POLITECNICO DI TORINO
DIMEAS – DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING

MASTER OF SCIENCE DEGREE IN AUTOMOTIVE ENGINEERING

MASTER OF SCIENCE DEGREE THESIS

APPLICATION OF MACHINE LEARNING ALGORITHMS FOR TOOL CONDITION MONITORING USING PYTHON

Supervisors:
Prof. Franco Lombardi
Prof. Giulia Bruno
Dott. Emiliano Traini

Candidate:
Sanjay Sankara Narayanan
S238730

Academic Year 2019 - 2020

ACKNOWLEDGMENTS

I would like to thank the supervisors, especially Prof. Giulia Bruno and Dott. Emiliano Traini for their immense support and patience during one of the most turning points in human history, the COVID-19 pandemic. The adoption to the new normal, primarily through video calls and virtual presentation albeit being cumbersome at first, was later attuned to, thanks to their seamless and incessant support.

Secondly, I am humbled and indebted to the supervisors for providing me with an opportunity to work on a project pertaining to the right juxtaposition of data science and theoretical course knowledge.

Finally, I would like to thank my family and friends for believing in me. This final milestone would not have been possible without them.

ABSTRACT

Enormous amounts of real-time data analyzed by using AI's analytical tools will improve decision-making and provide business users with improved visibility-whether it is minimizing asset downtime, enhancing manufacturing quality, automating production, forecasting demand, maximizing inventory levels or improving risk management. In this case study, we incorporate predictive maintenance for the milling process, to prevent the excessive expense and loss of time due to unexpected malfunctions of the cutting tool. With the help of machine learning algorithms and Python we interpret predictive maintenance (PdM) to ensure optimal product output.

This study provides a framework for applying machine learning to forecast tool wear, thus evaluating the tool's remaining useful life (RUL) for best output in terms of expense, efficiency and time. We will analyze the data set, clean, modify and remove various attributes before implementing it with different machine learning models to predict the remaining useful life and later comparing it with the wear performance of the actual tool.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 - INTRODUCTION

1.1 Industry 4.0

The role of AI instruments and techniques in smart manufacturing is a trending topic. The AI movement has passed its infancy, and several industries are proceeding with considerable activity. Today, more instruments on the factory floor, both large and small, are fitted with sensors that collect/share large amounts of data and record a multitude of actions.

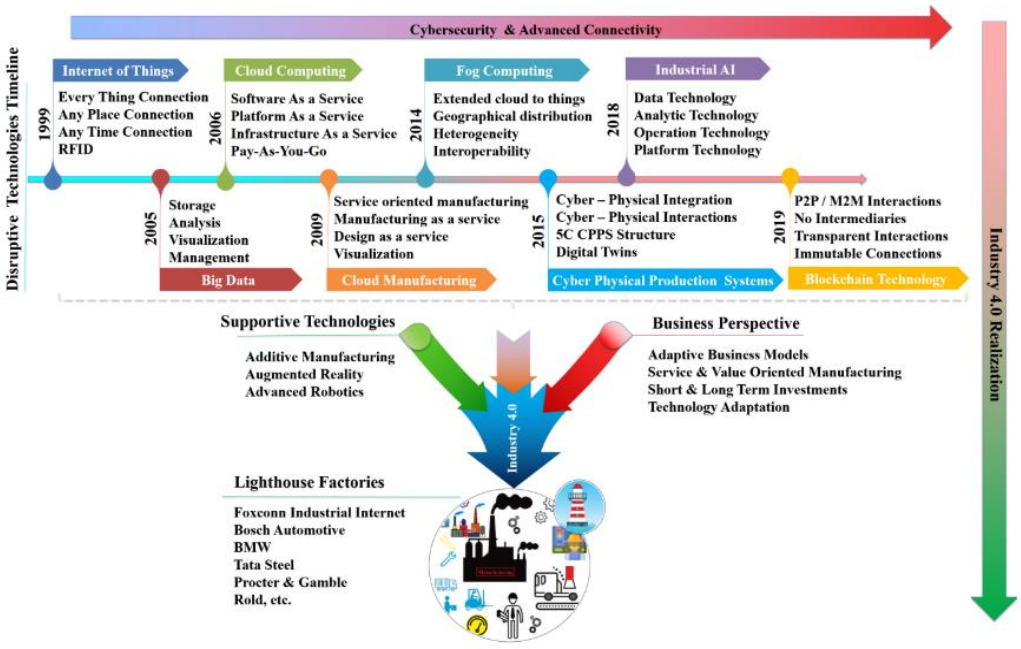


Figure 1: Evolution of disruptive technologies in Manufacturing

Manufacturing 4.0's real-world deployment began with increased efficiency, followed by improved flexibility, consistency, and speed. To shape a rapidly evolving on-demand production environment, a manufacturing stability can be accomplished by machine-to-machine and human-machine interactions. Furthermore, via real-time plant tracking and just-in-time repairs, quality improvement can be achieved. The degradation of processing facilities and instruments generally lowers the efficiency of the output and improves efficiency by increasing unplanned downtime. The smart prognostic and health management (PHM) tools have also become crucial for just-in-time maintenance, which ensures high-quality goods, minimizes unplanned downtime and improves customer loyalty [1].

1.2 Artificial Intelligence – Machine Learning

AI has become increasingly advanced and sophisticated, leading to reduced prices and hence making the platform more accessible across diverse sectors. This transition has had a massive effect on the production of goods which has contributed to more productivity, less human error, and more extraordinary total performance. To foresee when mechanical parts would need replacement, many businesses use artificial intelligence. Combined with historical evidence, machine learning produces an algorithm that detects possible problems when they emerge, helping organizational workers to take the steps required to eliminate problems that can delay or even interrupt development. One of the most valuable advantages of AI in real-time monitoring is that it gives a more accurate description of where any inefficiencies exist in the production chain and what causes the bottleneck. This ability to define the precise method that needs enhancement helps organizations to solve the problem promptly, results in saving time and expenses. However, predictive maintenance acts as an axis of growth for the implementation of the Industry 4.0 system. The goal is to obtain models that decrease diagnostic ambiguity [2].

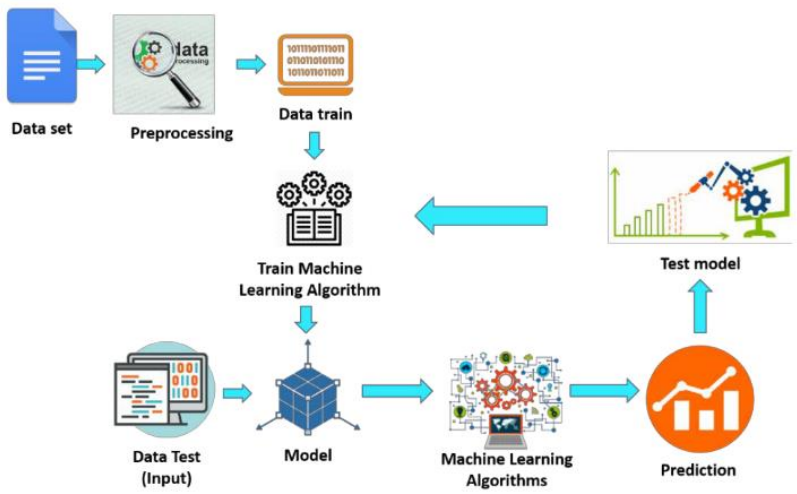


Figure 2: Machine Learning Stages

Machine learning is broadly classified into supervised (trains known inputs and outputs to predict future outputs) and unsupervised learning (finding patterns or structures in the input data) [3] .

1.2.1 Supervised Learning

It is a learning technique where the machine is taught and trained using data that's already labelled with correct answers. When the system is provided with a new set of data, it'll try to analyze and generate data based on the previously labelled data.
E.g., Classification, Regression

1.2.2 Unsupervised Learning

Here the data fed into the machine is neither labeled nor classified and the algorithm is made to find patterns and structures without guidance. Unlike supervised learning, the machine is not trained for prediction, rather it relies on itself for finding similarities and patterns [4].
E.g., Clustering

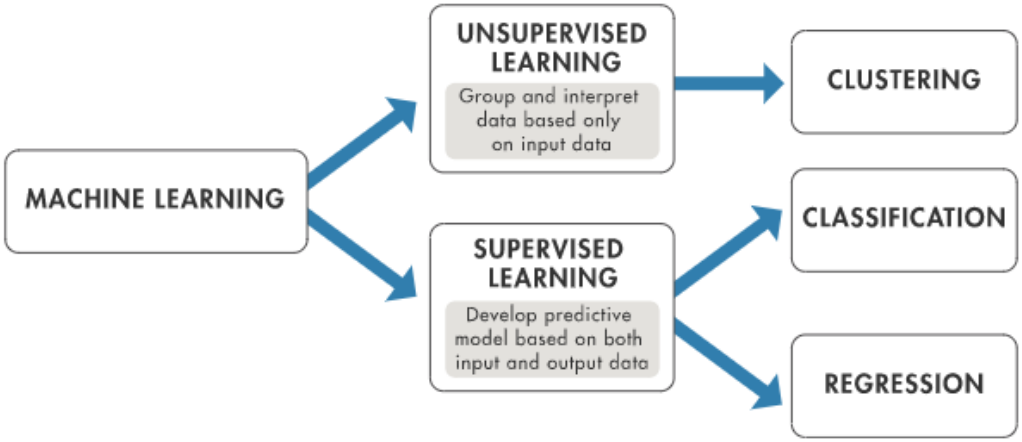


Figure 3 Machine Learning Techniques

Various languages can be used for machine learning such as Java, Python, R, JavaScript, Scala etc. but for the purpose of this thesis, we are restricting to Python.

1.3 Thesis objective and structure

The thesis describes the role and importance of Predictive Maintenance and Tool Condition Monitoring systems in the manufacturing industry. We use the data from NASA's Prognostic Center of Excellence (NASA – PCoE) to train machine learning models for the milling cutter. Python and its libraries are used to train the algorithms for the TCM model. It takes as input cutting parameters and sensors signals and returns the tool wear condition as output.

The thesis report is structured by shedding light on the introduction to tool maintenance and the literature review of predictive tool maintenance conducted and published in the recent years. We later transition to the milling process and the use of Python and its libraries for the analysis of this case study. The last part deals with data analysis, model training, testing and improvement using various machine learning algorithms. The thesis concludes with the interpretation of the results and drawing conclusion on using an improved model for predicting remaining useful life of a tool.

CHAPTER 2 – PREDICTIVE MAINTENANCE

2.1 Types of Maintenance

Maintenance can be categorized mainly into three:

2.1.1 Corrective Maintenance

This form of maintenance is focused on fixing the already existing faults, i.e. it only takes place where there is a crucial halt in the operation or machine [5]. This is the easiest maintenance method, since the manufacturing pause and the repair of the parts to be replaced are also required, bringing a direct expense to the operation. As a consequence, corrective repair is utilized in systems in which defects do not have a critical output effect.

2.1.2 Preventive Maintenance

Preventive maintenance is a maintenance strategy conducted to predict process / equipment faults with a planned timeline or procedure iterations. It is normally an optimal solution to eliminate errors. The goal of this method of maintenance is to minimize the amount of remedial maintenance steps introduced through routine inspections and replacement of worn pieces. Which includes maintenance, careful monitoring and the creation of a schedule that must be executed by trained employees. In fact, if it is not properly implemented, a breakdown can arise, resulting in productivity costs [6].

2.1.3 Predictive Maintenance

Predictive Maintenance (PdM) is based on constant monitoring of the integrity of a system or a process, allowing maintenance to be carried out only as required. In comparison, computational software focused on historical evidence (e.g. machine

learning techniques), quality considerations, mathematical estimation methods and engineering approaches allow for the early identification of errors [7].

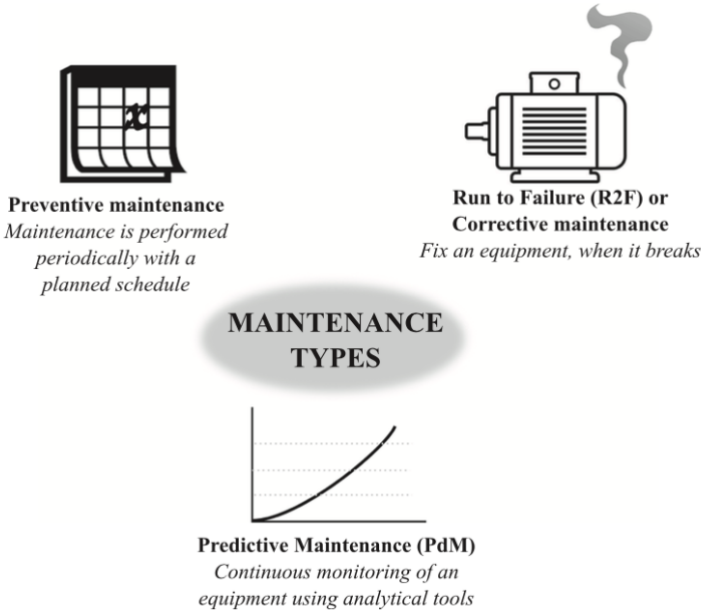


Figure 4. Types of Maintenance

2.2 Predictive Maintenance

Predictive Maintenance (PdM) is a form of maintenance that happens before failure occurs. Besides sensor readings, it is dependent on precise calculations, and maintenance is done by evaluating the calculated parameters. This maintenance is focused on maintaining the optimum time between repairs and minimizing the expense and volume of planned maintenance operations [8].

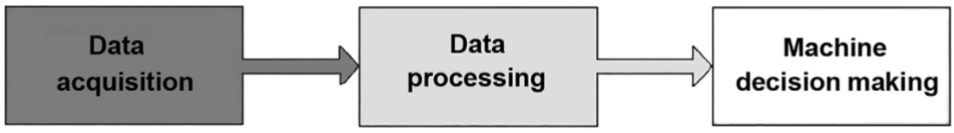


Figure 5. Steps needed to develop a pdM

2.2.1 Data Acquisition

The data collected using the pdM are of two types: event data and condition monitoring data. Event data includes the information on what happened to the asset and the type of maintenance applied, whereas condition monitoring data are related to the health measurements of the asset. There are wide variety of signals such as those from vibrations, acoustics, temperature, pressure etc. In order to accumulate these, various sensors are developed such as accelerometers, gyroscopes, etc. [8]

2.2.2 Data Processing

Obtained data can have missing, inconsistent or noise values. The stronger the data the better the data mining procedure. Usually preprocessing of the data is done to improve the quality of the data – it's usually done by preparing and transforming the dataset.

2.2.3 Machine Decision Making

This step involves both diagnostics and prognostics. Diagnostics includes detection, identification and isolation of the faults when they surface, whereas prognostics tries to predict the failures even before they occur. Often times this leads to a shift from an unsupervised problem to a supervised one as it's easier to develop one with great accuracy [8].

CHAPTER 3 – LITERATURE REVIEW

In the 21st century, Industry 4.0 along with the evolution of computing infrastructures: Artificial Intelligence, Big Data, Cloud Computing, IoT etc. has empowered the manufacturing status quo. With the help of ICT, it's possible to construct an integrative and collaborative system based on the Industry and customer demands [9].

To extract manufacturing data artificial intelligence is used. AI makes the system learn and adapt based on historical experience and training; This created system can even help in decision making. To improve the sustainability of Industry 4.0 it's important to develop a communicating tool between the machines and maintenance engineers pertaining the maintenance tasks. Doing so can help in reducing cost and downtime along with increased useful life of a tool.

A recent article even mentioned this unplanned downtime due to lack of maintenance strategy that accounted up to 20 percent loss in productivity and $50 Billion each year in cost [10].

Predictive maintenance—or PdM, for short—is a technique for foreseeing upkeep necessities in machines on a processing plant floor. By breaking down operational data from the machines, patterns develop that will permit administrators to anticipate when maintenance

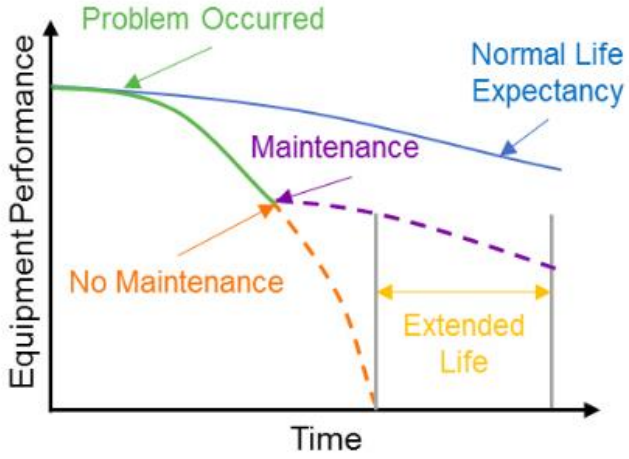


Figure 6: Extending the life of a tool with predictive maintenance (modified from Predictive Motor Maintenance. Fort Collins: 2016)

will be required on some random unit, taking into account the arrangement during less exorbitant occasions [11].

Before, makers would depend on reactive maintenance, also called the “if it ain’t broke, do not fix it” strategy. We can well envision that adjusting machines when they are broken is an enormous cost, both as unplanned downtime and the expected effect on different pieces of the machine.

Predictive maintenance depends on explicit data pulled from each machine, to recognize likely issues. A model would be a vibration investigation- A model that utilizes a standard of gathered execution information for a machine that will have the option to recognize changes. Deviations from the benchmark permit administrators to anticipate a requirement for maintenance before the issue gets genuine, bringing about hardware disappointment. Due to continuous improvement in data acquisition and the consequent growth in data accumulated, data driven methods have become important for tool conditioning to determine the state of an equipment [12]. Most of the works mentioned uses data from private experiments or were generated with the help of a simulator.

Various tool conditioning, feature extraction and machine learning techniques regarding milling process were discussed, [13] and suggested the use of single sensors (reduced system cost) as opposed to the notion of using multiple sensors for an Industrial environment. Prioritizing an on-line TCM system which utilizes indirect measurement rather than a direct measurement plays a significant role in determining the tool wear.

The capacity to copy human insight and see each part of a given problem through experience makes AI an ineluctable cycle in computerized reasoning. The custom of being compelled to a lot of calculations that were pre-introduced has hailed throughout the years with the creation of AI. Among a few AI methods looked, utilization of artificial neural networks (ANN) in process modeling and optimization has become very recognizable due to its capacity to anticipate the output rapidly and precisely. The viability and common sense of ANN models in assembling applications are looked into for showing its critical part in process modeling. Perceptions are accounted for in the investigation.

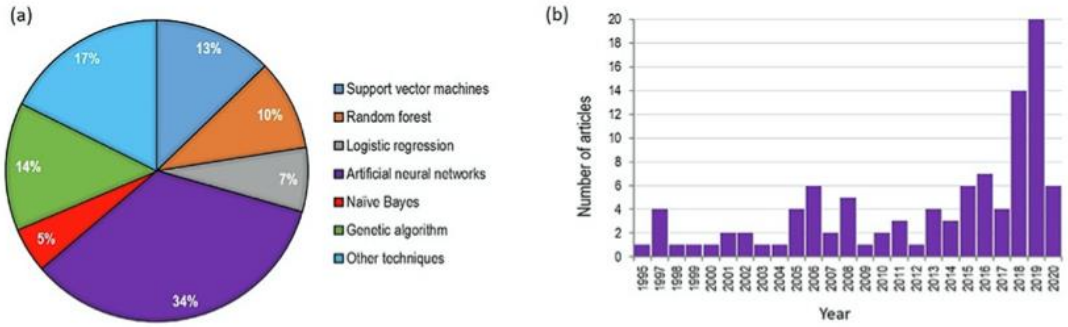


Fig. 2. Machine learning techniques contribution and year wise distribution of papers considered in the present review.

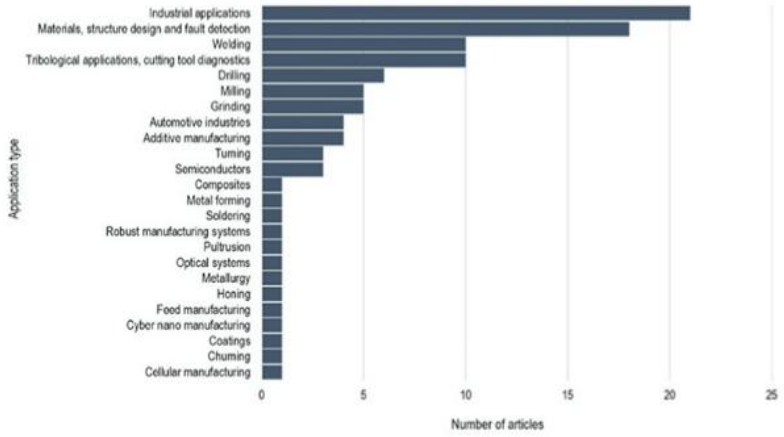


Figure 7 Application of ML techniques in different areas of manufacturing

The number of articles taken for showing the performance of every strategy is given in Fig. 2(a). From this, unmistakably the level of articles taken in this investigation for assessing the performance of ANN (34%) is high among all the methods. Commitment of articles for assessment of SVM, GA, and RF are 13%, 14%, and 10% individually, which demonstrates that equivalent significance were given to all the three procedures in portraying their performances [14].

A cloud-based approach was adopted for an event based IoT and ML architecture for predictive maintenance in Industry 4.0 [15]. They combined information with data about the failures, which made it helpful to train the data sets for predictive maintenance. Utilizing equipment logs to foresee issues represents a few issues that have not yet been completely investigated. Specifically, determining predictive features represent a significant test, as the

logs contain a gigantic measure of information that rarely incorporates data regarding the information for failure prediction [16].

In the study, they adjusted PdM application into a Big Data condition utilizing distributed computing and Big Data systems to give the strategy the capacity to scale and to handle the information of a huge number of associated machines. They fabricated a computational pipeline to manage high dimensional data and to show complex association occasion-based mistake totals parsed from unstructured log records. To accomplish this, they tried the theory that aggregated highlights registered by feature engineering on temporal data, that could be productively demonstrated with AI classifiers. The proposed application could assist with empowering PdM and main driver investigation of woodworking machines.

All in all, they introduced a log based PdM application by exploiting occasion-based triggers and the use of cutting-edge AI methods to construct predictive models. The fundamental bit of leeway of such a methodology is the utilization of aggregated event-based predictors (errors and cautioning occasions) as temporal attributes to foresee the possible machine down disappointments. The proposed application was sent a PdM application for carpentry by exploiting dispersed a Big Data condition to produce data driven predictive models that depend on historical log information.

On the other hand, during the primary stage of PdM, the signal highlights were separated from raw information, and afterward the SVR models with considering diverse length of signs at past occasions were built up to mirror the connection between monitoring data and tool life [17]. In the subsequent stage, the built models are utilized to foresee cutting device's RUL, and the best signal length for exact forecast result was obtained. Tool failure in the machine may prompt unscheduled down time, quality issues and even genuine mishaps. It is assessed that about 20% of the personal time is ascribed to instrument disappointment, which bring about critical monetary losses [18].

The data driven strategies used mostly included: Artificial neutral network, Markov models, and Support vector machine (SVM) et al. SVM is broadly utilized for RUL forecast [18]. The paper talks about two delicate computing methods, neuro fuzzy logic and SVR strategy for cutting RUL forecast.

However, in all actuality, the machine wellbeing state isn't just identified with the current observing information, yet additionally to the past occasions. Along these lines in the paper, a SVR based technique with considering distinctive length of signals at past occasions is proposed for RUL forecast. It comprised of two fundamental stages: an off-line and an on-line stage. In the primary stage, the signal highlights are separated from raw information, and afterward the SVR model with considering data at past occasions is set up. In the subsequent stage, the developed models are utilized to anticipate cutting instrument's RUL and the best signal length for exact forecast result was obtained. The proposed strategy is validated by the experimental information taken from a (CNC) rotor slot machine in a production line.

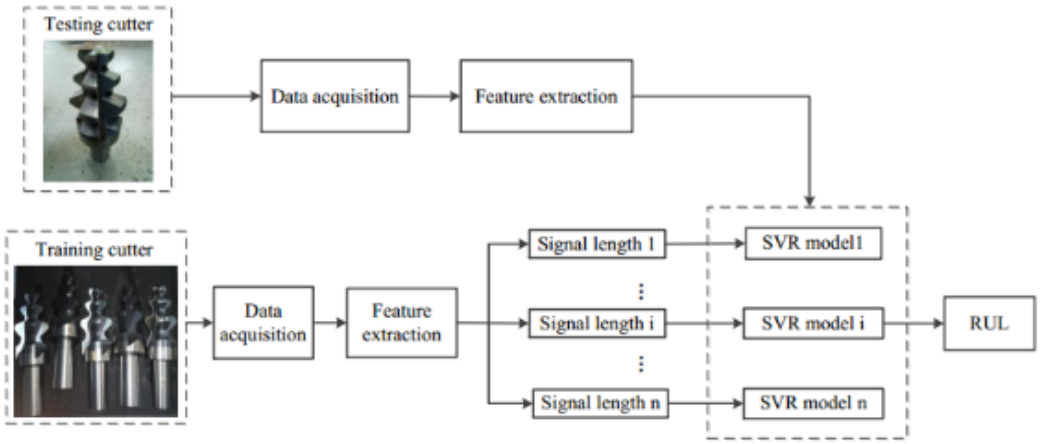


Figure 8 Framework for RUL Prediction

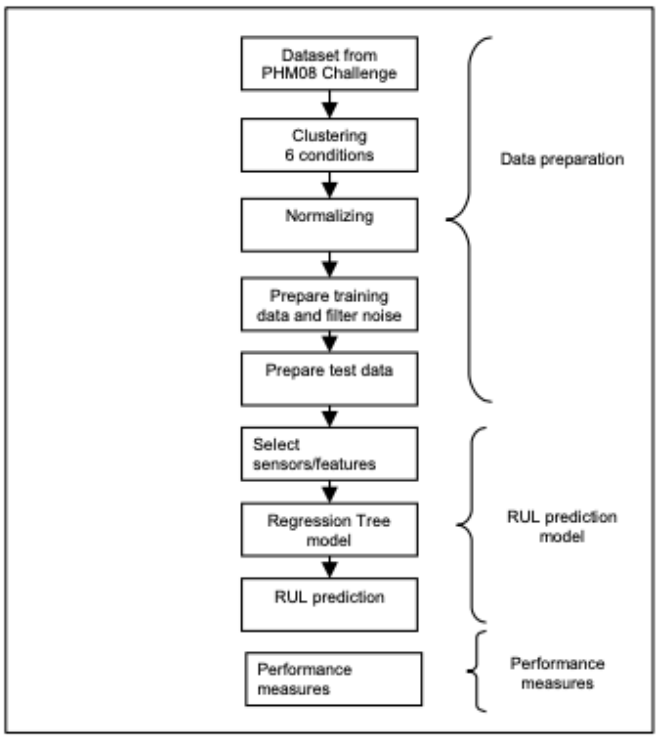
The framework for RUL prediction of cutting devices adopted is shown in Figure 3. The guideline of the proposed technique depends on two primary stages: an off-line stage and an on-line stage. In the principal stage, the signal highlights are extricated from raw monitoring data, and afterward the SVR models, various length of data at past occasions are set up to mirror the connection between monitoring signals and tool life. In the subsequent stage, the developed models are utilized to foresee cutting apparatus' RUL and doing so the best signal length for exact forecast result is obtained. The cutting parameters were the spindle speed of the slotting cutter - 200 rpm and the feed rate - 15mm/min.

AE is one of the best signals for tool wear monitoring, particularly for mechanical applications. It can maintain a strategic distance from the impedance of outer components on

the grounds that the recurrence of AE signals is a lot higher than that of machine vibration and ecological noises [19], [20]. Past studies [17], [21], [22] demonstrated a few AE highlights can show tool wear, in the work, 14 highlights were separated: Rise time(RT), Counts(C), Energy(E), Amplitude(A), Average frequency (AF), Root mean square (RMS), Average signal level (ASL), Counts To Peak (CTP), Reverberation Frequency (RF), Initiation Frequency (IF), Signal Strength (SS), Absolute Energy (AbE), Frequency Centroid (FC), and Peak frequency (PF).

In 2008, NASA Prognostic Center sorted out a prognostic test opened to people in general. The coordinator reproduced dataset to copy the airplane motor conduct from starting cycle until failure. Various scientists utilized this dataset to plan and propose RUL prediction models [23]. This paper proposed an improved RUL expectation model by utilizing regression tree (RT) method. RT has been utilized in other examination zones, for example, in clinical exploration, advertising research and considerably more particularly when the information has high autonomy factors. Regression tree (RT) is one of the classification strategies that can be utilized for prediction based on developing and pruning tree technique. This strategy was started via Automatic Interaction Detection (AIA) method and later it was improved by a group of scientists which set up the RT procedure [24]. Linear and multiple regressions can deal with a predetermined number of features in the dataset. If the number is small, regression strategy can deal with it well for forecast. In any case, when the quantity of features expands the model will get intricate to cooperation between features. To conquer this restriction, RT applies partitioning strategy where it lessens the number of highlights into a small fragment where it is more reasonable.

This technique has outperformed different strategies, for example, linear regression, ridge regression, support vector machine and neural network regarding precision rate. In the dataset, likewise, comprise sensor readings from three working conditions and 21 readings from sensors.



*F*igure 9 Steps used in the development of RT Model

Performance estimations are imperative to decide the viability of the proposed model. In the wake of gathering the expectation of RUL from the model, the prediction value is analyzed against the true value. The true value is the cycle value in testing data. The error performance was measured utilizing mean absolute error (MEA), mean absolute percentage error (MAPE) and mean squared error (MSE).

Method	MSE	MAE	MAPE
*DSR	6.21	26.30	0.14
*SVM	7.50	31.14	0.15
*RNN	6.80	29.01	0.14
*Comentropy based fusion	3.50	14.20	0.07
Regression tree (RT) (This study)	25.50	25.50	0.58

Figure 10 Performance of regression tree (RT) against existing RUL prediction model namely, Dempster-Shafer regression (DSR), Support vector machine (SVM), Recurrent neural network (RNN) and Comentrophy based fusion.

CHAPTER 4 – USE CASE: MILLING MACHINE

4.1 Milling Process

Being the most prevalent processes around for machining custom parts, Milling is a machining process used to remove materials by advancing a cutter into a workpiece. This can be done in various directions, axes, speed and pressure. During the cutting, due to sheer deformation, material is pushed off the work piece in tiny clumps to form chips. The speed and feed at which a milling process is performed is varied to suit a combination of variables [25]. The process initially consists of a roughing step in which the material is removed rapidly and economically as possible while leaving an appropriate layer of material for the next finishing point. The removal of the limited material allows for the tolerances of the dimensions and the degree of roughness of the surfaces that can be preserved in the initial process. The tool gradually starts wearing due to the heat and stress during the process which indirectly will compromise the performance of the cutting tool and the surface finish.

4.2 Types of Milling

Milling can be broadly classified into four main categories:

4.2.1 Face Milling

We achieve face milling when the surfaces are machined right angled to the axis of the cutter. They result in producing flat surfaces either through horizontal or vertical feeding.

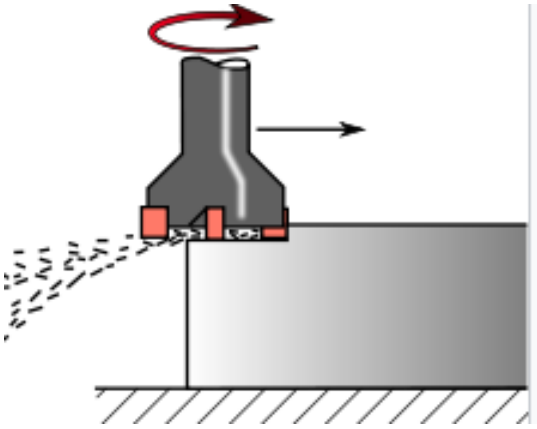


Figure 11. Face milling process

4.2.2 Plain or Slab Milling

It's the result of a milling process when surfaces are machined that are parallel to the axis of the cutter. Usually, the cutter is mounted on a standard machine arbor with the tool mounted parallel to the surface.

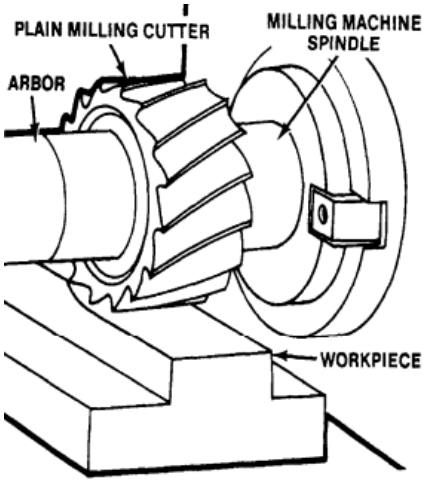


Figure 12. Plain milling process

4.2.3 Angular Milling

In angular milling surfaces are machined at an inclination neither parallel nor perpendicular to the axis of the cutter. Usually, a cutter with chamfers and grooves are used during the machining for attaining the end result.

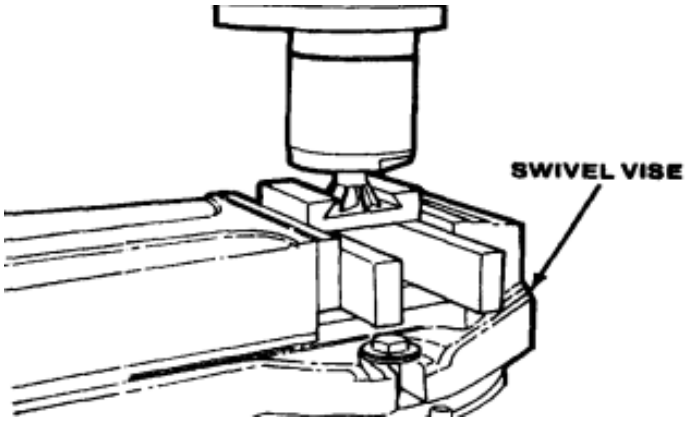


Figure 13. Angular milling process

4.2.4 Form Milling

When surfaces machined have irregular outline or contours consisting of curves or straight lines, this type of milling process is considered. They are usually machined using tools that are made out of the outline that has to be cut.

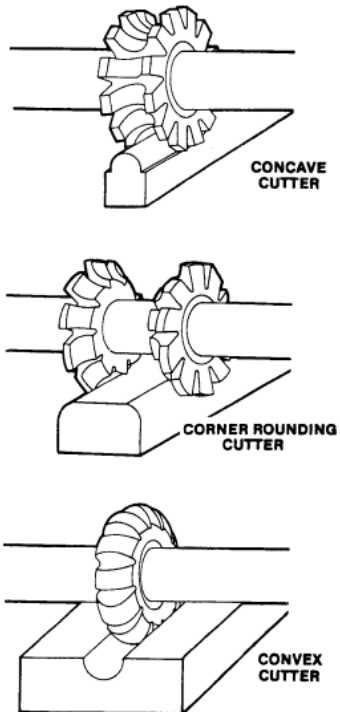


Figure 14. Form milling process

4.3 Tool Material

Milling cutting tools are selected based on the material of the workpiece and the final result and anticipated finish one's after. In order to produce effective and optimal outputs there are three important characteristics a tool material has to exhibit:

4.3.1 Hardness

It's the temperature resilience a cutting material possess while maintaining its strength during the operation. They should be harder than the workpiece undergoing the machining operation.

4.3.2 Toughness

A cutting tool should have enough toughness in order to undergo the machining process without failing or chipping.

4.3.3 Wear resistance

Wear resistance of a tool plays yet another vital role in deciding a machining tool. It's the characteristic property duration of a tool based on the type of material and sufficient tool life before its replaced or substituted.

The most common machining tool materials are those made of high speed steel (HSS), tool steel and cast alloys, carbides, ceramics, diamonds etc. [26]

4.4 Cutting tool and tool wear

At all times, the quality and the condition of the cutting tool is controlled and monitored. In machining operations, a PdM of cutting tools can not only warn whether a tool has to be replaced based on the wear length of the tool when it exceeds its wear limit but can also measure the tool's remaining usable life (RUL).

As shown in Fig., the conditions of the cutting tool can be described by the wear on the various face of the cutting tool. Although calculating the abrasive wear on the flank face of the cutting tool is a common procedure to define the state of the cutting tool, a flank wear limit is used as a rule of thumb to define the condition of the cutting tool during machine learning model evaluation and improvement [27].

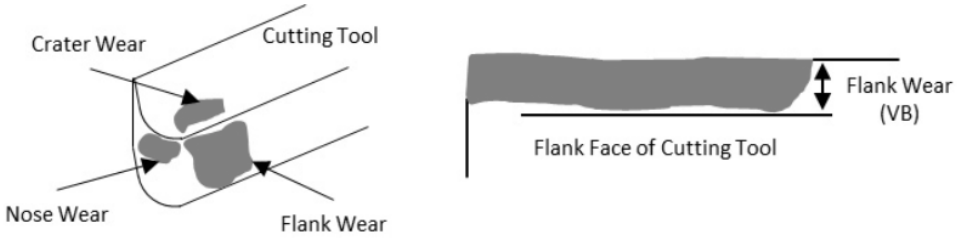


Figure 15. Types of cutting tool wear

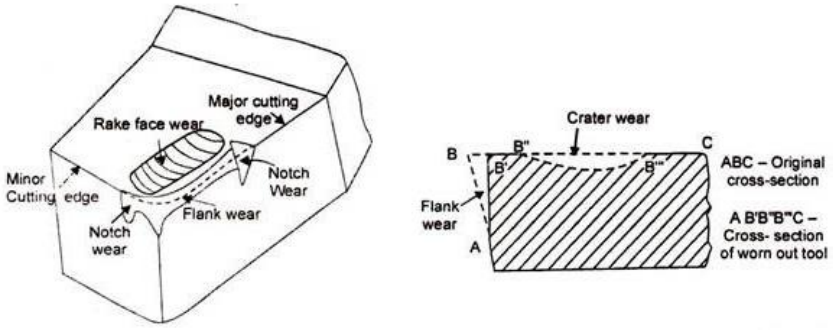


Figure 16. a. Tool wear phenomena, b. Flank and crater wear

The tool wear can be divided into two types in general due to their rubbing action and the affected regions:

4.3.1 Flank wear

It's one of the most common and important wear resulting from the abrasive/ adhesive wear of the cutting edge against a surface. Most often the wear is a result of high temperatures which affects the tool and the material alike. The average wear, V3 can be measured against the maximum size, VBMax. Tool life expectancy equation can be given by:

$V_C T^n = C$ considering the depth and feed of cut,

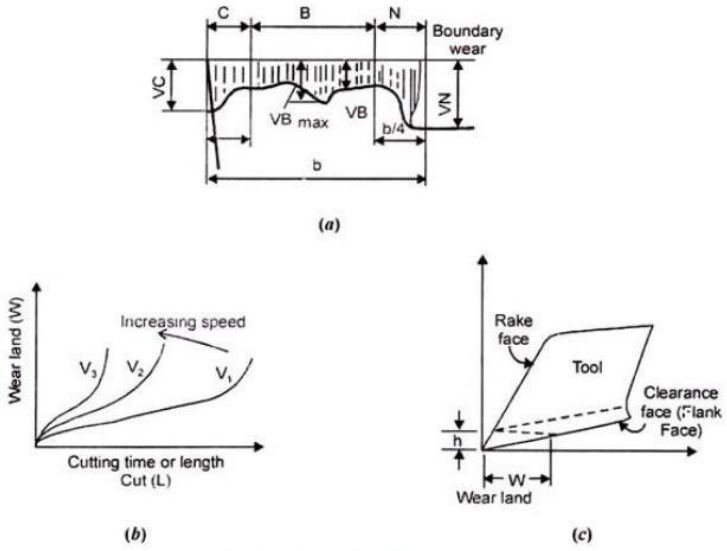


Figure 17. Flank wear: The wear on the flank face (relief or clearance face)

4.2.1 Crater wear

Wearing chips in the crater erodes the tool's rake face. For tool wear, it is relatively natural and does not significantly degrade the use of a tool until it gets bad enough to cause a breakdown of the cutting edge. Crater wear will improve the angle of the

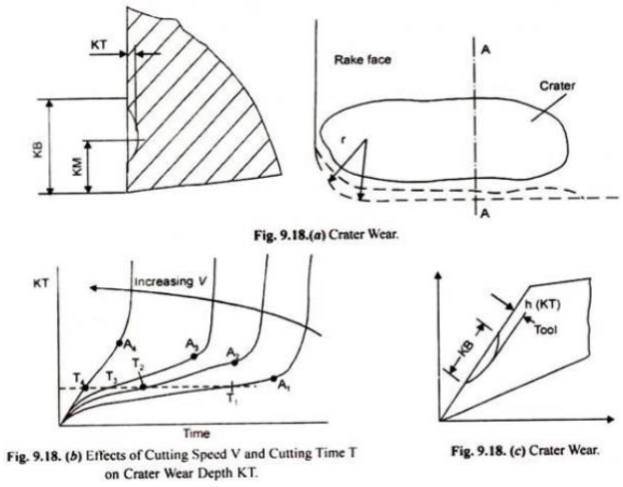


Figure 18. Crater wear: The crater resembling wear on the rake face of the tool.

working rake and decrease the cutting force, but the strength of the cutting edge may also be reduced. In ductile materials such as steel that manufacture long continuous chips, this is more common.

4.4 Experiment Description and Setup

The data in this set illustrates experiments from runs on a milling machine under varying operating conditions, such as various feeds, cutting depth and material for the workpiece using three types of sensors (acoustic emission sensor, vibration and current sensor). In particular, in a regular cut as well as entry cut and output cut, tool wear was investigated (Goebel, 1996), and the flank wear of the milling insert was registered. This dataset is a series of prognostic datasets donated by various institutions, organizations and corporations is provided by the Prognostic Center of Excellence (PCoE) at NASA's Ames Research Center. The dataset is organized in a 1x167 MATLAB struct with fields as shown below:

Table 1: Struct field names and description

Field name	**Description**
case	Case number (1-16)
run	Counter for experimental runs in each case
VB	Flank wear, measured after runs; Measurements for VB were not taken after each run
time	Duration of experiment (restarts for each case)
DOC	Depth of cut (does not vary for each case)
feed	Feed (does not vary for each case)
material	Material (does not vary for each case)
smcAC	AC spindle motor current
smcDC	DC spindle motor current
vib_table	Table vibration
vib_spindle	Spindle vibration
AE_table	Acoustic emission at table
AE_spindle	Acoustic emission at spindle

The experiment was conducted with 16 cases and multiple runs. The runs were dependent on the degree of the flank wear as it was based on the wear limit of the tool. When a certain measurement was not taken, flank wear for that field was left empty.

Fields	case	run	VB	time	DOC	feed	material	smcAC	smcDC	vib_table	vib_spindle	AE_table	AE_spindle
1	1	1	0	2	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
2	1	2	NaN	4	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
3	1	3	NaN	6	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
4	1	4	0.1100	7	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
5	1	5	NaN	11	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
6	1	6	0.2000	15	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
7	1	7	0.2400	19	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
8	1	8	0.2900	22	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
9	1	9	0.2800	26	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
10	1	10	0.2900	29	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
11	1	11	0.3800	32	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
12	1	12	0.4000	35	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
13	1	13	0.4300	38	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
14	1	14	0.4500	41	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
15	1	15	0.5000	44	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
16	1	16	NaN	46	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
17	1	17	0.4400	48	1.5000	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
18	2	1	0.0800	3	0.7500	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
19	2	2	0.1400	9	0.7500	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double
20	2	3	0.1400	12	0.7500	0.5000	1	9000x1 d...	9000x1 d...	9000x1 do...	9000x1 double	9000x1 do...	9000x1 double

Figure 19. Milling dataset structure in MATLAB

Table 2: Experimental Conditions

Case	Depth of Cut	Feed	Material
1	1.5	0.5	1- cast iron
2	0.75	0.5	1- cast iron
3	0.75	0.25	1- cast iron
4	1.5	0.25	1- cast iron
5	1.5	0.5	2- steel
6	1.5	0.25	2- steel
7	0.75	0.25	2- steel
8	0.75	0.5	2- steel
9	1.5	0.5	1- cast iron
10	1.5	0.25	1- cast iron
11	0.75	0.25	1- cast iron
12	0.75	0.5	1- cast iron
13	0.75	0.25	2- steel
14	0.75	0.5	2- steel
15	1.5	0.25	2- steel
16	1.5	0.5	2- steel

To predict the tool wear condition, six sensors were used to capture acoustic emissions, vibrations and current. The setup comprised of the table and the spindle of Matsuura machining center MC-510V, an acoustic sensor and a vibration sensor each mounted on the table and the spindle of the machining center. The acquired signals were amplified, filtered and fed through two removable mass storage (RMS) devices before entering the PC for data acquisition and processing. The signal from the spindle motor current sensor is fed directly into the PC without further processing [28].

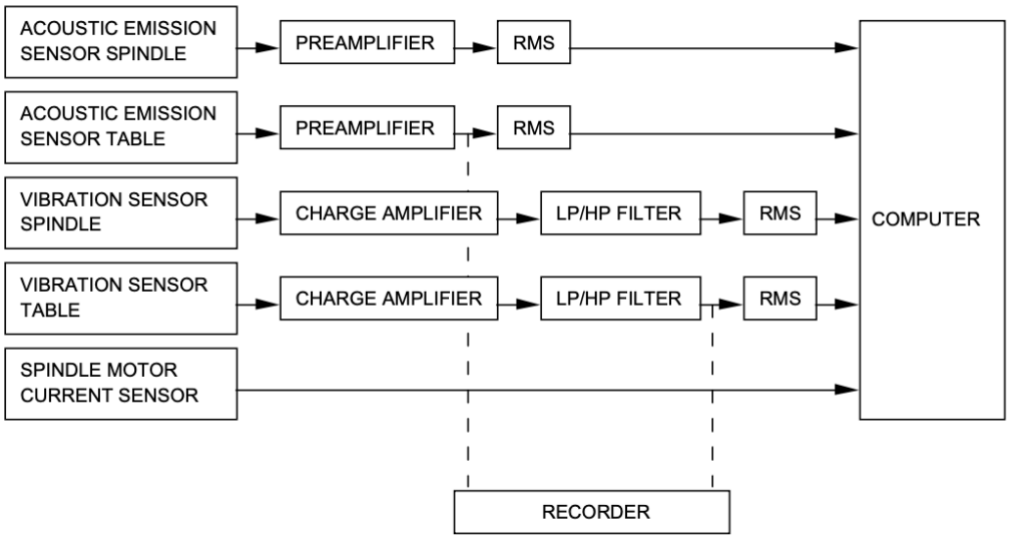


Figure 20. Experimental Setup

4.5 Python and Libraries

Python provides some of the strongest features and flexibilities that not only improve their competitiveness but also the consistency of the code, not to mention the vast libraries that help ease the workload. The following are numerous characteristics that place Python among the most preferred programming languages for Machine Learning, Deep Learning and Artificial Intelligence [29]:

1. Free and open source
2. Exhaustive libraries
3. Easy implementation and integration
4. Works well with C and C++

Some of the python libraries used for in this machine learning study includes [30] [31], [32]:

4.4.1 Numpy

With the assistance of a wide set of high-level mathematical functions, NumPy is a popular python library for large multi-dimensional array and matrix processing. It is useful in Machine Learning for basic scientific computations, linear algebra, fourier transform, and random number capabilities.

4.4.2 SciPy

SciPy is yet another popular library. At the core data structure, SciPy uses NumPy arrays, and comes with modules for various scientific programming activities widely used, including linear algebra, integration (calculus), ordinary differential equation solving, and signal processing.

4.4.3 Sci-kit learn

It's one of the most common ML libraries for classical ML algorithms. Two basic Python libraries i.e., NumPy and SciPy, are built on top of it. Most of the ML algorithms are supported by Scikit-learn. For data mining and data processing, Scikit-learn can also be used.

4.4.4 Pandas

Pandas is a NumPy-based data manipulation library that provides several useful functions for storing, indexing, merging, and grouping data. The primary data structure (Data Frame) is similar to what we can see on R i.e. heterogeneous name indexing data tables, time series operations, and data auto-alignment.

4.4.5 Matplotlib

When a programmer needs to imagine the patterns in the results, matplotlib can be helpful. It is a library for 2D plotting that is used to construct 2D graphs and plots. A module called pyplot makes plotting easier as it offers features for manipulating line types, font properties, axis formatting, etc. It offers different forms of data analysis graphs and plots, such as histograms, error charts, bar chats, etc.

4.4.6 Seaborn

Seaborn is a matplotlib-based library. It offers a high-level GUI to draw mathematical graphics that are appealing and insightful.

CHAPTER 5 – DATA ANALYSIS

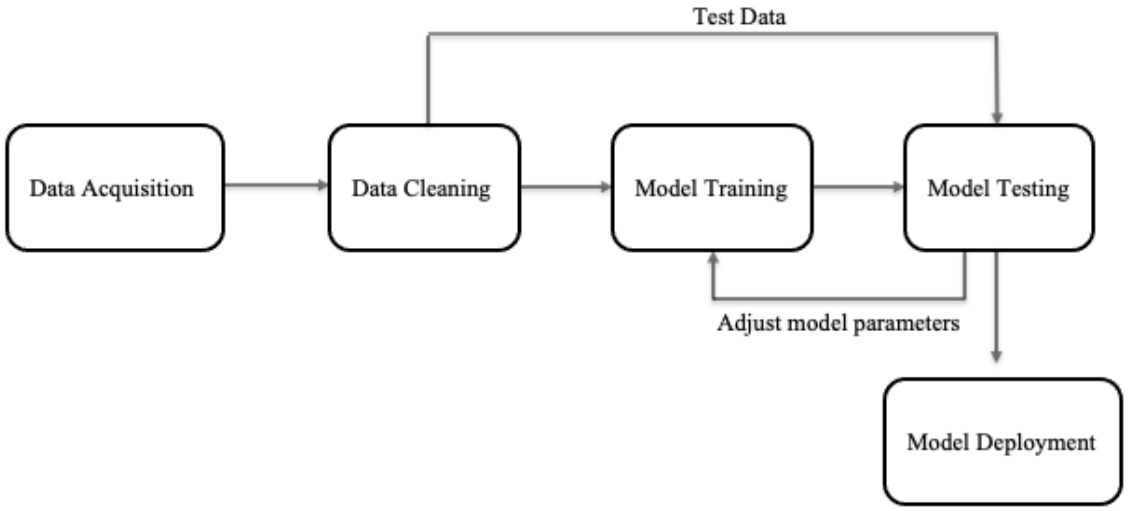


Figure 21. Machine Learning Pipeline

5.1 Importing Data

The 1x 167 matlab struct array is imported into the python workspace where the sensor readings and the process parameters are individually called and created as a combined data frame “mill” with 167 rows and 13 columns.

```
#Combining Dataframes (Process+Sensor)
```

```
mill = pd.concat([process,sensor_input], axis = 1)
mill.shape
```

```
(167, 13)
```

Figure 22 . Code Snippet - Mill

5.2 Data Cleaning

One of the main steps in the overall process of data processing is data cleansing. It is the method of finding and correcting missing, erroneous and unreliable information. The goal is

to resolve issues of data consistency that adversely affect the model's output and undermine the methodology and outcomes of the study. There are various kinds of problems with data consistency, including incomplete values (Nil, NULL, 0), redundant data, outliers or data that is unreliable. Quality data should possess the following characteristics [33]:

1. **Validity**
   How much the data can be conformed to the defined objective.
2. **Accuracy**
   Closeness to true values.
3. **Completeness**
   The degree to which the date set is complete.
4. **Consistency**
   Maintaining the consistency of the dataset across multiple datasets
5. **Uniformity**
   Use of same unit or measure across the dataset

5.2.1 Missing values

Missing values are quite coming in datasets. It just signifies that there's no value stored for that observation. There are two approaches to deal with missing values i.e., **deletion** and **imputation**. The first approach requires the elimination of observation data that has one or more missing values, either the deletion of the whole row or the deletion of the missing value columns, which is used in particular where there is no association between the missing variable and other variables. The second technique applies to the missed value approximation where the missing value depends on the value of every other variable.

During the experiment the flank wear was not measured at all times, which resulted in no VB measurements. For the purpose of this thesis since the cells containing missing values of VB weren't many (21 rows) and without VB values the supervised model couldn't be tested and trained, we decided to eliminate the entire row.

Resultant shape of the dataset: 146 x 13.

```
cleaned_data = mill.dropna()
filtered_data = mill[mill['VB'].isnull()]
cleaned_index = cleaned_data.index
cleaned_data.shape
```

```
(146, 13)
```

Figure 24 Code Snippet - Cleaned Data Shape

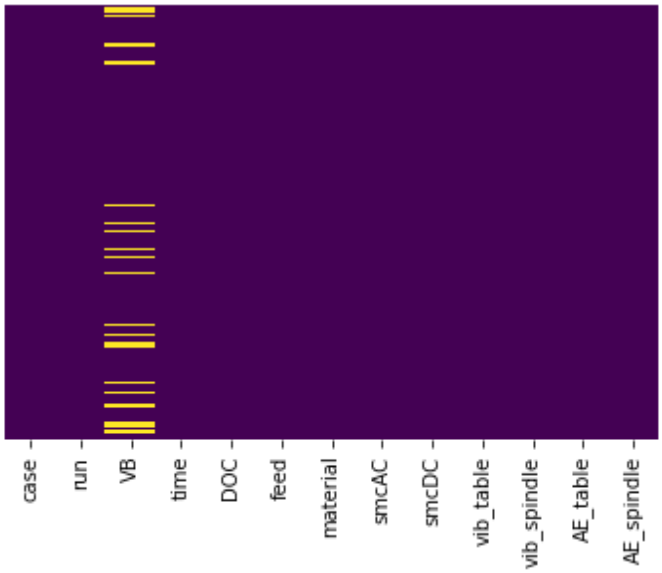


Figure 23 Null Heatmap

5.2.2 Outlier detection

An outlier is a point among the observation that is distant from the rest of the results. It may occur because of normal measuring fluctuations or it may signify experimental errors, such as human error and inaccuracy of the instrument. In data mining activities, outliers can cause significant issues and must be detected and evaluated in order to understand their existence and most importantly its removal from the dataset. Statistical approaches such as standard deviation, box plots, z-score or interquartile distribution methods (IQR) are the most used techniques to detect outliers.

In our study we'll be using z-score analysis. It's nothing but the relationship with the standard deviation and the mean of the group of data points. It's finding the

distribution where the mean is 0 and the standard deviation is 1. We re-scale and centralize the data when measuring the Z-score and search for data points that are too far from zero. The outliers would be viewed as those data points that are far from zero. A threshold of 3 or -3 is used in most situations. If the Z-score is greater than 3 or less than -3, those data points are marked as outliers [34].

We'll take into consideration the smcAC column here, where the below z-score code snippet was used to analyze each signal of the case/ run for the entire column and provided the outliers that were distant from the rest of each signal measurement. From the attached signal readings, it is clear that the outliers found (1 row) for a respective case or run will reflect across other sensor readings for the same case/run thereby which the entire row was removed.

Resultant shape of the dataset after outlier analysis: 145 x 13

```
for i in cleaned_index:
    meani[i]=np.mean(cleaned_data.smcAC[i])
    stddev[i]=np.std(cleaned_data.smcAC[i])
    z_score1[i]=np.sum(meani[i] - stddev[i])
    z_score=np.divide(z_score1, stddev[i])

df_z_score = pd.DataFrame(z_score)
df_z_score.columns = ['z_score']


#Removing outliers

df = df_z_score[(df_z_score.z_score <= -3) | (df_z_score.z_score >= 3)]
df_outliers = df.loc[(df!=0).any(axis=1)]
a= df_outliers.index

cleaned_outlier_data = cleaned_data.drop(index = a)
number_of_active_rows = cleaned_outlier_data. count

df1 = df_z_score[(df_z_score.z_score > -3) & (df_z_score.z_score < 3)]
df1_outliers = df1.loc[(df1!=0).any(axis=1)]

defective_index = df_outliers.index
clean_index = df1_outliers.index
cleaned_outlier_data.shape
```

```
(145, 13)
```

Figure 25 Code Snippet - Z Score Analysis

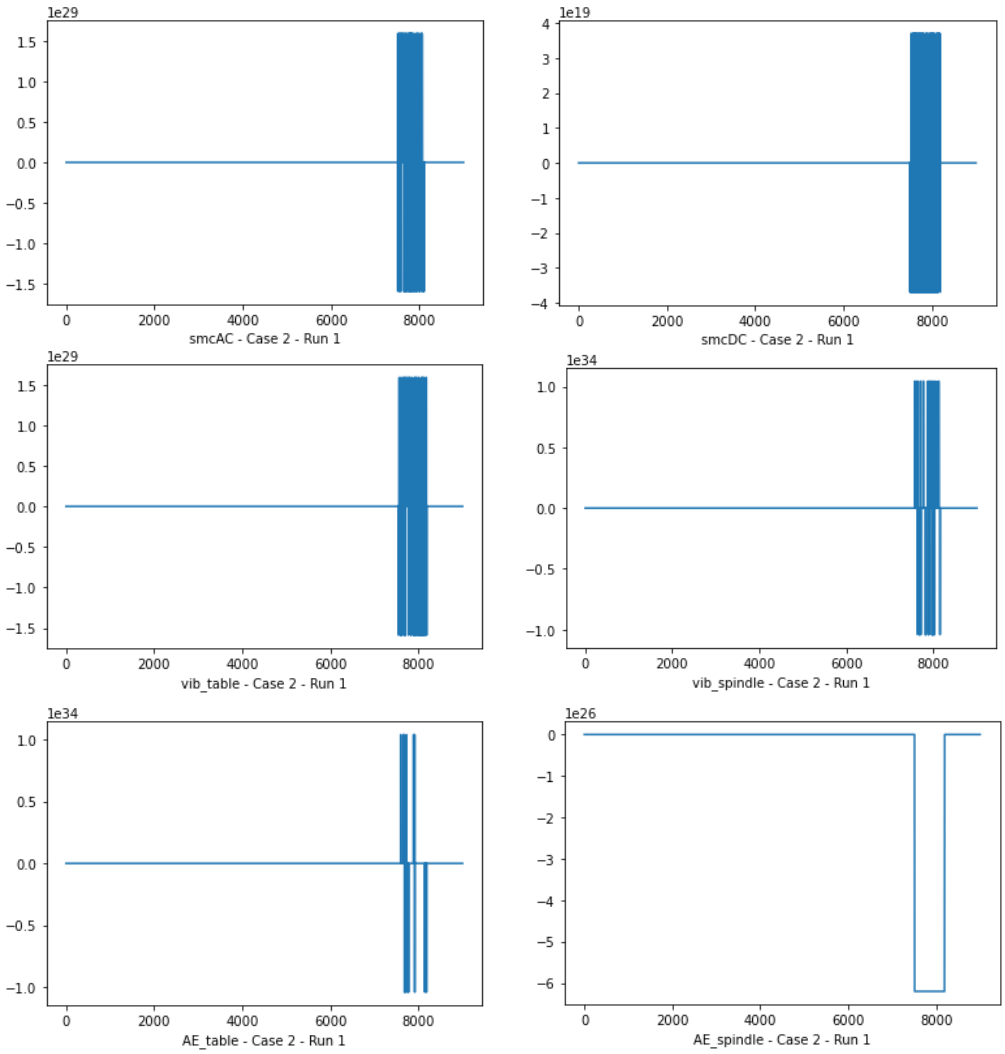


Figure 26 Outliers - Case 2, Run 1

5.3 Feature Extraction

Feature Extraction helps to minimize the number of features in a dataset by generating (and then discarding the original features) new features from the current ones. Any of the details found in the initial package of features should then be able to sum up this new reduced set of features. In this way, from a variation of the original set, a summarized version of the original features can be generated [35]. The quality and quantity of the features play a vital role in determining the result of the prediction.

The data captured by the sensors are used to extract the features in time, frequency and statistical domains. Each plays its role in extracting information pertaining to the tool wear.

5.3.1 Time Domain

Feature extraction in time domain helps to evaluate the magnitude of the signal. Some of the domain features used for this study are mentioned below which were introduced in the literature [36], [37], and proved effective to distinguish the signals and its related tool wear.

Table 3 - Time Domain Features Extracted

Index	Feature	Description
1	Maximum	$X_{MAX} = max(x_i)$
2	Mean	$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$
3	Root mean square	$X_{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$
4	Variance	$X_V = \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n-1}$
5	Standard deviation	$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n-1}}$
6	Skewness	$X_S = \frac{1}{n}\frac{\sum_{i=1}^{n}(x_i-\mu)^3}{\sigma^3}$
7	Kurtosis	$X_K = \frac{1}{n}\frac{\sum_{i=1}^{n}(x_i-\mu)^4}{\sigma^4}$
8	Peak-to-peak	$X_{P2P} = max(x_i)\text{-}min(x_i)$
9	Crest factor	$X_{CF} = \frac{max(xi)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}}$

The maximum and mean characteristics refer to the signal's maximum and mean amplitude. The square root of the mean square is RMS, and it represents a signal's

```
for i in clean_index:
    maxi[i]=np.max(cleaned_outlier_data.smcAC[i])
    mini[i]=np.min(cleaned_outlier_data.smcAC[i])
    meani[i]=np.mean(cleaned_outlier_data.smcAC[i])
    sizi=cleaned_outlier_data.smcAC[i].size
    sumofsquares=np.sum(cleaned_outlier_data.smcAC[i]**2)
    rms[i]=np.sqrt(sumofsquares/sizi)
    stddev[i]=np.std(cleaned_outlier_data.smcAC[i])
    skewi[i]=skew(cleaned_outlier_data.smcAC[i])

    kurt[i]=np.sum((cleaned_outlier_data.smcAC[i]-meani[i])**4)
    kurto[i]=np.divide(kurt[i], stddev[i]**4)
    kurti[i]=np.divide(kurto[i], sizi)

    p2p[i]=maxi[i]-mini[i]
```

Figure 27 Time Domain smcAC

average power. The signal dispersion around its mean value is calculated by variance and standard deviations. How often the signal fluctuates from the mean is determined by the standard deviation, while the variance reflects the power of this fluctuation. Skewness and kurtosis are used for signals that are not stationery to understand the probability density function. For the purpose of this thesis, we wouldn't be using variance and crest factor as they could be calculated from the rest of the features. These 7 features would be extracted from 6 sensors, giving a total of 42 new features in the data set. Time domain features help to convey the signal changes over time, but it cannot completely help us in providing the relation between wear and signals. It's for this reason we will be needing frequency domain.

5.3.2 Frequency Domain

It represents how much of each signal lies within each frequency band which is spread over a certain spectrum of frequencies. Using Fourier transform, a time function is transformed into a sum or integral of sine waves with different frequencies, each representing a frequency component which disintegrates a function into the sum of sine wave frequencies. The digital signal in time series is converted to frequency domain using Discrete Fourier Transformation (DFF) which results in statistics that are calculated on module and argument of the complex outputs of the corresponding DFT.

Just like the time domain features, we extract another list of statistics of band power spectrum [36]. At the end, 7 features each for the sensor readings

Table 4 Feature Domain Features Extracted for Mod / Arg

Index	Feature	Description
1	Maximum of band power spectrum	$S_{MAX} = max(S(f)_i)$
2	Sum of band power spectrum	$S_{SBP} = \sum_{i=1}^{n} S(f)_i$
3	Mean of band power spectrum	$S_{\mu} = \frac{1}{n}\sum_{i=1}^{n} S(f)_i$
4	Variance of band power spectrum	$S_V = \frac{\sum_{i=1}^{n}(S(f)_i - S_{\mu})^2}{n-1}$
5	Skewness of band power spectrum	$S_S = \frac{1}{n}\frac{\sum_{i=1}^{n}(S(f)_i - S_{\mu})^3}{S_V^{3/2}}$
6	Kurtosis of band power spectrum	$S_K = \frac{1}{n}\frac{\sum_{i=1}^{n}(S(f)_i - S_{\mu})^4}{S_V^{4/2}}$
7	Relative spectral peak per band	$S_{RSPPB} = \frac{max(S(f)_i)}{\frac{1}{n}\sum_{i=1}^{n} S(f)_i}$

corresponding to modulus and argument are extracted resulting in 84 columns for frequency domain in total.

Resultant dataset shape after time and frequency domain extraction:

145 x 126.

```
Data_feature_extraction = pd.concat([process, Time_domain, Frequency_domain], axis = 1)
Final_feature_extraction = Data_feature_extraction.dropna()

Normal = pd.concat([Time_domain, Frequency_domain], axis = 1)
Normal_input = Normal.dropna()
Normal_input.shape
```

```
(145, 126)
```

Figure 28 Feature Extraction Data Frame Shape

5.5 Normalisation

The aim of normalization is to increase the overall consistency of a dataset by rescaling the data dimension and preventing situations under which certain values overweight others since significant variations in number scale could trigger problems while attempting to combine values as characteristics during modeling. Some of the notably used normalization techniques in statistics include [38]:

1. The Maximum Absolute Scaling
2. The Min-Max Feature Scaling
3. The Z-Score Method
4. The Robust Scaling

For our study we use Min-Max normalization where all the features regarding the sensor signals (except the process parameters) are normalized. The range of features are then linearly scaled between 0 and 1 based on the following formula.

$$z = \frac{\mathrm{x} - \min(x)}{\max(x) - \min(x)}$$

5.6 Feature Selection

To improve the accuracy of a model and its performance efficiency, the number of features is reduced, at least those that increase the dimensionality. The data set is split into two: one containing the process parameters excluding VB, i.e., **Feature_selection_subset1** and the other containing the extracted features and VB, **Feature_selection_subset2**. The aim of feature selection is to remove those features that are correlated or derived from one another, in order to reduce redundancy. One of the common feature selection methods is the use of a correlation matrix, which removes highly correlated features based on Pearson's Correlation Coefficient.

```
Feature_selection_subset1 = Final_feature_extraction[['case','run','DOC','feed','material']]
Feature_selection_subset2_1 = Final_feature_extraction[['VB']]
Feature_selection_subset2_2 = df_normalized

Feature_selection_subset2_3 = [Feature_selection_subset2_1, Feature_selection_subset2_2]
Feature_selection_subset2 = pd.concat(Feature_selection_subset2_3, axis=1)
Feature_selection_subset2.drop('Maximum_power_smcAC_arg',axis=1,inplace=True)
Normalized_DB_=[Feature_selection_subset1, Feature_selection_subset2]
Normalized_DB = pd.concat(Normalized_DB_,axis=1)
```

Figure 29 Feature Selection Subsets

Pearson's Correlation Coefficient is determined as the covariance of the two variables divided by the product of their standard deviations. It implies a value between +1 and −1, where 1 reveals a positive linear correlation overall, −1 a negative one overall, and 0 suggests

that there is no linear correlation. The correlation matrix can be used as a method for choosing the appropriate characteristics. If there's a feature showing strong correlation with another feature, it means they are redundant and changes in one feature can impact the other - these kinds of features are removed from the dataset. This can be visualized with the help of a heatmap which is attached below.

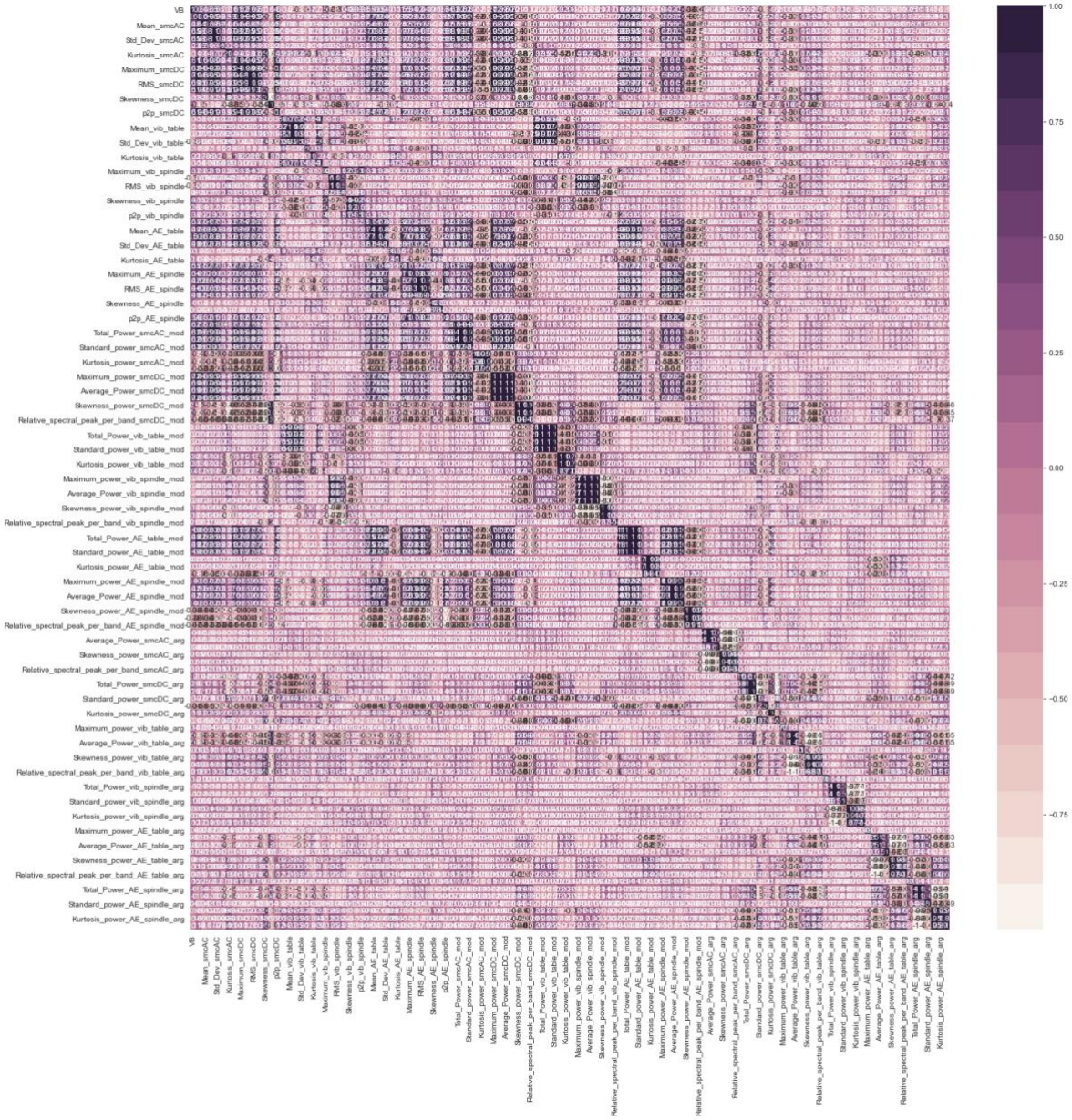


Figure 30 Feature Selection Heatmap

With the help of heatmap, the features which are correlated between 50% and 75% towards the target variable "VB" are filtered and selected for modelling.

```
#Correlation with output variable
cor_target = abs(cor['VB'])
cornew=cor_target.between(0.5, 0.75, inclusive=True)
relevant_features = cor_target[cornew]
relevant_features = pd.DataFrame(relevant_features)
```

Figure 31 Correlation Code Snippet

After applying correlation matrix, 22 features were shortlisted:

```
relevant_features.index
```

```
Index(['Maximum_smcAC', 'RMS_smcAC', 'Std_Dev_smcAC', 'p2p_smcAC',
       'Maximum_smcDC', 'Mean_smcDC', 'RMS_smcDC', 'Std_Dev_smcDC',
       'p2p_smcDC', 'Maximum_AE_table', 'RMS_AE_table', 'p2p_AE_table',
       'Maximum_AE_spindle', 'p2p_AE_spindle', 'Maximum_power_smcAC_mod',
       'Total_Power_smcAC_mod', 'Average_Power_smcAC_mod',
       'Standard_power_smcAC_mod', 'Maximum_power_smcDC_mod',
       'Total_Power_smcDC_mod', 'Average_Power_smcDC_mod',
       'Standard_power_smcDC_mod'],
      dtype='object')
```

Figure 32 Features Selected

5.7 Dataset Split

The normalized dataset is split 70 (10 cases) - 30 (6 cases) based on the selected featured for training and testing the models which are discussed further in the following sections.

```
#Dataset Split
```

```
Train = Normalized_DB[Normalized_DB.case <11] #70% of data
Test = Normalized_DB[Normalized_DB.case >=11] #30% of data
```

```
x_train = Train[['run','DOC','feed','material','Maximum_smcAC', 'RMS_smcAC', 'Std_Dev_smcAC',
                 'p2p_smcAC','Maximum_smcDC', 'Mean_smcDC', 'RMS_smcDC', 'Std_Dev_smcDC',
       'p2p_smcDC', 'Maximum_AE_table', 'RMS_AE_table', 'p2p_AE_table',
       'Maximum_AE_spindle', 'p2p_AE_spindle', 'Maximum_power_smcAC_mod',
       'Total_Power_smcAC_mod', 'Average_Power_smcAC_mod',
       'Standard_power_smcAC_mod', 'Maximum_power_smcDC_mod',
       'Total_Power_smcDC_mod', 'Average_Power_smcDC_mod',
       'Standard_power_smcDC_mod']]
y_train = Train.VB

x_test = Test[['run','DOC','feed','material','Maximum_smcAC', 'RMS_smcAC', 'Std_Dev_smcAC',
        'p2p_smcAC','Maximum_smcDC', 'Mean_smcDC', 'RMS_smcDC', 'Std_Dev_smcDC',
       'p2p_smcDC', 'Maximum_AE_table', 'RMS_AE_table', 'p2p_AE_table',
       'Maximum_AE_spindle', 'p2p_AE_spindle', 'Maximum_power_smcAC_mod',
       'Total_Power_smcAC_mod', 'Average_Power_smcAC_mod',
       'Standard_power_smcAC_mod', 'Maximum_power_smcDC_mod',
       'Total_Power_smcDC_mod', 'Average_Power_smcDC_mod',
       'Standard_power_smcDC_mod']]
y_test = Test.VB
```

Figure 33 Dataset Split

CHAPTER 6 – MODEL TRAINING AND TESTING

The normalized data set is split as mentioned above with 70% (10 cases) used for training the models and 30% (6 cases) for testing the models. Both the sets comprise of 26 columns and VB is considered as the target variable.

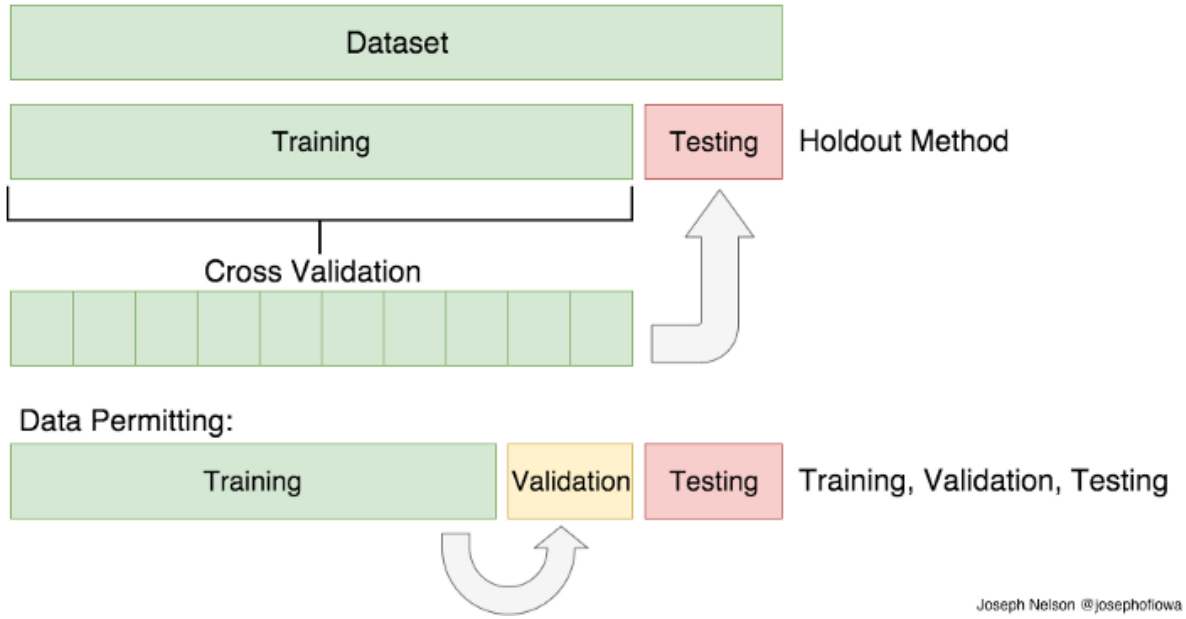


Figure 34 Dataset Split

We essentially try to build a model to predict the test data. Thus, to match the model and testing data to validate it, we use the training data. The created models are intended to predict the unknown outcomes, using the test set [39]. Later in the next section we will discuss further on the data validation set and how a model can be improved using K-fold cross validation. 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 "VB". The algorithms we will be covering ahead:

1. Linear Regression (LR)
2. Decision Forest (DF)
3. Boosted Decision Tree (BDT)
4. Neural Network (NN)
5. Ridge Regression (RR)

To measure the performance of a model certain performance evaluation metrics were used, and they are as follows:

1. Mean Squared Error (MSE)
2. Root Mean Square Error (RMSE)
3. Mean Absolute Error (MAE)
4. Coefficient of Determination (R^2)
5. Explained Variance

6.1 Evaluation Metrics

6.1.1 Mean Squared Error (MSE)

In mathematics, the estimator's mean squared error (MSE) or mean squared variance (MSD) calculates the average error squares, i.e., the average squared discrepancy between the expected amounts and the real value. MSE is a probability function that refers to the estimated squared error loss value. It's usually non negative and values close to zero are better [40]. The mean squared error function measures the mean square error, a probability metric referring to the squared (quadratic) error or loss value predicted [41].

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

6.1.2 Root Mean Square Error (RMSE)

The standard deviation of the residuals is basically the root mean square error (RMSE). Residuals are a measure of how far data points are out from the

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

regression line; RMSE is a measure of how these residuals are spaced out. It shows you, in other words, how concentrated the data is along the best fit line [42]. It's the square root of the average of the squared differences between the actual and predicted observations [43].

6.1.3 Mean Absolute Error (MAE)

In a series of projections, MAE calculates the average magnitude of the errors, without considering their course. It is the average of the absolute variations between forecast and real observation over the test set, where all individual differences have equal weight [43].

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

6.1.4 Coefficient of Determination (R^2)

It gives an indicator of fit efficiency and of how well unseen samples, by the proportion of explained variance, are likely to be expected by the model. Because such a difference depends on the dataset, R2 cannot be statistically similar across different datasets. The highest score available is 1.0 and it can be unfavorable.

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

A constant model that always forecasts the predicted value of y, regardless of the input characteristics, will get an R2 score of 0.0 [41].

6.1.5 Explained Variance

It is helpful to know how much initial variance can be described by the model in a linear regression problem. This idea is helpful for understanding the amount of data we lose by approximating the dataset. If this value is minimal, it means that there are strong oscillations in the data generation process and a linear model fails to catch them. It's an effective measure which is not quite different from R^2 [44]. The optimal value is when EV is close to 1.

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

6.2 Linear Regression

Linear Regression is an ML algorithm where the result is predicted by the use of known parameters that are correlated with the output. Rather than attempting to classify them into categories, it is used to predict values within a continuous range [45].

```
linreg = LinearRegression()
linreg.fit(x_train, y_train)

y_pred = linreg.predict(x_test)
y_pred = pd.DataFrame(y_pred)
```

Figure 35 Linear Regression Code Snippet

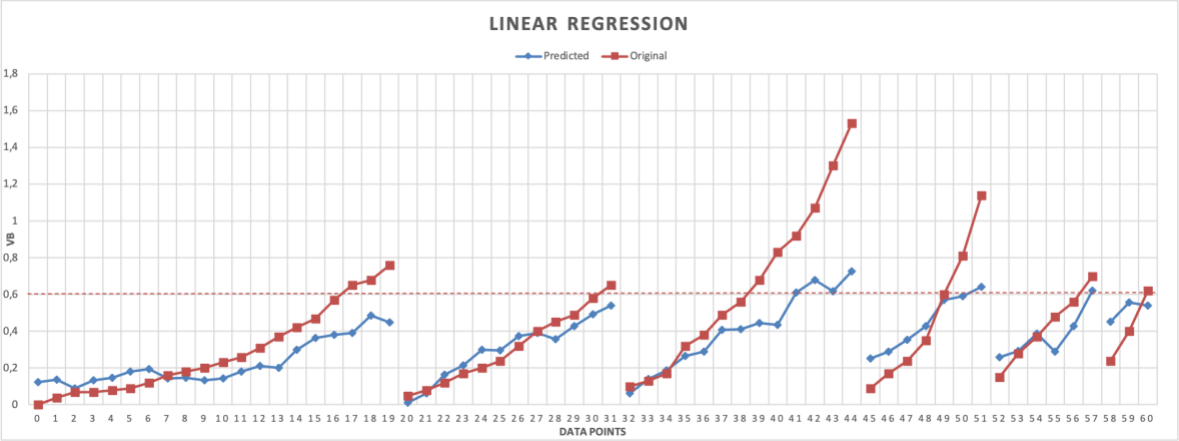


Figure 36 LR Wear Prediction - Case 11 - 16

```
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 = pd.DataFrame([lin_rmse, lin_mse, lin_mae, lin_r2, lin_variance])
lin_error.columns = ["Linear Regression"]
lin_error.index= ['RMSE', 'MSE', 'MAE', 'R^2', 'Explained Variance']
lin_error
```

	Linear Regression
RMSE	0.204355
MSE	0.041761
MAE	0.138893
R^2	0.598779
Explained Variance	0.657424

Figure 37 Linear Regression Evaluation Metrics

It's seen from the evaluation metrics and the wear prediction for cases 11-16 (test set) attached above that this model hasn't performed well. Moreover, the tool at certain instances have crossed the safe VB limit of 0.6. R^2 (0.598) and Explained variance (0.657) is another clear indication that supports the above inference.

6.3 Decision Forest

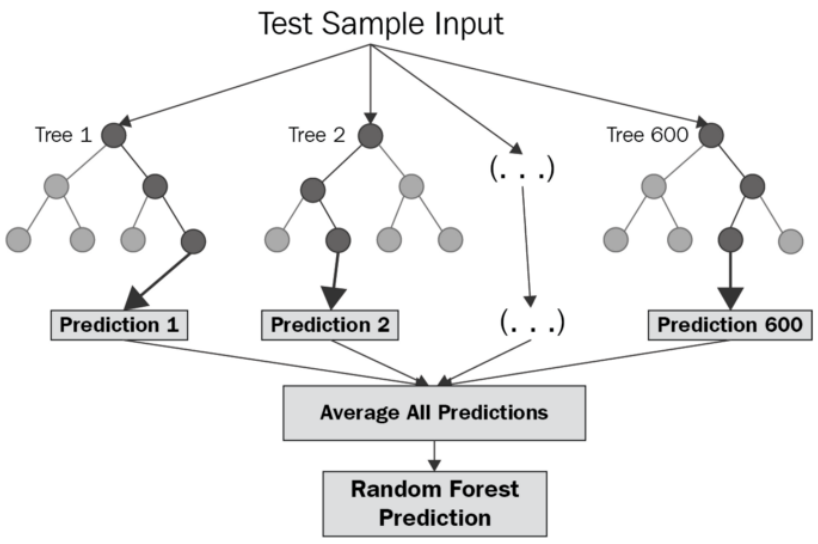


Figure 38 Random Forest Structure

Random forest is an algorithm for supervised learning which uses the classification and regression method of ensemble learning. Random forest is a technique for bagging (reducing variance) and not a technique for boosting (weighted averages). The trees in random woods run parallel to each other. When constructing the branches, there is no connection with these trees. It functions by constructing at training time a multitude of decision trees and generating the class that is the class mode (classification) or mean estimation (regression) of the individual trees [46].

```
from sklearn.ensemble import RandomForestRegressor

dforest = RandomForestRegressor(random_state = 500) #500 trees
dforest.fit(x_train, y_train)

y_pred = dforest.predict(x_test)
y_pred = pd.DataFrame(y_pred)
```

Figure 39 Decision Forest Code Snippet

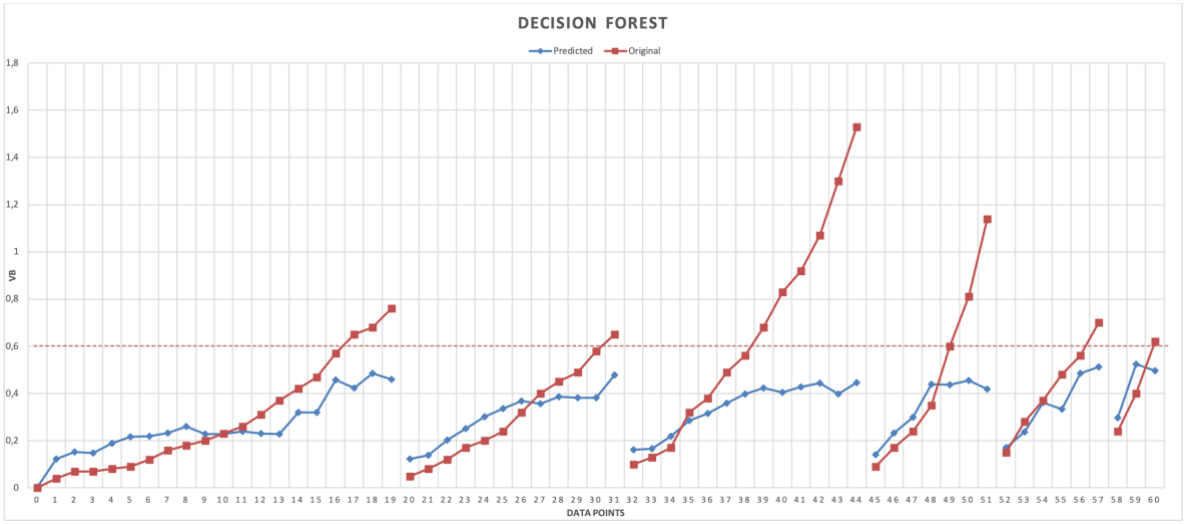


Figure 40 DF Wear Prediction - Case 11-16

```
df_rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
df_mse = mean_squared_error(y_test, y_pred)
df_mae = mean_absolute_error(y_test, y_pred)
df_r2 = metrics.r2_score(y_test, y_pred)
df_variance = metrics.explained_variance_score(y_test, y_pred)

df_error = pd.DataFrame([df_rmse, df_mse, df_mae, df_r2, df_variance])
df_error.columns = ["Decision Random Forest"]
df_error.index= ['RMSE', 'MSE', 'MAE', 'R^2', 'Explained Variance']
df_error
```

	Decision Random Forest
RMSE	0.261393
MSE	0.068326
MAE	0.161111
R^2	0.343549
Explained Variance	0.436600

Figure 41 Decision Forest Evaluation Metrics

It's seen from the evaluation metrics and the wear prediction for cases 11-16 (test set) attached above that this model hasn't performed well. Moreover, the tool at certain instances have crossed the safe VB limit of 0.6. R^2 (0.343) and Explained variance (0.436) is another clear indication that supports the above inference.

6.4 Boosted Decision Tree

In order to enhance the model, the gradient boosting algorithm sequentially blends weak learners in such a way that each new learner matches the residuals of the previous step. The final model aggregates the outcomes from each stage and achieves a good learner. The algorithm for gradient-boosted decision trees uses decision trees as weekly learners. To detect the residuals, a loss function is used [47].

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.datasets import load_boston
from sklearn.metrics import mean_absolute_error

regressor = GradientBoostingRegressor(max_depth=3,n_estimators=500,learning_rate=0.1)
regressor.fit(x_train, y_train)

errors = [mean_squared_error(y_test, y_pred) for y_pred in regressor.staged_predict(x_test)]
best_n_estimators = np.argmin(errors)

best_regressor = GradientBoostingRegressor(max_depth=3,n_estimators=best_n_estimators,learning_rate=0.1)
best_regressor.fit(x_train, y_train)
y_pred = best_regressor.predict(x_test)
y_pred = pd.DataFrame(y_pred)
```

Figure 42 Boosted Decision Tree Code Snippet

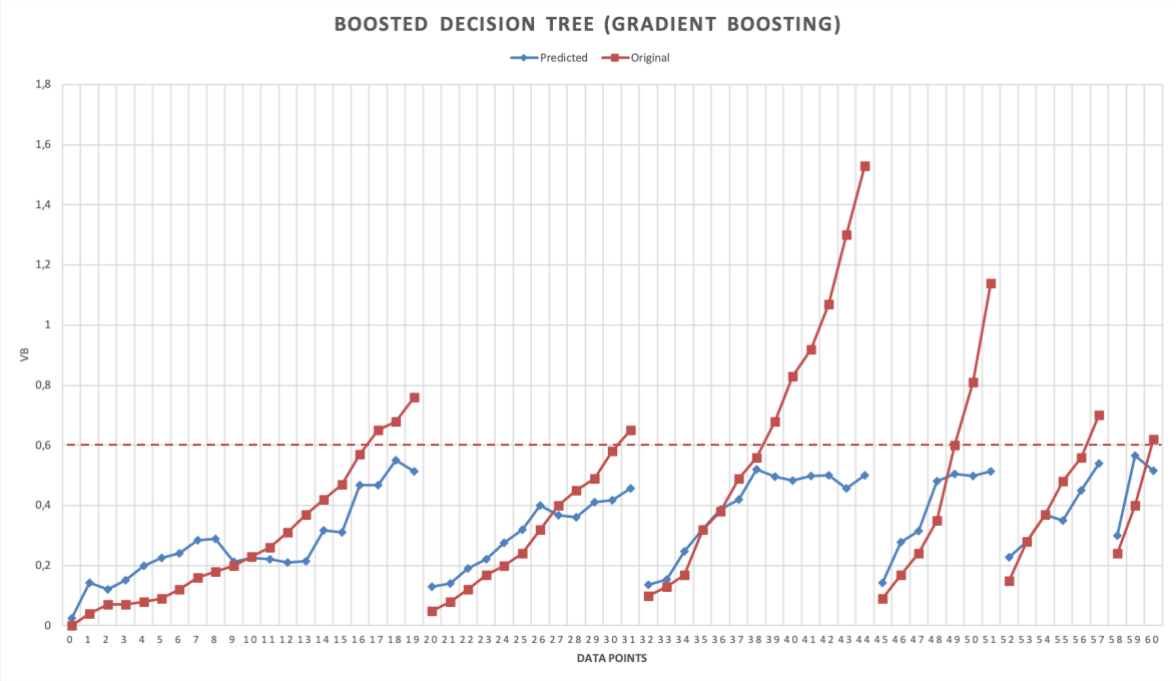


Figure 43 BDT Wear Prediction - Case 11 – 16

```
gb_rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
gb_mse = mean_squared_error(y_test, y_pred)
gb_mae = mean_absolute_error(y_test, y_pred)
gb_r2 = metrics.r2_score(y_test, y_pred)
gb_variance = metrics.explained_variance_score(y_test, y_pred)

gb_error = pd.DataFrame([gb_rmse, gb_mse, gb_mae, gb_r2, gb_variance])
gb_error.columns = ["Boosted Decision Tree"]
gb_error.index= ['RMSE', 'MSE', 'MAE', 'R^2', 'Explained Variance']
gb_error
```

	Boosted Decision Tree
RMSE	0.239297
MSE	0.057263
MAE	0.148646
R^2	0.449839
Explained Variance	0.504430

Figure 44 Boosted Decision Tree Evaluation Metrics

It's seen from the evaluation metrics and the wear prediction for cases 11-16 (test set) attached above that this model hasn't performed well. Moreover, the tool at certain instances have crossed the safe VB limit of 0.6. R^2 (0.449) and Explained variance (0.504) is another clear indication that supports the above inference.

6.5 Neural Network

Neural network is a method or algorithm for machine learning that aims to simulate neuron functioning for learning just like in the human brain. It is unreliable at first and adjusts itself after several iteration of data so that its accuracy increases. It's nothing but a computational system that gives predictions based on existing data [48].

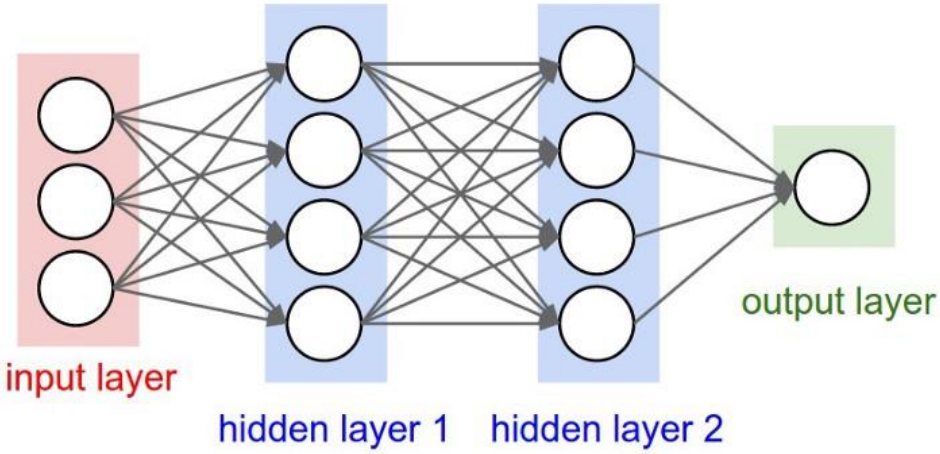


Figure 45 Feed Forward Neural Network

Each layer has random node numbers. The input layer has a number of nodes proportional to the dimensions of the features of the input data. And the hidden layers contain random node numbers. And the output layer consists of one node only if it's for regression and more than one if classification is an issue [48]. For this study we will be using the Multilayer Perceptron (MLP) from the scikit-learn library.

```
from sklearn.neural_network import MLPRegressor

mlpreg = MLPRegressor(hidden_layer_sizes=(100,),solver='lbfgs',alpha=0.0001,max_iter=2000,random_state = 1)
mlpreg.fit(x_train, y_train)

y_pred = mlpreg.predict(x_test)
y_pred = pd.DataFrame(y_pred)
```

Figure 46 Neural Network Code Snippet

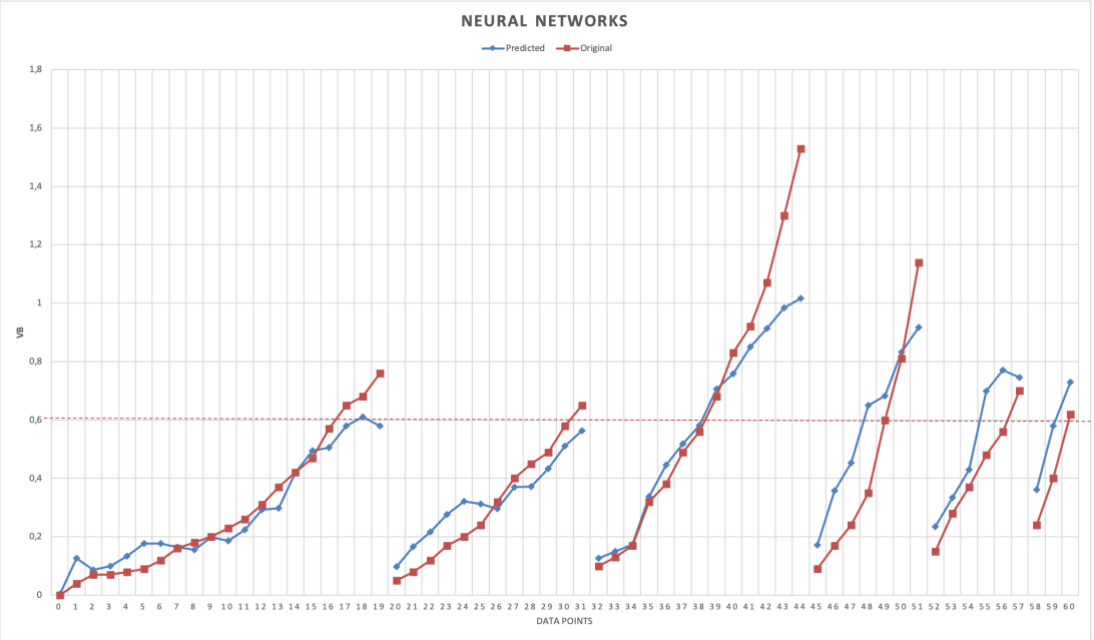


Figure 47 NN Wear Prediction - Case 11 - 16

```
nn_rmse  = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
nn_mse = mean_squared_error(y_test, y_pred)
nn_mae = mean_absolute_error(y_test, y_pred)
nn_r2 = metrics.r2_score(y_test, y_pred)
nn_variance = metrics.explained_variance_score(y_test, y_pred)

nn_error = pd.DataFrame([nn_rmse, nn_mse, nn_mae, nn_r2, nn_variance])
nn_error.columns = ["Neural Network"]
nn_error.index= ['RMSE', 'MSE', 'MAE', 'R^2', 'Explained Variance']
nn_error
```

	Neural Network
RMSE	0.125460
MSE	0.015740
MAE	0.087733
R^2	0.848774
Explained Variance	0.850407

Figure 48 Neural Network Evaluation Metrics

It's seen from the evaluation metrics and the wear prediction for cases 11-16 (test set) attached above that this model has outperformed the previous models. Even though, the tool at certain instances have crossed the safe VB limit of 0.6, the model has predicted and kept up with the actual VB reading. R^2 (0.848) and Explained variance (0.850) is another clear indication that supports the above inference.

6.6 Ridge Regression

In order to avoid over-fitting, which can arise from simple linear regression, Ridge regression is one of the simple strategies to minimize model complexity other than Lasso regression. The cost function is altered by adding a penalty i.e., the square of the magnitude of coefficients. Ridge regression essentially stresses on the coefficients and the penalty term regularizes the coefficients when the coefficients are large. In short, they shrink the coefficients and reduce model complexity and multi-collinearity [49].

```
regressor = RidgeCV(alphas=[0.1, 1, 10], store_cv_values=True)
regressor.fit(x_train, y_train)
cv_mse = np.mean(regressor.cv_values_, axis=0)
```

```
predict_y = regressor.predict(x_test)
predict_y=pd.DataFrame(predict_y)
```

Figure 49 Ridge Regression Code Snippet

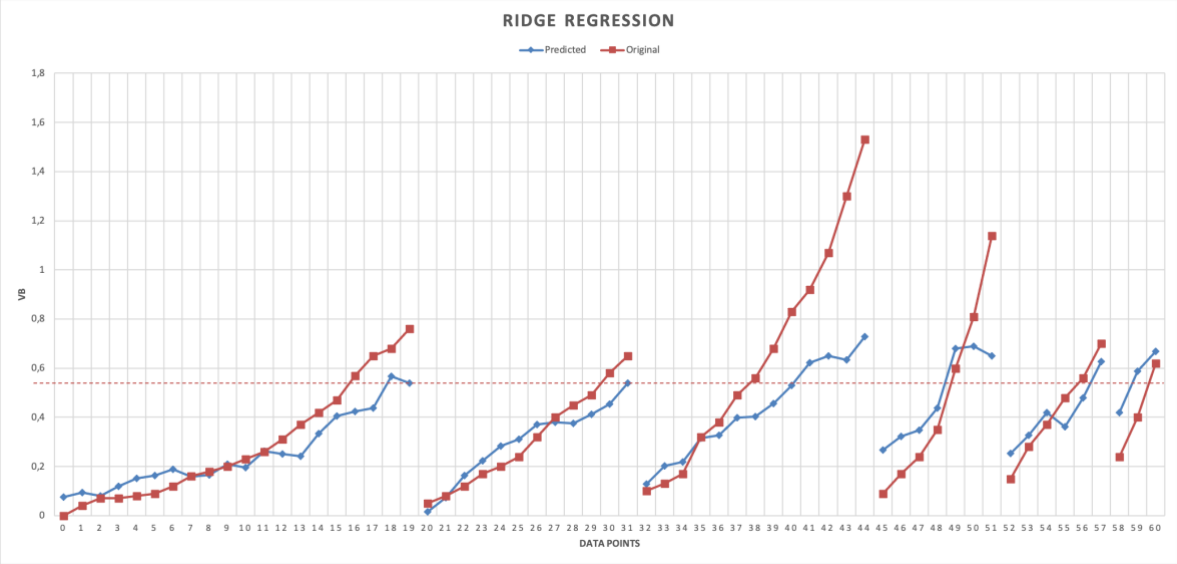


Figure 50 RR Wear Prediction - Case 11 - 16

```
rr_rmse  = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
rr_mse = mean_squared_error(y_test, y_pred)
rr_mae = mean_absolute_error(y_test, y_pred)
rr_r2 = metrics.r2_score(y_test, y_pred)
rr_variance = metrics.explained_variance_score(y_test, y_pred)

rr_error = pd.DataFrame([rr_rmse, rr_mse, rr_mae, rr_r2, rr_variance])
rr_error.columns = ["Ridge Regression"]
rr_error.index= ['RMSE', 'MSE', 'MAE', 'R^2', 'Explained Variance']
rr_error
```

	Ridge Regression
RMSE	0.187433
MSE	0.035131
MAE	0.130524
R^2	0.662475
Explained Variance	0.669664

Figure 51 Ridge Regression Evaluation Metrics

It's seen from the evaluation metrics and the wear prediction for cases 11-16 (test set) attached above that this model hasn't performed well. Moreover, the tool at certain instances have crossed the safe VB limit of 0.6. R^2 (0.662) and Explained variance (0.669) is another clear indication that supports the above inference.

Table 5 Summary of Evaluation Metrics

	LR	DRF	BDT	NN	RR
RMSE	0.204	0.261	0.239	0.125	0.187
MSE	0.041	0.068	0.057	0.0157	0.035
MAE	0.138	0.161	0.148	0.087	0.130
R^2	0.598	0.343	0.449	0.848	0.662
EV	0.657	0.436	0.504	0.850	0.669

CHAPTER 7 – MODEL IMPROVEMENT

We must be careful not to overfit our training data when constructing a model. In other words, in comparison to merely memorizing the results, we have to make sure that the model catches the underlying pattern. It's imperative that we verify how it treats unexpected data before using a model in output. Usually, this is achieved by dividing the details into two subsets, one for training and the other to assess the model's accuracy. Certain algorithms for machine learning depend on hyperparameters. Essentially, a hyperparameter is a user-set variable that dictates how the algorithm behaves. Step size in gradient descent and alpha in ridge regression are some examples of hyperparameters. When it comes to hyperparameters, there is no one size fits all [50].

There are many forms of cross validation, most frequently called k-fold cross validation. A test collection is always put off to the side for final evaluation in cross validation, but the validation set is no longer required. The training set is broken into k minor sets (or folds) in k-fold cross validation. Using k-1 of the folds, the model is then conditioned and the last is used as the validation set to calculate an output metric such as accuracy.

If a particular value is chosen for k, it can be used in the model comparison such as k=10 being a 10-fold cross-validation. If k=5, the dataset will be broken into 5 equivalent parts and the procedure below will run 5 times, with a different holdout set each time.

1. Take the set as a holdout or data collection for review
2. Take the remaining groups as a data collection for training
3. Fit a model on the training set and assess it on the test set.
4. Analyse the evaluation metrics and discard the model

Summarize the model's ability at the conclusion of the above step using evaluation metrics [51].

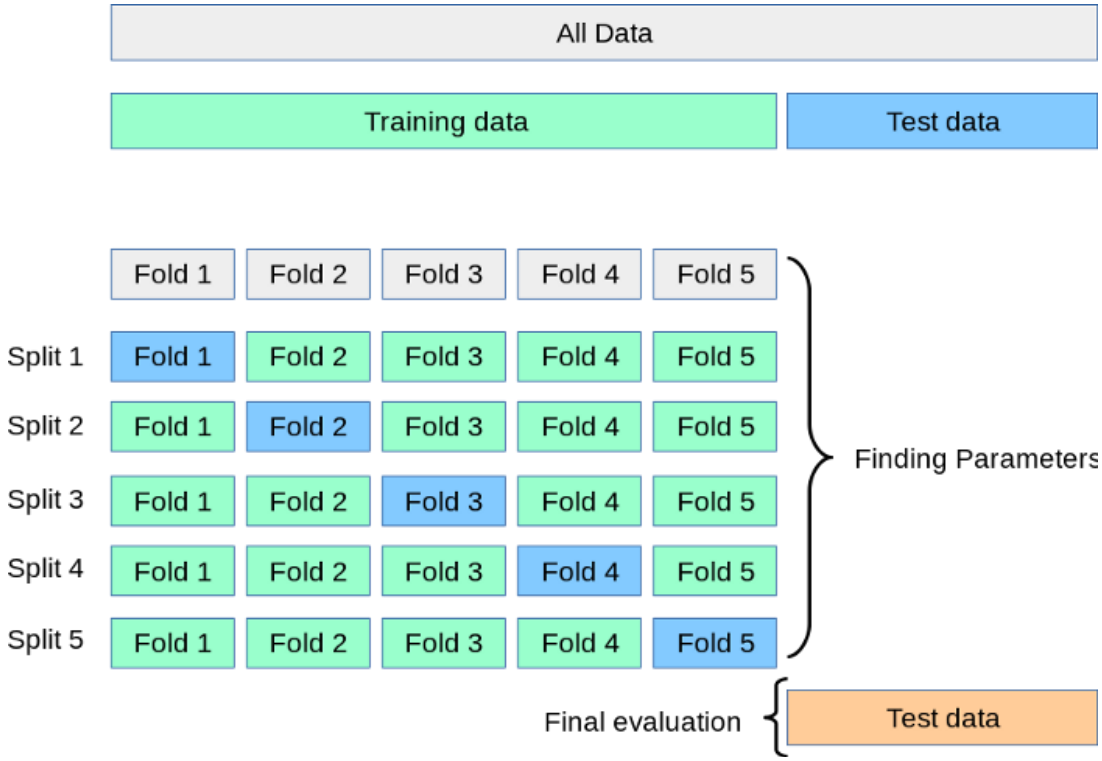


Figure 52 K-Fold Model Tuning

Those above trained and tested models are subjected to K-fold cross-validation to understand and analyze its improvements which would be later used towards the end of this study to draft a user input RUL determination for a tool taken into consideration. 10-fold cross-validation is conducted on the following models with their results and inferences discussed accordingly below:

1. Linear Regression (LR)
2. Decision Forest (DF)
3. Boosted Decision Tree (BDT)
4. Neural Network (NN)
5. Ridge Regression (RR)

7.1 Linear Regression (LR)

Table 6 K-Fold - LR

Fold Number	RMSE	MSE	MAE	R2	Explained Variance
0	0,247	0,061	0,169	-3,308	-2,195
1	0,075	0,006	0,069	0,762	0,964
2	0,074	0,005	0,058	0,798	0,870
3	0,075	0,006	0,063	-0,754	-0,721
4	0,074	0,005	0,066	0,751	0,764
5	0,091	0,008	0,083	0,667	0,694
6	0,077	0,006	0,053	0,925	0,927
7	0,087	0,008	0,075	0,819	0,821
8	0,200	0,040	0,132	0,381	0,600
9	0,128	0,016	0,099	0,479	0,579
Mean	**0,113**	**0,016**	**0,087**	**0,152**	**0,330**
Std Dev	**0,062**	**0,019**	**0,037**	**1,308**	**1,012**

As seen from the above summary of the 10-Fold cross validation done on the Linear Regression model, we can draw conclusions based on its mean and standard deviation stating its performance - In this case we see that due to erratic values for certain folds, the overall mean and standard deviation has been disturbed. Regardless, this improved model doesn't keep up with the rest of the models.

7.2 Decision Forest (DF)

Table 7 K-Fold - DF

Fold Number	RMSE	MSE	MAE	R2	Explained Variance
0	0,138	0,019	0,134	-0,343	0,913
1	0,085	0,007	0,071	0,694	0,905
2	0,079	0,006	0,059	0,772	0,818
3	0,043	0,002	0,031	0,443	0,718
4	0,082	0,007	0,066	0,691	0,892
5	0,080	0,006	0,064	0,740	0,823
6	0,150	0,022	0,117	0,716	0,722
7	0,119	0,014	0,089	0,663	0,701
8	0,097	0,009	0,055	0,855	0,857
9	0,217	0,047	0,178	-0,492	0,490
Mean	**0,109**	**0,014**	**0,086**	**0,474**	**0,784**
Std Dev	**0,049**	**0,013**	**0,044**	**0,483**	**0,130**

As seen from the above summary of the 10-Fold cross validation done on the Decision Forest model, we can draw conclusions based on its mean and standard deviation stating its performance. Though the evaluation metrics has given a mediocre score, it's still not the most reliable and accurate model to consider for RUL determination.

7.3 Boosted Decision Tree (BDT)

Table 8 K-Fold - BDT

Fold Number	RMSE	MSE	MAE	R2	Explained Variance
0	0,118	0,014	0,109	0,017	0,845
1	0,080	0,007	0,060	0,690	0,813
2	0,079	0,006	0,069	0,761	0,799
3	0,031	0,001	0,028	0,701	0,736
4	0,092	0,009	0,088	0,589	0,942
5	0,059	0,004	0,045	0,844	0,928
6	0,156	0,023	0,118	0,697	0,711
7	0,126	0,015	0,105	0,620	0,684
8	0,087	0,007	0,065	0,869	0,902
9	0,230	0,052	0,185	-0,642	0,453
Mean	**0,106**	**0,014**	**0,087**	**0,515**	**0,781**
Std Dev	**0,056**	**0,015**	**0,045**	**0,471**	**0,146**

As seen from the above summary of the 10-Fold cross validation done on the Boosted Decision Tree model, we can draw conclusions based on its mean and standard deviation stating its performance. Though the evaluation metrics has given a decent score, it's still not the most reliable and accurate model to consider for RUL determination. It's at par with the Decision Forest model.

7.4 Neural Network (NN)

Table 9 K-Fold - NN

Fold Number	RMSE	MSE	MAE	R2	Explained Variance
0	0,072	0,005	0,058	0,633	0,872
1	0,042	0,002	0,036	0,925	0,927
2	0,041	0,002	0,034	0,938	0,946
3	0,036	0,001	0,031	0,594	0,669
4	0,098	0,010	0,084	0,561	0,870
5	0,062	0,004	0,045	0,844	0,845
6	0,089	0,008	0,083	0,901	0,905
7	0,072	0,005	0,063	0,877	0,878
8	0,137	0,019	0,124	0,711	0,736
9	0,098	0,010	0,081	0,695	0,697
Mean	**0,075**	**0,006**	**0,064**	**0,768**	**0,835**
Std Dev	**0,032**	**0,005**	**0,029**	**0,145**	**0,098**

As seen from the above summary of the 10-Fold cross validation done on the Neural Network model, we can draw conclusions based on its mean and standard deviation stating its performance. This model is by far the one that's standing out from the rest with reliable metric accuracy and possibility for consideration in RUL determination.

7.5 Ridge Regression (RR)

Table 10 K-Fold - RR

Fold Number	RMSE	MSE	MAE	R2	Explained Variance
0	0,061	0,004	0,055	0,738	0,950
1	0,039	0,001	0,033	0,936	0,941
2	0,063	0,004	0,051	0,852	0,916
3	0,046	0,002	0,040	0,338	0,375
4	0,072	0,005	0,057	0,764	0,810
5	0,059	0,003	0,053	0,860	0,862
6	0,106	0,011	0,068	0,859	0,888
7	0,084	0,007	0,068	0,831	0,831
8	0,122	0,015	0,107	0,769	0,881
9	0,156	0,024	0,122	0,231	0,552
Mean	**0,081**	**0,008**	**0,065**	**0,718**	**0,801**
Std Dev	**0,037**	**0,007**	**0,028**	**0,237**	**0,188**

As seen from the above summary of the 10-Fold cross validation done on the Ridge Regression model, we can draw conclusions based on its mean and standard deviation stating its performance. This model comes second to the results obtained from K-Fold Neural Network with reliable metric accuracy and possibility for consideration in RUL determination.

In the next section we'll discuss the framework used to predict the RUL using the TCM results obtained with the NN model.

CHAPTER 8 – REMAINING USEFUL LIFE (RUL)

For estimating the RUL of the tool, we use the testing set (case 11-16) alongside the models that exhibited an acceptable performance in the above tests for the tool condition monitoring. In our case we will be using Neural Network to demonstrate the predicted RUL against the original runs. This is a near attempt to predict the RUL based on the available training dataset.

In order to determine the RUL, the following steps were adopted:

1. A data frame was created from the testing set comprising the process parameters – case, run, DOC, feed and material.
2. The predicted "VB" from the Neural Network is appended along with the actual "VB" values.
3. Now we have a data frame with the following columns: **case, run, DOC, feed, material, Predicted_VB and Actual_VB**.
4. RUL is further predicted for each run by calculating the factors below and then plotting to understand its behavior alongside appending them to the data frame created above:
    a. **Real or Predicted Used Life (%):** It's the result obtained when we divide the *actual or predicted VB* to the *safe VB limit, 0,6.*
    b. **Real or Predicted Max Runs:** It's the result obtained when we divide the *actual number of runs* to the *Real or Predicted Used Life (%).*
    c. **Real or Predicted RUL:** It's the result obtained when the *actual number of runs* are deducted from the *Real or Predicted Max Runs.*
5. The results obtained are plotted (RUL vs Runs) to analyze the RUL performance with the TCM results obtained earlier for the Neural Network model.

The 0 RUL corresponds to the safe VB limit of 0,6.

From cases 11 and 12 we can see that the predicted results for the RUL for the first 8-9 runs, though it predicts an unsatisfactory result, start improving as it runs for the rest of the case. We also see that for both the cases the actual RUL hits the 0 RUL (safe wear limit) first compared to the predicted RUL which means that the tool for that respective run has hit its limit while the predicted RUL says there a few more runs remaining to hit the tool's safe limit.

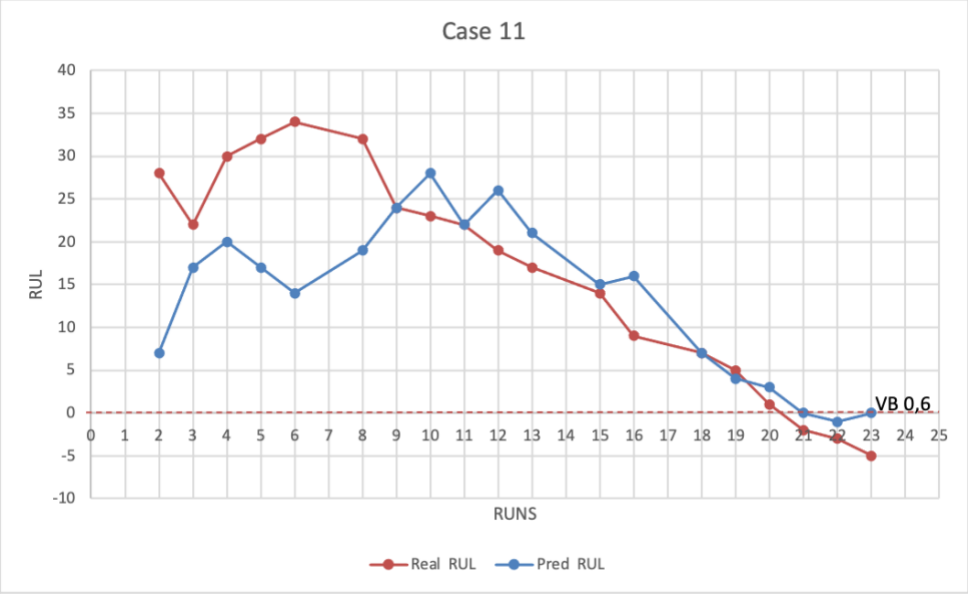


Figure 53 RUL vs Runs - Case 11

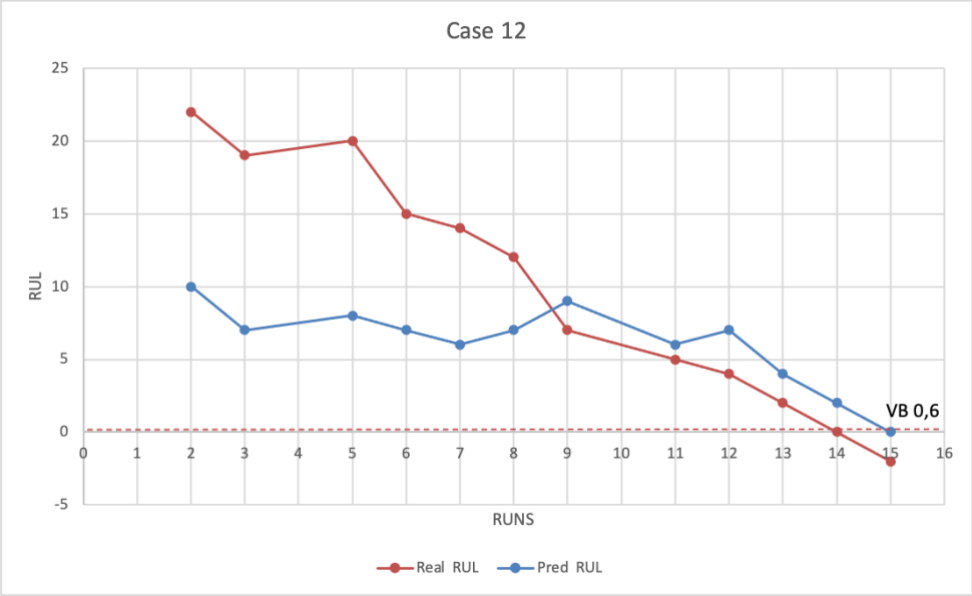


Figure 54 RUL vs Runs - Case 12

From cases 13 and 14 we can see that the predicted results for the RUL for the first 6-7 runs, though it predicts an unsatisfactory result, start improving as it runs for the rest of the case. Case 13 has given the most satisfying results compared to the rest of the test cases. It is evident that the actual and the predicted RUL coincides almost for most of the runs, which is a desirable result. We also see that, for case 14, the predicted RUL hits the 0 RUL (safe wear limit) first compared to the actual RUL which means that the tool for that respective run has hit its limit while the actual RUL says there a few more runs remaining to hit the tool's safe limit.

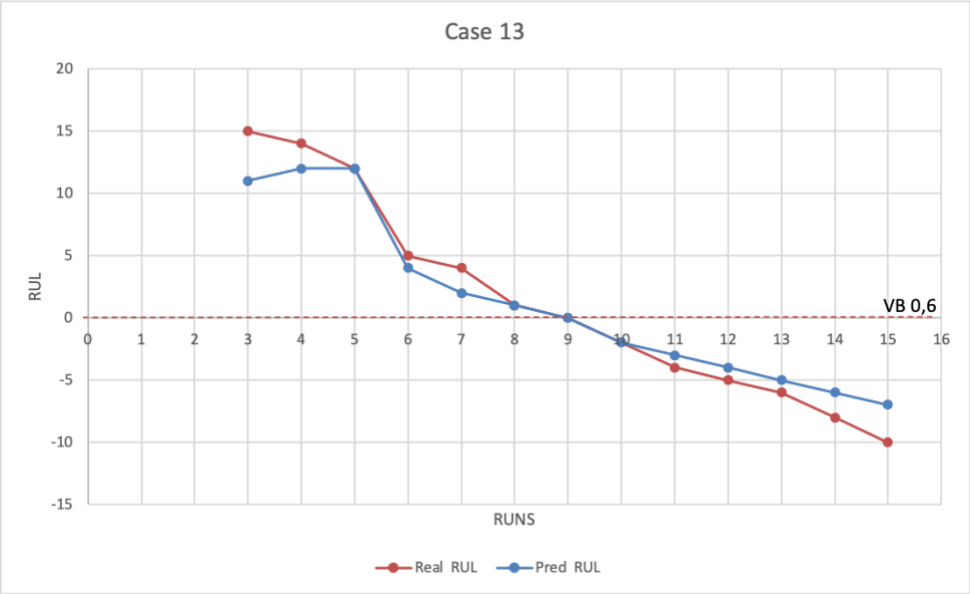


Figure 55 RUL vs Runs - Case 13

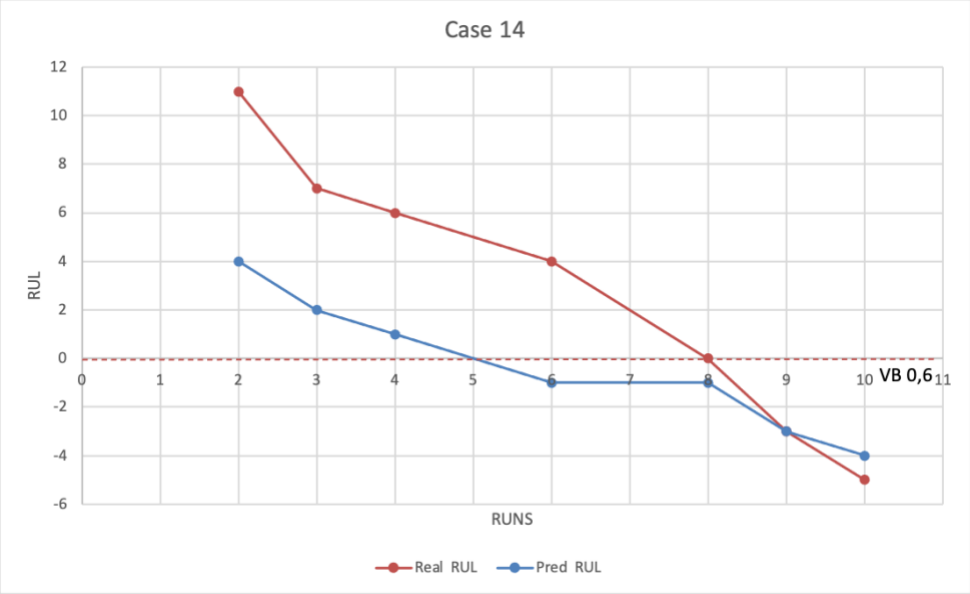


Figure 56 RUL vs Runs - Case 14

From cases 15 and 16, we can see that the predicted results for most of the runs provide an unsatisfactory result which could mainly account for the lack of sensor data for extraction and modelling. We also see that for both the cases the predicted RUL hits the 0 RUL (safe wear limit) first compared to the actual RUL which means that the tool for that respective run has hit its safe limit while the actual RUL says there a few more runs remaining to hit the tool's safe limit.

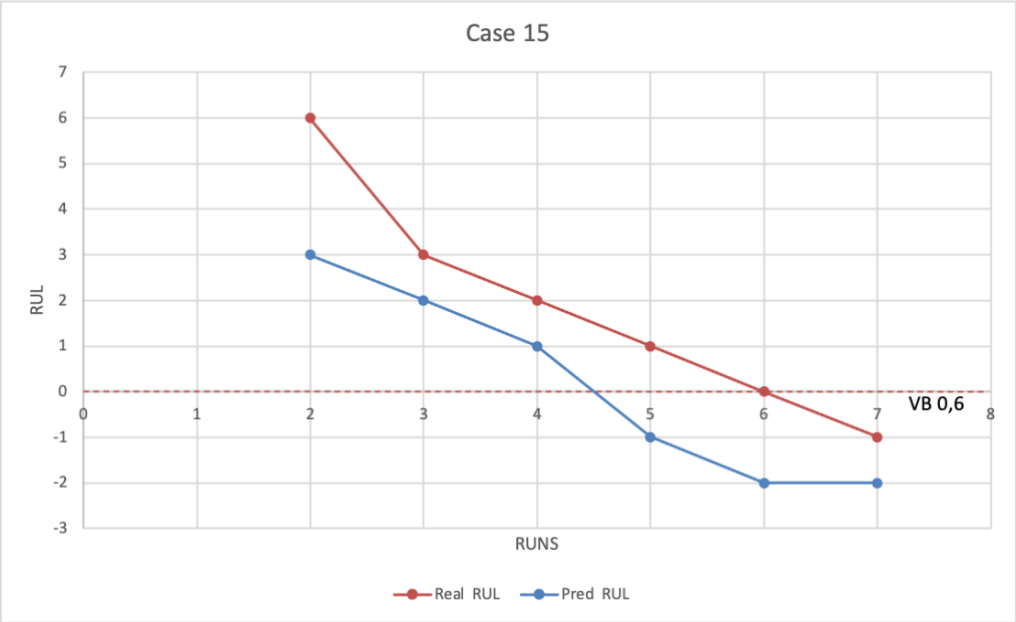


Figure 57 RUL vs Runs - Case 15

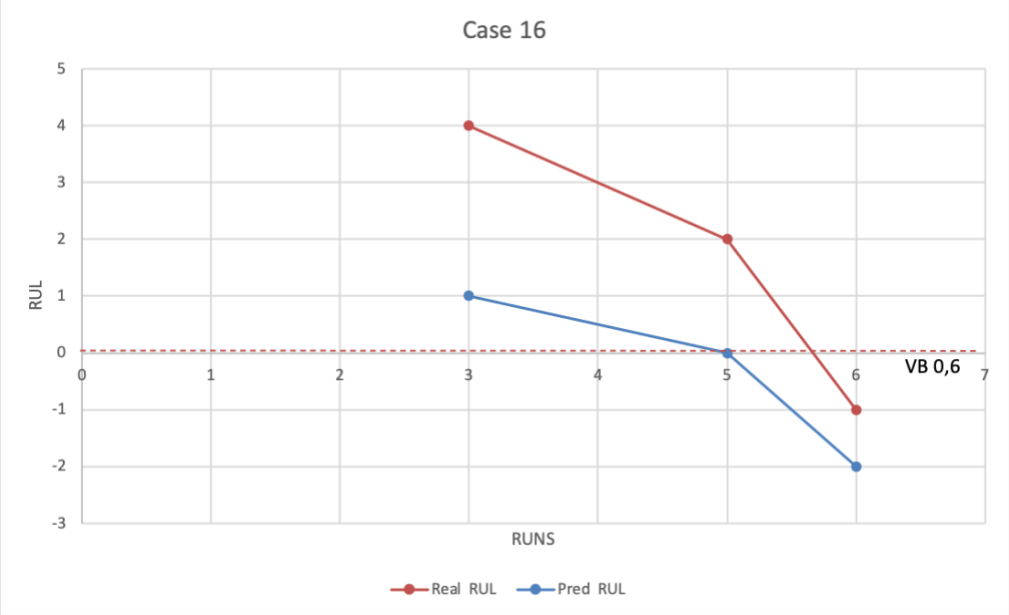


Figure 58 RUL vs Runs - Case 16

CHAPTER 9 – CONCLUSION AND FUTURE WORK

In this study we tried to understand the dataset furnished by NASA on milling operation to determine the tool condition monitoring (TCM) and the remaining useful life (RUL) of the tool through various machine learning algorithms. And based on the performance put forward by the ML algorithms, the one that stood out was later chosen for determining the RUL of the tool. The obtained results from the models were even further improved through k-fold cross validation to evaluate the difference in the accuracy metrics obtained during different folds.

Finally, we tried to estimate the RUL with the help of the testing set used to obtain the VB prediction results for Neural Network and determined the RUL performance alongside the actual runs for every case in the test set. Though the results were limited and basic due to the lack of concrete data, the framework used to estimate the RUL was a steppingstone towards predictive maintenance.

The big drawback used in this study is the amount of data used to train the model. This reality also determines the model's success outcomes. By trying to gather more data from multiple devices, by varying a different set of parameters and adding more kinds of sensors and data sources, this constraint could be overcome.

There’s scope for improvement in this study, whether we take the data analysis section or the signal processing part; one could delve into the subtleties for perfection and rectification in order to obtain a clean and satisfying result.

Future Work

One of the most important sections that could be touched upon is the signal processing part of the data set - we notice that the signals acquired through various sensors are subjected to machine noise and irrelevant information. This could be filtered out by understanding the range of the signals and the kind of sensors utilized to acquire data. This should facilitate to better filter the signals irrespective of the process parameters used or needed in order to predict better outcomes during data engineering or modelling. Attached below are the typical signals obtained for each sensor along with the results obtained through **change point identification (2nd Rows).** Here, the approach used to filter the signals is rudimentary and has been effective for most of the typical sensor signals. Basically, individual sensors have certain range between which they fluctuate, and this was considered for each sensor. This was later implemented as the window to filter the extraneous information or the irrelevant data. The applied change point detection techniques can be seen on the second row of the attached images.

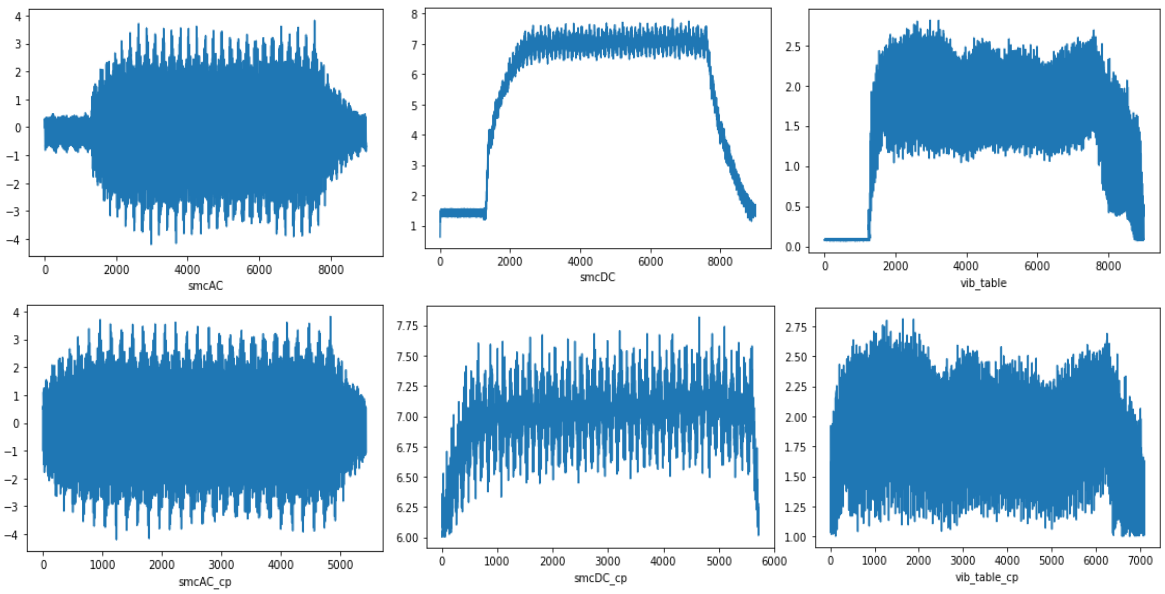


Figure 59 Row 1 - Original vs Row 2 - Cpd

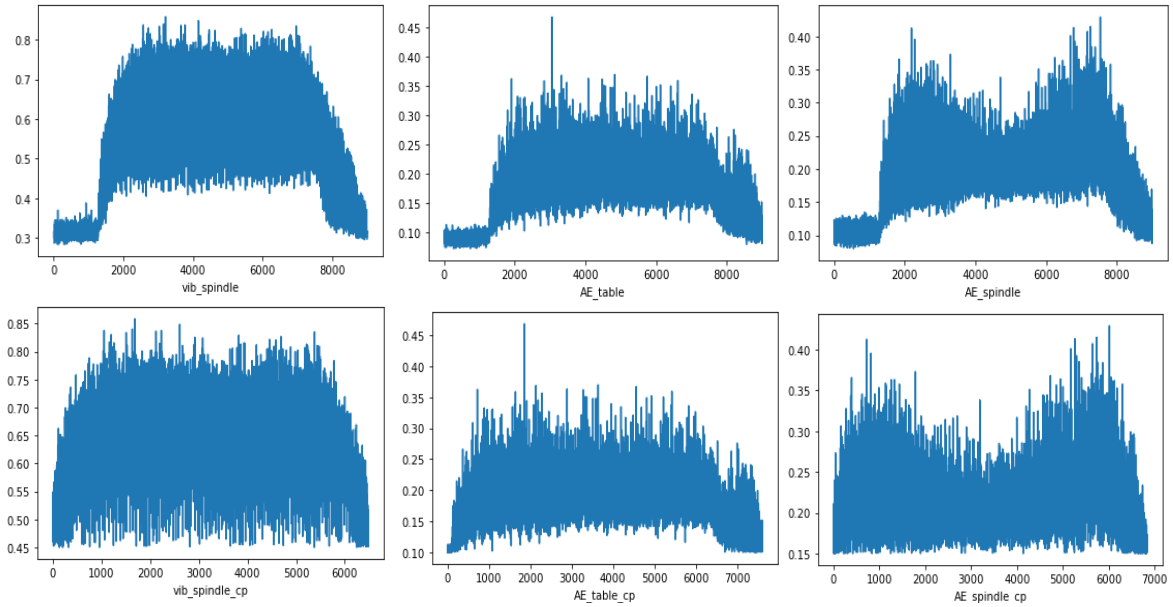


Figure 60 Row 1 - Original vs Row 2 - Cpd

The approach used here for change point detection (cpd) can be taken even a step forward by using an equivalent package for change point detection called **change point** [52] [53] in R (arguably the most robust) or the **rupture** [54] [55] package (library for off-line change point detection) which is native to python. There is yet another cumbersome way around where one could use the **rpy2** [56] package to import the change point function into python and implementing the changepoint detection but as mentioned earlier there are not many examples or works carried out especially pertaining to change point detection.

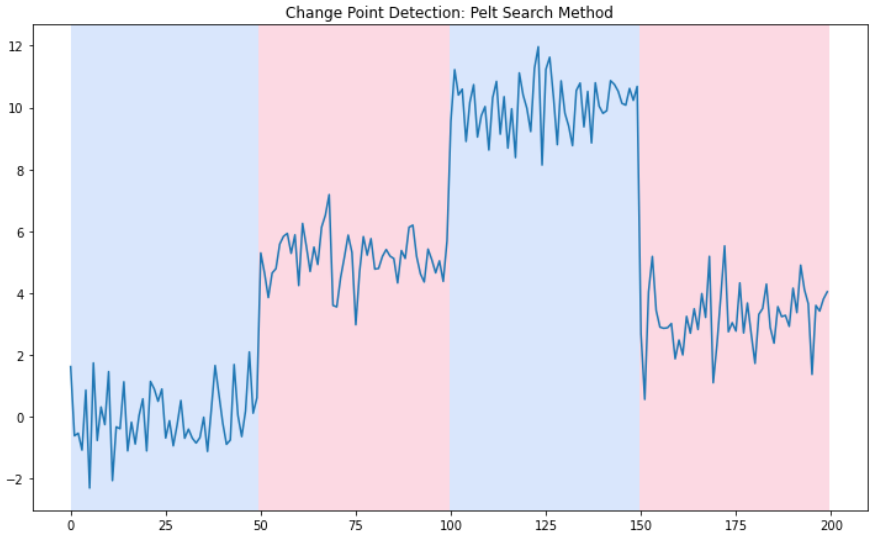


Figure 61 Example of change point detection using python rupture package

Yet another challenge while working on the data set was the lack of information. In our case the data was just quite enough to perform an analysis, but it would be ideal and optimal to have data in abundance whether its relevant to milling operation or any other manufacturing processes. Moreover, having enough data also means we will have a considerate proportion of it reserved for testing set, thereby which the accuracy of the model could be determined or compared later for the purpose of the study.

The results obtained by this study though it's the steppingstone, the prediction tool wear and RUL so obtained, can directly improve the man-machine collaboration in production. A web application could even be developed further to assist the operators in knowing the actual state of the tools and helping them to take better decisions regarding tool changes.

BIBLIOGRAPHY

[1] J. Lee, J. Singh, and M. Azamfar, “Industrial Artificial Intelligence.”

[2] I. S. Candanedo, E. H. Nieves, S. R. González, M. T. S. Martín, and A. G. Briones, “Machine learning predictive model for industry 4.0,” *Commun. Comput. Inf. Sci.*, vol. 877, no. July, pp. 501–510, 2018, doi: 10.1007/978-3-319-95204-8_42.

[3] MathWorks, “What Is Machine Learning? | How It Works, Techniques & Applications - MATLAB & Simulink,” no. May 2016. pp. 1–2, 2018.

[4] GeeksforGeeks, “Supervised and Unsupervised learning - GeeksforGeeks.” 2019, [Online]. Available: https://www.geeksforgeeks.org/supervised-unsupervised-learning/.

[5] M. Vathoopan, M. Johny, A. Zoitl, and A. Knoll, “Modular Fault Ascription and Corrective Maintenance Using a Digital Twin,” *IFAC-PapersOnLine*, 2018, doi: 10.1016/j.ifacol.2018.08.470.

[6] S. Vilarinho, I. Lopes, and J. A. Oliveira, “Preventive Maintenance Decisions through Maintenance Optimization Models: A Case Study,” *Procedia Manuf.*, 2017, doi: 10.1016/j.promfg.2017.07.241.

[7] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. da P. Francisco, J. P. Basto, and S. G. S. Alcalá, “A systematic literature review of machine learning methods applied to predictive maintenance,” *Comput. Ind. Eng.*, vol. 137, no. August, p. 106024, 2019, doi: 10.1016/j.cie.2019.106024.

[8] A. Jimenez-Cortadi, I. Irigoien, F. Boto, B. Sierra, and G. Rodriguez, “Predictive maintenance on the machining process and machine tool,” *Appl. Sci.*, vol. 10, no. 1, 2020, doi: 10.3390/app10010224.

[9] W. J. Lee, H. Wu, H. Yun, H. Kim, M. B. G. Jun, and J. W. Sutherland, “Predictive maintenance of machine tool systems using artificial intelligence techniques applied to machine condition data,” *Procedia CIRP*, vol. 80, pp. 506–511, 2019, doi: 10.1016/j.procir.2018.12.019.

[10] C. Coleman, S. Damofaran, and E. Deuel, “Predictive maintenance and the smart factory,” *Deloitte*, p. 8, 2017, doi:

https://www2.deloitte.com/content/dam/Deloitte/us/Documents/process-and-operations/us-cons-predictive-maintenance.pdf.

[11] "MachineMetrics Health What if you could stop a machine before a catastrophic failure ?"

[12] X. Li, D. Li, J. Wan, A. V. Vasilakos, C. F. Lai, and S. Wang, "A review of industrial wireless networks in the context of Industry 4.0," *Wirel. Networks*, 2017, doi: 10.1007/s11276-015-1133-7.

[13] T. Mohanraj, S. Shankar, R. Rajasekar, N. R. Sakthivel, and A. Pramanik, "Tool condition monitoring techniques in milling process-a review," *J. Mater. Res. Technol.*, vol. 9, no. 1, pp. 1032–1042, 2020, doi: 10.1016/j.jmrt.2019.10.031.

[14] U. Maheshwera, R. Paturi, and S. Cheruku, "Application and performance of machine learning techniques in manufacturing sector from the past two decades : A review," *Mater. Today Proc.*, no. xxxx, 2020, doi: 10.1016/j.matpr.2020.07.209.

[15] M. Calabrese *et al.*, "SOPHIA: An event-based IoT and machine learning architecture for predictive maintenance in industry 4.0," *Inf.*, vol. 11, no. 4, pp. 1–17, 2020, doi: 10.3390/INFO11040202.

[16] R. Sipos, D. Fradkin, F. Moerchen, and Z. Wang, "Log-based predictive maintenance," 2014, doi: 10.1145/2623330.2623340.

[17] Y. C. Liu, X. F. Hu, and S. X. Sun, "Remaining Useful Life Prediction of Cutting Tools Based on Support Vector Regression," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 576, no. 1, 2019, doi: 10.1088/1757-899X/576/1/012021.

[18] J. Gokulachandran and K. Mohandas, "Comparative study of two soft computing techniques for the prediction of remaining useful life of cutting tools," *J. Intell. Manuf.*, 2015, doi: 10.1007/s10845-013-0778-2.

[19] X. Li, "A brief review: Acoustic emission method for tool wear monitoring during turning," *Int. J. Mach. Tools Manuf.*, 2002, doi: 10.1016/S0890-6955(01)00108-0.

[20] Y. Zhou and W. Xue, "Review of tool condition monitoring methods in milling processes," *Int. J. Adv. Manuf. Technol.*, 2018, doi: 10.1007/s00170-018-1768-5.

[21] Q. Ren, M. Balazinski, L. Baron, K. Jemielniak, R. Botez, and S. Achiche, "Type-2 fuzzy

tool condition monitoring system based on acoustic emission in micromilling," *Inf. Sci. (Ny).*, 2014, doi: 10.1016/j.ins.2013.06.010.

[22] H. Li, Y. Wang, P. Zhao, X. Zhang, and P. Zhou, "Cutting tool operational reliability prediction based on acoustic emission and logistic regression model," *J. Intell. Manuf.*, 2015, doi: 10.1007/s10845-014-0941-4.

[23] A. A. Bakir, M. Zaman, A. Hassan, and M. F. A. Hamid, "Prediction of remaining useful life for mech equipment using regression," *J. Phys. Conf. Ser.*, vol. 1150, no. 1, 2019, doi: 10.1088/1742-6596/1150/1/012012.

[24] A. D. Gordon, L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and Regression Trees.," *Biometrics*, 1984, doi: 10.2307/2530946.

[25] "Milling (machining) - Wikipedia." pp. 1–37, 2020.

[26] "Cutting Tool Applications, Chapter 1: Cutting Tool Materials | American Machinist." [Online]. Available: https://www.americanmachinist.com/cutting-tools/media-gallery/21893840/cutting-tool-applications-chapter-1-cutting-tool-materials.

[27] A. Kothuru, S. P. Nooka, and R. Liu, "Application of audible sound signals for tool wear monitoring using machine learning techniques in end milling," *Int. J. Adv. Manuf. Technol.*, 2018, doi: 10.1007/s00170-017-1460-1.

[28] K. Goebel, N. Ames, A. Agogino, and U. C. Berkeley, "Documentation for Mill Data Set," pp. 1–10, 1996.

[29] C. D., "Best Python Libraries for Machine Learning and Deep Learning." [Online]. Available: https://towardsdatascience.com/best-python-libraries-for-machine-learning-and-deep-learning-b0bd40c7e8c.

[30] "Best Python libraries for Machine Learning - GeeksforGeeks." [Online]. Available: https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/.

[31] M. Waskom, "Seaborn: Statistical Data Visualization," *Seaborn*. 2020.

[32] "Scientific Applications — The Hitchhiker's Guide to Python." [Online]. Available: https://docs.python-guide.org/scenarios/scientific/.

[33] TABLEAU, "What is Data Cleansing? Benefits of Clean Data | Tableau." 2020, [Online]. Available: https://www.tableau.com/learn/articles/what-is-data-cleaning.

[34] “Ways to Detect and Remove the Outliers | by Natasha Sharma | Towards Data Science.” [Online]. Available: https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba.

[35] P. P. Ippolito, “Feature Extraction Techniques,” *Computer-Based Design and Manufacturing*. [Online]. Available: https://towardsdatascience.com/feature-extraction-techniques-d619b56e31be.

[36] C. Zhang, X. Yao, J. Zhang, and H. Jin, “Tool Condition Monitoring and Remaining Useful Life Prognostic Based on a Wireless Sensor in Dry Milling Operations,” *Sensors*, vol. 16, no. 6, p. 795, May 2016, doi: 10.3390/s16060795.

[37] W. Caesarendra and T. Tjahjowidodo, “A review of feature extraction methods in vibration-based condition monitoring and its application for degradation trend estimation of low-speed slew bearing,” *Machines*. 2017, doi: 10.3390/machines5040021.

[38] A. I. Moreno, “Data normalization with Pandas and Scikit-Learn _ by Amanda Iglesias Moreno _ Towards Data Science.” [Online]. Available: https://towardsdatascience.com/data-normalization-with-pandas-and-scikit-learn-7c1cc6ed6475.

[39] S. Lhessani, “What is the difference between training and test dataset?,” *Medium*. 2019, [Online]. Available: https://medium.com/@lhessani.sa/what-is-the-difference-between-training-and-test-dataset-91308080a4e8.

[40] Wikipedia, “Mean squared error - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Mean_squared_error#In_regression.

[41] “Model Evaluation Metrics.” [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error.

[42] 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/.

[43] JJ, “MAE and RMSE — Which Metric is Better? – Human in a Machine World –

Medium," *Medium: Human in a Machine World*. p. 1, 2016, [Online]. Available: https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d.

[44] 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.

[45] S. Statistics, "What is Linear Regression?," *2013*. p. 1, 2013, [Online]. Available: http://www.statisticssolutions.com/what-is-linear-regression/.

[46] A. Chakure, "Random Forest Regression," *Towardsdatascience*. p. 5, 2019, [Online]. Available: https://towardsdatascience.com/random-forest-and-its-implementation-71824ced454f.

[47] S. Yıldırım, "Gradient Boosted Decision Trees-Explained | by Soner Yıldırım | Towards Data Science." [Online]. Available: https://towardsdatascience.com/gradient-boosted-decision-trees-explained-9259bd8205af.

[48] Rajat Gupta, "Getting started with Neural Network for regression and Tensorflow." 2017, [Online]. Available: https://medium.com/@rajatgupta310198/getting-started-with-neural-network-for-regression-and-tensorflow-58ad3bd75223.

[49] S. Bhattacharyya, "Ridge and Lasso Regression: L1 and L2 Regularization | by Saptashwa Bhattacharyya | Towards Data Science." 2018, [Online]. Available: https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b.

[50] C. Maklin, "K-Fold Cross Validation Example Using Sklearn Python _ by Cory Maklin _ Towards Data Science." .

[51] R. Shaikh, "Cross Validation Explained: Evaluating estimator performance." [Online]. Available: https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85.

[52] R. Killick and I. A. Eckley, "An R Package for Changepoint Analysis," *J. Stat. Softw.*, vol. 58, no. 3, 2014.

[53] J. K. Lindeløv, "An overview of change point packages in R • mcp." 2020, [Online].

Available: https://lindeloev.github.io/mcp/articles/packages.html%0Ahttps://lindeloev.github.io/mcp/articles/packages.html?q=highest density interval#inference.

[54] C. Truong, L. Oudre, and N. Vayatis, “Selective review of offline change point detection methods,” *arXiv*, 2018.

[55] T. R. Blog, “Tech Rando A Brief Introduction to Change Point Detection using Python.” pp. 1–16, 2020, [Online]. Available: https://techrando.com/2019/08/14/a-brief-introduction-to-change-point-detection-using-python/.

[56] “rpy2 · PyPI.” [Online]. Available: https://pypi.org/project/rpy2/.