POLITECNICO DI TORINO

DIGEP – DEPARTMENT OF MANAGEMENT AND PRODUCTION ENGINEERING

MASTER OF SCIENCE DEGREE IN MANAGEMENT ENGINEERING

MASTER OF SCIENCE DEGREE THESIS

MACHINE LEARNING APPLICATIONS FOR SURFACE ROUGHNESS IN TURNING

Supervisors:
Prof. Franco Lombardi
Prof. Giulia Bruno

Candidate:
Vito Antonio Duca
S292936

Academic Year 2021 – 2022

Acknowledgements

I would like to extend my sincere gratitude towards my thesis supervisor, Prof. Giulia Bruno and Prof. Franco Lombardi for providing this opportunity and for valuable feedback and constructive suggestions during the conduct of this master thesis.

Moreover, I am grateful to my supervisors for granting me the chance to work on a project that combines data science and theoretical course knowledge in the best possible way.

Last, but not least, I would like to convey my deep appreciation to my family and friends for their unwavering support and constant encouragement throughout my years of study.

Abstract

Machining production relies heavily on the quality of the surface roughness, and researchers have spent years studying ways to predict it. Typically, three approaches are taken to model surface roughness: empirical methods, theoretical/simulation methods, and soft computing methods.

This study involved using Machine Learning models to analyze two datasets generated from machining experiments. These experiments involved turning AISI H13 steel with cutting fluid. The first dataset, which contained 324 samples, was based on theoretically new-tool conditions. The second dataset, which contained 288 samples, varied cutting tool flank wear in three levels. To increase the available data, a strategy was employed to boost by six times the number of measurements without increasing the number of experiments.

Machine Learning models were used to predict the output, which was the arithmetic mean deviation (Ra), for both datasets. Smaller datasets resulted in models that were prone to overfitting and performed worse than larger datasets. The models trained on the first dataset (without tool wear) were unable to generalize to the second dataset (with tool wear). This suggests that tool wear is a critical factor when using Machine Learning models to model surface roughness in turning processes. Additionally, Machine Learning models outperformed classical theoretical surface roughness equations.

The study also covers techniques for data cleansing, data manipulation to extract and select features, the use of these features in training various Machine Learning models and testing them to determine how turning process parameters and cutting tool wear affect the surface roughness parameter "Ra".

Lastly, the study highlights various tools that can be used for the regression analysis of Machine Learning models, such as Linear Regression, Decision Tree Regression, Random Forest Regression, Bayesian Ridge Regression, KNN Regression, Kernel Ridge Regression, and Neural Networks.

Contents

List of Figures

List of Tables

List of Code Snippets

Chapter 1 – Introduction

Synopsis

The opening chapter of the thesis provides an in-depth understanding of the rationale behind the chosen topic and introduces various terms associated with it. After a brief introduction to multiple aspects of Industry 4.0, the chapter provides insight on the key Enabling Technologies and the significant impact of Artificial Intelligence on the industry.

1.1 Industry 4.0

The industrialization process started at the end of the 18th century with the incorporation of mechanical production equipment, which was made possible by the utilization of steam power in the production of goods. This first industrial revolution was then followed by the second industrial revolution in the late 19th century, which was marked by the use of new energy sources, such as electricity, and the implementation of the assembly line, leading to mass production. The third industrial revolution came about in the early 1970s and was characterized by the use of knowledge in the field of electronics and information technology to enhance the automation of production processes. Subsequently, advancements in digital technologies have been rapidly transforming the industry and are continuing to do so, leading to the phenomenon referred to as Industry 4.0 [1] or the Fourth Industrial Revolution, which has had a profound impact on all major sectors of the economy and has spread globally.

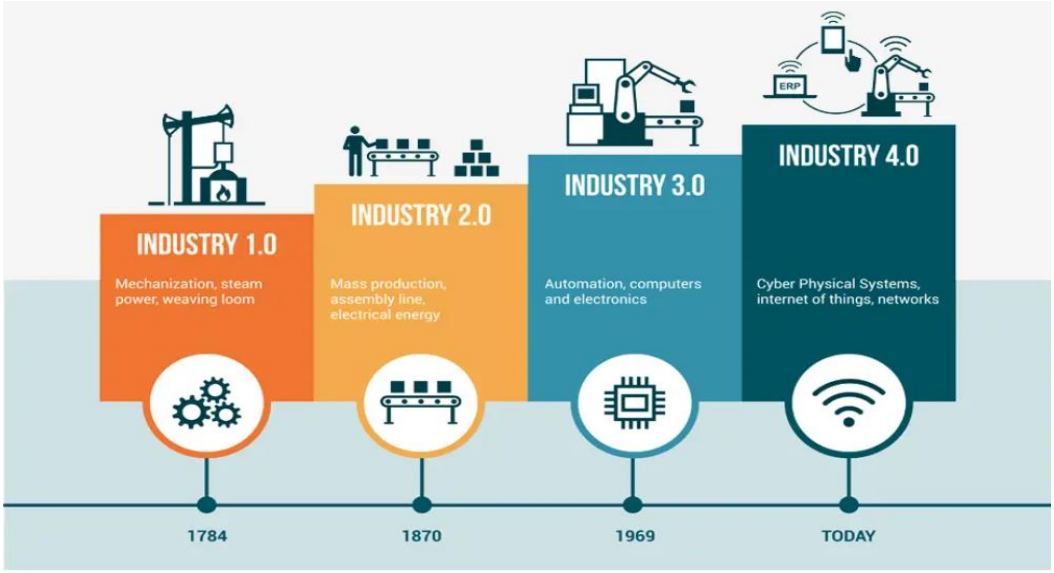


Figure 1: The four industrial revolutions (source: rematarlazzi.co.uk)

Many academics view Industry 4.0 as an evolution of the Third Industrial Revolution because it not only focuses on automating production processes, but it also introduces a new factory model based on the digitalization, integration, and connectivity of processes, not only within the company, but throughout the entire value chain, by combining information and digital technologies with production technologies. This results in a new approach to industrial production, no longer viewed as a series of separate steps, but as an integrated flow.

The concept of Industry 4.0 originated in Germany and was formalized in 2011 during the Hannover Messe [2]. It is an industrial strategy that involves a series of quick transformations in the design, production, operation, and service of production systems, transforming the entire industrial production sphere through the fusion of digital technology and the internet with traditional industry. Although Industry 4.0 is a widely recognized phenomenon in the industry, there is no clear-cut definition of it. Alternative terms used to describe it include Cyber Physical Systems, Internet of Things and Services, Smart Manufacturing, and Industrial Internet [1]. According to one of the most widely accepted definitions, Industry 4.0 refers to the Fourth Industrial Revolution, a new phase in the organization and management of the entire value chain during the product life cycle. This cycle is becoming increasingly customer-focused and covers everything from design and development to delivering the product to the end customer and its recycling, including all related services. All elements involved in the value chain are connected, making relevant information available in real-time and allowing for added value to be derived. Connecting people, objects, and systems creates dynamic, self-organizing, inter-company networks that can be updated in real-time and optimized based on various criteria, such as cost, availability, and resource consumption [3].

The key characteristics of the Industry 4.0 paradigm are the transformation of the value chain through digitization and interconnection of all its stages, from procurement to after-sales service. This transformation is driven by new technologies and devices, both hardware and software, that enable high levels of connectivity between people, devices, and machines, creating integrated systems that can respond autonomously and promptly to any changes in external conditions [3]. Data and information are transmitted in real-time within the company and between several companies within the same value chain, leading to a transition from a reactive to a proactive production system.

1.2 Enabling technologies

Key enabling technologies (KETs) are the backbone for implementing Industry 4.0. As defined by the European Commission, these technologies are knowledge-intensive and require high levels of R&D, rapid innovation, significant investment, and highly skilled jobs [4]. To fully leverage the potential of these technologies and create efficient and effective smart factory solutions, it is crucial to encourage synergies between them. The value generated by Industry 4.0 technologies can only be realized when they are considered as a whole, rather than as individual components. Therefore, the ability to derive synergies from the integration of these technologies is more important than their individual use. The following sections will provide an overview of the main enabling technologies of Industry 4.0, as recognized by academic and industry experts.

1.2.1 Internet of Things

The Internet of Things (IoT) was first introduced in 1999. It was defined by Kevin Ashton of MIT as a network of sensors and devices embedded in products, machinery, and other components that are connected to corporate information systems through the Internet. This technology expands the concept of communication by enabling interaction between machines, operators, processes, and sub-processes [5].

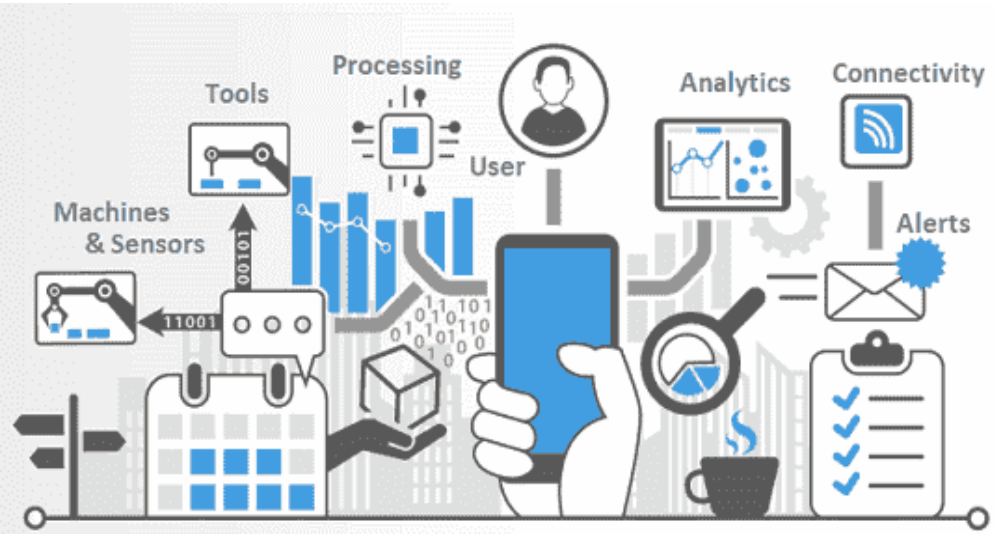


Figure 2: Industrial Internet of things (source: rocknetworks.com)

The foundation of IoT technology lies in the ability for each device in the network to have a unique identification through an IP address, enabling it to exchange data with other devices or software automatically using communication standards and protocols [1].

1.2.1.1 Industrial applications

The Internet of Things (IoT) technology has the potential to revolutionize various sectors, including robotics, automotive, home automation, telematics, smart cities, and remote control. Logistics, in particular, is well-suited for IoT applications, given the need for efficient information exchange. One example of IoT application in logistics is RFID (Radio Frequency Identification), which uses radio frequency transmission for unique object identification and data storage [6]. This technology enables product traceability throughout the production and assembly phase and helps operators keep track of processing and any problems that may have arisen during the production cycle. The IoT also enables timely detection of system malfunctions through sensors installed on machines, enabling predictive maintenance policies rather than reactive ones [7].

1.2.1.2 Challenges and open issues

Additionally, the security of IoT devices is often a concern as they can be vulnerable to hacking and cyberattacks. This highlights the need for proper security measures and protocols to be put in place, such as encryption, firewalls, and authentication mechanisms, to protect the sensitive data being transmitted and stored [7]. Furthermore, privacy is also an issue, as personal data and information may be collected and shared by IoT devices, which can raise privacy concerns. Therefore, it is important to ensure that privacy policies are in place to protect the privacy of individuals, and that the collected data is only used for the intended purposes. In conclusion, the potential benefits of the IoT for industry are vast, but there are also important challenges that must be addressed to ensure its sustainable and responsible development.

1.2.2 Big Data

Over the past few decades, the amount of data available to businesses has increased exponentially and there is a growing recognition of the significance of data as a source of value for creating new business models. Big Data encompasses "the volume of digital data available in the individual, physical, and industrial environment from sensors, machinery, IT infrastructure, mobile devices, electronic control units, telecommunications equipment" [8]. In 2001, Dug Laney presented a model called the 3V model, which highlights the unique features of Big Data, concentrating on the variables on which this technology develops [8]. These variables are:

- **Volume:** This refers to the size of databases employed to store data. These are expanding rapidly and necessitate high hardware capability and intricate algorithms to manage. The quantity of data kept has increased from terabytes to zettabytes (1 trillion bytes).
- **Variety:** The type of data accessible to businesses is no longer uniform but encompasses textual data, images, audio, and video. Integrating structured data (usually numerical) with unstructured data (text, images, video) necessitates specific, newly generated platforms and people capable of making them functional and effective, and ideally suited to the business's objectives.
- **Speed:** This pertains to the rate at which data is transmitted. A diminished speed of information gathering and processing would make companies reluctant to purchase data packages that include potential consumers or current customers. The same applies to computer security companies, where a data stream must be studied and analyzed swiftly to identify new cyber threats' presence or absence. Consequently, the speed of data transmission and analysis is a critical factor in the success of Big Data.

Today, Laney's paradigm has been enriched with two more variables, leading to the emergence of the so-called 5V Model of Big Data [8]. The added variables are:

- **Value**: it refers to the need for industries to appropriately store, classify, and analyze data to extract value and identify new business opportunities, due to the impact of digitization across all sectors.

- **Veracity:** it highlights the challenge of ensuring the reliability of data in the context of Big Data, where data management technologies, collection speeds, and sources are constantly changing. Despite these challenges, ensuring the quality and integrity of data is essential for creating useful and reliable analyses.

Given the preceding discussion, it can be argued that the success of Industry 4.0 hinges on the ability to perform real-time analysis and intelligent processing of vast amounts of data from diverse sources. To accomplish this, it is crucial to create algorithms that can efficiently and effectively store, analyze, and transmit the information contained in Big Data. By doing so, Big Data can become a valuable decision-making tool for companies.

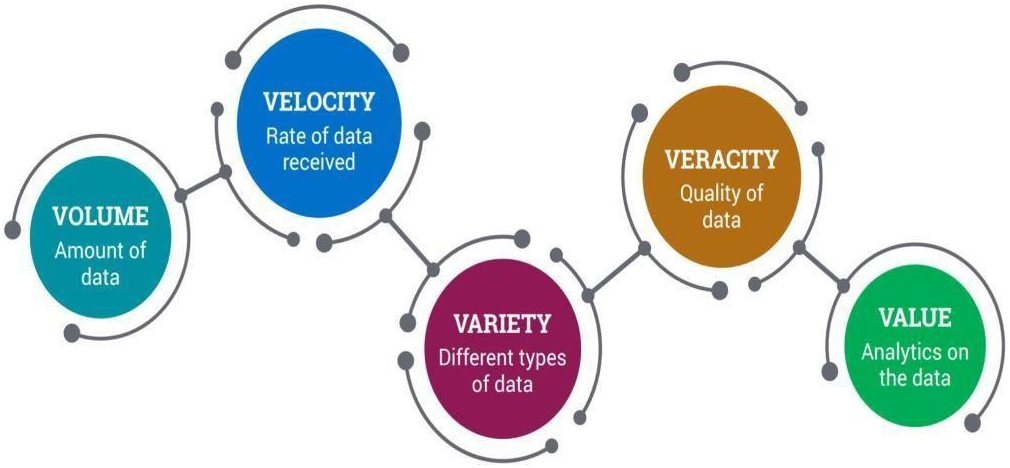


Figure 3: 5V Big Data (source: onlinelibrary.wiley.com)

1.2.1.3 Industrial applications

Big Data is a fundamental element of information management in Industry 4.0. It involves collecting and processing data from sensors and networked control units, enabling the monitoring of machines and plants to identify trends and anomalies. Data Mining and Data Analytics techniques allow for the automatic monitoring of systems and the generation of reports and KPIs. Data analysis algorithms are also useful for production quality control, enabling companies to identify anomalies, minimize errors and improve production processes. The implementation of Big Data represents a significant change in the way production processes are managed.

1.2.3 Cloud Computing

Cloud computing relies on a range of technologies that enable the processing and storage of data using distributed hardware and software resources across a network [9]. The key elements of a cloud platform include:

- A sophisticated data storage architecture;
- Computing nodes responsible for data processing;
- Controllers, for data migration.

When discussing cloud technology, there are three types of clouds: public, private, and hybrid. A public cloud is a service that provides infrastructure through a public network and data centers managed by third parties over the internet. A private cloud is a service that provides infrastructure and services through a private network and data centers managed internally by a company that restricts access to others. A hybrid cloud is a

combination of public and private clouds. Additionally, cloud services are categorized into three main types, including IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service) [9].

- SaaS is "a model that encompasses applications and software systems, accessible from any type of device (computer, smartphone, tablet, etc.) through the simple use of a client interface. In this way, the user does not have to worry about managing the resources and infrastructure, as they are controlled by the provider who supplies them.

- PaaS is "a model in which the services of online platforms are located, thanks to which a user, usually a developer, can carry out the installation of applications and web services that he or she intends to provide. In this case, the user can develop and run his or her own applications through the tools provided by the provider, who guarantees the proper functioning of the underlying infrastructure. (e.g. Google Cloud App Engine, Amazon dynamoDB, Google cloud data store).

- IaaS is a 'model in which virtualised hardware resources are made available so that users can create and manage their own infrastructure on the cloud according to their needs, without worrying about where the resources are allocated.

This means that the services provided by SaaS, PaaS, and IaaS differ based on their respective functionalities: IaaS provides support for PaaS, which in turn supports SaaS.

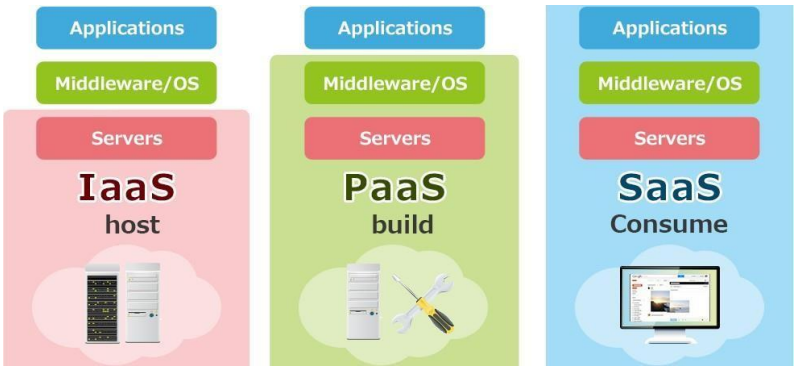


Figure 4: IaaS-PaaS-SaaS (source: vitolavecchia.altervista.org)

1.2.4 Fog Computing

Fog Computing refers to local data processing in a decentralized manner, as opposed to Cloud Computing which relies on a network of connected computers and servers [9]. This infrastructure can offer accurate predictions and become a valuable tool for planning production processes. Simulation, in general, involves modeling real systems over a specified time frame to predict their performance under various constraints and conditions. Designers can use simulation tools to evaluate product or machinery performance early in development, identifying and addressing issues before they become costly problems. This approach can lead to significant reductions in product development costs.

1.2.5 Digital Twin

The concept of Digital Twin, or Virtual Twin, is crucial in the field of simulation and prototyping. It refers to a virtual real-time representation of a physical device that can reproduce its behavior based on external stimuli perceived by sensors in the real system [7]. Working with a Digital Twin has the advantage of allowing simulations to be carried out to evaluate how the real model would react to changing boundary conditions, thus minimizing anomalies. Additionally, the virtual model enables remote operations to be performed, as it can be analyzed from anywhere, irrespective of the location of the physical device. This can lead to scenarios where technical staff can give instructions to operators working on a distant machine by analyzing its virtual twin.

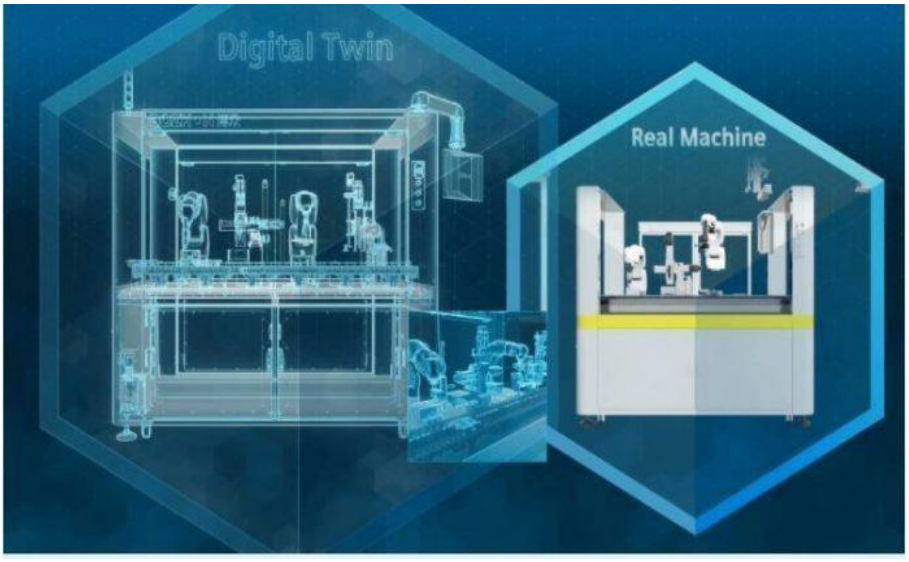


Figure 5: Digital Twin (source: industry.itismagazine.co.uk)

1.3 Impact of Artificial Intelligence in Industry

In modern business, companies strive to become more efficient by reducing costs and minimizing errors, while improving product performance. Before exploring the role of artificial intelligence (AI) in manufacturing, it's important to understand its general definition. Professor John McCarthy, an expert from Stanford University, defines AI: “It is a science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable” [10].

Thanks to the benefits of artificial intelligence and related technologies, companies are able to achieve higher levels of efficiency that can drive the entire business. This is possible because AI can identify inefficiencies in the production process or at the end of the production line, extract valuable manufacturing data, facilitate communication between machines and people, and reduce costs. The new manufacturing model, known as intelligent manufacturing, is based on AI, science, and technology and can lead to growth in design, production, and management. The main goal of intelligent manufacturing is to optimize manufacturing resources, improve business value, and enhance safety. Within the sector of intelligent manufacturing, different types of maintenance have been developed to prevent failures and increase reliability, including corrective, preventive, and predictive maintenance, depending on available resources.

1.3.1 Artificial intelligence division

The primary categorization of AI can be grouped into three categories, which include internal subsets of machine learning and deep learning [11]. Currently, there is a common misconception where the definitions of AI, ML, and DL are not accurately comprehended or applied, leading to confusion. To avoid this, the initial stage of this chapter will differentiate between various types of AI, followed by an in-depth analysis of each segment.

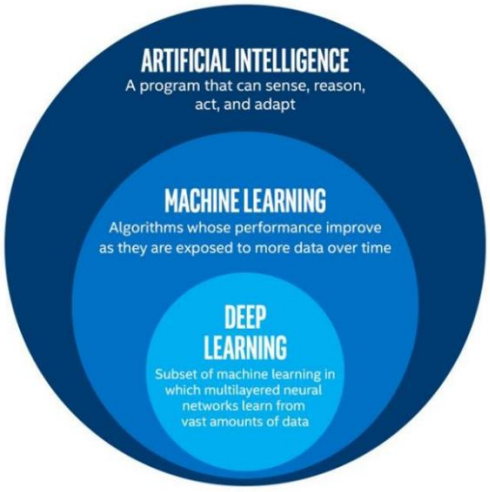


Figure 6: Division of Artificial intelligence (source: medium.com)

1.3.2 Types of AI: weak, strong and super intelligence

Weak AI, also known as Narrow AI or Artificial Narrow Intelligence (ANI) [12], is designed to perform specific tasks that do not match human intelligence and do not attempt to replicate it. The term "narrow" accurately describes this type due to its limited range of abilities, which are mostly goal-oriented for performing various tasks. This type is commonly used today in environments that enable robust applications.

Strong AI is made up of Artificial General Intelligence (AGI) and Artificial Super Intelligence (ASI) [12]. AGI is a form in which machines have intelligence equal to that of humans, or in other words, self-aware consciousness with the ability for problem-solving, learning, and planning for the future. Analyzing this type, it is possible to imagine human-like robots that can decide by themselves and learn without human intervention.

Additionally, there is another type called Artificial Super Intelligence (ASI), which is a type of self-conscious computer that can practically outmaster human intelligence and the abilities of the brain. However, this type is still mostly theoretical or treated as a research topic.

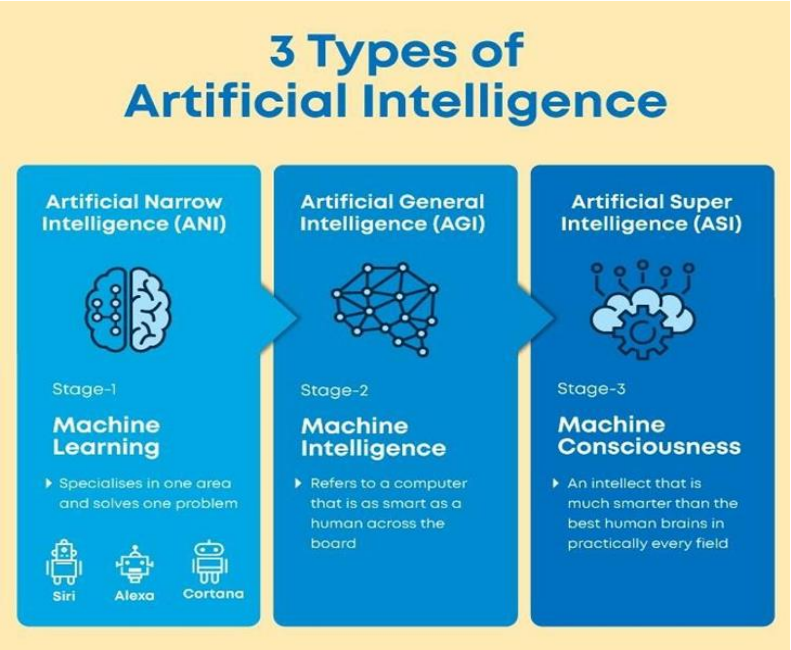


Figure 7: Three main types of Artificial Intelligence (source: medium.com)

1.4 Machine learning

Machine learning is a subfield of artificial intelligence that involves the use of algorithms to improve themselves based on experience and data. It is also referred to as the study of enabling machines to perform better without explicit programming. When a machine learns, it changes its structure, program, or data in response to input or external information with the expectation of further improvement [13]. To facilitate learning, three components are required: an algorithm, a dataset, and features. The dataset is a collection of similar entities and values that can be individually or collectively structured. Collecting this data can be a time-consuming operation, and it can be done by gathering open-source data or directly collecting it. Features are special data objects that are necessary for solving the problem at hand, and their selection is essential for effective pattern recognition, classification, and regression algorithms. Algorithms aim to allow the system to learn and develop its own experience without being explicitly programmed. There are various algorithms, such as deep learning, coevolutionary networks, and recommendation systems, each with a specific use.

1.4.1 Types of Machine Learning

There are four main types of machine learning, which can be classified based on the type of learning expected from the algorithms. These include Supervised Learning, Unsupervised Learning, Semi-supervised Learning, and Reinforcement Learning [13].

More details on these types of machine learning will be explained in the following sub-sections.

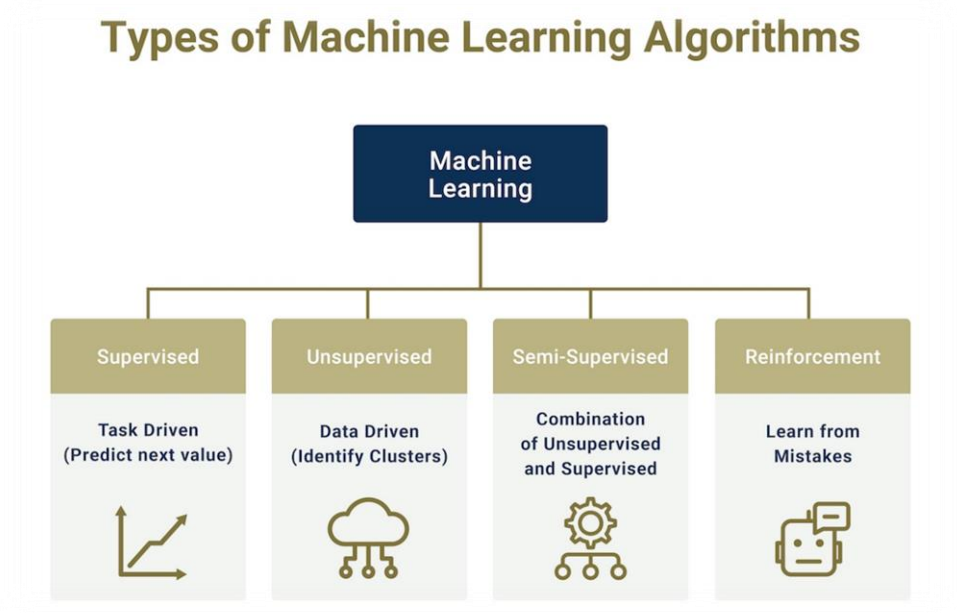


Figure 8: Types of Machine Learning (source: bootcamp.pe.gatech.edu)

1.4.2 Supervised Learning

The first type of machine learning algorithm is characterized by the output of a classified or labeled dataset that can be accurately predicted. When input data is applied to the model, it is adjusted and weighted until the model fits appropriately, avoiding overfitting or underfitting. Supervised learning involves providing labeled examples, each with a unique input signal and corresponding response, to teach the model [13]. Linear regression is an example of an algorithm that uses supervised learning, as do other machine learning algorithms such as K-Nearest Neighbors, Support Vector Machines, Decision Trees and Random Forests, and Artificial Neural Networks. Supervised learning models have found application in various areas thanks to their accurate and predictable output. Examples include predictive analytics, customer sentiment analysis, spam detection, and image and object recognition, which will be more fully explained in subsequent chapters. The two primary prediction problems of machine learning derived from supervised learning are regression and classification, which differ as follows:

- Classification involves finding a model or function that can separate data into multiple categorical classes based on given input parameters. In classification, data

is categorized under different labels, and then the labels are predicted for new data [13]. The resulting mapping function could be represented in the form of "IF-THEN" rules. Classification deals with problems where data can be divided into binary or multiple discrete labels.

- Regression involves finding a model or function for distinguishing data into continuous real values, and it can also identify distribution movement based on historical data [13]. Since a regression predictive model predicts a quantity, the accuracy of the model must be reported as an error in those predictions.

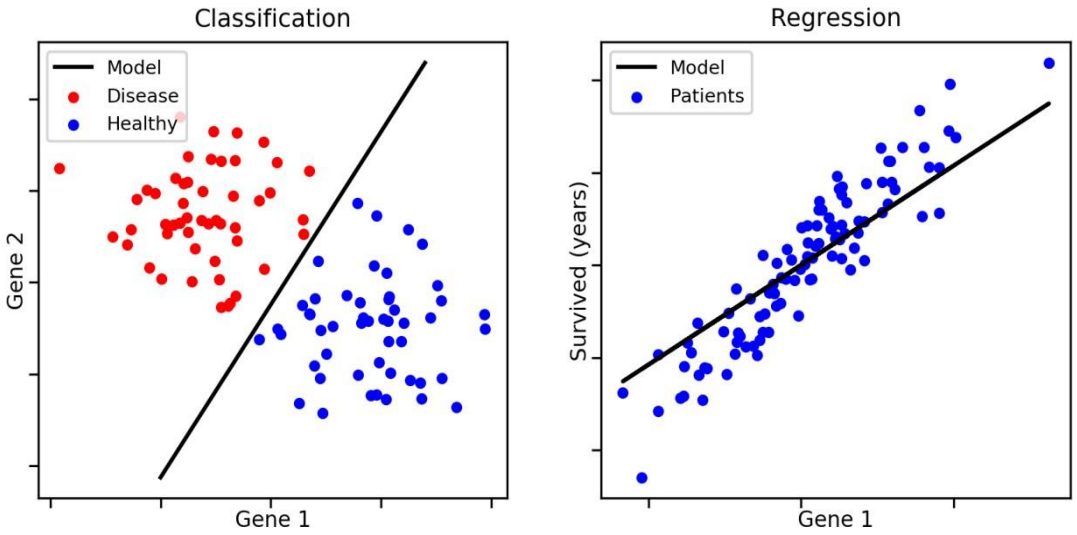


Figure 9: Difference between scope of Classification and Regression

As can be partially seen from the graph above, classification is a process that predicts discrete values in an unordered manner, while regression works with continuous values and predicts data in an ordered manner. Examples of classification algorithms include decision trees or logistic regression, while regression solutions include regression trees or linear regression, among others.

1.4.3 Unsupervised Learning

Unlike the previously explained, Unsupervised Learning does not require the user to know the final output. The model works on its own, without supervision or access to labeled output, allowing it to discover previously undetected patterns and information [13]. The main advantage of using UL is the ability to find all kinds of unknown patterns in data. Other machine learning algorithms that use this type of learning include K-Means for clustering, t-distributed Stochastic Neighbour Embedding (t-SNE) for visualization,

Principal Component Analysis for dimensionality reduction, and Apriori for association rule learning. In the manufacturing industry, a useful method of unsupervised learning is anomaly detection, where the model is trained with standard data to detect whether new data is normal or not, potentially detecting anomalies in machines or processes.

The three main tasks of unsupervised learning are clustering, association, and dimensionality reduction [13]. Clustering methods are used to find similarity and relationship patterns among data samples and cluster them into groups based on their features. Clustering plays an important role in determining groups among present unlabeled data and making assumptions about data points to determine their similarity.

1.4.4 Semi-supervised Learning

Semi-supervised learning is a type of ML that uses a combination of labeled and unlabeled data to train a model. It lies between supervised and unsupervised learning, and is useful in situations where there is a limited amount of labeled data and no additional resources to obtain more [13]. By using the large amount of unlabeled data available, the model can learn more about the data and improve its accuracy. This technique can be especially helpful in domains such as natural language processing and computer vision, where labeled data can be difficult and expensive to obtain.

1.4.5 Reinforcement Learning

The goal of Reinforcement Learning is to address closed-loop problems in which the actions of the learning system affect its future inputs. In this type of learning, the learner does not receive direct instruction on which action to take, but instead must discover it through trial and error [13]. In some cases, actions may have consequences not only in the immediate response but also in subsequent responses. Reinforcement Learning is characterized by three main factors: closed-loop, lack of direct instruction on which actions to take, and the consequences of actions that are important for an extended period of time [13]. One of the challenges of Reinforcement Learning is balancing exploration and exploitation. To obtain a large reward, an agent must prefer actions that have been effective in the past, but to discover such actions, it has to try new things. Another key aspect of Reinforcement Learning is its consideration of the entire, goal-directed problem interacting in an uncertain environment. Finally, one of the most interesting aspects of modern Reinforcement Learning is its productive interaction with other engineering and scientific disciplines, such

as psychology or neuroscience.

1.4.6 Deep Learning

Deep learning, also known as deep structured learning or hierarchical learning, is a newly emerged field and subcategory of research within machine learning. Rather than providing a thorough explanation right away, it's important to first provide a general definition in order to convey the overall concept of this particular area. “Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence. Deep Learning is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound and text” [14].

Deep learning is an essential component of cognitive computing, which has gained widespread interest due to its ability to enable applications to understand human input and respond in a way that people can comprehend. Deep learning technology has significantly enhanced computers' capacity to classify, identify, detect, and describe data, which means they can better understand it. This technique is particularly useful in areas such as image classification, speech recognition, and object recognition, where numerical data is essential. With the development of new algorithms and technologies, deep learning is rapidly advancing, with increased accuracy and effectiveness. As a result, new fields of study such as text translators and image classification are becoming increasingly useful [14].

However, the complexity of the algorithms and the vast amounts of data required for network learning mean that solving problems using deep learning requires significant computing power. The adaptive nature of the learning method allows for continuous self-improvement and adjustment, which makes it possible to implement more dynamic solutions. The increased accuracy and efficiency of neural networks in applications that have long used them is also contributing to their continued growth. With better algorithms and higher computational power, it is possible to gain a more precise understanding of the data being analyzed.

1.5 Types of maintenance in machine learning

Maintenance costs are a significant operating expense for the industrial sector, and can represent up to 50% of total production costs even without factoring in unplanned downtime [15]. As a result, a company's productivity and profitability depend heavily on the

maintenance process, so it's important to ensure that all equipment is working as reliably as possible. This article describes and compares three main types of maintenance: corrective, preventive, and predictive, as illustrated in Figure 10.

• **Corrective maintenance:** The first type of maintenance is focused on solving problems that have already occurred, and only takes place when the machine is in critical condition. This typically leads to production stoppages, resulting in reduced output and increased costs [15]. Repair time cannot be easily scheduled, so this time is often used for machining processes that do not have a significant impact on production. Corrective maintenance is typically performed by a human technician.

• **Preventive maintenance:** The second type of maintenance is preventive, and focuses on scheduled maintenance actions aimed at preventing spontaneous breakdowns, failures, and degradation of equipment, components, or spare parts. This type of maintenance is usually carried out outside of production time and is performed through periodic inspections of the system [15]. It is useful to keep a history of previous failures of parts or machines for more effective planning. Preventive maintenance is considered demanding, requiring precise supervision and planning, and should be carried out by qualified personnel. Incorrect application of preventive maintenance may lead to breakdowns, resulting in production downtime and increased costs.

• **Predictive maintenance:** This type of maintenance is not new to the industry but has gained relevance and has been developed in the Industry 4.0 scheme. In this case, maintenance is usually carried out before a breakdown occurs. It is based on precise formulas, sensor measurements, and analysis of measured parameters. The main goal is to guarantee maximum intervals between consecutive repairs and minimize the number of planned maintenance operations and associated costs. The predictive maintenance process involves three steps: data acquisition, data processing, and machine decision-making [15].

Figure 10: Types of Maintenance (source: www.heavy.ai)

Chapter 2 - Turning Process and Surface Roughness

Synopsis

This chapter focuses on metal cutting processes and provides a detailed explanation of the turning process, turning cutting tool, and the wear phenomena associated with cutting tool usage. It also provides an overview of Surface Roughness parameters and explains how the Surface Roughness is affected during Turning processes.

2.1 Metal cutting

Cutting is recognized as the earliest manufacturing technique ever invented, and it has a broad range of applications. However, this paper concentrates on metal cutting, which involves removing a thin layer of metal, known as a chip, from a larger body called the workpiece, using a wedge-shaped tool. In engineering, the term "cutting" or "machining" refers to chip-forming processes, which are mainly used in metal processing. Although these processes can be used on other materials like wood and plastic, the knowledge gained from metal processing cannot be completely applied to other materials [16]. As a result, in this text, the terms "cutting" and "machining" are interchangeable and are primarily associated with metal cutting. Metal cutting or machining is part of a general manufacturing hierarchy chain and is considered a secondary process, as depicted in Figure 11.

The fabrication of a product is typically broken down into three steps: primary shaping processes, which are responsible for the initial transformation of the raw material; secondary processes, which include material removal and other treatments such as heat and surface treatments; and finally, assembly processes, which involve joining, assembly, and testing of the product [17]. Primary shaping processes such as casting and forming shape the workpieces while conserving their masses. However, these processes may not provide sufficient surface quality and geometric tolerances. Therefore, material removal processes are necessary to finish the workpieces, and metal cutting processes fulfill this need [18].

Machining refers to various processes, all of which involve removing material from a workpiece in their unique ways. Broadly speaking, some processes utilize tools with defined geometry (such as milling, drilling, and turning), while others use tools with non-defined geometry (like grinding, honing, and lapping) [19]. Other material-removal-based

processes have more specific applications, such as electrical discharge and ultrasonic machining. However, in this review, the focus is on turning processes.

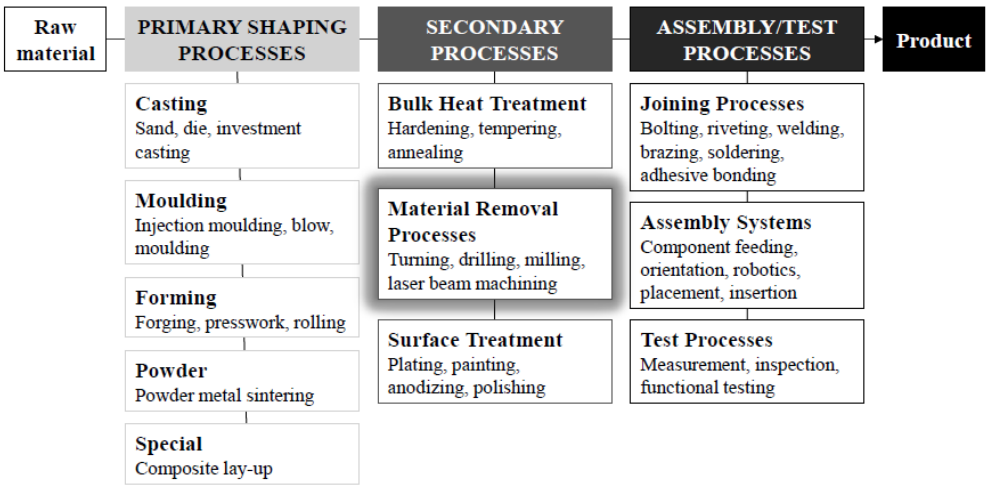


Figure 11: Hierarchy of manufacturing processes. (SWIFT; BOOKER, 2013).

2.2 Turning process

The turning process involves the use of single-point cutting tools with a geometrically defined cutting point. This means that there is only one theoretical cutting point. During machining, the cutting region of the tool is continuously in contact with the workpiece, resulting in a continuous cutting process [20]. Turning is primarily a rotational process, where the workpiece rotates while a translational feed motion removes material from it [21]. A typical single-point cutting tool is depicted in Figure 12. It consists of a major cutting edge, also known as the primary cutting edge, which forms the chip that flows through the rake face. The transient surface formed by the major cutting-edge flows through the flank face of the tool, as shown on the right side of Figure 13. The primary cutting edge physically divides the flank and rake faces.

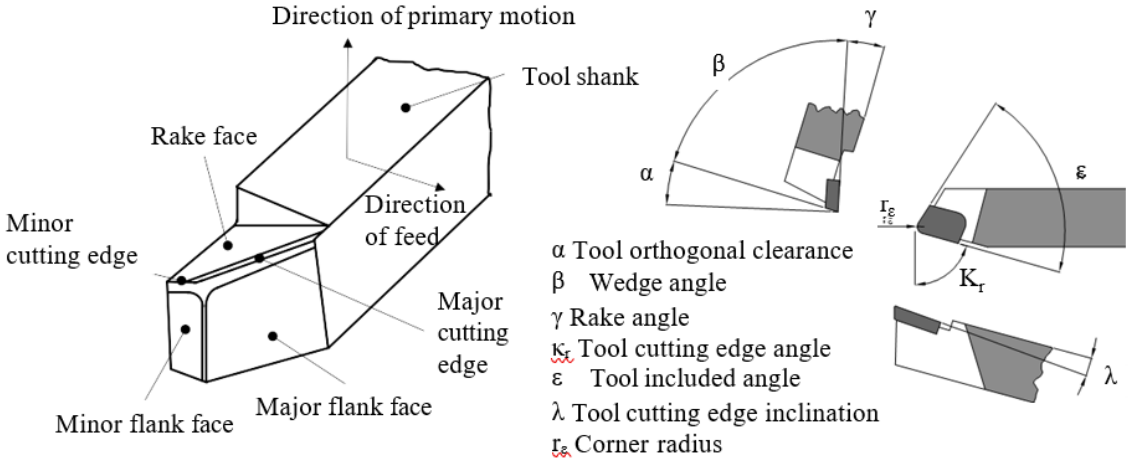


Figure 12: Cutting tool nomenclature and main angles. (KLOCKE, 2011; TSCHÄTSCH, 2009)

The geometry of the cutting tool can have a significant impact on the machining process by affecting various aspects such as forces, chip formation, surface roughness, and tool durability. Each angle or inclination of the tool has distinct characteristics that should be considered as a design parameter for the machining process [22]:

- The clearance angle α, can help reduce or avoid friction between the transient surface and the flank face of the tool, which can greatly improve tool lifespan.
- The rake angle γ is another crucial angle to consider, as it has a significant influence on forces, power requirements, surface finish, and heat generation.
- The cutting-edge inclination λ also play a role in directing the chips and controlling vibration
- The cutting-edge angle κr can reduce chatter, a type of harmful self-excited vibration that can lead to tool wear and increased forces.
- The corner radius rε, which refers to the radius of the curve that unites the minor cutting edge to the major cutting edge, can increase power demands by up to 20%. It is also a significant factor in most theoretical surface roughness models.

In addition, the selection of geometric features of the cutting tool must be based on the specific characteristics of the workpiece and tool materials used in the process, as these factors can greatly influence the machining performance [22].

Figure 13 illustrates a typical lathe machine used for turning operations, as well as its main components and axes. The cylindrical turning process is the most commonly used turning process and is depicted on the right side of the figure. The workpiece is clamped on the chuck, which is connected to the motor spindle that rotates the part. For heavier or longer

parts, the center-tailstock can be used to support both ends. The feed motion is achieved by moving the carriage along the feed rods and lead screws. The machine tool axes are conventionally designated as follows: the Z-axis aligns with the spindle direction, the X-axis is parallel to the longest direction of table movement, and the Y-axis is parallel to the shorter direction of table movement. A, B, and C correspond to angular movements around the X, Y, and Z axes, respectively [23]. This schematic fixture and axes system can be entirely translated into CNC (Computer Numerically Controlled) turning centers.

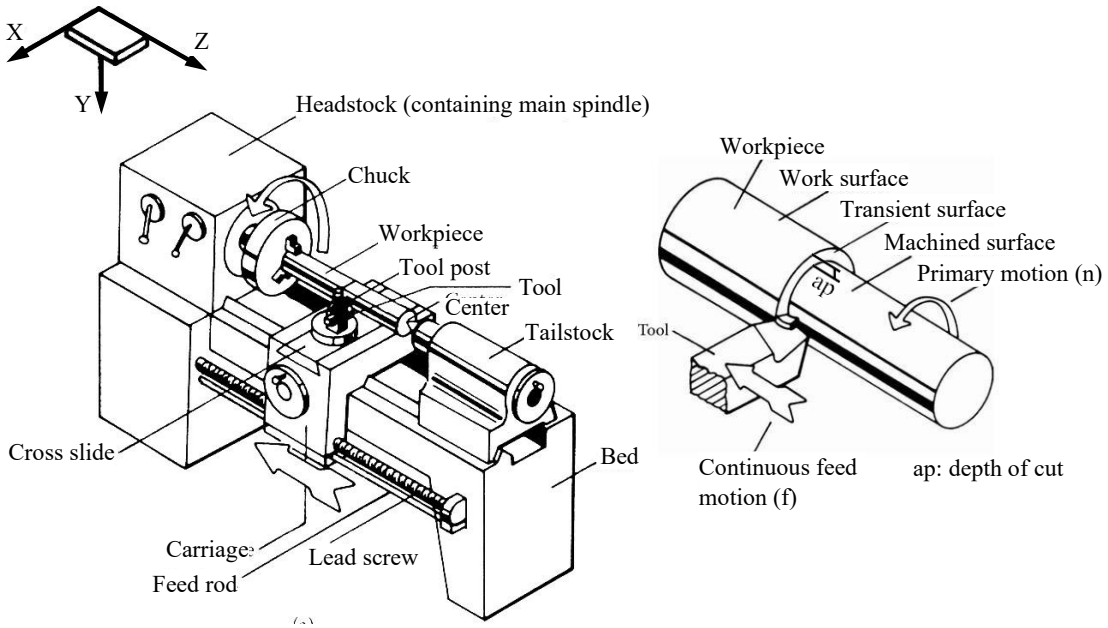


Figure 13: Lathe (left side) and cylindrical turning process (right side).

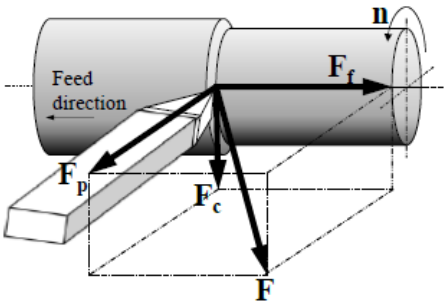
The cutting process in cylindrical turning can be analyzed using mathematical equations, which are presented in Table 1. One such equation is the cutting time equation, which can be used to calculate the time taken for a given process and relate it to measures of tool wear. Another useful equation is the cutting speed equation, which can be calculated by hand and used during the execution of experiments. The relationship between the width of cut, uncut chip thickness, and the sectional area of the chips can also be used to investigate the forces involved in the process.

Table 1: Some mathematical relations of cylindrical turning.

Item	*Mathematical relation*	*Unit*	*Eq.*	*Source*
Cutting time	$T_c = \frac{L}{f \cdot n}$	min	1	(BOOTHROYD; KNIGHT, 2006)
Cutting speed	$v_c = \pi \cdot d \cdot n$	m/min	2	(TSCHÄTSCH, 2009)
Width of cut	$b = \frac{ap}{\sin(\kappa r)}$	mm	3	(STEMMER, 1993)
Uncut chip thickness	$h = f \cdot \sin(\kappa_r)$	mm	4	(STEMMER, 1993)
Chip's sectional area	$A = a_p \cdot f = b \cdot h$	mm²	5	(TSCHÄTSCH, 2009)

2.3 Machining forces

Understanding the forces involved in metal cutting processes is crucial for various applications. These forces can inform the design of machine tools, cutting tools, and tool holders, as well as aid in the determination of cutting conditions. Furthermore, analyzing the forces can help identify wear mechanisms, monitor the process for tool wear or breakage, and classify materials based on their machinability [16]. Machinability refers to the ease with which a specific material can be cut under specific conditions, with larger forces typically indicating a harder-to-cut material.



F : Resultant force
F_c : Cutting force
F_f : Feed force
F_p : Passive force

Figure 14: Resultant force and its components for a general turning process (TSCHÄTSCH, 2009).

Figure 14 illustrates the forces involved in a typical turning process. The forces acting on the tool are of the same magnitude as those acting on the chip, but in opposite directions. The resultant force F can be broken down into three components that affect the tool [21]:

- The cutting force F_c : acts in the opposite direction of the cutting motion and perpendicular to the cutting edge. It is usually the most significant force in turning processes.
- The feed force F_f: acts in the opposite direction of the feed motion (or perpendicular to the cutting motion). It is also known as the thrust force F_t.
- The passive force F_p : acts in the same direction as the tool holder and tends to push the tool away from the workpiece. It is often neglected in turning operations because it is typically the smallest of the three components.

The force components can also be named after the names of the machine tool axes: F_c is equivalent to F_x because it acts along the X-axis; F_f is equivalent to F_z because it acts along the Z-axis; F_p is equivalent to F_y because it acts along the Y-axis. Alternatively, they can also be named based on their direction with respect to the workpiece: F_c , F_f, and F_p are also known as tangential force, axial force, and radial force, respectively [24].

The cutting force is the primary force used to assess cutting processes, especially when the other two components are insignificant. However, the specific cutting force, resultant force, and specific resultant force may also be utilized. Since the cutting force is typically the highest among the three components, it is used to compute the machining power required. This calculation is simply the product of the cutting force and cutting speed [24].

Dynamometers are used to measure machining forces, and various dynamometers have been developed over the past century, including force platforms that employ piezoelectric load cells to measure force in multiple directions and calculate resultants [16]. Piezoelectric dynamometers are known for their high static stiffness and rigidity, high natural frequency, high sensitivity, accuracy and reliability, and low temperature sensitivity, which produce accurate results [25]. The force measuring unit is the Newton (N).

The resultant force is the vector sum of the three components and can be determined using Equation 6, which is essentially a sum of orthogonal vectors employing the Pythagorean Theorem.

$$F = \sqrt{Fc^2 + Ff^2 + Fp^2} \qquad \text{Eq.6}$$

2.4 Cutting tool wear

During the metal cutting process, the cutting region, including the workpiece and cutting tool, is subjected to external thermal and mechanical loads [26]. These loads result in

deformation, separation, and friction in the cutting-edge area of the tool, creating a complex and challenging environment. The constant high compressive and thermal stresses cause the tool to wear over time, making it ineffective in cutting efficiently or even causing it to fail altogether. Tool wear is characterized by changes in the tool's shape due to the loss of material or deformation (ISO,1993), which can lead to poor quality final parts and defects. When designing and implementing metal cutting processes, it is important to consider the wear behavior, its causes, and possible solutions. Tool wear typically occurs through three primary mechanisms: abrasion, adhesion, and diffusion [27].

- Abrasion occurs when hard particles on the chip and workpiece mechanically remove material from the tool. These particles can be leftover strain-hardened material, previously removed fragments, or hard workpiece components.
- Adhesion is caused by friction movement between the tool and workpiece, which creates micro-welding between their surfaces. This causes tool material to be carried away, and the mechanism is influenced by the tendency of materials to bond, which can be caused by mechanical action or chemical interactions.
- Diffusion is caused by high temperatures in the cutting region, which trigger individual atoms to migrate from the tool surface to the workpiece or vice versa. This mechanism affects a narrow region of the contact interface between the tool and workpiece but is critical in reducing tool resistance to abrasion, particularly when the diffused atoms are essential constituents of the tool's alloying elements.

The causes of tool wear are closely related to the cutting temperature, which is strongly influenced by the cutting speed. Figure 15 illustrates how the wear mechanisms change with increasing cutting temperature. At low temperatures, the dominant wear mechanisms are adhesion and abrasion. As the temperature rises, diffusion and oxidation become more important. At very high temperatures, diffusion becomes the primary cause of tool wear.

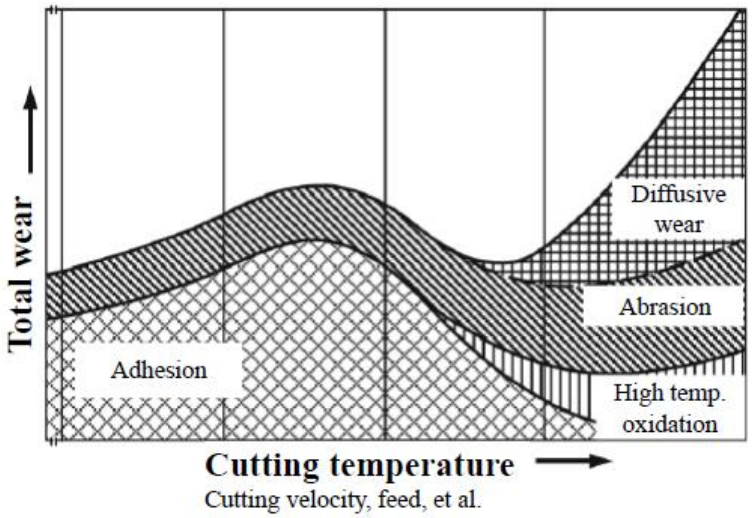


Figure 15: Wear causes dependent on cutting temperature (KLOCKE, 2011)

One practical method for determining when a cutting tool has reached the end of its useful life is by observing certain anomalies during cutting, such as the presence of smoke, excessive noise, unusually severe vibrations, changes in the surface finish or geometry of the workpiece, or changes in the dimensions of the cutting tool itself. Experienced operators are often relied upon to identify when the tool needs to be replaced or reground. Wear affects the dimensions of the tool in various ways, as depicted in Figure 16 for a typical turning operation. The different forms of wear include flaking, fragmentation, plastic deformation, smearing, notching, cratering, and flank wear. Of these, particular attention should be paid to crater and flank wear because they tend to progress more gradually than the others, making it easier to track their development over time. Additionally, they are the easiest forms of wear to measure.

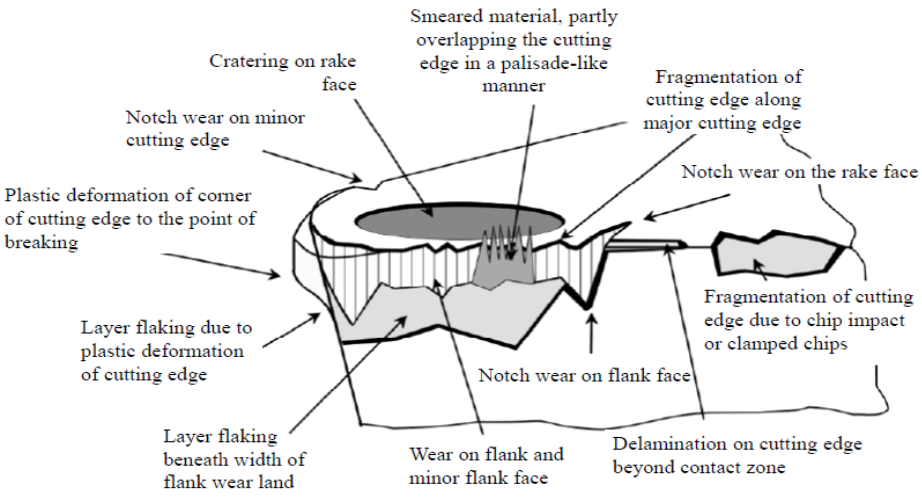


Figure 16: Forms of wear (KLOCKE, 2011).

Even though one can observe certain anomalies to determine whether a tool is broken or no longer cutting properly, it is important to replace the tool before it wears out completely to prevent negative consequences. Measures of wear are used to determine the extent of change in the tool's dimensions and can serve as a criterion for tool life (ISO 1993). These criteria vary depending on the type of tool, such as high-speed, sintered carbide, or ceramics. For sintered carbide cutting tools (as in this work), the criteria typically used are: the average width of the flank wear land $VB_B = 0.3$ mm (as shown in Figure 17) for regular wear, and the maximum width of the flank wear land VB_B max. = 0.6 mm if the wear is not regular. Additionally, criteria such as the depth of crater wear KT = 0.2 mm and the breakage of the crater can also be used. Wear measures are associated with visible changes in the tool's wear patterns during cutting, as shown in Figure 17, which displays the measures of wear defined by the International Standard ISO 3685(E) (1993), including the width of the flank wear VB_B, the depth of crater wear KT, and cutting-edge displacements (SV_α and SV_γ).

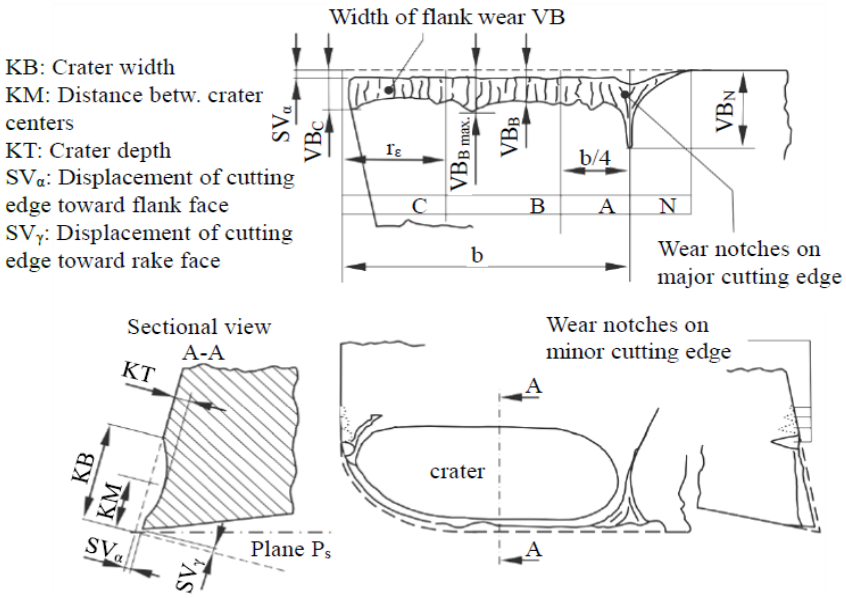


Figure 17: Tool wear measures (ISO, 1993).

The notch, indicated as VB_N in Figure 16, is not considered to be a form of abrasive wear like regions A, B, and C. Notch wear typically occurs as a result of chemical reactions and work-hardening of the workpiece surface from previous cuts. It affects both the flank and rake face of the tool, and it happens outside of the tool-workpiece interface (ISO, 1993). Figure 18 displays two types of wear: crater and flank wear. The rake face of the tool, where the interface between the chip and tool occurs, experiences wear in the form of a crater. Crater wear is the primary form of wear that is controlled in high-speed cutting, whereas under economic conditions [27], flank wear is typically the measure that is assessed when determining the condition of the tool. In this dissertation, the focus will be on assessing flank wear, which occurs at the interface between the tool and workpiece on the flank face of the tool. The dashed lines in Figure 18 represent the original state of the tool's rake and flank face prior to wear.

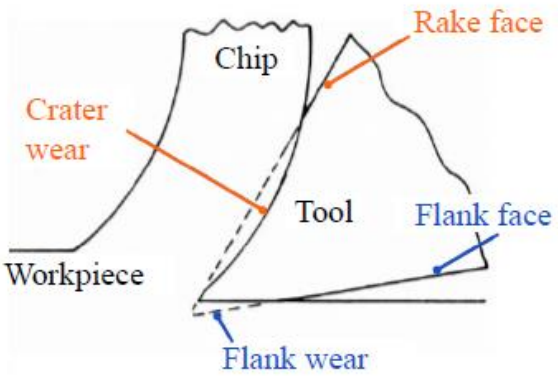


Figure 18: Most used forms to control tool wear (BOOTHROYD; KNIGHT, 2006).

In addition to wear measurands like the width of flank wear and the depth of crater wear, other tool-life criteria are also commonly used, including pre-defined limits on surface roughness, geometry deviation, cutting forces, and spindle power, among others. The summarized tool-life criteria are as follows [21]:

- Tool wear measurands (such as a limit on flank wear width);
- Workpiece result measurands (such as a limit on surface roughness);
- Process measurands (such as a limit on spindle electric power).

2.5 The machined surface topography

The surface of a machined part contains characteristics that are related to the part's function and performance, such as corrosion resistance, fatigue strength, and tribological behavior [28]. These characteristics are collectively known as surface integrity aspects, which are divided into external topography and internal aspects, including microstructure, mechanical properties, and residual stress. In this work, the focus is on the external topography aspect of surface integrity, specifically surface roughness. Using the definitions proposed by Davim (2010), a nominal surface is given by the original project design without considering any deviation, and a real profile is a resultant surface read by measuring assessment equipment (microscopes, profilometers, etc.) that considers the geometric deviations from the nominal surface, which are listed in Table 2. These deviations are classified into form and waviness errors, which are first and second orders, respectively, and roughness errors, which are third to sixth orders. The form, waviness, and roughness profiles are depicted in Figure 19, with the waviness profile (W-Profile) being the primary profile without roughness, and the roughness profile (R- Profile) being the primary profile without

waviness, obtained using low-pass and high-pass filters, respectively (ISO,1997).

Table 2: Geometric deviations of machined surfaces according to DIN 4760 (1982).

Order	*Deviation*	*Causes*
1st	Form errors	Errors of machine tool slides, elastic deformations, fixation of tool/workpiece, severe tool wear
2nd	Waviness	Eccentric rotation of workpiece/tool, vibrations, tool wear, inhomogeneity of processed material
3rd	Grooves	Tool edge form, process kinematics, chip morphology
4th	Cracks	Tool-nose wear, built-up-edge formation, mode of chip formation, galvanic procedures
5th	Crystalline structure	Crystallization mode, irregularities due to chemical reactions, corrosive damage
6th	Crystalline formation	Physical and chemical alterations in the material's fine structure, deformations of lattice

Macroscopic geometric imperfections like form errors and waviness can be eliminated by addressing their underlying causes, but microscopic geometric variations in the form of roughness will always be present to some degree. While it may seem that minimizing the deviation from the nominal surface will lead to better performance and function, it's important to recognize that reducing roughness requires greater investment in time and cost during the machining process. As a result, the ideal roughness level should be carefully selected based on the specific application for each part [28].

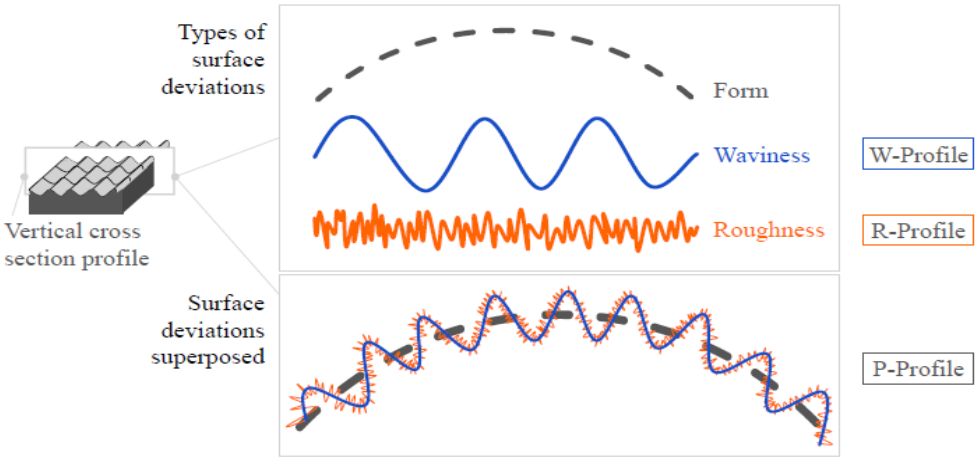


Figure 19: Types of surface geometric deviations (KLOCKE, 2011; REGO, 2020).

2.6 Surface parameters

Surface parameters are used to describe the topography of a surface, and they need to represent the entire surface statistically. In many industrial applications, surface values are evaluated and used for quality control when inspecting manufactured parts. This study utilized five of the approximately 60 available surface parameters, including Ra: Arithmetic mean deviation, RSm: Mean width of profile elements, Rsk: Skewness, Rku: Kurtosis, and Rt: Total height. These parameters can be classified into three groups based on their function [29]:

- Amplitude parameters: show vertical aspects of the profile.
- Spacing parameters: show horizontal aspects of the topography.
- Hybrid parameters: show vertical and horizontal aspects of the topography.

The Ra parameter is the most used to evaluate the roughness of machined surfaces and is calculated as the average deviation from the mean line (ISO, 1997). While it is widely used due to its ability to measure and represent the general profile, it does have limitations. For example, it does not differentiate between peaks and valleys and is not sensitive to slight deviations, which can limit understanding of the profile. Therefore, other parameters should be used in conjunction with Ra to obtain a more complete analysis of the surface [29]. Despite its limitations, Ra is helpful as it provides a standard index for comparing surfaces quantitatively. Table 3 outlines the surface roughness parameters utilized in this study, their symbols, and their mathematical representation according to standards.

Table 3: Definition of the surface roughness parameters used in this work according to ISO 4287 (1997)[5].

Roughness parameter	*Symbol*	*Mathematical representation*	*Eq.*
Arithmetic mean deviation	*Ra*	$Ra = \frac{1}{lr}\int_0^{lr} \lvert Z(x)\rvert \, dx$ Where $Z(x)$ are the absolute ordinate values within a sampling length (lr).	Eq. 7
Maximum peak height	*Rp*	$Rp = \max Zp_j \; \forall \; Zp_j \in [1, m]$ Where Zp_j is the height of the j^{th} peak within the sampling length, and m is the number of valleys in the sampling length.	Eq. 8
Maximum valley depth	*Rv*	$Rv = \max Zv_j \; \forall \; Zv_j \in [1, m]$ Where Zv_j is the depth of the j^{th} valley within the sampling length, and m is the number of valleys in the sampling length.	Eq. 9
Skewness	*Rsk*	$Rsk = \frac{1}{R_q^3}\left[\frac{1}{lr}\int_0^{lr} Z^3(x)\, dx\right]$	Eq. 10
Kurtosis	*Rku*	$Rku = \frac{1}{R_q^4}\left[\frac{1}{lr}\int_0^{lr} Z^4(x)\, dx\right]$	Eq. 11
Mean width of profile elements	*RSm*	$RSm = \frac{1}{m}\sum_{j=1}^{m} Xs_j$ Where Xs_j is the length of the j^{th} profile element within the sampling length (lr).	Eq. 12
Total height	*Rt*	$Rt = Rp + Rv$	Eq. 13

(*) A new series of standards, the ISO 21920, was released at the end of 2021 combining and replacing many of the previous surface roughness standards, such as the ISO 4287 that describes the main profile parameters. However, the changes do not affect this dissertation.

The parameter Rsk (skewness) is related to the distribution of peaks and valleys on a surface, with negative values indicating a predominance of valleys and positive values indicating a predominance of peaks. Kurtosis, on the other hand, is related to surface asperity and indicates the width of peaks and valleys. Values greater than 3 indicate a surface with narrower peaks and valleys, while values smaller than 3 indicate a smoother surface [30].

Table 4 displays the physical and functional properties of these surface roughness parameters and emphasizes their importance in various applications, as discussed earlier in

the introduction.

Table 4: Functional properties of the studied surface roughness parameters (DAVIM, 2010).

Functional properties	*Ra*	*Rsk*	*Rku*	*RSm*	*Rt*
Contact/Contact stiffness	*	*	*	**	**
Fatigue strength	*		*		**
Thermal conductivity	*			**	
Electrical conductivity	*			*	
Reflexivity					**
Friction and wear	*	**	**	*	**
Lubrication	*	**	*		**
Mechanical sealing	*	**			**
Fatigue corrosion	*	*		*	
Assembly tolerances	*				**

Note: two asterisks indicate a pronounced influence.

2.7 Surface roughness modeling in turning processes

It is crucial in this study to comprehend how surface roughness is predicted for machined parts. Therefore, it is necessary to introduce the classical theoretical equations that have been developed in the field to enable comparison with the models developed in this work. Additionally, understanding the general effects of process parameters on surface finish would be beneficial in comparing with the machining results.

2.7.1 Surface roughness modeling methods

In the pursuit of improving manufacturing performance while maintaining quality requirements, practitioners and engineers attempt to predict process behavior based on their experience and general conventions that may not be universally applicable. Researchers have tried to simulate complex cutting phenomena and establish cause-and-effect models between independent process variables (such as feed rate) and desired surface roughness values [31]. Regarding surface quality requirements, Benardos and Vosniakos (2003) categorized the effort to predict surface roughness into four research approaches. These approaches include:

a) Machining theory and simulation: which uses numerical computation and machining theory to simulate the cutting process based on tool properties and kinematics. This approach yields good results, but it may overlook important factors that can affect surface quality, such as tool wear and temperature.
b) Experimental/empirical investigations: which involves extensive testing to identify correlations between changing parameters/conditions and the surface response. This approach is conventional, but it can be difficult to replicate in other process conditions due to the high complexity of the surface generation phenomenon.
c) Design of experiments: which is similar to experimental investigations but is much more systematic and uses experimental planning to optimize available resources. Response surface methodology (RSM) and Taguchi techniques for "design of experiments" (DoE) are often used in this approach.
d) Artificial intelligence: which uses machine learning models to predict surface roughness in machining. This approach yields excellent results and has the potential for use in online monitoring and decision systems.

Additionally, He, Zong, and Zhang (2018) classified surface roughness prediction approaches into two categories:

- Theoretical modeling methods: these models are based on physical observations of turning processes, such as the periodic duplication of cutting tool edge/waviness, material spring back, plastic side flow, and other surface formation mechanisms.
- Empirical parametric modeling methods: these models are based on data mining regarding process parameters, such as the feed rate and surface roughness responses. These methods include the Response Surface Methodology (RSM), Machine Learning Algorithms, and Support Vector Machines (SVM), and can include other models.

Both categorizations proposed by Benardos and Vosniakos (2003) and He, Zong, and Zhang (2018) have similar concepts, but the latter is more concise and relevant to the present work, as depicted in Table 5. Therefore, the models proposed in this work are considered to be empirical parametric models according to the separation proposed by He, Zong, and Zhang (2018).

Table 5: Complementarity of propositions for the surface roughness prediction effort's separation.

He, Zong, and Zhang (2018) proposition	Benardos and Vosniakos (2003) proposition
Theoretical modeling methods	a) Machining theory and simulation
Empirical parametric modeling methods	b) Experimental/empirical investigations c) Design of experiments d) Artificial intelligence

2.7.2 Surface roughness influencing factors

Generating an actual surface in machining is a complicated process that involves various factors such as the machining method and parameters [28]. Typically, the actual surface roughness values are higher than those calculated using theoretical equations due to several reasons including chip-formation characteristics, tool wear, chatter on the machine-tool system, improper run-out on the part, and varying feed motion. Empirically perceived factors that influence the response of surface roughness can be collected [32]:

- Factors relative to the machine tool: depth of cut (a_p), feed rate (f), cutting speed (v_c), vibration amplitude acceleration (A/a), and vibration frequency (ω);
- Factors relative to the cutting tool: tool material, angles/inclinations, corner radius (r_ε), and tool edge waviness (w_1);
- Factors related to the workpiece material: material hardness, Young's modulus, grain size, orientation, inclusions, etc.;
- Factors related to the environment: coolant (type, pressure, etc.), temperature, and dampness.

In turning processes, the feed rate is the primary factor that influences the average surface roughness Ra, and there is a positive correlation between the feed rate and Ra (i.e., higher feed rate results in higher Ra). Figure 20 illustrates the general effect of feed rate and cutting speed on Ra. Lower cutting speeds can cause the formation of a built-up edge (BUE), which is the adhesion of workpiece material to the tool's tip, making it harder than the tool. This formation can significantly harm the surface roughness and the tool's condition. The depth of cut, on the other hand, does not seem to consistently affect the surface roughness Ra [28].

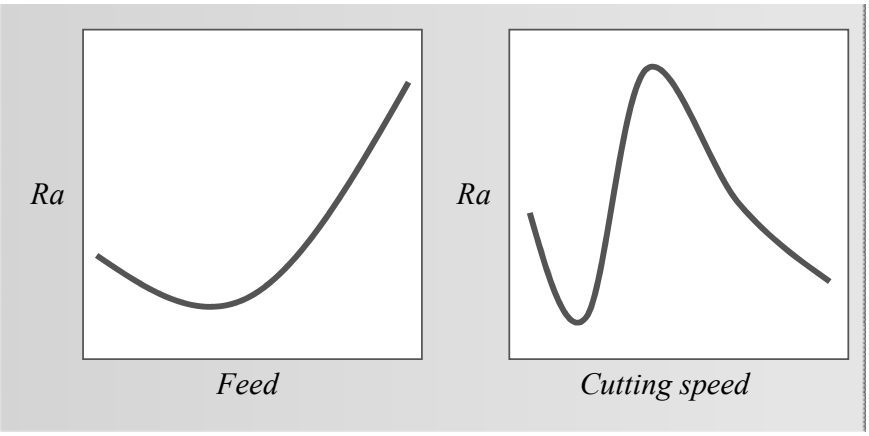


Figure 20: The general effect of the feed rate (f) and the cutting speed (v_c) in the average surface roughness (Ra) (DAVIM, 2010)

2.7.3 Surface roughness theoretical models

In the last few decades, several theoretical models have been proposed to describe surface roughness, particularly for the average height of the profile Ra and the total height of the profile Rt. Table 6 lists some of the most well-known models [27],[28],[32], which consider factors such as feed rate (f), corner radius (r_ε), and minor and principal cutting-edge angles (κ_r and $\kappa_{r'}$). Figure 21 displays the ideal surfaces produced by a perfectly sharp tool (left) and a rounded tool (right), highlighting the essential role of feed and tool geometry (especially the corner radius) in the machined surface.

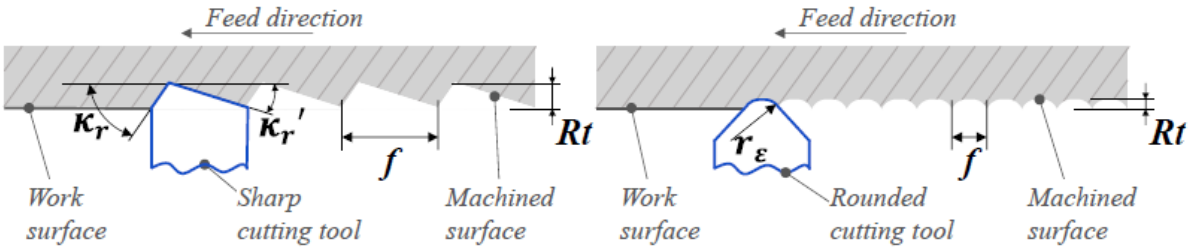


Figure 21: Surface generation influencing factors for a theoretical perfectly sharp (left) and rounded (right) cutting tool (DAVIM, 2010).

Other recent models, which are less well-known and more computationally complex, consider cutting dynamics factors such as vibration amplitude acceleration (A/a), vibration frequency (ω), minimum chip thickness (h_{min}), and an artificial variable called SE that indicates cutting stability [32]. However, the focus of this work is on proposing empirical parametric models, as described in the previous section, and comparing these new models

to the classical, simplistic theoretical models.

Table 6: Some theoretical equations of surface roughness Ra and Rt (BOOTHROYD; KNIGHT, 2006; DAVIM, 2010; HE; ZONG; ZHANG, 2018).

Theoretical equation	*Description*	*Equation*
$Ra = \frac{0.0321 * f^2}{r\varepsilon}$	Ra for a rounded tool and given feed rate (f) and corner radius (r_ε)	Eq. 14
$Rt = \frac{1}{8} * \frac{f^2}{r\varepsilon}$	Rt for a rounded tool	Eq. 15
$Ra = \frac{f * \tan(kr')}{4 * (1 + \tan(kr')}$	Ra for a perfectly sharp tool. κ_r' is the minor tool cutting edge angle	Eq. 16
$Ra = \frac{f}{\cot(kr) + \cot(kr')}$	Rt for a perfectly sharp tool. κ_r is the tool cutting edge angle	Eq. 17

Chapter 3 – Literature Review

Synopsis

This chapter deals with the literature reviews of some of the well-known papers in the field pertaining to topics such as Machining, Turning, Machine Learning and Surface Roughness prediction. These papers in addition with the work proposed by Canal (2022) in his Master Degree thesis have been utilized as framework in developing this thesis.

3.1 Literature Review

To properly research surface roughness prediction in turning using machine learning algorithms, it's important to examine what scientists from around the world have already accomplished. One way to contribute to the international research community is by utilizing research databases such as Scopus and Web of Science. By conducting a bibliometric search, we can gain insight into the topic's characteristics, key players, leading publishers, current state, and potential avenues for exploration in surface roughness prediction studies. Our literature review framework is based on Canal's (2022) Master Degree Thesis.

Machine learning is rapidly expanding due to the increasing volume of available data and the constant improvement of techniques to turn data into knowledge [33]. The surge of machine learning research is depicted in Figure 22 as an exponential curve, highlighting its significance to the global community and the growing awareness of its potential. The curve also implies that the field has yet to reach its full potential, with plenty of room for further development and discovery.

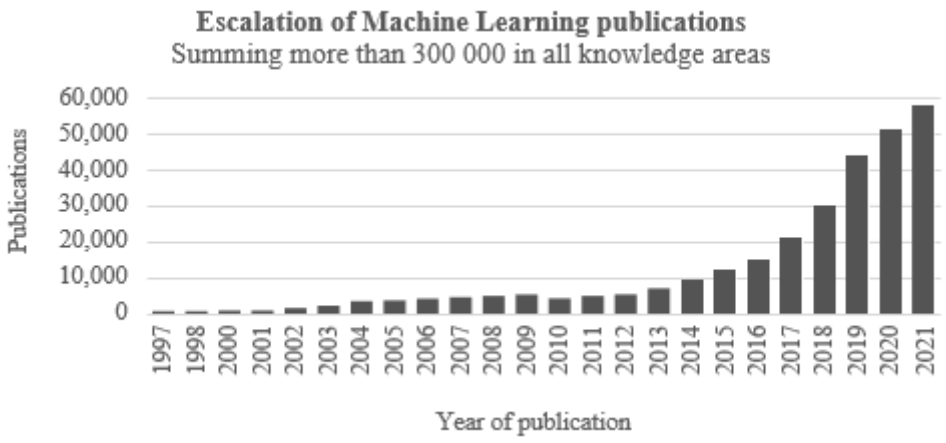


Figure 22: Publications regarding Machine Learning thematic through the years (Scopus)

The bibliometric results, shown in Figure 23, indicate that there is a significant amount of research being conducted on Machine Learning in the fields of engineering, materials science, and computer science, following the same exponential curve seen in Figure 22. Within the intersection of these three fields, which is the focus of this study, there are 83 relevant documents. Scopus provides information on the countries and affiliations that are producing these publications. Currently, India and China lead the research on this topic, with the National Institute of Technology (NIT), a network of institutes across India, having the highest number of published papers.

Although India and China account for more than 50% of the citations, other countries are also increasing their citations year by year, suggesting that this topic is of interest to researchers from around the world.

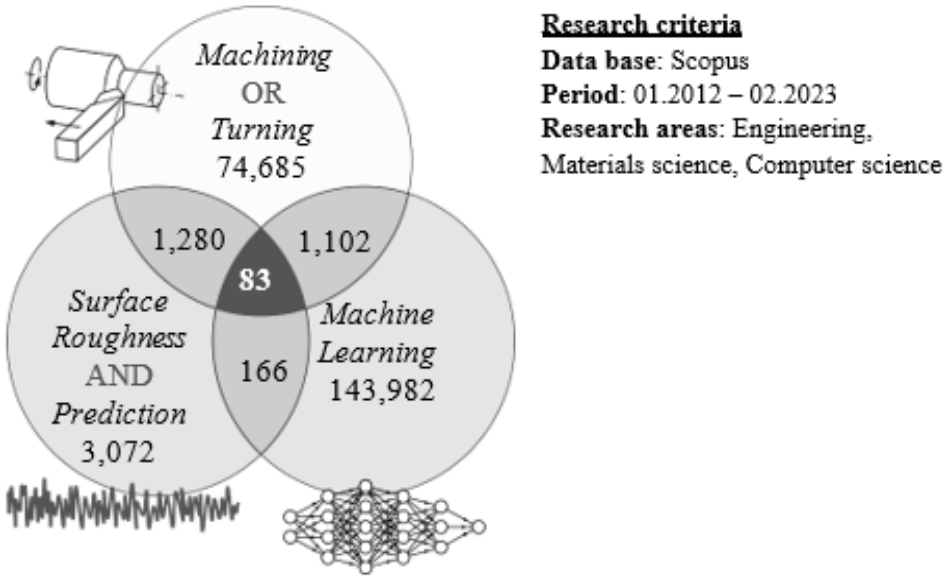


Figure 23: The bibliometric search of "surface roughness prediction in machining using Machine Learning."

Reading through 83 papers is a daunting task, and conducting a good literature review requires selecting the works that are most relevant to the thesis. Therefore, it's important to choose effective criteria to narrow down the number of relevant publications. Excluding papers published between 2012 and 2016 based on their publication date alone may not be the best approach as the field of machine learning is constantly evolving, with new research being published each year, as shown by the exponential curve in Figure 22.

A combination of criteria was used to refine the list of relevant publications. The first filter, shown in Figure 24, retained 28 papers, while the remaining 55 were assessed based on

their abstracts and subjectively determined relevance to the thesis. The second filter used several criteria, including the machine learning approach, the machining process, the workpiece material, whether it measured machining forces and tool wear, the number of citations, and the novelty of the papers. This resulted in 33 papers being selected for a full read, and ultimately, 19 papers were included in this literature review.

During the reading process, additional papers cited outside of the observation period were found to be relevant and were also included. From the 19 papers read, conclusions were drawn, which are discussed in the subsequent paragraphs.

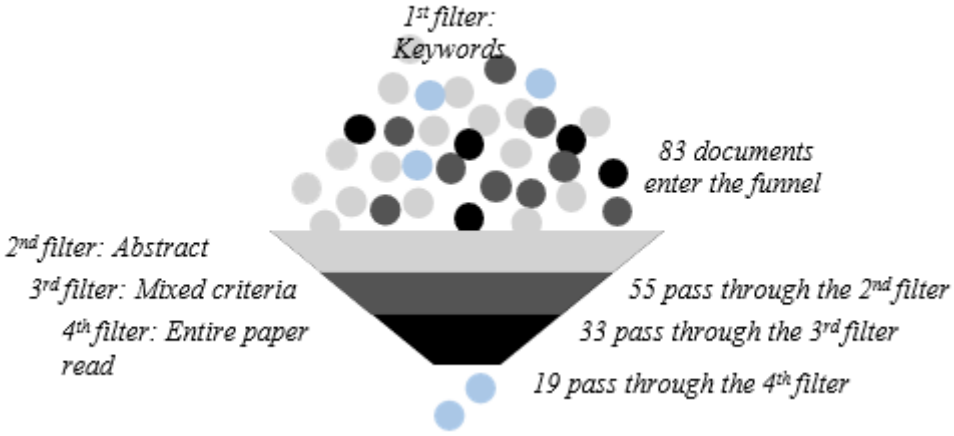


Figure 24: Choosing fitting papers to read for literature review

Among all the works analyzed, the Machine Learning algorithms showed relatively low levels of error when predicting on data that had never been seen before. Several of the research papers examined reported correlation coefficients (R) exceeding 0.90 [34] [35] [36]. Other works reached mean absolute percentage errors (MAPE) between 5% and 15% [37] [38] [39] [40].

Some works achieved surprisingly low errors:

- a root mean squared error RMSE = 0.0005 and an R-squared of 0.99 using only 23 samples to train the model [40];
- an MSE = 0.0001 and R-squared of 0.99 using a training dataset of 600 samples [41];
- a root mean squared percentage error RMSPE = 0.07% with a training dataset of 130 samples [42];
- a RMSE = 0.02 using only 23 training samples [43].

Several authors have achieved low error results by using a training set smaller than 55 samples, as cited in references [42] [34] [37] [39] [40] [35] [36] [44] [45]. However, in a study conducted by Pontes et al. (2012), the performance of various ML models was compared using different numbers of labeled training examples, ranging from 20 to 500 samples. The results suggested that generally, the ML model's performance in predicting Ra is improved with an increased number of training samples.

Most of the papers use the same three inputs: cutting speed (v_c), depth of cut (a_p), and feed rate (f), to predict some of the surface roughness parameters [47] [43] [34] [48] [38] [39] [45] [46].

However, some papers included additional factors as inputs, such as:

- Cooling methods, whether wet or dry cut, coolant placement (through the tool or outside), or different uses of minimum quantity lubrication (MQL) [35] [36] [49];
- Different cutting tools and workpieces with varying heat treatments [42];
- Cutting time, depth of cut, and cutting speed [37], [41];
- Cutting tool edge radius (r_ε) [40];
- Using depth of cut, feed rate, and cutting speed to predict the three components of machining force (F_x, F_y, F_z) and three components of tool vibration (V_x, V_y, V_z), then utilizing all nine factors as inputs to calculate the surface roughness Ra [44].

In terms of the results, most of the research studies examined developed Machine Learning algorithms, with a particular focus on Artificial Neural Networks (ANN) models, to forecast the surface roughness Ra [47] [41] [43] [44] [48] [42] [39] [40] [36] [49] [45]. Ra is the most used parameter for surface roughness in practical applications, making it the most sought-after prediction model.

However, some papers incorporated other outputs into their prediction models, such as:

- Predicting the resultant machining force and cutting tool flank wear width (VB) in addition to surface roughness Ra [48];
- Predicting the tool condition in terms of flank wear width (VB) along with surface roughness Ra [45];
- Predicting the machining time and cost [47].

Most of the research papers studied used RMSE and MSE as their cost functions [47] [41] [43] [44] [38] [42] [40] [35] [36] [49].

Despite its popularity and open-source nature, Python was not used in many of the reviewed papers.

3.2 Literature remarks

After examining recent developments, certain observations and patterns were made:

- The most frequently used process parameters were depth of cut, feed rate, and cutting speed;
- The parameter most commonly modeled was the average height of the profile (Ra);
- A machine learning model's ability to predict Ra improves with a larger number of training samples;
- The most commonly used cost functions were RMSE and MSE.

In addition to technical issues, there was a lack of transparency observed, which could hinder reproducibility. For example, most of the reviewed papers did not provide a description of the different machine learning models with their codes. A comprehensive review paper on the history and trends of data science suggests that reproducibility of soft computing research will be more widely practiced in the future, and that code and data must be universally citable and systematically retrievable [50].

The aim of this dissertation is to make the process of constructing Surface Roughness prediction models using machine learning algorithms as transparent as possible, by providing full access to the data, code, and output results. Based on the literature reviewed, two opportunities for contribution to this research area have been identified.

The first opportunity involves the tendency to produce more accurate models (with mean absolute percentage errors ranging from 1% to 10%) using fewer training samples. However, achieving minimal errors (MAPE < 2%) with very few training samples can lead to overfitting [51]. Therefore, a potential solution to this problem is to increase the amount of data without increasing the number of experiments.

The second opportunity comes from the fact that the ML models developed in previous studies did not adequately consider cutting tool wear. Given that cutting tools inevitably wear down when cutting metals, it is reasonable to assume that this phenomenon, which can alter cutting kinematics, should be considered [32]. Therefore, investigating how tool wear affects ML modelling of surface roughness configures a second opportunity for contribution.

3.3 Thesis objectives

The main goal of this dissertation is to contribute to the understanding of using machine learning to model metal cutting results. More precisely, the goal is to use Machine Learning models to predict surface finish results in the turning process. Toward this broad objective, the following specific objective stood out:

i. Statistically assess the effect of the factors (depth of cut, feed rate and tool condition, in terms of flank wear width) on the responses surface finish roughness parameters (Ra, Rsk, Rku, RSm, and Rt), and the machining forces;

ii. Develop ML models to predict the surface roughness parameter [Ra] and evaluate it with a chosen error metric;

iii. Propose a strategy to augment available data without increasing experiments;

iv. Investigate the influence of the dataset size on the ML models' prediction performance;

v. Investigate the influence of the tool condition, flank wear width, on the ML models' prediction performance;

vi. Compare the prediction performance of the ML models with one theoretical equation.

Chapter 4 - Material and Methods

Synopsis

The work can be divided into two parts: machining experiments and the development of Machine Learning algorithms. Figure 25 shows the two datasets obtained from two machining procedures. Experiment 1 involved varying three factors at three levels, resulting in a dataset of 324 labeled examples (factors, responses). Experiment 2 varied three factors at different levels, producing a dataset of 288 labeled examples. The data from Experiment 1 and Experiment 2 were then used in the development of Machine Learning algorithms in Chapter 6. The models differed in their inputs, training dataset size, and the dataset used for training. Afterward, the models were analyzed to determine the impact of the training set size and cutting tool wear on Surface Roughness parameters.

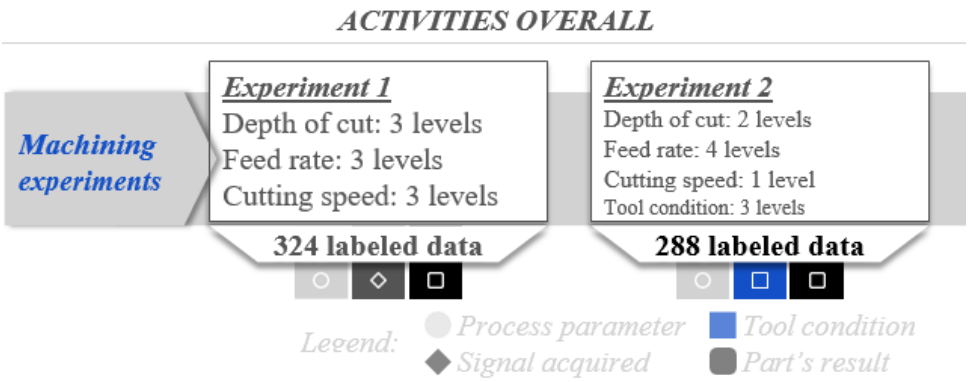


Figure 25: The overall activities. Machining experiments generated data; the results were statistically analyzed; machine learning models were developed and investigated.

4.1 Machining experimental procedures

Although the content of this work is primarily focused on computational science, its fundamental objective is to investigate the turning process in manufacturing. Thus, to support the results obtained from the computer procedures, detailed explanations of the machining experiments are provided. Two experiments were carried out as shown in Figure 25. The machining was carried out on a CNC turning center, using cutting fluid. All experiments were conducted in the Competence Center in Manufacturing (CCM) infrastructure, which is a laboratory at the Aeronautics Institute of Technology (ITA).

4.1.1 Experiment 1

The initial test investigated how three different process parameters affected the surface finish outcome of the steel workpiece. The experiment was set up based on the design in Table 7, with the cutting speed (v_c), feed rate (f), and depth of cut (a_p) being adjusted to three different levels in a full factorial design 3^3, resulting in a total of 27 unique conditions being evaluated. In addition, the experiment was replicated, giving a total of 54 trials. When multiple factors impact a response, factorial experiments are typically the most effective approach. This approach ensures that all possible combinations of factor levels are examined. The range for each parameter level was selected based on the tool manufacturer's recommendations to simulate real production conditions and to prevent any possible disturbances that could arise from out-of-range parameters.

Table 7: DoE of Experiment 1.

Independent variables (factors)

Factors	*Symbol*	*Units*	*Level*			*No. of levels*
Cutting speed	(v_c)	m/min	310	350	390	3
Feed rate	(f)	mm/rev	0.07	0.1	0.13	3
Depth of cut	(a_p)	mm	0.25	0.5	0.8	3

The steel workpiece was prepped to enable each unique condition (combination of process parameters) to be tested independently as a single machining operation, without being impacted by any other runs. To accomplish this, grooves were made in the workpiece to divide each machining area associated with each condition. This method also ensured that measurements were taken in the correct location, as the test parts were physically separated. The experimental design and acquired signals are illustrated in Figure 26.

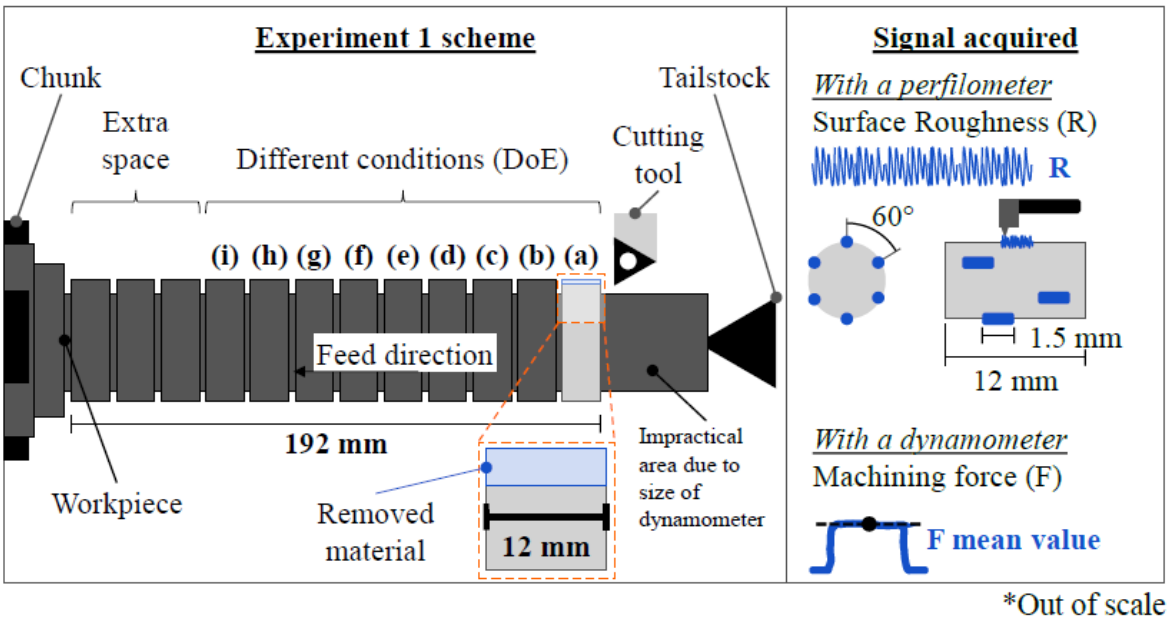


Figure 26: Experiment 1 Scheme (left) and the signal acquired (right).

Each testing length, area, or machining space (example (a) in Figure 26) had a length of 12 mm. To increase the amount of available data, surface roughness was measured in six different locations along the surface of the workpiece within the testing area. The measurement locations were 60º apart from each other, with the center of the workpiece's face serving as the reference point (Figure 26, right side). Measurements were taken perpendicular to the surface texture, as recommended in industry standards and practical manuals (ISO, 1996), and were randomly distributed throughout the 12 mm testing area to minimize systematic errors and ensure representative results.

The grooves machined into the workpiece were the exact width of the grooving tool, measuring 4 mm. Out of the total length of the workpiece, 12 testing areas were created, with nine utilized in the experiment, and three held in reserve for potential reruns. The space between the tailstock (or the right face of the workpiece) and the first testing area was too small to accommodate the dynamometer plate and, therefore, had to be machined and could not be utilized.

One factor that was varied in the experiment was the depth of cut. Randomizing the depth of cut could pose a problem, as machining a testing area with a smaller diameter than the previous one could result in a collision during the subsequent runs (Figure 27, Case 2). While there was enough space between each sample area to approach the cutting tool in each run, it was preferable to avoid the risk of collision. Consequently, the depth of cut was kept constant for each of the nine conditions throughout the workpiece length (Figure 27, Case 1). This was accomplished by grouping the experiment design into "9 + 9 + 9"

segments.

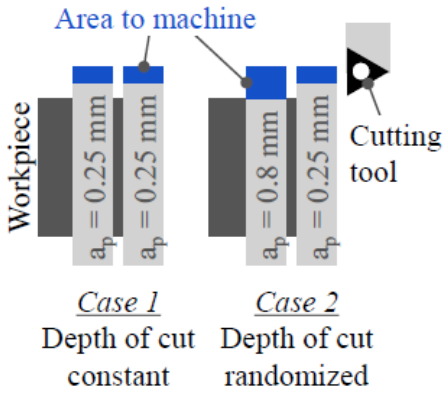


Figure 27: Case 1 (left) depicts the adopted procedure, and Case 2 (right) depicts a collision risk situation.

The first experiment was designed with a randomized approach to reduce the influence of extraneous factors and maintain the observations as independent random variables. Figure 28 displays the actual sequence of the tested conditions in Experiment 1. Randomization was applied both within and between groups. The cutting speed, with three levels, was used to form sub-groups (orange box) within the depth of cut groups (blue box), while the feed rate was randomized within the sub-groups. Thus, the effect of the feed rate was studied within each sub-group, the effect of the cutting speed was studied between sub-groups, and the effect of the depth of cut was studied between groups.

The nine testing areas were individually machined with a stop between runs. Starting from group 1, the following steps were followed:

1. initiate cutting with the current condition's process parameters (in the first case, condition 4, position a),
2. measure forces during the cut,
3. after completing the cut, measure surface roughness six times, varying the measurement position,
4. continue cutting the subsequent testing area until all conditions have been tested,
5. Move on to the next group and repeat steps 1-5 until all groups have been tested.

	Group 1								
	Sub-group 1			Sub-group 3			Sub-group 2		
f	0.13	0.07	0.10	0.07	0.13	0.10	0.10	0.13	0.07
v_c	310	310	310	390	390	390	350	350	350
a_p	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
Cond.	3	1	2	7	9	8	5	6	4

Workpiece

Feed direction

	Group 1 (2nd replica)								
	Sub-group 1			Sub-group 3			Sub-group 2		
f	0.07	0.13	0.10	0.07	0.10	0.13	0.07	0.13	0.10
v_c	310	310	310	390	390	390	350	350	350
a_p	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
Cond.	1	3	2	7	8	9	4	6	5

	Group 2								
	Sub-group 5			Sub-group 6			Sub-group 4		
f	0.07	0.13	0.10	0.13	0.07	0.10	0.10	0.13	0.07
v_c	350	350	350	390	390	390	310	310	310
a_p	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Cond.	13	15	14	18	16	17	11	12	10

	Group 2 (2nd replica)								
	Sub-group 6			Sub-group 5			Sub-group 4		
f	0.13	0.10	0.07	0.07	0.10	0.13	0.07	0.13	0.10
v_c	390	390	390	350	350	350	310	310	310
a_p	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Cond.	18	17	16	13	14	15	10	12	11

	Group 3								
	Sub-group 7			Sub-group 9			Sub-group 8		
f	0.10	0.07	0.13	0.13	0.10	0.07	0.10	0.13	0.07
v_c	310	310	310	390	390	390	350	350	350
a_p	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Cond.	20	19	21	27	26	25	23	24	22

	Group 3 (2nd replica)								
	Sub-group 9			Sub-group 8			Sub-group 7		
f	0.10	0.13	0.07	0.10	0.13	0.07	0.13	0.07	0.10
v_c	390	390	390	350	350	350	310	310	310
a_p	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Cond.	26	27	25	23	24	22	21	19	20

Figure 28: Randomization of the experiment and groups' division.

4.1.2 Experiment 2

Experiment 2 aimed to examine how the condition of the cutting tool, as measured by the flank wear height VB, affected the surface finish results. The setup for this experiment was the same as that of Experiment 1, which can be seen in Figure 26 (left side), with the testing areas divided by grooves. However, Experiment 2 used fewer testing areas and stopped at position (h) on the workpiece. Additionally, an extra measurement system was used in Experiment 2 to evaluate the tool's wear. The results of this experiment are presented in Figure 29.

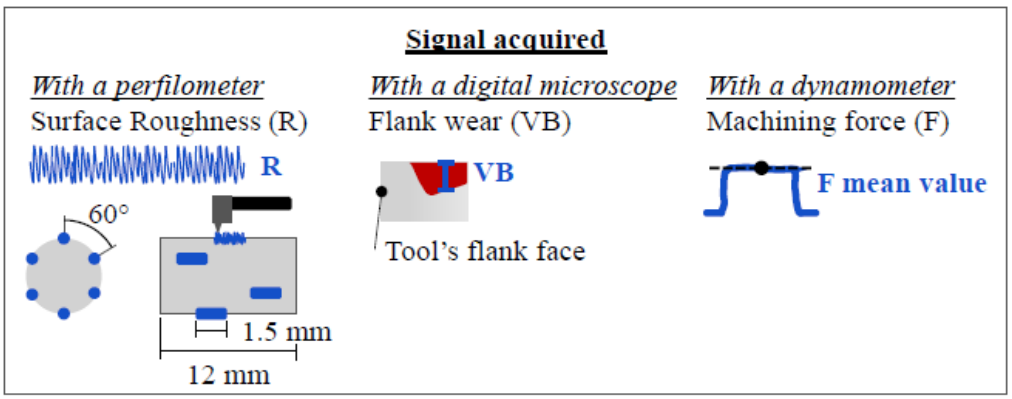


Figure 29: Information acquired at Experiment 2.

Before conducting Experiment 2, a pre-analysis of the results from Experiment 1 was carried out. As a result, the cutting speed was kept constant since it was found to have a minimal impact on the surface finish response. The depth of cut was only varied in two levels, as it had less influence on the surface finish compared to the feed rate. However, the depth of cut was still varied more than the cutting speed, as it was found to have a greater impact on the machining forces. The feed rate was varied four times as it was discovered to be the most influential factor on surface finish. Table 8 depicts the design of Experiment 2.

Table 8: DoE of Experiment 2.

Independent variables (factors)

Factors	*Symbol*	*Units*	*Level*				*No. of levels*
Cutting factors							
Cutting speed	(v_c)	m/min	350				1
Feed rate	(f)	mm/rev	0.07	0.09	0.11	0.13	4
Depth of cut	(a_p)	mm	0.25	0.5			2
Cutting tool factors							
Tool condition	(VB)	mm	0	0.1	0.3		

The categorical levels of tool condition (VB) used in Experiment 2 were approximated for the sake of simplification. The actual flank wear size of each cutting tool was slightly around these values. New tools, such as tool 44 and 53, had a VB of 0.000 mm, while mid-life tools like tool 13 and 33 had VB of 0.102 mm and 0.096 mm, respectively. End-life tools, such as tool 81 and 23, had VB of 0.300 mm and 0.305 mm, respectively.

Figure 30 shows the sequence of tool conditions tested during Experiment 2, with runs randomized in the same manner as in Experiment 1. The depth of cut was kept constant at each level to avoid collision, as explained in Figure 27, and replica conditions were grouped and randomized within each level. The tool condition was tested through six levels, with Levels 1 and 2 using new cutting tools. Levels 3 and 4 used mid-life tools with flank wear between 0.1 mm and 0.15 mm. Levels 5 and 6 used end-life cutting tools with a VB of more than or equal to 0.3 mm but not exceeding 0.35 mm.

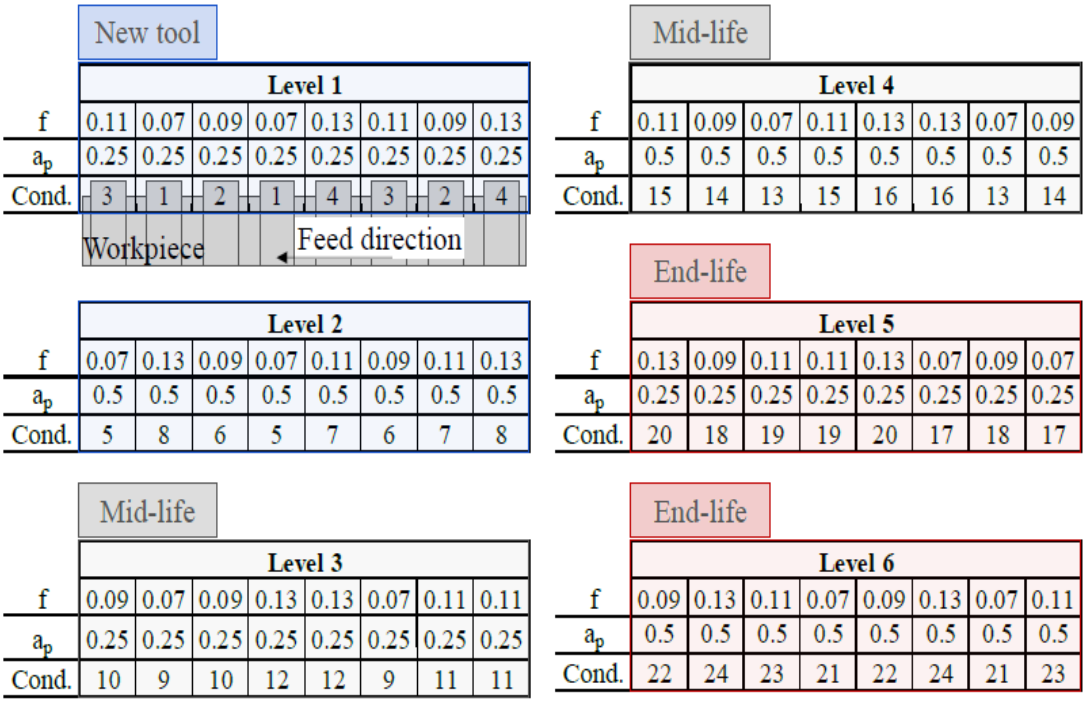


Figure 30: The sequence of conditions tested in Experiment 2. Three different tool conditions were tested through the levels.

Figure 31 illustrates the experimental setup utilized in Experiment 2. The assessment of tool wear (represented in blue) was conducted with a digital microscope mounted on a magnetic base that was held in the same position after each run and pointed towards the cutting tool's flank. The surface roughness measurement was performed in the same way as in Experiment 1. Figure 31 also displays the setup for Experiment 1, except for the tool wear assessment system.

In Experiment 2, the data was collected using the same approach presented in Experiment 1, measuring the surface roughness at six different locations after each cut. For each surface roughness parameter, 288 labeled pairs of input and response were obtained, resulting in a total of 1440 pairs of labeled examples collected in Experiment 2 considering the five surface roughness parameters.

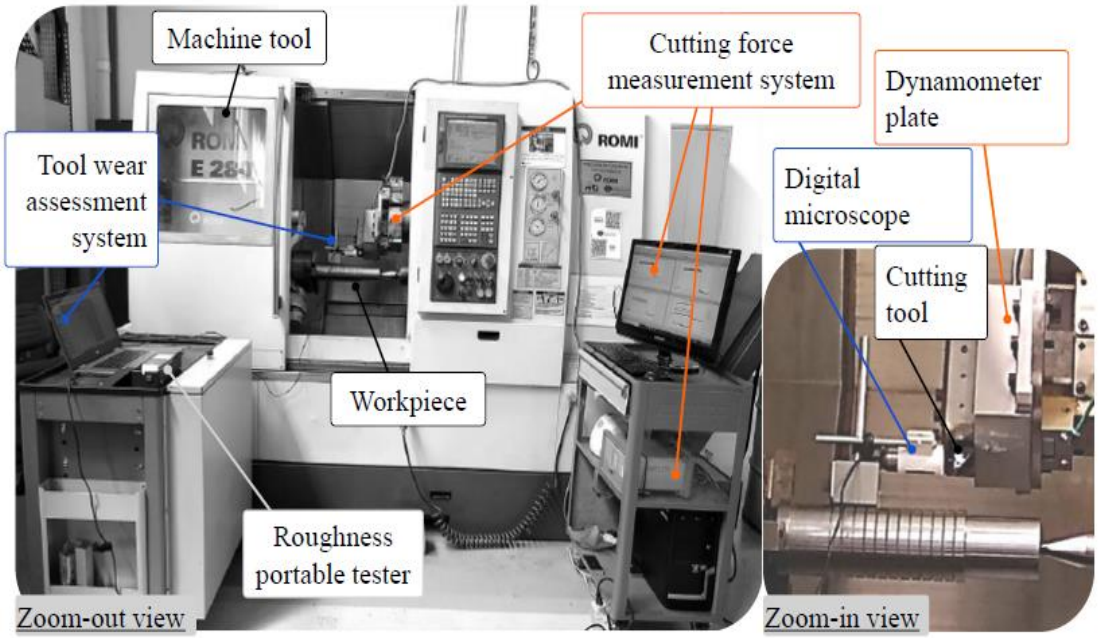


Figure 31: Experiment 2 setup. Zoom-out view (left) shows the whole setup, and the zoom-in (right) shows the digital microscope pointing at the tool to measure flank wear.

4.1.3 Workpiece material and cutting tool

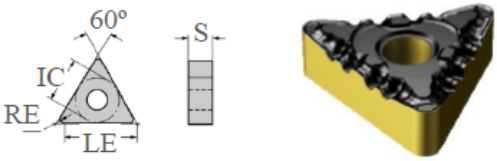
This section provides information about the material and cutting tool used in Experiment 1, and Experiment 2. The material used in all experiments was the AISI H13 steel, which was manufactured by Villares Metals. The chemical composition of the steel is provided in Table 9, and its average hardness is 200HV, measured transversally to the bar. This steel is commonly used in plastic injection molds due to its good mechanical properties, machinability, and wear resistance, among other characteristics [52].

Table 9: AISI H13 steel chemical composition (VILLARES METALS, 2006).

Element	*C*	*Si*	*Mn*	*Cr*	*Mo*	*V*
Percentage (%)	0.40	1.00	0.35	5.20	1.50	0.90

The cutting tool used in the experiments was a commercial model manufactured by Sandvik Coromant, with a hard metal substrate covered with TiCN, Al2O3, and TiN. This tool has six cutting edges and is recommended for finishing applications. Further specifications of the cutting tool can be found in Table 10.

Table 10: Cutting tool specifications.



Cutting tool commercial code:

ISO
TNMG 16 04 04-PF 4425
ANSI
TNMG 331-PF 4425

Geometric specifications

Item	*Symbol*	*Description*
Diameter of the inner circle	IC	9.525 mm
Edge radius	RE	0.397 mm
Effective cutting length	LE	16.098 mm
Width	S	4.762 mm

Functional specifications

Item	*Description*
Operation	Finishing
Size and shape	TN1604
Hand	N
Substrate	HC
Grade	4425
Coating	CVD TiCN+Al2O3+TiN
Clearance angle major	0

The tool shank (holder) used has code ISO: MTJNL 2020K 16M1 and is also manufactured by *Sandivik Coromant*. The grooving tool for preparing the workpiece was a *TaeguTec* TDJ 3 TT9080.

4.1.4 Machining forces processing

The force measurement system used in both Experiment 1 and Experiment 2 consisted of a Kistler Type 9265B dynamometer, a Type 5070 charge amplifier, a Type 2825A acquisition software, a computer, and other peripherals including a highly insulated cable, PCI interface, connection cable, and acquisition plate (A/D), as shown in Figure 32. During machining, the system collected data on the machining forces, which were later processed using the Kistler DynoWare software Type 2825A (as shown in Figure 32). The beginning and end of the cutting were removed to avoid erroneous force readings caused by the acceleration of the cutting tool and the tool-workpiece engagement. The software then calculated the mean value of the forces in the cutting zone, which was sufficient to describe the continuous turning process. The force values were reasonably equally distributed around the mean value line, so there was no need for smoothing or filtering, as these techniques did not significantly affect the observed mean value of the forces.

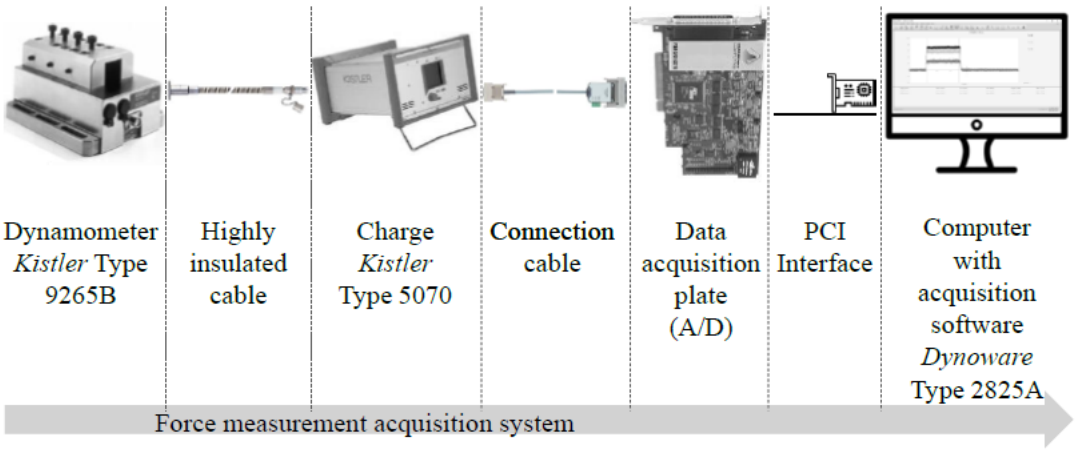


Figure 32: Force measurement acquisition system: connection configuration.

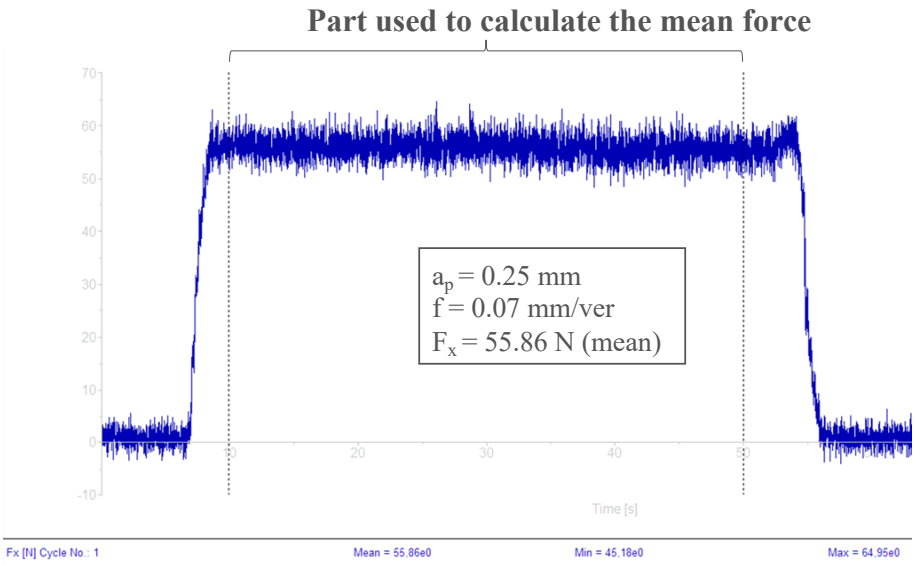


Figure 33: Machining forces visualization and means assessment. A discrete value was selected from the force time-series measurements: the mean.

The Kistler dynamometer system used in the experiments collected the three components of the resultant force, which are the cutting force (F_c) on the X-axis, the passive force (F_p) on the Y-axis, and the feed force (F_f) on the Z-axis, as shown in Figure 34. The mean values of these forces were collected and processed in the Kistler DynoWare software. The resultant force was then calculated using Equation 6.

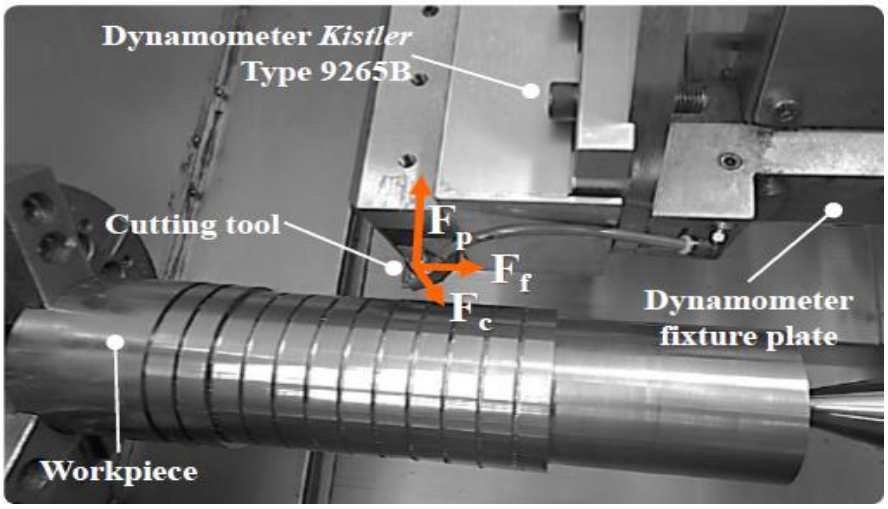


Figure 34: The machining force components measured in experiments 1 and 2.

4.1.5 Tool wear assessment

The wear of the tool flank was evaluated using a Dino-Lite model AM4113ZT digital microscope, as shown in Figure 35. The microscope was positioned on the workpiece using a magnetic base, allowing the tool wear to be measured without having to remove the tool from the machine.

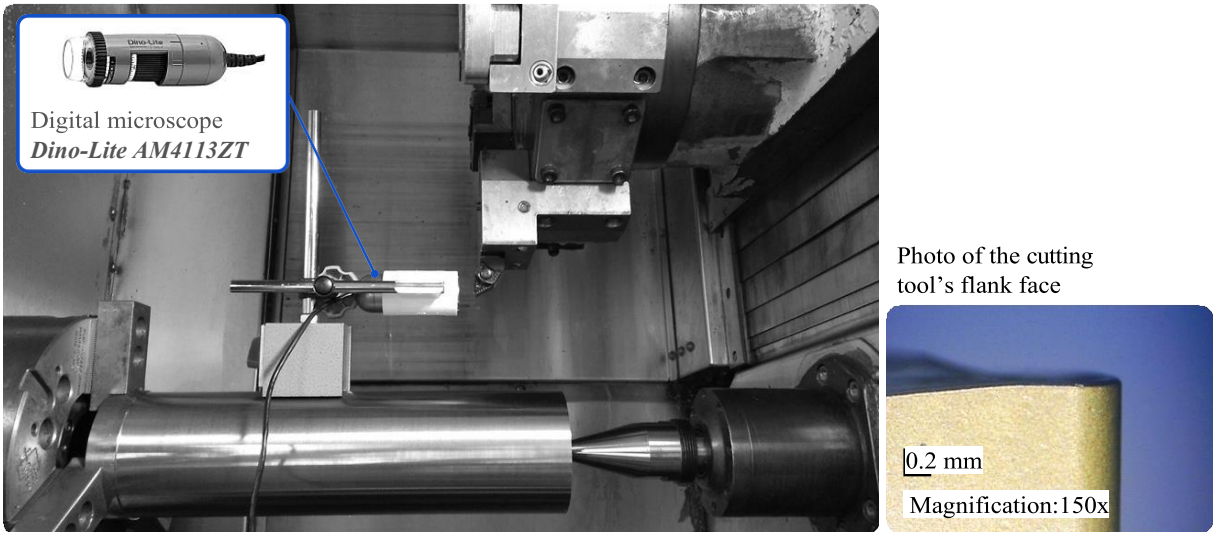


Figure 35: Tool wear assessment with a digital microscope.
Measurement setup (left) and result example (right).

4.1.6 Surface roughness processing

In order to measure the surface roughness, a Mitutoyo portable roughness tester model Surftest SJ-210 was used, as shown in Figure 36. The tester had a diamond tip with a radius of 2 µm and an angle of 60º. The surface roughness was measured in a direction

perpendicular to the theoretical surface, along the direction of the feed rate.

Figure 36: *Mitutoyo* portable roughness tester model Surftest SJ-210.
Measurement setup (left) and result example (right).

In order to read and interpret the surface roughness values, for the Machine Learning models, data must be organized in tables or data frames. However, the surface roughness measurements obtained in Experiment 1 and Experiment 2 were initially saved in text format (TXT) in a memory card with non-numeric characters attached to them. There were 824 documents, with each measurement corresponding to one document, and five surface roughness parameters were chosen for each measurement, resulting in 4120 values that needed to be collected from the text files and organized into data frames. To avoid errors and undesired outliers, an automated program was developed in Python to read and select the values from the TXT files and write them into Excel and comma-separated value format files (XLSX and CSV). The workflow from measuring to accessing the individual surface roughness values is shown in Figure 37.

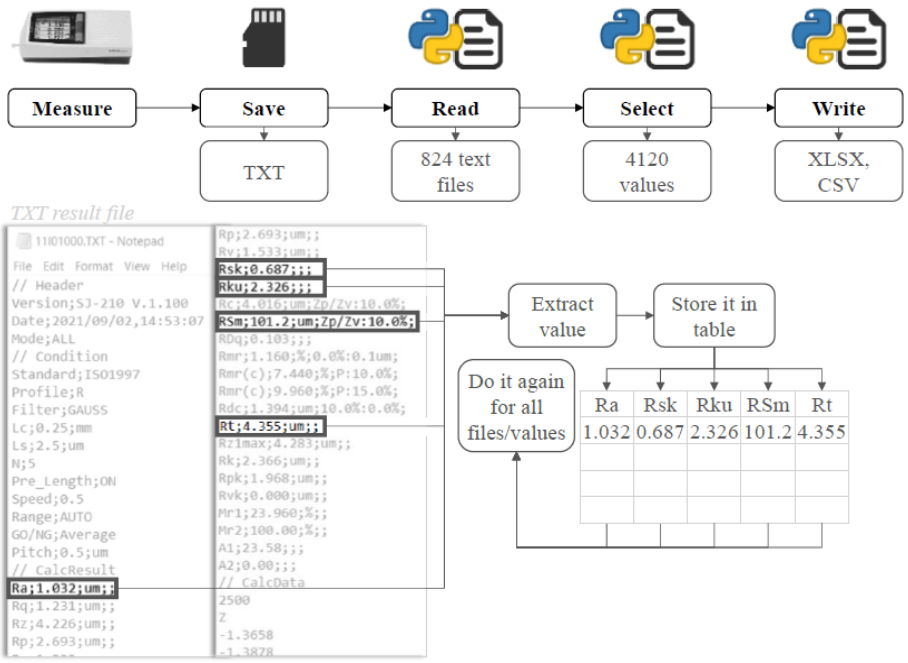


Figure 37: Surface roughness data processing-to-machine-learning workflow.

4.1.7 Strategy to increase available data

A strategy was used in both Experiment 1 and Experiment 2 to increase the amount of available data without conducting additional experiments. This approach has been mentioned earlier in the respective experiment sections, but it is emphasized here for clarification. The strategy involved measuring the surface roughness (the main dependent variable) six times for each combination tested, rather than conducting additional experiments.

Chapter 5 – ML algorithms and Evaluation metrics

Synopsis

For the scope of our study, we will be focusing on the regression analysis from supervised learning to train and test various ML models to predict the target variable "Ra". The algorithms we will be covering ahead:

- Linear Regression
- Decision Tree Regression
- Random forest Regression
- Bayesian Linear Regression
- KNN Regression
- Kernel Ridge Regression
- Neural Network Regression

To assess the effectiveness of the model, specific performance evaluation metrics were utilized. The ensuing list enumerates the metrics used:

- Mean Squared Error (MSE)
- Root Mean Square Error (RMSE)
- Mean Absolute Error (MAE)
- Coefficient of Determination (R^2)
- Explained Variance

5.1 Evaluation Metrics

5.1.1 Mean Squared Error (MSE)

In mathematics, the mean squared error (MSE) of an estimator is a metric that calculates the average of the squares of the errors, i.e., the average squared difference between the predicted and actual values. The MSE is a probability function that represents the estimated squared error loss value. Typically, it is non-negative, and values closer to zero indicate better performance [53]. The mean squared error function measures the mean of the squared (quadratic) prediction error or loss value, which is a probabilistic metric [54].

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

Equation 17: Mean Squared Error

5.1.2 Root Mean Square Error (RMSE)

The standard deviation of the residuals is equivalent to the root mean square error (RMSE). Residuals are an indicator of the distance between data points and the regression line, while RMSE indicates the distribution of these residuals. In other words, it illustrates the concentration of data along the best-fit line [55]. RMSE is calculated by taking the square root of the average of the squared differences between actual and predicted observations [56].

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Equation 18: Root Mean Squared Error

5.1.3 Mean Absolute Error (MAE)

In a series of predictions, the mean absolute error (MAE) calculates the average magnitude of the errors without considering their direction. It is the average of the absolute differences between the forecast and the actual observation over the test set, where each individual difference has equal weight [56].

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

Equation 19: Mean Absolute Error

5.1.4 Coefficient of Determination (R^2)

The coefficient of determination, denoted as R^2, provides an indication of how well a model fits the data and its ability to predict unseen samples by measuring the proportion of variance in the target variable that can be explained by the model. It is important to note that R^2 is dataset-specific and cannot be compared across different datasets. The maximum value of R^2 is 1.0, and a score close to 1.0 indicates a good fit, while a score close to 0.0 indicates a poor fit. A constant model that always predicts the mean value of the target

variable, regardless of the input features, will have an R^2 score of 0.0. However, R^2 can be negative when the numerator is greater than the denominator, the fraction will be greater than 1, therefore the R-squared coefficient will be less than zero [54].

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Equation 20: Coefficient of Determination (R^2)

5.1.5 Explained Variance

Knowing the amount of initial variance that can be explained by the model in a linear regression problem clarifies the extent to which the data is approximated. This concept also helps us understand the extent to which we lose information when approximating the dataset. A minimal value of explained variance (EV) indicates that there are significant fluctuations in the data generation process that cannot be captured by a linear model. EV is a useful metric that is similar to R^2 [57]. The optimal value of EV is 1, which indicates that the model can explain all the variance in the target variable.

$$EV = 1 - \frac{Var[Y - \tilde{Y}]}{Var[Y]}$$

Equation 21: Explained Variance

5.2 ML algorithms

5.2.1 Linear Regression

Linear regression is a modeling technique that assumes a linear connection between the input factors and the output variable. In the case of a singular input variable, it is known as simple linear regression, while in the case of several input variables, it is referred to as multiple linear regression. To determine the coefficients in simple linear regression, statistical methods can be used, which involve calculating statistical properties such as means, standard deviations, correlations, and covariance from the available data. It is necessary to have access to all the data to perform these calculations [58].

5.2.2 Decision Tree Regression

Decision tree learning is a popular technique utilized in data mining [59] that aims to

develop a model that can anticipate the value of a target variable based on multiple input variables. This approach constructs a regression model in the form of a tree structure. It partitions a dataset into increasingly smaller subsets while simultaneously building an associated decision tree. The final product is a tree that consists of decision nodes and leaf nodes. A decision node has two or more branches, each of which represents values for the attribute tested. On the other hand, a leaf node indicates a decision about the numerical target. The highest decision node in a tree, which corresponds to the most reliable predictor, is known as the root node. Decision trees can process both categorical and numerical data [60]. Here are some important terminologies related to Decision tree:

i. Root Node: This represents the entire sample or population, which is further divided into two or more homogeneous subsets.
ii. Splitting: The process of dividing a node into two or more sub-nodes is known as splitting.
iii. Decision Node: When a sub-node splits into further sub-nodes, it is referred to as a decision node.
iv. Leaf/Terminal Node: A node that does not split is called a leaf or terminal node.
v. Pruning: The process of removing sub-nodes of a decision node is called pruning. It can be thought of as the opposite of splitting.
vi. Branch/Sub-Tree: A subsection of the entire tree is referred to as a branch or sub-tree.
vii. Parent and Child Node: A node that is divided into sub-nodes is called the parent node of the sub-nodes, while the sub-nodes are the children of the parent node.

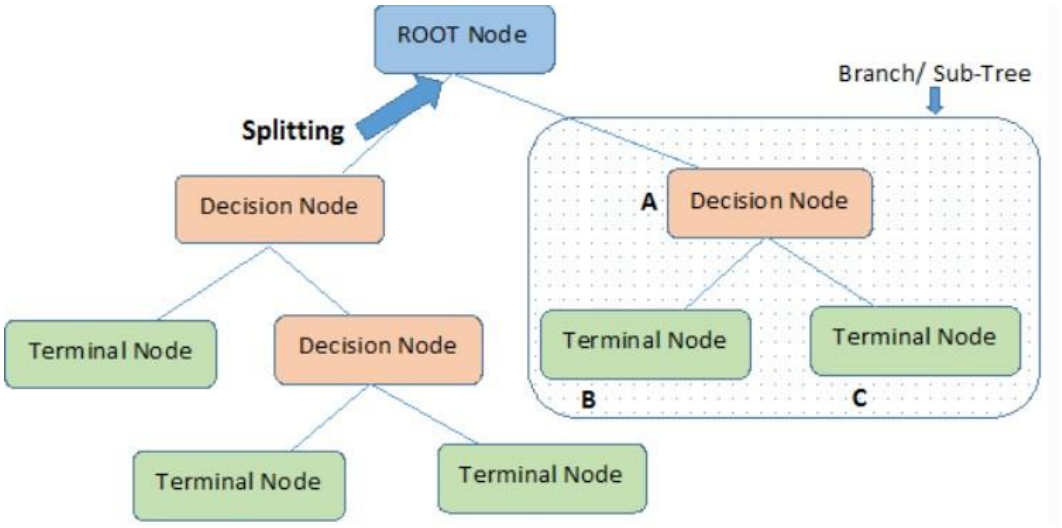


Figure 38: Decision Tree structure (source: www.datacamp.com)

5.2.3 Random Forest Regression

Random forest is a type of Supervised Learning algorithm that employs the ensemble learning approach for classification and regression tasks [61]. Ensemble methods involve combining the predictions of multiple machine learning algorithms to produce more accurate results than any individual model. An ensemble model is comprised of several models working together to make better predictions.

Random forest is a bagging technique and not a boosting technique. In a random forest, decision trees are run in parallel with no interaction between them during the tree-building process. Bootstrap, also known as bagging, refers to the process of random sampling with replacement. Bootstrap helps to better understand the bias and variance of the dataset by randomly sampling small subsets of data from the original dataset. It is a general procedure used to reduce the variance of algorithms with high variance, such as decision trees. Bagging allows each model to run independently and then aggregates the outputs without giving preference to any model.

Random forest creates multiple decision trees during the training phase and outputs the class that has the highest frequency of occurrences (for classification) or the mean prediction (for regression) of the individual trees. A random forest is a meta-estimator that aggregates several decision trees with some beneficial modifications [62]. For instance, the number of features that can be split at each node is limited to a certain percentage of the total, which helps prevent the ensemble model from relying too heavily on any individual feature and makes fair use of all potentially predictive features. Additionally, each tree selects a random sample from the original dataset when generating its splits, adding another element of randomness that helps to prevent overfitting.

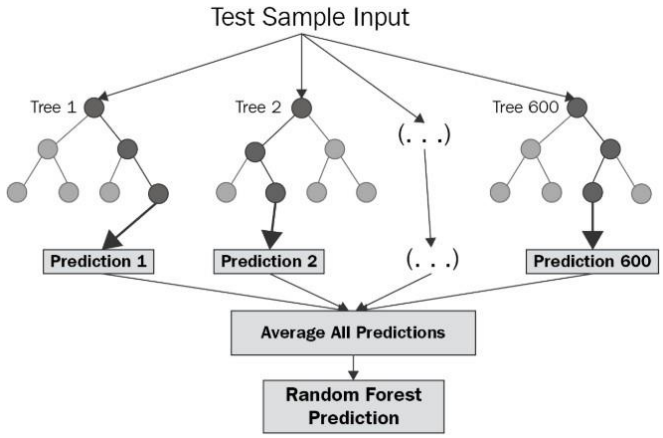


Figure 39: Random Forest Structure (source: www.researchgate.net)

Features and Benefits of Random Forest are:

- It is one of the most precise machine learning algorithms available and can produce a highly accurate classifier for many datasets.
- It runs efficiently on large databases.
- It can handle thousands of input variables without needing to delete any variables.
- It can provide estimates of the importance of variables in classification.
- It generates an unbiased estimate of generalization error as it builds the forest.
- It has an effective method for estimating missing data and can maintain accuracy when a large portion of the data is missing.

The Disadvantages of Random Forest are:

- It can overfit on datasets with noisy classification/regression tasks.
- It is biased in favour of categorical variables with more levels, so the variable importance scores generated by the algorithm may not be reliable for this type of data.

5.2.4 Bayesian Linear Regression

In statistics, Bayesian linear regression is an analytical method for linear regression that applies Bayesian inference to its statistical analysis. This method provides explicit outcomes for the posterior probability distributions of the model's parameters, given that the regression model has normally distributed errors and a specific prior distribution is assumed [63]. Bayesian regression techniques are also useful for integrating regularization parameters into the estimation process, without relying on fixed values, but rather dynamically tuning them to the available data.

Some of the benefits of Bayesian regression are:

- its flexibility to adapt to the available data and
- its capacity to integrate regularization parameters into the estimation procedure.

Nonetheless, a possible downside of Bayesian regression is:

- the time-consuming nature of the model's inference.

5.2.5 K-Nearest Neighbours Regression

K-Nearest Neighbours is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970’s as

a non- parametric technique [64].

A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbours. Another approach uses an inverse distance weighted average of the K nearest neighbours. KNN regression uses the same distance functions as KNN classification.

Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise; however, the compromise is that the distinct boundaries within the feature space are blurred. Cross-validation is another way to retrospectively determine a good K value by using an independent data set to validate your K value. The optimal K for most datasets is 10 or more. That produces much better results than 1-NN.

5.2.6 Kernel Ridge Regression

Kernel ridge regression (KRR) is a combination of ridge regression, which uses linear least squares with l2-norm regularization, and the kernel trick. It learns a linear function in the space created by the kernel and the data, resulting in a non-linear function in the original space when using non-linear kernels. The model learned by KRR is the same as that used in support vector regression (SVR), but different loss functions are used. KRR utilizes squared error loss, while SVR uses epsilon-insensitive loss, both combined with l2 regularization. While fitting a KRR model can be accomplished in closed-form and is typically faster for medium-sized datasets than SVR, the learned model is non-sparse and thus slower than SVR at prediction time since SVR learns a sparse model for epsilon > 0 [65].

5.2.7 Neural Network Regression

A neural network is a machine learning technique or algorithm that attempts to imitate the functioning of neurons in the human brain to facilitate learning. Initially, a neural network may be unstable, but after iterating through the data, it adjusts itself in such a way that its accuracy improves [66].

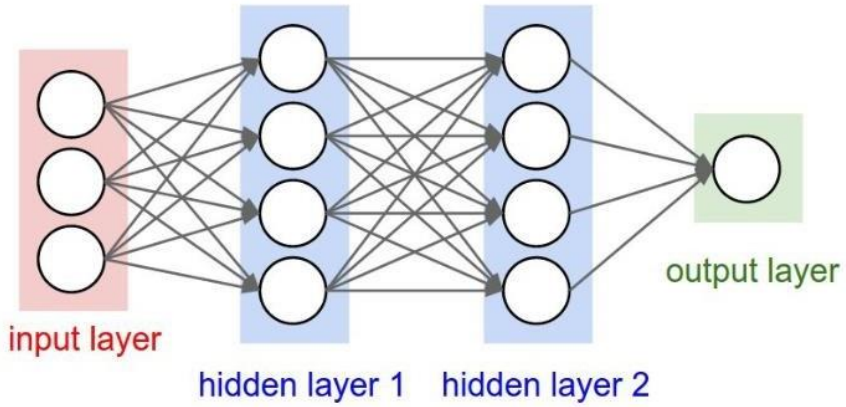


Figure 40: A simple feed neural network (source: www.medium.com)

Neural networks can be reduced to regression models, as a neural network can mimic any type of regression model. Even a very simple neural network with only one input neuron, one hidden neuron, and one output neuron is equivalent to a logistic regression. The neural network takes several dependent variables as input parameters, multiplies them by their weights (equivalent to regression coefficients), and runs them through a sigmoid activation function and a unit step function that closely resemble the logistic regression function with its error term [67]. When this neural network is trained, it will perform gradient descent to find the coefficients that better fit the data, arriving at the optimal linear regression coefficients (or optimal weights in neural network terms) [67].

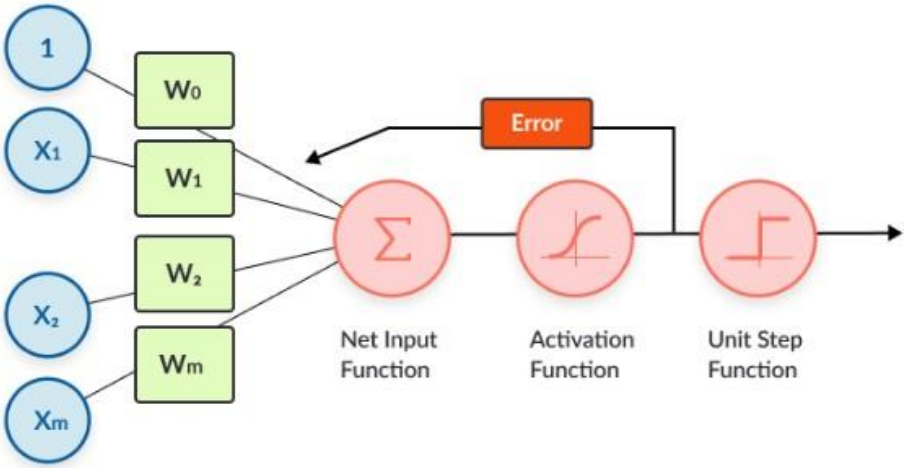


Figure 41: Regression in neural networks (source: www.medium.com)

Chapter 6 – Model Implementation

Synopsis

The opening section of this chapter focuses on the instruments employed in executing the model. The subsequent section deals with data analysis and facilitates comprehension of the data mining process. To enhance the accuracy of Surface Roughness prediction and simplify the process, it is essential to prepare the data carefully. The outlined procedure is similar for both Experiment 1 and Experiment 2, with a few exceptions that will be discussed in the following paragraphs.

6.1 Programming Language

To execute the thesis, Python has been utilized as the primary programming language. Python is a high-level, interpreted, and object-oriented programming language with dynamic semantics. Python's syntax is straightforward, and its emphasis on readability reduces the cost of program maintenance. Modules and packages are supported, which encourages modularity and code reuse. Python is a widely popular, general-purpose, and high-level programming language, ranking in the top 10 according to the latest TIOBE Programming Community Index (Figure 42). The language's simple syntax rules contribute to keeping the code base readable and applications maintainable. Python is preferred over other programming languages for several reasons [68]:

- Readable and Maintainable Code: Python's syntax rules enable the expression of concepts without additional code, emphasizing code readability by using English keywords instead of punctuation. The clean and readable code base of Python reduces the time and effort required to maintain and update the software.
- Compatible with Major Platforms and Systems: Python is compatible with many operating systems and platforms. Its interpreted nature allows for running the same code on different platforms without the need for recompilation.
- Open-Source Frameworks and Tools: Python's open-source nature allows developers to reduce software development costs significantly. Several open-source Python frameworks, libraries, and development tools can also help developers to accelerate development time without increasing development costs.
- Adopt Test Driven Development: Python is a suitable language for rapid creation of

software application prototypes. By adopting the test-driven development (TDD) approach, it becomes possible to perform coding and testing simultaneously in Python.

- Python's Standard Library: Python's extensive and robust standard library sets it apart from other programming languages. The library offers a broad range of modules that can be tailored to suit specific needs. With each module providing additional functionality to the Python application, we can save time and effort by avoiding the need to write new code. Python has a wealth of libraries and frameworks that can simplify the implementation of AI and ML algorithms. These libraries are pre-written code that can be used to solve common programming tasks. Python offers a wide variety of libraries for artificial intelligence and machine learning:

    - Keras, TensorFlow, and Scikit-learn for machine learning;
    - NumPy for high-performance scientific computing and data analysis;
    - SciPy for advanced computing;
    - Pandas for general-purpose data analysis;
    - and Seaborn for data visualization.

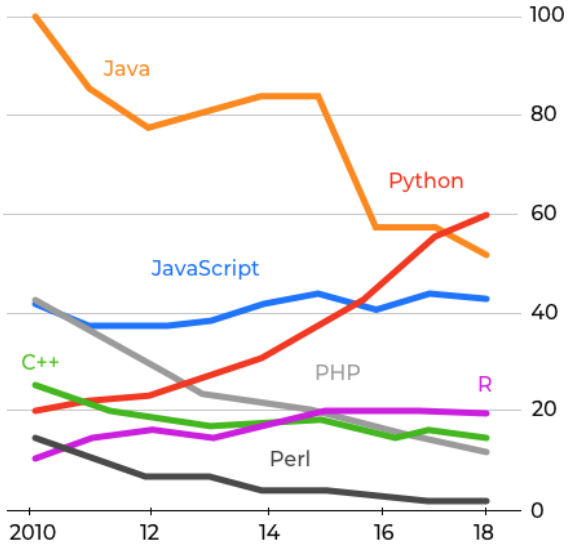


Figure 42: US, Google searches for coding languages,
100 = highest annual traffic for any language.
(source: www.kreyonsystems.com)

6.2 Integrated Development Environment

An Integrated Development Environment, also known as an IDE, is a tool that allows programmers to bring together various components of creating a computer program. IDEs

improve the efficiency of programmers by integrating common activities involved in software development into a single application, including editing source code, building executable files, and debugging [69].

For the implementation of this thesis, Spyder has been utilized as the IDE for running Python. Spyder is a powerful scientific environment created for Python, and tailored to the needs of scientists, engineers, and data analysts. It combines the advanced features of a comprehensive development tool, such as editing, analysis, debugging, and profiling, with the data exploration, interactive execution, deep inspection, and visual representation capabilities of a scientific package [70]. Spyder offers the following functionalities [71]:

- An editor equipped with syntax highlighting, introspection, and code completion;
- Support for multiple IPython consoles;
- The capability to explore and modify variables using a graphical user interface;
- A Help pane that can retrieve and display rich text documentation on functions, classes, and methods automatically or on request;
- A debugger that is linked to IPdb for executing code step-by-step;
- A run-time profiler for evaluating code performance;
- Project support that allows working on multiple development projects simultaneously;
- A "Find in Files" feature that allows for full regular expression search within a specified scope;
- An online help browser that enables users to search and view Python and package documentation within the IDE;
- A history log that records all user commands entered in each console;
- An internal console that permits introspection and control over Spyder's operation.

6.3 Used Libraries

A Python library is a compilation of functions and methods that enable users to execute a variety of tasks without having to write their own code. Below is a summary of the libraries utilized.

6.3.1 Matplotlib

Matplotlib is a Python library that allows users to generate 2-dimensional graphs and plots

using Python scripts. This library is particularly useful for mathematical or scientific applications that require multiple axes in a representation, as it enables users to create multiple plots simultaneously. Additionally, Matplotlib can be used to manipulate various characteristics of figures [72]. Features of Matplotlib include:

- The ability to create high-quality figures that are suitable for publication in hardcopy formats across different interactive platforms.
- Compatibility with various toolkits such as Python Scripts, IPython Shells, Jupyter Notebook, and other graphical user interfaces.
- Integration with third-party libraries such as seaborn, ggplot, and other projection and mapping toolkits such as basemap.

6.3.2 NumPy

NumPy is a widely used package for array processing in Python. It offers a range of tools for managing arrays of different dimensions, as well as matrices. It is known for being fast and efficient [72]. Features of NumPy include:

- The ability to perform modern mathematical implementations on large amounts of data, making project execution easier and hassle-free.
- NumPy creates new arrays when the shape of any N-dimensional array is changed and deletes the old ones.
- NumPy provides useful tools for integration with programming languages such as C, C++, and others.
- NumPy offers functionalities that are comparable to MATLAB, both allowing users to operate at a faster speed.

6.3.3 SciPy

SciPy is a python library that is open-source and can be utilized for both technical and scientific calculations. This python library is free and is a great fit for machine learning. However, SciPy is not only limited to computation tasks, it is also a preferred choice for image editing [72]. Features of SciPy:

- SciPy comes with different modules that are appropriate for optimization, integration, linear algebra, and statistics.

- SciPy efficiently utilizes NumPy arrays for general data structures, as NumPy is an integral part of SciPy.
- Finally, the SciPy community is always there to assist us with regular questions and solve any issues that may arise.

6.3.4 Pandas

Pandas is a software package for Python that is essential to learn for data science. It is specifically designed for the Python language, and is a fast, demonstrative, and adaptable platform that offers user-friendly data structures. With Pandas, we can effortlessly manipulate any type of data, including structured or time-series data [72]. Some of the key features of Pandas include:

- the provision of Series and DataFrames, which allow for easy organization, exploration, representation, and manipulation of data.
- Furthermore, Pandas has some unique features that enable us to handle missing data or values with precision.
- Pandas also provides a collection of built-in tools that allow us to read and write data in different web services, data structures, and databases.
- Additionally, Pandas supports a wide range of formats, including JSON, Excel, CSV, HDF5, and many more.

6.3.5 Scikit Learn

Scikit learn is a simple and useful python machine learning library. Most of it is written in Python, but includes also Cython, C, and C++. It is afree machine learning library. It is a flexible python package that can work in complete harmony with other python libraries and packages such as NumPy and SciPy [72]. Features of Scikit Learn:

- Scikit Learn comes with a clean and neat API. It also provides very useful documentation for beginners.
- It comes with different algorithms: classification, clustering, and regression. It also supports random forests, k-means, gradient boosting, and others.
- This package offers easy adaptability.
- Scikit Learn offers easy methods for data representation. Whether we want to present data as a table or matrix, it is all possible with Scikit Learn.

6.4 Pipeline for model implementation

The depicted image (Figure 43) illustrates the necessary steps to implement the ML models, with the process being identical for both Exp 1 and Exp 2, except for a few variations that will be emphasized when needed in the next paragraphs. Moving forward, the upcoming paragraphs on model implementation will pertain to Exp1 specifically.

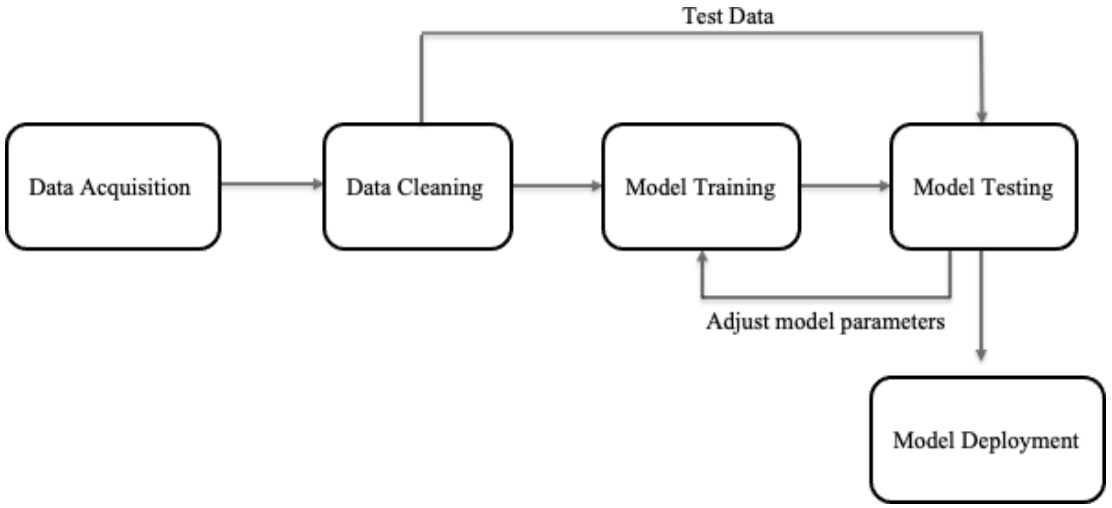


Figure 43: Model Implementation Pipeline

6.4.1 Importing Data

The first step was to arrange the measured responses (also known as dependent variables, output, or labels) and the factors (also known as independent variables, inputs, or features) in the same tables so that they could be easily read by the program. This involved extracting surface roughness and forces, as demonstrated in sections 4.1.5 and 4.1.7, and creating spreadsheets in Comma Separated Values (CSV) format using Python. These spreadsheets included:

- the relevant values from the experiments,
- the experimental factors,
- the responses.

The code used to import the data is presented below for the reader's reference:

```
import os
for dirname, _, filenames in os.walk (r'C:\Users\vitus\Desktop\TESI\File csv'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
exp_file_1 = 'Exp1.csv'
exp_complete_1 = pd.read_csv(r'C:\Users\vitus\Desktop\TESI\File csv\Exp1.csv ')
```

Code Snippet 1: Importing Data

6.4.2 Data Cleansing

Data cleaning, also known as data cleansing, is the process of identifying and correcting or removing errors, inconsistencies, duplicates, and other forms of inaccuracies in a dataset. The goal of data cleaning is to improve the quality and accuracy of the data, ensuring that it is reliable, consistent, and usable for analysis, modeling, and decision-making purposes [73]. This process often involves a combination of manual and automated techniques, including data profiling, data standardization, data validation, and data enrichment.

6.4.2.1 Missing Values

Based on the documentation for the dataset presented in the previous chapter, it is possible that some information from sections 4.1.5 and 4.1.7 may not have been properly extracted during the experiment or that there may be missing data. No entry should be recorded if measurements were not taken. To address this issue, the listwise deletion method, which removes all data for an observation that has one or more missing values, was implemented. This is a good approach, especially if the missing data is limited to a small number of observations, and those cases can be eliminated from the analysis [74].
Out of 4320 values in Experiment 1, there were no non-numeric values and missing values were not found.

To find and delete rows containing possible missing values is used the following code:

```
exp_complete_cleaned_1 = exp_complete_1.dropna()
```

Code Snippet 2: Missing values deletion

6.4.2.2 Drop unnecessary values

Table 11 displays the essential information for experiments, including experiment identification, replica, cutting tool used, group and subgroup (as illustrated in Figure 28), position in workpiece length (as shown in Figure 26), condition, tool condition (which was consistently zero in Experiment 1 since all tools were new with a wedge dimension VB = 0.0 mm), test length, initial and machined diameters, cutting time (not applicable for Experiment 1), surface roughness measurement angle indicating the angular position in which the surface roughness was accessed on the workpiece (as indicated in Figure 26), and run identification (comprising information from the other columns).
These information values in Table 11 maintain data traceability and are necessary for

organizing previous data and accessing a particular value, although they are not utilized in the ML algorithms.

In Experiment 2, the Tool condition column was not removed because the tools' condition (VB) was no longer at 0 for all the Tools, and its information was used in the ML algorithms.

Table 11: Experiment 1 general information.

Exp	Replica	Group	Subgroup	Position	Condition	Tool Condition	Length	Initial diameter	Machined diameter	Cutting time	R measur angle	Run ID
1	1	1	2	a	4	0	12	94	93.5	na	0	1_021_G1_4_a
1	1	1	2	a	4	0	12	94	93.5	na	60	1_021_G1_4_a
1	1	1	2	a	4	0	12	94	93.5	na	120	1_021_G1_4_a
1	1	1	2	a	4	0	12	94	93.5	na	180	1_021_G1_4_a
1	1	1	2	a	4	0	12	94	93.5	na	240	1_021_G1_4_a
1	1	1	2	a	4	0	12	94	93.5	na	300	1_021_G1_4_a

The code used to drop unnecessary values is presented below for the reader's reference:

```
exp_out_1 = exp_complete_cleaned_1.copy()
exp_out_1.drop(['Run_ID', 'Experiment', 'TCond’,
‘Replica','Group', 'Subgroup','Position', 'Condition','Machined_length', 'Init
_diameter', 'Final_diameter', 'CTime', 'R_measurement'],axis=1, inplace=True)
```

Code Snippet 3: Data Cleansing

6.4.2.3 Outlier Analysis

Subsequently, outliers were searched between the factors (a_p, v_c, f) and responses (Ra, Rsk, Rku, RSm, R_t, F_x, F_y, F_z, F) throughout the data frames.

An outlier refers to a data point that deviates from the other data points in the observation. It can be due to normal measurement fluctuations, or it can indicate experimental errors, such as human error or instrument inaccuracy. In data mining activities, outliers can cause significant problems and need to be detected and evaluated in order to understand their presence and, most importantly, to remove them from the dataset [75]. Statistical methods like standard deviation, box plots, z-score, or interquartile distribution (IQR) methods are commonly used to identify outliers. This study will use the IQR method for detecting outliers.

To use the IQR method, one first calculates the IQR of the dataset by finding the difference between the third quartile (Q3) and the first quartile (Q1). Q1 is the value that separates the lowest 25% of the data from the rest, while Q3 separates the lowest 75% of the data from the rest. After calculating the IQR, potential outliers can be identified by determining values

that lie beyond a certain threshold, typically defined as Q1 – 1.5 * IQR or Q3 + 1.5 * IQR. These values are considered outliers because they are more than 1.5 times the IQR away from the nearest quartile.

Using the IQR method for outlier analysis has some advantages over other methods, such as being less sensitive to extreme values like the mean and standard deviation. The IQR method identifies potential outliers by defining a threshold that is based on the distribution of the data, which makes it less affected by extreme values.

However, the IQR method may not be suitable for all datasets, especially those with a small sample size, as it may not provide a reliable estimate of the distribution of the data. In such cases, other methods or visual inspection may be needed to identify outliers.

Instead of removing the outlier, some analysts choose to replace it with the mean of all other observations for the selected condition. This is because the outlier may not be a measurement error or data entry error, but rather a genuine data point that belongs to the population being studied. This can help to reduce the impact of the outlier on the statistical analysis and provide a more accurate estimate of the central tendency of the data. In the specific case of Exp 1, 11 outliers were identified, but they mainly came from parameters that were extracted using a dynamometer and were believed to be genuine data points. Therefore, the outliers were not removed but replaced with their mean to avoid bias in the statistical analysis.

To detect and substitute outliers is used the following code:

```
outliers = pd.DataFrame(columns=exp_out_1.columns)
for i, row in exp_out_1.iterrows():
    if row.isna().any():
        continue
    else:
        q1=exp_out_1.quantile(0.25)
        q3=exp_out_1.quantile(0.75)
        IQR=q3-q1
        if ((row < (q1 - 1.5 * IQR)) | (row > (q3 + 1.5 * IQR))).any():
            outliers.loc[i] = row
print('number of outliers: '+ str(len(outliers)))
print('max outlier value: \n' + str(outliers.max()))
print('min outlier value: \n'+ str(outliers.min()))
outliers

#SET A FUNCTION TO SUBSTITUTE THE VALUE FOR THE OUTLIERS WITH THE MEAN VALUE
exp = exp_out_1.copy()
def impute_outliers_IQR(exp):
   q1=exp.quantile(0.25)
   q3=exp.quantile(0.75)
   IQR=q3-q1
   upper = exp[~(exp>(q3+1.5*IQR))].max()
   lower = exp[~(exp<(q1-1.5*IQR))].min()
   exp = np.where(exp > upper, exp.mean(), np.where(exp < lower, exp.mean(),
   exp))
```

```
    return exp

# outliers substitution
exp[["ap", "vc", "f","Fx", "Fy", "Fz", "F", "Ra","Rz","Rsk","Rku","RSm","Rt"]]
 = impute_outliers_IQR(exp[["ap", "vc", "f","Fx", "Fy", "Fz", "F", "Ra","Rz","
Rsk","Rku","RSm","Rt"]])
exp_describe = exp.describe()[["ap", "vc", "f","Fx", "Fy", "Fz", "F", "Ra","Rz
","Rsk","Rku","RSm","Rt"]]
exp.describe()[["ap", "vc", "f","Fx", "Fy", "Fz", "F", "Ra","Rz","Rsk","Rku","
RSm","Rt"]]
```

Code Snippet 4*: Removal of Outliers*

6.4.3 Model Training and Testing

From the previous operations a dataset is obtained, which has 324 rows and 14 columns. The target variable is 'Ra' and the other columns are features that are used to predict the target variable. One way to approach this task is to train the machine learning model using the entire dataset and then test it on the same dataset to evaluate its accuracy. However, this method would not be reliable as the model would simply memorize the target variable and always provide the same answer. In real-life scenarios, the model needs to be able to accurately predict new, unseen data [76].

To evaluate the model's ability to generalize to new data, it's important to test it on completely different datasets. For instance, if we're training a model to identify animals in images, we can train it on cats and dogs but test it on images of birds from a different source. This way, we can assess whether the model can generalize to new types of animals it hasn't encountered during training. Another reason for testing on different datasets is to evaluate the model's robustness. By testing it on data from different sources, it can be determined how well it performs under different conditions and identify any weaknesses or limitations that need to be addressed. For example, if the model performs well on one dataset but poorly on another, it may suggest that it has learned to overfit to specific patterns in the training data that aren't generalizable to other sources.

A way to assess how well a model can generalize is to use a separate hold-out set for testing, which is distinct from the training data. The 70-30% method is commonly used, where a fixed proportion of the available data is reserved for testing. However, since Experiment 1 dataset only includes 6 Tool IDs (tool_ids = [21,31,41,51,61,71]), the 70-30% method cannot be applied due to the limited number of Tool IDs. Instead, the data is split into Train and Test sets by assigning 5 Tool IDs to the Train set and 1 Tool ID to the Test set. This approach creates a new dataset with data from the Test set, similar to a real-world scenario. In essence, our goal is to create a model that can predict the results of the test set that are

currently unknown. Nevertheless, there are instances where the hold-out set may not accurately reflect the actual data distribution that the model will face in practical scenarios. K-fold cross validation can be used to enhance the model's performance (Figure 45).
Also for Experiment 2 the dataset included 6 Tool IDs (tool_ids = [43, 53, 33, 23, 81, 13]).

To Train and Test the dataset using the Hold – Out method, the following code is used:

```
tool_ids = [21,31,41,51,61,71]

x_train = []
y_train = []
x_test = []
y_test = []

for i in range(len(tool_ids)):
  train_ids = [x for j, x in enumerate(tool_ids) if j != i]
  test_id = tool_ids[i]

  train_indices = exp[exp['Tool_ID'].isin(train_ids)].index
  test_indices = exp[exp['Tool_ID'].isin([test_id])].index

  stratified_train_set = exp.loc[train_indices]
  stratified_test_set = exp.loc[test_indices]

  train = pd.DataFrame(stratified_train_set)
  test = pd.DataFrame(stratified_test_set)
```

Code Snippet 5*: Train -Test dataset split*

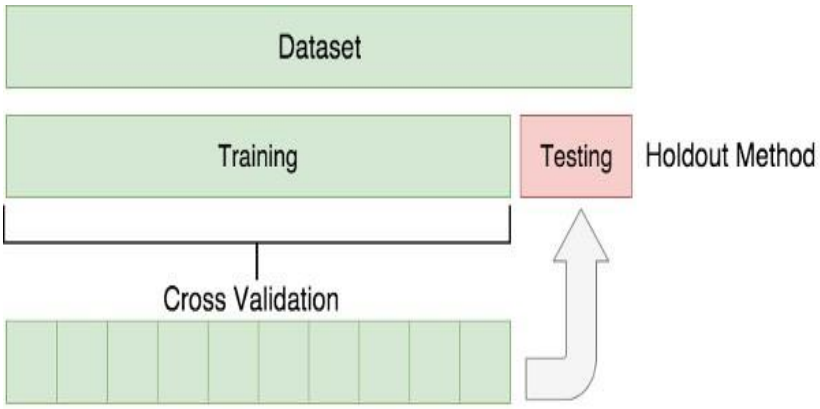


Figure 44: Dataset Split

6.4.4 Cross Validation

Cross-validation is a statistical technique utilized to estimate the efficacy of machine learning models. It is widely applied in practical machine learning to determine and choose a suitable model for a given predictive modeling problem. Cross-validation is popular because it is simple to comprehend, simple to implement, and the skill estimates derived from it usually have lower bias than other methods [77]. As there is often insufficient data to fully train a model, setting aside a portion of it for validation poses a challenge of

underfitting. By reducing the amount of training data, important patterns and trends in the dataset may be lost, increasing the error introduced by bias. Therefore, a method is needed that provides adequate data for model training while still leaving enough data for validation. K-fold cross-validation precisely achieves this scope [78]. Cross-validation is a resampling approach used to assess machine learning models on a limited dataset. The technique employs a single parameter, k, which determines the number of groups to split the dataset into. Consequently, the technique is often referred to as k-fold cross-validation. When a particular value of k is selected, it may be substituted in place of k in the model reference, for instance, k=5 becoming 5-fold cross-validation. The dataset will be divided into 5 equal parts and the procedure below will be executed 5 times, with a different holdout set each time (Figure 46). Cross-validation is primarily employed to estimate the efficacy of a machine learning model on unseen data, i.e., to estimate how the model is likely to perform generally when employed to make predictions on data that were not used during the model's training. For the implemented model K = 6, which is the number of tools used during Experiment 1 and represents the number of groups to split the dataset.

The general procedure is as follows:

1. Split the dataset into k groups
2. For each unique group:
    a. Take the group as a hold out or test data set
    b. Take the remaining groups as a training data set
    c. Fit a model on the training set and evaluate it on the test set
    d. Retain the evaluation score and discard the model
3. Summarize the skill of the model using the sample of model evaluation scores

Crucially, every data point in the sample is allocated to a specific group and remains in that group throughout the process. This ensures that each sample has the chance to be included in the holdout set once and utilized for training the model k-1 times.

The code for Cross Validation will be displayed in the following paragraphs when ML algorithms will be implemented.

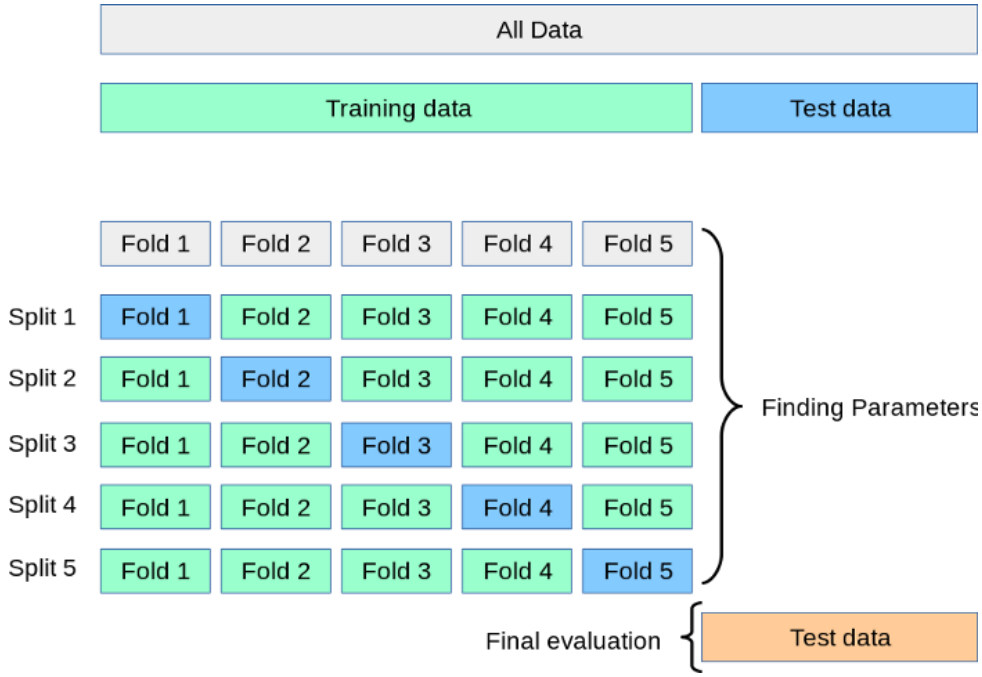


Figure 45: K-Fold Model Tuning

6.4.5 Features and Labels Extraction

To enhance the accuracy and performance efficiency of a model, the dataset is divided into two parts: features (input values, X) and labels (output values, Y) [79]. These pairs of features and labels will be utilized as labeled examples for the ML during the training phase. The potential features may include:

- Process parameters: depth of cut (a_p), feed rate (f), and cutting speed (v_c);
- Measured forces: cutting force (F_x), passive force (F_y), feed force (F_z), and resultant force (F). While machining forces are considered as response variables in the experimental procedure and analysis of results, they are used as input values in the machine learning model.

The process of defining labeled examples is executed through a simple piece of code that eliminates non-required labels and keeps only the ones of interest in the X and y sets. The possible labels are:

- The surface roughness results such as arithmetic mean deviation (Ra), skewness (Rsk), kurtosis (Rku), mean width of profile elements (RSm), and total height (R_t).

The code used to extract feature and labels is presented below for the reader's reference:

```
x_train.append(train[["ap", "vc", "f", "Fx", "Fy", "Fz", "F"]])
x_test.append(test[["ap", "vc", "f", "Fx", "Fy", "Fz", "F"]])
y_train.append(train[["Ra"]])
y_test.append(test[["Ra"]])
```

Code Snippet 6: Features and Labels extraction

6.4.6 Data Standardization

One crucial data transformation that needs to be implemented is feature scaling, which is often used to ensure faster convergence in Machine Learning algorithms. Generally speaking, these algorithms do not perform well when input numerical attributes have significantly different scales. To achieve uniform scaling across all attributes, two common techniques are used: min-max scaling and standardization [80]. The process of min-max scaling (also known as normalization) involves shifting and rescaling the values so that they fall within the range of 0 to 1. This is achieved by subtracting the minimum value and then dividing by the range (max minus min). On the other hand, standardization involves subtracting the mean value (resulting in standardized values with zero mean) and then dividing by the variance, producing a distribution with unit variance. Unlike min-max scaling, standardization does not restrict the values to a specific range, which can be problematic for certain algorithms (e.g., neural networks that expect values between 0 and 1). Nonetheless, standardization is less susceptible to the influence of outliers. To standardize the features, the scikit-learn function called StandardScaler is used, which transforms the values to a uniform range. This process is known as standardization.

Below, the reader can see the implemented code that standardizes the training values:

```
from sklearn.preprocessing import StandardScaler # define the scale method and
 range
scaler = StandardScaler()
for i in range(len(x_train)):
    scaler.fit(x_train[i])
    x_train[i] = scaler.transform(x_train[i])

for i in range(len(x_test)):
    scaler.fit(x_test[i])
    x_test[i] = scaler.transform(x_test[i])
```

Code Snippet 7: Standardization

6.4.7 Evaluation Metrics

To assess the performance of the model, measuring metrics are used. Some of the commonly used and provided within the scikitlearn library are:

- Mean Squared Error
- Root Mean Squared Error
- Mean Absolute Error
- Coefficient of Determination (R^2)
- Explained Variance

The code below is used to add inside the model evaluation metrics:

```
import sklearn.metrics as metrics
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

Code Snippet 8: Importing Evaluation Metrics

6.4.8 ML Algorithms implementation

To move forward with the best model selection for prediction of the response variable, i.e. 'Ra', with the help of one of the libraries of python i.e. Scikitlearn, the following models explained in the next paragraphs are trained and tested, using Cross validation. All the ML models analyzed have the same structure where the code appears to be evaluating the performance of a model using various metrics, such as mean squared error (MSE), mean absolute error (MAE), and R^2 score, among others. It also generates plots to visually compare the model's predictions (y_pred) to the actual target values (y_test).

Overall, the code in the next paragraphs provides a way to assess the performance of a ML model on various datasets and to compare those results visually. Results are printed to the console, displaying the recorded metrics for each pair and the means and standard deviations.

6.4.8.1 Linear Regression

The algorithm outlined below was used to carry out Linear Regression for this study:

```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
lin_error = pd.DataFrame(columns=['RMSE', 'MSE', 'MAE', 'R^2', 'EXPLAINED VARI
ANCE'])

train_test_pairs = [(x_train[i], y_train[i], x_test[i], y_test[i]) for i in ra
nge(len(x_train))]

for i, (x_train, y_train, x_test, y_test) in enumerate(train_test_pairs):
    linreg.fit(x_train, y_train)
    y_pred = linreg.predict(x_test)
    y_pred = pd.DataFrame(y_pred)

    lin_rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
    lin_mse = mean_squared_error(y_test, y_pred)
    lin_mae = mean_absolute_error(y_test, y_pred)
    lin_r2 = metrics.r2_score(y_test, y_pred)
```

```
    lin_variance = metrics.explained_variance_score(y_test, y_pred)
    lin_error.loc[i] = [lin_rmse, lin_mse, lin_mae, lin_r2, lin_variance]

    plt.figure()
    plt.plot(np.arange(len(y_test)), y_test, 'r', label='y_test')
    plt.plot(np.arange(len(y_pred)), y_pred, 'b', label='y_pred')
    plt.scatter(np.arange(len(y_test)), y_test, color='red', s=10)
    plt.scatter(np.arange(len(y_pred)), y_pred, color='blue', s=10)
    plt.title("Tool ID Scenario {} - LinReg: Comparison Chart y_test vs y_pred
".format(i+1))
    plt.xlabel("Sample Index")
    plt.ylabel("Ra")
    plt.show()

mean_rmse = lin_error['RMSE'].mean()
mean_mse = lin_error['MSE'].mean()
mean_mae = lin_error['MAE'].mean()
mean_r2 = lin_error['R^2'].mean()
mean_variance = lin_error['EXPLAINED VARIANCE'].mean()
lin_error.loc['Mean'] = [mean_rmse,mean_mse, mean_mae, mean_r2, mean_variance]

std_rmse = lin_error['RMSE'].std()
std_mse = lin_error['MSE'].std()
std_mae = lin_error['MAE'].std()
std_r2 = lin_error['R^2'].std()
std_variance = lin_error['EXPLAINED VARIANCE'].std()

lin_error.loc['Std Dev'] = [std_rmse, std_mse, std_mae, std_r2, std_variance]
print(lin_error)
```

Code Snippet 9: Linear Regression Algorithm

6.4.8.2 Decision Tree Regression

The following algorithm was employed to execute Decision Tree Regression:

```
from sklearn.tree import DecisionTreeRegressor
decreg = DecisionTreeRegressor(random_state = 42)
dec_error = pd.DataFrame(columns=['RMSE', 'MSE', 'MAE', 'R^2', 'EXPLAINED VARI
ANCE'])
for i, (x_train, y_train, x_test, y_test) in enumerate(train_test_pairs):
    decreg.fit(x_train, y_train)
    y_pred = decreg.predict(x_test)
    y_pred = pd.DataFrame(y_pred)
    #Error calculation
    dec_rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
    dec_mse = mean_squared_error(y_test, y_pred)
    dec_mae = mean_absolute_error(y_test, y_pred)
    dec_r2 = metrics.r2_score(y_test, y_pred)
    dec_variance = metrics.explained_variance_score(y_test, y_pred)
    dec_error.loc[i] = [dec_rmse, dec_mse, dec_mae, dec_r2, dec_variance]

    plt.figure()
    plt.plot(np.arange(len(y_test)), y_test, 'r', label='y_test')
    plt.plot(np.arange(len(y_pred)), y_pred, 'b', label='y_pred')
    plt.scatter(np.arange(len(y_test)), y_test, color='red', s=10)
    plt.scatter(np.arange(len(y_pred)), y_pred, color='blue', s=10)
    plt.title("Tool ID Scenario {} - DT: Comparison Chart y_test vs y_pred".fo
rmat(i+1))
    plt.xlabel("Sample Index")
    plt.ylabel("Ra")
    plt.show()

mean_rmse = dec_error['RMSE'].mean()
mean_mse = dec_error['MSE'].mean()
mean_mae = dec_error['MAE'].mean()
```

```
mean_r2 = dec_error['R^2'].mean()
mean_variance = dec_error['EXPLAINED VARIANCE'].mean()
dec_error.loc['Mean'] = [mean_rmse, mean_mse,mean_mae, mean_r2, mean_variance]

std_rmse = dec_error['RMSE'].std()
std_mse = dec_error['MSE'].std()
std_mae = dec_error['MAE'].std()
std_r2 = dec_error['R^2'].std()
std_variance = dec_error['EXPLAINED VARIANCE'].std()

dec_error.loc['Std Dev'] = [std_rmse, std_mse, std_mae, std_r2, std_variance]
print(dec_error)
```

Code Snippet 10: Decision Tree Regression Algorithm

6.4.8.3 Random forest Regression

Random Forest Regression was implemented using the following algorithm:

```
from sklearn.ensemble import RandomForestRegressor
ranreg = RandomForestRegressor(random_state = 42)
ran_error = pd.DataFrame(columns=['RMSE', 'MSE', 'MAE', 'R^2', 'EXPLAINED VARI
ANCE'])
for i, (x_train, y_train, x_test, y_test) in enumerate(train_test_pairs):
    ranreg.fit(x_train, y_train)
    y_pred = ranreg.predict(x_test)
    y_pred = pd.DataFrame(y_pred)
    #Error calculation
    ran_rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
    ran_mse = mean_squared_error(y_test, y_pred)
    ran_mae = mean_absolute_error(y_test, y_pred)
    ran_r2 = metrics.r2_score(y_test, y_pred)
    ran_variance = metrics.explained_variance_score(y_test, y_pred)
    ran_error.loc[i] = [ran_rmse, ran_mse, ran_mae, ran_r2, ran_variance]

    plt.figure()
    plt.plot(np.arange(len(y_test)), y_test, 'r', label='y_test')
    plt.plot(np.arange(len(y_pred)), y_pred, 'b', label='y_pred')
    plt.scatter(np.arange(len(y_test)), y_test, color='red', s=10)
    plt.scatter(np.arange(len(y_pred)), y_pred, color='blue', s=10)
    plt.title("Tool ID Scenario {} - RF: Comparison Chart y_test vs y_pred".fo
rmat(i+1))
    plt.xlabel("Sample Index")
    plt.ylabel("Ra")
    plt.show()

mean_rmse = ran_error['RMSE'].mean()
mean_mse = ran_error['MSE'].mean()
mean_mae = ran_error['MAE'].mean()
mean_r2 = ran_error['R^2'].mean()
mean_variance = ran_error['EXPLAINED VARIANCE'].mean()
ran_error.loc['Mean'] = [mean_rmse, mean_mse,mean_mae, mean_r2, mean_variance]

std_rmse = ran_error['RMSE'].std()
std_mse = ran_error['MSE'].std()
std_mae = ran_error['MAE'].std()
std_r2 = ran_error['R^2'].std()
std_variance = ran_error['EXPLAINED VARIANCE'].std()

ran_error.loc['Std Dev'] = [std_rmse, std_mse, std_mae, std_r2, std_variance]
print(ran_error)
```

Code Snippet 11: Random Forest Regression Algorithm

6.4.8.4 Bayesian Linear Regression

By following the given algorithm, Bayesian Linear Regression was implemented:

```
from sklearn.linear_model import BayesianRidge
bayreg = BayesianRidge()
bay_error = pd.DataFrame(columns=['RMSE', 'MSE', 'MAE', 'R^2', 'EXPLAINED VARI
ANCE'])

for i, (x_train, y_train, x_test, y_test) in enumerate(train_test_pairs):
    bayreg.fit(x_train, y_train)
    y_pred = bayreg.predict(x_test)
    y_pred = pd.DataFrame(y_pred)
    #Error calculation
    bay_rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
    bay_mse = mean_squared_error(y_test, y_pred)
    bay_mae = mean_absolute_error(y_test, y_pred)
    bay_r2 = metrics.r2_score(y_test, y_pred)
    bay_variance = metrics.explained_variance_score(y_test, y_pred)
    bay_error.loc[i] = [bay_rmse, bay_mse, bay_mae, bay_r2, bay_variance]

    plt.figure()
    plt.plot(np.arange(len(y_test)), y_test, 'r', label='y_test')
    plt.plot(np.arange(len(y_pred)), y_pred, 'b', label='y_pred')
    plt.scatter(np.arange(len(y_test)), y_test, color='red', s=10)
    plt.scatter(np.arange(len(y_pred)), y_pred, color='blue', s=10)
    plt.title("Tool ID Scenario {} - Bayesian: Comparison Chart y_test vs y_pr
ed".format(i+1))
    plt.xlabel("Sample Index")
    plt.ylabel("Ra")
    plt.show()

mean_rmse = bay_error['RMSE'].mean()
mean_mse = bay_error['MSE'].mean()
mean_mae = bay_error['MAE'].mean()
mean_r2 = bay_error['R^2'].mean()
mean_variance = bay_error['EXPLAINED VARIANCE'].mean()
bay_error.loc['Mean'] = [mean_rmse,mean_mse, mean_mae, mean_r2, mean_variance]

std_rmse = bay_error['RMSE'].std()
std_mse = bay_error['MSE'].std()
std_mae = bay_error['MAE'].std()
std_r2 = bay_error['R^2'].std()
std_variance = bay_error['EXPLAINED VARIANCE'].std()

bay_error.loc['Std Dev'] = [std_rmse, std_mse, std_mae, std_r2, std_variance]
print(bay_error)
```

Code Snippet 12: Bayesian Linear Regression Algorithm

6.4.8.5 K-Nearest Neighbours Regression

Using the algorithm specified below, K-Nearest Neighbours Regression has been implemented:

```
from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor(n_neighbors=10, weights = 'distance')
knn_error = pd.DataFrame(columns=['RMSE', 'MSE', 'MAE', 'R^2', 'EXPLAINED VARI
ANCE'])

for i, (x_train, y_train, x_test, y_test) in enumerate(train_test_pairs):
    knn.fit(x_train, y_train)
    y_pred = knn.predict(x_test)
    y_pred = pd.DataFrame(y_pred)
    #Error calculation
```

```
    knn_rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
    knn_mse = mean_squared_error(y_test, y_pred)
    knn_mae = mean_absolute_error(y_test, y_pred)
    knn_r2 = metrics.r2_score(y_test, y_pred)
    knn_variance = metrics.explained_variance_score(y_test, y_pred)
    knn_error.loc[i] = [knn_rmse, knn_mse, knn_mae, knn_r2, knn_variance]

    plt.figure()
    plt.plot(np.arange(len(y_test)), y_test, 'r', label='y_test')
    plt.plot(np.arange(len(y_pred)), y_pred, 'b', label='y_pred')
    plt.scatter(np.arange(len(y_test)), y_test, color='red', s=10)
    plt.scatter(np.arange(len(y_pred)), y_pred, color='blue', s=10)
    plt.title("Tool ID Scenario {} - K-
NN: Comparison Chart y_test vs y_pred".format(i+1))
    plt.xlabel("Sample Index")
    plt.ylabel("Ra")
    plt.show()

mean_rmse = knn_error['RMSE'].mean()
mean_mse = knn_error['MSE'].mean()
mean_mae = knn_error['MAE'].mean()
mean_r2 = knn_error['R^2'].mean()
mean_variance = knn_error['EXPLAINED VARIANCE'].mean()
knn_error.loc['Mean'] = [mean_rmse,mean_mse, mean_mae, mean_r2, mean_variance]

std_rmse = knn_error['RMSE'].std()
std_mse = knn_error['MSE'].std()
std_mae = knn_error['MAE'].std()
std_r2 = knn_error['R^2'].std()
std_variance = knn_error['EXPLAINED VARIANCE'].std()

knn_error.loc['Std Dev'] = [std_rmse, std_mse, std_mae, std_r2, std_variance]
print(knn_error)
```

Code Snippet 13: KNN Regression Algorithm

6.4.8.6 Kernel Ridge Regression

Kernel Ridge Regression has been employed in the current study, with the algorithm provided below:

```
from sklearn.kernel_ridge import KernelRidge
kreg = KernelRidge(alpha=0.5, kernel='rbf', gamma =0.5)
kernel_error = pd.DataFrame(columns=['RMSE', 'MSE', 'MAE', 'R^2', 'EXPLAINED V
ARIANCE'])

for i, (x_train, y_train, x_test, y_test) in enumerate(train_test_pairs):
    kreg.fit(x_train, y_train)
    y_pred = kreg.predict(x_test)
    y_pred = pd.DataFrame(y_pred)
    #Error calculation
    kernel_rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
    kernel_mse = mean_squared_error(y_test, y_pred)
    kernel_mae = mean_absolute_error(y_test, y_pred)
    kernel_r2 = metrics.r2_score(y_test, y_pred)
    kernel_variance = metrics.explained_variance_score(y_test, y_pred)
    kernel_error.loc[i] = [kernel_rmse, kernel_mse, kernel_mae, kernel_r2, ker
nel_variance]

    plt.figure()
    plt.plot(np.arange(len(y_test)), y_test, 'r', label='y_test')
    plt.plot(np.arange(len(y_pred)), y_pred, 'b', label='y_pred')
    plt.scatter(np.arange(len(y_test)), y_test, color='red', s=10)
    plt.scatter(np.arange(len(y_pred)), y_pred, color='blue', s=10)
```

```
    plt.title("Tool ID Scenario {} - Kernel: Comparison Chart y_test vs y_pred
".format(i+1))
    plt.xlabel("Sample Index")
    plt.ylabel("Ra")
    plt.show()

mean_rmse = kernel_error['RMSE'].mean()
mean_mse = kernel_error['MSE'].mean()
mean_mae = kernel_error['MAE'].mean()
mean_r2 = kernel_error['R^2'].mean()
mean_variance = kernel_error['EXPLAINED VARIANCE'].mean()
kernel_error.loc['Mean'] = [mean_rmse, mean_mse, mean_mae, mean_r2, mean_varia
nce]

std_rmse = kernel_error['RMSE'].std()
std_mse = kernel_error['MSE'].std()
std_mae = kernel_error['MAE'].std()
std_r2 = kernel_error['R^2'].std()
std_variance = kernel_error['EXPLAINED VARIANCE'].std()

kernel_error.loc['Std Dev'] = [std_rmse,std_mse,std_mae, std_r2, std_variance]

print(kernel_error)
```

Code Snippet 14: Kernel Ridge Regression Algorithm

6.4.8.7 Neural Network Regression

By utilizing the algorithm given below, Neural Network Regression was implemented:

```
from sklearn.neural_network import MLPRegressor
mlpreg = MLPRegressor(random_state = 1)
neu_error = pd.DataFrame(columns=['RMSE', 'MSE', 'MAE', 'R^2', 'EXPLAINED VARI
ANCE'])

for i, (x_train, y_train, x_test, y_test) in enumerate(train_test_pairs):
    mlpreg.fit(x_train, y_train)
    y_pred = mlpreg.predict(x_test)
    y_pred = pd.DataFrame(y_pred)
    #Error calculation
    neu_rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
    neu_mse = mean_squared_error(y_test, y_pred)
    neu_mae = mean_absolute_error(y_test, y_pred)
    neu_r2 = metrics.r2_score(y_test, y_pred)
    neu_variance = metrics.explained_variance_score(y_test, y_pred)
    neu_error.loc[i] = [neu_rmse, neu_mse, neu_mae, neu_r2, neu_variance]

    plt.figure()
    plt.plot(np.arange(len(y_test)), y_test, 'r', label='y_test')
    plt.plot(np.arange(len(y_pred)), y_pred, 'b', label='y_pred')
    plt.scatter(np.arange(len(y_test)), y_test, color='red', s=10)
    plt.scatter(np.arange(len(y_pred)), y_pred, color='blue', s=10)
    plt.title("Tool ID Scenario {} - Neural Network: Comparison Chart y_test v
s y_pred".format(i+1))
    plt.xlabel("Sample Index")
    plt.ylabel("Ra")
    plt.show()

mean_rmse = neu_error['RMSE'].mean()
mean_mse = neu_error['MSE'].mean()
mean_mae = neu_error['MAE'].mean()
mean_r2 = neu_error['R^2'].mean()
mean_variance = neu_error['EXPLAINED VARIANCE'].mean()
neu_error.loc['Mean'] = [mean_rmse,mean_mse, mean_mae, mean_r2, mean_variance]
```

```
std_rmse = neu_error['RMSE'].std()
std_mse = neu_error['MSE'].std()
std_mae = neu_error['MAE'].std()
std_r2 = neu_error['R^2'].std()
std_variance = neu_error['EXPLAINED VARIANCE'].std()

neu_error.loc['Std Dev'] = [std_rmse, std_mse, std_mae, std_r2, std_variance]
print(neu_error)
```

Code Snippet 15: Neural Network Regression Algorithm

Chapter 7 - Results and Discussion

Synopsis

In the previous chapter, different models were trained and tested to evaluate their performance using separate training and testing datasets. In a real-world scenario, not all models can be utilized, and therefore, one model must be chosen. One method to achieve this is by assessing the error metrics and selecting the one with the best metric performance. However, a question arises as to whether the method of measuring error metrics is efficient and consistent. To obtain conclusive results, all models are subjected to cross-validation to finalize the models for evaluating the surface roughness of the machined steel bar. The results and conclusions are discussed below.

7.1 Experiment 1

7.1.1 Linear Regression

The details on the performance of the model in each fold is as shown in table 12:

Table 12: Linear Regression performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
21	1,3370	1,7870	1,0444	-1,7934	-17,9252
31	1,5411	2,3749	1,1235	-42,8900	-42,1391
41	0,2789	0,0778	0,2126	0,3871	0,4410
51	1,9361	3,7483	1,4162	-35,6652	-35,3488
61	1,0715	1,1482	0,8222	-13,4892	-13,4352
71	0,1413	0,0200	0,1157	0,2986	0,3387
Mean	**1,0510**	**1,5261**	**0,7891**	**-18,2155**	**-18,0114**
Std Dev	**0,6493**	**1,3062**	**0,4756**	**16,4548**	**16,2322**

Overall, the model's performance is poor, as indicated by the low R^2 and Explained Variance scores. Additionally, there is significant variability in the model's performance across different folds, as indicated by the high standard deviation scores for R^2 and Explained Variance. The high standard deviation scores for MSE and RMSE also suggest that the model's performance is highly variable across different folds.

7.1.2 Decision Tree Regression

Table 13 shows the performance of Decision Tree Regression:

Table 13: Decision Tree Regression performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
21	0,1470	0,0216	0,1114	0,7711	0,7742
31	0,2929	0,0858	0,1921	-0,5854	-0,2665
41	0,2258	0,0510	0,1725	0,5982	0,6017
51	0,2315	0,0536	0,1851	0,4757	0,6219
61	0,2190	0,0480	0,1612	0,3949	0,4229
71	0,2132	0,0454	0,1877	-0,5976	-0,5662
Mean	**0,2216**	**0,0509**	**0,1683**	**0,1762**	**0,2647**
Std Dev	**0,0425**	**0,0188**	**0,0275**	**0,5550**	**0,4997**

Overall, the model shows acceptable performance in RMSE and MAE, with modest mean values and relatively insignificant standard deviations. However, the R^2 values suggest that the model may not be an appropriate fit for the data as the mean value is not high, and the standard deviation is high. Furthermore, the Explained Variance indicates significant variability across folds and Tool IDs, which implies that the model may not capture the entire data variability.

7.1.3 Random Forest Regression

Table 14 shows the scores for Random Forest Regression:

Table 14: Random Forest Regression performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
21	0,1367	0,0187	0,1187	0,8020	0,8155
31	0,2436	0,0593	0,1792	-0,0964	0,2871
41	0,2521	0,0636	0,1890	0,4993	0,5177
51	0,2556	0,0653	0,2150	0,3608	0,6048
61	0,1667	0,0278	0,1327	0,6494	0,6519
71	0,2100	0,0441	0,1888	-0,5496	-0,5002
Mean	**0,2108**	**0,0465**	**0,1706**	**0,2776**	**0,3962**
Std Dev	**0,0451**	**0,0180**	**0,0338**	**0,4643**	**0,4311**

In general, the model seems to exhibit good performance, as evidenced by the low values of RMSE, MSE, and MAE. However, the R^2 and Explained Variance values are improving, but they may not account for all the variation present in the data. Moreover, the

high standard deviations for R^2 and Explained Variance suggest that the model's performance may vary considerably depending on the specific fold of data being assessed.

7.1.4 Bayesian Linear Regression

Table 15 shows the scores for Bayesian Linear Regression:

Table 15: Bayesian Linear Regression

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
21	1,1636	1,3540	0,9060	-13,3410	-13,3319
31	1,2617	1,5918	0,8955	-28,4172	-27,6663
41	0,3089	0,0954	0,2356	0,2483	0,3023
51	1,7447	3,0440	1,2872	-28,7759	-28,4595
61	0,9125	0,8327	0,7066	-9,5083	-9,4542
71	0,1483	0,0220	0,1273	0,2266	0,2667
Mean	**0,9233**	**1,1567**	**0,6930**	**-13,2613**	**-13,0572**
Std Dev	**0,5515**	**1,0254**	**0,4019**	**11,8949**	**11,6884**

Overall, the model exhibits satisfactory performance concerning RMSE and MAE, with minimal mean values and relatively insignificant standard deviations. However, based on the R^2 values, it appears that the model might not be a suitable match for the data as the mean value is low, and the standard deviation is high. The Explained Variance further highlights significant variability across folds and Tool IDs, which implies that the model may not fully capture all the data's variability.

7.1.5 K-Nearest Neighbours

Table 16 shows the scores for KNN:

Table 16: K-Nearest Neighbours performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
21	0,1276	0,0163	0,0951	0,8277	0,8286
31	0,2556	0,0654	0,1978	-0,2077	0,1685
41	0,3043	0,0926	0,2290	0,2706	0,4492
51	0,2856	0,0815	0,2194	0,2023	0,6075
61	0,2018	0,0407	0,1624	0,4863	0,4873
71	0,1818	0,0331	0,1548	-0,1619	0,2937
Mean	**0,2261**	**0,0549**	**0,1764**	**0,2362**	**0,4725**
Std Dev	**0,0616**	**0,0271**	**0,0454**	**0,3583**	**0,2122**

The mean performance metrics of the model are satisfactory, as indicated by the low values

of RMSE, MSE, and MAE, and positive R^2 and Explained Variance. Additionally, the standard deviation is relatively low, indicating consistent performance across the folds. However, it is worth mentioning that the R^2 values were negative for some folds, suggesting that the model performed worse than a simple baseline model in those instances. Conversely, in one of the folds, the model achieved an R^2 value of 0.8277, indicating a good fit to the data. Overall, the model's performance seems to be moderately good, but there is room for improvement, particularly in cases where the R^2 value is negative.

7.1.6 Kernel Ridge Regression

Table 17 shows the scores for Kernel Ridge Regression:

Table 17: Kernel Ridge Regression performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
21	0,3388	0,1148	0,2710	-0,2161	0,5108
31	0,2389	0,0571	0,1627	-0,0546	0,1310
41	0,4931	0,2431	0,3969	-0,9153	0,3260
51	0,5604	0,3141	0,4846	-2,0724	0,2247
61	0,3471	0,1205	0,2856	-0,5206	0,3134
71	0,3537	0,1251	0,3411	-3,3973	0,6935
Mean	**0,3887**	**0,1624**	**0,3237**	**-1,1961**	**0,3666**
Std Dev	**0,1067**	**0,0876**	**0,1014**	**1,1840**	**0,1861**

The model's performance seems to be moderate, with relatively high variance overall. The negative R^2 and low Explained Variance values suggest that the model might not be suitable for the data. Additionally, the standard deviations of the metrics indicate that the model's performance variability across the six folds is high for R^2, Explained Variance, and MAE, while it is moderate for RMSE and MSE.

7.1.7 Neural Networks

Table 18 shows the scores for the neural networks:

Table 18: Neural Networks performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
21	0,2634	0,0694	0,2209	0,2650	0,4392
31	0,2413	0,0582	0,2050	-0,0761	-0,0629
41	0,4051	0,1641	0,3205	-0,2929	0,1351
51	0,4061	0,1649	0,3216	-0,6133	0,3404

61	0,2827	0,0799	0,1938	-0,0088	0,1907
71	0,2585	0,0668	0,1836	-1,3497	-0,6127
Mean	**0,3095**	**0,1006**	**0,2409**	**-0,3460**	**0,0716**
Std Dev	**0,0690**	**0,0457**	**0,0578**	**0,5231**	**0,3445**

In general, these findings indicate that the model's performance is not performing well, as indicated by the negative or near-zero values of R-squared and Explained Variance. Moreover, the RMSE and MSE values indicate a considerable amount of error in the model's predictions, while the MAE values suggest that the average absolute error is relatively high. The high standard deviations also suggest that the model's performance varies significantly across different folds and tools.

7.1.8 Experiment 1 – Final considerations

From the error metric scores, following models were shortlisted:

- K-Nearest Neighbours
- Random Forest Regression
- Decision Tree Regression

The K-Nearest Neighbours approach produced results with reliable metric accuracy, making it a viable option for detecting Surface Roughness. In Chapter 8, the framework used to compare the predicted Surface Roughness from the KNN model with actual values and a theoretical equation will be discussed.

7.2 Experiment 2

7.2.1 Linear Regression

Table 19 shows the scores for Linear Regression:

Table 19: Linear Regression performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
43	0,3913	0,1531	0,3683	-0,6716	-0,3635
53	0,3831	0,1468	0,3182	-0,9144	-0,5292
33	0,3105	0,0964	0,2227	-15,6087	-10,9125
23	0,2382	0,0567	0,1862	-12,8279	-1,1995
81	0,1771	0,0313	0,1606	-0,0936	-0,0935
13	0,3180	0,1011	0,2617	-8,4823	-7,1834
Mean	**0,3030**	**0,0976**	**0,2530**	**-4,5089**	**-3,3803**

Std Dev	**0,0759**	**0,0439**	**0,0725**	**5,7233**	**4,1631**

Based on the mean values, it appears that the model's performance is acceptable in terms of RMSE and MAE, but not satisfactory in terms of R^2 and Explained Variance. Furthermore, the high standard deviations suggest that there is a considerable amount of variability in the model's performance across different folds.

7.2.2 Decision Tree Regression

Table 20 shows the scores for Decision Tree Regression:

Table 20: Decision Tree Regression performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
43	0,4004	0,1603	0,3378	-0,7505	0,0205
53	0,3376	0,1140	0,3036	-0,4864	0,4018
33	0,2964	0,0878	0,2780	-14,1322	-3,6979
23	0,1957	0,0383	0,1617	-0,5409	0,3122
81	0,2385	0,0569	0,1835	-0,9840	-0,9148
13	0,2237	0,0500	0,1863	-3,6917	-2,7089
Mean	**0,2820**	**0,0846**	**0,2418**	**-3,4310**	**-1,0979**
Std Dev	**0,0708**	**0,0423**	**0,0674**	**4,9122**	**1,5746**

The mean values of RMSE, MSE, and MAE are lower than the standard deviations, which suggests that the model's performance varies significantly across different folds. The mean R^2 value is negative, indicating that the model is not a good fit for the data. Moreover, the mean explained variance value is also negative, suggesting that the model explains less variance than the mean of the observed values. Overall, the model's performance is poor, as indicated by the negative R^2 and explained variance values, as well as the high standard deviations of the evaluation metrics.

7.2.3 Random Forest Regression

Table 21 shows the scores for Random Forest Regression:

Table 21: Random Forest Regression performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
43	0,3097	0,0959	0,2861	-0,0472	0,4325
53	0,2972	0,0883	0,2563	-0,1524	0,3466
33	0,2454	0,0602	0,2414	-9,3725	-2,7410

23	0,1503	0,0226	0,1263	0,0907	0,1170
81	0,1434	0,0206	0,1008	0,2830	0,2835
13	0,2023	0,0409	0,1688	-2,8392	-2,0066
Mean	**0,2247**	**0,0548**	**0,1966**	**-2,0063**	**-0,5947**
Std Dev	**0,0653**	**0,0296**	**0,0689**	**3,4609**	**1,2793**

Overall, the model's performance varies significantly across different folds and tool IDs, as indicated by the high standard deviations. However, the model's performance, on average, shows relatively low error rates and a moderate ability to explain the variance in the data.

7.2.4 Bayesian Linear Regression

Table 22 shows the scores for Bayesian Linear Regression:

Table 22: Bayesian Linear Regression performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
43	0,2645	0,0699	0,2541	0,2363	0,5445
53	0,3662	0,1341	0,3010	-0,7493	-0,3641
33	0,2966	0,0880	0,2173	-14,1559	-9,4597
23	0,2058	0,0424	0,1824	-0,7049	-0,6216
81	0,1649	0,0272	0,1395	0,0516	0,0517
13	0,1359	0,0185	0,1177	-0,7314	0,5676
Mean	**0,2390**	**0,0633**	**0,2020**	**-2,6756**	**-1,5469**
Std Dev	**0,0789**	**0,0396**	**0,0634**	**5,1492**	**3,5653**

The metrics' standard deviations are high, which suggests that the model's performance differs considerably across the various folds. Moreover, the negative R^2 and explained variance values indicate that the model does not fit the data well.

7.2.5 K-Nearest Neighbours

Table 23 shows the scores for KNN:

Table 23: K-Nearest Neighbours performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
43	0,3060	0,0936	0,2877	-0,0224	0,4136
53	0,2991	0,0895	0,2576	-0,1672	0,1331
33	0,2667	0,0711	0,2395	-11,2503	-3,5823
23	0,1036	0,0107	0,0930	0,5678	0,5728
81	0,1398	0,0196	0,0970	0,3177	0,5197
13	0,2344	0,0550	0,2009	-4,1531	-0,8085

Mean	**0,2250**	**0,0566**	**0,1960**	**-2,4513**	**-0,4586**
Std Dev	**0,0773**	**0,0320**	**0,0759**	**4,2469**	**1,4726**

The evaluation metrics vary significantly across different folds, with some folds having negative R^2 or explained variance values, indicating poor model fit. Furthermore, the high standard deviations suggest that there is a considerable amount of variability in the model's performance across different folds. Overall, the model shows some potential for prediction, but further analysis and improvement may be required.

7.2.6 Kernel Ridge Regression

Table 24 shows the scores for Kernel Ridge Regression:

Table 24: Kernel Ridge Regression performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
43	0,3893	0,1515	0,2540	-0,6545	-0,0872
53	0,5177	0,2680	0,4672	-2,4961	0,3514
33	0,2108	0,0444	0,1924	-6,6552	-0,2775
23	0,4670	0,2181	0,4382	-7,7742	-0,0477
81	0,3866	0,1495	0,3437	-4,2150	-0,0949
13	0,4723	0,2231	0,4566	-19,9167	-0,3676
Mean	**0,4073**	**0,1758**	**0,3587**	**-6,9520**	**-0,0873**
Std Dev	**0,0995**	**0,0720**	**0,1053**	**6,2697**	**0,2269**

The models' performance is subpar, with negative R^2 values observed in certain instances. The standard deviation of these metrics is quite high, implying that the models' performance differs significantly across different folds and tool IDs.

7.2.7 Neural Networks

Table 25 shows the scores for the Neural Networks:

Table 25: Neural Networks performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
43	0,2416	0,0584	0,2109	0,3628	0,3969
53	0,4481	0,2008	0,3745	-1,6193	-0,4881
33	0,1911	0,0365	0,1708	-5,2884	-4,8873
23	0,4360	0,1901	0,3887	-6,6505	-2,1438
81	0,3498	0,1224	0,3235	-3,2683	0,3780
13	0,3283	0,1078	0,3017	-9,1083	-0,5756

Mean	**0,3325**	**0,1193**	**0,2950**	**-4,2620**	**-1,2200**
Std Dev	**0,0937**	**0,0611**	**0,0801**	**3,1519**	**1,8449**

The provided standard deviations for each metric across all six folds indicate that the machine learning model is not performing well. The model has high error rates and negative R^2 and explained variance values, suggesting that it may not be a good fit for the data.

7.2.8 Experiment 2 – Final Considerations

From the error metric scores, following models were selected:

- Random Forest Regression
- K-Nearest Neighbours

The Random Forest Regression method has the more accurate metric results, making it a feasible alternative for detecting Surface Roughness. Chapter 8 will cover the methodology employed to compare the predicted Surface Roughness obtained from the Random Forest Regression model against real values and a theoretical equation.

Chapter 8 – ML models vs Theoretical Equation

Synopsis

Chapter 7 provides the basis for this chapter's findings. The K-Nearest Neighbours algorithm demonstrated higher accuracy in Experiment 1, whereas the Random Forest Regression method showed greater accuracy in Experiment 2. The theoretical equation for Ra (Eq.14) in Table 6 will be utilized in both experiments, and its predictive capabilities will be compared to those of the machine learning models identified in the previous chapter.

$$Ra = \frac{0.0321 * f^2}{r\varepsilon} \qquad \text{Eq. 14}$$

The parameters contained in Eq. 14 are the feed rate and the Edge radius ($r\varepsilon$ = 0.397 mm) which is a geometric specification (Table 10) of the cutting tool used for the Experiments.

8.1 Experiment 1: KNN vs Theoretical Equation

Below it is shown the procedure to calculate the dataset containing the values for Ra_theoretical using the theoretical equation (Eq.14):

```
tool_ids = [21,31,41,51,61,71]

x_train = []
y_train = []
x_test = []
y_test = []
y_theoretical = []

for i in range(len(tool_ids)):
  train_ids = [x for j, x in enumerate(tool_ids) if j != i]
  test_id = tool_ids[i]

  train_indices = exp[exp['Tool_ID'].isin(train_ids)].index
  test_indices = exp[exp['Tool_ID'].isin([test_id])].index

  stratified_train_set = exp.loc[train_indices]
  stratified_test_set = exp.loc[test_indices]

  train = pd.DataFrame(stratified_train_set)
  test = pd.DataFrame(stratified_test_set)

  # DEFINE FEATURES AND LABELS:
  x_train.append(train[["ap", "vc", "f", "Fx", "Fy", "Fz", "F"]])
  x_test.append(test[["ap", "vc", "f", "Fx", "Fy", "Fz", "F"]])
  y_train.append(train[["Ra"]])
  y_test.append(test[["Ra"]])

  f = x_test[i]["f"].values
  Ra_theor = ((0.0321 * f**2) / 0.397) * 1000 # Ra is expressed in um
  y_theoretical.append(Ra_theor)
print(f"\nDataFrame {i}:\n{x_test[i]}\n\n
Ra theoretical DataFrame {i}:\n{y_theoretical[i]}")
y_theoretical = np.array(y_theoretical)
```

Code Snippet 16: Dataframe creation for y_theoretical

The new dataframe containing the values for Ra theoretical has been created and the goal is to compare the actual values contained in y_test with y_theoretical, calculating error metrics that will be compared with the error metrics calculated in Chapter 7 for the KNN algorithm.

```
rmse2 = np.sqrt(metrics.mean_squared_error(y_test,y_theoretical))
mse2 = mean_squared_error(y_test,y_theoretical)
mae2 = mean_absolute_error(y_test,y_theoretical)
r2T = metrics.r2_score(y_test,y_theoretical)
variance2 = metrics.explained_variance_score(y_test,y_theoretical)
error2.loc[i] = [rmse2, mse2, mae2, r2T, variance2]
print(error2)
```

Code Snippet 17: Evaluation metrics - y_test vs y_theoretical

Table 26 shows the scores for Error metrics when y_test and y_theoretical are compared:

Table 26: Actual values vs Theoretical values performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
21	0,2339	0,0547	0,1852	0,4206	0,7723
31	0,4336	0,1880	0,3258	-2,4746	-0,5134
41	0,1888	0,0356	0,1297	0,7192	0,7814
51	0,2658	0,0707	0,2001	0,3089	0,3095
61	0,3239	0,1049	0,2473	-0,3240	0,2448
71	0,3195	0,1021	0,2752	-2,5892	-1,9983
Mean	**0,2943**	**0,0927**	**0,2272**	**-0,6565**	**-0,0673**
Std Dev	**0,0780**	**0,0492**	**0,0638**	**1,3623**	**0,9654**

The prediction performance using the K-Nearest Neighbours algorithm are better than the ones made with the theoretical equation. Using Table 16 the performances of the two models can be compared. However, Experiment 1 with the theoretical equation produced even a reasonable model.

The following code shows the construction of the graph where Actual values (y_test), Predicted values using KNN algorithm (y_pred) and Theoretical values (y_theoretical) are plotted to give the reader a visual interpretation.

```
plt.figure()
plt.plot(np.arange(len(y_test)), y_test, 'r', label='y_test')
plt.plot(np.arange(len(y_pred)), y_pred, 'b', label='y_pred')
plt.plot(np.arange(len(y_theoretica)),y_theoretical,'g',label='y_theoretical')
plt.scatter(np.arange(len(y_test)), y_test, color='red', s=10)
plt.scatter(np.arange(len(y_pred)), y_pred, color='blue', s=10)
plt.scatter(np.arange(len(y_theoretical)), y_theoretical, color='green', s=10)
```

```
plt.legend(loc = 'upper right')
plt.title("Tool ID Scenario {} - KNN: y_test vs y_pred vs y_theoretical".forma
t(i+1))
plt.xlabel("Sample Index")
plt.ylabel("Ra")
plt.show()
```

Code Snippet 18: Plots generation

The following pictures show the plots generated from the above code:

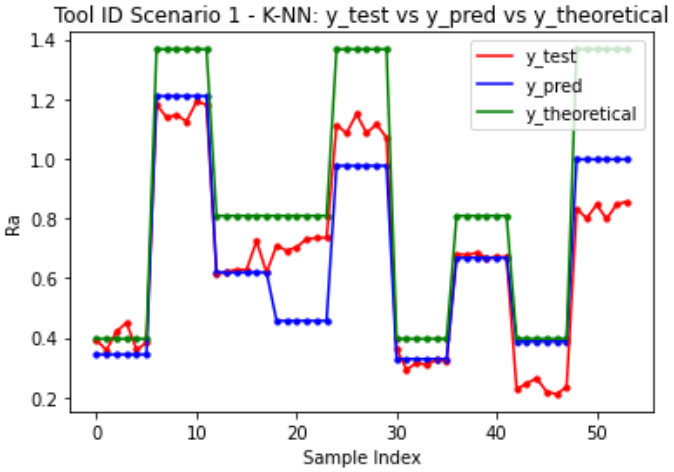


Figure 46: K fold 1 - y_test vs y_pred vs y_theoretical

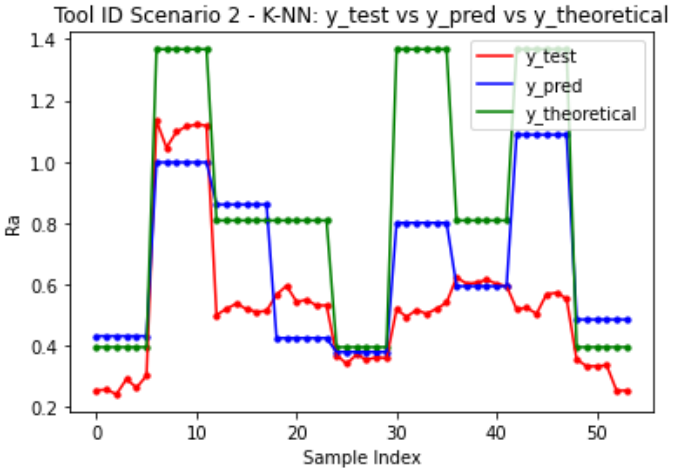


Figure 47: K fold 2 - y_test vs y_pred vs y_theoretical

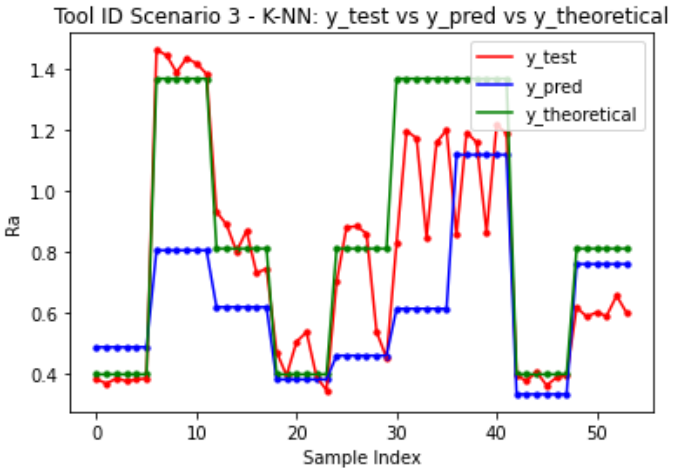


Figure 48: K fold 3 - y_test vs y_pred vs y_theoretical

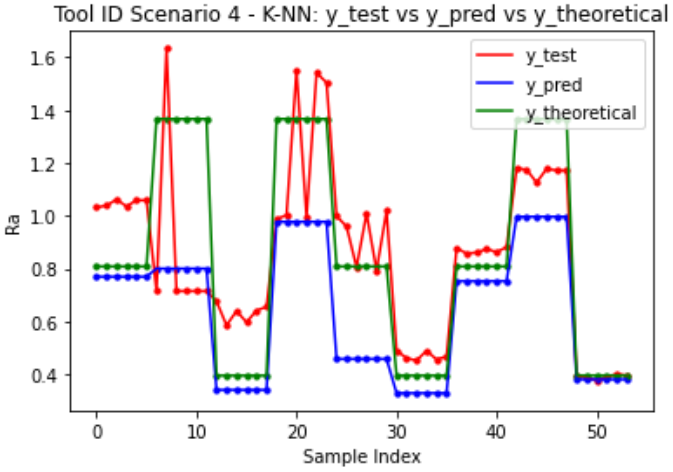


Figure 49: K fold 4 - y_test vs y_pred vs y_theoretical

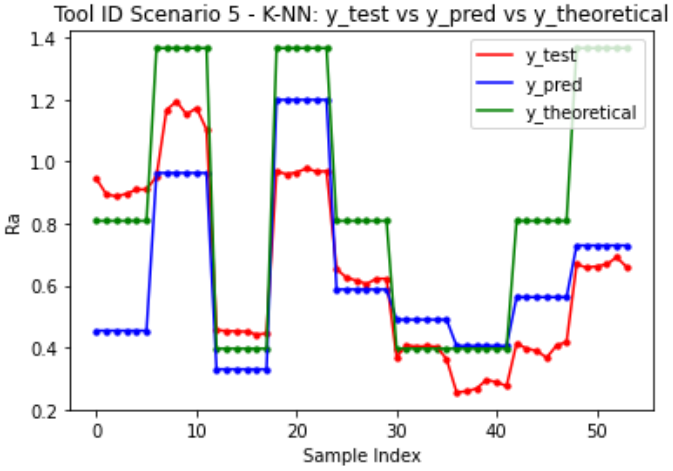


Figure 50: K fold 5 - y_test vs y_pred vs y_theoretical

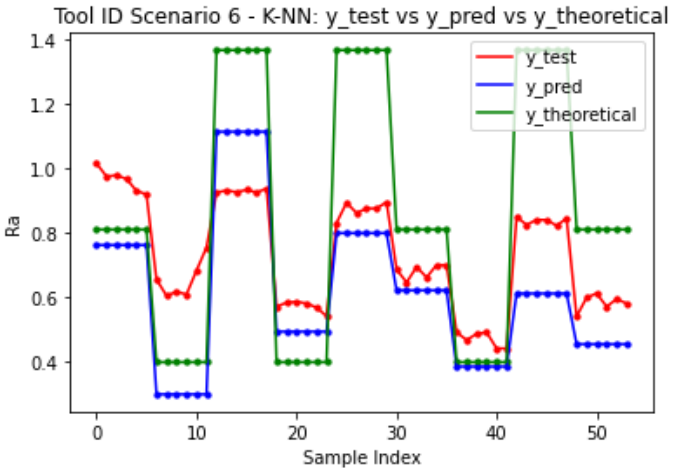


Figure 51: K fold 6 - y_test vs y_pred vs y_theoretical

8.2 Experiment 2: Random Forest Regression vs Theoretical Equation

Experiment 2 follows the logic implemented in Experiment 1 and the code implemented for this case is not reported to avoid repetition.

Table 27 shows the scores for Error metrics when y_test2 and y_theoretical2 are compared:

Table 27: Actual values vs Theoretical values performance

Fold Number = Tool ID	RMSE	MSE	MAE	R^2	EXPLAINED VARIANCE
43	0,3938	0,1550	0,3639	-0,6928	0,7528
53	0,3015	0,0909	0,2065	-0,1859	-0,1010
33	0,4858	0,2360	0,4048	-39,6598	-17,1525
23	0,3230	0,1043	0,2786	-3,1985	-1,8073
81	0,3084	0,0951	0,2612	-2,3190	-0,5455
13	0,3034	0,0921	0,2508	-7,6321	-6,1485
Mean	**0,3527**	**0,1289**	**0,2943**	**-8,9480**	**-4,1670**
Std Dev	**0,0674**	**0,0528**	**0,0683**	**13,9456**	**6,2178**

However, for Experiment 2, the predictions made with the theoretical equation did not follow the same behavior as Experiment 1, producing mean errors of much larger magnitudes.

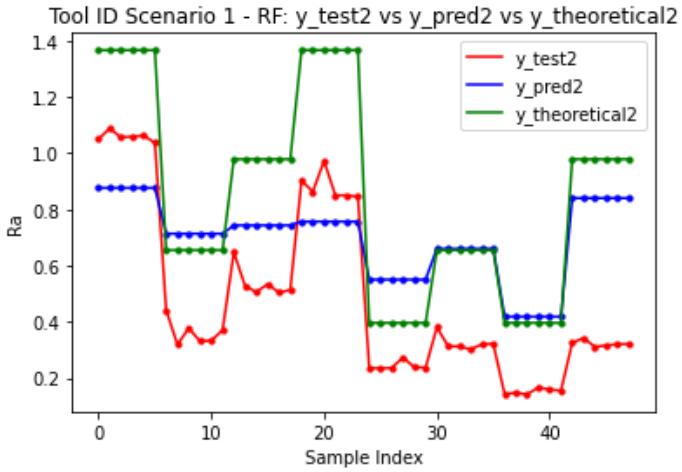


Figure 52: K fold 1 - y_test2 vs y_pred2 vs y_theoretical2

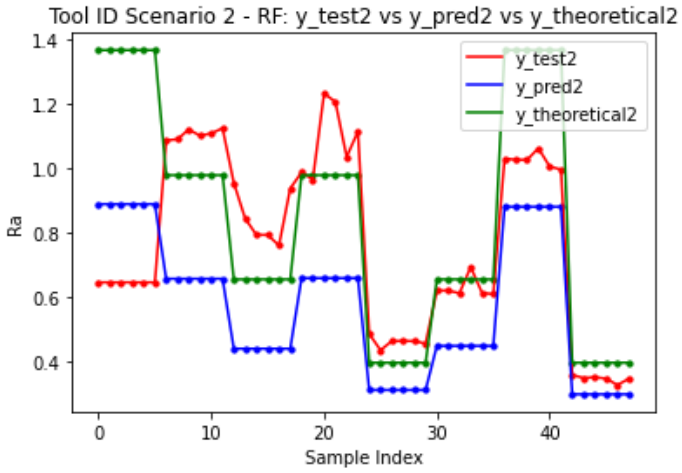


Figure 53: K fold 2 - y_test2 vs y_pred2 vs y_theoretical2

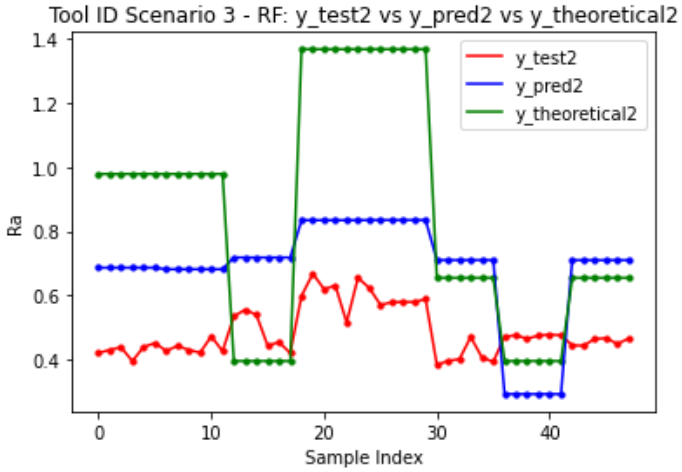


Figure 54: K fold 3 - y_test2 vs y_pred2 vs y_theoretical2

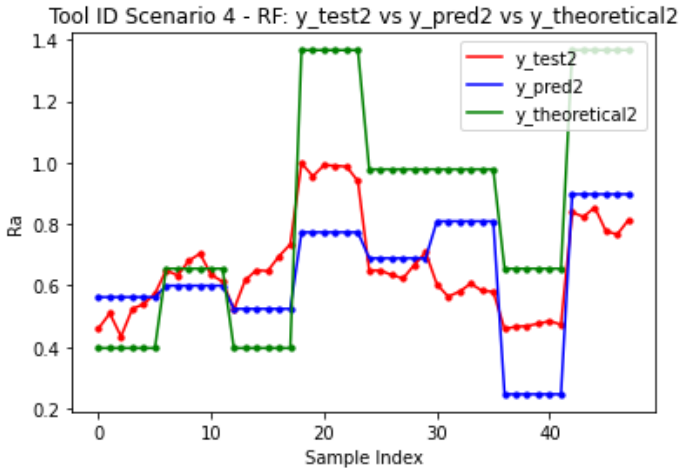


Figure 55: K fold 4 - y_test2 vs y_pred2 vs y_theoretical2

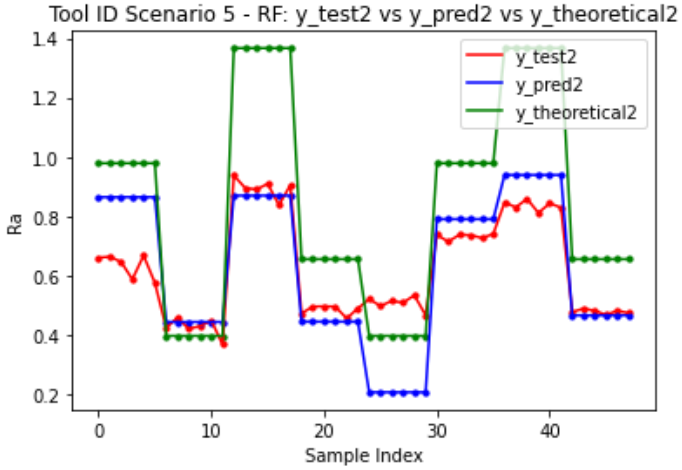


Figure 56: K fold 5 - y_test2 vs y_pred2 vs y_theoretical2

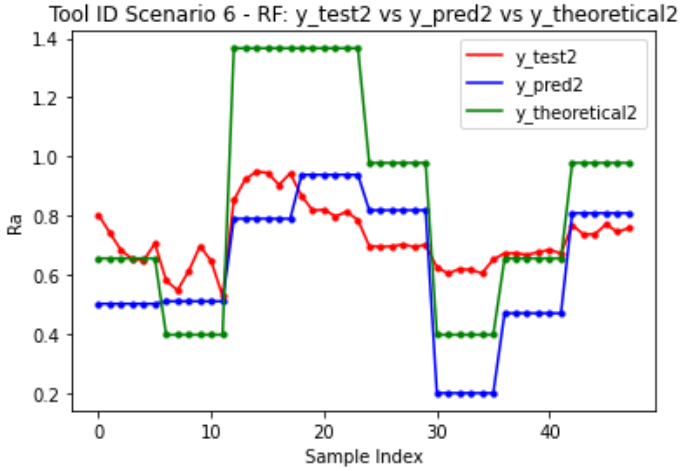


Figure 57: K fold 6 - y_test2 vs y_pred2 vs y_theoretical2

These performances generated by the equation in Experiment 2 must be discussed in the

light of tool wear presence. Experiment 2 reproduced the actual production situation where tool wear occurs, and such disturbance has made the theoretical equation much worse in predicting surface roughness values.

Chapter 9 – Conclusions and Future Work

Synopsis

The final chapter provides a summary of the study, emphasizing the key steps taken. Additionally, it presents suggestions for future work that could build upon this study as a foundation.

9.1 Conclusions

The objective of this study was to analyze a dataset provided by the Competence Center in Manufacturing (CCM) infrastructure, located at the Aeronautics Institute of Technology (ITA) in Brazil. The provided dataset deals with turning operations and the goal of this work is to predict Surface Roughness (Ra) using Machine Learning models. Two machining experiments were conducted to obtain data to train the ML models. The accuracy of the models was further improved by using k-fold cross validation to evaluate the accuracy metrics obtained during different folds.

Based on the performance of the ML algorithms, the one that performed the best for both experiments was chosen to determine the Surface Roughness of the steel bar. Finally, the predicted results generated from the ML models were compared with a theoretical equation. Subsequently, the predicted results generated by the ML models were compared with a theoretical equation. The soft computing method used to model the surface roughness parameter Ra exhibited excellent predictive capabilities, with evaluation metrics that outperformed the classical theoretical equation. Although the results were limited and basic due to the lack of sufficient data, the framework utilized to estimate surface roughness represented a crucial step towards geometry detection studies.

The major drawback of this study was the limited amount of data used to train the model, which affected its overall success. This constraint could be overcome by gathering additional data from multiple devices, varying different parameters, and incorporating more types of sensors and data sources. There is significant room for improvement in this study One could delve deeper into the nuances of the study for perfection and rectification in order to obtain a cleaner and more satisfactory result.

9.2 Suggestions for Future Works

While the machine learning framework presented in this work provides a comprehensive approach for predicting surface roughness, there are several areas in which improvements

can be made to further enhance prediction accuracy and provide better guidance to operators regarding steel bar geometry defect analysis.
Some of the key areas where further development could be beneficial include:

- Additional areas for exploration include using the dataset generated in this study to further explore ML modelling:

    1. Modeling more complex surface roughness parameters such as Skewness and Kurtosis, which were significantly affected by cutting tool wear.
    2. Exploring the use of cutting tool wear as an input feature
    3. Utilizing available data to map different responses, such as machining forces or cutting tool wear.

- Exploration of different features, responses, or manufacturing processes generating new experimental data:

    1. Collecting different features. Accelerometers, for example, tend to be more affordable and easier to install in production systems, and software to monitor machine signals is already commercially available, making acceleration, vibration, and electric current attractive features to use.
    2. Modeling different responses, such as the cutting tool wear, by collecting data from more complete tool-life tests;
    3. Expand the methodology to other:
        3.1 Manufacturing processes, such as milling or grinding;
        3.2 Machines to explore their difference in terms of rigidity;
        3.3 Materials, especially regarding their hardness.

Bibliography

[1] IBM - Deutschland | IBM [Internet]. What is Industry 4.0 and how does it work? | IBM; [Online]. Available: https://www.ibm.com/topics/industry-4-0

[2] Campbell C. Wikipedia: The Free Encyclopedia. http://en.wikipedia.org/wiki, s.vv. Theatre, Performance, and Performance Studies. TDR The Drama Rev [Internet]. Novembre 2009 [consultato il 21 marzo 2023];53(4):185-7. [Online]. Available: https://doi.org/10.1162/dram.2009.53.4.185

[3] i-SCOOP [Internet]. Industry 4.0 and the fourth industrial revolution explained [Online]. Available: https://www.i-scoop.eu/industry-4-0/.

[4] DAVIES R. Policy Commons . Industry 4.0: Digitalisation for productivity and growth; 22 settembre 2015. [Online]. Available: https://policycommons.net/artifacts/1335939/industry-40/1942749/.

[5] Filin SA, Yakushev AZ. https://www.fin-izdat.com/journal/digest/detail.php?ID=75034. Dig Finance [Internet]. 30 settembre 2019 ;24(3):256-65 [Online]. Available: https://doi.org/10.24891/df.24.3.256

[6] Zhou K, Taigang Liu, Lifeng Zhou. Industry 4.0: Towards future industrial opportunities and challenges. In: 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD) [Internet]; 15-17 agosto 2015; Zhangjiajie, China. IEEE; 2015 [Online].
Available: https://doi.org/10.1109/fskd.2015.7382284

[7] Pure - Login [Internet]. - University of Leoben research gateway[Online].
Available: https://pure.unileoben.ac.at/portal/en/publications/digitalization-and-digital-transformation-in-metal-forming-key-technologies-challenges-and-current-developments-of-industry-40-applications(e0144032-87a1-4d1b-901b-28647aae7ec1)/export.html

[8] Ylijoki O, Porras J. Perspectives to Definition of Big Data: A Mapping Study and Discussion. J Innov Manag [Internet]. 4 maggio 2016 ; 4(1):69-91. [Online]. Available: https://doi.org/10.24840/2183-0606_004.001_0006

[9] Qi Q, Tao F. A Smart Manufacturing Service System Based on Edge Computing, Fog Computing, and Cloud Computing. IEEE Access. 2019;7:86769-77 [Online].
Available: https://doi.org/10.1109/access.2019.2923610

[10] "www.tutorialspoint.com" [Online]. Available:
https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_overview.htm.

[11] Khedri A, Kalantari N, Vadiati M. Corrigendum: Water Supply 20 (3), 909–921: Comparison study of artificial intelligence method for short term groundwater level prediction in the northeast Gachsaran unconfined aquifer, https://iwaponline.com/ws/article/20/3/909/72227/Comparison-study-of-artificial-intelligence-method?searchresult = 1. Water Supply . 26 giugno 2020;20(6):2440. [Online].
Available: https://doi.org/10.2166/ws.2020.133

[12] EZTUIR Home [Online]. Available: http://eztuir.ztu.edu.ua/jspui/bitstream/123456789/6479/1/142.pdf

[13] Oladipupo T. New Advances in Machine Learning: InTech; 2010. Types of Machine Learning Algorithms; [Online]. Available: https://doi.org/10.5772/9385

[14] Zhang WJ, Yang G, Lin Y, Ji C, Gupta MM. On Definition of Deep Learning. In: 2018 World Automation Congress (WAC) [Internet]; 3-6 giugno 2018; Stevenson, WA. IEEE; 2018 [Online]. Available: https://doi.org/10.23919/wac.2018.8430387

[15] ToolSense . The 6 Types of Maintenance Explained | ToolSense Glossary [Online]. Available: https://toolsense.io/maintenance/the-6-types-of-maintenance-definitions-benefits-examples/#:~:text=What%20Are%20the%20Different%20Types,Predictive%20Maintenance%20and%20Reactive%20Maintenance

[16] Trent, E.; Wright, P. Metal Cutting. 4th Ed. Woburn, Ma: Butterworth-Heinemann, 2000. 464 P.

[17] Swift, K. G.; Booker, J. D. Manufacturing process selection handbook. Oxford, Ma: Elsevier, 2013.

[18] Beddoes, J.; Bibby, M. J. Principles Of Metal Manufacturing Processes. Carleton: Elsevier, 1999.

[19] Davim, J. P. (Ed.). Traditional Machining Processes. Berlin: Springer-Verlag, 2015.

[20] Machado, A. R. *Et Al*. Teoria Da Usinagem Dos Materiais. São Paulo: Blucher, 2009. 371 P.

[21] Klocke, F. Manufacturing Processes 1. Berlin, Heidelberg: Springer, 2011. 524 P.

[22] Stemmer, C. E. Ferramentas De Corte I. 3rd Ed. Florianópolis, Sc: Editora Da Ufsc, 1993. 249 P.

[23] Altintas, Y. Manufacturing Automation. 2nd Ed. Cambridge, Uk: Cambridge University Press, 2012. (Robotics And Automation Handbook).

[24] Smith, G. T. Cutting Tool Technology: Industrial Handbook. London: Springer, 2008. 605 P.

[25] Tschätsch, H. Applied Machining Technology. Berlin: Springer, 2009. 375 P.

[26] Brinksmeier, E. *Et Al*. Process Signatures: A New Approach To Solve The Inverse Surface Integrity Problem In Machining Processes. Procedia Cirp, V. 13, P. 429–434, 2014.

[27] Boothroyd, G.; Knight, W. A. Fundamentals Of Machining And Machine Tools. 3rd Ed. Boca Raton, Fl: Taylor & Francis, 2006.

[28] Davim, J. P. (Ed.). Surface Integrity In Machining. London: Springer, 2010.

[29] Gadelmawla, E. S. *Et Al*. Roughness Parameters. Journal Of Materials Processing Technology, V. 123, P. 133–145, 2002.

[30] Mundim, R. B.; Lucas, E. O. Effect Of Tool Wear On Surface Topography In Finish Turning Of Abnt 1045 Steel. *In*: Brazilian Congress Of Mechanical Engineering, 21, 2011, Natal, Rn. Proceedings [...]. Natal, Rn: Abcm, 2011.

[31] Benardos, P. G.; Vosniakos, G. C. Predicting Surface Roughness In Machining: A Review. International Journal Of Machine Tools And Manufacture, V. 43, N. 8, P. 833–844, 2003.

[32] He, C. L.; Zong, W. J.; Zhang, J. J. Influencing Factors And Theoretical Modeling Methods Of Surface Roughness In Turning Process: State-Of-The-Art. International Journal Of Machine Tools And Manufacture, V. 129, P. 15–26, Feb. 2018.

[33] Alpaydin, E. Introduction To Machine Learning. 3rd Ed. Massachusetts, Ma: Mit Press, 2014. 640 P. (Adaptative Computation And Machine Learning).

[34] Çaydaş, U.; Ekici, S. Support Vector Machines Models For Surface Roughness Prediction In Cnc Turning Of Aisi 304 Austenitic Stainless Steel. Journal Of Intelligent Manufacturing, V. 23, N. 3, P. 639–650, June 2012.

[35] Mia, M. *Et al*. Effect of time-controlled MQL pulsing on surface roughness in hard turning by statistical analysis and artificial neural network. The International Journal of Advanced Manufacturing Technology, v. 91, n. 9–12, p. 3211–3223, Aug. 2017.

[36] Mia, M.; Dhar, N. R. Prediction Of Surface Roughness In Hard Turning Under High Pressure Coolant Using Artificial Neural Network. Measurement, V. 92, P. 464–474, Oct. 2016.

[37] Garg, A.; Tai, K. Stepwise Approach For The Evolution Of Generalized Genetic Programming Model In Prediction Of Surface Finish Of The Turning Process. Advances In Engineering Software, V. 78, P. 16–27, Dec. 2014.

[38] Jurkovic, Z. *Et Al*. A Comparison Of Machine Learning Methods For Cutting Parameters Prediction In High Speed Turning Process. Journal Of Intelligent Manufacturing, V. 29, N. 8, P. 1683–1693, Dec. 2018.

[39] Laouissi, A. *et al*. Investigation, modeling, and optimization of cutting parameters in turning of gray cast iron using coated and uncoated silicon nitride ceramic tools. Based on ANN, RSM, and GA optimization. The International Journal of Advanced Manufacturing Technology, v. 101, n. 1–4, p. 523–548, Mar. 2019.

[40] Meddour, I. *Et Al*. Prediction Of Surface Roughness And Cutting Forces Using Rsm, Ann, And Nsga-Ii In Finish Turning Of Aisi 4140 Hardened Steel With Mixed Ceramic Tool. The International Journal Of Advanced Manufacturing Technology, V. 97, N. 5–8, P. 1931– 1949, July 2018.

[41] Asiltürk, I. Predicting Surface Roughness Of Hardened Aisi 1040 Based On Cutting Parameters Using Neural Networks And Multiple Regression. The International Journal Of

[42] Kara, F. *Et Al*. Effect Of Machinability, Microstructure And Hardness Of Deep Cryogenic Treatment In Hard Turning Of Aisi D2 Steel With Ceramic Cutting. Journal Of Materials Research And Technology, V. 9, N. 1, P. 969–983, Jan. 2020.

[43] Beatrice, B. A. *Et Al*. Surface Roughness Prediction Using Artificial Neural Network In Hard Turning Of Aisi H13 Steel With Minimal Cutting Fluid Application. Procedia Engineering, V. 97, P. 205–211, 2014.

[44] Chen, Y. *Et Al*. A Nested-Ann Prediction Model For Surface Roughness Considering The Effects Of Cutting Forces And Tool Vibrations. Measurement, V. 98, P. 25–34, Feb. 2017.

[45] Senthilkumaar, J. S.; Selvarani, P.; Arunachalam, R. M. Intelligent

Optimization And Selection Of Machining Parameters In Finish Turning And Facing Of Inconel 718. The International Journal Of Advanced Manufacturing Technology, V. 58, N. 9–12, P. 885–894, Feb. 2012.

[46] Pontes, F. J. *Et Al*. Optimization Of Radial Basis Function Neural Network Employed For Prediction Of Surface Roughness In Hard Turning Process Using Taguchi's Orthogonal Arrays. Expert Systems With Applications, V. 39, N. 9, P. 7776–7787, July 2012.

[47] Abbas, A. *Et Al*. Ann Surface Roughness Optimization Of Az61 Magnesium Alloy Finish Turning: Minimum Machining Times At Prime Machining Costs. Materials, V. 11, N. 5, P. 808, May 2018.

[48] Jafarian, F.; Taghipour, M.; Amirabadi, H. Application Of Artificial Neural Network And Optimization Algorithms For Optimizing Surface Roughness, Tool Life And Cutting Forces In Turning Operation. Journal Of Mechanical Science And Technology, V. 27, N. 5, P. 1469–1477, May 2013.

[49] Saric, T.; Simunovic, G.; Simunovic, K. Use Of Neural Networks In Prediction And Simulation Of Steel Surface Roughness. International Journal Of Simulation Modelling, V. 12, N. 4, P. 225–236, Dec. 2013.

[50] Donoho, D. 50 Years Of Data Science. Journal Of Computational And Graphical Statistics, V. 26, N. 4, P. 745–766, Oct. 2017.

[51] Shi, D.; Gindy, N. N. Tool Wear Predictive Model Based On Least Squares Support Vector Machines. Mechanical Systems And Signal Processing, V. 21, N. 4, P. 1799–1814, May 2007.

[52] Montgomery, D. C. Design And Analysis Of Experiments. 8th Ed. New York: John Wiley & Sons, 2013.

[53] Villares Metals. Vh13iso. São Paulo: Villares Metals, 2006. 2 p.

[54] Wikipedia, "Mean squared error - Wikipedia." [Online]. Available:

https://en.wikipedia.org/wiki/Mean_squared_error#In_regression
[55] “Model Evaluation Metrics.” [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error.
[56] S. Glen, “RMSE: Root Mean Square Error - Statistics How To,” *StatisticsHowTo.com:Elementary Statistics for the rest of us!* 2017, [Online]. Available:https://www.statisticshowto.com/probability-and-statistics/regression- analysis/rmse-root-mean-square-error/.
[57] O.Reilly, “Explained variance - Machine Learning Algorithms - Second Edition [Book].” [Online]. Available: https://www.oreilly.com/library/view/machine-learning- algorithms/9781789347999/49c4b96f-a567-4513-8e91-9f41b0b4dab0.xhtml.
[58] S. Statistics, “What is Linear Regression?,” *2013*. p. 1, 2013, [Online]. Available: [
[59] Levene M. ROKACH LIOR AND MAIMON ODED * Data Mining with Decision Trees: Theory and Applications. * World Scientific (2008). ISBN-13: 978-981-277-171-1. 244 pp. Hardcover. Comput J [Internet]. 2 dicembre 2008;53(4):489.
[60] "saedsayad.com," [Online]. Available: https://www.saedsayad.com/decision_tree_reg.htm.
[61] A. Chakure, "Towards Datascience," [Online]. Available: https://towardsdatascience.com/random-forest-and-its-implementation-71824ced454f.
[62] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Bayesian_linear_regression.
[63] "Scikit Learn," [Online]. Available: https://scikit- learn.org/stable/modules/linear_model.html#bayesian-ridge-regression.
[64] "www.saedsayad.com," [Online]. Available: https://www.saedsayad.com/k_nearest_neighbors_reg.htm.
[65] "scikit learn," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.kernel_ridge.KernelRidge.html.
[66] R. Gupta, "Medium.com," 26 06 2017. [Online]. Available: https://medium.com/@rajatgupta310198/getting-started-with-neural-network-for- regression-and-tensorflow-58ad3bd75223.
[67] "missinglink.ai," [Online]. Available: https://missinglink.ai/guides/neural-network- concepts/neural-networks-regression-part-1-overkill-opportunity/
[68] Mindfire Solutions, "Medium," 3 October 2017. [Online]. Available: https://medium.com/@mindfiresolutions.usa/python-7-important-reasons-why-you- should-use-python-5801a98a0d0bhttps://medium.com/@mindfiresolutions.usa/python-7-important- reasons-why-you-should-use-python-5801a98a0d0b.
[69] "Code Academy," [Online]. Available: https://www.codecademy.com/articles/what-is-an-ide.
[70] "Spyder," [Online]. Available: https://www.spyder-ide.org/.
[71] "Spyder," [Online]. Available: https://docs.spyder-ide.org/overview.html.
[72] ROWNOK ARA, "ubuntupit," [Online]. Available: https://www.ubuntupit.com/best-python-libraries-and-packages-for-beginners/.
[73] Wu S. A review on coarse warranty data and analysis. Reliab Eng Amp Syst Saf [Internet]. Giugno 2013 [consultato il 22 marzo 2023];114:1-11.
[74] A. Swalin, "How to Handle Missing Data," 2018. [Online]. Available: https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4.

[75] C. Agarwal, Outlier Analysis, Springer, 2017.
[76] K. Markham, "Github," 2019. [Online]. Available: https://github.com/justmarkham/scikit-learn-videos.
[77] J. Brownlee, "Machine Learning Mastery," 23 May 2018. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/.
[78] P. Gupta, "towards Data Science," 5 June 2017. [Online]. Available: https://towardsdatascience.com/cross-validation-in-machine-learning- 72924a69872f.
[79] "Deep AI," [Online]. Available: https://deepai.org/machine-learning-glossary-and- terms/feature-extraction.
[80] Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Incorporated; 2022.