POLITECNICO DI TORINO

DIGEP – DEPARTMENT OF ENGINEERING MANAGEMENT AND PRODUCTION

Master of Science Degree in Management and Engineering

Application of Machine Learning techniques in manufacturing

Supervisor:
Prof. Giulia Bruno

Candidate:
Riccardo Furno

Academic Year 2022-2023

Abstract

Industry 4.0 is the last phenomenon that has brought the industry to a new level of process optimisation, and one of its key elements is the Machine Learning.
This thesis aims to depict the current Machine Learning methodologies and techniques present in the current industrial system, highlighting challenges and future opportunities. The study is going also to dig and further analyse the fields in which these algorithms are used the most, like the Digital Twin, a new technology that is becoming a standard in the industrial technologies.

Acknowledgements

Words cannot express my gratitude to my professor and chair of my committee for her invaluable patience and feedback. I also could not have undertaken this journey without my Defence Committee, who generously provided knowledge and expertise.

I am also grateful to all colleagues and mates of mine, for the countless hours spent together working and studying.

Lastly, I would be remiss in not mentioning my family, especially my parents. Their belief and sustain in me have been fundamental to reach the end of this journey at the Polytechnic of Turin.

Table of Contents

List of Abbreviations

Abbreviation	*Meaning*	*Page (first)*
AI	Artificial Intelligence	15
AM	Additive Manufacturing	18
ANN	Artificial Neural Network	11
BLR	Bayesian Linear Regression	53
BRNN	Bidirectional Recurrent Neural Network	12
CAS	Context-Aware System	31
CNN	Convolutional Neural Network	11
DJ	Decision Jungle	53
DT	Digital Twin	9
DeT	Decision Tree	13
GP	Gaussian Process Regression	11
IoT	Internet of Things	15
KNN	K-Nearest Neighbour	13
LoR	Logistic Regression	14
LR	Linear Regression	10
deLSTM	Long Short-Term Memory	12
MAE	Mean Absolute Error	43
ML	Machine Learning	5
NB	Naïve Bayes	12
NN	Neural Network	11
PN	Petri Nets	18
RF	Random Forest	10
RMSE	Mean Squared Error	44

RNN	Recurrent Neural Network	12
RUL	Remaining Useful Life	11
SOM	Self-Organising Map	14
SV	Support Vector	22
SVM	Support Vector Machine	10
SVR	Support Vector Regression	9
WAAM	Wire Arc Additive Manufacturing	18

List of Figures

List of Tables

List of Equations

List of Code snippets

1. Introduction

The Machine Learning, or self-teaching-computer, are a type of statistical based approach to the solution of different types of human problems, or human-related problems, their applications are countless and may be implemented for medical diagnose or to online advertising.

In this work, however, the focus will be on the purely industrial applications of the Machine Learning: a typical example of industrial application, is the tool wear detection: through the collection of raw data from the machine the algorithm enables a preset of command that are able to foresee when and whether the device is going to be worn and have to be substituted. Despite the name, the Machine Learning techniques are purely based on statistics concepts, however there are a multitude of different models that can be used for different scopes, each with different commands and parameters, and hence more or less effective depending on the type of the problem.

This thesis is going to review, using articles retrieved on Scopus, the main principles that define a Machine Learning model and their applications in the industrial field, and also how they can relate to other Industry 4.0 technologies like the Digital Twin, The following chapter will discuss the most common models that are usually implemented in the articles reviewed, distinguishing between classification and regression models, analysing also how they can often be implemented in Python. The fourth chapter will discuss and depicts the techniques that are used for the implementation of the Machine Learning, from the data collection to the evaluation of the models.

The last chapter will discuss a practical implementation of a Machine Learning algorithm with a personal elaborated code and a public retrieved dataset, showing and commenting all the methodologies that have been seen in the other chapters and how they can affect the efficiency of the model, including the evaluation of the model.

2. State-of-the-art

2.1. Paper selection

A systematic and methodological search for articles on Scopus is obviously necessary, to understand what the most efficient methods are and what are the most feasible solutions. In this chapter three different queries have been tested and they are going to be analysed, providing a short description for any result that has been found.

2.1.1. First query: machine learning for prediction applications

For this first section the following query has been computed: *TITLE-ABS-KEY (machine AND learning AND prediction AND manufacturing) AND (EXCLUDE (SUBJAREA , "medi") OR[…]),* the query includes also a long list of EXCLUDE(SUBJAREA) commands, it has been useful to, indeed, rule out all the articles that have a topic different from the manufacture, industry or engineering in general, such as medicine, social study etc. The following tables [*Table 1*] are divided in 5 columns: the first for the title of the article (linked to the bibliography), the second reference to the operating area of the article, the third to the topic the article is dealing with, the fourth is about the foreseen methods used in the procedure (the methods in bold have been found to be the most precise solution) and eventually the last column is a brief dataset description, and if the datasets are public, private or simulated.

***Table 1**: first iteration analysis by article*

Title	***Area***	***Topic***	***Methods Used***	***Datasets Used***
A use case to implement machine learning for lifetime prediction of manufacturing tools [1]	Drilling Machines	Flank Wear Prediction	**RF,** ANN, SVM, DeT	Several Process Parameters (*Privates*)
Model predictive control in milling based on support vector machines [2]	Milling Machines	Quality prediction	**SVM**	Several Process Parameters (*Privates*)
Fatigue-life prediction of additively manufactured metals by continuous damage mechanics (CDM)-informed machine learning with sensitive features [3]	AlSi10MG Alloy	Quality prediction	**SVM** and RF	Several Process Parameters (*Privates*)
Machine learning and deep learning based predictive quality in manufacturing: a systematic review [4]	Review	Quality prediction	None	None Used
A dimensionally augmented and physics-informed machine learning for	High-entropy Alloy	Quality prediction	**SVM,** BNN, ANN, RF	Several Process Parameters (*Privates*)

quality prediction of additively manufactured high-entropy alloy [5]				
Machine Learning-Enabled Prediction of 3D-Printed Microneedle Features [6]	Microneedle	Feature Prediction	**CNN,** SVM, Gaussian	Several Process Parameters – Both numerical and images - (*Privates*)
A fatigue life prediction approach for laser-directed energy deposition titanium alloys by using support vector regression based on pore-induced failures [7]	Titanium Alloys	Fatigue-life prediction through continuous damage	**SVR,** ANN, RF, GPR	Several Process Parameters – Both numerical and images - (*Privates*)
Multistage quality control in manufacturing process using blockchain with machine learning technique [8]	Review	Increase productivity with Quality prediction	**ANN,** KNN	Several Process Parameters (Simulations)
Machine learning-enabled prediction of density and defects in additively manufactured Inconel 718 alloy [9]	Inconel 718 Alloy	Defect presence prediction	SVM, CNN**,** Naïve Bayes, LR, **RF**, KNN, Kernel SVM, Gradient Boosting, DeT, ANN	Several Process Parameters (*Privates*)
Monitoring and predicting the surface generation and surface roughness in ultraprecision machining: A critical review [10]	Ultraprecision Machining	Fatigue-life prediction through continuous damage	**Hybrid and ANN,** RNN, CNN	Several Process Parameters (*Privates*)
Data-driven fatigue life prediction in additive manufactured titanium alloy: A damage mechanics-based machine learning framework [11]	Titanium Alloy	Fatigue-life prediction through continuous damage	**RF**	Several Process Parameters *(Simulated)*
A machine learning framework with dataset-knowledgeability pre-assessment and a local decision-boundary crispness score: An industry 4.0-based case study on composite autoclave manufacturing [12]	Polymer composite	Pipeline, Decision Making	**ANN**	Several Process Parameters (*Privates*)
A joint classification-regression method for multi-stage remaining useful life prediction [13]	Review	Lifetime prediction	**KNN and SVR,** LR, LSTM	Several Process Parameters *(Simulated)*
Application of Machine Learning to the Prediction of Surface Roughness in Diamond Machining [14]	Diamond	Shape deformation prediction	**ANN, RF**	Several Process Parameters (*Privates*)
Prediction of Mechanical Properties of Wrought Aluminium Alloys Using Feature Engineering Assisted Machine Learning Approach [15]	Aluminium Alloys	Quality prediction	**SVR-RBF(Hybrid),** SVM, RF	Several Process Parameters (*Privates*)
Quality Prediction of Drilled and Reamed Bores Based on Torque Measurements and the Machine Learning Method of Random Forest [16]	Drilling Machine	Quality prediction	**RF,** ANN, CNN, SVR	Several Process Parameters (*Privates*)
Log-based predictive maintenance in discrete parts manufacturing [17]	Discrete parts Manufacturing	Maintenance prediction	**RF**	Different datasets are used, none of the publisher (*Privates)*
A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests [18]	Milling Machines	Flank Wear Prediction	**RF,** ANN, SVR	Several Process Parameters (*Privates)*
Online Remaining Useful Life Prediction of Milling Cutters Based on Multisource Data and Feature Learning [19]	Milling Machines	Flank Wear Prediction	**MSFLRUL**	Several Process Parameters (*Privates*)
Tool remaining useful life prediction using bidirectional recurrent neural networks (BRNN) [20]	Cutting Tools	Lifetime prediction	**BRNN**	Several Process Parameters (*Privates*)
Machine learning for monitoring and predictive maintenance of cutting tool wear for clean-cut	Clamping Machines	Cutter Wear Prediction	SVM, **ANOVA,** RNN	Several Process Parameters (*Privates*)

machining machines [21]				
Machine learning based fatigue life prediction with effects of additive manufacturing process parameters for printed SS 316L [22]	Stainless Steel 316L	Fatigue-life prediction through continuous damage	**RF,** ANN, SVM	Several Process Parameters (*Privates*)
Prediction of geometry deviations in additive manufactured parts: comparison of linear regression with machine learning algorithms [23]	Injection moulding	Quality prediction	**SVR,** DeT, LR	Several Process Parameters (*Privates*)
A tool condition monitoring method based on two-layer angle kernel extreme learning machine and binary differential evolution for milling [24]	Milling machine	Monitoring Method	**TAKELM**	Several Process Parameters (*Privates*)
Prediction of microstructural defects in additive manufacturing from powder bed quality using digital image correlation [25]	Metal Components	Shape deformation prediction	**Naïve Bayes**	Several Process Parameters (*Privates*)
Automated Geometric Shape Deviation Modelling for Additive Manufacturing Systems via Bayesian Neural Networks [26]	Review	Shape deformation prediction	**Bayesian NN**	Several Process Parameters – Both numerical and images - (*Privates*)
Prediction of selective laser melting part quality using hybrid Bayesian network [27]	Melting Machine	Prediction of melting part	**Bayesian NN,** Gaussian Regression	Several Process Parameters (*Privates*)
Prediction for Manufacturing Factors in a Steel Plate Rolling Smart Factory Using Data Clustering-Based Machine Learning [28]	Review	Lifetime prediction	Gaussian Regression**,** RF, GB	None used
A defect-based physics-informed machine learning framework for fatigue finite life prediction in additive manufacturing [29]	AlSi10Mg	Fatigue-life prediction	**PINN**	Several Process Parameters (*Publics*)
Machine Learning in CNC Machining: Best Practices [30]	Milling Machine	Flank wear prediction	**RF**	Several Process Parameters (*Privates*)
Deep multi-task network based on sparse feature learning for tool wear prediction [31]	Milling Machine	Lifetime prediction through tool wear	**DMTL**	Process Parameters collected with experiments (*Privates*)
Meta domain generalization for smart manufacturing: Tool wear prediction with small data [32]	Review	Tool wear prediction	**MDG**	Several Process Parameters (*Publics-provided by NASA*)
Tool Wear Monitoring Using Machine Learning [33]	Review	Tool wear prediction	**SVM,** NN	Several Process Parameters – both acoustic and numeric - (*Privates*)
Automated Domain Adaptation in Tool Condition Monitoring using Generative Adversarial Networks [34]	Review	Tool condition monitoring	**GAN**	Several Process Parameters (*Privates*)
Tool Life Stage Prediction in Micro-milling from Force Signal Analysis Using Machine Learning Methods [35]	Milling Machine	Tool wear prediction	**RF**	Several Process Parameters – both imaging and numeric - (*Publics*)
Correlating tool wear and surface integrity of a CNC turning process using Naïve based classifiers [36]	Turning Machine	Tool wear prediction	**Naïve Based**	Several Process Parameters (*Privates*)
Tool Remaining Useful Life Prediction Method Based on Time-frequency Features Fusion and Long Short-term Memory Network [37]	CNC Machine	Tool wear prediction	LR, Bayes, **SVR,**	Several Process Parameters (*Privates*)

Machine learning based approach for process supervision to predict tool wear during machining [38]	AlSi10Mg	Tool wear prediction	RF**, SVM**	Several Process Parameters (*Publics*)
An intelligent prediction model of the tool wear based on machine learning in turning high strength steel [39]	Steel Turning	Tool wear prediction and quality of the product	**SVR**, ANN	Several Process Parameters (*Privates*)
Logistic classification for tool life modelling in machining [40]	Milling Machine	Tool wear prediction	**LoR**	Several Process Parameters (*Privates*)
Tool Condition Monitoring for High-Performance Machining Systems—A Review [41]	Review	Tool wear prediction	None	None used
Implementation of Machine Learning techniques for prognostic of railway wheel flange wear [42]	Railway Wheel	Flange Wear prediction	**LoR,** ANN	Several Process Parameters (*Privates*)
Machine Learning Framework for Predictive Maintenance in Milling [43]	Milling Machine	Predictive Maintenance	LR, DF, BLR, **BDT, NN**, LoR, DJ, BDT	Several Process Parameters (*Privates*)
In-process Tool Wear Prediction System Based on Machine Learning Techniques and Force Analysis [44]	Machining	Flank Wear Prediction	**CNN**	Several Process Parameters (*Privates*)
Tool Wear and Tool Life Estimation Based on Linear Regression Learning [45]	Cutting Tool	Tool wear prediction	**LR**	Several Process Parameters (*Privates*)

Most of the analysed articles have used Python as implementation tool, few others Matlab. However, in several documents, there is no reference to the software used. Almost all the articles have used only numerical parameters (in the column the typology is specified only where there were more than one). Unfortunately, almost all the articles have private data or at least the publisher did not provide information about the availability.

Eventually, some graphs from the SCOPUS Analysis have been collected and shown underneath [*Figure 1*] this section to furnish some additional particulars regarding the articles presented formerly. One of the most important interesting facts is, indeed, the increasing number of articles on this topic, especially in the last 8 years.

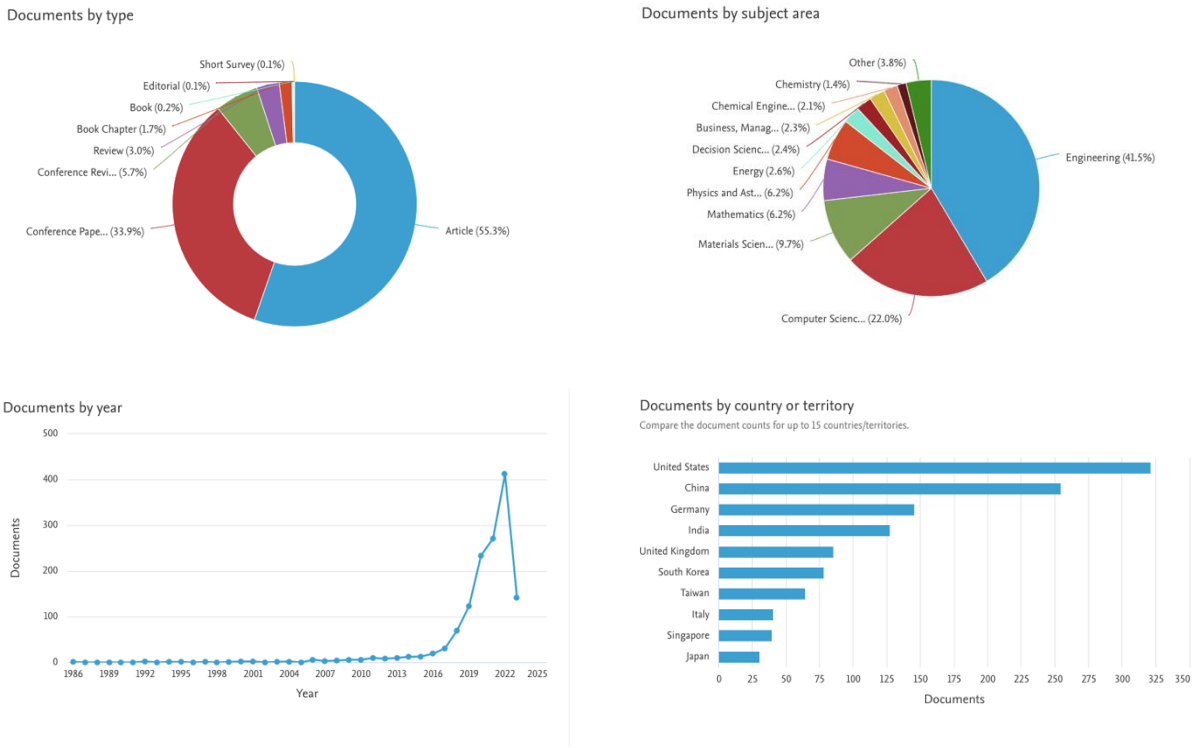

Figure 1: *diagrams and charts of the result of the first interaction;*

2.1.2. Second query: machine learning and digital twin

For the second iteration, a slightly different query has been chosen: *TITLE-ABS-KEY (machine AND learning AND manufacturing AND digital AND twin)* and, like the first iteration, the off-topic subjects have been excluded. The table [*Table 2*] presented beneath, differently from the previous one [*Table 1*] differentiates the Machine Learning (ML) topics with the ones of Digital Twins, since the two arguments are related but they are complementary and almost never they have the same purpose.

Table 2: *second iteration analysis by article*

Title	***Area***	***ML Topic***	***DT Topic***	***Methods Used***	***Datasets Used***
Hybrid learning-based digital twin for manufacturing process: Modelling framework and implementation [46]	Metal Cutting Machines	Quality prediction	Increasing the knowledge of the features	**ANN**	Several Process Parameters (*Privates*)
Toward a smart wire arc additive manufacturing system: A review on current developments and a framework of digital twin [47]	Wire-Arc-Additive-Manufacturing (WAAM)	Quality prediction	Real Time Monitoring	ANN, RNN	Several Process Parameters (*Privates*)

A bio-inspired LIDA cognitive-based Digital Twin architecture for unmanned maintenance of machine tools [48]	Drilling machine	Tool wear prediction	Maintenance Self-Evaluation	RF, **SVM,** CNN	Several Process Parameters (*Privates*)
An AR-assisted Deep Reinforcement Learning-based approach towards mutual-cognitive safe human-robot interaction [49]	Safety Measures	Safety Protocols Prediction	Real Time Monitoring	**PPO**	None used
A multi-access edge computing enabled framework for the construction of a knowledge-sharing intelligent machine tool swarm in Industry 4.0 [50]	Swarm Tool	Quality prediction	Real Time Monitoring	Not Specified	Several Process Parameters (*Privates*)
Petri Nets-Based Modeling Solution for Cyber–Physical Product Control Considering Scheduling, Deployment, and Data-Driven Monitoring [51]	CLIA	Not mentioned	Real Time Monitoring	None	Several Process Parameters (*Privates*)
Metaverse and AI Digital Twinning of 42SiCr Steel Alloys [52]	Steel Alloys	Digitalisation	Real Time Monitoring	RF, **DeT,** LR, Gradient Bosting Regression	Several Process Parameters (*Privates*)
Digital twin assisted: Fault diagnosis using deep transfer learning for machining tool condition [53]	Milling and Drilling Machines	Quality prediction/Tool wear prediction	Real Time Monitoring	**SVM**	Several Process Parameters (*Privates*)
Dynamic Scheduling Optimization of Production Workshops Based on Digital Twin [54]	AGV Implant	Production Simulation Analysis	Scheduling, Control and Monitoring	Not Specified	Several Process Parameters (*Privates*)
A digital twin implementation architecture for wire + arc additive manufacturing based on ISO 23247 [55]	WAAM	Quality prediction	Real Time Monitoring	**CNN**	Several Process Parameters (*Privates*)
Machine Learning for Design Optimization of Electromagnetic Devices: Recent Developments and Future Directions [56]	EM Devices	Quality prediction	Possible Future implementation	**ANN,** RF, SVM, DNN**, CNN**	Several Process Parameters (*Privates*)
Mechanistic models for additive manufacturing of metallic components [57]	Additive Manufacturing (Review)	Quality prediction/Tool wear prediction	Increasing the knowledge of the features	ANN, SVM, RF	Several Process Parameters (*Privates*)
Simulation-Optimization of Digital Twin [58]	Beverage Manufacturing Plant	Quality prediction (Not Mentioned)	Increasing the knowledge of the features	None	Several Process Parameters *(Simulation*)
Digital Twin Security Threats and Countermeasures: An Introduction [59]	Review	Not Mentioned	Threats and Countermeasures	None	None used
Knowledge Project – Concept, Methodology and Innovations for Artificial Intelligence in Industry 4.0 [60]	Review	Not Mentioned	Presentation of the Project and Methodology	None	None used
A Framework of Dynamic Data Driven Digital Twin for Complex Engineering Products: The Example of Aircraft Engine Health Management [61]	Aircraft Engine	Tool wear prediction	Predictive Maintenance	RF, RNN, **LSTM**	Several Process Parameters (*Publics-provided by NASA*)
Synthetic datasets for Deep Learning in computer-vision assisted tasks in manufacturing [62]	Robotic Arm	Classification of the images	Real Time Monitoring	**CNN**	Several Process Parameters (*Privates*)
Reinforcement Learning Based Production Control of Semi-automated Manufacturing Systems [63]	Car Engine Components	Distribution of the tasks	Increasing the knowledge of the features	**ANN**	Several Process Parameters (*Privates*)
Incorporating process physics phenomena in formation of digital twins: laser welding case [64]	Laser Welding	Not Mentioned	Real Time Monitoring	None	Several Process Parameters (*Privates*)
Foresighted digital twin for situational agent selection in production control [65]	Manufacturing Implant (Not Specified)	Not Mentioned	Real Time Monitoring	None	Several Process Parameters (*Privates*)
Digital twin improved via visual question answering for vision-language interactive mode in human–machine collaboration [66]	VQA	Recognition of Words, Images and Sounds	Real Time Monitoring	**CNN**	Several Process Parameters – both acoustic and numeric - (*Privates*)
Digital-twin-driven geometric optimization of centrifugal impeller with	Centrifugal Impeller (Milling Machine)	Tool wear prediction	Increasing the knowledge of the features	Not Specified	Several Process Parameters (*Privates*)

free-form blades for five-axis flank milling [67]					
Smart Manufacturing Control with Cloud-embedded Digital Twins [68]	Manufacture Controller	Quality prediction/Tool wear prediction	Real Time Monitoring	**SVM**	Several Process Parameters (*Privates*)
Combining Simulation and Machine Learning as Digital Twin for the Manufacturing of Overmolded Thermoplastic Composites [69]	Thermoplastic Composites	Quality prediction	Real Time Monitoring	**RF,** DeT	Several Process Parameters (*Privates*)
Towards Real-time Process Monitoring and Machine Learning for Manufacturing Composite Structures [70]	CFRP	Tool wear prediction	Predictive Maintenance	Not Specified	Several Process Parameters (*Simulated)*
Physics-based modeling and information-theoretic sensor and settings selection for tool wear detection in precision machining [71]	Milling Machine	Tool wear prediction	Increasing the knowledge of the features	**KNN**	Several Process Parameters – Images, Sonorous and Numerical- (*Privates*)
Digital twin-driven supervised machine learning for the development of artificial intelligence applications in manufacturing [72]	Multiple Machinery	Quality prediction/Tool wear prediction	Increasing the knowledge of the features	**ANN, CNN**	Several Process Parameters – both image and numeric - (*Simulated*)
An effective architecture of digital twin system to support human decision making and AI-driven autonomy [73]	Manufacturing Implant (Not Specified)	Creating the DT	Predictive Maintenance	Not Specified	Several Process Parameters (*Privates*)
Understanding of the Modeling Method in Additive Manufacturing [74]	Review	Not Mentioned	Challenges, Descriptions and Applications	None	None used
The prediction method of tool life on small lot turning process – Development of Digital Twin for production [75]	Turning Machine	Tool wear prediction	Increasing the knowledge of the features	CNN, BP, SVM	Several Process Parameters – both image and numeric - (*Private*)
Context Aware Control Systems: An Engineering Applications Perspective [76]	Review	General Description	Challenges, Descriptions and Applications	KNN, CNN, Bayesian Regression	None used
Digital twin for cutting tool: Modeling, application, and service strategy [77]	Cutting Tool	Tool wear prediction	Real Time Monitoring	ANN and CNN, **SVM**	Several Process Parameters (*Privates*)
Digital Twin: Enabling Technologies, Challenges and Open Research [78]	Review	Not Mentioned	Challenges, Descriptions and Applications	None	None used
An Implementation Approach for an Academic Learning Factory for the Metal Forming Industry with Special Focus on Digital Twins and Finite Element Analysis [79]	Metal Forming	Not Mentioned	Real Time Monitoring	None	Several Process Parameters (*Privates*)
Machine Learning based Digital Twin Framework for Production Optimization in Petrochemical Industry [80]	Petrochemical	Training the DT	Real Time Monitoring	**RF**	Several Process Parameters – both image and numeric - (*Simulated*)
A supervised machine learning approach to data-driven simulation of resilient supplier selection in digital manufacturing [81]	Supplier Selection	Performance prediction	Increasing the knowledge of the features	**SVM,** ANN, Bayesian Network, Naive Bayes, DeT, **KNN,** LR	Several Process Parameters (*Privates*)
Machine Learning based Continuous Knowledge Engineering for Additive Manufacturing [82]	Additive Manufacturing	Quality prediction	Real Time Monitoring	KNN, SVM, NN, **DeT**	Several Process Parameters (*Privates*)
Enhancing Digital Twins through Reinforcement Learning [83]	Metal Manufacturing	Quality Prediction	Safety Policy Controller	**Bayesian Neural Network**	Several Process Parameters (*Privates*)
Electric Motor Production 4.0 – Application Potentials of Industry 4.0 Technologies in the Manufacturing of Electric Motors [84]	Electric Motors (Review)	Challenges, Descriptions and Applications	Challenges, Descriptions and Applications	None	None used

A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications [85]	Review	Not Mentioned	Challenges, Descriptions and Applications	None	None used
Digital Twin for Machining Tool Condition Prediction [86]	Milling Machine	Not Mentioned	Integration of digital and physical knowledge	None	Several Process Parameters (*Privates*)
Lead time prediction in a flow-shop environment with analytical and machine learning approaches [87]	Lead Time	Lead time prediction	Lead time prediction	LR, **RF**, SVR	Several Process Parameters (*Privates*)
Data Construction Method for the Applications of Workshop Digital Twin System [88]	Data Construction	Verification of the Hypothesis	Tool wear prediction	Naïve Bayes, RF	Several Process Parameters (*Public*)
Digital Twin-enabled Collaborative Data Management for Metal Additive Manufacturing Systems [89]	Additive Manufacturing	Decision-Making Support (Quality prediction)	Increasing the knowledge of the features	**CNN**	Several Process Parameters (*Privates*)
Intelligent welding system technologies: State-of-the-art review and perspectives [90]	Welding Machine (Review)	General Description	Real Time Monitoring	DeT, SVM, ANN, CNN, RNN	None used

As in the first interaction, here some graphs extrapolated from SCOPUS are shown [*Figure 2*], to provide a better understanding of where and when this research is pursued the most. It's interesting noticing how ten years ago the number of research documents about was nought and has raised just in the last two years.

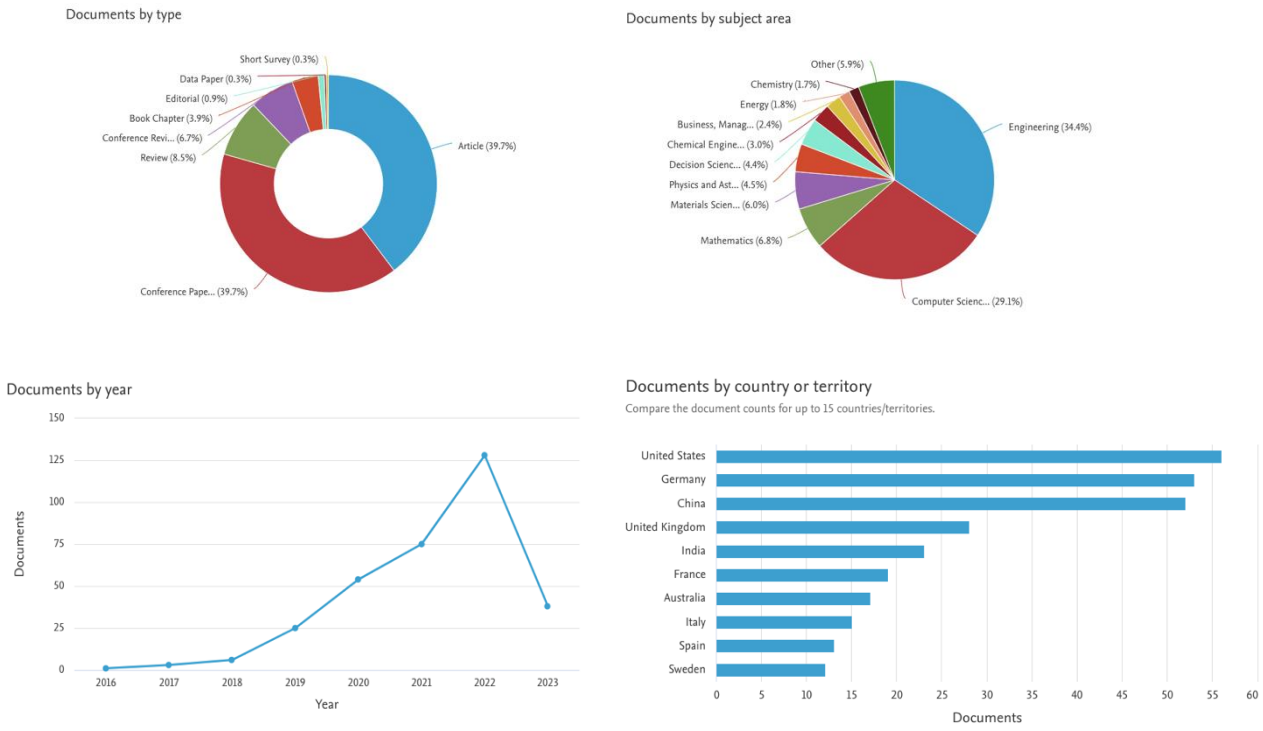

Figure 2: *diagrams and charts of the results of the second iteration;*

2.2. Machine Learning and Industrial applications

This section is going to analyse, the industrial applications for ML algorithms that have been highlighted by the sources of [*Table 1*]. The main industrial applications are essentially: tool wear (25) and quality prediction (12), several others may have been applied for specific works. The tools on which the ML models have been applied are also object of discussion since there are mainly three different devices the sources are focus on: milling machines (10), drillers (4) and alloy (10). Eventually, a subsection is going to briefly describe the not common ML method that have been applied for specific purposes.

2.2.1. Tool wear prediction

One of the main fields of intervention that ML faces, at least in industrial applications, is the wear prediction of the machinery. Drillers, Millers and in general attrition-based tools, equipment that has a flank face operating directly on a workpiece surface causing frictions that lead toward the wear of the machine are all potential subject of application for the tool wear prediction task for a ML algorithm. Typically, the tool wear of an industrial machine is measured mainly on the flank wear (measured as VB) [*Figure 3*]: the attrition that is generated on the flank face of the tool leads to a consistent loss of the latter, compromising therefore the functionality of the machine [*Figure 4*] [30]. In this context, the ML has indeed proven to be efficent and brought a high degree of innovation and performance in the industry. In particular, the ML objective is to detect patterns, according to the features provided and predict a likely value (in this case, tool wear), using statistical and probability models [33]. The tool life, therefore, can be computeted using the Taylor's Formula [*Equation 1*]:

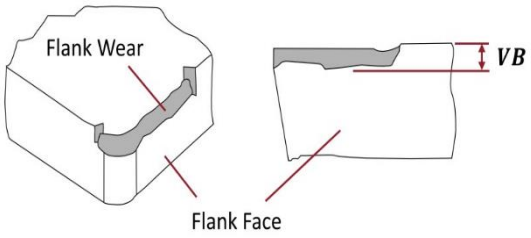

Figure 3: *tool wear that can be observed on the device [30];*

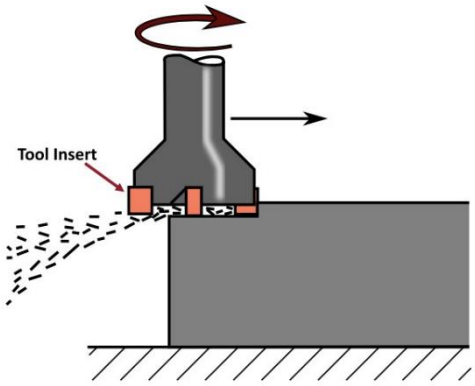

Figure 4: *a milling machine operating along a piece [30];*

$$V_C T^n = C \quad (1)$$

Where, V_C is the cutting speed of the tool, T^n the tool life expected before the flank wear will be too extended and C a constant that has to be computed experimentally. The other values depend on the metal and on the machine that are used. The ML is hence used (at least in this task) to foresee the value of T^n, using the other parameters as regressors. However, the ML may be also applied to classify the tool if worn or still usable according to the value of the regressors. These two "types" of ML might seem similar, but the models used for the implementation are very different, since the former pursue a prediction on continuous values and the latter aim to predict a categorical output, anyway these models and their implications are going to be seen in the details in the following chapter.

An example of implementation of ML is provided by R.Oberlé *et al.* [1] precisely explain the argument of this subsection: the authors, at first, prepared the framework, that is showed aside [*Figure 5*] . They have implemented a Random Forest (RF) method to predict the tool wear in cutting machines, it has to find a relationship between the input (i.e., torque, temperature, speed, etc.) and the output data (i.e., the tool wear) during the so called "*training phase*", verified during the "*testing phase*" and eventually applied on new input data computing, hence, the prediction. As described by the work of D.Wang *et al.*[32], the collection and the management of these data may be problematic, since their amount in order to be statistically significant has to be enormous, hence the authors proposed to implement a meta domain generalisation: the algorithm aims to generalise an ML model in a source domain into a target domain, where there are few or no data, training and testing these datasets.

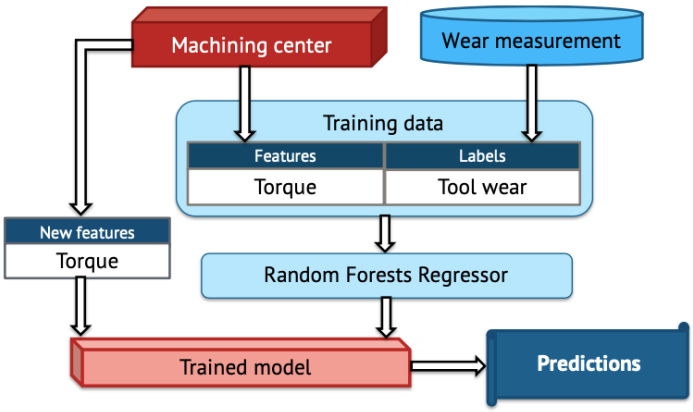

Figure 5: *conceptual model for tool life prediction from [1];*

The decision of the best model to be applied is, obviously, determined by the number of error that it is expected to do, according to different methods that are going to be discussed in the third chapter, like R^2 or the Mean Average Error. Speaking of which, the RF is one of the most used and efficient (statistical speaking) ML models in both regression and classification

problems, speaking about the latter: A. Varghese *et al.* [35], for example, used the RF to classify the tool wear condition in two different stages, according to how the machine was reaching the end of its useful life and how fast the machine was cutting.

Different authors [9], [11], [14], [22], [17] and [42] have indeed implemented an RF model for different machinery (including milling and turning) and for quality prediction (mainly alloy plates) field as well, such as [16], illustrating the best practices and ideas they applied during the optimisation of the model.

However, most of the sources in their work, do not implement directly a precise model, they compare the results of the most likely to be efficient methods and eventually they select the one that has the best evaluation method, like D.Wu *et al.* [18], that analysing a miller, confirm the accuracy of the RF, since it has been capable of outperforming other solutions: like Artificial Neural Network (ANN) and Support Vector Regression (SVR). The authors affirm that they used different sensors to collect the data directly from the machine (e.g., cutting force sensor, vibration sensor, acoustic emission sensor) through 315 milling tests.

Anyway, the RF is not always the best answer, nor it may be applied everywhere: A. Gouarir *et al.* [44] and M.R. Sarabi *et al.* [6] have both applied Convolutional Neural Network (CNN) as a model to predict the tool wear of machining through the direct analysis of image and the prediction embedded in all the Neural Network (NN). Always based on the CNN there is the work of J.Wu *et al.* [13] that implemented a joint classification method for the multi-stage Remaining Useful Life (RUL) like the one of [35].

Speaking of which, J. He *et al.* [31] proposed a method, Deep Multi-Task Learning (DMTL), derived from the NN, with the peculiarity of being capable of computing the prediction of different tasks simultaneously, it was tested on three different cutting machines in different working conditions and the method has been able to outperform the other models, revealing hence its efficiency.

Some other methods, require an optimisation or even to be fused to other components/methods in order to become reliable, like N.S.Karuppusamy *et al.* [45] that have to combine the Linear Regression (LR) and Principal Component Analysis (PCA), to obtain a hybrid method much more effective than the LR alone for predicting the tool wear of a cutting tool edge.

Eventually, it is also possible to combine in one algorithm both the classification and the regression problems: in the work of E.Traini *et al.* [43], the authors studied the implementation of a milling cutting tool, applicable to different machines. The model is thought to predict wear, and to classify the output given by the former to decide when it has to be substituted.

2.2.2. Quality prediction

A similar topic to the tool wear prediction is the quality prediction, as suggest the name, its aim is to predict the quality of the product, providing quality-enhancing insights and hence a decrease in the reject rate of the line [4] or it may be used to study the predicted surface roughness as in the work of [10]. An example of this topic is provided by J. L. Bartlett *et al.* [25], that implemented a Naïve-Bayes (NB) as a classification model, for the distortion along the metal components. The image aside [*Figure 6*] depict the process applied in the article: the system from a picture extracts the topology of the piece and then detects any difference in the height on the surface, the NB, eventually, classify (according to the settings decided), whether the product is defective or regular as in the article of N.K. Mandal *et al.* [36]. Eventually, below is reported an image [*Figure 7*] show three possible defects that may be detected by the process illustrated above.

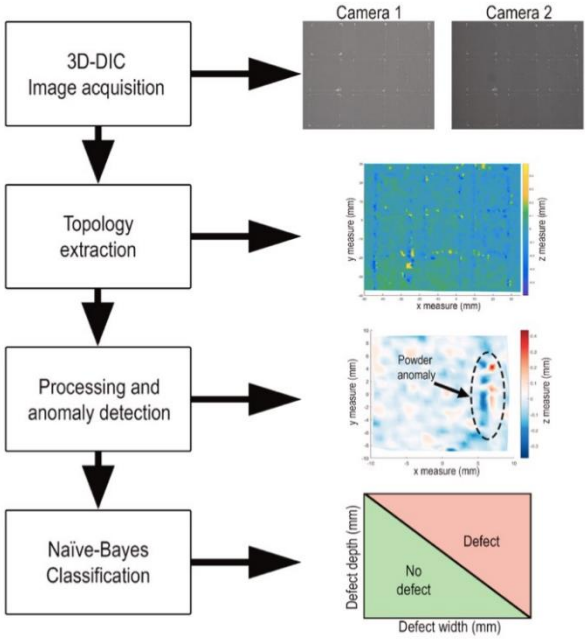

Figure 6: *process scheme of [25];*

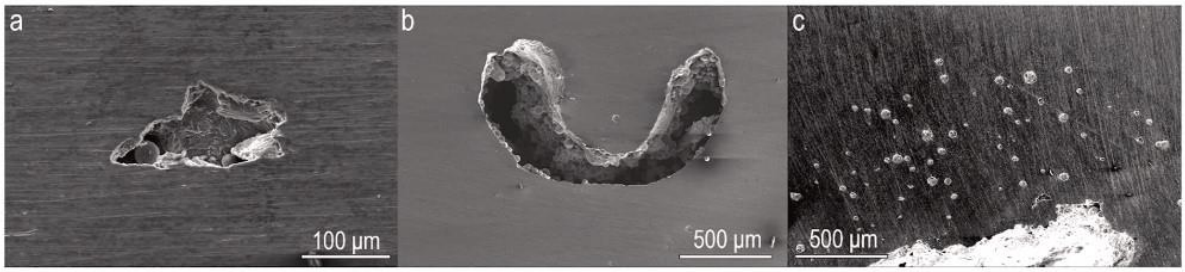

Figure 7: *defects detected in the process of [25];*

The works of [2] and [3] are both focused the Support Vector Machine (SVM), in particular the latter discuss the implementation of the SVM method on Continuous Damage Control (CDM), a technique that through a continuous damage is able to provide information to the ML about the life and the remaining resistance of the material on which is working. Actually, it seems that SVM is rather effective in the task of quality prediction: references [5] and [38]

have indeed implemented a model to predict the quality of the different types of alloys, with an SVM algorithm that systematically outperformed the RF one. The model has been able, in both cases, to predict the deterioration of the samples with good accuracy considering both morphology, size and defect location. The SVM is a classification model, its version for the prediction for continuous values have been implemented by both L. Dang *et al.* [7] and I. Baturynska *et al.* [23], with the Support Vector Regression (SVR) as a classification ML algorithm in Additive Manufacturing to detect the formation of pores on metal surfaces, discussing also about the optimisation problem that is always necessary for any ML algorithm.

2.2.3. Alternative methods for Machine Learning

This section groups the articles that have implemented uncommon or peculiar regression methods that may represents an alternative solution to the other seen above, or a combination of two common methods. Normally, however, these approaches are dedicated to specific case and may be harder to implement then the ones already studied.

The article of T.F. De Barrena *et al.* [20], for example, treats the implementation of the Bidirectional Recurrent Neural Network (BiRNN), a derivation of the Recurrent Neural Network (RNN), that is typically used for algorithms that have to recognise a speech or a handwriting, but in this case has been applied to compute the RUL. Another derivation of the NN has been studied by the sources [26] and [27], the Bayesian Neural Network (BNN) as model for the prediction, the model may be either discrete or continuous, so in order to receive a more precise result, the model has been proposed is a hybrid solution of the two versions of the BNN. Eventually, E. Salvati *et al.* [29] implemented a new neural network: the Physics-Informed Neural Network (PINN), that combines the numerical analysis embedded in the NN models with the underlying physics of the studied phenomenon.

The sources [21] and [37] investigated the effectiveness of a prediction tool method based on time-frequency features implementing the Long Short-Term Memory (LSTM) network, the extraction of the data is performed through the time-frequency domain analysis, and then the LSTM is applied to combine the multidimensional features and predict the RUL.

The combination proposed by M. Cheng *et al.* [39] and M. Hu *et al.* [15] for tool wear prediction, compared the SVM and the combination GS-SVM, that showed a reduction of the MSE of 85%. A hybrid model is thought to be an algorithm able to gain the ability of both the models it is based on, and therefore enhance both their capabilities, in the case of the GS and

SVM have been hybridised since the SVM is necessary to the prediction of the tool wear, and the GS is able to perform a grid search on the product, hence improving the ability of the SVM itself.

B. Lutz *et al.* [34] proposed the Generative Adversarial Network (GAN) as the model for the prediction of Tool-Condition-Monitoring, the peculiarity is that this innovative model is able to analyse and study the images obtained from the tool itself, some activity of labelling is still required, but the algorithm definitely reduces the effort in this operation.

Y. Zhou *et al.* [24] studied for Tool Condition Monitoring another algorithm: Two-Layer Angle Kernel Extreme Learning Machine (TAKELM), a direct variation of KELM, a model considered efficient especially in the case of a small dataset, however, it does underperform with the extraction of inherent features in raw data and hence the TAKELM is supposed to overcome these two drawbacks.

L.Guo *et al.* [19] proposed the model MSFLRUL, that includes both the data acquisition, deletion of outliers and the prediction itself of the RUL model for milling machines and has been found capable of effectively foreseeing the remaining life of the tool.

J. Karandikar *et al.* [40] worked on a model to classify the tool wear of a milling machine through an algorithm of Logistic Regression (LoR), the authors worked on a classification problem since, using shop floor data, the wear can only be measured at the time of tool replacement.

2.2.4. Miscellanea

In this subsection, the utmost articles are shortly analysed and explained. One of the most interesting of these applications is the work of B. Crawforda *et al.* [12] designed and tested a pipeline for decision-making, in an Industrial implant, implemented on the basis of an ANN algorithm. The proposed system had the transparency of the model increased and so was the dataset and provided a score ranking globally and locally.

J. Gu *et al.* [8] studied Blockchain Technology (BCT) to overcome both quality management and data protection issues, due to the high volume of these latter.

In the work of M. Li *et al.* [33], they discussed and tested the efficiency of Self Organising Maps (SOM), useful for identifying the features for tool wear monitoring through the use of competitive learning strategies. The models used for the prediction are the NN and the SVM,

and the former has obtained better results than the latter, but the SOM has demonstrated significant capability into improving the efficiency of the regression method.
Eventually, the references,, [28] and [41] provide useful reviews about the optimal criteria for choosing features, the quality classifications, optimisation challenges, and the most used and efficient models divided by task and process.

2.3. Machine Learning and digital twin

As in the previous section, the articles from the second table [*Table 2]* are going to be explored and studied according to their topic and field of application. Moreover, are also provided several subsections where peculiar sources have been analysed in the details, since this work consider them necessary to give to the reader a sufficient knowledge to continue in the further sections of this chapter. Preliminary, three main fields of applications have been detected: monitoring (19), simulation (3) and predictive maintenance (14) that, in the sources, may have been applied differently and on a different tool, like drillers (3), milling machines (6) and cutting machines (3) or directly the whole implant/production line (9). Notable, is also the numerous applications of DT on additive manufacturing systems (5) and other implementations that show how elastic the DTs can be.
Between these articles, there are also some reviews that debate on the DT, not on certain applications but on the implications that come with the implementation of these technologies.

2.3.1. Digital twin definition

In this section is going to be described the DT, an innovative technology that is spreading through all the industrial fields. It usually has been described as "the forefront technology of Industry 4.0" [78], for what concerns data analysis and the connectivity on the Internet of things (IoT). The DT is, indeed, expected to be one of the most important innovations brought by the Industry 4.0 revolution, but what is exactly a DT?
According to the NASA: "*A DT is an integrated multiphysics, multiscale, probabilistic simulation of an as-built vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin*" [78].

More in general, a DTs can be defined as a virtual machine (or computer based) model that simulates, emulates, or mirrors a physical entity (the so-called physical twin). The DT is, therefore, linked to its twin through a unique key, identifying the physical entity, however it cannot be simply considered a model or a simulation, it has to be considered an evolving counterpart that follows the lifecycle of the physical twin and learn from it, sometimes being also able to monitor and control the physical entity (or process). Specifically talking, the twining it's possible by the use of a continuous communication and interaction, through the synchronisation between the DT and the physical part, and/or an external entity that supervise the whole process [85]. Big-data storage and analytics are all technologies that nowadays are hugely common and affordable and hence, relatively easy to implement. On the other hand, prediction, monitoring and control is entrusted to both Artificial Intelligence (AI) and ML, enabling the predictive maintenance approach and real-time monitoring of the system. [85], moreover the AI may be a valid assistant for the DT to improve the product quality and enhancing the supply chain efficiency [60]. The image aside [*Figure 8*] summarises the behaviour of a DT and how it relates with its physical counterpart.

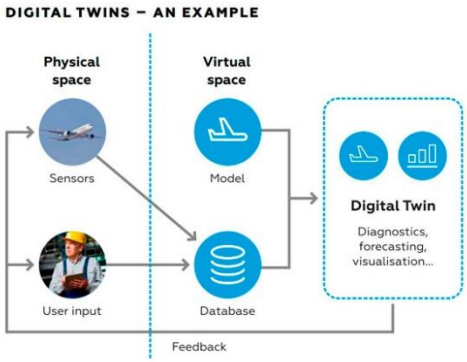

Figure 8: *a DT system from [112];*

As already explained above, the DT system is provided with both AI and ML to ensure the highest level of autonomy and precision available, however the system cannot usually be considered fully autonomous, but they still require an adequate amount of human operators especially when the operation performed is of the diagnosis type or when modification or testing are required [85].

At the begin of the section, there has been expressed a general description of a DT, in this subsection, according to the work of B.R. Barricelli *et al.* [85], are going to be showed the main characteristic that a proper DT system is supposed to be implemented with:

- Both the DT and the physical counterpart have to be equipped with networking devices to ensure the continuous exchange of data, even with a cloud-based connection, however, DT system may be completely cloud-based as in [56];
- All the data exchanged among the system have to be stored in a data storage system, accessible by the DT as well. The data to be stored, moreover, have to be both the dynamic and the static, that will work as a "memory" of the physical twin;

- The DT has to be able to manage and process high-dimensional data, hence it has to be equipped with *high-dimensional data-(de)coding* and *data fusion algorithms* to manage the data coming from different sources;
- A proper DT system implies the use of an AI, based on supervised/unsupervised learning models, enabling different applications: prediction, pattern recognition, outliers detection as shown in the work of Z. Huang *et al.* [46];
- Obviously, the DT has to provide *modelling* and *simulation* applications for depict in the best possible manner the current state of the two twins. Alternatively, it may be also applicated for the creation of simulated datasets for training ML algorithms [62];

As highlighted by the work of A. Fuller *et al.* [78], several similar terms spreaded in the field of the DT might be misleading and cause of incomprehension and hence they need to be clarified:

- *Digital Model:* a digital version of a pre-existing physical object, it lacks of any data exchange and there is no communication with its physical counterpart;
- *Digital Shadow:* a digital version of a physical object, as the name suggests, the former receive information from the physical part but not vice versa;
- *Digital Twin:* the DT is characterised by a reciprocal exchange of data between the digital and the physical twin, and a modification in one leads automatically to a change in the other.

The following image [*Figure 9*], provide a graphical description of what it has just been described above:

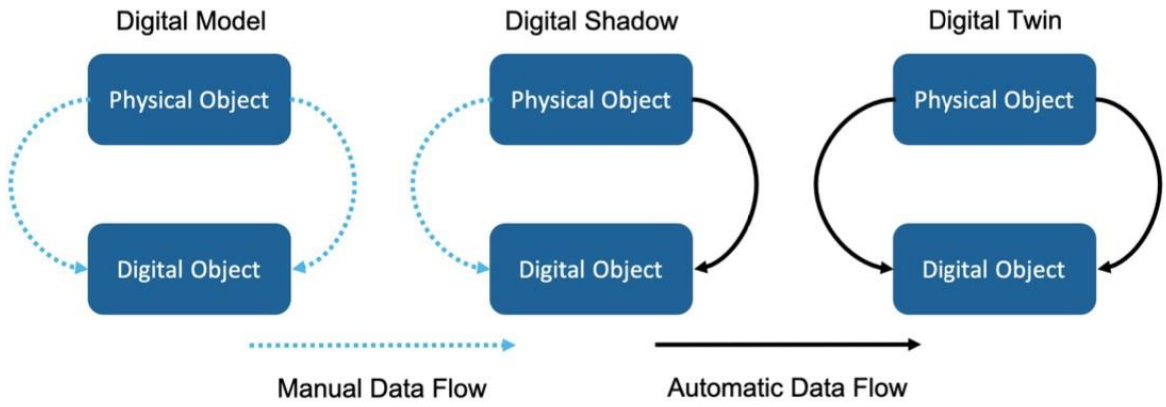

Figure 9: *Digital Model, Shadow and Twin, from [78];*

The work of Y. Xie *et al.* [77] has highlighted the procedure of a manufacturing task within an industrial environment, about the cutting machine as the following lines are going to portrait. The following data are, therefore, required to prepare a DT system for the whole life of the manufacturing tool:

- *Market Analysis data:* quantity and production volumes may be determined by the customer demand, market data and segmentation;
- *Development data:* improvement and modification on the tool design or specifications, can be obtained and shaped by the historical, functional and parametrical data;
- *Production plan data:* inventory, supplier data are all information that are definitely useful to understanding the tool and the line itself;
- *Manufacturing data:* the raw materials, once arrived at the implant are inspected during the quality testing and then used to assemble the products ;
- *Usage and service:* failure data, tool wear status and general utilisation information are, eventually, obtained directly at the usage stage of the machine. This particular type of data, since there is a huge amount of these parameters and they are also supposed to be cloud-shared, necessary to consider the implementation of specific technologies and techniques to deal with these problems, like: Apache Spark, Apache Hadoop and MapReduce that have been presented by [73]

The tool life cycle and the type of information that are obtained at each stage are depicted graphically in the image below [*Figure 11*].

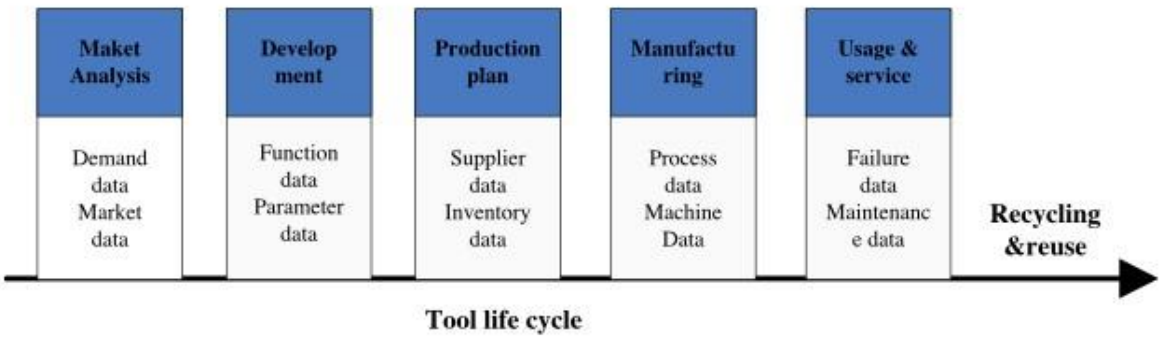

Figure 10: *data obtained over the stages of the tool cycle from [77];*

In case the amount of data available from the manufacturing phase, the work of S. Stieber *et al.* [70] explains the introduction of the "*Transfer Learning*", a technique that allows to obtain

data in high volume: initially a model is trained with simulated data and hence tuned on the actual tool with real-world data.

The DT is thought to be a complete map for its physical part, obviously the data acquisition is performed priorly to be merged with the machine status, followed with an analysis of these data. The data fusion allows the creation of a bi-directional and real-time share of data, the ML, on the other hand, is able to predict event such as tool failure over the machine life cycle. The following image [*Figure 12*] shows the interaction between basic and extended data, that are going to be explained after the figure.

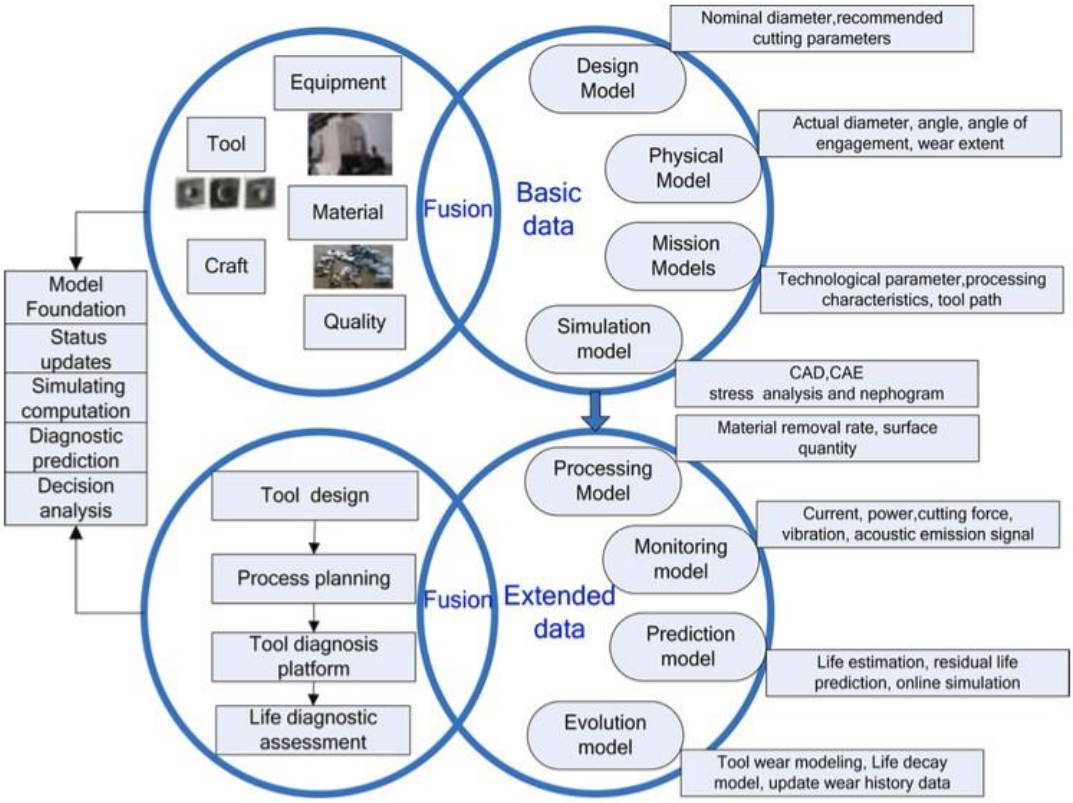

Figure 11: *interaction between extended and basic data, from [77];*

The so called "*Basic data*" are a range of data that include the design dimensions, various state data (feed speed, cutting depth), processing time etc., the shape of the digital model is thought to be a perfect representation of its physical counterpart, in every possible aspect, fact that is possible through the use of these basic data. On the other hand, the "*Extended data*" are intended to be all the series of parameters, specifications and information that are not directly accessible from the tool, but that they are computable by the basic data. Moreover, the extended data may be calculated with the use of ML models, especially for tool wear parameters.

The DT is hence thought to portrait the tool life cycle, from the begin to the decay, therefore also depicting the current efficiency status of the machine, indexes are usually calculated at this stage: RUL, surface quality, etc..

The fusion of these data is performed over three different layers [77]:

- *Data level:* acquired by customers or engineers, all the initial information is here received and stored;
- *Feature level:* the recognition and the prediction itself of the wear, take place at this layer;
- *Decision level:* tool failure, maintenance and tool wear status data are obtained at this stage, to assist in the human procedure of decision making;

Furthermore, the following image [*Figure 13*], displays the relationship between these three levels.

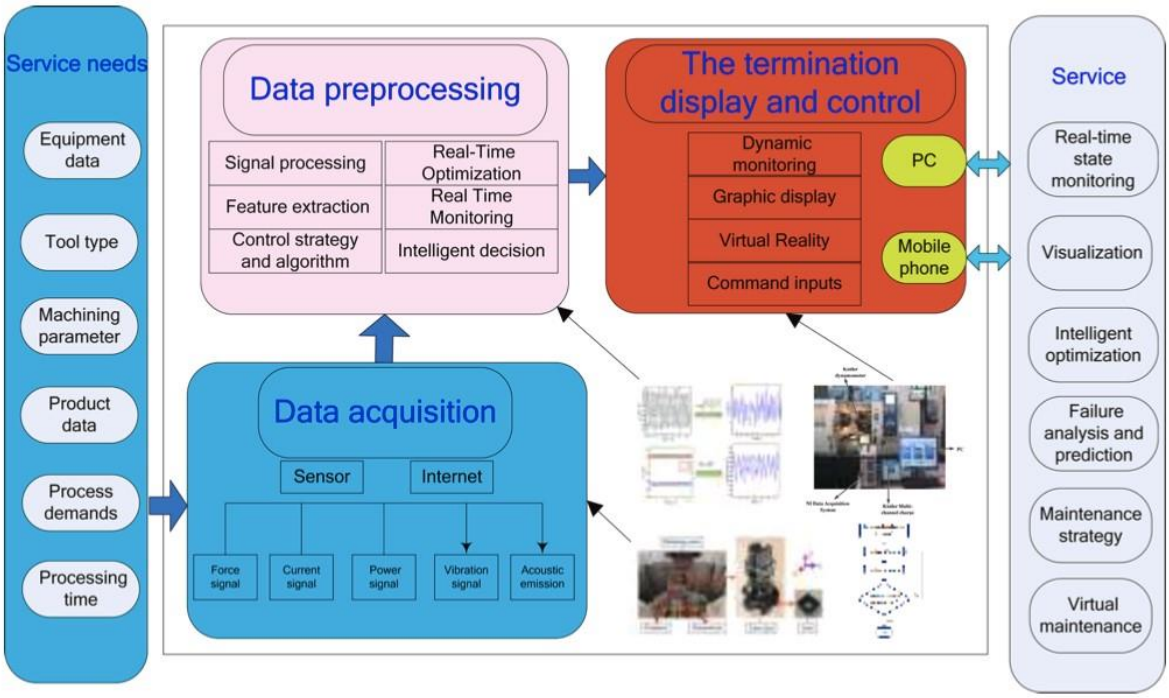

Figure 12: *relationships between layers during the fusion of all the data [77];*

As explained by the work of E.Karaaslan *et al.* [59], the DT has undoubtedly enabled a new level of innovation at the industry level, however several threats may target this newborn technology, and the protection of the corporation's technologies is a priority, especially in this

era. This section is going to review and analyse them, providing the best solution for the problem, therefore even the implementation of the technology itself requires a knowledge of the risks that come with the DT.

The main threats that have been identified by [59], are described in the following lines:

- *Physical threats:* since the interaction with the physical part, even the DT may be physically damaged, or may even provoke injuries (even fatal) to human personnel, hence it is generally considered as a top-tier in a risk-assestment, however the DT may represent the best chance to prevent any fatal injury since it can be programmed to avoid or restrict certain movement if organic presence is detected in the system, as explained by the work of C. Li *et al.* [49] and in C. Cronrath *et al.* [83];
- *System threats:* attackers might attack the operative system that host the DT, rather than the DT itself, through the use of malicious programs;
- *Software threats:* any unauthorised access to the code may compromise the whole DT since, the code is the very blueprint of the DT, and would let the attacker to have access to sensible information, such as the vulnerabilities and the characteristics of the DT;
- *ML threats:* ML algorithms are also target of malicious program, they are more vulnerable in the training/testing phase, and may compromise the reliability of the program or decrease the performance;

The countermeasure, on the other hand, proposed by [59] are showed in the following table [*Table 3*]:

Table 3: *threats and countermeasures for DT [59];*

Threats	***Countermeasures***
Physical Threats	Physical Security
Data Modification Threats	Tamper-proof and Tamper-resistant hardware, Hash, Blockchain, IPFS
Software Threats	Software Hardening, Secure SDLC, Security Testing
Data Communication Threats	Network Resiliency, Cryptographic solutions, Blockchain, Firewall, IDS
System Threats	Firewall, IDS, Antimalware, System Hardening

Data Storing Threats	Cryptographic solutions
ML Threats	Data sanitization, Algorithm robustness enhancement, Security Assessment Mechanism, Privacy Preserving Techniques

As may be seen by the table, most of these applications/techniques are quite easy to implement and should be implemented anyhow, however the implementation of some of these countermeasures should not be any prior to a risk-assessment performed in consideration of the likelihood and the impact of each event since the important cost that these techniques would have on the company/industry.

The previous lines have described the DTs in the detail, studying their features and characteristics, however the DT technology comes with issues and challenges as well, that are going to be described in this section:

- *Ethical issues:* by nature, the DTs technology is expected to compute and analyse huge amount of data, some of this data, may be classified as "*personal data*" (i.e., the height of a human operator that work with a machine), according to the definition of the GDPR, at the art. 4 [114], and hence under the protection of the European act;
- *Cost of implementation:* as explained in other sections, might be characterised by an important expenditure that can result in an unbearable sunk cost, that may compromise the whole finance of a small company. Worsen by the absence of a standardised system of DT [78];
- *Threats and securities:* the security is a source of primary preoccupation for the implementation of a DT system, since this technology has to elaborate data that may be considered delicate, the protection of these information is of primary importance for the company that have to protect against any attempt of breach [59];
- *Misleading predictions and technical limitations:* an issue that might has not seem obvious is the fact that a DT, even though it is innovative and effective, it may also be misplaced or commit errors that are not so easy to detect or correct. Another perilous scenario may be considering these system not as in assistance of human-lead task but as a substitute for them.
- *Trust:* as highlighted by [76], the rapid evolution of technology that has touched every sector of the ordinary life, like food and smart cities, that may not be accepted easily

by everyone. Therefore, the Context-Aware System (CAS) may become an important factor to the implementation of these new technologies;

2.3.2. DT and ML for production control and monitoring

As highlighted by the work of Q.Min *et al.* [80], the DT would enable a continuous interaction and analysis: a system non-DT based, is strictly dependent on the presence of an expert human operator or at top, to a one-time ML output. On the other hand, the DT system is able to guarantee that the data collected in real-time in the physical factory are stored with the historical ones and both elaborated (training and testing) and provide real-time feedback to the industry for the production control, there are also examples of implementation of DT for the direct control of machinery in an industrial environment, such as the work of S. Wu *et al.* [54], where the DT operates on Automated Guided Vehicles (AGV) or the article of Z.Liu *et al.* [51] with the implementation of Petri Nets (PN).

he ML is perfect for any implementation of DT, since it may be applied in different task and situation to obtaining the most useful information: it may be able to compute and foreseen the expected Lead Time in the machines [87][63] or it may be applied to predict the supplier performance over time [81].

Hower this system cannot be expected to be costless nor easy to implement, since it would require a great amount of chip and sensor, virtual environment, connection between the machines and the virtual environment etc., The source [80] also provides a graphical representation of how a DT relates to the physical environment [*Figure 10*]

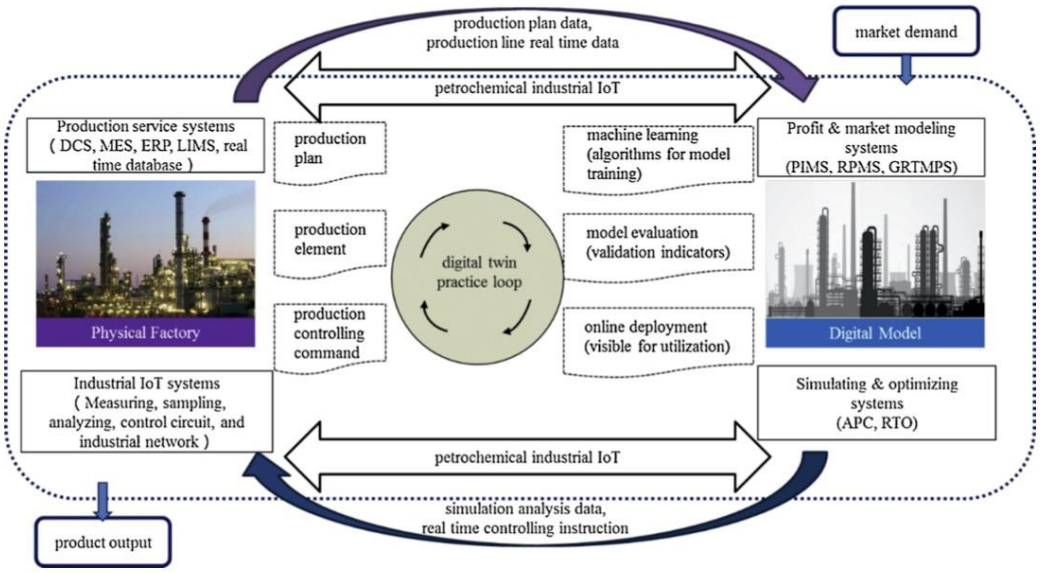

Figure 13: *A DT for industrial production control [80];*

According to the picture above, the DT (initially training by a ML model using historical data, then deployed online) receives the real-time data from the physical environment and elaborates various data (optimisations, market modelling and profit maximisation), and the computed results are hence used to manage the factory, through secondary systems, such as MES and ERP, or even more peculiarly, the ERP can be managed directly by the DT system as described in the work of M. Dehghanimohammadabadi *et al.* [58], similarly a DT can automatically chose the strategy (i.e., FIFO, LIFO) according to the necessity of the implant [65].

According to the sources analysed, the Additive Manufacturing (AM), and also the Wire Arc Additive Manufacturing (WAAM), seems to be appealing field of application for the most recent innovation such as the DTs. The AM, also known as 3D printing, is the technique of building parts, layer-on-layer, directly from the raw materials, the WAAM, in particular is a subcategory that uses the heat generated by wires to shape the material to be used [47]. Typically, the AM relies on trial-and-error techniques before achieving any defect-free product, the DT may be applied to reduce the time necessary to this trial and error since it can provide a higher level of detail for the data received [57], however it may also be implemented for a 'general' task of monitoring of the machinery, predicting the process parameters [74]. As highlighted by the work of . H.Ko *et al*. [82], the evaluation performed on these models are either physical, either numerical: the former refers to the prediction precision of the physical phenomena, the latter, on the other hand, refers to the efficiency of the mathematical model

used for the ML. Eventually, the sources [89] and [55] both provide additional implementation procedures for DT systems in WAAM manufacture.

2.3.3. DT and ML for simulation

Another, important, field of application for the DT is the simulation: simulations of event that cannot be recreated on a real device either for human endanger, for time constraints [80] or because it is too complicated to figure and understand numerous relationships in a device allowing the detection of errors in the project at an early stage [84].
A world that applies the DT technology consistently is the Formula 1, where the time constraint is important, and on the other hand, an extremely high efficiency is required. Each team has to verify the reliability of countless of components each week, and other parts like powertrains and tyre compounds have their worn status checked weekly [117]. The DTs perform a primary task in developing the car performance, since they can receive continuously new data coming from the vehicles, and hence provide the engineers with new data to work on to develop or take decisions during the race. A further example of this field of application has been provided by Tesla, that implemented a DT for the simulation of the car engine and other component [80]. In these applications, the relationship, between the DT and the AI is of primary importance, because the final output is generally performed by the AI that using data analytics can improve the quality of the forecast of the components, in particular it may be used to enhanced to train the ML training phase and improving the final output of the AI [72][67]. Z. Wua *et al.* [61], is one of the few sources that explained in the detail the framework of their proposed a RNN based model to monitor the health of aircraft engines, thought to manage the monitoring, visualisation, data storage, analysis and eventually control resulting in a system capable of calculating the RUL, to diagnose and prognose the engine.

2.3.4. DT and ML for predictive maintenance

The last field of application of DT that is going to be studied in this work is the predictive maintenance, that may be even considered a consistent part of the monitoring and control field.

The DT may be in fact created to mirror a device or a machine that is subject to worn, they are able to detect both anomalies and predict maintenance to a single object. As explained in the work of T Bornagiu *et al.* [68], the process is divided in four different layers:

- *Collecting data streams:* linked to the physical part, the DT collect several types of data, not only from the device but also from the environment, events etc. Eventually the system should also be prepared to properly construct and assemble the data and convert these data according to a standard, as explained in [88] and [50]. The positioning of the sensors over the device is a complicated task and should be performed by an expert, however a sensibility analysis can be defined as a good practise to avoid any uncertainty [71];
- *Processing analysis data streams:* in this face the DT has to separate and align data from the streams in standard time slices and grouping them according to their covariance. In this layer, an anomaly may be detected and if necessary, the DT may signal to block the machine and isolate defective resources as shown in [86], applied for CNC milling machine;
- *Machine Learning:* extract online insights from the aggregated data streams from the shop floor of resources, process and environment and paralleling computing patterns and orders. The use of ML usually implies prediction, classification, and clustering, performed with different models, however the NNs have proven themselves to be the most effective method for tool wear prediction with historical data availability [75]. The system should also be able to manage and deal the conflicts of data in the and to direct the system [90];
- *Decision making:* the outputs given by the other layers, allow enabling the predictive maintenance and a global optimisation of the process, however it is possible that the DT itself can be able to perform self-maintenance and self-optimisation [53], or at least provide the exact guideline to the human operator [48]. As depicted by [64], there are situations where the algorithm has been designed to receive more than one output (obtained with different method), and the DT system itself decides. After several training and testing phase, which one is the method that best describe the phenomenon of the device on which the DT is working on;

A typical characteristic of this application of DT system is a strict cooperation with a human operator, and several solutions to facilitate the interaction have been proposed, one example is

depicted in the work of T. Wang *et al.* [66], they implemented a DT system with a Visual question answering (VQA), an embedded technology that, through the deep learning, is capable of visual understanding, text information understanding and reasoning, enabling the DT to understand simple question from the operator or to answer to multiple-choice questions.

3. Machine learning algorithms

In this chapter the most common and useful algorithms are going to be studied and analysed, depicting the mathematical calculations behind them and the architecture of these models. Preliminary, it may be useful to provide some general definition:

- *Unsupervised Learning*: reveals the underlying pattern in the dataset not explicitly presented [102].
- *Supervised Learning:* learns a function to make a prediction of a defined label based of the dataset provided [102].
- *Reinforcement Learning:* the ML learn to operate accordingly to the interaction with the environment, similarly to a trial-and-error approach [102].

However, the ML model analysed here are mainly of the supervised learning.

According to the sources analysed the most used models are displayed in the following chart [*Figure 14*] and hence the ones that are going to be analysed in the following sections:

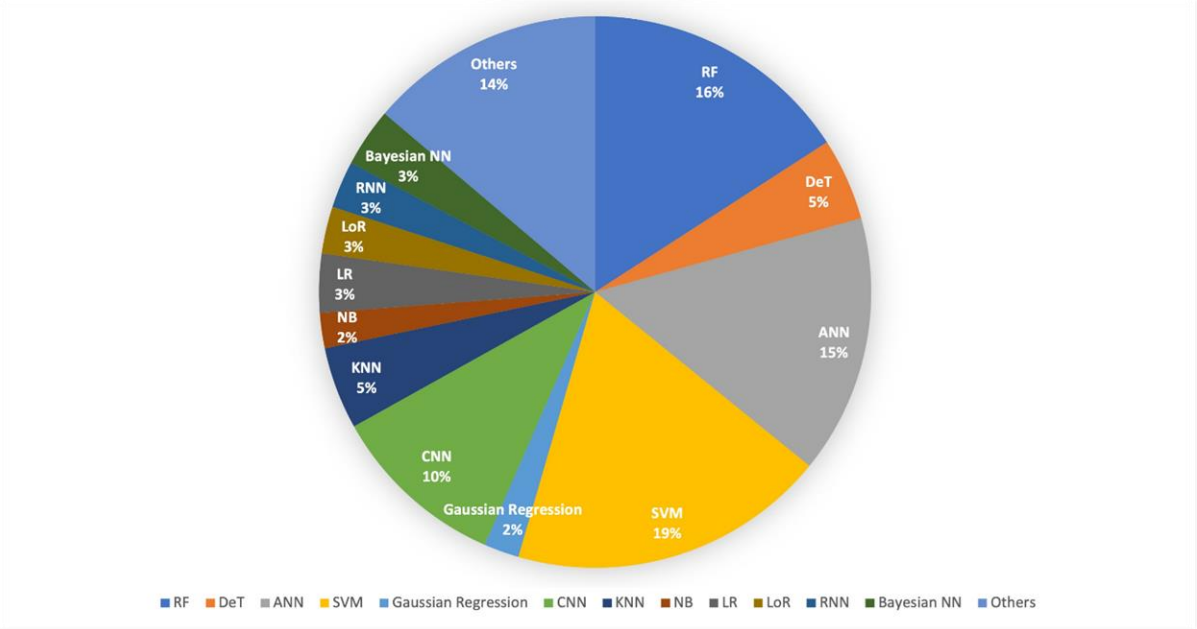

Figure 14: *pie chart with the models most seen in the sources;*

Eventually, in the section 2.3., the main libraries that are necessary for any implementation in Python of the ML are described briefly.

3.1. Classification ML Models

As already discussed in the first chapter, while describing the work of the authors there are two type of ML models: one is the regression, used to predict continuous values and the second one is the classification that, as the name suggest, classify the values in two or more categories. In this section, the latter are going to be studied: the classification, in the details, is a process that aims to find a model that automatically divide the dataset in the category needed, according to several parameters provided. In this section, hence, the most important classification models are going to be analysed.

3.1.1. Decision Tree

The DeT has been applied by O.Khalaj *et al.* [52] for the research of features in the implementation of a DT for steel alloys obtaining relevant results, moreover this model can be used for both classification and regression, however it is definitely more effective in the classification processes [100]. The logic behind a DeT, that is a supervised model, is quite simple: it is a tree structured (see the image [*Figure 15*] aside from the source [52]) classifier composed by three types of nodes: *Root*, *Interior* and *Leaf.* The root node is obviously the initial node and represents the whole dataset, the interiors are the features of the dataset and eventually the leaves are the outcome of the queries. Hence, the output or the leaf is determined exclusively by a True/False mechanism. It has the advantage of being quite simple to understand and may help with the data cleaning problem, however it is usually subject to overfitting (inability to generalise the algorithm) that makes it unreliable to run efficiently [100]. The RF algorithm is a derivation of the DeT that is able to deal with this last problem and it is going to be analysed also in its classificator version in the following subsection.

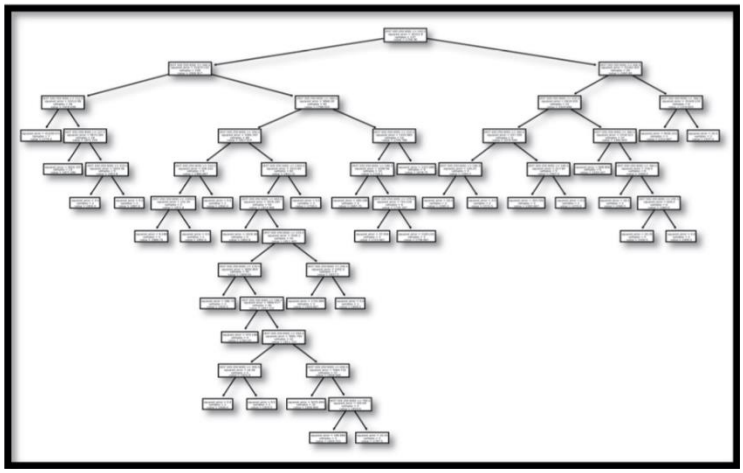

Figure 15: *graphic representation of the DeT of [52];*

3.1.2. Random Forest

According to the several sources reviewed in chapter 1, RF is considered one of the most effective ML algorithms for both classification and regression problems. RF is a decision tree, and indeed a derivation of DeT described at the subsection 2.1.1.. The RF, differently from DeT, is based on *Ensemble Learning*: a technique that implies the use of multiple trees, averaging the result, maximising the efficiency and hence resolving the overfitting problem, these models are therefore trained with different samples of the original dataset through the use of the *Bootstrapping* technique that randomly chose these samples [101]. These procedures are shown graphically in the figure below *[Figure 17*], and moreover it is also reported the computational flow chart [*Figure 16*], both coming from the source [11].

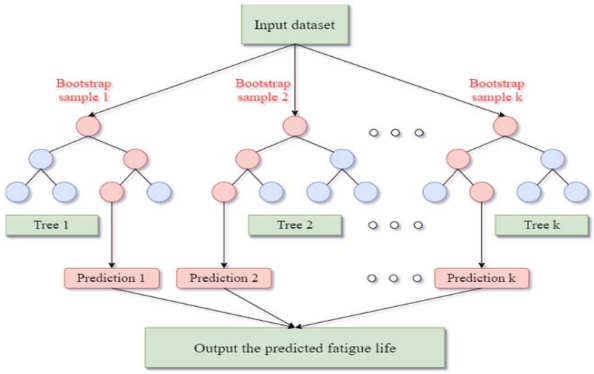

Figure 17: *a schematic diagram of the RF of [11];*

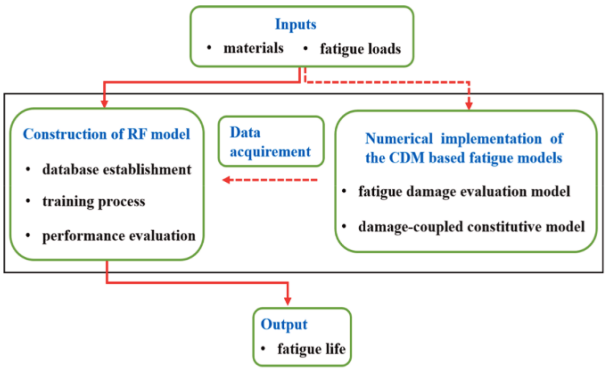

Figure 16: *computational flow chart of the RF of [11];*

A further analysis regarding the regressor version of the RF is going to be performed in the following chapter 2.2.3..

3.1.3. Support Vector Machine

The SVM is a supervised ML algorithm that can be used for both classification and regression problem, however the SVM is more reliable in the former, and it's one of the main classificator, being easy to implement, fast and robust [33]. The SVM can be divided in two categories:

- *Linear SVM:* used for linearly separable data, meaning that if a dataset can be classified into two classes by using a single, straight, line then it is a linearly separable data [103];
- *Non-linear SVM:* if a dataset cannot be classified using a straight line [103];

There may be a multitude of "lines" for divide the dataset, but the best decision boundary has to be determined anyhow and it is called *Hyperplane.* The data that are closest to the hyperplane, and hence affecting it are called *Support Vector* (SV) because their role [103]. The graph [*Figure 18*] aside shows the concepts that have just been described. The work of H. Wang *et al.* [5] highlights the importance of the Kernel Functions in the SVM model to avoid the curse of dimensionality by replacing high-dimensional data calculation, beneath this subsection a table [*Table 4*] depicts the main Kernel Functions.

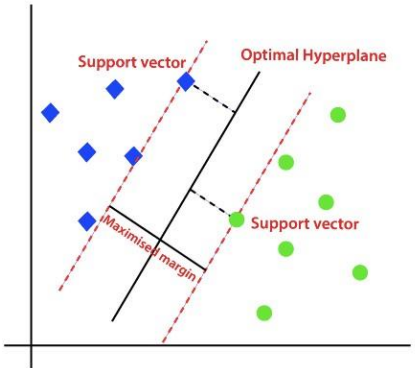

Figure 18: *graph illustrating the main components of a SVM model from [103];*

Table 4: *Kernel Functions from [5];*

Kernel function	***Analytical formula***	***Parameter***
Linear Kernel	$k(x_i, x) = (x_i^t, x)$	-
Polynomial Kernel	$k(x_i, x) = (x_i^t, x)^d$	$d \geq 1$
Radial basis function Kernel	$k(x_i, x) = e^{-\frac{\|x_i - x\|^2}{2\sigma^2}}$	$d > 0$
Sigmoid Kernel	$k(x_i, x) = \tanh(\beta x_i^t x + \theta)$	$\beta > 0, \theta < 0$

Conversely, the role of these Kernel Functions is to convert the raw data into a suitable format for the SVM algorithm to process, as shown in the figure aside [*Figure 19*] from [5]. Where *w* is the normal vector, output of the Kernel Function. At the very end of the algorithm, the result of the SVM is given by the following system of equations [104] :

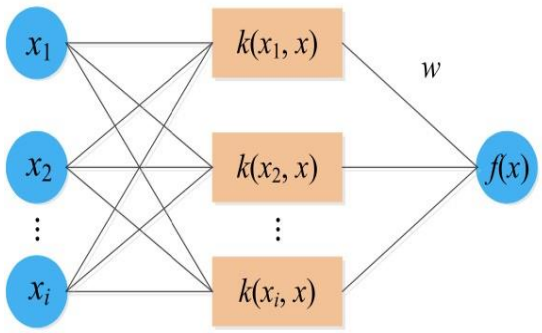

Figure 19: *network structure of the SVM proposed by [5];*

$$y = \begin{cases} 1 : w^t x + b \geq 0 \\ 0 : w^t x + b < 0 \end{cases} \tag{2}$$

3.1.4. Naïve Bayes

The NB, a supervised ML model as well, is called this way since it depends on the Bayes' theorem [25]:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3}$$

The NB classifier simplifies the assumption that the predictors are conditionally independent of each other given the class, however it doesn't seem to affect the robustness of the model results since, they are related to the highest probability and only the single observation might be inaccurate. Actually, it has been proven that the model does work efficiently even when there is a dependency among the classes. Therefore, under the assumption of independent variables the probability of class A occurring due to a known value of data B is given by [25]:

$$P(A = k|B1, \dots, Bn) = \frac{\pi(A = k \prod_{j=1}^{n} P(B_j|A = k)}{\sum_{k=1}^{K} \pi(A = k) \prod_{j=1}^{n} P(B_j|A = k)} \tag{4}$$

As already explained, the above equation gives the probability of A=k (*i.e.*, worn, or not worn), but the algorithm requires an additional part that gives a value or another according to what is the classification foreseen, this equation is the Bayes' classifier [105]:

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y) \tag{5}$$

The GP model, eventually, is a similar approach to both the classification and regression problems, but it is slightly less common than the NB.

3.1.5. K-Nearest Neighbour

The KNN is one of the simplest ML models among the supervised ones: it assumes the similarity between data and categorise it to the most similar available, its peculiarity is, indeed, that it doesn't require an actual training phase, but it stores all the dataset for classify the new data [106]. K, is an integer chosen by the user and it is the number of "neighbours" to consider. [8]. A sequence of the procedure just described is depicted in the following picture [*Figure 20*] below:

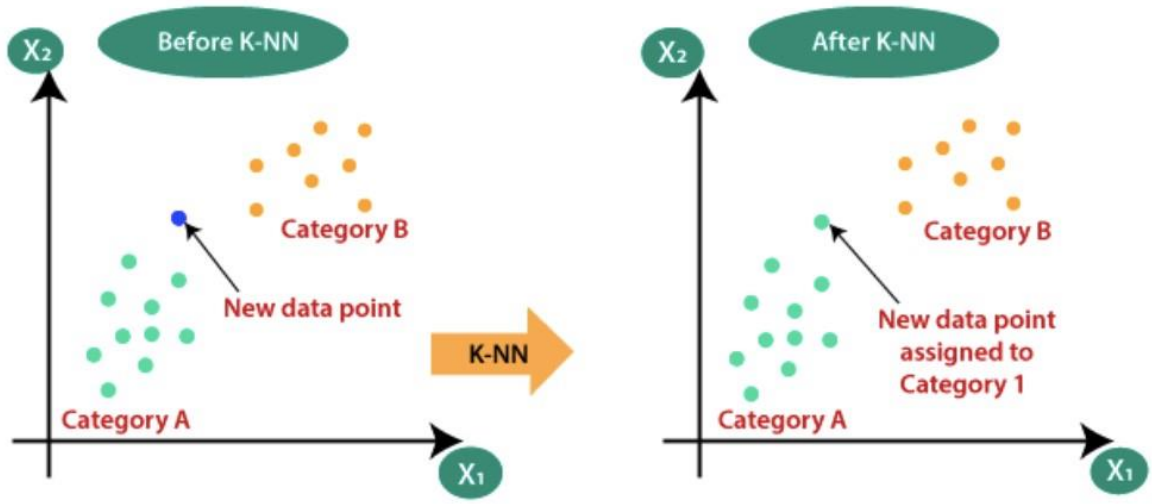

Figure 20: *sequence of the operation carried by the KNN algorithm [106];*

The nearest neighbour is chosen with the Euclidean distance between the data points, obviously the closest is chosen [106]. The choice of K can be done with the elbow method or a cross-validation [8], conversely the higher K is, the more KNN is immune to the outliers.

3.1.6. Artificial Neural Network

The ANN is part of the family of the NN, a type of ML based on the human biology since it reproduces the connection present in a human brain, concept that can be shown graphically with the following images [*Figure 21*][*Figure 22*] [107]:

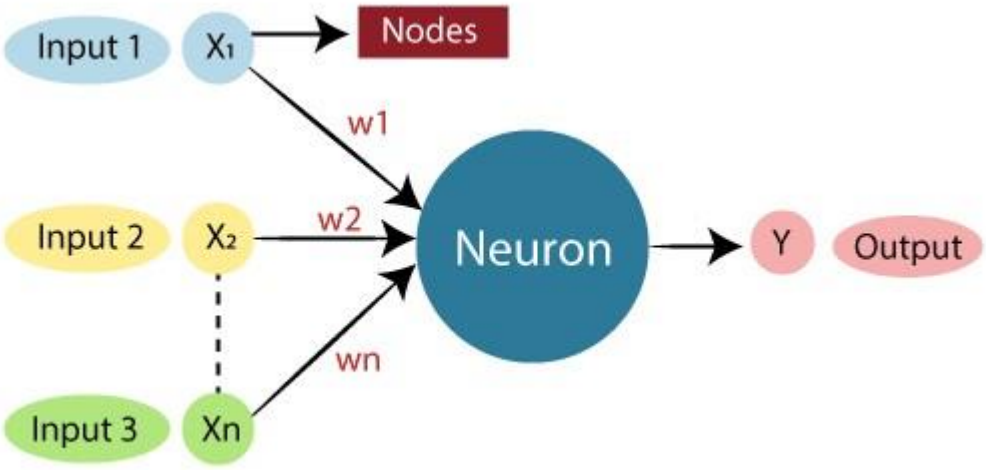

Figure 21: *graphical representation of the system running in a NN model [107];*

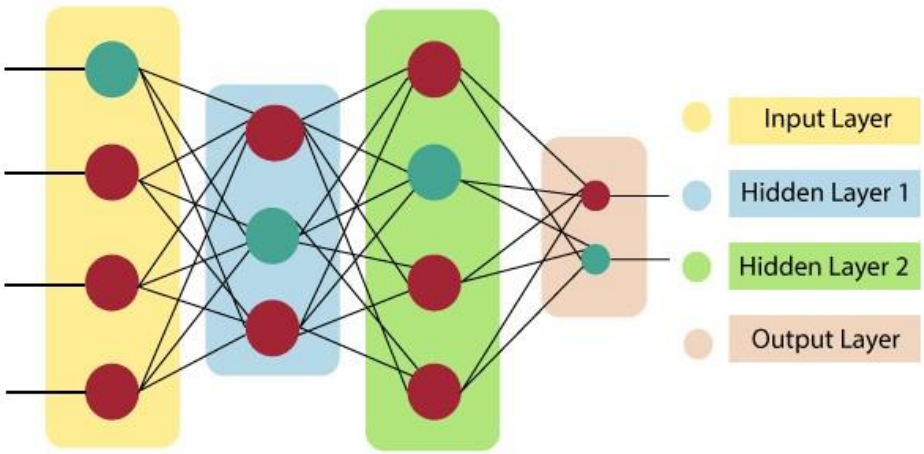

Figure 22: *ANN inner structure [107];*

In the details, the second picture shows the inner structure of an ANN , the hidden layers are the phases where the algorithm make its calculation and tries to find a pattern in the available data. The following equation is the output expected by the ANN model:

$$y_j = f_i\left(\sum w_{ij}x_i - \theta_i\right) \tag{6}$$

Where x_i is the input signal of neuron, f_i is the activation function and Θ_i is the error threshold of hidden layer neuron [5]. The activation function is a set of transfer function used to obtain the required output, it may be binary, linear, hyperbolic, etc. [107]. In order to minimise the error between output the result and the test, the weight and the threshold are updated by the back-propagation algorithm and gradient descend algorithm during the training as shown by the following equations [5], the iteration stop when the threshold is reached as showed.

$$w_{ij}(\sigma + 1) = w_{ij}(\sigma) - l_r \frac{\partial e(\sigma)}{\partial w_{ij}(\sigma)} \tag{7}$$

$$\theta_{ij}(\sigma + 1) = \theta_{ij}(\sigma) - l_r \frac{\partial e(\sigma)}{\partial \theta_{ij}(\sigma)} \tag{8}$$

3.1.7. Convolutional Neural Network

The CNN, a component of the NN family, is famous for being one of the most used algorithms for the processing of grid-like topology, such as an image [108]. A CNN is typically divided in three different layers:

- Convolutional layer: it performs a dot product using two matrices, one composed by learnable parameters and the other by a portion of the receptive field [108];
- Pooling layer: it replaces the output from the convolutional layer by computing a statistic of the nearby outputs [108];
- Fully Connected Layer: it is useful to connect the input and the output [108];

The following image [*Figure 23*] show the typical structure of a CNN model [109], where the input is convolved for producing the activation map, letting the model to learn the features of the input.

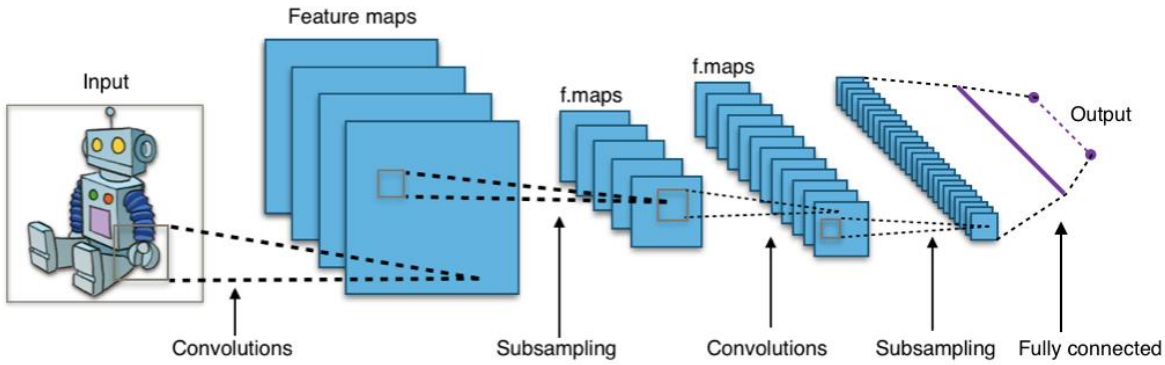

Figure 23: *typical structure of a CNN model from [109];*

3.1.8. Recurrent Neural Network

The RNN is going to be the last model to be seen in this work, and it is mostly used for processing information that is sequential: a speech, for example, indeed its name, recurrent, comes by the fact that the model repeats the process for each element and the output is depended to the previous calculations [110]. Mathematically speaking, the process that allows the RNN is provided by the following equations:

$$h^t = g_h(w_i x^t + w_R h^{t-1} + b_h) \tag{9}$$

$$Y^t = g_y(w_y h^t + b_y) \tag{10}$$

Where *w* is the weight of the value, *x* the input and *b* the bias of the observation. The process is, therefore, repeated continuously for training and hence improve the quality of the output [111].

3.2. Regression ML Models

As already explained in the section 2.1. the regression is used to foresee the continuous value of a dependent variable in relation with another one that is an independent variable, and it is also called regressor. Although, what is exactly a regression? A regression is defined as a statistical technique that compute the strength of the relationship between the regressor and the dependent variable using experimental data. A certain percentage of the data available is intended to be trained in the fitting process of the model the rest is used to verify the validity of the model: the lower the obtained error, the more accurate is the model. In this section the most important regression models are going to be studied.

3.2.1. Linear Regression

The LR is one of the simplest supervised algorithms and one of the easier to implement, it's simpleness may be useful when the prediction is just considering one regressor and the relationship between is supposed to be linear. This model has been used by N.S. Karuppusamy *et al.* [45], providing also the equation shown underneath this description.

$$Y = \beta_0 + \sum_{i=0}^{n} \beta_i x_i + \varepsilon \tag{11}$$

Where x_i is the regressor of the i-th component, Y is the dependent variable, β are the coefficient of each component of the equation and eventually ε represents a residual to the dependent variable Y. The coefficients β are estimated with the following equation:

$$\beta = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{12}$$

This equation is obtained by the ε minimisation problem of the (10).

3.2.2. Logistic Regression

The LoR is depicted in the work of J. Karandikar *et al.* [40] even though in the article it has been used as a classification method, it is not a core competency of its, but it can be used as classificator when a threshold command is used in the algorithm. The LoR model, a supervised algorithm, is a Log shaped (s-curve) graph and typically, it is used when Y is a dichotomous variable (labelled either 0 or 1) implying that all the values between are the probabilities of Y to be one of the two. Its equation model is given by:

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{13}$$

Where $p(x)_k$ is the probability that the respective Y_k has one of the two values or (if it is not dichotomous) p(x) is equal to Y.

3.2.3. Random Forest

The analysis performed in the subsection 2.1.2., showed the classificator side of the RF, the work of Z. Zhou *et al.* [11], however showed how even in the regression role the RF performed efficiently. The RF is, indeed, composed by several DeT composed by leaves and roots. The training stage of the RF model consists in developing these DeT, uncorrelated, and the final result is given by the average of the result of all the DeT, on which the RF was trained specifically for. Therefore, the final output of the RF for the regression is given by [11]:

$$Y^{pre} = \frac{1}{k}\sum_{i=0}^{k} Y_i^{pre} = \frac{1}{k}\sum_{i=0}^{k} f\left(X, S_n^{k)}\right) \tag{14}$$

As already explained, Y^{pre} is given by the average output of the i-th tree, or alternatively may be computed directly in the second equation where $(S_n^1, \dots, S_n^k)$ is the vector of the bootstrap samples, and X represents the input variables. The non-linear relationship between X and S, f(X, S) is built once the training stage is completed [11].

3.2.4. Support Vector Machine

After the analysis portrayed in the subsection 2.1.3., the SVM regressor is going to be studied with the help of the source [2] that used it for predictive control on milling machines. The SVM in the regression role may also be called SVR. M. Ay *et al.* [2] provide a full description to the SVR: it has been firstly studied by Vapnik and Lerner in 1963, originally thought to be just a classification model, but it was developed later by Vapnik in 2000 with the following regression equation:

$$y = \omega^T \cdot \theta(x) + \rho \tag{15}$$

Where ρ is the bias, x is the input vector and ω^t Θ(x) is a function useful to map the input and weight the result.

The learning ability of the algorithm is described by Vapnik as the optimisation of a cost function J:

$$\min J = \frac{1}{2} \cdot \|w\|_2^2 + C \cdot \sum_{k=1}^{N} (\varepsilon_k + + \varepsilon_k^*) \tag{16}$$

Where the first term represents the structural error and the second the empirical error. The solution of the optimisation problem leads to the Lagrange coefficients α_l, and w can now be computed as follows:

$$w = \sum_{l=1}^{N} (\alpha_l - \alpha_l^*) \cdot \theta(x_l) \tag{17}$$

Hence, the initial formula of Vapnik can be reworked with the new term [2].

3.3. Python libraries

Python is one of the most important and diffused programming languages among the object-oriented ones, and it's characterised by a great readability and a relatively easy implementation According to most of the sources of chapter 1, Python is the most used language for the

implementation of ML algorithms, it may be confronted with Matlab, but the former is undebatable easier to implement and to read, easier to transport and it posses far more capabilities (at least with the countless libraries that can be imported in the code). The following subsections are going to briefly reviews the main libraries that are implemented in ML algortihms.

3.3.1. Matplotlib

As the name suggests, Matplotlib is a library able to compute laborious mathematical operations and plotting the results of these last-mentioned. Matplotlib is a fundamental package for the implementation of a ML algorithm, it allows to verify graphically the result or see the pattern of the data. Eventually, Matplotlib is also capable of plot several charts at once, allowing even a certain amount of customisation of the graph plotted. Below is reported the importing line of the library [*Code snippet 1*], importing it with the command "*as*" allows to short the name of the method once called in a function or in a class.

```
import matplotlib.pyplot as plt
```

Code snippet 1: *importing line of Matplotlib;*

3.3.2. Numpy

NumPy is Python package fundamental for scientific computation, enabling calculation with arrays and matrices, but also gives to the user several command (as the square root), essential but absent in the vanilla release of Python, and since its structure, the library accelerates and makes smoother all the calculations that are performed in the algorithm [95]. Below has been provided the import line of the NumPy library [*Code snippet 2*].

```
import numpy as np
```

Code snippet 2: *importing line of NumPy;*

3.3.3. Seaborn

Seaborn is a Python package and an extension of Matplotlib, that allows the algorithm to elaborate and plot graphs with a statistical purpose (i.e., "*scatterplot*" and "*correlation matrix*")

as described in [96]. Below is reported the import line of the Seaborn package [*Code snippet 3*].

```
import seaborn as sns
```

Code snippet 3: *importing line of Seaborn library;*

3.3.4. Pandas

Pandas is a data manipulation and analysis library. The operation that Pandas is able to perform are several: join, merge, split, indexing, select, grouping and ordering, it is able to both create the database in-app or importing and reading it from other sources (i.e., local hardware or in-cloud database) in multiple format: xls, SQL and csv. Pandas is also known since it's capable of performing data cleaning and since it is extraordinary easy to use and to work on. Below, is reported the importing line of the package [*Code snippet 4*].

```
import pandas as pd
df = pd.read_csv('/Users/administrator/... /experiment.csv')
```

Code snippet 4: *importing line of Pandas and of an external file;*

The second line shows the command to use an external file as database in Python, it will not be saved and hence any manipulation prior to the stop of the application run has to be repeated in any following run.

3.3.5. Scikit Learn

Scikit Learn, eventually, is the library that mathematically enables the ML algorithm since it brings to the code both the regression and the classification models. Scikit Learn, moreover, is thought to work in collaboration with the Python libraries that have been described in the other subsections, especially Pandas and Numpy [98]. Underneath this description some command for importing the packages from Scikit Learn are showed [*Code Snippet 5*]

```
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix,
f1_score, ConfusionMatrixDisplay
```

Code snippet 5: *implementation of Scikit Learn library and importation of necessary packages;*

Metrics is a module that contains the statistical evaluations of the ML modules present in Scikit Learn, including F1, R^2 and Mean Absolute Error (MAE), hence it's also a fundamental package for any implementation of ML algorithms.

4. Steps of Machine Learning implementation

This chapter is going to review step-by-step the main phases that characterise any implementation of a ML algorithm, from the collection of the data to the evaluation of the model, distinguishing, when possible, particular cases or different situations that may occur during the implementation.

4.1. Collection of the data

The first step is obviously the collection of the data, that varies according to what type of task or machine is going to be analysed and prepared for.
Normally the data are obtained experimentally, hence with several datasets where the authors may have varied some condition and kept constant others, as a rule of thumb: to keep constant are supposed to be only the condition that are not expected to change in an actual application (i.e., the material or the federate). Otherwise, some sources have also implemented a ML model using external datasets that have been provided by public sources like NASA or Universities.
There are, therefore, different ways to extract a dataset from a working machine [115]:

- *Sensors:* fundamental for any dataset, they can be placed on the machine or nearby, in the environment. They have the advantage of being relatively low-cost solutions and are easy to position, however they require precision when they have to be placed on the machine since, a misplaced sensor may lead toward an unsatisfactory dataset or worse, to a biased dataset;
- *Operators:* some information can only be collected by physical operators. Typically, these types of information are Boolean (i.e., worn/not worn) and have to be bounded with other types of information to be relevant for the ML algorithm;
- *Connected systems:* in some cases, the machines are connected to other systems or a DT, that are able to provide more accurate, in real time and that would be harder to collect manually;
- *Machine tools:* most of the modern machine are already capable of produce and share, reducing hence the need of the sources listed above;

The following image [*Figure 24*] shows a generical configuration for a dataset collection, it is also possible to see the placement of different sensors that have to collect different data type:

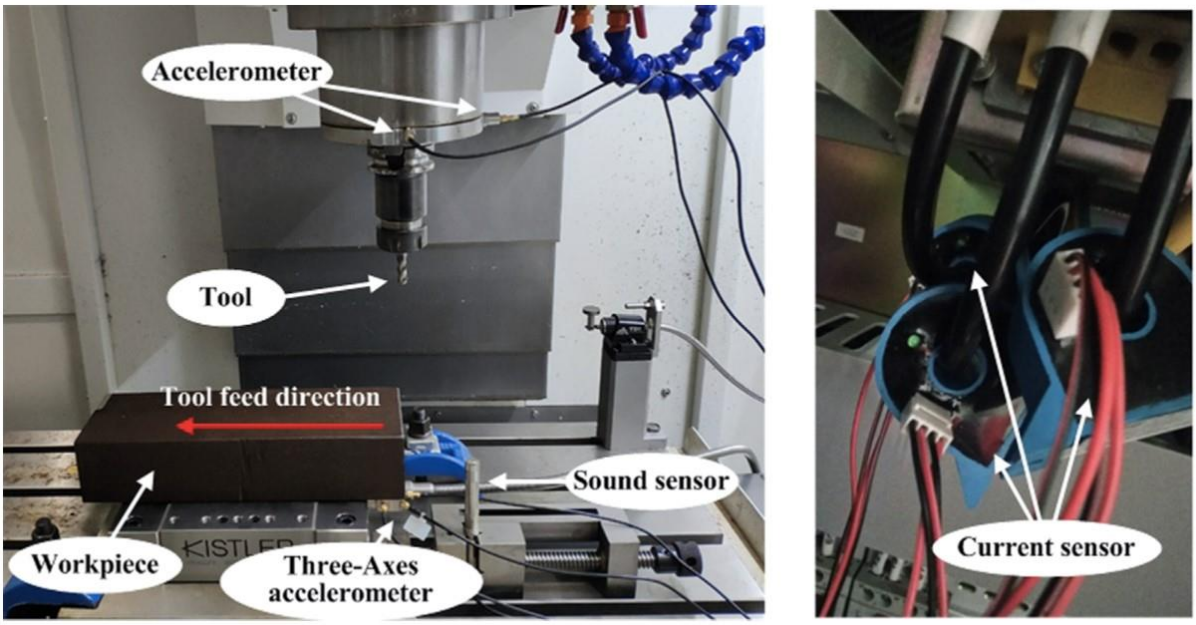

Figure 24: *experiment station from [24]*

4.2. Data normalisation and cleaning

As already explained the first section, the number of features to be extracted are several or even more and an obvious problem is that each feature has a different dimension, unit of measure etc., this inconvenient may lead to an extended time for the training in the model (typically 80% of the whole dataset is used in this phase) and an eventual failure in the convergence of the model [3]. The most common solution to this problem is the normalisation of the dataset and especially the z-score standardisation, to eliminate the difference between the feature and utilise the whole information available from the dataset. The formula of the z-score is the following:

$$x^* = \frac{x - \mu}{\sigma} \tag{18}$$

The standardisation remaps the dataset to an interval [-1,1], equalising the weight of all the values. Other known techniques are the '*Feature Clipping*' that, as the name suggests, clip the

value in the datasets to avoids the outliers, or the '*Log Scaling*' that is able to compress a wide range of values into a narrower one using the following formula:

$$x' = log(x) \tag{19}$$

The normalisation became necessary especially when in the dataset are present also information that are categorical and that can assume value way higher than the measurements that have been taken on the machine, even though they are not likely to be that important.
Eventually, the last operation that has to be performed directly on the dataset, is the data cleaning in order to ensure the algorithm to be more efficient and faster. Typically, the data that should be removed are:

- *Duplicates:* in high-volume collection of data is very likely that duplicate values are present in the dataset, these rows should be removed since they don't add information to the algorithm, but it just makes it slower;
- *Outliers:* the outliers are value that are significantly different from the rest of the dataset, maybe because collected under particular circumstances, and hence should be removed from the dataset, however in some cases they may be kept in the database, but the ML chosen should be one that is unlikely to be biased by outliers;
- *Incomplete rows:* sometimes, may happen that the collection of the data is only partially, due to errors in the system or in the sensor, however the status of incomplete makes them unusable for a ML algorithm hence they should be removed;
- *Structural errors:* errors may be anything that is not consistent with the rest of the dataset and that can be defined as mislabelled categories, theoretically these errors may be corrected in the dataset and not removed, at least in some cases;

4.3. Selection of the features

Typically, during the collection of the data several different features are gathered, even though a certain amount of them is not decisive for the algorithm and may even hamper the regression. The best practise for identifying empirically the feature is considering the correlation of two features, the Pearson Correlation Coefficient is reported below [3]:

$$Corr(x_i, x_j) = \frac{Corr(x_i, x_j)}{\sqrt{Var(x_i)} \cdot \sqrt{Var(x_j)}} \tag{20}$$

A graphical and quick solution is the heat matrix showed aside [*Figure 25*] from [3]: each square shows the correlation [-1,1] between the two features with a different intensity of the colour accordingly.

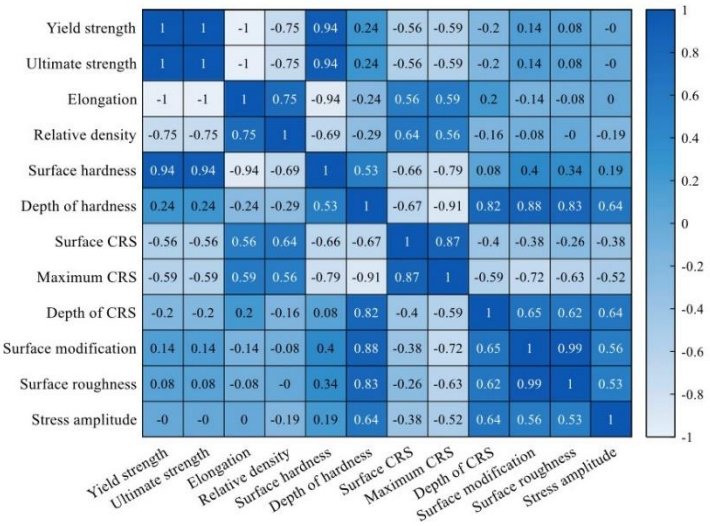

Figure 25: *heat matrix between features from [3];*

According to this matrix, the feature should be selected when they reach a certain grade of correlation in the label that is going to be the target of the algorithm. However, there are some 'trick' to enhance the result, for example in the work of [43] didn't considered any feature with a higher correlation then 0.9, since they don't add any relevant information to the model. Alternatively, it can be used the Anova F-Value that gives an idea of how much two features are related [17].

Eventually, each feature has to contain only value that can be related to other values: typically, there may be several featured labelled with strings (name of the material, worn/ not worn, phase), hence all these features should be converted in a value: worn/not worn will became 1/0 for the calculation and then it may be reconverted back for the final output.

4.6. Split of the dataset into train and test sets

The split of the dataset may be seen as a simple procedure; however, it is a delicate operation that has to be studied very carefully: the dimensions of the two derived sets and the way that the dataset has been collected are both parameters that may affect importantly the efficiency of the algorithm.

The dimensions of the dataset are typically 80% of the dataset that is used for the training phase and the remaining 20% to the test of the model. However different strategies might be taken into account for a specific model that have a particular learning capabilities increasing or decreasing the size of the training size,

A second matter is the way in which the whole dataset has been collected: it is important indeed to maintain the i.i.d. condition in entire dataset, and especially between the train and the test sets. In a typical industrial environment, the train dataset is collected separately to prepare the model and then verified on a new dataset which is indeed the test dataset, otherwise, the model will be negatively affected, and its output will be a value that does not reflect the actual prediction ability of the model.
Eventually, there is another method to furtherly verify the validity of the model: the k-fold cross validation [43], it's a procedure in which different outputs, obtained from different dataset (ideally, they are all supposed to be approximately the same size), are collected and their results are averaged for all the evaluators, resulting in just one final output for each performance metric.

4.4. Optimisation of the model

Although the several approaches presented above for enhance the results of the ML algorithm there is still the need for a further phase for optimise the model, in this section some of these are going to be reviewed:

- *Early stopping:* during the training phase, the main objective is to reduce the loss function (i.e., errors), with this technique the train of the model stops when the evaluator stops improving [20];
- *Training process hyperparameter:* the hyperparameter is a parameter that control the learning process, and its effect may be enhanced through the so-called Cross Validation. Several iterations are performed during the training phase and then compared to define the best hyperparameter [20];
- *General optimisation:* during the implementation of the ML algorithm is possible to configure different parameters on different models enhancing the result of a model according to its specific requirement and needs;

4.5. Evaluation of the Machine Learning model

The decisive final task, it's obviously the evaluation of the model, or the evaluation between the models. Typically, this phase it has to be different according to which type of ML algorithm has been performed: classification models require different evaluation method then the ones

that can be used on the regressions. In this section both typologies are going to be reviews. Eventually, after the evaluation has been performed would be wise to step back to the optimisation phase to search if anything could have been done better and hence enhancing furtherly the algorithm.

4.5.1. Regression evaluations

The first to be reviewed are the most common: the regression ones, the main evaluators, according to most of the sources are: the MAE, Root Mean Squared Error (RMSE), R^2.
The MAE is one of the most common measures for errors in statistic, therefore its principle is simple: an arithmetic average of the absolute errors. Its equation is reported below this introduction:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{21}$$

Where, y_i is the prediction, x_i is the regressor and n is the total number of observations. Obviously, the lower is the MAE the better is the model, and generally it is considered a good indicator of the magnitude of errors. However, the RMSE gives more weight to large errors, highlighting hence systematic errors in the algorithm [1], so it may be more important in the evaluation. The formula RMSE is the following:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - x_i)^2}{n}} \tag{22}$$

Eventually, the R^2 or coefficient of determination, it is commonly considered as the most important evaluator in statistics: it determines the portion of variability that can be explained by the model, hence the higher R^2 the better is the model. The formula of the coefficient is the following:

$$R^2 = 1 - \frac{SS_{Res}}{SS_{Tot}} \tag{23}$$

Where SS_{Res} and SS_{Tot} are respectively the sum squares of the residual errors of the data model, and the total errors. All these evaluators can be easily implemented in any Python algorithm using the module Metrics of Scikit Learn, however the effectiveness of a regression model may be also verified graphically, as showed in the following image [*Figure 26*]:

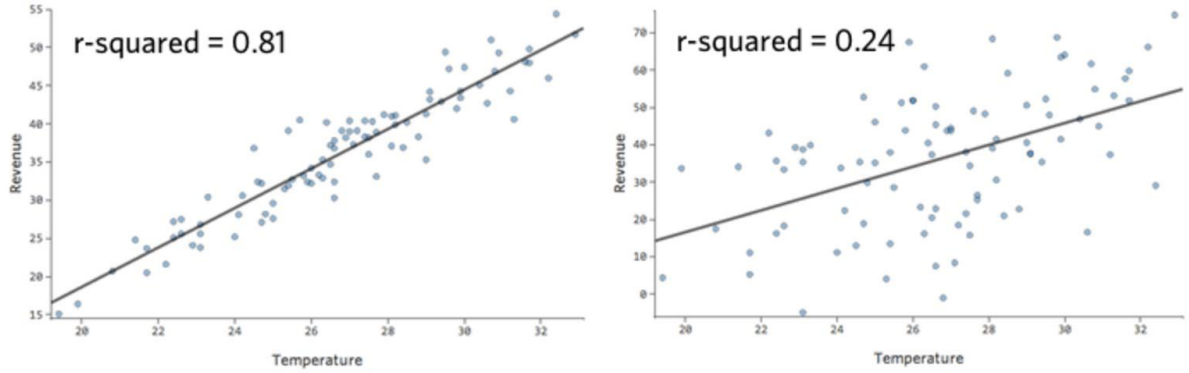

Figure 26: *two graphs showing different levels of quality;*

The graph on the left, portraits a situation where the model is a good solution: it detected the general pattern of the data and the R^2 is acceptable. On the other hand, the graph on the right has completely missed the pattern and the R^2 value is indeed very low. Moreover, both the graphs allow to determine whether there is a bias in the model: since the dots are equally distributed along the line of regression, the model is definitely noy biased.

Other solutions considered in the sources are: Relative Squared Error (RelSE), Normalised Root Mean Error (NRMSE), Mean Absolute Percentage Error (MAPE), Relative Absolute Error (RAE), that are not significantly different from the ones seen above, however any application should study what is the best evaluator that fits for the algorithm that is working on, since each of them has its pros and cons.

4.5.2. Classification evaluations

As already explained, the classification methods require a different form of evaluations since the different type of output (i.e., labelled). Typically, there are mainly three types of evaluation for classification algorithms: accuracy, precision, recall and F1. The first, the accuracy is simply the number of correct predictions over the total predictions:

$$ACC = \frac{Correct\ Predictions}{Total\ Predictions} \tag{24}$$

Although, the accuracy gives an idea of how accurate the model is, it provides no clue about where there is an error or how to correct it. A potential solution to this problem may be given by the precision evaluator: it considers the relationship between the true positive (the predictions labelled as '1' when they are actually '1'), it may be really useful to correct the problem of false positive. The formula is the following:

$$PREC = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{25}$$

A similar idea is the one of the recall (called also sensitivity) : it considers the number of true positive, over the number of both true positives and false negatives, hence the number of what is actually true. The formula is showed beneath;

$$REC = \frac{True\ Positives}{True\ Positives + False\ Negative} \tag{26}$$

Eventually, the F1, that is a combination between recall and precision, considering both true positive and false negative. Its formula is the following:

$$F1 = \frac{2 \cdot (Precision \cdot Recall)}{Precision + Recall} \tag{27}$$

That, similarly to the R^2, the higher the better; it can also be rewritten with a different β (normally is 1), to give a different importance between false negatives and true positives.
It is also possible to verify the correctness of the model through the so-called confusion matrix, that shows the number of measurement that were correct or wrong, allowing the user also to perceive the presence of any bias or misjudgement in the algorithm. The following image [*Figure 27*], depicts the composition of a confusion matrix that typically is applied on ML models of classification.

		Predicted	
		Negative **(N)** -	Positive **(P)** +
Actual	Negative -	True Negative **(TN)**	**False Positive (FP) Type I Error**
	Positive +	**False Negative (FN) Type II Error**	True Positive **(TP)**

Figure 27: *confusion matrix for a classification problem;*

The above image shows just two labels, but the confusion matrix can be applied on algorithms with more labels and in any case it would be possible to distinguish between the correct prediction and the various type of errors that may occur.
Just as the regression evaluators, all of these techniques are naturally implemented in the metrics module of Scikit Learn.

4.6. Confront of two ML outputs

In this section, two ML are going to be analysed according to highlight the differences between them, and depicting the features that highlights the characteristics of the two models.

4.6.1. Implementation and evaluation of tool wear algorithm

There are numerous examples of ML algorithms applied on tool wear predictions, one of the most detailed is the one provided by E. Traini *et al.* [43], they applied several ML models for the prediction of flank wear (measured with VB) on milling [*Figure 28*]. A milling machine is a rotating tool with multiple cutting edges that gradually remove material from the surface of the workpiece, that gradually advance on the worktable: hence, knowing how the machine works, it would be possible to already start working on the features to be used in the ML, such as feed rate, cutting speed and spindle speed.

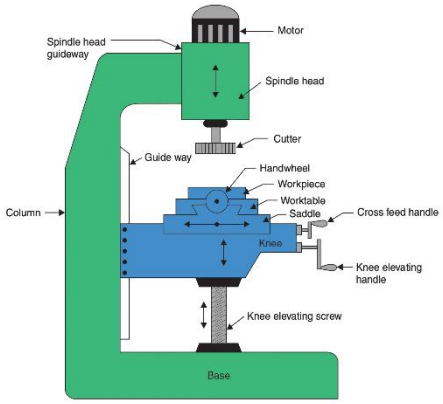

Figure 28: *milling machine illustration;*

The dataset used for this article was obtained experimentally, and the model has the peculiarity of combine both regression and classification algorithms, as showed in the image aside [*Figure 29*]. Initially the dataset is collected through analogical sensor and then manipulated during the pre-processing phase for the generation of the features and obviously, both normalisation and transformation of the data in order to be suitable for the processing itself, that is computed considering the training, the testing, and the evaluation phase.

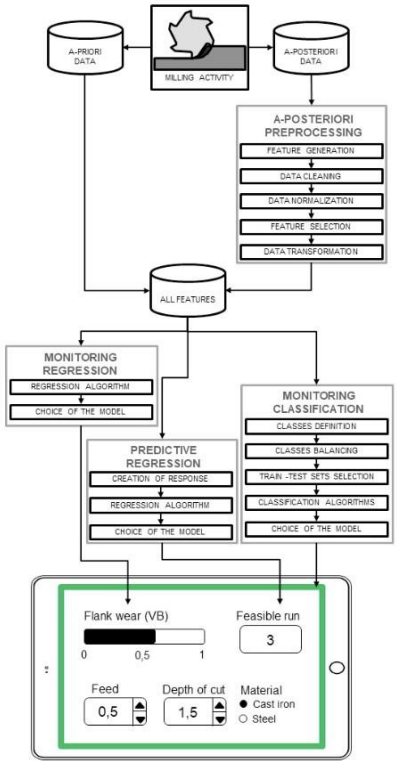

Figure 29: *framework proposed by [43];*

A first '*monitoring*' regression model and a classification one is supposed to determine whether the tool has to be replaced or otherwise, a second regression '*predictive*', instead, has to compute the RUL, hence for the couple the target variable is the VB and for the second regression is, indeed, the RUL.

The models applied for the regression are: LR, RF, Bayesian Linear Regression (BLR), DeT and NN, evaluated with RMSE, Relative Squared Error (RelSE) and R^2.
On the other hand, for the classification are chosen: LoR, RF, DeT, NN and Decision Jungle (DJ), evaluated with ACC, and the percentages of correct responses SP and WP.
To enhance the results given by the ML algorithm has been used the hyperparameter Tuning method to match the optimal value for each model.
The results obtained by the model above are showed in the tables [*Table 5*][*Table 6]* below:

Table 5: *results obtained for the two regressions in [43];*

	$RMSE_{VB}$	$RelSE_{VB}$	R^2_{VB}	$RMSE_{RUL}$	$RelSE_{RUL}$	R^2_{RUL}
LR	0.110	0.182	0.817	1.671	0.178	0.822
RF	0.123	0.225	0.781	1.517	0.134	0.866
BLR	0.116	0.194	0.813	1.640	0.174	0.826
DeT	0.122	0.218	0.794	1.615	0.156	0.844
NN	0.110	0.179	0.821	0.581	0.022	0.979

Table 6: *results obtained for classification in [43];*

	SP	**WP**	**ACC**
LoR	0.960	0.9	0.941
RF	0.936	0.917	0.930
BLR	0.952	0.917	0.941
DeT	0.960	0.950	0.957
NN	0.936	0.9	0.924

As clearly showed, the best model for the classification is the DT and the NN for both regressions, since the results obtained in the R^2.
Typically, a table is the main way to visualise and confront the results, since it is clean and immediate to read and understand. However other sources showed the results on the graphs, and it may be a solution even more feasible to find a systematic error (i.e., an error om higher value of the target). The following image shows the output visualised through a plot [*Figure 30*].

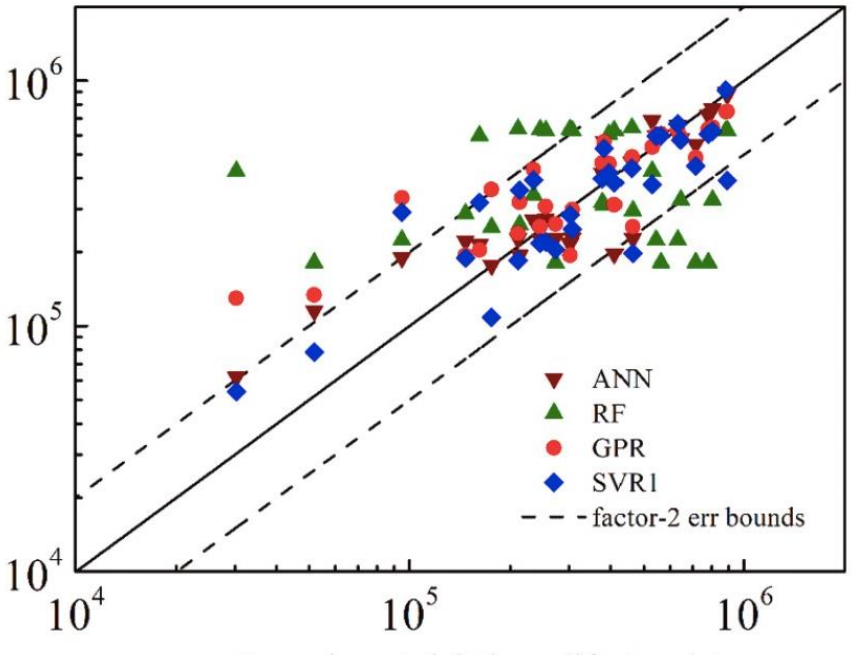

Figure 30: *ML model output displayed in a graph from [7];*

The graph clearly shows the good results obtained by both ANN and SVR, however a table for reporting the values obtained by the evaluators is necessary in any case. The graph also shows the bounds of the error, beyond these lines the prediction is not acceptable.

4.6.2. Implementation end evaluation of quality control algorithm

This subsection, instead, is going to review an implementation of a quality control algorithm for ML, the work of S. Shorr *et al.* [16] is a good representation of the scope of the algorithm. In the article are analysed drilled and reamed bores in an early stage of machining (milling machine), the algorithm indeed should help improving the process planning, avoiding waste and guarantee the quality of the workpiece. The dataset was obtained by an industry in Germany, producing hydraulic valves to be assembled as shown in the following picture [*Figure 31*]:

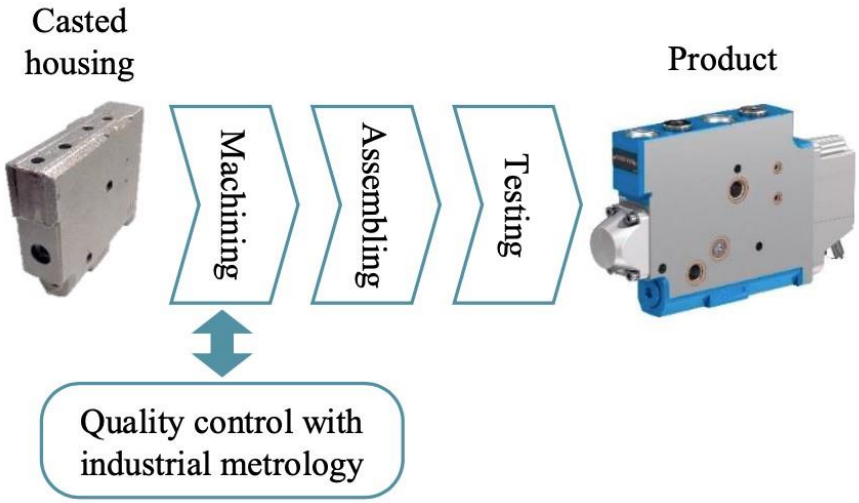

Figure 31: *final product and the phase where the ML analysis is performed from [16];*

As may be seen, firstly the housing is machined with a drilling machine and eventually assembled in the final product. The quality of the bores of the housing are defined by several parameters: force, torque, power, vibration, or acoustic emission that may also be applied for the detection of tool wear that is itself a feature for the quality of the bores during machining. To reduce the amount of data to be analysed and hence enhancing the prediction itself, a selection of the features through correlation matrix [*Figure 32*] is performed. The standard deviation, the mean, skewness, kurtosis, minimum and maximum value are computed for the spindle and the z-axis, and the feature selected are the ones with the highest correlation (i.e., intensity of green). Moreover, the targets to be determined are identified in the 'concentricity' and the 'diameter', values that strictly correlate to the quality of the product. The model to be used initially was decided prior to the analysis, using a test dataset, and resulted in the RF to be the most accurate (lower error) [*Figure 33*]:

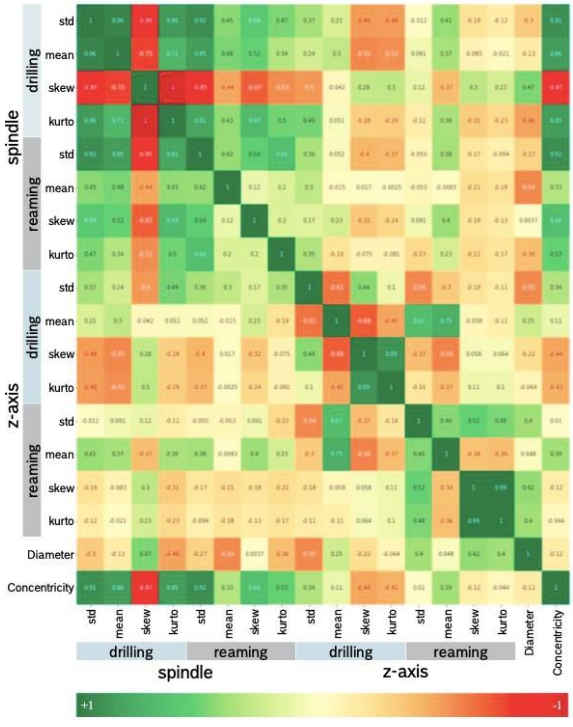

Figure 32: *correlation matrix from [16];*

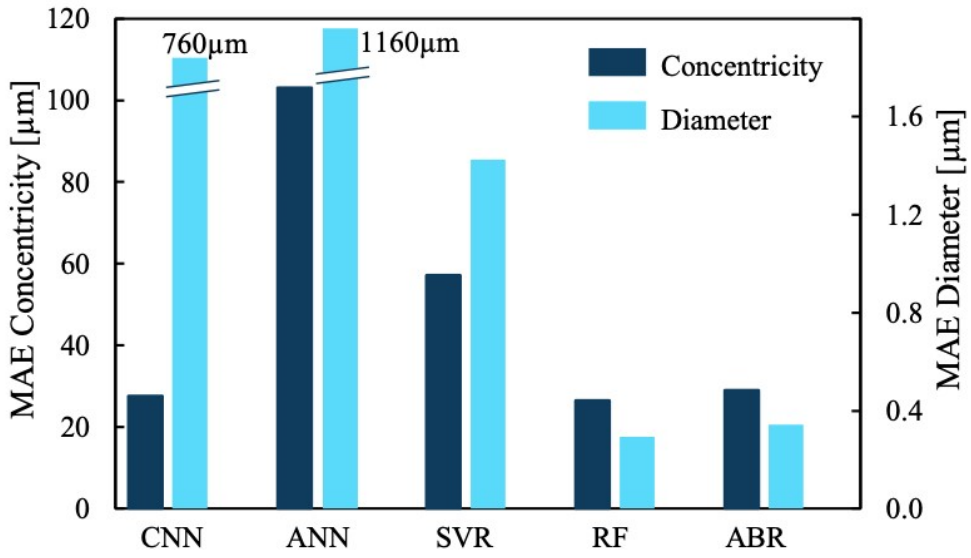

Figure 33: *graphical output of the initial result in [16];*

Accordingly, RF is indeed the model with the lowest MAE for both diameter and concentricity, while CNN and ANN are dramatically inaccurate for the diameter. Eventually, the RF was the only model applied in the algorithm, it had to compute the predictions of both the targets, determining eventually if the quality boundaries are complied. The following graphs show the learning curves of the RF over the testing phase [*Figure 34*]:

Where, the training dataset size was 85% of the total dataset and the final statistical result are reported below [*Table 7*]:

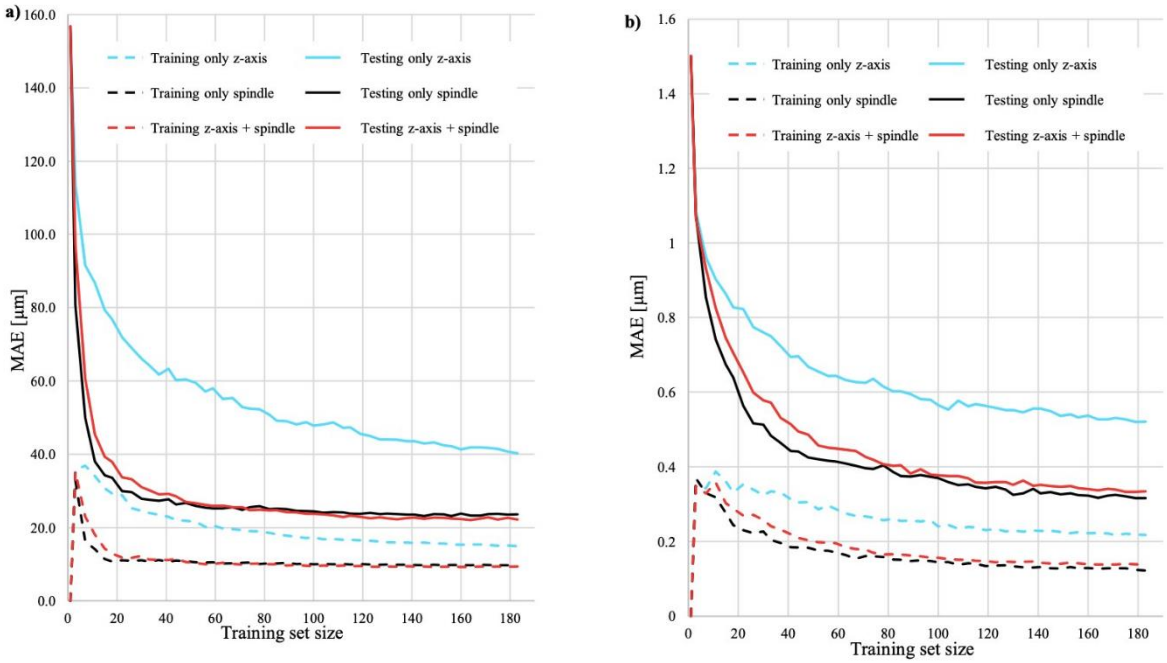

Figure 34: *graphs showing the learning curves of the RF, (a) of concentricity and (b) of diameter, from [16];*

Table 7: *results of [16] displayed in a table;*

	Concentricity	***Diameter***
MAE	17.1	0.27
MAPE	27.0	0.002
Maximum Error	83.0	0.72
R^2	96.3	94.1

The results show how the results (considering especially the R^2) are acceptable for both the target parameters, however it is not neglectable that the model could be improved furtherly considering the MAPE and MAE for the concentricity.

Moreover, studying the results with a graphical output [*Figure 35*] has helped the authors to understand the results of the prediction was influenced by the batch it was referring to: the third batch prediction, for example, had problems to correctly predict the most small and large diameter but the model could be still considered effective.

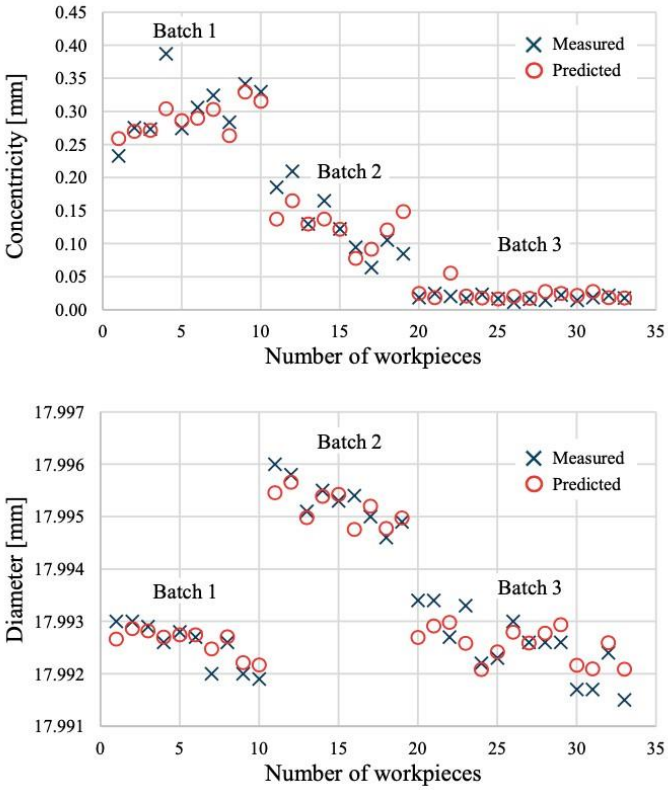

Figure 35: *graphical output obtained in [16], divided in the 3 batches in which each data has been collected;*

5. Example of a ML implementation

This chapter is going to reviews a simple classification ML model that has been computed using publicly available dataset. The study is going to show how the implementation of a classification algorithm can be performed to determine whether a CNC milling machine is worn or not, and how it should be executed practically and how to evaluate the results. The implementation of the algorithm is going to be explained in the following steps:

- *Dataset:* the dataset has to be correctly analysed in order to justify all the other choices that have been made, moreover it will also be provided a brief explanation of the CNC milling machine from which the dataset has been collected. Moreover, a description of the cross validation performed here is going to be provided as well;
- *Data normalisation:* the dataset is composed by several parameters that have to be analysed for the feature selection, to determine whether an actual correlation is present the data normalisation is a delicate operation that has to be performed;
- *Feature selection:* the features selection is a necessity in any ML algorithm since it has to provide to the model the parameters that have been identified to be correlated the most to the target feature. The model cannot be applied to the entire dataset since problems of perfect collinearity can arise from the execution of the model and therefore not all the features can be used for the application;
- *Models and Optimisation:* a description of the model that has been chosen for the analysis and the optimisation of the parameters that has been performed on them, moreover to maximising the effectiveness of the algorithm a procedure for K-Fold Cross-Validation has been implemented as well and here explained and justified;
- *Output:* eventually the output and an analysis of the results have to be performed, therefore finding which is the most efficient model;

5.1. Dataset

The dataset used are publicly available [116], they have been collected by the University of Michigan on a CNC Milling machine, and divided into 18 different experiments, all provided in .csv, they come with a 19^{th} file that represent the condition under each experiment has been

performed. In this section a general description of the machine and the dataset is going to be given, and how the importation/manipulation in Python is performed.

5.1.1. CNC milling machine

As already explained at the beginning of the chapter, the machine analysed is a CNC milling machine, which is basically a milling machine combined with Computer Numerical Control (CNC) [*Figure 36*]. The CNC is a technique of indirect control over the machine that, using a set of predefined instructions, is able to manipulate the object automatically and without any human intervention, hence the machine interacts only with the code that has to be converted into instructions (i.e., Cartesians coordinates), allowing the machine to be extremely precise while working [117].

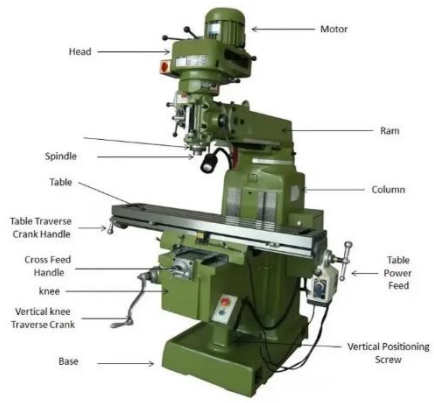

Figure 36: *CNC milling machine from [117];*

5.1.2. Dataset description

The total amount of dataset, one for each experiment conducted, is 18 and all of them are composed by the same features with the same dimensionality, the features that have been collected, have also been described by the source [116]:

- X1_ActualPosition: actual x position of part (mm);
- X1_ActualVelocity: actual x velocity of part (mm/s);
- X1_ActualAcceleration: actual x acceleration of part (mm/s/s);
- X1_CommandPosition: reference x position of part (mm);
- X1_CommandVelocity: reference x velocity of part (mm/s);
- X1_CommandAcceleration: reference x acceleration of part (mm/s/s);
- X1_CurrentFeedback: current (A);
- X1_DCBusVoltage: voltage (V);
- X1_OutputCurrent: current (A);
- X1_OutputVoltage: voltage (V);

- X1_OutputPower: power (kW);
- Y1_ActualPosition: actual y position of part (mm);
- Y1_ActualVelocity: actual y velocity of part (mm/s);
- Y1_ActualAcceleration: actual y acceleration of part (mm/s/s);
- Y1_CommandPosition: reference y position of part (mm);
- Y1_CommandVelocity: reference y velocity of part (mm/s);
- Y1_CommandAcceleration: reference y acceleration of part (mm/s/s);
- Y1_CurrentFeedback: current (A);
- Y1_DCBusVoltage: voltage (V);
- Y1_OutputCurrent: current (A);
- Y1_OutputVoltage: voltage (V);
- Y1_OutputPower: power (kW);
- Z1_ActualPosition: actual z position of part (mm);
- Z1_ActualVelocity: actual z velocity of part (mm/s);
- Z1_ActualAcceleration: actual z acceleration of part (mm/s/s);
- Z1_CommandPosition: reference z position of part (mm);
- Z1_CommandVelocity: reference z velocity of part (mm/s);
- Z1_CommandAcceleration: reference z acceleration of part (mm/s/s);
- Z1_CurrentFeedback: current (A);
- Z1_DCBusVoltage: voltage (V);
- Z1_OutputCurrent: current (A);
- Z1_OutputVoltage: voltage (V);
- S1_ActualPosition: actual position of spindle (mm);
- S1_ActualVelocity: actual velocity of spindle (mm/s);
- S1_ActualAcceleration: actual acceleration of spindle (mm/s/s);
- S1_CommandPosition: reference position of spindle (mm);
- S1_CommandVelocity: reference velocity of spindle (mm/s);
- S1_CommandAcceleration: reference acceleration of spindle (mm/s/s);
- S1_CurrentFeedback: current (A);
- S1_DCBusVoltage: voltage (V);
- S1_OutputCurrent: current (A);
- S1_OutputVoltage: voltage (V);

- S1_OutputPower: current (A);
- S1_SystemInertia: torque inertia (kg*m^2);
- M1_CURRENT_PROGRAM_NUMBER: number the program is listed under on the CNC;
- M1_sequence_number: line of G-code being executed;
- M1_CURRENT_FEEDRATE: instantaneous feed rate of spindle;
- Machining_Process: the current machining stage being performed. Includes preparation, tracing up and down the "S" curve involving different layers, and repositioning of the spindle as it moves through the air to a certain starting point;

Eventually, the datasets are bounded to a 19th file, called 'train', which is a summary of the characteristics of all the experiments, its features have also been described by [116], as follows:

- No : experiment number;
- material : wax (same for all the experiments);
- feed_rate : relative velocity of the cutting tool along the workpiece (mm/s);
- clamp_pressure : pressure used to hold the workpiece in position(bar);
- tool_condition : label for unworn and worn tools;
- machining_completed : indicator for if machining was completed without the workpiece moving out of the pneumatic in position;
- passed_visual_inspection: indicator for if the workpiece passed visual inspection, only available for experiments where machining was completed;

Obviously, the last three features have been collected after the experiments were completed, while the first four features have been collected prior to the experiments were conducted.

5.1.3. Implementation on Python

The following lines shows how the datasets have been loaded in the algorithm and how combine them with the train file [*Code Snippet 6*]:

```
def datatable():
   experiment_result = pd.read_csv("/Users/../Test/train.csv")
   experiment_result['passed_visual_inspection'] =
experiment_result['passed_visual_inspection'].fillna('no')
   df = pd.read_csv('/Users/../Test/experiment_01.csv')
   frames = []
   for i in range(1, 19):
```

```
        exp_num = '0' + str(i) if i < 10 else str(i)
        frame = pd.read_csv(f"/Users/../Test/experiment_{exp_num}.csv")
        exp_result_row = experiment_result[experiment_result['No'] == i]
        frame['exp_num'] = i
        frame['material'] = exp_result_row.iloc[0]['material']
        frame['feedrate'] = exp_result_row.iloc[0]['feedrate']
        frame['clamp_pressure'] = exp_result_row.iloc[0]['clamp_pressure']
        frame['tool_condition'] = exp_result_row.iloc[0]['tool_condition']
        frame['machining_finalized'] =
exp_result_row.iloc[0]['machining_finalized']
        frame['passed_visual_inspection'] =
exp_result_row.iloc[0]['passed_visual_inspection']

        frames.append(frame)
        df = pd.concat(frames, ignore_index=True)
    mapping_1 = {'Starting': 0,
                 'Prep': 1,
                 'Layer 1 Up': 2,
                 'Layer 1 Down': 3,
                 'Repositioning': 4,
                 'Layer 2 Up': 5,
                 'Layer 2 Down': 6,
                 'Layer 3 Up': 7,
                 'Layer 3 Down': 8,
                 'end': 9,
                 'End': 9}
    mapping_2 = {'worn': 0,
                 'unworn': 1}
    mapping_3 = {'no': 0,
                 'yes': 1}

    df['Machining_Process'].replace(mapping_1, inplace=True)
    df['tool_condition'].replace(mapping_2, inplace=True)
    df['passed_visual_inspection'].replace(mapping_3, inplace=True)
    df['machining_finalized'].replace(mapping_3, inplace=True)
    df = df.sample(frac=1)
    df.drop_duplicates()
    print(df)
    column_headers = list(df.columns.values)
    print("The Column Header :", column_headers)
```

```
return df, df_test
```

Code snippet 6: *implementation of the dataset in the algorithm;*

The code above is the function “datatable” that has been applied for this analysis, it has to resolve two different problems: merging 18 similar files and merge these 18 files with another one that is totally different. The solution that has been found is to, firstly, add as a feature the experiment number (“exp_num”) and then combine all the values present in “train” for that experiment to all the rows present in each experiment. The following is the output that has been obtained:

```
“The Column Header : ['X1_ActualPosition', 'X1_ActualVelocity',
'X1_ActualAcceleration', 'X1_CommandPosition', 'X1_CommandVelocity',
'X1_CommandAcceleration', 'X1_CurrentFeedback', 'X1_DCBusVoltage',
'X1_OutputCurrent', 'X1_OutputVoltage', 'X1_OutputPower',
'Y1_ActualPosition', 'Y1_ActualVelocity', 'Y1_ActualAcceleration',
'Y1_CommandPosition', 'Y1_CommandVelocity', 'Y1_CommandAcceleration',
'Y1_CurrentFeedback', 'Y1_DCBusVoltage', 'Y1_OutputCurrent',
'Y1_OutputVoltage', 'Y1_OutputPower', 'Z1_ActualPosition',
'Z1_ActualVelocity', 'Z1_ActualAcceleration', 'Z1_CommandPosition',
'Z1_CommandVelocity', 'Z1_CommandAcceleration', 'Z1_CurrentFeedback',
'Z1_DCBusVoltage', 'Z1_OutputCurrent', 'Z1_OutputVoltage',
'S1_ActualPosition', 'S1_ActualVelocity', 'S1_ActualAcceleration',
'S1_CommandPosition', 'S1_CommandVelocity', 'S1_CommandAcceleration',
'S1_CurrentFeedback', 'S1_DCBusVoltage', 'S1_OutputCurrent',
'S1_OutputVoltage', 'S1_OutputPower', 'S1_SystemInertia',
'M1_CURRENT_PROGRAM_NUMBER', 'M1_sequence_number', 'M1_CURRENT_FEEDRATE',
'Machining_Process', 'exp_num', 'material', 'feedrate', 'clamp_pressure',
'tool_condition', 'machining_finalized', 'passed_visual_inspection']”
```

Moreover, the function has to solve some data transformation problems as explained in [43]: the features “tool_condition”, “machining_finalized”, “passed_visual_inspection” and “Machining_Process” are all features where the values are strings, hence they have to converted in values. To do so the “dictionary” method has been applied: for each string present in the column, a correspondent numerical value has been associated with the dictionaries (called “mapping_i”) and modified in the dataframe through the method “.replace” native of Pandas. Eventually, the function removes the duplicates present in the database (df.drop.duplicates()) and randomise the order of the row present in the database

(df.sample) in order to don't compromise the training phase that is going to be performed in the other functions present in the algorithm.
As explained in the last chapter, the test and train dataset should be separated since the beginning and they cannot be merged and divided any after, hence the best solution is to divide already the dataset, resulting in two different return for the function showed above.

5.2. Data normalisation and cleaning

The following lines of Python illustrate the implementation of data cleaning and normalisation in the application prepared for this work [*Code snippet 7*]:

```
def cleaning(test, train):
    df, df_test = datatable(test, train)
    to_drop = []
    to_drop_2 = []
    for col in df.columns:
        if len(df[col].unique()) == 1:
            to_drop.append(col)
    df.drop(to_drop, axis=1, inplace=True)
    for col in df.columns:
        if len(df[col].unique()) == 1:
            to_drop_2.append(col)
    df.drop(to_drop_2, axis=1, inplace=True)
    cols_to_norm = ['X1_ActualPosition', 'X1_ActualVelocity',
'X1_ActualAcceleration', 'X1_CommandPosition',
                    'X1_CommandVelocity', 'X1_CommandAcceleration',
'X1_CurrentFeedback', 'X1_DCBusVoltage',
                    'X1_OutputCurrent', 'X1_OutputVoltage',
'X1_OutputPower', 'Y1_ActualPosition', 'Y1_ActualVelocity',
                    'Y1_ActualAcceleration', 'Y1_CommandPosition',
'Y1_CommandVelocity', 'Y1_CommandAcceleration',
                    'Y1_CurrentFeedback', 'Y1_DCBusVoltage',
'Y1_OutputCurrent', 'Y1_OutputVoltage', 'Y1_OutputPower',
                    'Z1_ActualPosition', 'Z1_ActualVelocity',
'Z1_ActualAcceleration', 'Z1_CommandPosition',
                    'Z1_CommandVelocity', 'Z1_CommandAcceleration',
                    'S1_ActualPosition', 'S1_ActualVelocity',
```

```
                    'S1_ActualAcceleration', 'S1_CommandPosition',
'S1_CommandVelocity', 'S1_CommandAcceleration',
                    'S1_CurrentFeedback', 'S1_DCBusVoltage',
'S1_OutputCurrent', 'S1_OutputVoltage', 'S1_OutputPower',
                    'M1_sequence_number', 'M1_CURRENT_FEEDRATE',
                    'Machining_Process', 'feedrate', 'clamp_pressure',
                    'machining_finalized', 'passed_visual_inspection']
    df[cols_to_norm] = MinMaxScaler().fit_transform(df[cols_to_norm])
    df.dropna(axis=0, how="any", subset=['tool_condition'], inplace=True)
    df = df.fillna("", inplace=False)
    df_test[cols_to_norm] =
MinMaxScaler().fit_transform(df_test[cols_to_norm])
    df_test.dropna(axis=0, how="any", subset=['tool_condition'],
inplace=True)
    df_test = df_test.fillna("", inplace=False)
    print(df)
    labels = ['not worn', 'worn']
    df['tool_condition'].value_counts().plot(kind='pie', labels=labels)
    print(df_test)
    return df, df_test
```

Code snippet 7: *data cleaning function and normalisation;*

The function above is the one used to clean the datasets and normalise the values, firstly the datatable function is deployed to import the databases, and all the column that are not completes are removed, then a list containing the name of the columns of the database that have to be normalised, note that all the column are selected for the normalisation but “tool_condition”, since the target of the analysis and if normalised it would become a continuous variable and no longer a categorical one. The normalisation is performed with the method MinMaxScaler(), from the package “Preprocessing” of SciKit Learn, and it applies a scaling normalisation to all columns that have been selected in a range between 0 and 1, on the basis of what are the

maximum and the minimum for each parameter. Furthermore, the function proceeds dropping all the rows that have no value in "tool_condition" since useless.

Eventually, the function counts and plot the number of worn and unworn rows are present in the train dataframe: an excessive number on a side, or the other, might lead to a biased prediction. The result of this final part of the code, however, is plotted for each iteration like in the first one as showed aside [*Figure 37*] in order to consider any bias present in the testing dataset. Therefore, there is no need to have a test dataset unbiased, since the algorithm should be able to determine in both cases.

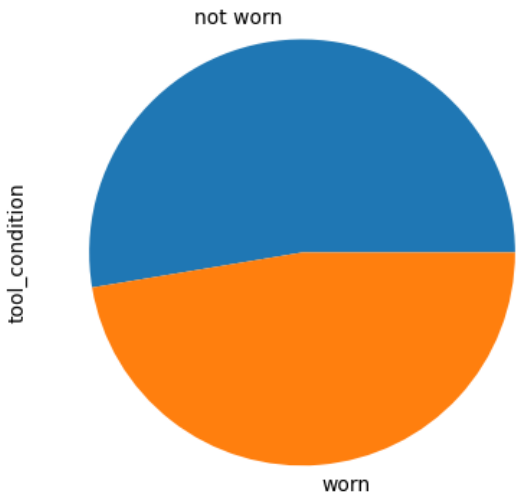

Figure 37: *pie-chart of the number of worn and not worn rows in the dataset of [116];*

5.3. Selection of the features

The following task is the selection of the features, that has been performed using the heat matrix applied on the train database, the following lines represents the lines [*Code snippet 8*] to obtain the correlation matrix and then, the output is showed [*Figure 38*]

```
df, df_test = cleaning(test, train)
fig, ax = plt.subplots(figsize=(40, 40))
sns.heatmap(df.corr(), annot=True, cmap='Blues', ax=ax)
plt.show()
```

Code snippet 8: *heatmap/correlation matrix deployment;*

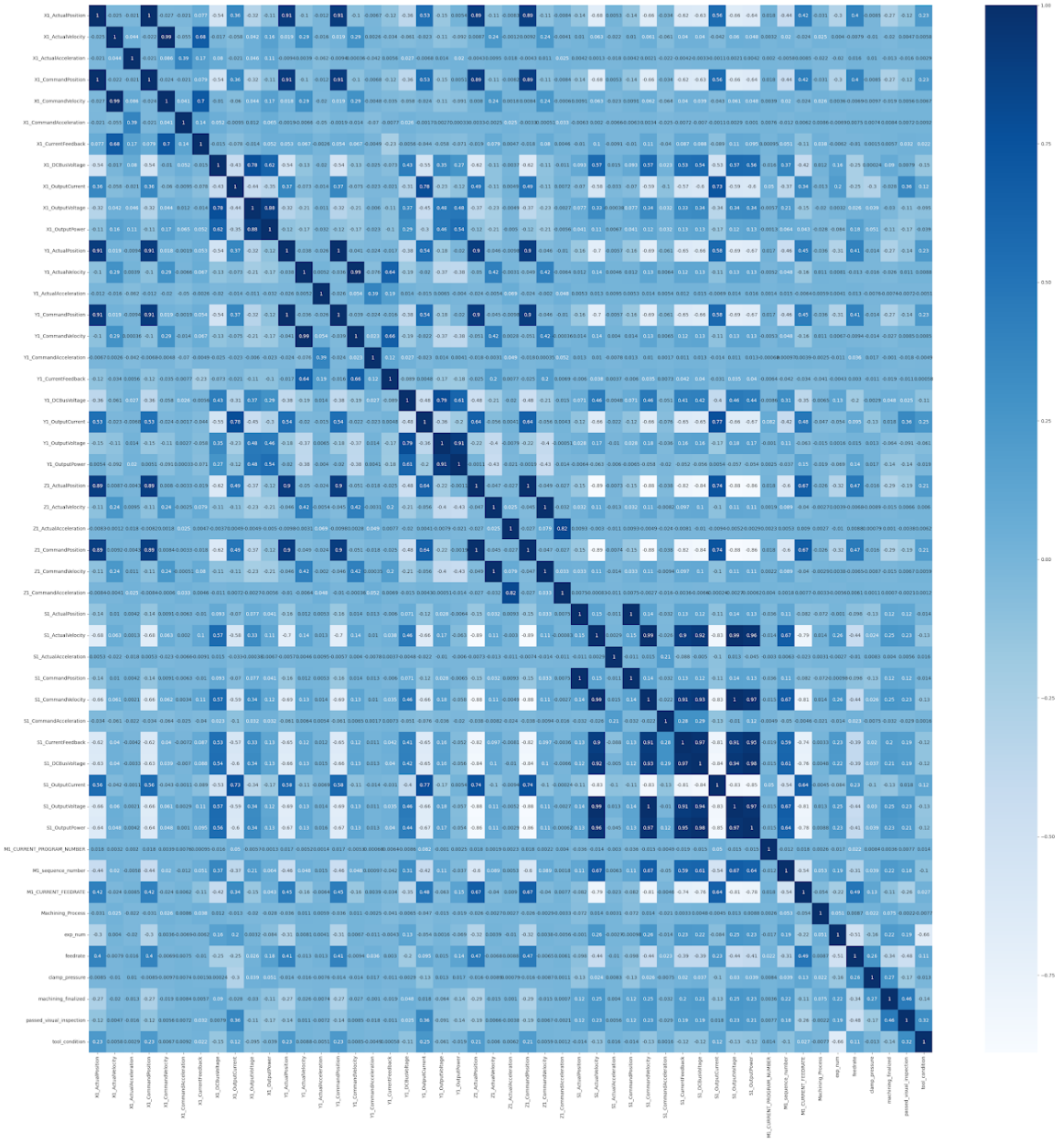

Figure 38: *heatmatrix resulting of the dataset;*

The target feature, “tool_condition” on the last row, has to be checked on each column to understand which are the target that may be considered for the regression. Accordingly, all the features with a higher correlation then 0.05 are hence chosen for the prediction, the feature “passed_visual_inspection” and “num_exp”, even though are related are the target they are not going to be considered for the prediction since they have been evaluated correlated because the construction of the dataset and hence their presence would just falsify

the results of the prediction. Therefore, the features that are going to be considered for the prediction are the following:

```
['X1_ActualPosition', 'X1_CommandPosition',
             'X1_DCBusVoltage',
             'X1_OutputCurrent', 'Y1_ActualPosition',
             'Y1_CommandPosition',
             'Y1_DCBusVoltage', 'Y1_OutputCurrent', 'Y1_OutputVoltage',
             'Z1_ActualPosition', 'Z1_CommandPosition',
             'S1_ActualVelocity',
             'S1_ActualAcceleration', 'S1_CommandVelocity',
             'S1_CurrentFeedback', 'S1_DCBusVoltage', 'S1_OutputCurrent',
'S1_OutputVoltage', 'S1_OutputPower',
             'M1_sequence_number', 'feedrate', 'clamp_pressure']
```

5.4. Classification models

The models that have been chosen are here showed, presented with the parameters that can be modified and their corrispondant default value:

- Random Forest: it can be imported by SciKit Learn as RandomForestClassifier, and the parameters that can be modified are the following: (*n_estimators=100*, *, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='sqrt'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*);
- K-Nearest-Neighbour: it is imported form Scikit Lean as neighbours.KNeighboursClassifier(), it can be modified in the following parameters: (*n_neighbors=5*, *, *weights='uniform'*, *algorithm='auto'*, *leaf_size=30*, *p=2*, *metric='minkowski'*, *metric_params=None*, *n_jobs=None*);
- Decision Tree: it is imported as DecisionTreeClassifier(), can be modified in (*, *criterion='gini'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *class_weight=None*, *ccp_alpha=0.0*);

- Logistic Regression: imported as LogisticRegression, is the only one that is not properly a classifier, but it is often applied in this role, it can be modified as follows (*penalty='l2'*, *, *dual=False*, *tol=0.0001*, *C=1.0*, *fit_intercept=True*, *intercept_scaling=1*, *class_weight=None*, *random_state=None*, *solver='lbfgs'*, *max_iter=100*, *multi_class='auto'*, *verbose=0*, *warm_start=False*, *n_jobs=None*, *l1_ratio=None*);
- SVM: it is imported as svm.SVC, it's the classification variant of the SVM as explained in the third chapter, and it can be modified in: (*, *C=1.0*, *kernel='rbf'*, *degree=3*, *gamma='scale'*, *coef0=0.0*, *shrinking=True*, *probability=False*, *tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=-1*, *decision_function_shape='ovr'*, *break_ties=False*, *random_state=None*);
- Naïve Bayes: imported as naive_bayes.GaussianNB, it is a gaussian variation of the Naïve Bayes model and it can be modified in: (*, *priors=None*, *var_smoothing=1e-09*);
- ANN: imported as neural_network.MLPClassifier, it is the Python most common method for a neural network implementation, its parameters are: (*hidden_layer_sizes=(100,)*, *activation='relu'*, *, *solver='adam'*, *alpha=0.0001*, *batch_size='auto'*, *learning_rate='constant'*, *learning_rate_init=0.001*, *power_t=0.5*, *max_iter=200*, *shuffle=True*, *random_state=None*, *tol=0.0001*, *verbose=False*, *warm_start=False*, *momentum=0.9*, *nesterovs_momentum=True*, *early_stopping=False*, *validation_fraction=0.1*, *beta_1=0.9*, *beta_2=0.999*, *epsilon=1e-08*, *n_iter_no_change=10*, *max_fun=15000*);

These models have been chosen according to what has been studied in the third chapter, in the section of the classification, in order to show a complete reviews about these algorithms, that therefore, they are also supposed to be the most common in any ML implementation.
Each of these models has been implemented as follows [*Code snippet 9*]:

```
from sklearn.linear_model import LogisticRegression
import numpy as np
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn import neighbors, naive_bayes
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
```

```
def random_forest(df, df_test):
   X_train = df[['X1_ActualPosition', 'X1_CommandPosition',
                 'X1_DCBusVoltage',
                 'X1_OutputCurrent', 'Y1_ActualPosition',
                 'Y1_CommandPosition',
                 'Y1_DCBusVoltage', 'Y1_OutputCurrent', 'Y1_OutputVoltage',
                 'Z1_ActualPosition', 'Z1_CommandPosition',
                 'S1_ActualVelocity',
                 'S1_ActualAcceleration', 'S1_CommandVelocity',
                 'S1_CurrentFeedback', 'S1_DCBusVoltage',
'S1_OutputCurrent', 'S1_OutputVoltage', 'S1_OutputPower',
                 'M1_sequence_number', 'feedrate', 'clamp_pressure']]
   y_train = df["tool_condition"]
   X_test = df_test[['X1_ActualPosition', 'X1_CommandPosition',
                     'X1_DCBusVoltage',
                     'X1_OutputCurrent', 'Y1_ActualPosition',
                     'Y1_CommandPosition',
                     'Y1_DCBusVoltage', 'Y1_OutputCurrent',
'Y1_OutputVoltage',
                     'Z1_ActualPosition', 'Z1_CommandPosition',
                     'S1_ActualVelocity',
                     'S1_ActualAcceleration', 'S1_CommandVelocity',
                     'S1_CurrentFeedback', 'S1_DCBusVoltage',
'S1_OutputCurrent', 'S1_OutputVoltage', 'S1_OutputPower',
                     'M1_sequence_number', 'feedrate', 'clamp_pressure']]
   y_test = df_test["tool_condition"]
   model = RandomForestClassifier()
   model.fit(X_train, y_train)
   y_pred = model.predict(X_test)
   return y_test, y_pred
```

Code snippet 9: *implementation of a model in the algorithm;*

The two datasets are imported in the function and two subset for each are obtained for splitting each dataset to a Y and X for testing and for training, that is going to be fitted in the model, called RandomForestClassifier() in this case. The function is therefore yet incomplete since it is going to be modified in the following section for the optimisation

5.5. Optimisation of the models

Eventually, there is the optimisation of the models phase, where for each model the parameters are selected in order to have the best possible output. For this application the GridSearch method has been applied, it is one of the most common hyperparameter tuning methods, hence the implementation of the model is modified as follows [*Code snippet 10*]:

```
from sklearn.linear_model import LogisticRegression
import numpy as np
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn import neighbors, naive_bayes
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier



def random_forest(df, df_test):
   param_grid = {'min_samples_split': [7, 14],
                 'max_features': [5, 10, 20],
                 'criterion': ['gini', 'entropy']}
   X_train = df[[...]]
   y_train = df["tool_condition"]
   X_test = df_test[[...]]
   y_test = df_test["tool_condition"]
   model = RandomForestClassifier(max_depth=None, min_samples_split=7,
n_estimators=400, criterion='gini',
                                  max_features=5)
   # grid_search = GridSearchCV(model, param_grid=param_grid, cv=3,
n_jobs=8)
   # grid_search.fit(X_train, y_train)
   # print('Best parameters:', grid_search.best_params_)
   model.fit(X_train, y_train)
   y_pred = model.predict(X_test)
   return y_test, y_pred
```

Code snippet 10: *variation of implementation of the model with hyperparameter search;*

The param_grid list, contains the parameters that have to tuned to maximise the optimisation of the model. As the program runs the above algorithm return a string with the optimal

parameters that have to be inserted in the model manually, therefore after this procedure these lines may be converted into comment since they would just increase the time for the algorithm to complete. For the same reason, just 4 parameters have been chosen for tuning but theoretically all the parameters seen in the last section may be added.
The following lines show the optimisation of each model present in the algorithm [*Code snippet 11*]:

```
model = RandomForestClassifier(max_depth=None, min_samples_split=7,
n_estimators=400, criterion='gini',
                               max_features=5)
model = neighbors.KNeighborsClassifier(n_neighbors=8, weights='distance')
model = DecisionTreeClassifier(criterion='log_loss', max_depth=None,
min_samples_split=2, splitter='best')
model = LogisticRegression(penalty=None, solver='saga', max_iter=10000)
model = SVC(probability=True, shrinking=True, kernel='rbf')
model = naive_bayes.GaussianNB(var_smoothing=0.1873817422860384)
model = MLPClassifier(max_iter=1000, hidden_layer_sizes=13, random_state=9)
```

Code snippet 11: *results of the parameter tuning for each method;*

5.6. Outputs and evaluations

In this last section are provided the outputs and the evaluations that have been obtained in this work, moreover each solution is going to be analysed and commented.
As already explained in the other sections, in this application a procedure for the K-Fold Cross Analysis has been implemented in the algorithm: the 18 experiments are inserted in a different database for each of the three iteration that has been performed, the following lines show the subdivision of the dataset over the three experiments [*Code snippet 12*]:

```
test_1 = [12, 16, 17, 18]
train_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15]
test_2 = [1, 2, 6, 9]
train_2 = [3, 4, 5, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18]
test_3 = [8, 9, 11, 12]
train_3 = [1, 2, 3, 4, 5, 6, 7, 10, 13, 14, 15, 16, 17, 18]
```

Code snippet 12: *subdivision of the datasets over the experiments;*

The subdivision has been thought to test the algorithm under a similar amount of worn and unworn rows; the results are hence averaged in order to provide a definitive result for each evaluation metric.

What follow are two functions that have been used for the production of the output [*Code snippet 13*][*Code snipper 14*]:

```
def final_evaluation(evaluations, method, color):
    data = {'Random Forest': evaluations[0], 'KNN': evaluations[1],
            'Decision Tree': evaluations[2], 'Logistic Regression':
evaluations[3], 'SVM': evaluations[4],
            'Naive Bayes': evaluations[5], 'ANN': evaluations[6]}
    Names = list(data.keys())
    values = list(data.values())
    fig = plt.figure(figsize=(10, 5))
    plt.bar(Names, values, color=color,
            width=0.4)
    plt.xlabel("Methods")
    plt.ylabel(method)
    plt.title("Evaluation of the methods selected")
    plt.show()
```

Code snippet 13: *function for plotting the outputs;*

```
def evaluator(y_test, y_pred, evaluations, evaluations_r, method):
    precision = metrics.precision_score(y_test, y_pred)
    accuracy = metrics.accuracy_score(y_test, y_pred)
    recall = metrics.recall_score(y_test, y_pred)
    f1 = metrics.f1_score(y_test, y_pred)
    print(pd.DataFrame([precision, accuracy, recall, f1],
index=['Precision', 'Accuracy', 'Recall', 'F1'],
                       columns=[method]))
    evaluations.append(accuracy)
    evaluations_r.append(f1)
    return evaluations, evaluations_r
```

Code snippet 14: *function for producing the evaluation of each model;*

As may be seen in the second function, the evaluator used for this application have been: F1, precision, accuracy, and recall, according to what has been seen in the third chapter. Each time a model is applied in the algorithm, the evaluator function is also applied, that return a table with the evaluation of each model, the final evaluation function is triggered as well to append the values of the evaluation to a list that is needed for plotting the following results [*Figure 39*][*Figure 40*]

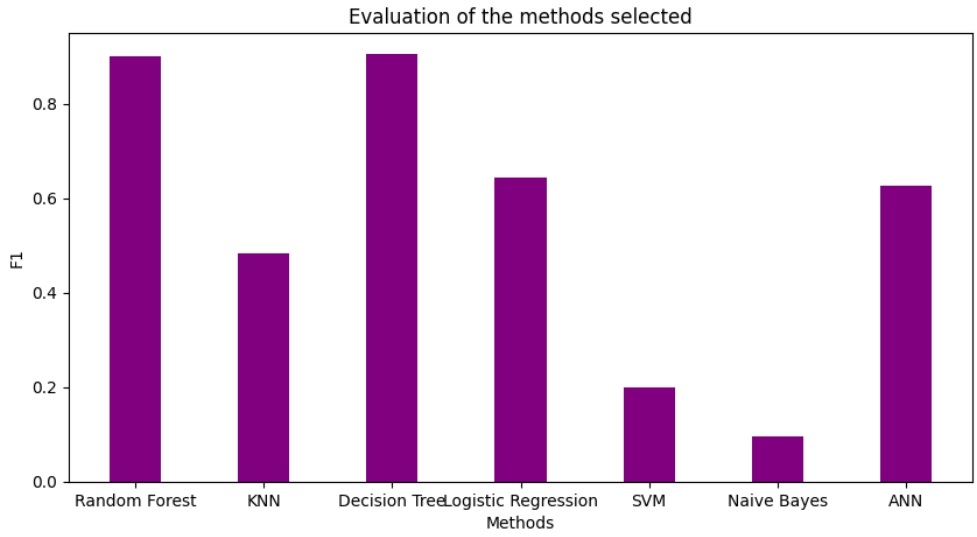

Figure 39: *bar chart plotting the F1 results of the algorithm for each model;*

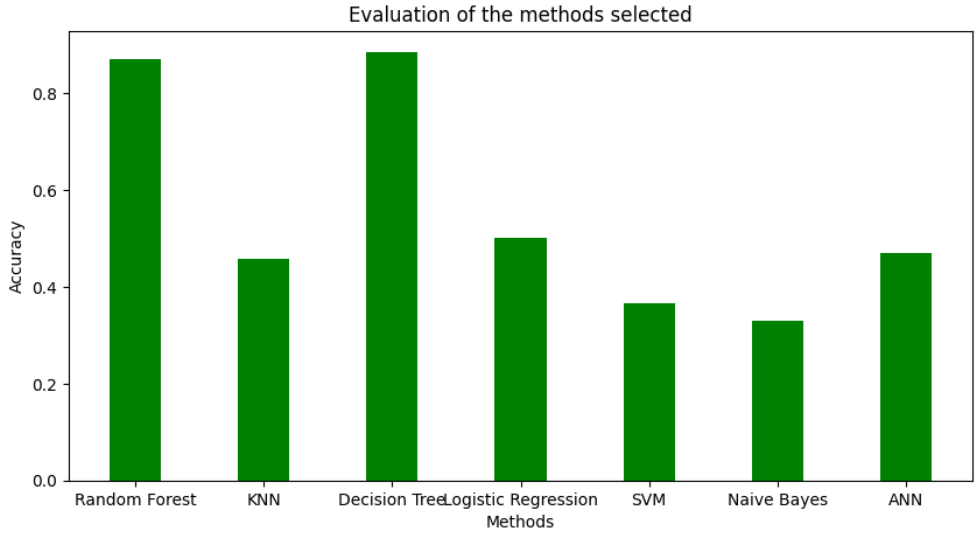

Figure 40: *bar chart plotting the F1 results of the algorithm for each model;*

As may be seen, the RF and the DT have both obtained good results, and ANN and LoR are mediocre but can still be considered acceptable, KNN and SVM are highly inefficient, and the NB just did not detect the pattern, however more detailed comment and evaluation are provided in the following subsections.
Eventually, below this paragraph it has been provided the function that has been implemented for the whole procedure that has been seen in the last sections [*Code snippet 15*]

```
def plotter(test, train):
    evaluations = []
    evaluations_r = []
    df, df_test = cleaning(test, train)
    fig, ax = plt.subplots(figsize=(40, 40))
    sns.heatmap(df.corr(), annot=True, cmap='Blues', ax=ax)
    plt.show()
    y_test, y_pred = random_forest(df, df_test)
    cm = confusion_matrix(y_test, y_pred, labels=[0, 1])
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0,
1])
    plot('Random Forest', disp)
    evaluations, evaluations_r = evaluator(y_test, y_pred, evaluations,
evaluations_r, "Random Forest")
```

Code snippet 15: *implementation of model, evaluation, and confusion matrix in the algorithm;*

This procedure is the same for the all the other models, so it would be pointless to show their lines as well. Initially two lists are created (void) to be used in the evaluator function, then the function of cleaning the database is triggered and it returns the two databases (train and test), normalised, to this function. The results obtained in the model function are called and plotted in the confusion matrix. Eventually the function evaluator is triggered that append the evaluations to the lists declared at the begin of the function, the function therefore proceeds this way for each models that has been seen above.
The following subsections, as already mentioned, will briefly discuss the results of each model and show all the other result and other possible optimisations.

5.6.1. Random Forest

The RF is resulted to be the second-best model for this application, and for what can be seen in the confusion matrix aside [*Figure 41*], the model is balanced and provided acceptable results. However, it also shows a bias in the output for the second type error, that are five times bigger then the ones of first type, and the overall result is definitely worse than the DeT.

The following table, shows the exact value of the results obtained by the model [*Table 8*]:

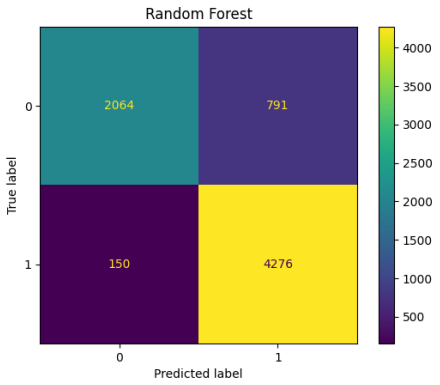

Figure 41: *confusion matrix of the RF in the first iteration;*

Table 8: *results of the RF model;*

Precision	***Accuracy***	***Recall***	***F1***
0.841	0.869	0.965	0.899
0.762	0.645	0.645	0.640
0.989	0.749	0.683	0.808
0.864	**0.754**	**0.733**	**0.782**

The last row reports the averaged results of the three rows above, and these numbers have to be considered as the results, it has been chosen to report also the other values in order to investigate any further problematic in the dataset or in the model.

As may be seen, the results obtained are substantially good for all the iteration performed, moreover the table shows also that the bias toward the second type error is not systematic since the recall is strangely different only in the first iteration while in the other experiments the results are lower, however the bias may be toward the first type errors as showed by the precision in the third iteration.

Eventually, the RF is indeed the second better choice for the algorithm however the gap between the DeT is important, on the other hand a higher level of tuning of the parameters that may reveal a better choice for the analysis.

5.6.2. KNN

The KNN obtained what can be defined an inefficient result: it did not detect the pattern and definitely it suffers of a huge bias toward the first type error as it has been depicted in the confusion matrix [*Figure 42*], a further analysis may be obtained consulting the table beneath shows the overall results obtained by the KNN in the algorithm [*Table 9*]:

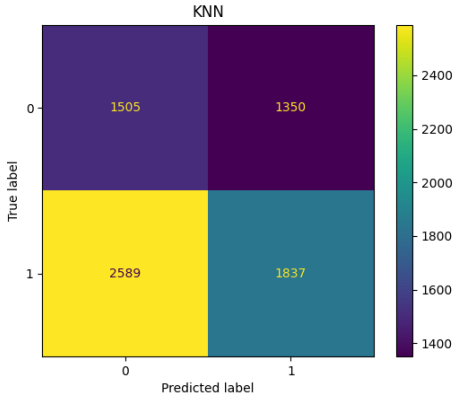

Figure 42: *confusion matrix of the KNN in the first iteration;*

Table 9: *results of the KNN model;*

Precision	***Accuracy***	***Recall***	***F1***
0.576	0.459	0.415	0.482
0.403	0.335	0.336	0.367
0.777	0.373	0.265	0.395
0.585	**0.389**	**0.339**	**0.415**

As already mentioned above, this model suffers of a bias toward the first type error as showed by the precision parameter, that is still high for all the experiments that have been performed. The optimisation has been performed on most of the parameters, hence it's hard to say that a better optimisation will improve somehow the model that on the basis of the table above is just inadequate to this dataset.

5.6.3. Decision Tree

The DeT obtained the better results of all the other models for all the experiments, and especially better than the ones obtained by the RF, even though this latter is supposed to be a "correction" of the DeT and looking at the confusion matrix [*Figure 43*], it appears also as it is not suffering relevant bias toward one

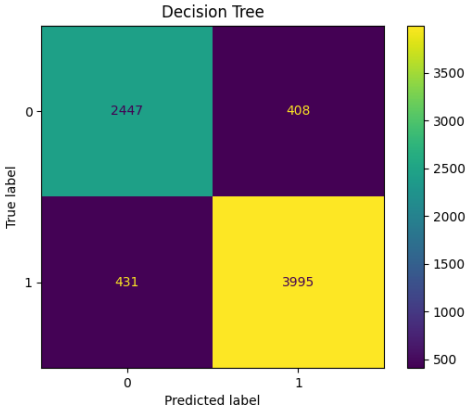

Figure 43: *confusion matrix obtained by the DeT in the first iteration;*

type of error, and even the absolute error amount is not big as the ones obtained by the other models.

The following table shows the results obtained by the DeT in details [*Table 10*]:

Table 10: *results of the DeT model;*

Precision	***Accuracy***	***Recall***	***F1***
0.906	0.883	0.901	0.903
0.838	0.741	0.679	0.750
0.904	0.895	0.966	0.934
0.883	**0.840**	**0.849**	**0.862**

As may be seen, the results are very good, and there is no evident sign of any bias toward any type of error, and therefore the DeT has provided acceptable estimations for the dataset utilised.

5.6.4. Logistic Regression

The LoR is the only model that is technically a regression, applied to the classification and in this work. As may be seen by the confusion matrix aside [*Figure 44*] the results are mediocre and hence the model is better than others but not comparable to the results obtained by the DeT. It can be said that the results seem heavily biased, and the number of errors is important. The following table shows the results obtained by the LoR for each experiment [*Table 11*]:

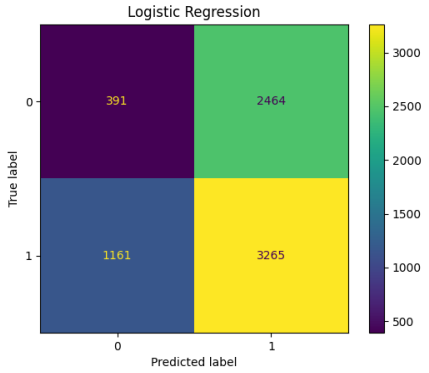

Figure 44: *confusion matrix of the LoR in the first iteration;*

Table 11: *results obtained by the LoR model;*

Precision	***Accuracy***	***Recall***	***F1***
0.567	0.502	0.737	0.643
0.438	0.342	0.529	0.479
0.714	0.368	0.306	0.429
0.573	**0.404**	**0.524**	**0.517**

As may be seen, the results are medially around 0.5, however looking at the single measurement, it is clear that the results are influenced by the datasets on which the model is tested and hence the model is not able to detect efficiently a path, however, as already said for other models, a different set of parameters obtained with a more accurate optimisation might be able to revert this condition and making the model more precise.

5.6.5. SVC

The SVM, or SVC, since it is implemented for a classification algorithm, obtained a awful result and characterised even by a high degree of error, seeing the output depicted in the confusion matrix [*Figure 45*], it seems also that a bias toward the first type error is heavily present in the model.

The following table reports the output of the SVC in the algorithm [*Table 12*]:

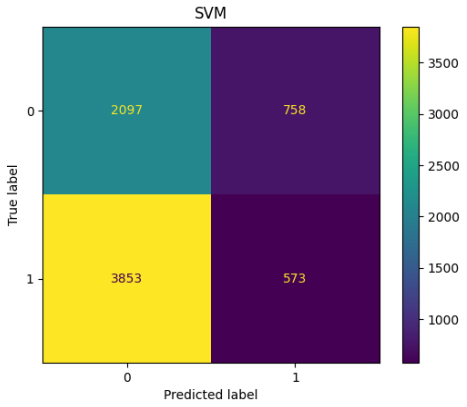

Figure 45: *confusion matrix of the SVC obtained in the first iteration;*

Table 12: *results obtained by the SVC model;*

Precision	***Accuracy***	***Recall***	***F1***
0.430	0.366	0.129	0.199
0.569	0.566	0.988	0.722
0.215	0.196	0.015	0.028
0.405	**0.376**	**0.377**	**0.317**

A bias toward the false negative is confirmed. The overall values show that the model did not detect the pattern, however a further hyperparameter tuning might be able to slightly increase the capabilities of the model, but it is unlikely that it would make the model acceptable.

5.6.6. Naïve Bayes

The NB obtained the worst results as may be seen in the in the confusion matrix [*Figure 36*], moreover it also depicts a situation where the model is totally biased, the predictions it made define the model as unusable for this algorithm and it also has been confirmed by the following table [*Table 13*] :

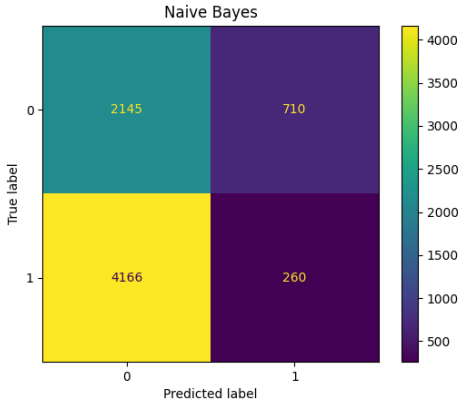

Figure 46: *confusion matrix by the NB in the first iteration;*

Table 13: *results obtained by the NB;*

Precision	***Accuracy***	***Recall***	***F1***
0.268	0.330	0.058	0.096
0.484	0.413	0.376	0.423
0.527	0.235	0.107	0.178
0.426	**0.326**	**0.180**	**0.232**

As quite clear by the outputs reported, the NB is biased and still committed an enormous amount of error. It is quite clear that the model did not detect any path for the target feature and hence is not able to be used in an actual situation. Since for it's hyperparameter tuning have been used all the parameters available, it is unlikely that under other conditions the algorithm would obtain a better result,

5.6.7. ANN

The ANN is the only NN that has been applied for this algorithm, the results obtained are however mediocre as may be seen in the confusion matrix [*Figure 47*]: the model is heavily biased toward the second type errors, and it has never been actually able to correctly detect a path for the wear of the machine. The confusion matrix depicts also a situation where there

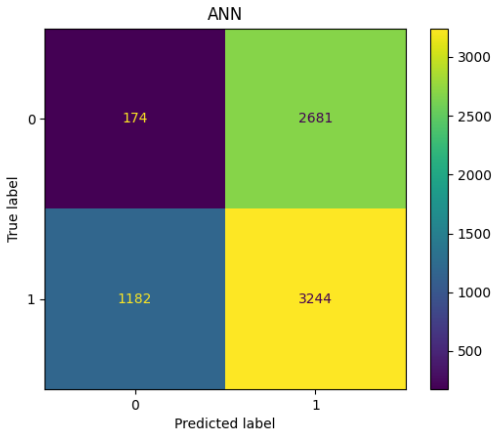

Figure 47: *confusion matrix obtained by the ANN in the first iteration;*

is a tendency to define as “worn” almost all the data available and hence it already proves the inadequacy of the model to the algorithm. As a further example of that, the report from the three iteration is displayed beneath this paragraph [*Table 14*]:

Table 14: *output report for the ANN;*

Precision	***Accuracy***	***Recall***	***F1***
0.495	0.385	0.590	0.538
0.571	0.569	0.994	0.725
0.368	0.186	0.072	0.120
0.478	**0.380**	**0.552**	**0.461**

The bias is evidently highlighted by the Recall metrics that shows a huge variance in the three experiments, that indeed influences the F1 as well. However, the values reached are purely due to the averaged nature of the experiments since a low value is balanced by an higher one, hence the model is just unable to detect the path of the wear of the machine and it should not be applied for an actual situation.

5.6.8. Comment

In this chapter several different models have been implemented and tested on a classification algorithm to verify both the implementation theories seen in the other chapters and to provide a deeper understanding of this methodologies. The final evaluation portrayed a situation where the DeT outperformed all the other models, and the RF model is almost as balanced.
This work also reviewed the implications and complications that may arise from any ML implementation and provide practical suggestions for the preparation of a ML algorithm.
All the confusion matrixes and the outputs presented in this chapter have been computed on the Python algorithm that has been presented as well and the compiler itself did not return any error for this algorithm.

6. Conclusions

This work has studied the applications and the implementation of the ML algorithms, from the collection of the dataset to the evaluation of the models. Initially the review was focused on a purely theoretical analysis with a study on the applications like the quality and the tool wear predictions, and even applications that included the DT systems, then the analysis started to consider more practical views to see the MLs, as studying the most common ML models, from the RF to the KNN and which Python libraries are usually implied for the deployment of a ML or its preparation.

This work studied then, the general methodologies that are used for the implementation of ML: how to collect a useful dataset from a machine and how to manipulate it (normalisation and sampling) clean the data that cannot be used, how to select the features (regressors) for the analysis and how a ML can be optimised using the hyperparameter tuning. Moreover, the study reviewed the most used evaluation methods for a ML algorithm, for both regression and classification models, including numerical and graphical solutions.

Eventually the thesis, studied the implementation of a ML through a personal use case with a public dataset. The use case briefly studied the CNC milling machine that was the machine the dataset was collected from, the dataset itself and how this data has been cleaned and normalised, the following task has been the selection of the features and choosing the ML model to apply to the algorithm, that were chosen accordingly to what has been seen in the analysis of the sources. The final phase of this implementation was the evaluation that has been fully portrayed for the whole set of models, with all the evaluation methods that have been seen in the former chapter, the results where that the RF and DeT both obtained acceptable results and they can actually be used for a ML implementation.

All this procedures that have been seen in the last chapter are presented with the python algorithm that was used for this personal implementation, in order to provide the reader a full understanding of what has been done and how it can be furtherly improved.

6.1. Future work

This thesis reviewed the most common techniques and methodologies that can be applied for the implementation of a ML algorithm, eventually providing a practical solution for a classification problem. Although, this latter approach is complete and fully works, it may be

improved furtherly with an expanded session of hyperparameter tuning, or even with the implementation of Deep Learning methodologies that have not really studied in this work.
Moreover, a proper study with the DTs should be considered to increase the knowledge of this latter, and to understand the relationship between DT and ML.
Eventually, a further study for a regression algorithm should also be considered in order to review the other kind of ML that has not be seen in this practical implementation.

Bibliography

[1] R. Oberlé, S.Schorr, L.Yi, M.Glatt, D.Bähre, J. C. Aurich, ”A Use Case to Implement Machine Learning for Life Time Prediction of Manufacturing Tools”, *Procedia CIRP,* Volume 93, Pages 1484-1489, Institute of Production Engineering, Saarland University, 2020.

[2] M. Ay, S. Stemmler, M.Schwenzer, D.Abel, Thomas Bergs, “Model Predictive Control in Milling based on Support Vector Machines”, *IFAC-PapersOnLine,* Volume 52, Issue 13, Pages 1797-1802, RWTH Aachen University, 2019.

[3] H.Wang, B.Li, F.Xuan, “Fatigue-life prediction of additively manufactured metals by continuous damage mechanics (CDM)-informed machine learning with sensitive features”, *International Journal of Fatigue,* Volume 164, East China University of Science and Technology, 2022.

[4] H.Tercan, T.Meisen, “Machine learning and deep learning based predictive quality in manufacturing: a systematic review”, *Journal of Intelligent Manufacturing,* Volume 33, Issue 7, Pages 1879 - 1905, University of Wuppertal, 2022.

[5] H.Wang, B. Li, F.Xuan, “A dimensionally augmented and physics-informed machine learning for quality prediction of additively manufactured high-entropy alloy”, *Journal of Materials Processing Technology,* Volume 307, Article number 117637, School of Mechanical and Power Engineering, East China University of Science and Technology, 2022.

[6] M.R.Sarabi, M.M.Alseed, A.A.Karagoz, S.Tasoglu, “Machine Learning-Enabled Prediction of 3D-Printed Microneedle Features”, *Biosensors*, Volume 12, Issue 7, Article number 491, Graduate School of Sciences & Engineering, Koç University, 2022.

[7] L. Dang, X. He, D. Tang, Y. Li, T. Wang, “A fatigue life prediction approach for laser-directed energy deposition titanium alloys by using support vector regression based on pore-induced failures “, International Journal of Fatigue, Volume 159, Issue 106748, Beihang University, 2022.

[8] J.Gu, L.Zhao, X.Yue, N.I. Arshad, Mohamad, H.Ummul, “Multistage quality control in the manufacturing process using blockchain with machine learning technique“, *Information Processing and management,* Volume 60, Issue 4, Article number 103341, Yangzhou University, 2023.

[9] A.K.Sah, M. Agilan, S. Dineshraj, M.R. Rahul, B. Govind “Machine learning-enabled prediction of density and defects in additively manufactured Inconel 718 alloy”, *Materials*

Today Communications, Volume 30, Article number 103193, Indian Institute of Technology, 2022.

[10] K.Manjunath, S.Tewary, N.Khatri, K.Cheng, "Monitoring and predicting the surface generation and surface roughness in ultraprecision machining: A critical review", *Machines*, Volume 9, Issue 12, Article number 369, Central Scientific Instruments Organisation, 2021.

[11] Z. Zhan, W. Hu, Q. Meng, "Data-driven fatigue life prediction in additive manufactured titanium alloy: A damage mechanics-based machine learning framework", *Engineering Fracture Mechanics*, Volume 252, Article number 107850, Beihang University,2021.

[12] B.Crawforda, R.Sourki, H.Khayyamb, A.S. Milani "A machine learning framework with dataset-knowledgeability pre-assessment and a local decision-boundary crispness score: An industry 4.0-based case study on composite autoclave manufacturing", *Computers in Industry*, Volume 132, Article number 103510, University of British Columbia, 2021.

[13] J.Wu, M. Wu, Z. Chen, Xiaoli Li, Ruqiang Yan, "A joint classification-regression method for multi-stage remaining useful life prediction", *Journal of Manufacturing Systems*, Volume 58, Part A, Pages 109-119, Singapore School of Mechanical Engineering, 2021.

[14] N.E.Sizemore, M.L. Nogueira, N.P.Greis, M.A. Davies, "Application of Machine Learning to the Prediction of Surface Roughness in Diamond Machining", *Procedia Manufacturing*, Volume 48, Pages 1029-1040, University of North Carolina, 2020.

[15] M.Hu, Q.Tan, R.Knibbe, S.Wang, X.Li, T.Wu, S.Jarin, M.Zhang, "Prediction of Mechanical Properties of Wrought Aluminium Alloys Using Feature Engineering Assisted Machine Learning Approach", *Metallurgical and Materials Transactions A: Physical Metallurgy and Materials Science,* Volume 52, Issue 7, Pages 2873 - 2884, School of Mechanical and Mining Engineering, 2021.

[16] S. Schorr, M. Möller, J.Heib, D.Bähre, "Quality Prediction of Drilled and Reamed Bores Based on Torque Measurements and the Machine Learning Method of Random Forest", *Procedia Manufacturing,* Volume 48, Pages 894-901, RWTH Aachen University, 2020

[17] C.Gutschi, N.Furian, J.Suschnigg, D.Neubacher, S.Voessner, "Log-based predictive maintenance in discrete parts manufacturing", *Procedia CIRP,* Volume 79, Pages 528-533, Graz University of Technology, 2019.

[18] D.Wu, C.Jennings, J.Terpenny, R.X.Gao, S.Kumara, "A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests", *ASME*, Volume 139, Issue 7, Pennsylvania State University, 2017.

[19] L.Guo, Y.Yu, H.Gao, T.Feng, Y.Liu, "Online Remaining Useful Life Prediction of Milling Cutters Based on Multisource Data and Feature Learning", *IEEE Transactions on*

Industrial Informatics, Volume 18, Issue 8, Pages 5199 - 52081, Southwest Jiaotong University,2022.

[20] T.F.De Barrena, J.L.Ferrando, A.García, X.Badiola,M.S.de Buruaga, J.Vicente, "Tool remaining useful life prediction using bidirectional recurrent neural networks (BRNN)", *The International Journal of Advanced Manufacturing Technology,* Volume 125, Issue 9-10, Pages 4027 - 4045, Fundación Vicomtech, 2023.

[21] A. Bonci, A. Di Biase, A.F. Dragoni, S. Longhi, P. Sernani, A. Zega, "Machine learning for monitoring and predictive maintenance of cutting tool wear for clean-cut machining machines ", *IEEE International Conference on Emerging Technologies and Factory Automation*, Volume 2022, Polytechnic University of Marche,2022.

[22] Z. Zhan, H. Li, "Machine learning based fatigue life prediction with effects of additive manufacturing process parameters for printed SS 316L", International Journal of Fatigue, Volume 142, Article 105941, Beihang University, 2021.

[23] I. Baturynska, K. Martinsen, "Prediction of geometry deviations in additive manufactured parts: comparison of linear regression with machine learning algorithms ", *Journal of Intelligent Manufacturing*, Volume 32, Issue 1, Pages 179 - 200, Norwegian University of Science and Technology,2021.

[24] Y.Zhou, B.Sun, W.Sun, "A tool condition monitoring method based on two-layer angle kernel extreme learning machine and binary differential evolution for milling", *Measurement: Journal of the International Measurement Confederation*, Volume 16615, Article number 108186, Wenzhou University, 2020.

[25] J. L. Bartlett, A.Jarama, J.Jones, X.Li, "Prediction of microstructural defects in additive manufacturing from powder bed quality using digital image correlation", *Materials Science and Engineering: A*, Volume 794, Article 140002, University of Virginia, 2020.

[26] R. de Souza Borges Ferreira. A. Sabbaghi, Q. Huang, "Automated Geometric Shape Deviation Modeling for Additive Manufacturing Systems via Bayesian Neural Networks", *IEEE Transactions on Automation Science and Engineering,* Volume 17, Issue 2, Pages 584 - 598, Article number 8851391, University of Southern California, 2020.

[27] N.Hertlein, S.Deshpande, V. Venugopal, M. Kumar, S. Anand, "Prediction of selective laser melting part quality using hybrid Bayesian network", *Additive Manufacturing*, Volume 32, Article number 101089, University of Cincinnati, 2020.

[28] C.Y. Park, J.W.Kim, B.Kim, J.Lee, "Prediction for Manufacturing Factors in a Steel Plate Rolling Smart Factory Using Data Clustering-Based Machine Learning", *IEEE,* Volume 8, Pages 60890 - 60905, Article number 9046761, University, Pohang, 2020.

[29] E.Salvati, A.Tognan, L.Laurenti, M.Pelegatti, F.De Bona, “A defect-based physics-informed machine learning framework for fatigue finite life prediction in additive manufacturing”, *Materials and Design*, Volume 222, Article number 111089, University of Udine,2022.

[30] T.von Hahn, C. K. Mechefske, “Machine Learning in CNC Machining: Best Practices “, *Safety of Machinery: Design, Monitoring, Manufacturing*, Queen’s University, 2022.

[31] J. He, C. Yin, Y. Wang, “Deep multi-task network based on sparse feature learning for tool wear prediction”, *Sage Journals,* Nanjing University of Science and Technology, 2020.

[32] D. Wang, Q. Liu, D. Wu, L. Wang, “Meta domain generalization for smart manufacturing: Tool wear prediction with small data”, *Journal of Manufacturing Systems*, Volume 62, Pages 441-449, University of Central Florida, 2022.

[33] M. Li, M. Burzo, “Tool Wear Monitoring Using Machine Learning “, *IEEE Canadian Conference on Electrical and Computer Engineering,* University of Michigan-Flint, 2021.

[34] B.Lutz, D. Kisskalt, D.Regulin, B.Aybar, J.Franke, “Automated Domain Adaptation in Tool Condition Monitoring using Generative Adversarial Networks”, *IEEE International Conference on Automation Science and Engineering*, Volume 2021, Pages 1326 - 133123, Institute for Factory Automation and Production Systems,2021.

[35] A.Varghese, V.Kulkarni, S.S.Joshi, “Tool Life Stage Prediction in Micro-milling from Force Signal Analysis Using Machine Learning Methods“, *ASME*, Volume 143, Issue 5, Indian Institute of Technology, 2020.

[36] N.K.Mandal, N.K.Singh, N.H. Tarafdar, A.Hazra, “Correlating tool wear and surface integrity of a CNC turning process using Naïve based classifiers”, *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Volume 235, Issue 5, Pages 772 - 781, Indian Institute of Technology, 2021.

[37] B. Pang, D.Yuan, D.Li, Z.Di, “Tool Remaining Useful Life Prediction Method Based on Time-frequency Features Fusion and Long Short-term Memory Network”, *2021 Global Reliability and Prognostics and Health Management, PHM-Nanjing 2021*, Shandong University, 2021.

[38] V.Parwal, B.K. Rout, “Machine learning based approach for process supervision to predict tool wear during machining“, *Procedia CIRP*, Volume 98, Pages 133-138, Birla Institute of Technology and Science Pilan, 2021.

[39] M.Cheng, L.Jiao, X.Shi, X.Wang, P.Yan, Y.Li, “An intelligent prediction model of the tool wear based on machine learning in turning high strength steel”, *Proceedings of the*

Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, Volume 234, Issue 13, Pages 1580 - 15971, Beijing Institute of Technology, 2020.
[40] J. Karandikar, T. Schmitz, S. Smith, "Logistic classification for tool life modeling in machining", *Procedia CIRP,* Volume 101, Pages 106-109, University of Tennessee, 2021.
[41] A. Mohamed, M. Hassan, R. M'Saoubi, H. Attia, "Tool Condition Monitoring for High-Performance Machining Systems—A Review", *Sensors*, Volume 22, Issue 6, Article number 2206, McGill University, 2022.
[42] C.J.Fourie, J. A.Du Plessis, "Implementation of Machine Learning techniques for prognostic of railway wheel flange wear", *The South African Journal of Industrial Engineering,* University of Stellenbosch, 2020.
[43] E.Traini, G.Bruno, G. D'Antonio, F. Lombardi, "Machine Learning Framework for Predictive Maintenance in Milling", *IFAC-PapersOnLine*, Volume 52, Issue 13, Pages 177-182, Politecnico di Torino, 2019.
[44] A. Gouarir, G. Martínez-Arellano, G. Terrazas, P. Benardos, S. Ratchev, "In-process Tool Wear Prediction System Based on Machine Learning Techniques and Force Analysis", *Procedia CIRP*, Volume 77, Pages 501-504, The University of Nottingham, 2018.
[45] N.S.Karuppusamy; P.Pandian P; H.Lee; B.Kang, "Tool wear and tool life estimation based on linear regression learning", *IEEE*, Christ University, 2015.
[46] Z. Huang, M. Fey, C. Liu, E. Beysel, X. Xu, C. Brecher, "Hybrid learning-based digital twin for manufacturing process: Modelling framework and implementation", *Robotics and Computer-Integrated Manufacturing*, Volume 82, RWTH Aachen University, 2023.
[47] H. Mu, F. He, L. Yuan, P. Commins, H. Wang, Z. Pan, "Toward a smart wire arc additive manufacturing system: A review on current developments and a framework of digital twin", *Journal of Manufacturing Systems,* Volume 67, Pages 174-189, University of Wollongong, 2023.
[48] J. Lv, X. Li, Y. Sun, Y. Zheng, J. Bao, "A bio-inspired LIDA cognitive-based Digital Twin architecture for unmanned maintenance of machine tools", *Robotics and Computer-Integrated Manufacturing*, Volume 80, Donghua University, 2023.
[49] C. Li, P. Zheng, Y. Yin, Y. M. Pang, S. Huo, "An AR-assisted Deep Reinforcement Learning-based approach towards mutual-cognitive safe human-robot interaction", *Robotics and Computer-Integrated Manufacturing,* Volume 80, Hong Kong Polytechnic University, 2023.
[50] C. Zhang, G. Zhou, J. Li, F. Chang, K. Ding, D. Ma, "A multi-access edge computing enabled framework for the construction of a knowledge-sharing intelligent machine tool swarm

in Industry 4.0", *Journal of Manufacturing Systems*, Volume 66, Pages 56-70, Xi'an Jiaotong University, 2023.

[51] Z. Liu, L. Hu, W. Hu, Jianrong Tan, "Petri Nets-Based Modeling Solution for Cyber-Physical Product Control Considering Scheduling, Deployment, and Data-Driven Monitoring", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Volume 53, Issue 2, Zhejiang University, 2023.

[52] O. Khalaj, M.Jamshidi, P.Hassas, M. Hosseininezhad, B. Mašek, C. Štadler, J. Svoboda, "Metaverse and AI Digital Twinning of 42SiCr Steel Alloys", *Mathematics*, University of West Bohemia, 2023.

[53] B. D. Deebak, F. Al-Turjman, "Digital twin assisted: Fault diagnosis using deep transfer learning for machining tool condition", *Intelligent Systems*, 2022.

[54] S. Wu, W. Xiang, W. Li, L. Chen, C. Wu, "Dynamic Scheduling Optimization of Production Workshops Based on Digital Twin", *Applied Science*, University of Shanghai, 2022.

[55] D. B. Kim, G. Shao, G. Jo, "A digital twin implementation architecture for wire + arc additive manufacturing based on ISO 23247", *Manufacturing Letters*, Volume 34, Pages 1-5, Tennessee Technological University, 2022.

[56] Y. Li, G.Lei, G. Bramerdorfer, S. Peng, X. Sun, J. Zhu, "Machine Learning for Design Optimization of Electromagnetic Devices: Recent Developments and Future Directions", *Applied Sciences*, Volume 11, Issue 4, Pages 1 - 242, Article number 1627, Zhongyuan University of Technology, 2021.

[57] H.L. Wei, T. Mukherjee, W. Zhang, J.S. Zuback, G.L. Knapp, A. De, T. DebRoy, "Mechanistic models for additive manufacturing of metallic components", *Progress in Materials Science*, Volume 116, Nanjing University of Science and Technology, 2021.

[58] M. Dehghanimohammadabadi, S.Belsare, R.Thiesing, "Simulation-Optimization of Digital Twin", *Proceedings - Winter Simulation Conference,* Volume 2021, Northeastern University, 2021.

[59] Enis Karaarslan, Mohammed Babiker, "Digital Twin Security Threats and Countermeasures: An Introduction", *14th International Conference on Information Security and Cryptology, ISCTURKEY 2021*, Pages 7 - 11, Muğla Sitki Koçman University, 2021.

[60] S.Alvarez-Napagao, B. Ashmore, M.Barroso, C. Barrué, C. Beecks, F. Berns, I. Bosi, S.A. Chala, N. Ciulli, M. Garcia-Gasulla, A.Grass, D.Ioannidis, "Knowledge Project – Concept, Methodology and Innovations for Artificial Intelligence in Industry 4.0", *IEEE International Conference on Industrial Informatics (INDIN),* Universitat Politècnica de Catalunya, 2021.

[61] Z. Wua, J. Li, “A Framework of Dynamic Data Driven Digital Twin for Complex Engineering Products: The Example of Aircraft Engine Health Management”, *Procedia Manufacturing*, Volume 55, Pages 139-146, Virginia State University, 2021.

[62] C. Manettas, N.Nikolakis, K. Alexopoulos, “Synthetic datasets for Deep Learning in computer-vision-assisted tasks in manufacturing”, *Procedia CIRP,* Volume 103, Pages 237-242, 2021.

[63] L. Overbeck, A. Hugues, M.C. May, A. Kuhnle, G. Lanza, “Reinforcement Learning Based Production Control of Semi-automated Manufacturing Systems”, *Procedia CIRP*, Volume 103, Pages 170 - 175, Karlsruhe Institute of Technology, 2021.

[64] A. Papacharalampopoulos, K. Sabatakakis, P. Stavropoulos, “Incorporating process physics phenomena in formation of digital twins: laser welding case”, *Procedia CIRP,* Volume 99, Pages 490 - 495, University of Patras, 2021.

[65] M. C. May, L. Overbeck, M. Wurster, A. Kuhnle, G. Lanza, “Foresighted digital twin for situational agent selection in production control”, *Procedia CIRP*, Volume 99, Pages 27-32, Karlsruhe Institute of Technology, 2021.

[66] T. Wang, J. Li, Z. Kong, X.Liu, H. Snoussi, H. Lv, “Digital twin improved via visual question answering for vision-language interactive mode in human–machine collaboration”, *Journal of Manufacturing Systems*, Volume 58, Pages 261 - 269, Beihang University, 2021.

[67] Y. Zhou, T. Xing, Y. Song, Y. Li, X. Zhu, G. Li, S. Ding, “Digital-twin-driven geometric optimization of centrifugal impeller with free-form blades for five-axis flank milling”, Journal of Manufacturing Systems, Volume 58, Part B, Pages 22-35, Beihang University, 2021.

[68] T. Borangiu, S. Răileanu, A. Silişteanu, S. Anton, F. Anton, “Smart Manufacturing Control with Cloud-embedded Digital Twins”, *IEEE*, University Politehnica of Bucharest, 2020.

[69] A.Hürkamp, S. Gellrich, Tim Ossowski, J. Beuscher, S. Thiede, C. Herrmann, K. Dröder, “Combining Simulation and Machine Learning as Digital Twin for the Manufacturing of Overmolded Thermoplastic Composites”, *Journal of Manufacturing and Materials Processing*, Volume 4, Article number 92, Technische Universität Braunschweig, 2020.

[70] S. Stieber, A. Hoffmann, A. Schiendorfer, W. Reif, M. Beyrle, J. Faber, M. Richter, M. Sause, “Towards Real-time Process Monitoring and Machine Learning for Manufacturing Composite Structures”, *IEEE*, University of Augsburg, 2020.

[71] U. Awasthi, Z. Wang, N. Mannan, K. R. Pattipati, G. M. Bollas, “Physics-based modeling and information-theoretic sensor and settings selection for tool wear detection in precision

machining", *Journal of Manufacturing Processes*, Volume 81, Pages 127-140, UTC Institute for Advanced Systems Engineering, 2022.
[72] K. Alexopoulos, N. Nikolakis, G. Chryssolouris, "Digital twin-driven supervised machine learning for the development of artificial intelligence applications in manufacturing", *International Journal of Computer Integrated Manufacturing,* Volume 33, Issue 5, Pages 429 - 4393, University of Patras, 2020.
[73] F. Mostafa, L. Tao, W. Yu, "An effective architecture of digital twin system to support human decision making and AI-driven autonomy", *Concurrency and Computation: Practice and Experience*, Volume 33, Issue 19, La Trobe University, 2021.
[74] Z. Chen, "Understanding of the Modeling Method in Additive Manufacturing", *IOP Conference Series: Materials Science and Engineering,* Volume 711, Issue 17, 2020.
[75] S. M. Bazaz, M. Lohtander, J. Varis, "The prediction method of tool life on small lot turning process – Development of Digital Twin for production", *Procedia Manufacturing*, Volume 51, Lappeenranta-Lahti University of Technology, 2020.
[76] R. A. C. Diaz, M. Ghita, D. Copot, I. Roxana Birs, C. Muresan, "Context-Aware Control Systems: An Engineering Applications Perspective", *IEEE Access*, Ghent University, 2020.
[77] Y. Xie, K. Lian, Q. Liu, C. Zhang, H. Liu, "Digital twin for cutting tool: Modeling, application and service strategy", *Journal of Manufacturing Systems*, Volume 58, Part B, Pages 305-312, Huazhong University of Science and Technology, 2021.
[78] A.Fuller, Z.Fan, C. Day, C, Barlow, "Digital Twin: Enabling Technologies, Challenges and Open Research", *IEEE Access*, Volume 8, Pages 108952 - 10897, Keele University, 2020.
[79] B. J. Ralph, A. Schwarz, M. Stockinger, "An Implementation Approach for an Academic Learning Factory for the Metal Forming Industry with Special Focus on Digital Twins and Finite Element Analysis", Procedia Manufacturing, Volume 45, Pages 253-258, Montanuniversität Leoben, 2020.
[80] Q. Min, Y. Lu, Z. Liu, C. Su, B. Wang, "Machine Learning based Digital Twin Framework for Production Optimization in Petrochemical Industry", *International Journal of Information Management*, Volume 49, Pages 502-519, 2021.
[81] I. M. Cavalcante, E. M. Frazzon, F. A. Forcellini, D. Ivanov, "A supervised machine learning approach to data-driven simulation of resilient supplier selection in digital manufacturing", *International Journal of Information Management,* Volume 49, Pages 86 - 97, Federal University of Santa Catarina, 2019.

[82] H. Ko, P. Witherell, N. Y. Ndiaye, Y. Lu, "Machine Learning based Continuous Knowledge Engineering for Additive Manufacturing", *IEEE*, Nanyang Technological University, 2019.

[83] C. Cronrath, A. R. Aderiani, B. Lennartson, "Enhancing Digital Twins through Reinforcement Learning", *IEEE*, 2019.

[84] A. Mayr, M. Weigelt, J. von Lindenfels, J. Seefried, M. Ziegler, A. Mahr, N. Urban, A. Kühl, F. Hüttel, J. Franke, "Electric Motor Production 4.0 – Application Potentials of Industry 4.0 Technologies in the Manufacturing of Electric Motors", *2018 8th International Electric Drives Production Conference, EDPC 2018 - Proceedings*, Article number 8658294, Friedrich-Alexander University Erlangen-Nuremberg, 2018.

[85] B. R. Barricelli, E. Casiraghi, D. Fogli, "A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications", *IEEE Access*, Università degli Studi di Brescia, 2019.

[86] Q. Qiao, J. Wang, L. Ye, R. X. Gao, "Digital Twin for Machining Tool Condition Prediction", *Procedia CIRP*, Volume 81, Pages 1388-1393, China University of Petroleum, 2019.

[87] D. Gyulai, A. Pfeiffer, G. Nick, V. Gallina, W. Sihn, L. Monostori, "Lead time prediction in a flow-shop environment with analytical and machine learning approaches", *IFAC-PapersOnLine*, Volume 51, Issue 11, Pages 1029-1034, Hungarian Academy of Sciences, 2018.

[88] T. Kong, T. Hu, T. Zhou, Y. Ye, "Data Construction Method for the Applications of Workshop Digital Twin System", *Journal of Manufacturing Systems*, Volume 58, Pages 323-328, Shandong University, 2021.

[89] C. Liu, L. Le Roux, C. Körner, O. Tabaste, F. Lacan, S. Bigot, "Digital Twin-enabled Collaborative Data Management for Metal Additive Manufacturing Systems", Journal of Manufacturing Systems Volume 62, Pages 857-874, Cardiff University, 2022.

[90] B. Wang, S. J. Hu, L. Sun, T. Freiheit," Intelligent welding system technologies: State-of-the-art review and perspectives", *Journal of Manufacturing Systems,* Volume 56, Pages 373-391, University of Michigan, 2020.

[91] "StackOverFlow" [Online]. Available: https://stackoverflow.com.

[92] "Scopus" [Online]. Available: https://www-scopus-com.

[93] "Wikipedia" [Online]. Available: https://en.wikipedia.org/wiki/Tool_wear

[94] "MatPlotLib" [Online]. Available: https://matplotlib.org

[95] "NumPy" [Online]. Available: https://numpy.org

[96] “Seaborn” [Online]. Available: seaborn: statistical data visualization — seaborn 0.12.2 documentation (pydata.org)
[97] “Pandas” [Online]. Available: https://pandas.pydata.org
[98] “SciKitLearn” [Online]. Available: https://scikit-learn.org/stable
[99] “Wikipedia” [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning
[100] “Toward Data Science” [Online]. Available: https://towardsdatascience.com/machine-learning-basics-decision-tree-regression-1d73ea003fda
[101] “Toward Data Science” [Online]. Available: Machine Learning Basics: Random Forest Regression | by Gurucharan M K | Towards Data Science
[102] “Toward Data Science” [Online]. Available: Top 6 Machine Learning Algorithms for Classification | by Destin Gong | Towards Data Science
[103] “Javapoint” [Online]. Available: Support Vector Machine (SVM) Algorithm - Javatpoint
[104] “GeeksforGeeks” [Online]. Available: Support Vector Machine (SVM) Algorithm - GeeksforGeeks
[105] “Toward Data Science” [Online]. Available: Naive Bayes Classifier. What is a classifier? | by Rohith Gandhi | Towards Data Science
[106] “Javapoint” [Online]. Available: K-Nearest Neighbor(KNN) Algorithm for Machine Learning - Javatpoint
[107] “Javapoint” [Online]. Available: Artificial Neural Network Tutorial - Javatpoint
[108] “Toward Data Science” [Online]. Available: Convolutional Neural Networks, Explained | by Mayank Mishra | Towards Data Science
[109] “Wikipedia” [Online]. Available: Convolutional neural network - Wikipedia
[110] “Toward Data Science” [Online]. Available: Recurrent Neural Networks. Remembering what’s important | by Mahendran Venkatachalam | Towards Data Science
[111] “Javapoint” [Online]. Available: Recurrent Neural Network (RNN) in TensorFlow - Javatpoint
[112] “AnyLogic” [Online]. Available: Digital Twin Development and Deployment – AnyLogic Simulation Software
[113] “ICE” [Online]. Available: Digital twins in F1 and the built environment | Institution of Civil Engineers (ICE)
[114] “GDPR” [Online]. Available: General Data Protection Regulation (GDPR) – Official Legal Text (gdpr-info.eu)
[115] “Production Machining” [Online]. Available: How to Collect and Use Machine Data | Production Machining

[116] “Kaggle” [Online]. Available: CNC Mill Tool Wear | Kaggle

[117] “All3DP” [Online]. Available: What Is CNC Milling? – Simply Explained | All3DP