DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

Machine learning in predictive maintenance of industrial robots

SIMONE MORETTINI

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Machine learning in predictive maintenance of industrial robots

SIMONE MORETTINI

Master's Programme, ICT Innovation, 120 credits
Date: July 30, 2021

Supervisors: Ehsan Poorhadi, Jacob Edstrom
Examiner: Elena Troubitsyna
School of Electrical Engineering and Computer Science

Abstract

Industrial robots are a key component for several industrial applications. Like all mechanical tools, they do not last forever. The solution to extend the life of the machine is to perform maintenance on the degraded components. The optimal approach is called predictive maintenance, which aims to forecast the best moment for performing maintenance on the robot. This minimizes maintenance costs as well as prevents mechanical failure that can lead to unplanned production stops. There already exist methods to perform predictive maintenance on industrial robots, but these methods require additional sensors. This research aims to predict the anomalies by only using data from the sensors that already are used to control the robot.

A machine learning approach is proposed for implementing predictive maintenance of industrial robots, using the torque profiles as input data. The algorithms selected are tested on simulated data created using wear and temperature models. The torque profiles from the simulator are used to extract a health index for each joint, which in turn are used to detect anomalous states of the robot. The health index has a fast exponential growth trend which is difficult to predict in advance. A Gaussian process regressor, an Exponentron, and hybrid algorithms are applied for the prediction of the time series of the health state to implement the predictive maintenance. The predictions are evaluated considering the accuracy of the time series prediction and the precision of anomaly forecasting.

The investigated methods are shown to be able to predict the development of the wear and to detect the anomalies in advance. The results reveal that the hybrid approach obtained by combining predictions from different algorithms outperforms the other solutions. Eventually, the analysis of the results shows that the algorithms are sensitive to the quality of the data and do not perform well when the data present a low sampling rate or missing samples.

Keywords

Predictive maintenance, Industrial robots, Gaussian process regression, Exponentron, Hybrid algorithms, Time series prediction.

Sammanfattning

Industrirobotar är en nyckelkomponent för flera industriella applikationer. Likt alla mekaniska verktyg håller de inte för alltid. Lösningen för att förlänga maskinens livslängd är att utföra underhåll på de slitna komponenterna. Det optimala tillvägagångssättet kallas prediktivt underhåll, vilket innebär att förutsäga den bästa tidpunkten för att utföra underhåll på roboten. Detta minimerar både kostnaderna för underhåll samt förebygger mekaniska fel som kan leda till oplanerade produktionsstopp. Det finns redan metoder för att utföra prediktivt underhåll på industriella robotar, men dessa metoder kräver ytterligare sensorer. Denna forskning syftar till att förutsäga avvikelserna genom att endast använda data från de sensorer som redan används för att reglera roboten.

En maskininlärningsmetod föreslås för implementering av prediktivt underhåll av industriella robotar, med hjälp av vridmomentprofiler som indata. Metoderna testas på simulerad data som skapats med hjälp av slitage- och temperaturmodeller. Vridmomenten används för att extrahera ett hälsoindex för varje axel, vilket i sin tur används för att upptäcka anomalier hos roboten. Hälsoindexet har en snabb exponentiell tillväxttrend som är svår att förutsäga i förväg. En Gaussisk processregressor, en Exponentron och hybridalgoritmer används för prediktion av tidsserien för hälsoindexet för att implementera det prediktiva underhållet. Förutsägelserna utvärderas baserat på träffsäkerheten av förutsägelsen för tidsserien samt precisionen för förutsagda avvikelser.

De undersökta metoderna visar sig kunna förutsäga utvecklingen av slitage och upptäcka avvikelser i förväg. Resultaten uppvisar att hybridmetoden som erhålls genom att kombinera prediktioner från olika algoritmer överträffar de andra lösningarna. I analysen av prestandan visas att algoritmerna är känsliga för kvaliteten av datat och att de inte fungerar bra när datat har låg samplingsfrekvens eller då datapunkter saknas.

Nyckelord

Prediktivt underhåll, Industriella robotar, Gaussian process regression, Exponentron, Hybridalgoritmer, Prediktivt av tidsserier.

Acknowledgments

A special thanks go to my supervisor Jacob Edstrom and the other colleagues for giving me the possibility of working on this topic and for helping me out on the technical side of my thesis.

I would like to thank Ehsan Poorhadi, my supervisor at KTH, for his feedback and suggestions for this report. I also would like to thanks Jessica Kelly, my opponent, that gave me precious comments on my degree project.

Eventually, I thanks my family for giving me the possibility of following my master and specializing in my passions.

Stockholm, July 2021
Simone Morettini

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

ANN Artificial Neural Network

ARIMA Autoregressive integrated moving average

DoF Degrees of Freedom

ETS Exponential smoothing

FN False Negative

FNR False Negative Rate

FP False Positive

FPR False Positive Rate

GP Gaussian Process

GPR Gaussian Process Regressor

KPI Key Performance Indicator

LMA Levenberg–Marquardt Algorithm

MAPE Mean Absolute Percentage Error

ML Machine Learning

PdM Predictive Maintenance

PE Percent Error

RBF Radial Basis Function

RF Random Forest

RMSE Root Mean Square Error

RQ Rational Quadratic

TN True Negative

TP True Positive

Chapter 1

Introduction

The manufacturing industry has undergone multiple revolutions throughout the course of history and, nowadays, is inside the fourth industrial revolution. The aforementioned revolution is often referred to as Industry 4.0, which is a broad term. It describes the application of multiple technologies like the Internet of Things, big data, cloud computing and other systems to the industrial sector [1]. This revolution enables factories to improve in different aspects such as productivity, environmental impact and quality of products. A wide area that can benefit from these technologies is the maintenance of equipment and machines.

Most factories aim to automate as many processes as possible to optimize the use of resources. Machines that are central in most cases are the industrial robots that permit the substitution of human labour in a wide variety of tasks. Robots, like any machines, cannot operate forever. The extended usage causes pieces to deteriorate and not work as they should. The solution to avoid failures or wrong behaviours is to fix or substitute parts.

Maintenance extends the life of the robots but it comes with the cost caused by the necessary resources and the time during which the machine is not working for the main production objective. It is important that the benefits are more than the drawbacks and so maintenance has to be made intelligently. Predictive Maintenance (PdM) aims to predict when maintenance is needed on a machine by forecasting when the equipment will not work anymore as expected. The PdM is a wide research area that contains many different techniques and methods. Based on the situation and the type and quantity of data available, different algorithms can be selected. A family of solutions that perform well in PdM is Machine Learning (ML) algorithms. In fact, they are good at finding out patterns and trends on a high volume of data as the

ones produced by industrial machines [2][3]. Moreover, they require minimal intervention from humans and they can improve themselves by acquiring more information.

The following sections present a brief introduction to this research. First of all, the problem and the research methodology used to answer the research question are presented. Then, the main contribution and delimitation are highlighted. Eventually, the ethics and sustainability impact of this work are discussed.

1.1 Problem definition

PdM is a technique that aims to forecast the health state of equipment and tools to estimate when maintenance is needed. The focus of this project is on robotic manipulators or robotic arms, programmable mechanical arms capable of performing similar functions that can be achieved by the human arm. They are often used in industry to automate manufacturing processes. Though the research targets a robotic arm manipulator, it can be generalized to other domains to improve the lifetime of tools and equipment for industry or final users. The robots used in this research already implement an algorithm to evaluate the health state of the system. The method enables real-time detection of cases where the robot is not working nominally or is near a failure state. The index works by analyzing the current state of the robot. To implement a PdM strategy, the current state of the system is not enough and a forecast of the future state development is needed.

Purpose

The objective of this work is to find the best algorithm for PdM tasks for a robotic manipulator. The prediction of the robot's state of health is beneficial since it enables precise maintenance scheduling and reduces running downtime of the robots during the processes.

Research question

Is it possible to predict the wear behaviour that causes failure on robotic manipulators without adding extra hardware to the machines? Robots have data, likes torques, that are available for analysis without the introduction of specific sensors. Considering the precision of the prediction and the capacity of generalization, which is the optimal algorithm to predict the health of an

industrial robot?

Goals

The goals of this project needed to answer the research questions are the following:

- Creation of realistic torque data influenced by wear using a simulator. Fulfilling this point will enable the production of more data for validating the algorithm. In fact, the data available from real robots are very limited and does not permit the achievement of complete work.
- Implementation of multiple algorithms for forecasting the trend of the health status of the robots.
- Validation and analysis of the algorithms by running them on the data produced with the simulation and on the data obtained from failure cases that occurred on real robotic manipulators.

1.2 Research Methodology

The degree project is developed using a quantitative research methodology since the experiments and tests are supported by the analysis of measurable variables. The experimental method is used for developing and testing prediction algorithms. Eventually, a deductive approach is used to extract conclusions from the results.

Most of the degree project is carried out using an experimental research strategy design. In fact, the predictions are made using data produced by a simulator. An ulterior validation is made through ex post facto research using data recorded from real failure cases of a robotic manipulator. The simulated data are made under various scenarios by changing some aspects of the simulations: the load on the end effector, the temperature of the joint and the wear formation trend. These three aspects are selected based on common scenarios of the robot usage. These data permit to verify if the algorithms can generalize.

1.3 Contributions

The outcome of this research is an analysis of multiple solutions applied for implementing PdM for industrial robots. In literature, there are multiple

approaches of PdM using different sensors that are specifically added to the robotic manipulator. In this thesis, the novelty is in using only the torques applied in the robot joints to predict the health state of the robot.

The algorithms used for the predictions can be applied not only for PdM but to any time series forecast presenting an exponential growth trend. Specific attention is needed for these data. The classic algorithms based only on past data are not sufficient for performing accurate predictions. Finally, a detailed approach is proposed to implement the wear in the joints of industrial robots.

1.4 Delimitations

The data about failure cases of robotic manipulators available are not enough to correctly implement a PdM algorithm. For this reason, a simulation of these data is used. The main limitation of this project is the quality and the quantity of the data simulated. The wear formation and the corresponding friction are complex phenomena that depend on a large amount of factors. For the scope of this work, some of the parameters are approximated. Moreover, the backlash can not be modelled on the simulator used and so it is not considered. The quantity of data simulated, limited by time constraints, restricted the viable methods and some interesting solutions that need more data, like deep learning, had to be excluded.

1.5 Ethics and Sustainability

The implementation of PdM strategies for robots has an impact on ethics and sustainability. From a sustainability point of view, the consequences of this research are mainly two. First of all, the impact is on energy consumption. A high level of wear causes high friction that is needed to be overcome by the robot with higher forces and so with higher current consumption. Maintenance at the right moment allows the reduction of wasted energy due to too much friction. Secondly, a correctly planned prediction enables the optimization of the processes and a reduction of resources consumed.

Furthermore, the PdM supports the idea of repairing things before they break to reduce waste and increase the life of the object [4]. In fact, maintenance at the right moment can avoid significant failures of parts that otherwise would need a complete substitution instead of only a small-scale fix. The effect is a decrease in the amount of waste produced in the industry.

From an ethical point of view, the PdM will impact the work of service

personnel and technicians. Their work will be optimized and fewer resources will be wasted on unnecessary maintenance. The downside is a decrease in job security for the experts who currently plan the maintenance activities. However, their time can be more effectively focused on high probability failure cases or on the optimization of the maintenance activities.

1.6 Thesis Outline

In the following chapters, the project implementation and results will be described in details. Chapter 2 presents the relevant background information about PdM, industrial robots, ML and an analysis of similar works to the one of this project. The chapter is closed with the presentation of the theory behind the algorithms used in this work. Chapter 3 reports the methodology and method used to solve the problem and answer the research question. The datasets adopted are described together with the steps for pre-processing the data and the implementation of the algorithms. Eventually, the evaluation metrics to judge the results are discussed. Chapter 4 presents the results obtained along with an analysis, and in Chapter 5 a discussion of them is made. Finally Chapter 6 reports the conclusion of this thesis. This last chapter also proposes possible future works.

Chapter 2

Background

This chapter presents the background information about the main topics of this thesis. The field of PdM is described in the first section and then robotic manipulators are presented. A special focus is given to the joints of the robotic arm and the effect of the wear on the friction. In the last sections, the field of ML is presented together with related works. Eventually, the theory behind the algorithms used in this degree project is exposed.

2.1 Predictive Maintenance

Maintenance is an essential task in the industry that increases the life of tools and machines. Its planning varies based on the specific situation and requirements. The existing policies can be classified into four categories [5].

The simplest method is *corrective maintenance* which consists of acting only after a failure. This approach is not optimal because it causes the unavailability of the tool and the stop of the processes that depend on it. However, sometimes the adjustment or the substitution is sufficiently fast and cheap that waiting for the break is an acceptable solution.

The second solution is *condition-based maintenance* where, through constant monitoring of the machines, their health state is computed. The maintenance action is made only when the systems are not in a healthy condition. It is an improvement of the previous method since it avoids total failure.

The third approach is called *scheduled maintenance*. In this case, the maintenance is planned based on historical failure data. The objective is to anticipate failures by doing periodical checks and performing actions if needed. This method is an ulterior improvement in respect to the previous

ones presented but, still, it has negative repercussions because sometimes unnecessary operations are conducted.

The last category is *PdM*, which consists of the use of tools to predict when the equipment will fail and when corrective operations are needed. This solution is better than previous ones since it avoids the possibility of unnecessary actions and the stopping of the process that uses a certain tool. Moreover, it is favourable since, by knowing in advance when an action is required, it is possible to plan the operation to reduce the minimum downtime of the industry.

Though PdM is so effective, it is not widely used in respect to the other three due to its complexity for implementation which makes it not always worth it. Some exceptions are expensive tools or dangerous environments, such as nuclear plants [6]. Emerging technologies such as IoT, Big data, ML and Deep learning are making PdM more accessible and attractive for an increasing number of cases [2].

PdM can be implemented using a wide variety of approaches. Every solution is chosen depending on the context and mainly on the data available. From a high level, three categories are distinguishable: model-based approaches, data-driven approaches or hybrid approaches [3]. The first one uses the knowledge of the system and analytical models to predict the condition of the system. For the data-driven approach, the model of the system is learned from past data with machine learning algorithms or statistical instruments. The final approach is a mix between the two other solutions where both historical data and the physical model are used for predicting the behaviour.

2.2 Overview of Industrial Robotics

Robots are machines made to automatically execute physical tasks and make decisions that would otherwise be done by humans. They are composed of articulated mechanical systems, a sensor system to acquire data about the internal and external status of the robot, a control system that acts as the brain of the robot and an actuation system to move the mechanical components. Two main categories are recognizable: mobile robots and robot manipulators.

Mobile robots are characterized by a mobile base that permits them to move in the environment. The locomotion can use a system of wheels or multiple rigid bodies to emulate two or more legs.

Robotic manipulators or robotic arms are robots, usually with a fixed base, capable of performing functions similar to the ones of a human arm.

They are often used in industry to automate manufacturing processes as welding, cutting, painting and other repetitive actions.

A manipulator is composed of an arm and a wrist that enable the movement of an end-effector toward a position to perform a certain task. The arm and the wrist consist of a sequence of *links* and *joints*. Links are rigid bodies connected by joints. The relative motion of the links is constrained by the joints, which can be of prismatic or revolute type. A prismatic joint enables a relative translation along an axis, and the revolute permit a relative rotation through two links.

The configuration of links and joints forms the kinematic chain of the robot. By changing the length, number, and types of components in the chain is possible to obtain manipulators with different movements capacities. In industrial applications, it is common to have a six Degrees of Freedom (DoF) robot called an "elbow" due to its similarity with the arm of a human. An example of an elbow manipulator is showed in Figure 2.1.

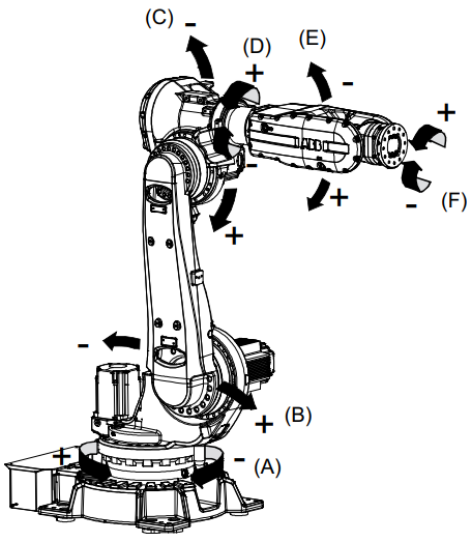


Figure 2.1: Robotic manipulator with six DoF. The image is from the product specification of the IRB 6620 *.

For any robotic manipulator, to achieve a task, some movements have to be performed. In the following section, the components needed to perform the tasks are presented. Eventually, an analysis of the joints friction and the wear phenomena is given.

*https://search.abb.com/library/Download.aspx?DocumentID=3HAC025861-001&Action=Launch

2.2.1 Controlling the robot

Robot tasks consist of one or more ordered locations that the end effector has to reach. For some tasks, like for moving objects, it is sufficient to define two points, the pick-up and release locations. For other tasks, it is required that the end effector passes through a certain number of points. These requirements are the inputs to the motion planner of the robot which extrapolates a trajectory for the end effector to fulfill the task.

To move the end effector along a certain trajectory, an accurate mathematical model of the robot is necessary to define the input for the joint actuators. Two studies are needed: the kinematic analysis describes the motion without considering force and torques, while the dynamic analysis uses the forces in relation to the torques applied to the robot and its motion.

The kinematic model extracts the analytical relationship between the joints and the end effector in terms of location, velocity and accelerations. The relationship can be seen in two ways: the direct kinematic problem aims to describe the end effector motion as a function of the joint motion, the inverse kinematic problem is the opposite process and so it is used to find the joint motion needed to obtain a certain end-effector movement.

The dynamic model is derived from the kinematic one and described by a series of equations of motion. The functions put the forces and the moment applied to the robot in relation to its motion. A model is required for evaluating the torques, called feed-forward torques, which are needed for moving the robot in the objective configuration.

Eventually, a control algorithm is necessary for adjusting the motion of the end effector. With the dynamic and the kinematic analyses it is possible to define the input needed for the joints to reach a certain configuration, but it is not enough. The manipulator is an articulated system where the motion of the links influence each other and the dynamics will never be able to take into consideration all forces and friction that will affect the robot. For this reason, the control system will optimize the joint input to minimize the error position of the end effector at run time. In this project, the profiles produced by the control algorithm will be referred to as torques and will be distinguished from the feed-forward torques produced by the dynamic model.

The control algorithm is especially important for this project since it is the one that indirectly has to deal with the wear of the joints and that causes the shift of the torque that in this work is used for predicting the state of the joints.

2.2.2 Joint friction and wear

The movement of the robot is due to the rotation or translation of the joints. The motion is obtained using electrical motors that convert current into torques and forces actuated on joints. The resulting relative motion between the surfaces and their interaction at a nanoscale level of the mechanical elements of the joint causes the friction force. This phenomenon is studied in the science of tribology together with the lubrification and wear phenomena.

Friction is modelled as a function of speed. The friction curve in Figure 2.2 shows how the friction level changes in relation to the speed. The non-linear behaviour of the friction is known as the Stribeck effect and it is caused by the lubricants used to reduce the wear process. At low speeds, the asperities of the surfaces in contact refrain the movement. Increasing the velocity causes an increment of the lubricant layer and so a decrement of contacts. After a certain velocity, the friction resumes the growing behaviour because of the shearing of the lubricant layer.

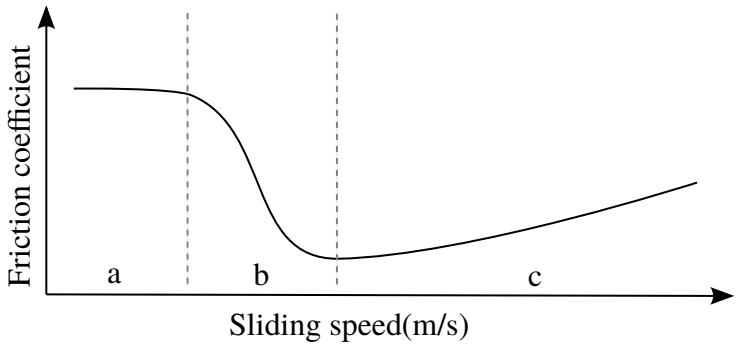


Figure 2.2: The Stribeck curve can be divided in three regions: boundary lubrication (a), mixed lubrication (b) and elasto-hydrodynamic lubrication (c) [7].

Friction models have two main purposes, improving the control algorithm by considering the interference, and implementing fault detectors. Wear is one of the factors that influences the frictions in joints together with the type of material, the temperature, the lubricant, and the load.

Robotic joints are composed of several components interacting with each other. Modelling the friction at the component level is not common due to the complexity. A straightforward approach is to do a “lumped” joint friction model where all the effects of the single parts are considered collectively. A common static friction model is derived from the LuGre model and in [8] is expressed with

$$\tau_f(\dot{\varphi}) = [f_c + f_s e^{-|\frac{\dot{\varphi}}{\dot{\varphi}_s}|^\alpha}] sign(\dot{\varphi}) + f_v \dot{\varphi}, \tag{2.1}$$

where f_c is the Coulomb friction, f_s is the static friction parameter, f_v is a constant to model the friction proportional to the speed, $\dot{\varphi}_s$ is the Stribeck velocity, α is the exponent of the Stribeck non-linearity and $\dot{\varphi}$ is the speed of the joint.

Among the influencing factors of the frictions, for this project, the wear effects are of particular interest. Using the model of the wear friction profile obtained by observations in [9], it is possible to expand Equation 2.1 and obtain the more general form

$$\tau_f(\dot{\varphi}, \omega) = [f_c + f_{s,\omega} \omega \xi e^{-|\frac{\dot{\varphi}}{(\dot{\varphi}_{s,\omega} \omega \xi}|^\alpha}] sign(\dot{\varphi}) + f_{v,\omega} \omega \xi \dot{\varphi}, \tag{2.2}$$

where ξ indicates the temperature effects on the joint and ω indicates the degree of wear in the joint. The degree of wear is not a physical quantity and so it can assume a value in the interval (0,100) where 0 means that there is no wear and 100 denotes the failure state of the robot. The coefficients $f_{s,\omega}$, $\dot{\varphi}_{s,\omega}$, $f_{v,\omega}$ and α strictly depend on the robot and can be obtained by observations.

The development of wear over time in a robot joint cannot be generalized with a formula. In [10] the wear is approximated with an exponentially growing trend. This choice is motivated by the fact that wear is a result of the ageing of the material, and without external actions, it can only increase. Moreover, the wear generates friction which increases the stress on the robot that accelerate the development of wear, which is why the wear usually has exponential growth.

2.2.3 Health index

The health state of a robot's joints can be inferred from the change of the torque profile of the joint actuator over time. The torque profile created by the motion control of a robot depends mainly on the program the robot executes. So, executions of the same program always end with the same profile. The exception is when some external factor interferes, like wear formation in the joints.

A health index can be derived by the difference between the torque profile used when a joint was in a nominal state, called reference torque, and the current torque profile. The wear concentration increases the friction and so the torque profile will change to overcome the increased friction. This simple solution would work but will be very sensitive to the noise and external

influences of the torque.

In [11] an improved version of the basic algorithm described is proposed. The problem of the noise present in the signal is solved by transforming the data to the distribution domain. The transformation is made using the Kernel density estimator defined with

$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\frac{x - x_i}{h}, \tag{2.3}$$

where K is a kernel function, h represent the bandwidth of the kernel and n is the number of observation x_i of the variable x.

The difference between the signals in the distribution domain is evaluated using the *Kullback-Leibler divergence* defined as

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x)log\frac{p(x)}{q(x)}dx, \tag{2.4}$$

where P and Q are two continuous probability distributions and $p(x)$ and $q(x)$ their probability densities.

The effect of the disturbances is removed from the deviation evaluated by using a filter implemented with an exponential moving average that acts as a low pass filter. Eventually, an accumulated sum of deviations is performed. This last step is implemented to reduce the influence of disturbances. The index is then used to detect the anomalies using a threshold. When the index is higher than the threshold value, the algorithm detects a failure.

2.3 Machine learning

A common approach to extract hidden and complex relationships in data is ML. It is a broad term which refers to several algorithms that have in common the characteristic of improving themselves through processing data. Other algorithms, in contrast, are static and do not gain experience and always have the same behaviour. ML performs very well in problems with high dimensional data and complex and chaotic environments.

The algorithms can be divided into three categories based on how training is performed. In supervised learning, some inputs and outputs are given to the algorithms so they can learn the function that correlates them. In unsupervised learning, the algorithm learns from the data it receives without having any

evaluation from its output, and it is used to find structures and relations. Finally, in reinforcement learning, the algorithm interacts with an environment and learns from the feedback that it receives based on its actions.

Another categorization of ML algorithm is based on the output of the algorithm. A classification is made when the output is a limited set of values, instead, a regression when the output can be any value within a range. The third category is the clustering algorithms where the outputs are groups of similar data. The last approach is used commonly in unsupervised learning to extract hidden structure and relations.

ML algorithm are founding applications in almost every field, PdM is not an exception. Every connotation of algorithms is used from supervised learning to unsupervised, from regression to classification as reported in these literature review papers [5][12].

2.4 Related Work

The potentiality of PdM for industry optimization and sustainability made it an active research field. The possibility to use PdM for almost any tool and environment made it an attractive topic where every research uses different solutions based on the needs and the available data. In the literature, the approaches can be classified based on the instrument used: statistical approaches, artificial intelligence approaches and model-based approaches. The first two categories are similar but one extracts information based on past failures, the second one uses a ML approach to model the system and make predictions. The model-based instead uses the knowledge of the system to create prediction models. Eventually, it is common to mix the approaches to use the strong points of the individual methods.

A common approach is the use of Artificial Neural Networks (ANNs) for fault prediction, as in [13] where accelerometer data and sound vibrations are used to make fault detections. In [14] failure estimation is made using a more statistical approach for scenarios where it is not feasible to collect more data with new sensors. The data are obtained from the Enterprise Resource Planning system that collects all the failure notifications of the machines in the product line. Then an ANN is used for predicting the time to failure based on the historical events. The system is applied to a packaging robot and it has positive results thanks to the high quantity of labelled data.

Another application of ANN is proposed in [15] where it is implemented to predict the end of life of rotational equipment. The algorithm is used to identify the degradation percentage of the machine on a “scale” that goes from

defect-free to a maximum degradation which corresponds to a failure. The data used for training and testing are collected by recording the vibration of the machinery until a failure is reached. Gathering data through experiments makes it possible to have labelled data and to use a supervised learning algorithm. Additionally, the paper has proposed a feature that goes beyond PdM. By modelling a cost matrix that considers the cost for the replacement of the material, for the stop of the production line, and labour cost, they can propose an optimal replacement strategy.

The task of fault diagnosis is sometimes transformed into a classification problem as in [16]. Different from the papers discussed previously, Bayesian networks were used to manage uncertainty in the prediction and to represent the relations between multiple pieces of equipment. The training of the algorithm is made by learning sequentially the probabilities using a fractional updating table that represents the distribution of every random variable present in the model.

Deep learning was also studied for PdM. In [17] the remaining useful life of a turbofan engine is evaluated using Deep Convolution Neural Networks (DCNN). The data were pre-processed with a time window approach to be in a format that the DCNN can elaborate. An advantage of using such an approach is the possibility to use the raw data without pre-processing them since the model performs well in the task of extracting the feature itself. But on the other side is a method that needs a high amount of labelled data for relevant training. Another deep learning model for time series is studied in [18] where the Gated Recurrent Unit (GRU) model is used to approximate the system and the observation model of the tool to forecast the healthy state of the equipment.

One of the algorithms that performs better in PdM is Random Forest (RF). In [5] it is showed that it is applied in very different uses cases. An example of the application of RF is presented in [3] where PdM is performed on a cutting tree machine. The predictive algorithm is implemented on Azure Machine Learning Studio and high accuracy of classification is obtained on predicting the future state machines. For the training, a labelled dataset of more than half a million data points is used.

In [19] survival analysis, extremely randomized trees, K-nearest neighbour and Convolutional Neural Networks are compared in the same task of PdM for predicting the broken status of a robot. Another comparison is made in [20] where it is showed that RF is slower to train but outperforms in accuracy ANN and support vector regression in tools wear prediction. Multiple sensors were used for data acquisition such as a dynamometer, an

acoustic sensor, and an accelerometer.

The problem of PdM was tackled from a different perspective in [21] where the variations of the accuracy reached by the end effector during the usage of the robot are used to predict its health status. The results of the paper showed that there is a causality between the end effector accuracy and the health condition of the joints. The complexity of this approach is that additional hardware is needed to detect very accurately the end-effector position. In fact, the effect of the wear on the end effector location is very small since the control algorithm continuously tries to compensate for the position error.

A more analytical approach in comparison to ML is proposed in [22] where PdM on an industrial robot is performed using a motor current signal. The main idea used is to detect the faulty condition of the bell of the robot by comparing the current signal recorded during a working cycle with the ones recorded in an initial nominal condition of the robot. The comparison was not implemented using raw data but from feature extracting in the pre-processing phase. The information was extracted from the current signal using the wavelet transform which is a variant of the Fourier transform where a different wave base is used. This technique performs better in the analysis of non-stationary signals than the one produced by a robot engine due to the capacity of capturing information about frequency and location.

The idea of extracting a health index of equipment using a difference between data in a working situation and the data in a nominal state is applied also in [23]. In this case, the data used are the torques applied by the engines on robotic manipulators. The data are pre-processed and moved in the distribution domain and then the health index is extracted. By setting a threshold on the health index it is possible to send an alert of when the machine is entering a fault state. This solution can be used for condition-based maintenance but not for PdM since there is no prediction implemented.

A similar problem to the one analyzed in this degree project is the one presented in [24] where the forecasting of the state of health of batteries is carried out. The algorithm used is a Gaussian Process Regressor (GPR) that permits to do regressions of time series data. The end of life of a battery is predicted by forecasting the time series representing the capacity of the battery in the function of the recharge cycles. This method provides also the uncertainty of the prediction. This feature is valuable for PdM since it permits the contextualization of the forecast.

An approach to improve the PdM is proposed in [25] where the forecast of the status of the system is evaluated using a hybrid approach. The

essential concept is to merge a data-driven approach with a physical model-based method. In the paper, the algorithm is developed for a Computerised numerically controlled machine (CNCM) but the key concepts can be adapted to every case where is possible to analytically model the system under analysis.

2.5 Theoretical Framework

This section presents the theory behind the algorithms used and implemented in this degree project. All the algorithms studied are for time series prediction. Time series are sequences of ordered values in the form

$$s = \{s_0, s_1, ..., s_N\}, \tag{2.5}$$

where $s_i \in \mathbb{R}$ for $0 \leq i \leq N$. Knowing a section of the series $\{s_0, s_1, ..., s_{k-1}\}$, a prediction algorithm aims to predict the next values $\{\hat{s}_k, \hat{s}_{k+1}, \hat{s}_{k+2}...\}$.

The first three methods presented in the following sections, i.e. Autoregressive integrated moving average (ARIMA), Exponential smoothing (ETS), and Levenberg-Marquardt algorithm, are general-purpose methods as such they can fit a large variety of time series. They are of the type batch since the parameters do not change during the prediction phase. The next method described is the Exponentron that instead can predict time series with exponential decay and it uses the approach of online learning where the improvement of the algorithm takes place also in the prediction phase. Eventually, GPR is presented. This last solution is a general-purpose algorithm but can be tuned to work with specific data trends.

2.5.1 Autoregressive integrated moving average

One of the most common algorithms used for forecasting a series of data is ARIMA. It is a versatile method that can support a wide range of time series behaviour. The only requirement is that the data are stationary and so the statistical properties remain constant over the sequence. Non-stationary series can be transformed in stationary by differentiating or using other nonlinear transformations such as the logarithmic function. The ARIMA works as a filter that aims to extract the signal by removing the noise and then to use the model of the signal to make forecasts [26].

The model can be seen as a union of three different components from which is formed the word ARIMA. The first element is the Autoregressive

(AR) part. This part aims to capture the impact of past data. It is implemented by performing a regression with the last p values of the sequence. AR is modeled with

$$\hat{s}_k = c + \sum_{i=1}^{p} \phi_i s_{k-i} + \epsilon_k, \tag{2.6}$$

where ϕ_i is a model parameter to weigh the past data points, c is a constant and ϵ is a random error.

The second component is Integration (I). It makes the data stationary and it is realized by differencing the data by degree d.

The last element is the Moving Average (MA) that is used for removing random changes in the data and extract information from the previous error terms. It is performed by making the moving average of the last q forecast errors. The MA model is given by

$$\hat{s}_k = \mu_k + \sum_{j=1}^{q} \Theta_j \epsilon_{k-i} + \epsilon_k, \tag{2.7}$$

where μ is the mean of the series until the instant k, $\epsilon_k = s_k - \hat{s_k}$ is the error of the past prediction and Θ_j is a model parameter that weight the past errors.

Merging Equation 2.6 and 2.7 produces the complete ARIMA model presented in Equation 2.8, where s' is the differenced series. Changing the three parameters p, d and q gives more importance to one component or to another. Different models can be obtained with different configurations that can work on different series. Some common models are reported in the Table 2.1.

$$\hat{s}'_k = c + \sum_{i=1}^{p} \phi_i s'_{k-i} + \sum_{j=1}^{q} \Theta_j \epsilon_{k-i} + \epsilon_k \tag{2.8}$$

Table 2.1: Common ARIMA models.

Configuration(p, d, q)	**Model**
$(0, 0, 0)$	White noise
$(0, 1, 0)$	Random walk
$(p, 0, 0)$	Autoregression
$(0, 0, q)$	Moving average

2.5.2 Exponential smoothing

The ETS is a general-purpose time series prediction. The forecast is obtained as a weighted average of the past data. The key concept is that older data have less importance on the prediction and weights are chosen accordingly. As the name of the method suggests, the weights decrease exponentially for older observations. The method described until now is called simple exponential smoothing. Holt extended it by adding the possibility of forecasting data with a trend [27].

The extended method is composed by

$$\hat{s}_{k+h} = l_{k-1} + hb_{k-1}, \tag{2.9}$$

$$l_k = \alpha s_k + (1-\alpha)(l_{k-1} + b_{k-1}), \tag{2.10}$$

$$b_k = \beta(l_k - l_{k-1}) + (1-\beta)b_{k-1}, \tag{2.11}$$

where l_k is the estimate of the level of the series, b_k is the estimate of the trend of the series, h is the size of the step ahead for the prediction, α and β are the smoothing parameters. Equation 2.9 does the forecast, Equation 2.10 extracts the level and Equation 2.11 models the trend part of the sequence[28].

2.5.3 Levenberg–Marquardt algorithm

Extracting a function that describes a set of data with an exponential trend is called exponential regression. The process can be traced back to a curve-fitting problem where the objective is to find a set of parameters Θ such that a parameterized mathematical model $f(\Theta)$ better fit a set of data points. For the exponential regression $\Theta = \{a, b, c, t_0\}$ and the function is

$$f(\Theta) = a + be^{c(x-t0)}. \tag{2.12}$$

A generally used method for solving the curve-fitting problem is the Levenberg–Marquardt Algorithm (LMA). It permits to solve non-linear least square problems [29]. Given m points (x_i, y_i), the aim is to find the parameters Θ of the model $f(\Theta)$ by minimizing the sum of the squares of the errors [30][31]:

$$\Theta = \underset{\Theta}{argmin} \sum_{i=1}^{m} \left[y_i - f(x_i, \Theta) \right]^2 . \tag{2.13}$$

LMA is an iterative procedure that combines the gradient descent method with the Gauss-Newton method. The first algorithm optimizes the parameter by updating them in the steepest-descent direction by using the gradient of the objective function, the second, instead, assume that the model is locally quadratic in the function of the parameters and so it searches the minimum of the quadratic form. When the solution is farther from the correct one, the LMA acts more as a gradient-descent and when it is closer it acts more as a Gaussian-Newton method.

LMA requires as input, other than the points to fit, an initial set of parameters Θ_0. The algorithm starts from the initial guess and produces a series of guesses until a maximum number of iterations is reached or the relative change at each step is below a certain threshold or the error is small enough. As a consequence, the choice of an initial guess determines the velocity of convergence and the quality of the final result.

2.5.4 Exponentron

The Exponentron is an online ML algorithm introduced in [32]. It is designed for making predictions of time series with an exponential decay behaviour. It is derived from the regression of an exponential function and it is adapted for time series forecasting. The algorithm performs online learning and so it improves itself when it is fed with new data. This feature permits to have a dynamic algorithm capable of adapting to the input. In contrast, in non-online learning, the model is static since the training is performed only initially.

The algorithm aims to approximate a generic time series with an exponential equation in the form

$$\hat{s_t}(\Theta) = a + be^{-c(t-t_0)}, \tag{2.14}$$

where $\Theta = (a, b, c) \in \mathbb{R}^3$ is the set of parameters that the algorithm aims to estimate during the learning, $t_0 \in \mathbb{R}$ instead is a time offset parameter that is calculated with the following mathematical statement derived from Equation 2.14 by imposing $\hat{s_t} = s_0$:

$$t_0 = log((s_0 - a)/b)/c. \tag{2.15}$$

The set Θ need to be initialized arbitrarily but respecting the constraints $a \leq s_0, b \geq 0$ and $c \geq 0$. After the initialization, at any k-th step, the algorithm makes the prediction $\hat{s_k}$ using the Equation 2.14 with the set of parameters Θ_{k-1}. Then the error is evaluated:

$$\epsilon_k = (\hat{s_k} - s_k)^2. \tag{2.16}$$

Eventually, the parameters Θ are updated to minimize the error with the following equations:

$$a_k = min\left\{s0, a_{k-1} - 2\eta_k(\hat{s}_k - s_k)\right\}, \tag{2.17}$$

$$b_k = max\left\{0, b_{k-1} - 2\eta_k(\hat{s}_k - s_k)e^{-c_{k-1}(k-t_0)}\right\}, \tag{2.18}$$

$$c_k = min\left\{0, c_{k-1} - 2\eta_k(\hat{s}_k - s_k)b_{k-1}(k-t_0)e^{-c_{k-1}(k-t0)}\right\}, \tag{2.19}$$

where η is the learning rate. The update equations are derived from the gradient of the loss error function.

The advantage of this method is that it permits incremental learning from a stream of data. The disadvantage is that it can model only exponential trends in contrast with ARIMA and ETS. This is not a limitation if the time series available follows that trend.

2.5.5 Gaussian Process Regression

In supervised learning problems, two approaches are possible: the first one consists of selecting a class of function and finding the parameter to fit the data, and the second one consists in considering all the possible functions and give them a probability degree of correction related to how likely each function represents the data. The problem of the first approach is that if the wrong class of algorithms is selected, it will not work properly. The second approach needs to be approximated since it is not possible to deal with infinite sets of functions.

An approximation can be obtained with a generalization of the Gaussian probability distribution called Gaussian Process (GP) that, instead of modelling a random variable or vector, models functions. A complete theoretical presentation is reported in the book [33]. Before defining the process it is important to recall that a function can be described with a vector

where every value indicates the function value $f(x)$ at a particular x.

A GP is a set of random variables where the joint distribution of each subset of variables is a Gaussian distribution. A process $f(x)$ is defined as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{2.20}$$

where $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ are the mean function and the covariance functions defined as

$$m(\mathbf{x}) = \mathbb{E}\left[f(\mathbf{x})\right], \tag{2.21}$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}\left[((f(\mathbf{x}) - m(\mathbf{x})), (f(\mathbf{x}') - m(\mathbf{x}'))\right]. \tag{2.22}$$

For the Gaussian distribution, it is possible to sample gaussian distributed numbers, instead, in GP it is possible to sample functions. To draw samples it is enough to select a set $X = x_1, x_2, ..., x_n$ of input points, evaluate the covariance matrix using Equation 2.22 and then generate a random Gaussian vector from

$$Y \sim \mathcal{N}(\mathbf{0}, K(X, X)). \tag{2.23}$$

The outcome function is a mapping between the input points and the generated vector. In Figure 2.3 some functions are drawn from the prior distribution. The last step to use a GP as a predictor consists of the integration of the knowledge provided by the training data about the posterior distribution.

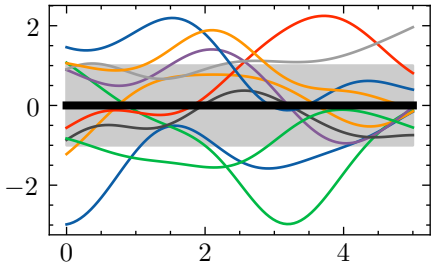


Figure 2.3: Functions drawn from the prior distribution of a GP. The shaded area represent the standard deviation of the processes and the black line is the mean function.

Let's introduce the observation that will be used as a training set

$X_* = (x_i, y_i)|i = 1, ..., n_*$. To model realistic situations, the observations are modelled by adding some noise to the real values $y_i = f(\mathbf{x}) + \epsilon$ where ϵ a Gaussian noise with variance σ_n^2. The prior joint distribution will become

$$\begin{bmatrix} Y_* \\ Y \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X_*, X_*) + \sigma_n^2 I & K(X_*, X) \\ K(X, X_*) & K(X, X) \end{bmatrix}\right). \tag{2.24}$$

Now the prior joint distribution can be restricted to contain only the function that satisfies the observations. This can be made by conditioning the distribution on the observation and so obtaining the posterior distribution:

$$Y|X, X_*, Y_* \sim \mathcal{N}(\bar{y}, cov(y)), \tag{2.25}$$

$$\bar{y} = \frac{K(X_*, X)}{K(X_*, X_* + \sigma_n^2 I)} Y_*, \tag{2.26}$$

$$cov(y) = K(X, X) - \frac{K(X, X_*)}{K(X_*, X_*) + \sigma_n^2 I} K(X_*, X). \tag{2.27}$$

Figure 2.4 shows how the integration of the observations changes the distribution of the functions. The covariance function is the key concept in GP since it identifies the relationship between any pairs of random variables (x, x') and so the properties of the distribution over the function. Kernels permit to model stationary, non-stationary, and periodic behaviors.

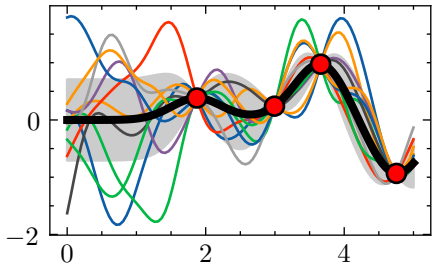


Figure 2.4: Functions drawn from the posterior distribution of a GP. The shaded area represent the standard deviation of the processes and the black line represent the average function.

The optimal values for the hyper-parameters Θ of a kernel are estimated by minimizing the negative log of the marginal likelihood defined as $-\log p(y|x, \Theta)$ using a gradient-based optimization algorithm.

GP was presented in this section for regression problems but, with some

adaptation, it can be used for classification problems.

Kernels

Kernel functions are used to create the covariance matrix. Any function that produces a positive semi-definite matrix is adequate. In this section, some standard functions used in this project are presented. A more detailed analysis of covariance functions is reported in Chapter 4 of [33].

White Noise The white noise kernel is given by

$$k_w(x, x') = \sigma^2 \delta(x, x'), \tag{2.28}$$

where σ^2 is the variance and δ is a Dirac delta function. It models an independent and identically distributed noise on the GP. Since the noise is uncorrelated and identical, the resulting covariance matrix is a matrix with the variance in the diagonal and 0 elsewhere.

Radial Basis Function The Radial Basis Function (RBF) or Squared exponential is given by

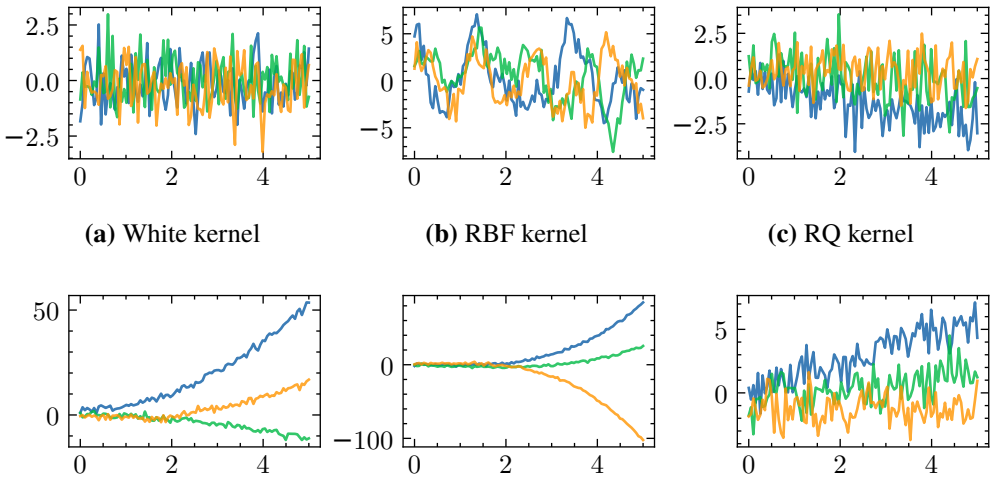
$$k_{RBF}(x, x') = \sigma^2 \exp\left(-0.5 \times \frac{|x - x'|^2}{l^2}\right), \tag{2.29}$$

where σ^2 is the noise variance and l is the lengthscale. The variance influences the change in the vertical scale of the kernel function. The lengthscale influences the horizontal scale and it represents the distance to which the elements influence each other. This kernel models a condition where near points are highly correlated since they have a covariance near 1.

RationalQuadratic The Rational Quadratic (RQ) is given by

$$k_{RQ}(x, x') = \sigma^2 \left(1 \times \frac{|x - x'|^2}{2\alpha l^2}\right)^{-\alpha}, \tag{2.30}$$

where σ^2 and l are the same definitions as before and α is the scale-mixture and it defines the weight to different lengthscales. RQ can be obtained by summing an infinite number of RBF kernel with different lengthscale.



(a) White kernel **(b)** RBF kernel **(c)** RQ kernel

(d) Polynomial kernel of degree 2 **(e)** Polynomial kernel of degree 3 **(f)** Dot kernel

Figure 2.5: Random samples from several covariance functions.

Dot product The dot product kernel is given by

$$k_{dot}(x, x') = \sigma^2 + x \cdot x', \tag{2.31}$$

This kernel, differently from the previous one presented is non-stationary. When σ^2 is equal to zero is called the homogeneous linear kernel.

Polynomial From the dot product kernel can be derived another non-stationary kernel called Polynomial given by

$$k_{pol}(x, x') = \sigma^2(b + x \cdot x')^d, \tag{2.32}$$

where d is a positive integer and it represents the degree of the polynomial and b is the bias.

Figure 2.5 shows some samples from the kernels described.

Combination of kernels The covariance functions can be combined for modelling complex behaviours. The only limitation is that the result matrix is positive semi-definite. Two operations that respect the constraint are the addition and the product. Adding two kernels correspond to do an element-wise addition of two covariance matrices. It can be thought of as an OR operation since the covariance is low only if both inputs have a low covariance

and it is given by

$$k(x, x') = k_1(x, x') + k_2(x, x'). \tag{2.33}$$

The product of two kernels consists of the element-wise multiplication of the corresponding covariance matrices. It can be interpreted as an AND operator because the covariance of two multiplied kernels is high only when they have both high covariance and it is given by

$$k(x, x') = k_1(x, x')k_2(x, x'). \tag{2.34}$$

Chapter 3

Methods

The purpose of this chapter is to present the methods used for answering the research questions presented in Chapter 1. The flowchart in Figure 3.1 shows the overall steps followed. The first essential component to reach the goal is obtaining the data for testing the algorithms. The datasets used and the pre-processing procedures are described in Section 3.1. Then the algorithm implementations and their configurations are discussed in Section 3.2. Finally, the metrics to compare and evaluate the performance of the algorithms are presented in Section 3.3.

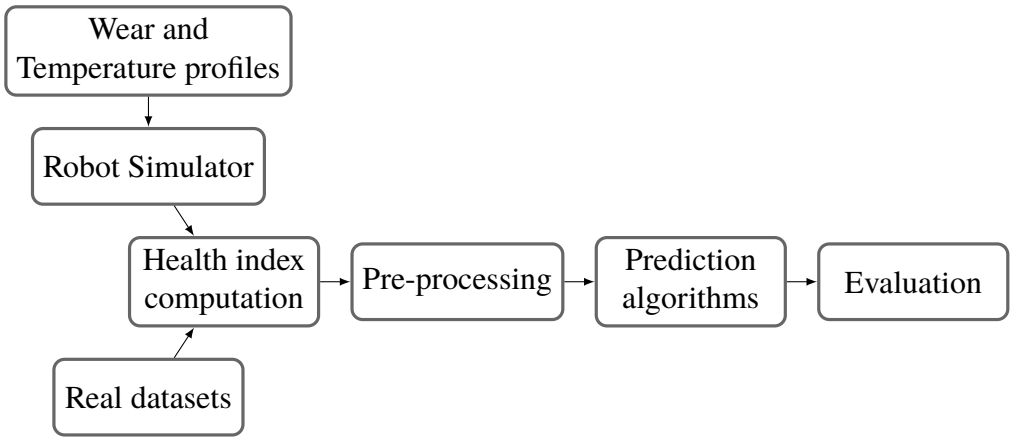


Figure 3.1: The flow of the steps of the methods for answering the research questions. The first two boxes in the upper part are needed for the creation of the simulated datasets. Then, the health index is computed using the torques produced by the robot simulator or the ones available in the real datasets. The health index produced are pre-processed and then used as input for the prediction algorithms. Eventually, the predictions are evaluated using some metrics.

3.1 Datasets

In this research two datasets are used. The main dataset was produced for this work through simulations using an industrial robot simulator with a physics engine. These data are used for testing and comparing the algorithms. The other dataset is made of real data recorded during the usage and testing of real robotic manipulators. These data are used to test the robustness and the capacity for generalization of the algorithms.

The datasets consist of torques, feedforward torques, and velocity recorded for each axis of a robot. The records are made every time the robot executes a routine. For the real data, the routine is a specific test cycle performed for fault detection. This test cycle is not always run within a constant interval, so the data produced have missing values. Moreover, the real data present errors and sometimes a low sampling rate. Because of these problems, the data were interpolated to obtain a constant sampling period of two hours. In the simulated data the cycle can be any program that the robot executes repetitively. In the following sections, the approach used to generate the simulated data is presented.

3.1.1 Data simulation

The data required for the algorithms were obtained with the simulator of the investigated robot brand by running a program on a simulated robot and logging the torques, feedforward torques and velocity for every axis. The advantage of using a simulator to produce data is the possibility of easily changing the physical parameters that can affect the robot execution through the software's physic engine. In particular, in this thesis, three phenomena were modelled: temperature change, load change and wear development.

In every simulation, the robot executes the same program for N cycles. At every run t the friction is determined using Equation 2.2. The following sections describe how the wear and temperature parameters of Equation 2.2 are computed and then how the load change is modelled during the run of the simulation.

Modelling the wear

The wear is introduced in the simulation by acting on the parameter ω in the Equation 2.2. The behavior of ω is modeled with the parameterized exponential function

$$\omega(t) = e^{m(t-2)}, \tag{3.1}$$

where $m \sim \mathcal{U}(5.2, 7.5)$ is a random number uniformly distributed and t represents the cycle. Such function is selected based on the wear formation described in Section 2.2.2. The output of the function is always an exponential growth trend with different growth rates. To increase the variety of wear profiles, two other variables are evaluated randomly:

$$start \sim \mathcal{U}(-0.5, 1.8), \tag{3.2}$$

$$end \sim \mathcal{U}(min(start + 1, 2.5), \tag{3.3}$$

that are used to set the range of the independent variable $t \in [start, end]$ in Equation 3.1. The variable range enables the selection of different parts of the exponential function. For example, when end is small, an almost constant wear behaviour is modelled, and when end is big, an exponential growth behaviour is obtained. Figure 3.2 shows some examples of profiles obtained after the step described.

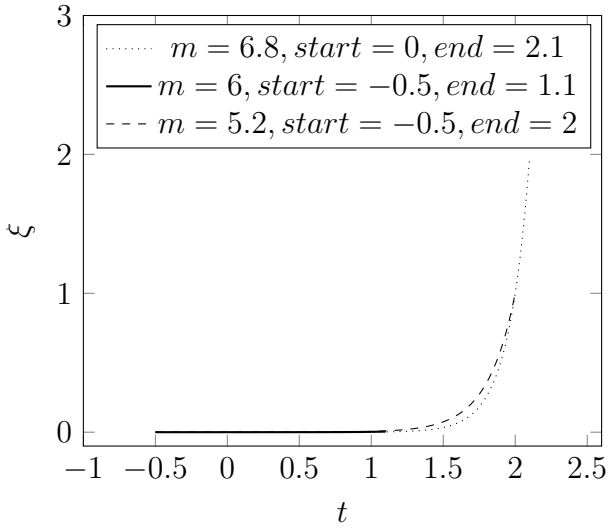


Figure 3.2: Example of wear profiles with different growing trends. Before being used, the profiles in the figure needs to be scaled on the x axis from 0 to the length of the simulation.

The value $\omega(start)$ is subtracted from the profile to make it always start from 0. Then it is normalized from the range $[\omega(-0.5), \omega(2.5)]$ to $[1, maxV]$ where $maxV$ is a value that differs for every axis and it is the amount of wear

that causes the simulator to go on fault state because of too much friction. $maxV$ is evaluated manually by testing increasing numbers until the simulator stops the execution of the robot because the motion control is not able to compensate for the wear.

All the values presented in this section to define the bounds for the uniform distribution are heuristically obtained. The selection is made by testing multiple combinations until the resulting exponential growths were similar to the ones presented in the real datasets.

The results from this phase are pieces of exponential functions with various level of exponential growth and some almost constant trends that represent the axes that are not near a failure. All the parameters presented are evaluated from scratch for all the axes, so every joint has a different wear trend.

Modelling the temperature

Industrial robots usually run continuously in production lines without stops. The constant running causes an almost constant temperature at the joints, so the influence of the temperature in the friction is constant and not affecting the health index. Instead, when the robot is still for a certain amount of time the lubricant becomes colder, the friction increases and the health index is affected until the lubricant warms up again and the friction goes back to the levels before the robot stopped.

The temperature effect ξ is modelled with

$$\xi = \begin{cases} 0 & \text{if } x < c_T, x > c_T + t_w \\ T_{max} - (x - c_T) * \dfrac{T_{max}}{t_w} & \text{if } c_T \leq x \leq c_T + t_w \end{cases}, \tag{3.4}$$

where there is an increase of a random number $T_{max} \sim \mathcal{U}(0, 0.1)$ and a linear decrease to 0 in $t_w = 15$ runs of the robot. The starting point c_T is selected randomly. The parameters selection was made heuristically using tests on the robot simulators. The aim of modelling the temperature is to increase the noise in torque signals, so a more accurate model is out of the scope of the thesis. The temperature effect ξ is applied to all the joints simultaneously in Equation 2.2 because if the robot is not running all the axes cool down. Figure 3.3 shows some examples of temperature trends.

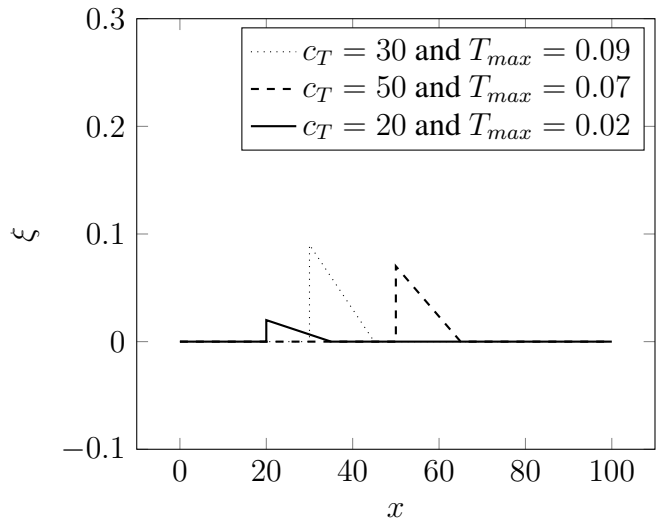


Figure 3.3: Example of some temperature models.

Modelling the load

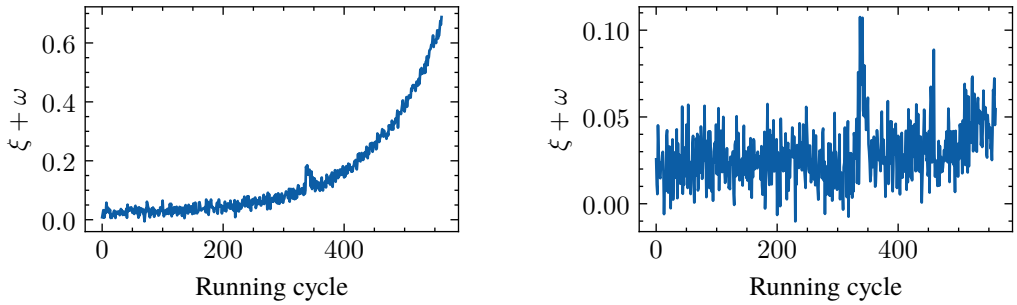
Based on the robot's tasks and the instruments used, the forces in the end effector can change. Moreover, for some activities, the robot has to move masses. When these weights are known in advance, they are included in the dynamic model of the robot as forces applied to the end effector. So the feedforward torques change accordingly to the new weights to fulfil the task. The health index evaluator checks the feedforward torques and when it changes a new ground truth or reference torque for the health index is selected. Thus, in a nominal case, the load change does not cause the detection of an anomaly.

There are situations where the forces at the end effector differ from the ones defined. This situation can happen in case the instrument is modified without update the parameters of the robot. When it happens, the torques change without that the ground truth of the health index is updated because the feedforward torque remains the same, so the health index wrongly increase. The friction, due to the change of load and not for the wear, should be managed by the health index evaluator since it can be detected by noticing that there is a sudden increase of the index on all axes.

If the change in load is small the health indicator can fail to detect it. For this reason, it is modelled in the simulation to have realistic data. The casual change of load is modelled by randomly changing the mass of the payload of the robot.

Profiles obtained

Using the algorithm exposed in the previous sections a wide variety of profiles were created. Two of the profiles are reported in Figure 3.4. The noise and randomization of the parameter guarantee the emulation of different cases to cover as many real situations are possible. The effect of the load on the end effector can not be shown in the figures because the random weight is applied to the end effector and not as a parameter of the friction function.



(a) **(b)**

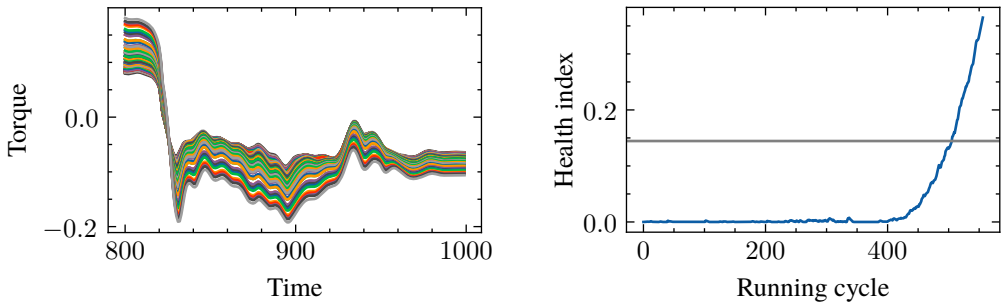
Figure 3.4: Example of two profiles obtained by summing the temperature and the wear profiles. In (a) there is a peak around $x = 380$ due to temperature and then an increase due to wear, instead, in (b) there is no wear but there is the influence of the temperature at the same cycle.

3.1.2 Pre-processing of the data

The dataset described in the previous sections consists of raw data. The first operation is the extraction of the health index of a robotic joint using the algorithm developed in [11] and described in the section 2.2.3. From every simulation, six time series are produced representing the health index of the six axes. In Figure 3.5 the input and the output of the health index algorithm are shown.

To increase the amount of data for testing the algorithm synthetic data are created by sampling the time series multiple times. Given a time series s with length N_s, M time series s^i are created with the form

$$s^i = \{s_j, s_{j+1}, .., s_{N_i}\} \subset s, \tag{3.5}$$



(a) **(b)**

Figure 3.5: Pre-processing of the data. The original data obtained by the simulator are in (a) where there is a section of the torque profiles obtained by running the same program multiple times and changing the wear. The health score evaluated by processing the torques is reported in (b). The grey horizontal line represents the threshold. When the index surpasses the threshold, an anomaly is detected.

where N_i is the length of the sub-sample, j is the start point for the sample and $i = 1, 2, ..., M$. The length and starting point differ from the simulated and real data. For the first type of data $j = 0$ and $N_i = 150, 155, ..., N_s$, instead, for the second, $j = 0, 6, 12, ..., N_s - 800$ and $N_i = 800$. These choices are based on the average length of the two types of datasets. The simulated synthetic datasets will be composed of $M = \lceil (N_s - 150)/5 \rceil$ time series, the real synthetic datasets will be made of $M = \lceil (N_s - 800)/6 \rceil$ time series. The sub-samples of the simulated datasets have different lengths. As in real cases, the anomalies can appear at any instants in time and the data can have different lengths. The new time series created will be referred to as *synthetic data*.

Some of the algorithms implemented do a kind of exponential regression and so to avoid computational problems the time dimension is scaled to a fixed range $[0, 12]$. The range of data in $[0, 10]$ will be used as training, the rest for testing the prediction.

Final datasets

The steps described in the previous sections produced two final datasets that are used as inputs for the implemented algorithms. Tables 3.1 and 3.2 contain the summaries of the final datasets produced using respectively the simulated and real data. In the tables, the *Length* column refers to the number of samples on the original dataset, the *N synt* column indicates the number of time series produced with sub-sampling from the original data, and the column

Anomalies reports the number of anomalies in the original and synthetic datasets. The column *Configurations* of the Table 3.1 reports the configuration of the simulator when the data were produced: $Load$ indicates that a load was applied, $Temp$ that the temperature effect was introduced and $Noisy$ that a strong noise was added to the wear parameter.

Table 3.1: Simulated datasets.

N	Length	N synt.	Anomalies		Configurations
			Original	Synt.	
2	193	24	2	16	
3	299	120	0	0	
4	494	328	1	19	
5	498	318	0	0	
6	200	18	0	0	
7	400	229	1	16	
11	363	188	1	25	Load + Temp
12	390	234	2	43	Load + Temp
13	194	30	3	24	Load + Temp
16	477	334	2	45	Load + Temp
17	565	416	2	62	Load + Temp + Noisy
18	600	458	2	50	Load + Temp + Noisy
19	562	397	1	27	Load + Temp + Noisy
20	591	402	0	0	Load + Temp + Noisy
21	469	311	2	50	Load + Temp + Noisy
22	551	390	1	35	Load + Temp + Noisy
23	554	390	1	42	Load + Temp + Noisy
24	551	372	0	0	Load + Temp + Noisy
25	657	488	1	24	Load + Temp + Noisy
Total	-	6040	30	563	

3.2 Algorithm implementation

This section aims to explain how the algorithms are implemented and which choices are made. The first section presents the baseline algorithms. Then a detailed analysis of the proposed solutions is reported. The difference between the baseline algorithms and the other ones is that the firsts are general-purpose solutions for time series forecasting and the others are optimized for the type of data used in this project. In particular, the time series analyzed

Table 3.2: Real datasets.

N	Length	N synt.	Anomalies	
			Original	Synt.
1	2503	1574	1	41
2	2673	1743	1	34
3	1524	357	1	24
4	2581	1652	1	56
5	1962	1008	0	0
6	7522	5484	1	53
Total	-	37382	20	398

expect to have a long, static, and noisy period, that represents the normal state of the robot, and an occasional exponential increase due to the wear formation close to the breakdown of the robot. The baseline methods are used to evaluate the quality of the other algorithms.

3.2.1 Baseline algorithms

The ARIMA and ETS are standard methods for time series prediction. The implementation used for the first algorithm is the one available in the *Python* library *SciPy* [34] and for ETS the one provided in the library *statsmodel* [35]. The model configuration used for ARIMA is $(0, 2, 1)$ which means that the order of the autoregressive part is zero, the degree of differentiation is two and the order of the moving average is one. For the ETS, the Holt-Winters model is used with an additive trend and no seasonal component.

Moreover, two other variants of ARIMA and ETS are also tested. To the original algorithms, an initial pre-processing step is added. It consists of the evaluation of the logarithmic of the data points. This stage is added to make the data more stationary since the two algorithms work well with stationary data. The output of the algorithms is then transformed back to the original scale by using the exponential function. The algorithms that use the logarithmic function will be called in this report ARIMA-log and ETS-log.

The third algorithm, the exponential regression, is more specific for data containing an exponential growth but is not designed for time series forecasting. It is implemented using the function *curve_fit* of the *Python* library *statsmodel* [35] that use the Levenberg–Marquardt algorithm. The initial guess selected for the function reported in Equation 2.12 is $a = 0$,

$b = 10^{-6}$, $c = 1.597$ and $t_0 = 5$.

3.2.2 Exponentron

The Exponentron (Exp) presented in Chapter 2 is designed for working in time series with an exponential decay behaviour. In this work, the data has an exponential increase trend, so the Equation 2.14 is substitute with the following

$$\hat{s}_t(\Theta) = a + be^{c(t-t_0)}, \tag{3.6}$$

where the negation of the exponent is removed. Moreover, the t_0 is initialized with a fixed value and not evaluated with Equation 2.15. This choice is caused by the fact that the data used begins always with 0 or almost 0 values and so t_0 can be selected as a fixed value.

The parameters a, b and c differ from each other by multiple orders of magnitude. This is a consequence of the trend of the health index. Using the same η in Equations 2.17, 2.18 and 2.19 produce update steps of similar magnitude. So, a similar update step will result in too big or too small steps that cause a wrong convergence. Thus, different learning rates were used for each parameter.

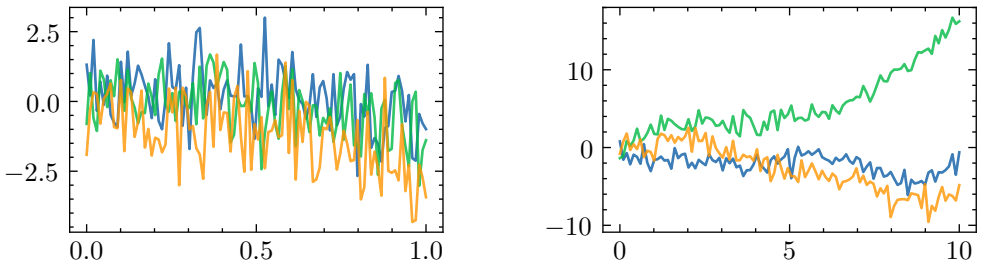
To avoid the non-convergence, the parameters are limited in some ranges. The minimum and maximum values for each parameter are selected based on the knowledge of the data, especially the fact that the health index cannot have a decreasing behaviour and it can not have negative values. The learning rate and the initial values instead are selected with a grid search. The initial values, learning rate and the range for each parameter are reported in Table 3.3.

Table 3.3: Parameters for Exponentron.

Parameter	Initial value	Learning rate	Lower limit	Upper limit
a	0	0.005	-0.02	8
b	13e-7	5e-9	1e-9	0.1
c	1.5978553	5e-6	1	2

3.2.3 GPR

The GPR is implemented using the *Python* library *Pyro*[36]. Using a grid search approach, multiple kernels combination were tested. The two that



(a) k_p **(b)** k_d

Figure 3.6: Random samples from the kernels used.

resulted to better model exponential data are:

$$k_p = k_{pol_4} k_{RBF}, \tag{3.7}$$

$$k_d = k_{RQ} k_{dot} + k_w. \tag{3.8}$$

The hyper-parameters of the kernels can be optimized in two ways, the first approach is by finding them from scratch for every time series, the second approach is by optimizing them for a generic time series and then tune them for the specific prediction. The first approach is highly computational expensive because of the high number of iteration of the optimizer for every algorithm execution. The second approach instead is faster since only during the creation of the model the optimizer needs a long optimization.

The optimization of the hyper-parameters is made using the Adam optimizer. The model is created by optimizing the hyperparameters on a generic exponential trend function using 1400 iterations. Then every time the model is used for a prediction the optimizer is run for 10 iterations to tune the hyperparameters for the specific case.

3.2.4 Hybrid algorithm

In literature, it is demonstrated that combining time series prediction algorithms often leads to improvements in the accuracy obtained with the single solutions [37]. Considering $Y = \{y_1, y_2, ..., y_N\}$ the time series to be forecasted, n models for the time series prediction will produce $\hat{Y}^{(i)} = \left\{\hat{y}_1^{(i)}, \hat{y}_2^{(i)}, ...\hat{y}_N^{(i)}\right\}$ for each i-th model ($i = 1, 2, ..n$). The combination of the n series generate $\hat{Y}^{(c)} = \left\{\hat{y}_1^{(c)}, \hat{y}_2^{(c)}, ...\hat{y}_N^{(c)}\right\}$ and is given by

$$\hat{y}_k^{(c)} = f(\hat{y}_k^{(1)}, \hat{y}_k^{(2)}, ..., \hat{y}_k^{(n)}) \quad \forall k = 1, 2, ..., N, \tag{3.9}$$

where f is some function that takes as input the individual forecast and outputs a single forecast.

In [38] is proven that a simple function f, which usually produce good forecasts, is the average with the same weights for all the predictions. An advanced version consists of using a weighted average:

$$f(\hat{y}_k^{(1)}, \hat{y}_k^{(2)}, ..., \hat{y}_k^{(n)}) = \sum_{i=1}^{n} w^{(i)} \hat{y}_k^{(i)} \quad \forall k = 1, 2, ..., N, \tag{3.10}$$

where $w^{(i)}$ is the assigned weight to the i-th prediction model. Multiple methods are available for the selection of the weights. In *error-based* methods, the weights are determined based on the past forecasted error. In *variance-based* approaches, the weights are obtained by minimization of the variance of the past error.

The method used in this work is derived from [39] which computes the weights from the variances of the errors of the forecasting models:

$$w^{(i)} = \frac{(\sigma^{(i)})^{-2}}{\sum_{j=1}^{n} (\sigma^{(j)})^{-2}}, \tag{3.11}$$

where $\sigma^{(i)2}$ is the variance error of the i-th prediction model. The variances are evaluated on the error of the algorithm obtained in the past predictions. Equation 3.11 gives higher weights to the algorithms which perform better since the better a model is, the smaller the error variance is. In this work, the variance σ^2 is substituted with the standard deviation σ. The substitution was guided by the improved performance using the σ.

The time series analyzed in this work has the characteristic of being constant for long intervals and then increases abruptly. The standard deviation of the errors obtained from the data training is not helpful because all the algorithms predict very accurately the constant part. Instead, the algorithm has very different behaviour on predicting the data when they have exponential growth. Moreover, the accuracy is not constant for every horizon prediction and so multiple standard deviations are used for each model to take into consideration multiple horizon predictions.

The horizon space is discretized into M ranges so that every model has M standard deviations, one for each prediction horizon. The Equation 3.11 becomes

$$w_h^{(i)} = \frac{\sigma_h^{(i)-1}}{\sum_{j=1}^{n} \sigma_h^{(j)-1}}, \tag{3.12}$$

where $h = 1, 2, ..., M$ indicate the horizon. The standard deviation is obtained using the training data and then the weight are used on the testing data.

In Table 3.4 the hybrid models created are reported. Every model differs on the type of algorithms combined. The selection of the combination is based on the test of the single algorithms. For example, *Hybrid 1* is made considering that GPR-dot tends to underestimate the anomalies and the Exponentron tends to overestimate and so to behave opposite to GPR.

Table 3.4: Hybrid models studied. The composition column indicates the type of algorithms merged.

Model	**Composition**
Hybrid 1	Exp, GPR-dot
Hybrid 2	Exp, ARIMA
Hybrid 3	Exp, ETS-log
Hybrid 4	Exp, GPR-dot, GPR-pol
Hybrid 5	Exp, GPR-dot, ARIMA
Hybrid 6	Exp, GPR-dot, ETS-log
Hybrid 7	Exp, GPR-dot, GPR-pol, ARIMA, ETS-log

3.3 Evaluation metrics

The evaluation of the algorithms considers two aspects: the accuracy in estimating the health index and the accuracy in predicting the anomalies. The first evaluation procedures show how much an algorithm can predict the health index in general. The second one focus on anomaly forecasting. There could be cases where the algorithm estimates well the data but does not predict well the anomaly or vice-versa. The following sections present the two metrics.

3.3.1 Time series prediction accuracy

The accuracy of the time series prediction in the estimation of health index is computed with Root Mean Square Error (RMSE) that is defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}, \tag{3.13}$$

where N is the number of predicted values, $\hat{y}_i$ is the estimated value of the real y_i. This metric will output a single value for each time series.

3.3.2 Anomaly prediction accuracy

An anomaly is detected when the health index exceeds a certain threshold. It follows that an anomaly is predicted when the health index estimated by the prediction algorithm exceed the same threshold. The time when the real anomaly was verified and the time when the anomaly is predicted will be defined as t_a and $\hat{t}_a$ respectively. The anomaly prediction error is the difference of the two instants:

$$e_t = \hat{t}_a - t_a. \tag{3.14}$$

The prediction algorithms make a prediction for a certain interval called *horizon* that corresponds to the maximum length of the test data. The prediction can be labelled based on when and if the anomaly is predicted and when and if it happens:

- **True Positive (TP)**: The distance d_t is smaller than the prediction horizon of the algorithm $|d_t| < horizon$ and so the anomaly is correctly predicted.
- **False Positive (FP)**: The distance is bigger than the prediction horizon of the algorithm $|d_t| > horizon$ or the anomaly never happen on the real data.
- **True Negative (TN)**: In the prediction horizon there is no real anomaly and the algorithm does not predict an anomaly.
- **False Negative (FN)**: The algorithm does not predict an anomaly but there is one in the real data inside the prediction horizon.

The time series of which the predictions can not be judged are excluded in the evaluation. It happens when the horizon exceeds the length of the testing data. An exception is made when the test data contains an anomaly, in that case, the sub-sample is considered also if the horizon exceeds the length of the data.

For the TP predictions the accuracy can be computer using the Percent Error (PE):

$$PE = \frac{\hat{t_{ai}} - t_{a_i}}{t_{a_i}} * 100\%. \tag{3.15}$$

This metric is used to discuss the type of predictions and to understand if they are overestimation, so $PE < 0$, or underestimation, $PE > 0$. A more general metrics to summarize the performance of an algorithm is the Mean Absolute Percentage Error (MAPE) defined as

$$MAPE_a = \frac{100}{N_{TP}} \sum_{i=1}^{N_{TP}} \left| \frac{\hat{t_{ai}} - t_{a_i}}{t_{a_i}} \right|, \tag{3.16}$$

where N_{TP} is the number of predictions in the category TP, $\hat{t_{ai}}$ and t_{a_i} refers to the time instant of the anomaly predicted and real of the i-th run of the algorithm.

The four categories described previously are not enough for comparing the performance of the algorithms. They are not taking into consideration the quantity of data and the balance of the data itself. The Key Performance Indicator (KPI) that best fits the requirement of the problem are selected and described shortly in the following paragraphs.

False negative rate

The False Negative Rate (FNR) is defined as

$$FNR = \frac{FN}{FN + TP}. \tag{3.17}$$

It describes the probability of wrongly classifying positive data as negative. In the case of anomaly prediction, it is the rate of missing an anomaly.

False positive rate

The False Positive Rate (FPR) or false positive ratio is obtained using the following equation:

$$FPR = \frac{FP}{FP + TN}. \tag{3.18}$$

It describes the probability of wrongly classifying negative data as positive.

Accuracy

The ratio of the number of correct prediction over the total number of prediction made is called accuracy and is defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{3.19}$$

This metric describes the capacity of the algorithm in making the correct classifications. It does not work well if the data are strongly unbalanced. For example, let consider a situation where the 98% of samples are of class T and 2% of class F. An algorithm that always predicts class T will have an accuracy equal to 98% and so it can be considered a good algorithm. Though the algorithm is wrong the accuracy classifies it as a good one and that is opposite to the expectation.

Recall

The recall is the ratio between the true positive and the number of cases that should be classified as positive. It is defined as:

$$Recall = \frac{TP}{TP + FN}. \tag{3.20}$$

It describes the ability of the model to find all the relevant cases. This metric has the opposite problem of accuracy because by classifying all the samples as True the classifier will be considered correct.

Precision

The precision is the ratio of the correct positive prediction on all the positive predictions:

$$Precision = \frac{TP}{TP + FP}. \tag{3.21}$$

This KPI expresses the ability to classify positive only the real positive samples.

F1 score

The precision and recall can be combined in a single metric called F1 score and defined as

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}. \tag{3.22}$$

This metric does a harmonic means of the previous two KPI giving the same weight to both the two input metrics. This average is one of the possible combinations, if one metric is more important than the other is possible to compute scores using different weights.

3.3.3 Cross validation

The algorithms implemented are tuned based on theoretical knowledge of the expected behaviour of the time series. The algorithms do not need to use data to be optimized and there is no need to split the datasets in training and testing. The hybrid algorithm is the exception since it needs data to evaluate the weights for the combination of the algorithms. For this algorithm, the validation of the solution needs some special considerations.

To analyze the performance, the hybrid algorithm has to run on unseen data. This requirement is satisfied by using the K-fold cross-validation. The data are partitioned randomly in K folds. In particular, $K = 10$ is used because it is demonstrated to be generally a reasonable choice [40]. Every subset is made so it contains the same percentage of anomalies as the other sets. The algorithms are then executed ten times as the number of folds. In every execution, a set is used for testing and the remaining ones for training the algorithm. Using this method, the algorithm is validated always on data

previously unseen by itself.

Eventually, the results obtained from each run on the test data are averaged to measure the performance of the algorithm.

Chapter 4

Results and Analysis

This chapter reports the results obtained. The analyses of the algorithms are made by considering the accuracy of the time series prediction and the ability to predict the anomalies. The results achieved using the simulated datasets are reported in Section 4.1. In Section 4.2 the results reached using the real data are presented.

4.1 Simulated datasets

The data analyzed in this section is obtained by running the algorithms on the synthetic simulated data produced in the pre-processing phase. In Table 4.1 the values of the metrics presented in Section 3.3 are reported for all the algorithms. The first column reports the RMSE, it is obtained by averaging the RMSE evaluated on each time series prediction. Its value differs by orders of magnitude between the different algorithms. These variations are caused by the trends of the data. In fact, the algorithms expect to manage an exponential growth and if they wrongly estimate the beginning of the growth they can simply reach high values and high errors for the nature of exponential functions.

Based on the time series prediction accuracy described by the RMSE metric, the baseline algorithms perform worse compared to the ones optimized for the problem. The hybrid algorithms are strongly influenced by the methods used in the combinations and so the ones using a baseline algorithm have higher error compared to the other hybrids.

The RMSE in the time series prediction describes only part of the performance of the methods. An extremely high value does not imply a bad quality on anomaly prediction. Sometimes the algorithm predicts the anomaly

Table 4.1: Results obtained using the simulated datasets. In the first part of the table there are the baseline algorithms, followed by the optimized and finally the hybrids. The highlighted values represent the best metrics computed and the best algorithms.

Algorithm	RMSE	$MAPE_a$	FPR	FNR	Acc.	Prec.	Recall	F1
ARIMA	4.55e+288	6.43	0.40	49.82	0.97	**0.87**	0.50	0.64
ARIMA-log	1.07e+44	6.01	1.65	46.95	0.96	0.64	0.53	0.58
ETS	5.23e+301	6.45	30.27	19.81	0.70	0.15	0.80	0.25
ETS-log	1.61e+46	6.47	1.43	51.48	0.96	0.65	0.49	0.56
Exp.Reg.	1.24e+89	6.21	3.07	41.97	0.95	0.54	0.58	0.56
Exponentron	**9.11e-02**	6.18	4.40	10.94	0.95	0.57	**0.89**	**0.70**
GPR-dot	**1.95e-03**	6.56	0.91	42.14	0.97	0.78	**0.58**	**0.67**
GPR-pol	**5.33e-03**	6.22	2.68	39.24	0.95	0.57	**0.61**	**0.59**
Hybrid 1	**3.20e-03**	6.27	0.83	25.64	0.98	0.85	0.74	**0.79**
Hybrid 2	9.56e+02	9.73	4.33	8.03	0.95	0.59	**0.92**	0.72
Hybrid 3	4.45e+23	6.10	4.40	8.03	0.95	0.59	**0.92**	0.72
Hybrid 4	**2.41e-03**	6.23	0.81	29.74	0.98	0.84	0.70	0.77
Hybrid 5	**1.80e+01**	10.62	0.85	25.18	0.98	0.84	0.75	**0.79**
Hybrid 6	7.30e+20	6.22	0.87	25.27	0.98	0.84	0.75	**0.79**
Hybrid 7	3.18e+20	11.01	0.85	29.74	0.98	0.83	0.70	0.76

and then it diverges and produces a high RMSE. To analyze that aspect the problem needs to be considered as a classification problem where each time series can be classified in one that will evolve in an anomaly or not.

The second metric reported in Table 4.1 is the $MAPE_a$. It regards the deviation in percentage between the instant when the anomaly is predicted and the instant when it actually occurs. This metric shows that on average all the algorithms predict the anomaly with a similar error around 6% with some exception where the error average is of 10%. However, this index does not take into consideration the amount of anomaly not predicted.

The rest of the metrics show how each algorithm performs in the classification. The FNR and FPR change strongly between the algorithms. The baseline algorithms tend to not predict the anomaly as the high FNRs show. Similarly it happens for the GPR that have a FNR of around 40%. The FPR instead is around 1% for all algorithms except for the Exponential regression and the Exponentron that instead tends to predict more anomalies than the actual ones. The accuracy is optimal for all the models. This is caused by the unbalanced nature of the dataset. In fact, by simply predicting no anomalies the accuracy is high. The precision and recall data are more interesting. ARIMA has higher precision but low recall. It means that it can classify the positive cases well by reducing the false positives but it wrongly

predicts many anomalies, in fact, it has a high FNR. The algorithms with better recall instead are the Hybrid 2 and Hybrid 3 that at the same time have a low precision. Between the non-hybrid algorithms, the Exponentron is very good in detecting the anomalies since it has a recall of 0.89.

Finally, a comparison of the algorithms can be made using the F1 score. From this index, the Exponentron results in the best single algorithm without considering the combinations. Overall, Hybrid 1, Hybrid 5, and Hybrid 6 have the best score, and so they are the best choices for the problem studied.

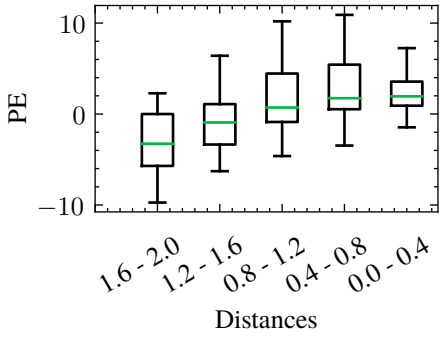
The results discussed summarize the performance of the algorithms but do not provide any insights on single predictions. It is particularly interesting to analyze how the TP and the FN are distributed based on the distance between the end of the training data and the real anomaly. The distance can assume any values inside the interval $(0, 2]$ where a value near 0 means that the anomaly happens at the beginning of the testing interval and a value near 2 means that the anomaly happens at the end of the forecasting horizon, and so at the end of the test interval. To analyze the algorithms, the range has been divided into 5 subranges.

Figure 4.1a and 4.1b report the prediction errors PE for the TP cases for the Exponentron and for the Hybrid 1. The Exponentron has mostly negative PE, which means it tends to predict the anomaly too early. Instead, Hybrid 1 has an error more concentrated near 0 and so it is more accurate. To have a full comprehension of the prediction it is important to take into consideration also the FNR shown in the Figures 4.1c and 4.1d where it is possible to notice that the Hybrid algorithm misses more anomalies than the Exponentron. Both the algorithms have a common trend in which the prediction quality increases when the anomaly is near the end of the training data.

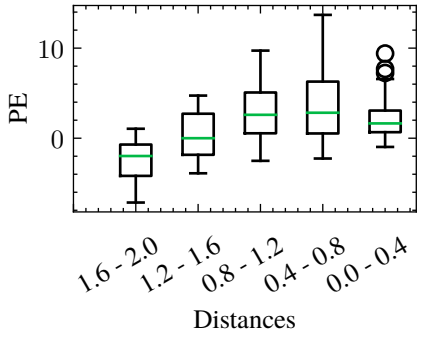
4.2 Real datasets

In this section, the results obtained by running the algorithms on real data are presented. The value of the KPI chosen for each method is reported in Table 4.2. The baseline algorithms perform on average very poorly based on the accuracy in the time series prediction described by the RMSE. Instead, for the other algorithms, the accuracy is a little worse than the one obtained with the simulated datasets. In the table, some algorithms do not have any values for some metrics. That happens because those algorithms have zero TP and so the metrics cannot be computed.

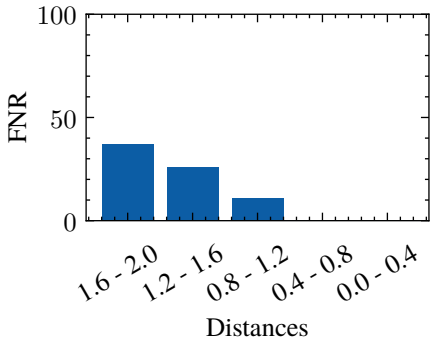
When the anomalies are detected, the MAPE_a is between 5% to 8%, and so, in some cases, it is smaller if compared with the outcome in the simulated



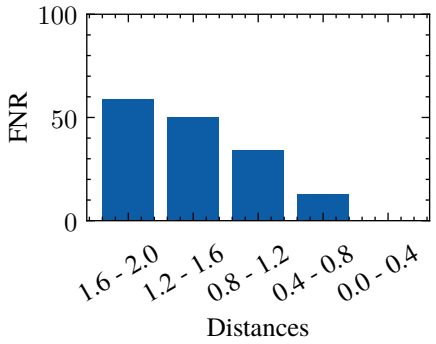
(a) PE distribution for the TP using the Exponentron.



(b) PE distribution for the TP using the Hybrid 1.



(c) FNR using the Exponentron.



(d) FNR using the Hybrid 1.

Figure 4.1: PE distribution and FNR for the two best algorithms on the simulated datasets. The range in the x-axis refer to the distance between the end of the training data and the actual verification of the anomaly. Smaller the distance of the anomaly to the training data and smaller are the errors.

Table 4.2: Results obtained using the real datasets. On the upper part of the table the baseline algorithms are reported, then the optimized ones, and finally the hybrids. The missing values are caused due to the zero TP. The algorithms with better performance are highlighted.

Algorithm	RMSE	$MAPE_a$	FPR	FNR	Acc.	Prec.	Recall	F1
ARIMA	**4.30e+194**	**5.79**	0.25	52.00	0.99	0.75	**0.48**	**0.59**
ARIMA-log	7.45e+64	5.67	1.52	39.05	0.98	0.40	0.61	0.48
ETS	3.20e+250	5.59	6.47	41.90	0.93	0.13	0.58	0.21
ETS-log	7.66e+81	5.60	2.55	41.90	0.97	0.27	0.58	0.37
Exp.Reg.	1.60e+173	8.07	3.10	38.81	0.96	0.22	0.61	0.32
Exponentron	**5.52e+00**	7.14	8.27	22.94	0.91	0.14	**0.77**	0.23
GPR-dot	**5.61e-03**	6.69	0.48	50.00	0.99	0.62	**0.50**	**0.55**
GPR-pol	**3.19e-02**	6.99	5.47	19.33	0.94	0.22	**0.81**	0.34
Hybrid 1	1.33e-02	7.65	1.33	40.00	0.98	0.29	0.60	0.39
Hybrid 2	5.01e-03	-	0.00	100.00	0.99	-	0.00	-
Hybrid 3	4.11e+52	7.02	7.98	22.94	0.92	0.08	0.77	0.14
Hybrid 4	1.08e-02	6.91	2.05	32.38	0.98	0.22	0.68	0.33
Hybrid 5	5.01e-03	-	0.00	100.00	0.99	-	0.00	-
Hybrid 6	1.24e+50	7.53	1.34	40.00	0.98	0.28	0.60	0.39
Hybrid 7	5.01e-03	-	0.00	100.00	0.99	-	0.00	-

datasets. This improvement is also caused by the low number of anomalies detected as the high values of FNR show. The high number of FN is the main issue for the results obtained in the real datasets. A cause is that the wear increases fast and the algorithms tend to not predict the anomaly until it is near to it. Figure 4.2 shows a case where the algorithm does not predict the anomaly and it gives a FN. In this specific case, the wear grows fast and is not predictable at the beginning of the test window.

The FPR is higher than in the simulated datasets because of the noise. Like in the previous section, the accuracy in the classification is near the maximum of 100% thanks to the unbalanced aspect of the time series studied. The precision is low for almost all of the algorithms. The noisiness of the data causes a lot of false positives. The exceptions are the ARIMA and GPR-dot that instead have a high precision. The recall is high for the Exponentron and for some of the hybrid algorithms. From the F1 score, the ARIMA, one of the baseline algorithms, is the method that performs better in classification. Considering both the quality of classifications of the time series predictions, the GPR-dot is the one that offers a high F1 score and a good RMSE. All the hybrid algorithms have a low F1 score and so they perform worse than the original methods in classification.

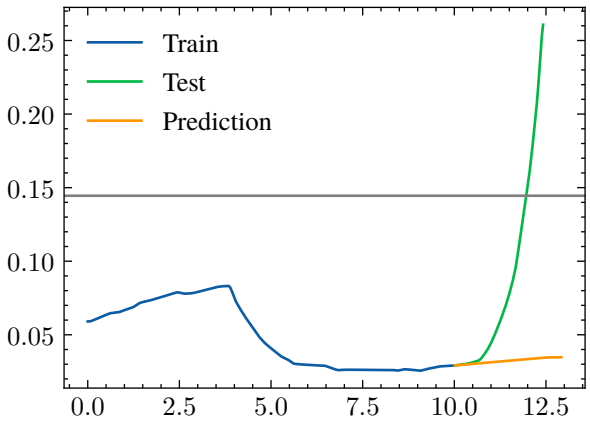
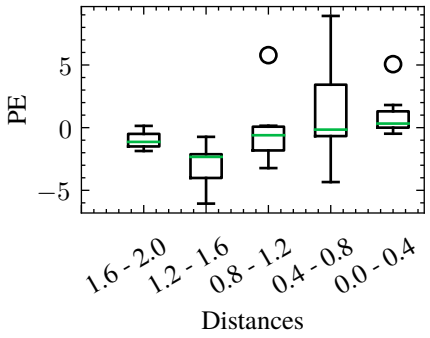


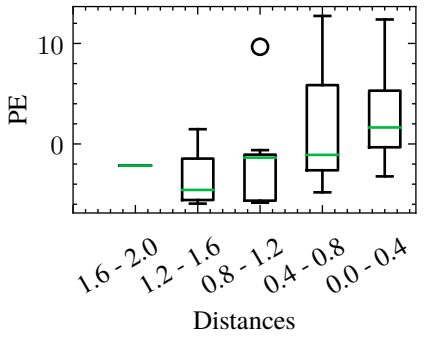
Figure 4.2: Prediction made using ARIMA on a synthetic real time series. The algorithm fails to predict the anomaly since the growth of the health index is not detected in the training data.

The charts in Figure 4.3 show in detail the behaviours of the two best algorithms. The box-plots shows the PE for the TP cases and the bar plot shows the FNR. The distributions of the errors are more concentrated for the ARIMA algorithms where almost all the PE are close to zero. Though the ARIMA has a higher F1 score, it presents a higher FNR in respect to the GPR-dot.

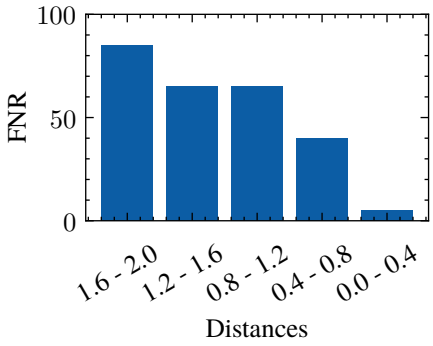
For the real datasets, it is possible to transform the MAPE_a to a meaningful time value since the original data are time labelled. The algorithms use a window of 600 samples for training and 200 for testing that corresponds to a total of $(600+200)*2 = 1600$ hours of data where 2 is the sampling period of the data explained in Section 3.1. Considering a case where the anomaly occurs in the middle of the algorithm horizon ($horizon/2 = 200/2 = 100$), and choosing the MAPE_a of ARIMA, 5.79%, the error in hours is $((600 + 100) * 2) * 5.79\% = 81.06$ hours. It means that an anomaly prediction made with almost 8 days (half the horizon is 200 hours that is around 8 days) in advance has an error of approximately 3 days (81.06 hours).



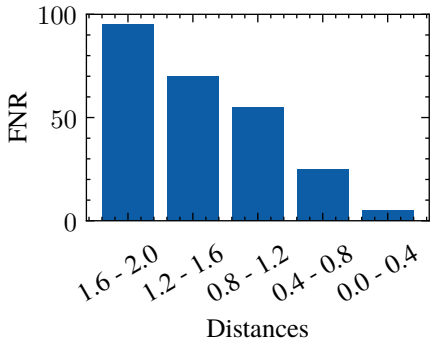
(a) PE distribution for the TP using the ARIMA.



(b) PE distribution for the TP using the GPR-dot.



(c) FNR using the ARIMA.



(d) FNR using the GPR-dot.

Figure 4.3: PE distribution and FNR for the two best algorithms on the real datasets. The range in the x-axis refers to the distance between the end of the training data and the actual verification of the anomaly.

Chapter 5

Discussion

The first outcome from the results is that a time series with an initial constant flat trend and a sudden exponential growth needs special considerations compared to more stationary data. The time series grows from zero to high values in very few samples, so the algorithms need to understand the trend using very few data points. The optimized algorithms expect to receive such type of trend, so they can predict the exponential increase with few points. The metrics show that there is a big difference between the results with the simulated and real data.

Simulated datasets

For the simulated data, Exponentron and GPR outperform the baseline approaches, but the best methods are the combinations of multiple algorithms as the GPR with the Exponentron or the GPR with the Exponentron and the ARIMA or the ETS-log. On the time series prediction, the RMSE differs of multiple orders of magnitude between the traditional methods and the new ones proposed.

About the baseline algorithms, the metrics show that pre-processing the data using the logarithmic function improves the RMSE for both ETS and ARIMA, but it improves the F1 score only for ETS. These improvements are because ETS and ARIMA are methods for stationary data. However, the F1 scores for the baseline algorithms do not differ too much from the ones of the other algorithms. This shows that the baseline algorithms are capable of predicting the anomalies. Nevertheless, the anomaly prediction is evaluated using a specific threshold. If the threshold is changed, the high RMSE for the baseline algorithms would not guarantee that the F1 score remains good. Instead is probable that a change of threshold will cause a decrease in the F1

score. For this reason, the baseline algorithms are not stable enough for the PdM.

The Exponentron and the Exponential regression are applying a similar approach to make the predictions. The first one updates the parameter of the exponential equations with some steps as soon as it receives new data, the second one uses a batch approach to approximate a series of points. This second way is worse than the first one. The motivation is probably due to the nature of the algorithm. Online learning permits a gradual change of the parameters and so noisy points impact the update less. On the other side, the batch update tries to find a function that fit all the points giving all of them the same weight. Hence, the exponential regression converges to bad values or do not converge.

The Exponentron is good at predicting the anomalies as the low FNR proves. On the other side, it has a high FPR because it tends to predict fast growth as soon as it detects a growing trend that sometimes lasts for only a few instants and it is due to noise, temperature change or load change. The GPR has a more cautious behaviour: to predict a growing trend it needs more data points. Eventually, the Exponentron is the best algorithm because the F1 score that merges the precision and the recall is the higher one. Moreover, the advantage of the Exponentron is that it is an algorithm with a computational cost of $O(n)$ since for every new data point only a constant number of operations are made. Instead, the GPR has a computational cost of $O(n^3)$ due to the need for matrix inversion.

The combination of the algorithms increases the computation time since multiple algorithms have to be run instead of a single one. However, in the simulated datasets, it improves the quality of the predictions. In Figure 5.1 is shown as the hybrid algorithm produces a prediction that is better than the original forecasts. One predicts growth too fast and one too slow, the hybrid instead accurately approximates the health index trend.

The highest F1 score obtained is 0.79. The optimal value for this metric is 1, so the outcome is not optimal. The prediction can be probably improved but there is a limit that can not be surpassed. In fact, sometimes the growth of the wear is too fast and the algorithms cannot predict the anomaly until it is very near. In those cases, it is difficult to forecast the anomaly since if the health index of the past data is flat there is no information to predict an increase in wear as shown in Figure 5.2. The box-plot in Figure 4.1b supports this theory by showing that the anomaly prediction improves when the anomalies verify near the end of the training data.

The classifications are evaluated using the F1 score that gives the

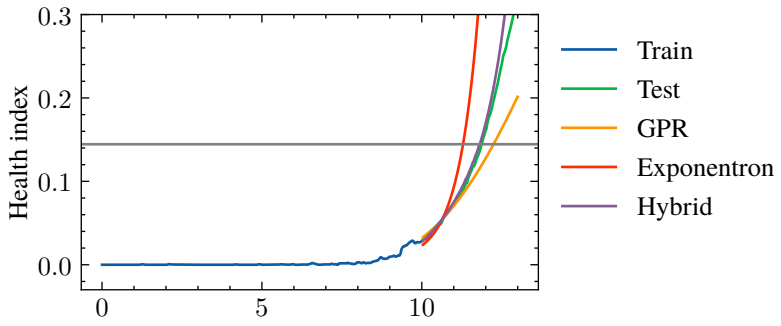


Figure 5.1: One time series from the simulated dataset and the predictions made. The horizontal grey line represents the threshold to identify the anomalies. The Hybrid algorithm perfectly predicts the anomaly, instead, the Exponentron and the GPR are less accurate.

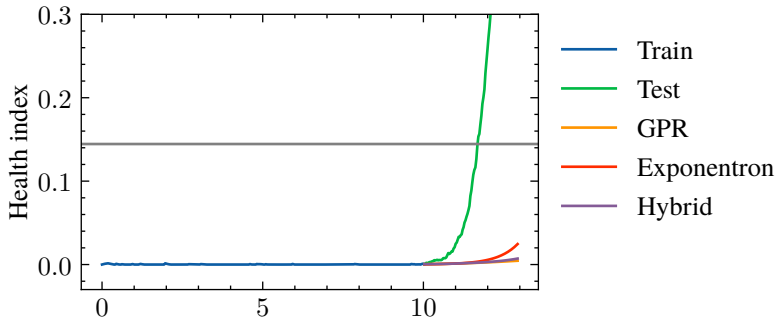


Figure 5.2: Wrong prediction using various algorithms. The exponential growth starts in the test section of the series, so no algorithm can detect it using only the training data.

same importance to precision and recall. It means that FN predictions and FP predictions have the same consequences but it is not always the case. Personalized scores can be defined to weigh the precision and recall in different ways. The weights can be derived from the cost of missing an anomaly prediction and the cost of wrongly predicting an anomaly. In the FN case, the costs are due to the need to change the broken parts of the robots. In the FP case, the costs come from the resources used for the unnecessary maintenance since the robot is in a nominal condition. Different users of the robot have different costs and so the weights should be in function of them. If the cost of FP is higher, the precision should receive more weight, in the other case, the recall should have more importance. Using different scores the best algorithm could change.

Real datasets

Using the real datasets the results are adverse. The algorithms can predict the anomalies but the number of FP and FN are too high to consider the algorithms successful. However, the real data used present a low sampling rate and a high level of noise. From those predictions, it is possible to conclude that the algorithms studied perform better with more data because more information is available. Instead, the real datasets had missing data that cause the issues. An interpolation was made to increase the number of data points but that approximation cannot recover the missing information. By recording the data more frequently and by applying an advanced filter to exclude the temperature effect and the load change from the health index the algorithms would perform as in the simulated datasets. The alternative to the improved motion sampling algorithm is a more complex prediction algorithm that can handle highly noisy data. Such an approach could come to a much higher cost for computation than treating and sampling data more often.

Chapter 6

Conclusions and Future work

This closing chapter aims to presents the conclusion of this thesis work. Section 6.1 summarizes the content of the thesis and draws the final conclusion of the project, highlighting the advantages and the limitations of the solutions. Section 6.2 presents some future possible investigations and issues that can be analyzed to continue the work presented here.

6.1 Conclusion

The purpose of this thesis is to analyze the possibility of implementing PdM for industrial robots using already available data from the machine. The problem comes from the need to optimize maintenance processes by implementing them only when they are necessary. The aim is to implement it in robots already in use, on which adding new sensors is not feasible. Moreover, this work aims to study multiple strategies for PdM to find out which one is more appropriate.

The first step of the project was about dataset creation using a simulator of industrial robots. For this phase, the wear, temperature and random load changes were modelled to simulate both the usury of the gearbox of the robot and other phenomena that create disturbances. The three categories of algorithms were implemented for performing the prediction of the health state of the robots: baseline algorithms, optimized algorithms for the expected trend of the data, and hybrid methods that combine multiple algorithms. Eventually, the methods were tested on the available data.

The outcome of the project is that it is possible to perform PdM on industrial robots using the torque profiles as input data. However, the solutions proposed need to be improved to be used in a real scenario since none of

the implemented approaches have both FPR and FNR small. The algorithm that best fulfils the prediction task is the Hybrid 1 (Exponentron + GPR-dot). However, without considering the combination approaches that add complexity, because multiple algorithms have to be executed, the Exponentron appears to be a valid and fast algorithm for the problems studied.

A main limitation of the methods came out from the executions on the real data. The low quality of the prediction demonstrates that the methods studied need data with a high sample rate and a limited amount of noise, otherwise the performance decreases strongly.

6.2 Future work

Due to the breadth of the problem, some aspects were not investigated in this research. A first possible future research would be the study of deep learning approaches for making PdM of industrial robots using the torque data. Such a solution would enable to analyze directly the torques data without extracting the health index that can lose some features present in the original data. For deep learning approaches, more data would be required than the one produced for this project.

Another possible future work could be a module that does the post-processing of the results given by the algorithms proposed. Some of the FP obtained are due to the temperature effect on the friction and the load effect. These two phenomena influence all the axes of the robot and so they are detectable by analyzing all the axes together. An algorithm could do a fusion of the predictions. If it noticed that multiple axes have a strong growth of the health index, it could recognize that it is not due to wear and adjust the prediction accordingly.

The algorithms were analyzed and tested using a batch approach, so in a real case scenario, the algorithms need to be trained on the past data every time a prediction is required. This method works in simple cases but it is not scalable when the PdM is performed for thousands of robots. A study should be made to find an online way of implementing the algorithms. With some changes, both Exponentron and GPR can be used in online cases.

An interesting research aspect is the study of a method for choosing the best algorithm based on the requirements of the user. Since no method is perfect it would be useful to have an algorithm that based on the cost caused by having false positive and false negative chooses the algorithm or the combination of algorithms that maximize the return for the user.

References

[1] A. C. Pereira and F. Romero, "A review of the meanings and the implications of the Industry 4.0 concept," *Procedia Manufacturing*, vol. 13, pp. 1206–1214, Jan. 2017. doi: 10.1016/j.promfg.2017.09.032

[2] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng, "A Survey of Predictive Maintenance: Systems, Purposes and Approaches," *arXiv:1912.07383 [cs, eess]*, Dec. 2019.

[3] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni, and J. Loncarski, "Machine Learning approach for Predictive Maintenance in Industry 4.0," in *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, Jul. 2018. doi: 10.1109/MESA.2018.8449150 pp. 1–6.

[4] S. Sauvé, S. Bernard, and P. Sloan, "Environmental sciences, sustainable development and circular economy: Alternative concepts for trans-disciplinary research," *Environmental Development*, vol. 17, pp. 48–56, Jan. 2016. doi: 10.1016/j.envdev.2015.09.002

[5] Z. M. Çınar, A. Abdussalam Nuhu, Q. Zeeshan, O. Korhan, M. Asmael, and B. Safaei, "Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0," *Sustainability*, vol. 12, no. 19, p. 8211, Jan. 2020. doi: 10.3390/su12198211

[6] H. M. Hashemian, "State-of-the-Art Predictive Maintenance Techniques," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 1, pp. 226–236, Jan. 2011. doi: 10.1109/TIM.2010.2047662

[7] H. Joyner, C. Pernell, and C. Daubert, "Impact of Oil-in-Water Emulsion Composition and Preparation Method on Emulsion Physical Properties and Friction Behaviors," *Tribology Letters*, vol. 56, pp. 143–160, Oct. 2014. doi: 10.1007/s11249-014-0393-1

[8] A. C. Bittencourt, E. Wernholt, S. Sander-Tavallaey, and T. Brogårdh, "An extended friction model to capture load and temperature effects in robot joints," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010. doi: 10.1109/IROS.2010.5650358 pp. 6161–6167.

[9] A. C. Bittencourt and P. Axelsson, "Modeling and Experiment Design for Identification of Wear in a Robot Joint Under Load and Temperature Uncertainties Based on Friction Data," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 5, pp. 1694–1706, Oct. 2014. doi: 10.1109/TMECH.2013.2293001

[10] V. Sathish, S. Ramaswamy, and S. Butail, "A simulation based approach to detect wear in industrial robots," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2015. doi: 10.1109/CoASE.2015.7294325 pp. 1570–1575.

[11] D. Timotijevic, "Anomaly Detection in Time-Series," *LU-CS-EX*, 2020.

[12] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. d. P. Francisco, J. P. Basto, and S. G. S. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106024, Nov. 2019. doi: 10.1016/j.cie.2019.106024

[13] I. Eski, S. Erkaya, S. Savas, and S. Yildirim, "Fault detection on robot manipulators using artificial neural networks," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 115–123, Feb. 2011. doi: 10.1016/j.rcim.2010.06.017

[14] O. KOCA, O. T. Kaymakci, and M. Mercimek, "Advanced Predictive Maintenance with Machine Learning Failure Estimation in Industrial Packaging Robots," in *2020 International Conference on Development and Application Systems (DAS)*, May 2020. doi: 10.1109/DAS49615.2020.9108913 pp. 1–6.

[15] S. Wu, N. Gebraeel, M. A. Lawley, and Y. Yih, "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*, vol. 37, no. 2, pp. 226–236, Mar. 2007. doi: 10.1109/TSMCA.2006.886368

[16] E. Gilabert and A. Arnaiz, "Intelligent automation systems for predictive maintenance: A case study," *Robotics and Computer-Integrated Manufacturing*, vol. 22, no. 5, pp. 543–549, Oct. 2006. doi: 10.1016/j.rcim.2005.12.010

[17] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliability Engineering & System Safety*, vol. 172, pp. 1–11, Apr. 2018. doi: 10.1016/j.ress.2017.11.021

[18] J. Wang, J. Yan, C. Li, R. X. Gao, and R. Zhao, "Deep heterogeneous GRU model for predictive analytics in smart manufacturing: Application to tool wear prediction," *Computers in Industry*, vol. 111, pp. 1–14, Oct. 2019. doi: 10.1016/j.compind.2019.06.001

[19] R. Pinto and T. Cerquitelli, "Robot fault detection and remaining life estimation for predictive maintenance," *Procedia Computer Science*, vol. 151, pp. 709–716, Jan. 2019. doi: 10.1016/j.procs.2019.04.094

[20] D. Wu, C. Jennings, J. Terpenny, R. Gao, and S. Kumara, "A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests," *Journal of Manufacturing Science and Engineering*, vol. 139, Apr. 2017. doi: 10.1115/1.4036350

[21] T. Borgi, A. Hidri, B. Neef, and M. S. Naceur, "Data analytics for predictive maintenance of industrial robots," in *2017 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, Jan. 2017. doi: 10.1109/ASET.2017.7983729 pp. 412–417.

[22] A. Bonci, S. Longhi, G. Nabissi, and F. Verdini, "Predictive Maintenance System using motor current signal analysis for Industrial Robot," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sep. 2019. doi: 10.1109/ETFA.2019.8869067 pp. 1453–1456.

[23] A. C. Bittencourt, K. Saarinen, and S. Sander-Tavallaey, "A Data-driven Method for Monitoring Systems that Operate Repetitively -Applications to Wear Monitoring in an Industrial Robot Joint1," *IFAC Proceedings Volumes*, vol. 45, no. 20, pp. 198–203, Jan. 2012. doi: 10.3182/20120829-3-MX-2028.00044

[24] R. R. Richardson, M. A. Osborne, and D. A. Howey, "Gaussian process regression for forecasting battery state of health," *Journal of Power Sources*, vol. 357, pp. 209–219, Jul. 2017. doi: 10.1016/j.jpowsour.2017.05.004

[25] W. Luo, T. Hu, Y. Ye, C. Zhang, and Y. Wei, "A hybrid predictive maintenance approach for CNC machine tool driven by Digital Twin," *Robotics and Computer-Integrated Manufacturing*, vol. 65, p. 101974, Oct. 2020. doi: 10.1016/j.rcim.2020.101974

[26] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, May 2015. ISBN 978-1-118-67492-5

[27] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, no. 1, pp. 5–10, Jan. 2004. doi: 10.1016/j.ijforecast.2003.09.015

[28] E. S. Gardner, "Exponential smoothing: The state of the art," *Journal of Forecasting*, vol. 4, no. 1, pp. 1–28, 1985. doi: https://doi.org/10.1002/for.3980040103

[29] Å. Björck, "9. Nonlinear Least Squares Problems," in *Numerical Methods for Least Squares Problems*, ser. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, Jan. 1996, pp. 339–358. ISBN 978-0-89871-360-2

[30] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, Jun. 1963. doi: 10.1137/0111030

[31] J. Nocedal and S. J. Wright, Eds., *Nonlinear Least-Squares Problems*, ser. Springer Series in Operations Research and Financial Engineering. New York, NY: Springer, 1999, pp. 250–275. ISBN 978-0-387-22742-9

[32] A. Rosenfeld, M. Cohen, S. Kraus, and J. Keshet, "Online prediction of time series with assumed behavior," *Engineering Applications of Artificial Intelligence*, vol. 88, p. 103358, Feb. 2020. doi: 10.1016/j.engappai.2019.103358

[33] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*, ser. Adaptive computation and machine learning. Cambridge, Mass: MIT Press, 2006. ISBN 978-0-262-18253-9

[34] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for

Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. doi: 10.1038/s41592-019-0686-2

[35] S. Seabold and J. Perktold, "statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.

[36] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep Universal Probabilistic Programming," *Journal of Machine Learning Research*, 2018.

[37] R. T. Clemen, "Combining forecasts: A review and annotated bibliography," *International Journal of Forecasting*, vol. 5, no. 4, pp. 559–583, Jan. 1989. doi: 10.1016/0169-2070(89)90012-5

[38] A. Timmermann, "Chapter 4 Forecast Combinations," in *Handbook of Economic Forecasting*, G. Elliott, C. W. J. Granger, and A. Timmermann, Eds. Elsevier, Jan. 2006, vol. 1, pp. 135–196.

[39] J. M. Bates and C. W. J. Granger, "The Combination of Forecasts," *OR*, vol. 20, no. 4, pp. 451–468, 1969. doi: 10.2307/3008764

[40] R. KOHAVI, "A study of cross-validation and bootstrap for accuracy estimation and model selection," *International joint conference on artificial intelligence, 1995*, pp. 1137–1143, 1995.

TRITA -EECS-EX-2021:653