (more than 9 hours of training), such as Google Colab.

In this section, concerning the training of Machine Learning algorithms, will be presented the accuracy obtained, training times, confusion matrix and discussions on the results. These findings will be crucial in understanding the overall effectiveness of the model and in identifying the next steps for creating an advanced and customized monitoring and predictive maintenance system for the company.

Three different Datasets were used for training the algorithms:

- Time Domain Dataset: contains the outputs of the Signal Processing techniques in the time domain, temperature information and motor shaft rotation speed.
- FFT Dataset: contains the outputs of the Signal Processing techniques in the time domain, the outputs of the Fast Fourier Transform, information on the temperature and speed of rotation of the motor shaft.
- Denoising Dataset: contains the same information of the FFT Dataset with the difference, that all techniques were applied after a denoising operation to "clean" the raw signals.

Furthermore, two different approaches were explored in the creation of the training and test sets. The first, considered more efficient, involved a subdivision based on prior knowledge of the machine's historical conditions. This method made it possible to balance the classes

within the test set, ensuring the presence of all categories to be predicted. In this case, the temporal sequence of events was not strictly followed, focusing instead on the objective of obtaining a representative test set.

On the other hand, a second subdivision strategy was adopted that strictly followed the chronological order of events. In this case, the training set included data from the first day of analysis (01/01/2022) until it represented approximately 75% of the size of the original dataset. Consequently, the test set included the dates and the remaining information. However, it should be noted that this second approach resulted in the non-presence in the test set of some classes previously seen during the training of the model. These details have been presented and discussed in detail in the previous Chapter.

The performance of the algorithms will be evaluated on the basis of the different Datasets, but also on the basis of the two approaches for creating the training and test sets.

5.2.1 Results obtained using the first approach of creating training and test sets

This section will present the results obtained from the Fault Recognition phase by applying Machine Learning algorithms. In particular, this section will discuss the performance of the ML algorithms trained by the different Datasets that have been subjected to data splitting on the basis of prior knowledge of the machine state.

The following figure represents a heat map of the correlations between the features in the training set obtained from this first subdivision strategy.

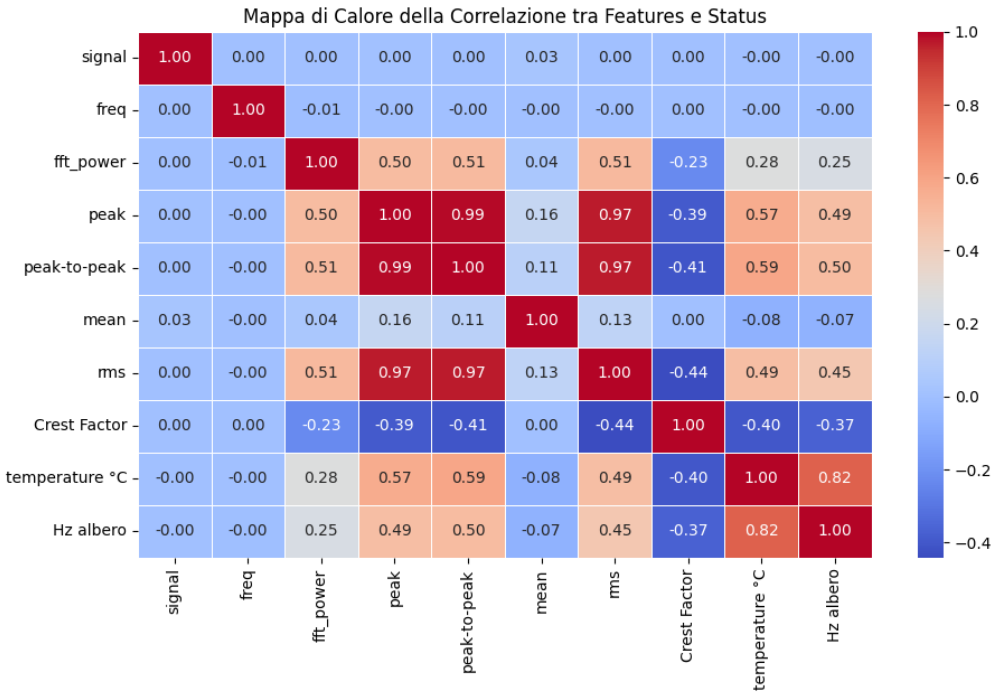


Figure 64: Heatmap of correlations between features using the first subdivision strategy

In the field of data analysis and Machine Learning, understanding the relationships that exist between variables or 'features' in a dataset is of crucial importance. The correlation heat map is a powerful and informative graphical tool used to visualize these relationships in a clear and intuitive manner. A correlation value of 1 indicates a perfect correlation between two features, while a value of -1 indicates a perfectly inverse correlation.

Based on the training sets and the correlations between the features shown, the classification algorithms will be trained. In particular, for each Dataset created, the algorithms will be trained and evaluated.

In the table below, we see the results obtained from the training of the algorithms using the Denoising dataset.

Denoising Dataset	**Decision Tree**	**Random Forest**	**k-NN**
Accuracy	72%	65%	57%
Training Time (s)	49	327	98

As can be observed, Decision Tree is the algorithm that performed best using this Dataset in the training phase, achieving an accuracy of 72%. It is also the one that requires the minimum training time, stabilizing below the one-minute threshold.

In contrast, the results obtained from the application of the FFT Dataset are:

FFT Dataset	**Decision Tree**	**Random Forest**	**k-NN**
Accuracy	68%	62%	54%
Training Time (s)	36	228	92

Instead, applying the Dataset containing only the outputs of the time domain techniques and the information of motor temperature and drive shaft rotation speed, the following performances were obtained:

Only Time Dataset	**Decision Tree**	**Random Forest**	**k-NN**
Accuracy	67%	66%	53%
Training Time (s)	32	102	105

In the analysis of the results obtained from the training of the machine learning algorithms, a series of relevant considerations emerged. The use of three different datasets, each characterized by specific data pre-processing conditions, made it possible to assess the impact of such pre-processing on the performance of the algorithms.

With regard to accuracy, it emerged that the Denoising dataset produced the highest results among the three datasets. In fact, in this application the Decision Tree (DT) achieved an accuracy of 72%, followed by the Random Forest (RF) with an accuracy of 65% and the K-NN with 57%. This suggests that the application of the denoising technique to the raw signals significantly improved the prediction ability of the algorithms. However, it is important to note that despite the improved performance, training with the Denoising Dataset required significantly more time, particularly for the Random Forest.

The FFT Dataset showed intermediate performance, with the accuracy standing at 68% for Decision Tree, 62% for Random Forest and 54% for K-NN. Also in this case, the use of Fast Fourier Transform information proved to be beneficial to performance, although not as much as denoising. In addition, training times were shorter compared to the Denoising Dataset.

Finally, the dataset containing only the outputs of the techniques in the time domain showed acceptable but slightly lower results than the other two Datasets, with a maximum accuracy of 67% for Decision Tree, 66% for Random Forest and 53% for K-NN. In general, this Dataset required the least training time. This is in line with what was expected, as the dataset

containing only the outputs of Signal Processing techniques in the time domain, being the simplest, is the one that shows on average the least time to train the algorithms.

In summary, Decision Tree obtained the best accuracy by exploiting the Denoising Dataset. Random Forest obtained the best results by exploiting the simplest dataset, the one with only the metrics obtained from the application of time domain techniques. While k-NN, recorded the best accuracy by exploiting the Denoising dataset.

The Random Forest is the algorithm that requires the most training time, while the Decision Tree is the fastest. As could be expected since the Random Forest is based on the use of multiple decision trees.

The best performing combination in terms of accuracy was therefore the Decision Tree applied to the Denoising Dataset. In order to investigate and understand more about the performance of this application, the Confusion Matrix will now be analyzed. The Confusion Matrix plays a fundamental role in predictive maintenance, as it allows an analysis of how the algorithm distinguishes between different machine state classes.

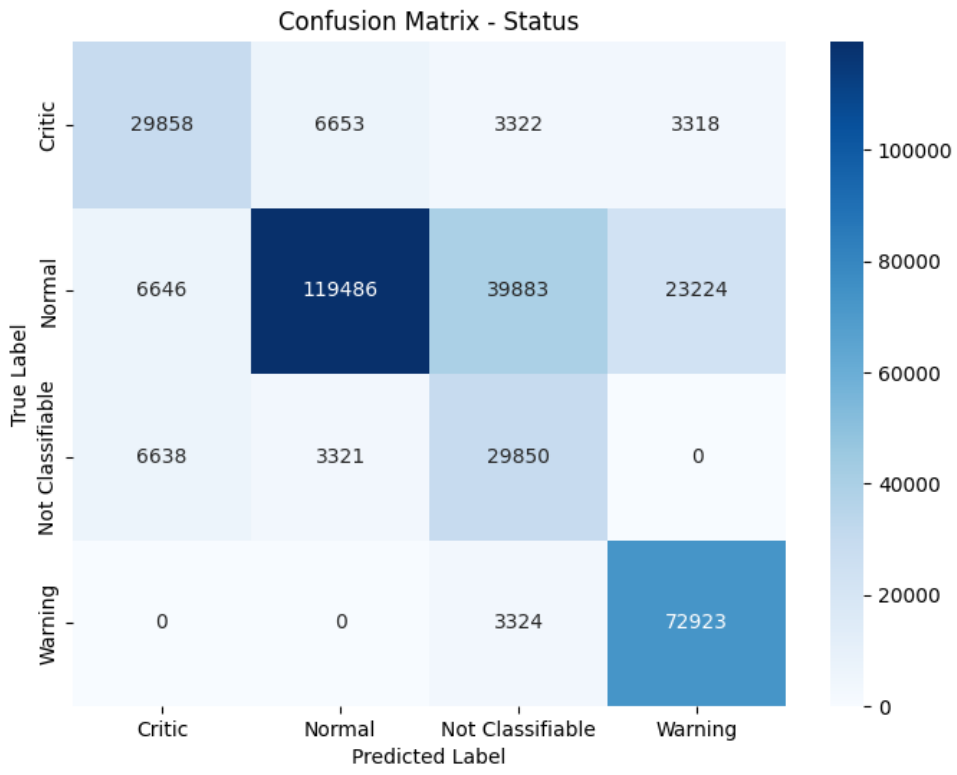


Figure 65: Confusion Matrix for the Decision Tree application

As can be seen, the algorithm experiences difficulties when it predicts the class 'Not Classifiable' when the correct prediction should have been 'Normal'. The 'Not Classifiable' class in this application relates to the last quarter of 2022 because the sensor had mistakenly separated from the engine and thus was no longer able to acquire machine vibrations correctly. However, the sensor continued to record vibrations caused by noise caused by the plant, operators, and by the environment more generally. The model's difficulty in recognizing this class is comprehensible and also acceptable by the company. In fact, the values obtained from the Signal Processing phase referring to the 'Not Classifiable' class are on average lower than those belonging to the 'Normal' class and extremely lower than those relating to the 'Warning' and 'Critical' classes. Consequently, a model mistake in predicting a 'Not Classifiable' state that would in reality be 'Normal' does not cause any real practical problems since it does not imply potential damage to the machine. It would be different if the algorithm predicted 'Normal' or 'Not Classifiable' class when the machine is operating in a fault or pre-fault condition. This situation would cause physical damage to the machine if the failure occurred thus making a Predictive Maintenance approach meaningless. However, as the Confusion Matrix shows, this type of error rarely occurred during the testing phase of the Decision Tree applied to the Denoising Dataset.

Continuing to investigate the best-performing combination, which is the application of Decision Tree to the results obtained from Signal Processing techniques once the raw signal has been "cleaned" of noise, an attempt was undertaken to improve performance. To try to improve the performance of this application, an investigation was conducted to understand how much and which features were most important to the model.

By exploiting a particular function of the 'DecisionTreeClassifier' it is possible to calculate the importance of features in the application of the model.

Finding the “most important features” means identifying those input variables that have the greatest impact or incidence on the model. However, it is important to note that the importance of a feature is relative to the context of the model and the specific dataset. In the figure below, the most important features for applying the Decision Tree to the Denoising Dataset are highlighted:

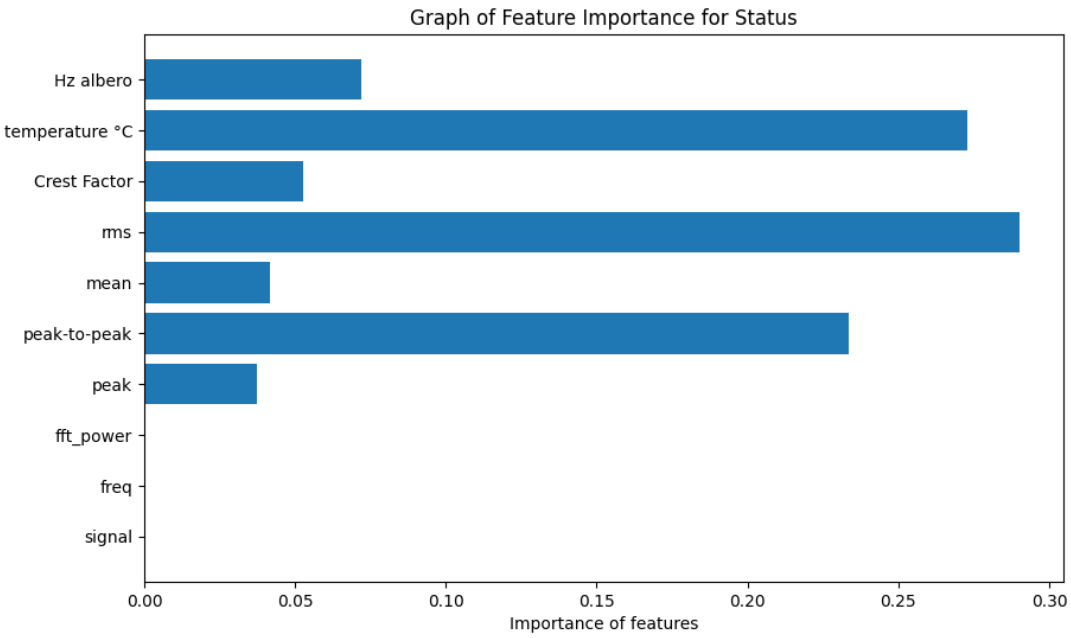


Figure 66: Visualisation of the features' importance for Decision Tree-Denoising dataset model

As can be observed, the features 'signal', 'freq' and 'fft_power' are poorly used by the model to make machine state predictions. The first feature represents the raw values measured by the accelerometer, while the remaining two features represent the outputs of the Fast Fourier Transform. Despite their unimportance to the model in consideration, even without considering them in the training phase of the algorithm the performance does not increase, on the contrary it decreases slightly.

In conclusion, the analysis showed that data preprocessing has a significant impact on the performance of machine learning algorithms. The application of the denoising technique combined with the Decision Tree classification capability led to the most promising results, but at the expense of longer training times. However, since these were differences of a few seconds between applications, the result obtained was considered acceptable. The Confusion Matrix suggests that the performance of this application could be further improved by reviewing the temporal partitioning of signals that were considered 'Not-Classifiable' a priori.

5.2.2 Results obtained using the second approach of creating training and test sets

In the context of analyzing the results obtained for the design of the Predictive Maintenance model, it is crucial to consider the effect of splitting the data into training and test sets. While the former strategy relies on prior knowledge of the engine state to create balanced datasets, the latter strategy takes a rigorous chronological approach, respecting the temporal order of measurements. This choice introduces a temporal dimension to the performance evaluation of machine learning algorithms.

In this section, we will explore in detail the results obtained using the second data partitioning strategy.

First of all, in the heatmap below are presented the correlations found among the features in the set used in training the algorithms:

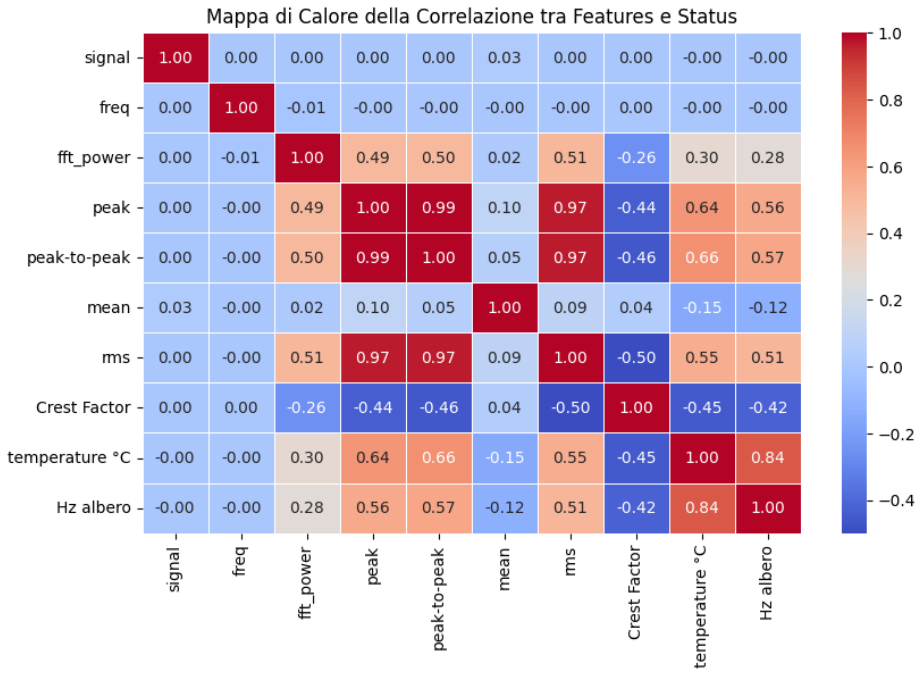


Figure 67: Heatmap of correlations between features using the second subdivision strategy

In general, using a stringent approach toward chronological sequencing, compared to the first strategy of dividing the Datasets into training and test sets, the correlations are slightly lower.

As with the previously presented approach, each Dataset is used to train the classification algorithms, and the performance of these, in terms of accuracy and training time, was explored.

In the table below we see the results obtained from training the algorithms using the Denoising Dataset.

Denoising Dataset	**Decision Tree**	**Random Forest**	**k-NN**
Accuracy	14%	8%	7%
Training Time (s)	38	198	105

Applying the FFT Dataset instead, the following results were obtained:

FFT Dataset	**Decision Tree**	**Random Forest**	**k-NN**
Accuracy	11%	6%	8%
Training Time (s)	28	197	74

Using the Dataset consisting only of the application of Signal Processing techniques in the time domain and the information of temperatures and production rates, the following outcomes were obtained:

Only Time Dataset	**Decision Tree**	**Random Forest**	**k-NN**
Accuracy	11%	7%	7%
Training Time (s)	22	93	173

In general, by using this data partitioning strategy, training times were reduced in almost all applications. In particular, Decision Tree and Random Forest dramatically reduced their training times. But this can be explained as the test set, using this approach, does not contain all the classes on which the algorithms were trained. As a result, prediction algorithms that rely on a 'decision tree' structure will perform fewer operations to make the prediction or classification.

Despite the reduction in training time, this strategy of partitioning Datasets resulted in very low performance values. The reasons for this failure may be several. First, the test set does not have all the classes to really test the training of the algorithm. In addition, the real world is characterized by changes in machine behavior over time due to the wear, maintenance, environmental conditions and other factors. The strategy based on the chronological order of measurements may not be able to effectively capture such changes if the dataset is not constantly updated to reflect them.

However, the main reason for the failure of this strategy lies in the significant class imbalance between the test set and the training set. In particular, the construction of the test set based on the chronological order of measurements resulted in an almost exclusive dominance of the 'Normal' class within the test set, while the training set contains only a limited amount of information related to the 'Normal' class compared to the other machine status classes. This significant imbalance between the classes of training and test data resulted in a poor ability of the model to recognize the state in which the machine was operating.

In conclusion, analysis of the results obtained from the different dataset partitioning strategies clearly showed the success of the first strategy based on creating training and test sets according to prior knowledge of the machine's historical status. This approach proved to be extremely advantageous in terms of accuracy in predicting the machine status.

In particular, the combination of Denoising Dataset and Decision Tree proved to be the most effective, with an accuracy of 72%. Despite some slight increases in training time compared to other applications of Decision Tree, the time still remains within acceptable limits for the company, thus providing a solid foundation for the success of the approach.

These results highlight that the combination of Signal Processing techniques, such as the time-domain techniques exposed at the beginning of the Chapter and the Fast Fourier Transform applied once the vibrational signal has been "cleaned" of noise, and machine learning algorithms, such as Decision Tree, has proven to be highly effective in recognizing the status of machines and provides a solid groundwork for the development of a reliable and efficient Predictive Maintenance system for the company.

5.3 Application and evaluation of the model to latest data

In pursuit of the goal of developing a reliable Predictive Maintenance model applicable to the business context, an additional evaluation phase was conducted. This phase aims to test the extent of validity of Machine Learning models trained using the first dataset partitioning strategy based on prior knowledge of machine condition.

In particular, we aim to examine how well the models perform on more recently collected vibrational data not included in either the training set or the test set originally used for

training and validation. This test represents a significant challenge as it involves the evaluation of whether the model identified as most effective, which is the Decision Tree trained using the Denoising Dataset, is able to maintain high performance even when subjected to more recent and previously unobserved signals.

Therefore, the goal of this evaluation phase is double: firstly, to assess the adaptability of the models to the latest machine conditions, and secondly, to confirm the effectiveness of the winning combination of algorithm and dataset identified in the previous phases. The presentation of the results of this test provides an additional level of insight into the model's ability to maintain high performance in detecting critical machine conditions and helps to consolidate the validity of the Predictive Maintenance approach developed for the company.

For this purpose then, the models previously trained with the best performing Dataset partitioning strategy were tested on the most recently collected vibrational signals not included in the model training and evaluation.

The vibrational signals collected by the sensor from the first of July 2023 until the seven of September 2023 were then extracted and processed.

After that, the classification algorithms (Decision Tree, Random Forest, and k-NN) previously trained with each Dataset were re-evaluated with these newer test datasets.

In recent months, the machine has always operated under Normal conditions, so the models are expected to predict this class of machine status.

The following table presents the performance obtained from the models previously trained with the Denoising Dataset and tested on the more recently collected vibrational data:

Denoising Dataset	**Decision Tree**	**Random Forest**	**k-NN**
Accuracy	91%	92%	61%

Applying the models trained with the FFT Dataset instead, the following outcomes were obtained:

FFT Dataset	**Decision Tree**	**Random Forest**	**k-NN**
Accuracy	72%	74%	54%

Finally, the accuracies obtained by applying the algorithms trained with the Dataset containing only the techniques in the time domain are:

Only Time Dataset	**Decision Tree**	**Random Forest**	**k-NN**
Accuracy	71%	73%	58%

The additional testing conducted on the vibrational data collected in a more recent period provided important confirmations and insights regarding the effectiveness of the Predictive Maintenance models developed. In particular, it was found that the Denoising Dataset continues to emerge as the best performing of those considered, with Decision Tree recording an accuracy of 91%, Random Forest with a remarkable 92%, and K-NN reaching 61%.

It is interesting to note that the Random Forest, in general, proves to be the highest performing algorithm, while in the previous test the Decision Tree emerged as better. Even if, again, the Decision Tree performs likewise well with the Denoising Dataset with only one percentage point difference from the Random Forest. It should also be remembered that Decision Tree involves less training time than Random Forest training.

The FFT Dataset, as a confirmation of the test by performing previously, provides intermediate results among the three datasets used for algorithm training, confirming its usefulness.

However, it is important to note that the K-NN shows lower performance overall than the other two algorithms for this real application. However, it should be kept in mind that it remains a valuable component in the Predictive Maintenance toolkit.

Wanting to investigate more in depth the superior performance, although slightly, of the Random Forest compared to the Decision Tree the confusion matrices of the application of these two algorithms trained with the Denoising Dataset on the most recent vibrational data are shown below.

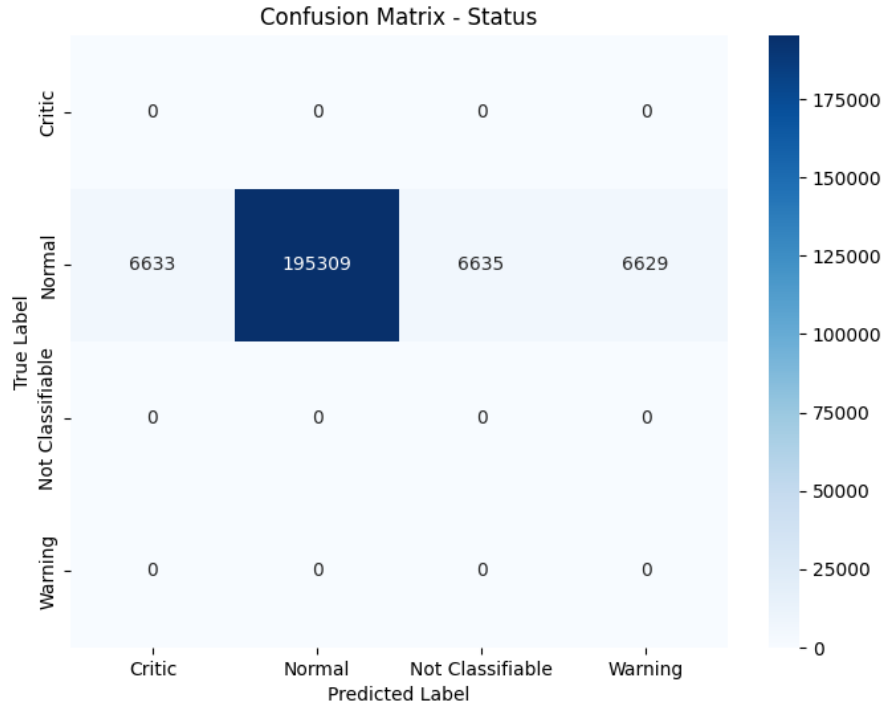


Figure 68: Decision Tree - Confusion Matrix

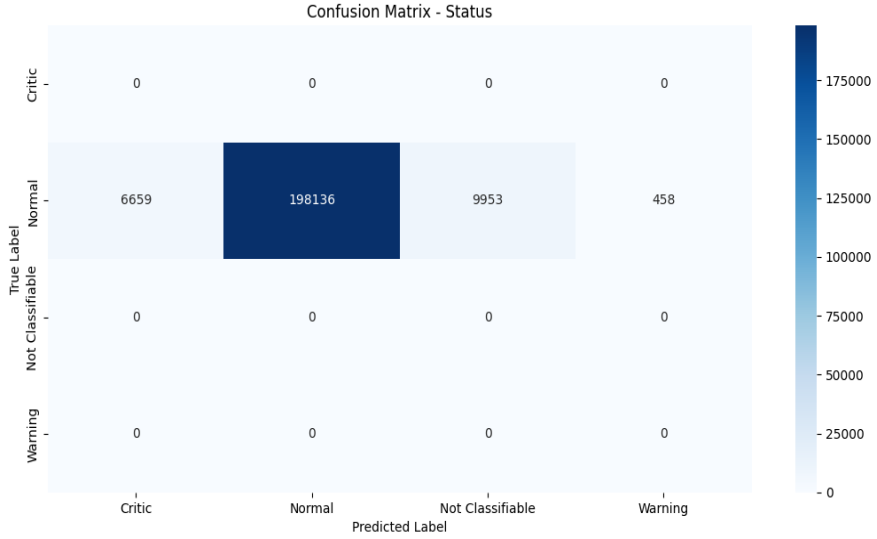


Figure 69: Random Forest - Confusion Matrix

As can be seen from the confusion matrices, the Decision Tree shows more difficulty in predicting the state of the Normal machine by incorrectly predicting the Warning class. On the other hand, the Random Forest is slightly more inclined to the error of predicting the 'Not Classifiable' class where the actual state of the machine is Normal.

It was desired to investigate the Confusion Matrices of the application of the two algorithms as it was expected that the Decision Tree would fail more than the Random Forest in predicting the class 'Not Classifiable' where the operating conditions were 'Normal'. In fact, it was explained previously that this type of error does not cause problems for the company as it does not generate any alarmism. On the other hand, it is different if the algorithm provides a 'Warning' or 'Critic' class where instead the actual conditions are Normal; in fact, in this case a false alarm is created that may result in an unnecessary maintenance intervention. However, what happened was not as expected. In fact, from the Confusion Matrix it can be seen that the Decision Tree would have generated more false alarms than the Random Forest.

This further evaluation shows that the Random Forest has higher performance than the other algorithms in diagnosing the Normal class, although the Decision Tree performed very good in this case as well. In addition, this test highlights that the best strategy for partitioning datasets is definitely the one based on a priori knowledge of the machine state, without strictly respecting the chronological order of events.

5.4 Training and evaluation of a new model using the latest data

In the context of evaluating the robustness and predictive capabilities of the developed Predictive Maintenance model, an additional significant test was conducted. This test aims to explore how the previously trained Machine Learning algorithms react to a new dataset. Therefore, this is to represent a realistic situation in which new information are continuously acquired and incorporated into the system.

Specifically, a new training set was created that covered the entire period of available historical data, from January 2022 to June 2023, using the Denoising Dataset approach. These data now present in one training set were those divided between training set and test set in the first test performed. The Signal Processing techniques present in the Denoising Dataset were applied to this dataset as these emerged as more accurate methods for the case study under consideration. Then, the three Machine Learning algorithms, Decision Tree, Random Forest and k-NN, were trained on this new dataset and evaluated using a test set consisting of the raw data collected during the most recent period.

This approach makes it possible to examine how previously trained models behave when exposed to updated data and, in particular, to assess whether models previously identified as the best performers continue to maintain their predictive advantage. The results of this final test provide a crucial insight into the resilience of the model and its ability to adapt to changing conditions, which are key aspects from the perspective of a Predictive Maintenance approach.

The following table shows the new training times of the algorithms, and especially the accuracies they achieved.

Denoising Dataset	**Decision Tree**	**Random Forest**	**k-NN**
Accuracy	78%	94%	72%
Training Time (s)	49	327	98

The table shows that Random Forest additionally improves its performance in terms of accuracy. However, it requires the most training time. This is due to the fact that the training set is larger in size than those used previously.

The k-NN also further improves its performance in terms of accuracy.

Decision Tree, on the contrary, decreases its performance compared with the last testing.

In general, compared with the application case in which the training set and test set were split on the a priori knowledge of the machine state, all three algorithms improved their performance. However, comparing them with the previous test performed on the same test set, Random Forest and k-NN improved their performance. In contrast, Decision Tree decreased its accuracy.

However, by conducting other optimization attempts, it was observed that by decreasing the size of the training set the Decision Tree's accuracy increased, even returning to values around 90%.

Summarizing, as the size of the training set increases, the performance of Random Forest and K-Nearest Neighbors also increases. The Decision Tree, on the other hand, showed criticality in this regard, decreasing its accuracy as the size of the dataset increased. It should be mentioned, however, that the Decision Tree has the shortest training time.

Finally, a further investigation was conducted to try to interpret these results. Specifically, it was intended to see if the misclassified data were close to each other or if they were scattered throughout the signal. This would help to understand whether an algorithm is wrong in predicting a set of data that share common traits with each other. For this purpose, the next visualizations aim to present the correct and wrong predictions of the three algorithms on the test set under consideration. On the x-axis there will be the numbers of samples observed in the test set. On the y-axis, instead, there will be the values of the signals acquired by the sensor. For each value observed by the sensor, the prediction is plotted. In this way, the distribution of correct or incorrect predictions can be verified.

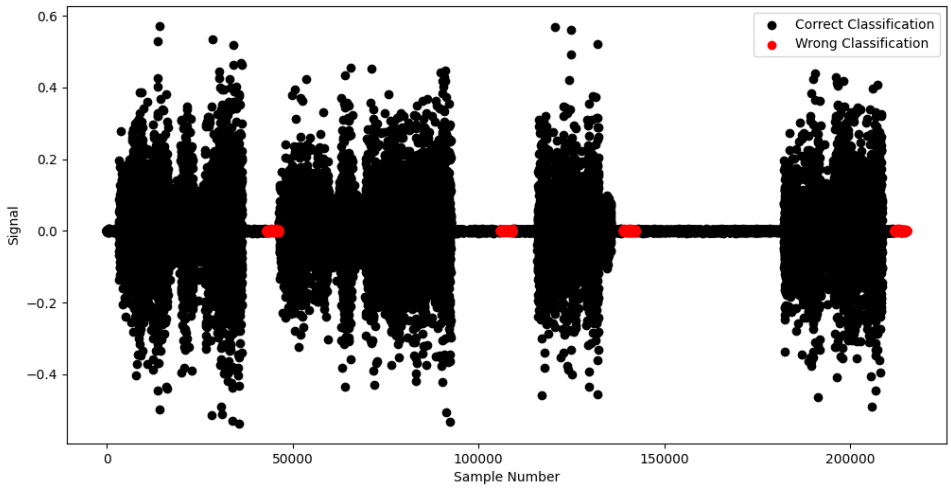


Figure 70: Distribution map of Random Forest classifications

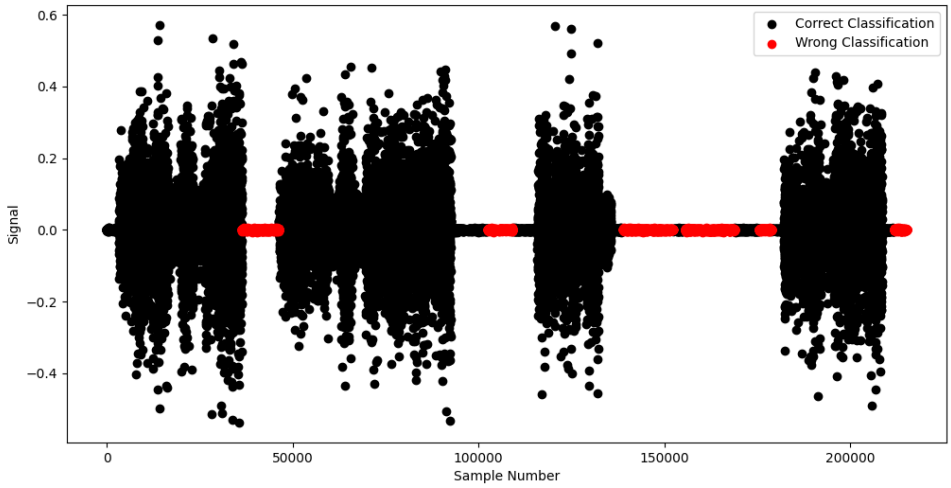


Figure 71: Distribution map of Decision Tree classifications

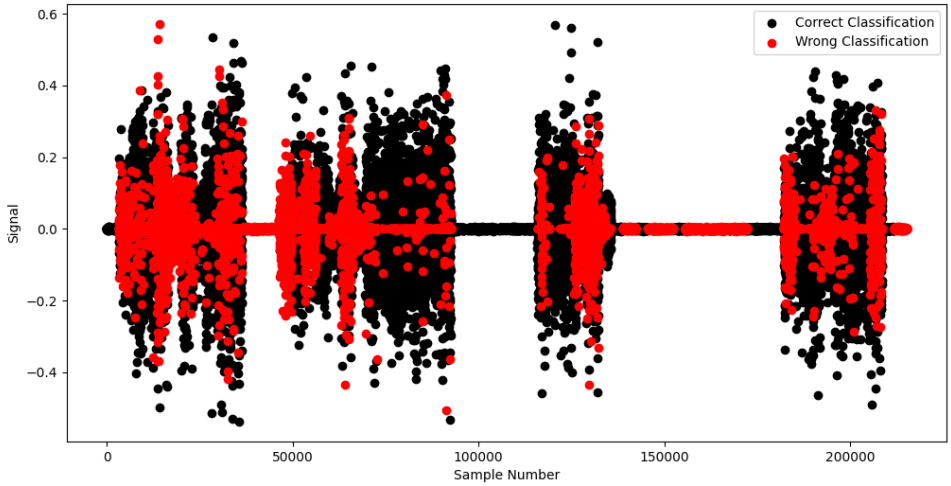


Figure 72: Distribution map of k-NN classifications

From the distribution maps of the algorithm classifications, many things were understood. The Random Forest shows how it has the best accuracy. However, it can be seen that the few classifications missed by the algorithm always fall where the machine is standing still. On the other hand, the Decision Tree looking only at the performance in terms of accuracy would have been the second best performing model. However, when analyzing the distribution map of the classifications, it can be seen that all of the misclassifications also have in common that they struggle to predict the 'Normal' class when the machine is stopped. Most of the misclassifications in fact fall after half of the samples, this time period dating back to the vibrations collected in August. In August there was the company closure and therefore production was not activated. Finally, k-NN proves to be the least performing algorithm of the three with classification errors scattered throughout the signal.

This analysis is therefore of considerable importance. In fact, it can be concluded that the Decision Tree misclassifies the class 'Normal' when the machine is standing still. This problem could be solved by filtering for production speeds greater than zero, or by assigning a special class to the situation when there is a production stop. In this way, the Decision Tree could also achieve accuracies above 90%. On the other hand, the Random Forest proves to be the best performing model in terms of accuracy, capable of recognizing downtime and thus predicting the 'Normal' class.

The last test represents the outcome of the application of the second strategy of splitting the Datasets. In fact, the training set and the test set strictly respect the chronological order of events. This strategy, this time greatly improved the performance of the models as the classes

were well distributed between the training set and the test set. In fact, the class 'Normal' in this case is well distributed between the training set and the test set.

In conclusion, the detailed analysis of the results obtained from a series of significant tests within this study clearly demonstrates the effectiveness and relevance of the Predictive Maintenance approach based on vibrational analysis and the use of Machine Learning algorithms.

The application of Signal Processing techniques, including those in the time domain, FFT and STFT, followed by the Denoising strategy, enabled the accurate capture of variations in machine signals under normal and critical conditions. This process demonstrated the ability to reliably identify impending faults, representing a crucial step toward more efficient, economical, and safe maintenance.

In addition, the comparative evaluation of different Machine Learning algorithms emphasized the importance of carefully selecting the combination of dataset and algorithms to maximize model performance. The Random Forest performed exceptionally well in predicting the normal status, improving the performance of this model with an increase in the size of the training set. Decision Tree showed flexibility in recognizing different classes when all were present in the test set but showed a decrease in performance as the size of the training dataset increased. However, it was noticed how it could be a high-performance model when handling downtimes. In fact, the Decision Tree works very well when the machine is operating, but misclassifies when the machine is standing still. Meanwhile, k-NN was shown to benefit from the increase in training set size, although it performed less well than the other two algorithms.

Overall, although the preferred combination is the Random Forest trained with the Denoising Dataset, the Decision Tree based model could be of support. Both models, based on the Denoising Dataset and trained with either the Decision Tree or the Random Forest, thus represent valuable resources to support maintenance decisions and preventive activities during emergencies.

The implementation and approach employed in the design of a Predictive Maintenance model based on vibration analysis has demonstrated satisfactory results. This study provided

tangible evidence of the effectiveness of Signal Processing techniques in the time and frequency domain, especially when applied after a signal Denoising phase.

These results highlight the substantial value of a Predictive Maintenance approach for companies seeking to improve the reliability of their industrial equipment. The adoption of this model not only enables early detection of failures and the avoidance of costly downtime, but also provides a clear demonstration of how data-driven decisions can significantly improve overall operational efficiency and business competitiveness. Predictive Maintenance represents a proactive strategy that, as demonstrated, can be successfully supported by advanced data analytics and Machine Learning algorithms, enabling companies to achieve new levels of reliability and productivity.

CONCLUSIONS

The advent of Industry 4.0 has significantly transformed the manufacturing sector by leveraging new digital technologies, automation of production processes, plant connectivity (IoT) and Artificial Intelligence (AI). In such a context, where the company is inclined to invest in digital transformation with the goal of becoming smart factory, Predictive Maintenance is becoming increasingly relevant.

Predictive Maintenance is a banner methodology of Industry 4.0 with the aim of preserving the operation and health of industrial plants. Predictive Maintenance, essentially, is based on the ability to obtain real-time data and analyze it, through Machine Learning algorithms, to predict when an industrial equipment or machine component will need maintenance or assistance. The objective is to increase the service life and reliability of the machine as much as possible by performing maintenance activities at the most convenient time for the company.

The theoretical foundations of Predictive Maintenance go back decades, however, becoming popular only recently. Its surprising actuality comes from the advancement of today's technologies. Real-time data acquisition and the computational capabilities of algorithms capable of processing huge amounts of data have enabled the practical advent of Predictive Maintenance.

The benefits of implementing Predictive Maintenance are numerous. First and foremost, it allows a significant reduction in maintenance costs as maintenance is carried out only when really needed. Production optimization is another benefit, as predicting the fault organizes targeted and effective maintenance without the risk of having to stop production due to a sudden failure. In addition, the ability to predict failure also means improving operator and environmental safety and optimizing inventory management.

For RDM Group, by presenting continuous production, the ability to predict a failure assumes a crucial role, as it allows to avoid production downtime that could last for a long period. In addition, the economic benefits of performing maintenance only when really needed and not on a preventive basis have sparked the company's interest in taking a Predictive Maintenance approach.

The present project addressed the challenges that the field of Predictive Maintenance brings through an approach that combined vibrational analysis with the implementation of Machine Learning algorithms. Specifically, the study was conducted on a rotating machine at an RDM Group manufacturing plant.

Through vibration analysis and the use of Machine Learning algorithms, important results were achieved that demonstrate the potential of this methodology in the optimization of plant maintenance.

Key phases of this research were the data acquisition and Signal Processing phases. The former was made possible by sensors, mono-axial accelerometers, interconnected with each other. The subsequent accurate analysis of the acquired signals made it possible to accurately detect variations in the operating conditions of the machines. The study included an in-depth analysis of different Signal Processing techniques, including time domain techniques, frequency domain techniques, time & frequency domain methods and denoising techniques.

The creation of four distinct datasets, each created from different Signal Processing techniques, allowed for an exhaustive exploration of the potential of each technique. In particular, the Denoising Dataset emerged as the best performing, demonstrating the importance of applying denoising techniques to the raw data before proceeding with analysis. The final aim of creating different datasets was to assess which information was most meaningful for training Machine Learning algorithms for the case study under consideration. This multi-level approach proved to be effective in identifying faults and critical machine conditions, improving the accuracy of the final model.

Each Dataset, not only includes the outputs of the Signal Processing techniques, but also motor temperature information and information on the rotational speed of the motor shaft. This last fact proved to be fundamental for the training of the algorithms, since the vibration level does not only increase due to the presence of an anomaly or fault, but also increases as the production speed increases.

The last phase explored in this research was Fault Recognition. This stage concerns the training of Machine Learning algorithms to predict and recognize the state of the machine. The training and evaluation of three different Machine Learning algorithms, respectively Decision Tree, Random Forest and k-NN, showed that Random Forest achieved the highest performance in terms of accuracy and normal state recognition capability. This algorithm was demonstrated to improve further as the size of the training dataset increased, providing

reliable and consistent results. The Decision Tree, although flexible in recognizing different classes, showed a tendency to decrease in performance as the size of the training dataset increased. However, it has proven to be a valuable resource especially if downtimes are managed differently. In this way, its accuracy would also be significantly increased. The k-NN, although it improved performance with larger datasets, performed lower than the other two algorithms.

After a rigorous series of tests and the analysis of various datasets, it is evident that the best performing model has emerged from the combination of the Random Forest algorithm and the Denoising dataset. The latter dataset, composed of raw data subjected to advanced denoising techniques, was further enriched with signal processing techniques in the time domain, the application of the Fast Fourier Transform, and the inclusion of crucial information such as engine temperature and production speed. This combination has proven to be highly effective in accurately recognizing machine status, with superior accuracy and predictive capability compared to other configurations. However, the application of the Decision Tree model remains a valuable resource that can assist in decision-making.

There are several opportunities for future improvement and development that can be undertaken to further fine-tune the proposed model. Firstly, a key area for improvement could be the optimization of dataset partitioning. The recognized importance of this division strategy between training and test sets, as demonstrated by the results obtained, suggests the possibility of exploring further approaches to maximize the effectiveness of the model. Secondly, the continuous acquisition of data and the accumulation of experience in the field will be a key opportunity to improve the model. Additional data coming from machines during production and new failure cases will enrich the training set and allow the model to gain a deeper understanding of system dynamics. Finally, the scope of the model's capabilities can be extended. Besides recognizing the state of the machine, it is possible to explore the idea of having a model be able to identify the exact location of the fault. This capability would allow maintenance activities to be directed more precisely, minimizing downtime and optimizing operational efficiency even more.

In conclusion, the research and development path addressed in this thesis led to significant and highly promising results in the field of Predictive Maintenance. Through a careful and systematic analysis of data pre-processing techniques, Machine Learning techniques and

different dataset configurations, an extremely effective and versatile Predictive Maintenance model has emerged.

This project represents a significant contribution in the field of Predictive Maintenance and offers a solid basis for further developments and practical applications in industry. The adoption of this solution will enable the company to improve resource management, the safety in the workplace, prevent costly failures and increase operational efficiency, demonstrating the tangible value of Predictive Maintenance in the modern industrial context.

BIBLIOGRAPHY

[1] AHMED, Umair; ALI, Fakhre; JENNIONS, Ian. A review of aircraft auxiliary power unit faults, diagnostics and acoustic measurements. Progress in Aerospace Sciences, 2021, 124: 100721

[2] SHUKLA, Khyati; NEFTI-MEZIANI, Samia; DAVIS, Steve. A heuristic approach on predictive maintenance techniques: Limitations and scope. Advances in Mechanical Engineering, 2022, 14.6: 16878132221101009.

[3] GOYAL, Deepam; PABLA, B. S. The vibration monitoring methods and signal processing techniques for structural health monitoring: a review. Archives of Computational Methods in Engineering, 2016, 23: 585-594

[4] MOHD GHAZALI, Mohamad Hazwan; RAHIMAN, Wan. Vibration analysis for machine monitoring and diagnosis: a systematic review. Shock and Vibration, 2021, 2021: 1-25.

[5] SCHEFFER, Cornelius; GIRDHAR, Paresh. Practical machinery vibration analysis and predictive maintenance. Elsevier, 2004.

[6] https://www.docenti.unina.it/webdocenti-be/allegati/materiale-didattico/77199.

[7] https://mathematicalmysteries.org/sine-wave/

[8] ALTHUBAITI, Adnan; ELASHA, Faris; TEIXEIRA, Joao Amaral. Fault diagnosis and health management of bearings in rotating equipment based on vibration analysis–a review. Journal of Vibroengineering, 2022, 24.1: 46-74.

[9] LEBOLD, Mitchell, et al. Review of vibration analysis methods for gearbox diagnostics and prognostics. In: Proceedings of the 54th meeting of the society for machinery failure prevention technology. Virginia Beach, VA, 2000. p. 16.

[10] AL-BALUSHI, Khamis R., et al. Energy Index technique for detection of acoustic emissions associated with incipient bearing failures. Applied Acoustics, 2010, 71.9: 812-821.

[11] PATIDAR, Shyam; SONI, Pradeep Kumar. An overview on vibration analysis techniques for the diagnosis of rolling element bearing faults. International Journal of Engineering Trends and Technology (IJETT), 2013, 4.5: 1804-1809.

[12] https://power-mi.com/content/rolling-element-bearing-components-and-failing-frequencies

[13] BOUDIAF, Adel, et al. A summary of vibration analysis techniques for fault detection and diagnosis in bearing. In: 2016 8th International Conference on Modelling, Identification and Control (ICMIC). IEEE, 2016. p. 37-42.

[14] SAMATAS, Gerasimos G.; MOUMGIAKMAS, Seraphim S.; PAPAKOSTAS, George A. Predictive maintenance-bridging artificial intelligence and iot. In: 2021 IEEE World AI IoT Congress (AIIoT). IEEE, 2021. p. 0413-0419.

[15] AMRUTHNATH, Nagdev; GUPTA, Tarun. A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance. In: 2018 5th international conference on industrial engineering and applications (ICIEA). IEEE, 2018. p. 355-361.

[16] CHARBUTY, Bahzad; ABDULAZEEZ, Adnan. Classification based on decision tree algorithm for machine learning. Journal of Applied Science and Technology Trends, 2021, 2.01: 20-28.

[17] DAMANIK, Irfan Sudahri, et al. Decision tree optimization in C4. 5 algorithm using genetic algorithm. In: Journal of Physics: Conference Series. IOP Publishing, 2019. p. 012012.

[18] ABDULKAREEM, Nasiba Mahdi; ABDULAZEEZ, Adnan Mohsin. Machine learning classification based on Radom Forest Algorithm: A review. International journal of science and business, 2021, 5.2: 128-142.

[19] REIS, Itamar; BARON, Dalya; SHAHAF, Sahar. Probabilistic random forest: A machine learning algorithm for noisy data sets. The Astronomical Journal, 2018, 157.1: 16.

[20] AHMAD, Baseer, et al. Intelligent predictive maintenance model for rolling components of a machine based on speed and vibration. In: 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC). IEEE, 2021. p. 459-464.

[21] PINHEIRO, Allan Alves; BRANDAO, Iago Modesto; DA COSTA, Cesar. Vibration analysis in turbomachines using machine learning techniques. European Journal of Engineering and Technology Research, 2019, 4.2: 12-16.

[22] KATARIA, Aman; SINGH, M. D. A review of data classification using k-nearest neighbour algorithm. International Journal of Emerging Technology and Advanced Engineering, 2013, 3.6: 354-360.

[23] WU, Sze-jung, et al. A neural network integrated decision support system for condition-based optimal predictive maintenance policy. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2007, 37.2: 226-236.

[24] HUANG, Yanbo. Advances in artificial neural networks–methodological development and application. Algorithms, 2009, 2.3: 973-1007.

[25] CAKIR, Mustafa; GUVENC, Mehmet Ali; MISTIKOGLU, Selcuk. The experimental application of popular machine learning algorithms on predictive maintenance and the design of IIoT based condition monitoring system. Computers & Industrial Engineering, 2021, 151: 106948.

[26] https://plat.ai/wp-content/uploads/Table1-2.png.webp

[27] https://www.skf.com/ph/productinfo/productid-6319%2FC3

[28] DOLENC, Boštjan; BOŠKOSKI, Pavle; JURIČIĆ, Đani. Distributed bearing fault diagnosis based on vibration analysis. Mechanical Systems and Signal Processing, 2016, 66: 521-532.

[29] https://www.aitertc.it/web/community/wiki-tc/teorema-del-campionamento-di-nyquist-shannon/#:~:text=il%20teorema%20di%20Shannon%2DNyquist,con%20la%20frequenza%20pi%C3%B9%20alta).

[30] https://numpy.org/doc/stable/

[31] https://www.w3schools.com/python/numpy/numpy_intro.asp

[32] https://pandas.pydata.org/docs/getting_started/overview.html

[33] https://docs.python.org/3/library/re.html#regular-expression-objects

[34] https://docs.scipy.org/doc/scipy/tutorial/index.html

[35] https://pywavelets.readthedocs.io/en/latest/

[36] https://pypi.org/project/scikit-learn/

[37] https://matplotlib.org/

[38] https://seaborn.pydata.org/