CUTTING FORCE PREDICTION BY MACHINE LEARNING

OKAN YÜKSEL FINDIKLI

FEBRUARY 2025

ÇANKAYA UNIVERSITY

GRADUATE SCHOOL
MECHANICAL ENGINEERING DEPARTMENT
MECHANICAL ENGINEERING MASTER'S THESIS

CUTTING FORCE PREDICTION BY MACHINE LEARNING

OKAN YÜKSEL FINDIKLI

FEBRUARY 2025

ABSTRACT

CUTTING FORCE PREDICTION BY MACHINE LEARNING

FINDIKLI, OKAN YÜKSEL
MECHANICAL ENGINEERING MASTER'S THESIS

Supervisor: Assoc. Prof. Dr. Samet AKAR
February 2025, 100 pages

This thesis investigates the application of machine learning methods to predict cutting forces by blending them with simulation data using the finite element method (FEM). It analyzes the prediction capabilities and prediction accuracy using various machine learning methods (neural networks, decision trees). The research part emphasizes the importance of machine learning methods in optimizing process parameters, increasing efficiency, and reducing tool wear. In my study, it was shown that the data generated should be processed before learning and transformed as required in terms of the quality of the data generated and the suitability of machine learning methods. In order to provide the most accurate estimates, processes such as removing unnecessary parameters in the data, normalization and correlation analysis were used. As a result of these, it showed the importance of the relationship between the cutting forces and process parameters of the algorithms, that this relationship can be modeled and applied in production areas for process optimization. At the same time, this research emphasizes the importance of integrating cutting force prediction with machine learning into industrial systems. The results show that expanding the dataset and collecting data from production sites will pave the way for progress in issues such as increased efficiency. During the analysis, workpiece material titanium was selected.

Keywords: Finite element, Neural network, Cutting force prediction, Machine learning

ÖZET

MAKİNE ÖĞRENMESİ İLE KESME KUVVETİ TAHMİNİ

FINDIKLI, OKAN YÜKSEL
MAKİNE MÜHENDİSLİĞİ YÜKSEK LİSANS TEZİ

Danışman: Doç. Dr. Samet AKAR
Şubat 2025, 100 sayfa

Bu tez, Sonlu Elemanlar Yöntemi (FEM) simülasyonlarından elde edilen verileri kullanarak metal kesme işlemlerinde kesme kuvvetlerini tahmin etmek için makine öğrenimi yöntemlerinin uygulanmasını araştırmaktadır. Sinir ağları, karar ağaçları ve topluluk yöntemleri gibi çeşitli makine öğrenimi algoritmalarının tahmin yeteneklerini analiz ederek, çalışma kesme kuvvetlerini doğru bir şekilde tahmin etmedeki etkinliklerini göstermektedir. Araştırma, makine öğreniminin işlem parametrelerini optimize etme, takım aşınmasını en aza indirme ve endüstriyel verimliliği iyileştirmedeki dönüştürücü potansiyelini vurgulamaktadır.

Çalışma, veri kalitesini ve makine öğrenimi modelleri için uygunluğu artırmak amacıyla simülasyon veri kümelerinin ön işlenmesini ve dönüştürülmesini içermektedir. Doğru tahminleri sağlamak için aykırı değer kaldırma, normalleştirme ve korelasyon analizi gibi teknikler kullanılmıştır. Sonuçlar, makine öğrenimi algoritmalarının işlem parametreleri ve kesme kuvvetleri arasındaki karmaşık ilişkileri etkili bir şekilde modelleyebileceğini ve işlem optimizasyonu için eyleme geçirilebilir içgörüler sağlayabileceğini doğrulamaktadır. Sonuçlar, veri parametreleri genişletilir ve üretim sahalarından daha çok veri alınırsa endüstriyel ortamlarda ilerlemenin önün açmaktadır. Tez, Arthur Samuel'in 1950'lerdeki öncü çalışmalarıyla başlayan makine öğreniminin tarihsel gelişimini ve bu alandaki önemli kilometre taşlarını kapsamlı bir şekilde inceleyerek başlar. Bu bağlamda, metal şekillendirme işlemlerinde kesme kuvveti tahmini için sonlu elemanlar yönteminin kullanımı üzerinde özel bir vurgu yapılmıştır. Bu çalışmanın özgün katkısı, makine

öğrenimi tekniklerinin metal şekillendirme alanında nasıl uygulanabileceğini ortaya koymasıdır. Bu, alandaki mevcut literatüre değerli bir katkı sağlamakta ve bu tür yöntemlerin sanayi uygulamalarında pratik yararlarını göstermektedir. Bu sonuçlar, makine öğrenimi yöntemlerinin, veri kalitesi, etik kaygılar ve algoritmik önyargılar gibi potansiyel zorluklarına rağmen, kesme kuvveti tahmininde başarılı bir şekilde kullanılabileceğini göstermektedir. Analizler sırasında iş parçası titanium seçilmiştir.

Anahtar Kelimeler: Makine öğrenmesi kesme kuvveti tahmini, Sonlu elemanlar yöntemi (FEM), Veri analizi, Sinir ağları karar ağaçları

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to everyone who supported me throughout this thesis journey. My sincere thanks go to my advisor. Assoc. Prof. Dr. Samet AKAR, for his invaluable guidance, constructive feedback, and continuous encouragement, which played a significant role in shapig this work. I am also deeply grateful to my wife, Zeliha Selen FINDIKLI, for her unwavering support and patience during this process. Additionally, I extend my thanks to my company for their assistance throughout my thesis period an during the coursework phase of my master's program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

SYMBOLS

F	:Cutting Force
Fx	:Cutting Force in X-Direction
Fy	:Cutting Force in Y-Direction
d	:Depth of Cut
Vc	:Cutting Speed
α	:Rake Angle
γ	:Clearance Angle
E	:Elastic Modulus
G	:Shear Modulus
ε	:Strain
σ	:Stress

ABBREVIATIONS

FEM	:Finite Element Method
AI	:Artificial Intelligence
ML	:Machine Learning
SML	:Supervised Machine Learning
UML	:Unsupervised Machine Learning
CNC	:Computer Numerical Control
KDE	:Kernel Density Estimation
BOW	:Bag of Words
LST	:Linear Strain Triangle
CST	:Constant Strain Triangle
MSE	:Mean Squared Error
ROC	:Receiver Operating Characteristic
AUC	:Area Under Curve

IQR :Interquartile Range

XGBM :XGBoost Machine (Gradient Boosting Algorithm for Large Dataset)

CHAPTER I
INTRODUCTION

1.1. MACHINE LEARNING

Machine learning is followed in almost all industries and areas, which include Banking, Entertainment, Finance, Biotechnology, Education, Gaming, and Marketing sectors. As data is increasing day by day, the scope is also increasing correspondingly in these aspects. This technology allows finding new information and insights from tons of data and generating what would be happening in forthcoming series. This allows the analysis of complex and voluminous data that surpass human capacity, thus paving the way for the integration of machine learning. Machine learning allows the derivation of very accurate forward-looking analyses from data by training algorithms to discover hidden patterns and correlations within complex datasets, as noted by [1].

Sitting as a subfield of artificial intelligence and including a superset of deep learning, machine learning can be the area in which neural networks and deep learning methods lie. Applications in the fields of machine learning become more effective with increases in the amount of data they process. In other words, it can be said that through the interaction that machine learning systems have with users, they are learning continuously to make accurate predictions.

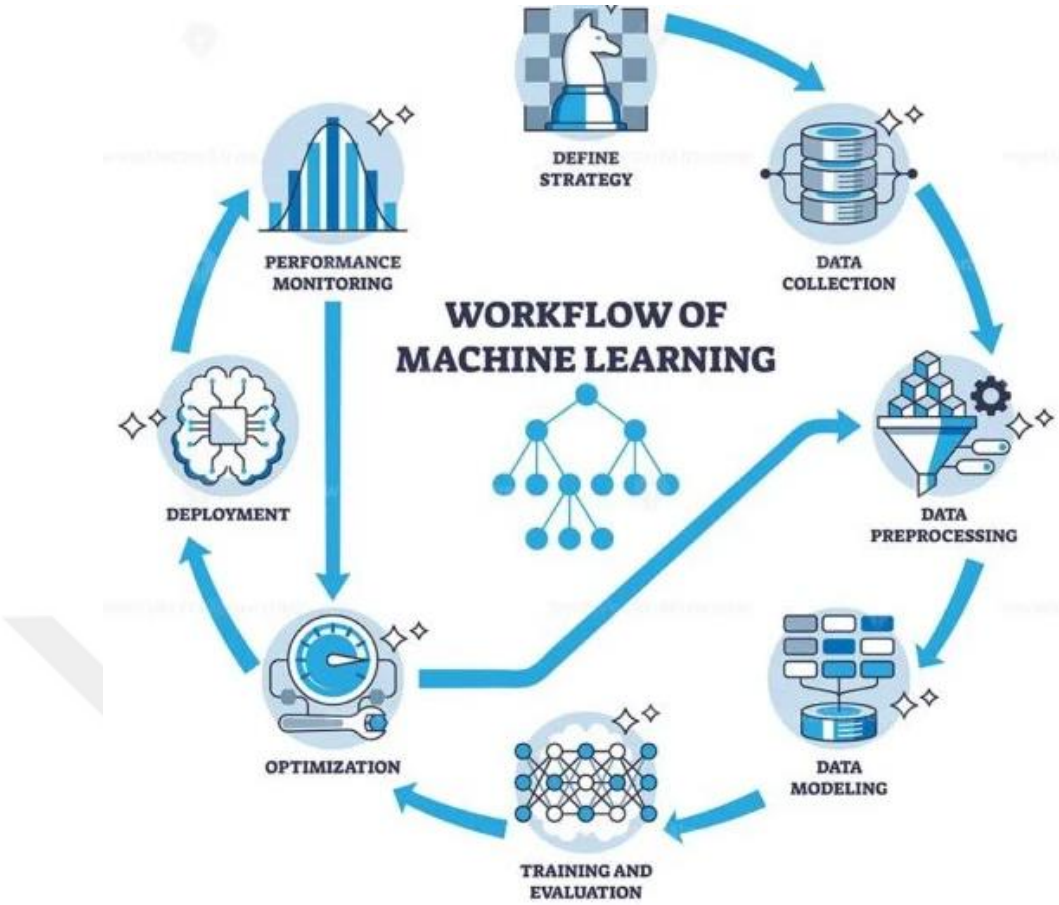

Figure 1: Machine Learning Workflow Diagram

As presented in Figure 1, machine learning is interconnected to deep learning and artificial intelligence. Machine learning is a pre-requisite process for developing artificial intelligence systems. It can process datasets and develop models through the use of the best algorithms. Statistical methods are being used to train the algorithms in this area so it can perform classification and prediction. Rather than becoming less effective due to the increase in the volume of data, machine learning improves the capacity of the model to predict more accurate and more reliable results. Machine learning was initially coined by Arthur Lee Samuel in the year 1959 during the development of a checkers program for IBM. Samuel defined machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed" [2]. In 1962, checkers champion Robert Nealey was defeated by the IBM7094 computer from IBM. This system used a scoring function with different positions on the board that determined the chance of winning and losing for each player. It relied on the minimax algorithm to decide about moves in advance [3]. Samuel enhanced the performance of his program by implementing supplementary

mechanisms; one of those techniques became widely known as "rote learning." The latter refers to the process when a program memorizes all the positions that pieces could have. With the help of a combination of this type of learning and reward function values, Samuel contributed much to establishing machine learning [4]. Although this may be considered modest by today's standards, the achievement nonetheless laid the bedrock for modern artificial intelligence and represented an important milestone in the history of technological development.

In traditional ways of programming, the developer would need to instruct in detail how the data is processed to generate a particular output. This instruction, most of the time, involves conditional structures such as IF-THEN statements, enabling the program to react according to different scenarios, depending on a given condition [5]. On the other hand, machine learning is a completely automated process with limited interference from a human operator. Instead, it deals with the development of algorithms that can per se analyze existing problems on their own and draw insights from past data to make sense. In fact, because of this characteristic, many people, in general, tend to use the terms "machine learning" and "artificial intelligence" loosely and interchangeably. Artificial intelligence can be described broadly as the capabilities of machines to learn new skills and solve problems independently in a manner somewhat similar to human reasoning. Machine learning, however, is an approach toward AI that concerns systems with the ability to learn from data. In this process, algorithms identify patterns through analysis of labeled data examples and then apply what they have learned to new, unseen data. For instance, a machine that has been trained using objects classified as either "Apple" or "Pear" would, after some time and training, be able to classify such objects on its own with no more intervention. Further clarification is given in Figure 2, which shows a relationship that clearly differentiates the two concepts from each other [6].

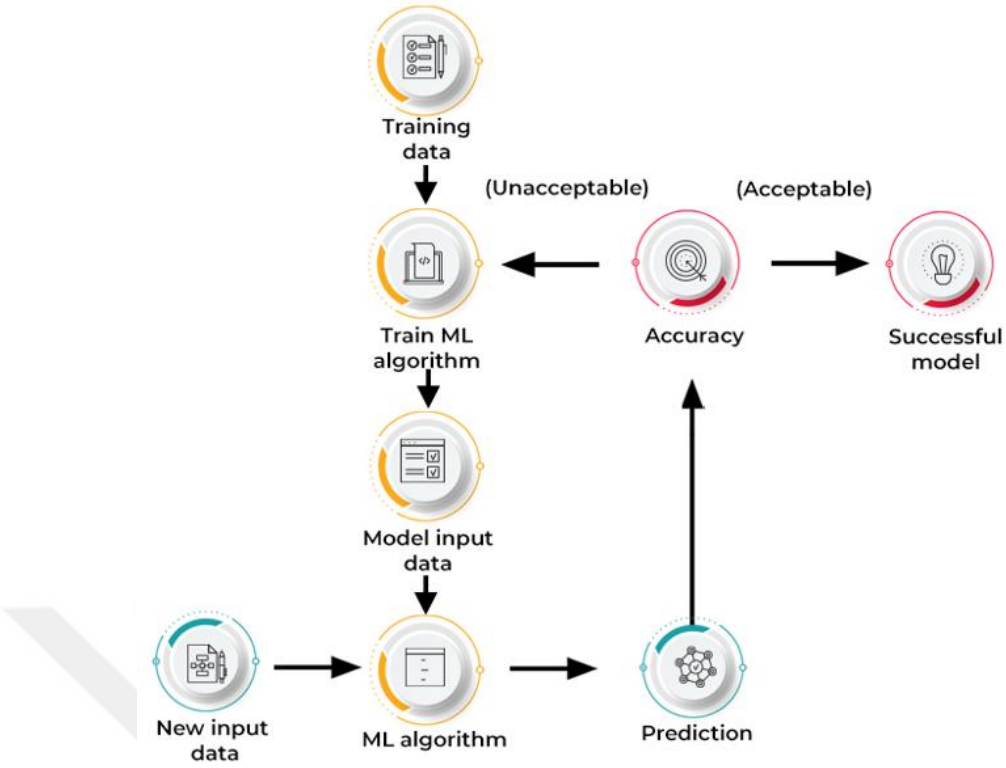

Figure 2: Working Principle of Machine Learning

In standard programming, developers need to inform the user with detailed instructions on how the data should be processed in order to obtain results. These instructions written for informational purposes include some conditional structures such as if-then statements that can process different scenarios according to predefined conditions of the written program. [5]. In fact, machine learning works as an automated process that requires minimal human intervention. Instead of relying on explicit instructions, it involves creating algorithms that analyze current problems and extract insights from past data to function independently [7]. Because of these features, the terms "machine learning" and "artificial intelligence" are often mistakenly used interchangeably [8].

In artificial intelligence, it refers to the ability of machines to grasp new skills and solve problems on their own, similar to how we brainstorm or simply reason. Machine learning focuses on systems that can learn from data. During this process, algorithms analyze labeled data samples to identify patterns and then apply what they have learned to new, unseen data [9]. For example, if a machine is trained to recognize objects as "Table" or "Chair", it can eventually categorize them without needing any additional input. This relationship is further clarified in Figure 2, which shows the distinction between these concepts [6].

1.2. MACHINE LEARNING METHODS

Machine learning consists of several different types of learning. Each of these reveals different aspects of how systems acquire knowledge and the types of algorithms used to acquire this knowledge. The methods used can be divided into: Supervised Machine Learning (SML) Unsupervised Machine Learning (UML) Semi-supervised Learning Reinforcement Learning Of these, the two most frequently used types are supervised and unsupervised learning [10].

1.2.1. Supervised Learning

The learning of the machine with labeled datasets to predict any new and unseen data is called supervised learning. The quality of these predictions depends a lot upon the accuracy and reliability of the dataset used for training. In this case, because of the availability of labels in the training data, it will involve applying techniques from supervised learning. The main aim here is to classify the input data to their corresponding output values belonging to a particular class. The model's predicted results differ from the actual outcomes, something known as the error rate, and the minimization of this error is a key goal of supervised learning. Generally speaking, supervised learning can be classified into two major classes: classification and regression. As put by Cunningham et al. (2008), [11] this approach is hinged on the idea of mapping a set of inputs X to a corresponding set of outputs Y, using that mapping to predict previously unseen data. Its statistical underpinnings are based on concepts such as Risk Reduction, Empirical Risk Minimization, and Risk Boundaries. While unsupervised learning does not have pre-set outputs, in supervised learning the inputs and outputs are dealt with well in advance. Hence, ensemble learning incorporates supervised learning into the model through two basic approaches: bagging and boosting. Bagging involves making sub-sets of the original dataset where each single model is trained on one subset, ultimately combining the prediction of all those models to arrive at improved accuracy. The most famous example of such a bagging algorithm includes the Random Forest. On the other hand, boosting algorithms train iteratively, putting their focus on data points where models in the chain so far have had relatively low performance on, consequently refining each other, and improving the overall performance. These algorithms include AdaBoost, Gradient Boosting Machines (GBM), as well as XGBoost (XGBM). According to Wasilewska & Golenko 2019 [12], in general, the applications of Supervised Learning are common

in a huge number of areas across various disciplines. For instance, Balid, Tafish & Refai (2018) [13] have implemented this for surface water quality classification, while Iwata Ito & Kise (2014) [14] have used it to extract opinions of tourists in their reviews. Not less important is the implementation of supervised learning; for example, in diabetes-based research or in assessing credit risk factors in financial data, as found by Kolukisa et al. (2023) [15].

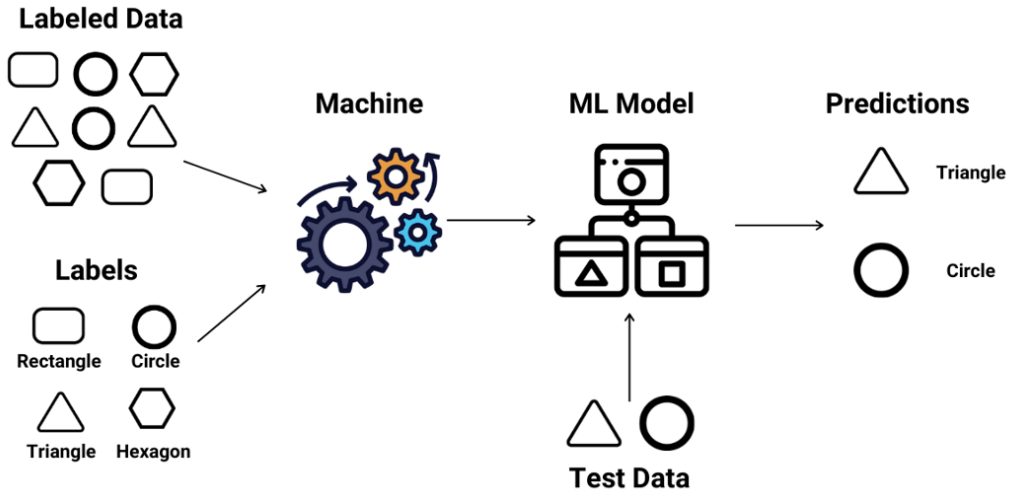

Figure 3: Steps of Supervised Learning [16]

1.2.1.1. Regression

Regression is a statistical method that deals with the analysis of the dependence of one dependent variable on one or more independent variables. In this method, the independent variables are selected looking to estimate their effects on the dependent variable, and the selected independent variables must not be directly related to each other. These independent variables, in turn, influence the dependent variable, which may have a positive or negative relationship with them [17].

Thirdly, is a positive relationship, where, with an increase in the value of the independent variable, the dependent variable rises. In most cases, when more advertisements are done, sales increase, thereby depicting this kind of relationship. On the other hand, a negative relationship occurs where the dependent variable decreases with an increase in the independent variable. This would include increasing product prices that diminish consumer demand [18]. This relationship is represented through regression analysis in some mathematical model that can be used to predict future sets of the dependent variable. This makes regression a powerful tool with wide

applications in disciplines such as economics, psychology, health sciences, and social sciences.

Equation 1: Single Independent Variable Regression Equation [19].

$$y=mx+b \tag{1.1}$$

A linear regression model describes the nature of relationship between one or more independent variables xxx with dependent variable yyy. This means that, in this framework, the dependent variable will be changed or affected by the independent variable or variables. In other words, for every positive or negative increase in xxx, yyy increases or decreases correspondingly. The coefficient, mmm, indicates both the direction (whether plus or minus) and the extent of the influence. For example, if mmm is 0.1, increasing xxx by 10 units will raise yyy by 1 unit.

On the contrary, when mmm is 0, changing xxx would not at all impact yyy in any way. According to Chollet (2021) [20] in this model, the intercept, bbb, is the value of yyy at which independent variables take zero value. Linear regression models make their predictions on a linear basis. Therefore, any outliers that might be there in data make a significant impact on the model. Outliers usually bias the overall performance of the model and come out with untrustworthy predictions [21]. As a result, it is essential to carefully evaluate the dataset for outliers and address them appropriately before applying linear regression [22].

Table 1: Positive Consumer Feedback – Sales Volume

Average User Rating	Average Sales Quantity
5	5000
4	4000
3	3000
2	2000
1	1000

Table 1 depicts a positive trend of "Sales Quantity" featuring "Consumer Rating." Here, the independent variable "Consumer Rating" is the scores assigned by the user from 1–5. The analysis itself suggests that sales are impacted by the ratings.

The products rated 5 out of 5 by previous users have the highest sales, while those rated 1 out of 5 tend to have considerably lower sales [23]. This is further visualized in Figure 15, which gives a clearer graphical representation of the direct positive correlation between consumer ratings and sales quantities [24].

Figure 4: Average Consumer Rating

Figure 4 represents the positive relationship between "Average Consumer Rating" and "Average Sales Quantity." The positive correlation is a form of relationship that indicates when one factor increases, the other one increases as well. This therefore, means that with the increasing consumers' rating, the sales quantities also increases in proportion [25].

Through With the help of mathematical analysis, such data can be furthered into the construction of predictive models for scenarios in the future. Such models offer one the capability to make predictions while providing more in-depth understanding related to the relationship that prevails between dependent and independent variables. A negative correlation between two variables takes place if, with the rise in one, the other shows a decline. A very common example is the relation between the prices of commodities and consumer demand, whereby increased prices mean decreased demand. Indeed, regression analysis is, according to Figure 7, a strong means to identify and make sense of the relationships among variables in a wide array of contexts.

Table 2: Negative Consumer Feedback – Sales Volume

Average Product Price	Average Sales Quantity
1000	5000
1050	4000
1100	3000
1150	2000
1200	1000

This chart illustrates the negative correlation of average prices versus volume of sales between five substitute products. From this analysis, it can be determined that high-priced products reflect lower average sales compared with other products. It is a negative correlation because as product prices go up, the sales quantities go down. This would specifically mean that, because of this relationship, consumers show lower preference for higher-priced substitute products. Such insights provide substantial information to marketers and business strategists in order to optimize pricing strategies and refine market positioning to better meet consumer preferences.

Figure 5: Graph of Negative Consumer Feedback – Sales Volume

Regression analysis can be conducted in many ways, foremost among them linear and nonlinear, multiple regression, each with specific applications depending on the nature of data and relationships concerned. These methods try to describe the effects of the independent variables on the dependent variable through mathematical

models accordingly. It usually finds its application in cases where the relationship between the variables can be modeled as a straight line and thus is suitable for more simple, linear associations. Conversely, nonlinear regression is used when the relations between variables are much more complex and better described by a nonlinear model.

Multiple regression extends this to various independent variables and the effect each has on a dependent variable. However, should multicollinearity occur, or high correlation between independent variables, the model will not be as reliable and can result in misleading interpretation. Logistic regression is another type of regression that is specifically applied when the dependent variable is categorical and often binary in nature, such as yes/no.

1.2.1.2. Naive Bayes

Bayesian data analysis has shown that Naive Bayes, despite its simplicity, can perform exceptionally well in certain contexts [26]. This algorithm independently evaluates the relationships of all variables in a dataset and calculates their probabilities based on the proportions in the overall data. These proportions are then applied to classify the data [27].

This method is effective when used with large datasets under the assumption that variables are independent. Its simple approach and computational efficiency make it a popular choice for applications such as spam filtering and text classification where performance and speed are critical [8].

Equation 2: Bayes' Theorem [11]

$$P(A/B)=\frac{P(B/A)P(A)}{P(B)} \tag{1.2}$$

In probability theory, P(X|Y) represents the probability of event Y occurring, given that an event X has occurred. Similarly, the opposite is true. P(X) and P(Y) represent the independent probabilities of events X and Y, respectively. The Naive Bayes classifier uses these probabilities for classification tasks [28]. Aggarwal and Kaur (2013) compared this theory with other supervised learning algorithms for text data and found it easier to apply. But they stated that it struggles with new or unknown

words, which may affect the approach. Addressing this limitation often requires techniques such as n-grams or embeddings to improve performance [29].

1.2.1.3. Logistic Regression

This algorithm is a widely used method for classifying results. Based on the logistic function, it produces outputs between 0 and 1, which can be interpreted as probabilities. Instead of modeling continuous variables, this method aims to model the probability of a dependent variable belonging to a certain class. It is effective in two-possible-outcome and binary classification problems. Its versatility and reliability have made it a fundamental tool in fields such as medicine, finance, marketing, and social sciences [30].

Equation 3: Logistic Function [31]

$$\frac{1}{1-e^{-x}} \tag{1.3}$$

Classified data can be classified as ordinal or nominal. It consists of categories that do not have any internal order, such as skin color, marital status. However, ordinal data has a defined order, such as education levels. This method is used to analyze such data.

1.2.1.4. Decision Tree

It is a supervised learning application with a tree-like structure to classify data according to predefined boundaries. It classifies the dataset by examining subnodes in more detail. It acts as the starting variable, which we can call the root node at the top.

It branches from the beginning into parent and child nodes representing increasingly more detailed classes of data [32].

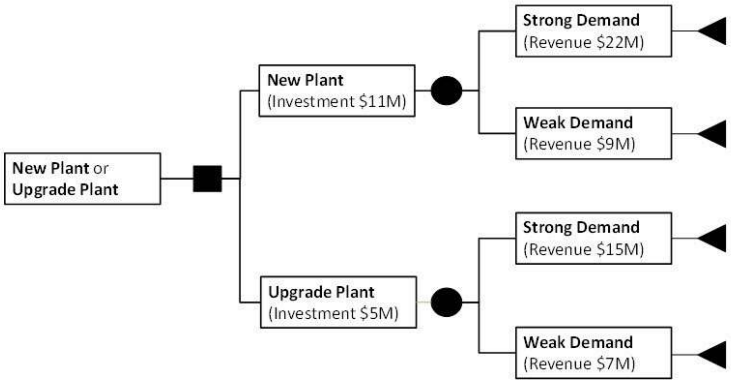

Figure 6: Decision Tree Example

It simplifies the interpretation of complex data sets by translating the relationships between variables into a visually interpretable form. Using mathematical measures, the algorithm determines how to divide the data at each level. This flexibility accommodates the unique characteristics of decision trees and enables them to effectively interpret diverse data sets [33].

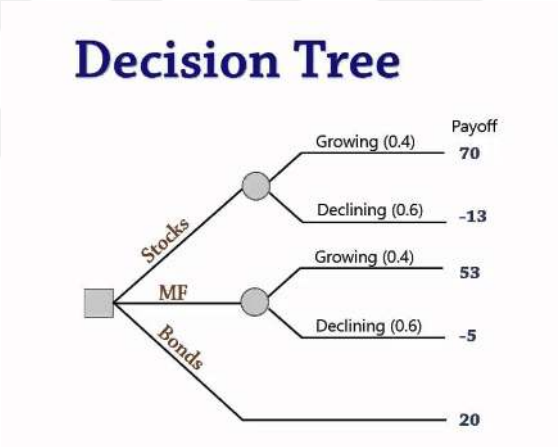

Figure 7: Decision Tree Example

Decision trees visually represent decision-making criteria using “yes” and “no” branches, which increases clarity and usability for decision makers. This transparency makes decision trees important algorithms for analyzing data and making informed decisions [34].

1.2.1.5. Random Forest

It is a method that creates multiple decision trees using different components of the dataset. Decision trees tend to overfit the data when dealing with complex features. Random Forest addresses this issue by training each tree on a random

selection of features. By doing this, it reduces overfitting and can even shine a light on the features that really matter for making predictions [35].

What's neat about Random Forest is its ability to focus on individual features or zoom out and look at a bunch of features together. This gives it a pretty solid grip on complex datasets. The magic behind it all is called "ensemble learning." Basically, it combines the outputs of multiple models (in this case, those decision trees) to make one super-accurate prediction.

1.2.1.6. Gradient Boosting Classifier and XGBClassifier

This method involves running multiple decision trees in a series and gradually improving their accuracy. It works like this: One tree is trained, checks which step it is wrong at, and then builds another tree that tries to correct the errors. This step-by-step scenario continuously improves the model and makes the predictions more accurate over time. In this process, a metric called mean squared error (MSE) is what basically measures how far the model's predictions are from the true values [35].

This formula involves squaring the differences between the actual and predicted values, counting all errors as positive, and then averaging the squared errors. The MSE is susceptible to outliers, meaning that a single odd piece of data can throw off the entire calculation.

Then there's XGBClassifier, which takes Gradient Boosting and gives it an upgrade. It adds L1 regularization, which is a fancy way of saying it helps prevent overfitting. This makes XGBClassifier a go-to choice for handling large, complex datasets. It's efficient, precise, and pretty much a favorite for any task where accuracy is king [36].

Equation 4: Mean Squared Error

$$\frac{1}{\eta}\sum_{i=1}^{n}(g\text{-}t)^2 \tag{1.4}$$

1.2.2. Unsupervised Learning

Unsupervised learning is a fascinating field of machine learning, where the algorithm essentially works without any supervision—hence the name. Unlike supervised learning, which pairs inputs (let's call them `x`) with expected outputs (`y`) to

learn their relationship, unsupervised learning skips the outputs entirely. It doesn't even bother with the reward and penalty system used in reinforcement learning. Instead, it's all about diving into unlabeled data and uncovering hidden patterns or relationships you might not even expect. This can be especially helpful when dealing with messy, noisy datasets [37].

Broadly speaking, unsupervised learning is split into two categories: clustering and dimensionality reduction. Clustering, as the name suggests, groups data into clusters—think of it as sorting a messy pile of socks into neat little bundles. Algorithms like k-means handle this by dividing the dataset into a predefined number of clusters (k). You tell the algorithm how many clusters you want, and it does the rest, organizing your data into meaningful groups.

Dimensionality reduction is another story altogether. Here, the goal is to simplify complex data by reducing the number of dimensions while keeping the most critical information intact. Imagine you've got customer data that spans both time and location. Instead of juggling both, you could focus on just time or location, making the data easier to analyze. This technique isn't just a time-saver; it's also the backbone for clean, effective data visualization [38].

1.2.2.1. Clustering

Now, let's get into clustering, specifically when working with text data. If you've ever tried processing text, you'll know it's not a walk in the park. Documents can be wildly different in length, making things tricky. One workaround is to treat each word as a feature, but honestly, that can get out of hand with large datasets. That's where the Bag of Words (BOW) method steps in as the hero of the hour [39].

The BOW technique breaks text down into its most basic elements. It creates a list of unique words and counts how often each one shows up in a set of documents. Picture this: you're analyzing 20 documents, and you want to figure out which words pop up the most. BOW lets you do that. It's like turning a bunch of messy sentences into something the algorithm can actually work with.

But of course, it's not perfect. One major downside is that BOW doesn't consider word order. So, "cat eats fish" and "fish eats cat" would look the same to the algorithm.

Speaking of order, there's another tool in the arsenal: the **n-gram technique**. This method captures sequences of words (or characters) to provide some context. For

instance, an **n-gram** could be a single word (unigram), a pair of words (bigram), or even a sequence of three words (trigram). Using trigrams for the sentence “Esra Şahin Hacettepe University Computer Engineering” would give you these:

- "Esra Şahin Hacettepe"
- "Şahin Hacettepe University"
- "Hacettepe University Computer"
- "University Computer Engineering"

It’s not a perfect system either—sometimes it gets confused by punctuation or weird word choices—but it’s a step up from plain ol’ word counts.

So, whether you’re clustering customer behaviors or tackling massive text datasets, unsupervised learning has your back. Sure, it has its quirks and challenges, but when it works, it’s like finding treasure in a data-filled haystack.

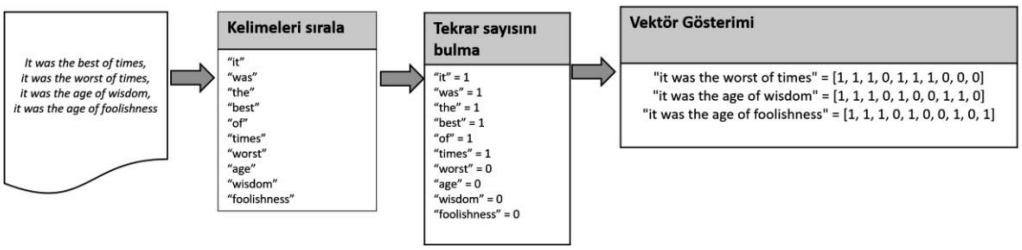

Figure 8: Example of the Bag of Words (BOW) Method

Equation 5: BOW Formula

$$P(w_1^n)=\Pi_{k=1}^{n} P\left(w_k \mid w_{k-n-1}^{k-1}\right) \tag{1.5}$$

1.2.2.2. Dimensionality Reduction

When we talk about "dimensionality," we’re essentially referring to the number of variables or features in a dataset. Think of it as how many ingredients you’re juggling while trying to cook something. Dimensionality reduction is the process of cutting down on those ingredients (features), keeping only what’s absolutely necessary. This can involve techniques like **feature extraction** (creating new, more compact features) or **feature selection** (just picking the best ones). The goal? Simplify the dataset without losing the important stuff [40].

It’s like tidying up a cluttered desk—getting rid of irrelevant papers while keeping the essential documents in sight. This approach is especially handy for huge

datasets, like microarray data, where analyzing everything at once would be overwhelming and inefficient.

1.3. TESTING A MACHINE LEARNING MODEL

Testing a machine learning model is like taking a car for a test drive. You want to see how it performs under different conditions and whether it's balanced or wobbly. This is where concepts like bias and variance come into play.

1.3.1. Bias and Variance

Let's break it down:

Bias measures how far off a model's predictions are from the actual values.

Think of it as the car veering off the road because it wasn't trained properly.

Variance, on the other hand, is about how much the predictions change when you test the model on different datasets. Imagine the car swerving unpredictably every time you drive on a slightly different road [41].

An ideal model would have both low bias (it understands the road) and low variance (it's stable on all roads). But reality? It's rarely that perfect.

If a model has high bias but low variance, it has not learned enough – it is underfitting. Conversely, low bias but high variance means the model has memorized the training data but is struggling with something new – it is overfitting. Balancing the two is the holy grail of machine learning.

1.3.2. Overfitting

Speaking of overfitting, it's practically like passing the test but failing the real test. It looks good on the training data, but fails when it encounters something unusual. This is because the model is very good at memorizing the training data, including all its oddities and noise. Smaller datasets or overly complex models are particularly prone to this problem [42].

- **Get more data**: A bigger dataset reduces the impact of noise.
- **Stop training early**: Don't let the model overthink things.
- **Simplify the model:** Use fewer features to avoid overcomplicating.
- **Regularization:** A technique that penalizes overly large weights for certain features.

1.3.3. Regularization

Now let's talk about how to do the regularization. It's like telling the model to put all its eggs in one basket. It shows the effect of each feature on the model's predictions. If some weights are too big, it means the model's relying too much on specific features, which can lead to overfitting. Regularization helps fix that.

There are two main types of regularization:

1. LASSO (Least Absolute Shrinkage and Selection Operator): This one's pretty aggressive. It can straight-up reduce some feature weights to zero, essentially booting them out of the model. Great for when you've got way too many features.

2. Ridge: Ridge is a bit softer—it doesn't eliminate features entirely but reduces their impact by shrinking their weights. This way, all features still contribute, just less significantly [43].

Both methods work wonders for improving the model's stability and keeping it grounded. But, as always, it's about finding the right balance—overdoing it can strip the model of useful information.

1.3.3.1. Underfitting

Underfitting happens when a model just doesn't "get" the data it's supposed to learn. It's like trying to learn a new language with just the alphabet—you'll miss the grammar and context completely. This typically happens when the algorithm is too simple to capture the dataset's complexity. For example, imagine plotting your training data as blue dots, the model's predictions as red dots, and the perfect fit as a smooth blue line. If the red dots are nowhere near that ideal blue line, you've got underfitting on your hands [44].

If there is insufficient fit, the model will perform poorly on both the training data and the new data that follows. This is like being subjected to a test without preparation, with insufficient information to make accurate predictions.

If we want to fix this, we need to add more layers to the neural networks or try different, more advanced ways of learning. Such adjustments will help it perform better.

Ironically, stopping too early to stop over-learning can backfire and cause under-fitting. If we stop training too early during learning, the model may not have enough time to complete learning. This is actually a delicate balance. While giving the

model enough time to learn, training should not be so long that it starts to memorize noise. [45].

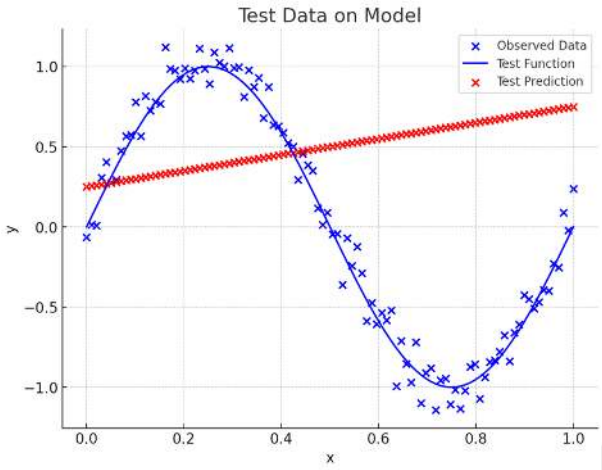

Figure 9: Example of Underfitting

1.3.4. Confusion Matrix, Accuracy, Sensitivity and Precision

The Confusion Matrix is the result of a supervised learning model. It doesn't just tell us the overall score (accuracy); it breaks everything down to show exactly where the model got it right and where it got it wrong. This poses a risk in high-stakes areas like healthcare, where a single mistake can have serious consequences [46].

The Confusion Matrix organizes predictions into four categories [47]:

1. True Positive (TP): Here the model correctly identifies a positive experiment. For example, if it correctly detects the disease in a sick person, it can be called TP.

2. False Negative (FN) (a.k.a. Type II Error): The opposite of TP, the model skips a positive example. If it says someone is not sick when they actually are, it is a FN

3. False Positive (FP) (a.k.a. Type I Error): The model marks something that is not positive as positive, for example, misdiagnosing a healthy person as sick

4. True Negative (TN): The model correctly identifies a negative instance. For example, confirming that a healthy person is indeed healthy

In fields like healthcare, a Confusion Matrix isn't just a fancy table—it's a lifesaver. It helps identify weaknesses in the model, like whether it's too trigger-happy with false positives or too lax with false negatives. These insights guide improvements, ensuring predictions are as accurate and reliable as possible [48].

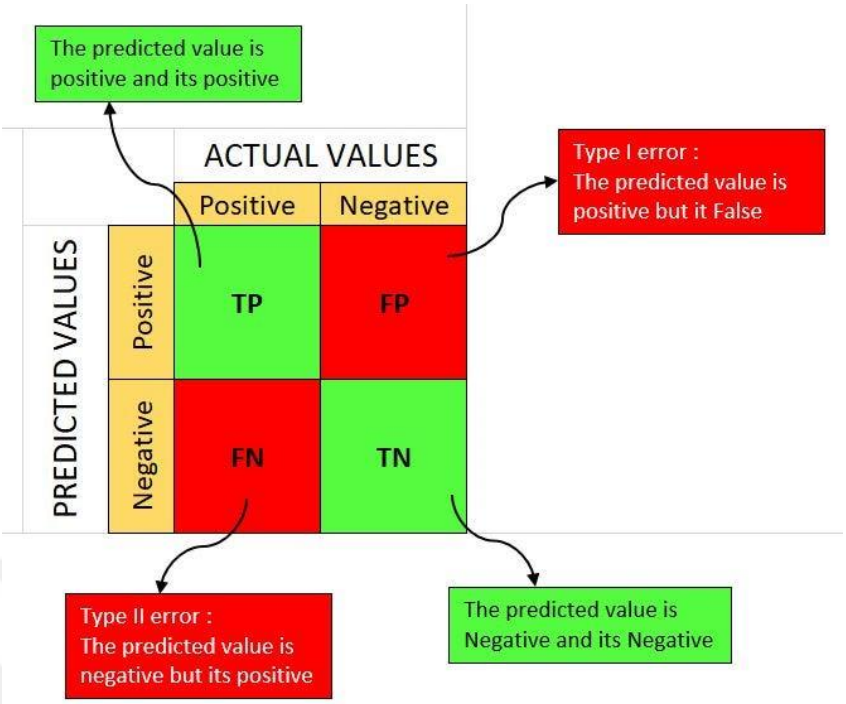

Figure 10: Confusion Matrix

1.3.5. Accuracy

Accuracy is the simplest way to measure how well an algorithm performs. It's calculated by dividing the number of correct predictions (both True Positives and True Negatives) by the total number of samples in the dataset. Basically, it tells you what percentage of the predictions were spot-on [49].

For instance, let's say you're testing a model on 100 individuals, and it correctly identifies 90 people as either positive or negative. The accuracy formula would look something like this:

Accuracy = (True Positives + True Negatives) / Total Samples

In this case, the accuracy would be 90%.

While accuracy gives you a nice general idea of how well the model performs, it doesn't dive into the details. For example, it doesn't tell you whether the errors are skewed toward false positives or false negatives—critical info in certain fields.

Equation 6: Accuracy Formula

$$\frac{GP+GN}{GP+GN+SP+SN} \tag{1.6}$$

1.3.6. Recall

Recall (or sensitivity) zooms in on the model's ability to find the actual positive cases. It's calculated by dividing the number of True Positives by the total number of actual positives (which is the sum of True Positives and False Negatives). Think of it as the model's ability to not miss anything important [50].

Let's break it down with an example:

In a health screening of 100 individuals, the model correctly identifies 20 people with cancer (True Positives) but misses 30 others who also have cancer (False Negatives). The recall would be:

Recall = True Positives / (True Positives + False Negatives)

So in this case: Recall = 20 / (20 + 30) = 40%.

That 40% means the model caught only 40% of the positive cases—a worrying figure in high-stakes fields like healthcare, where missing a diagnosis can be life-threatening.

1.3.7. Precision

Precision, also known as Positive Predictive Value, takes a different angle. It looks at how many of the cases the model labeled as positive were actually positive. In simpler terms, precision measures how trustworthy the positive predictions are [51].

Imagine the same health screening scenario, but this time the model flags 50 people as having cancer. Out of those, 40 actually have cancer (True Positives), while 10 are healthy but misclassified (False Positives). The precision formula is:

Precision = True Positives / (True Positives + False Positives)

For this case: Precision = 40 / (40 + 10) = 80%.

This tells you that 80% of the time, when the model says someone has cancer, it's correct. Precision is particularly important in cases where false positives are a big deal—like in healthcare, where misdiagnosing healthy people can lead to unnecessary anxiety, tests, and treatments [52].

Accuracy, recall, and precision each tell you something different about a model's performance. Accuracy gives you the big picture, but it can sometimes be misleading.

Equation 7: Precision Formula

$$\frac{GP}{(GP+SP)} \quad (1.7)$$

1.3.8. Specificity

Specificity, known as the true negative rate, measures how well an algorithm identifies examples that truly belong to the "negative" class. It tells us how good the model is at avoiding false alarms [53].

For example, let's consider a health test performed on 40 healthy individuals who do not have cancer. If the model correctly identifies 30 of them as healthy (True Negative) but incorrectly identifies 10 as sick (False Positive), the following result will be obtained.

Specificity = True Negatives / (True Negatives + False Positives)

Specificity = 30 / (30 + 10) = 75%.

1.3.9. ROC and AUC Curve

This curve can be thought of as a chart showing the performance of classification models. It serves as a visual way to evaluate how well the model distinguishes between positive and negative classes. The curve basically shows two metrics.:

• True Positive Rate (TPR) (aka sensitivity) on the y-axis shows how many positive events the model caught.

• False Positive Rate (FPR) on the x-axis is calculated as 1−Specificity1 - \text{Specificity}1−Specificity. It tells you how often the model incorrectly classified negatives as positives

Area Under the Curve (AUC) summarizes the performance of the model..

• An AUC of 1 means the model is perfect.

• An AUC of 0.5? Oof, that's no better than flipping a coin [54].

For example, in a cancer detection model, a high AUC would mean the algorithm effectively balances sensitivity (catching actual cases) and specificity (avoiding false positives). This balance is critical in medical settings, where both missing cases and misdiagnosing healthy people can have severe consequences [55].

Let’s not forget that the ROC curve includes a baseline (the diagonal blue line), representing random guessing. If the algorithm’s performance (the orange line) hugs that baseline too closely… well, it’s probably not worth much.

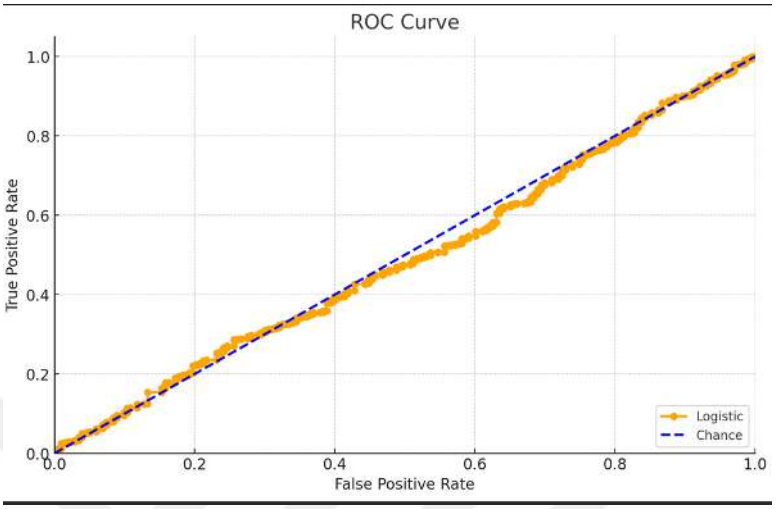

Figure 11: Sample ROC-AUC Curve

1.3.10. F1 Score

The F1 Score is like the Swiss Army knife of performance metrics—it combines Precision and Recall into a single number. Instead of averaging them, it uses the harmonic mean, which ensures that both metrics are given equal weight.

The F1 Score formula is:

F1 Score = 2×Precision×RecallPrecision+Recall2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}2×Precision+RecallPrecision×Recall

For example, let's say an algorithm has the best precision (1.0) but the Recall is only 0.5. If you calculate the arithmetic mean, you get 0.75 but the harmonic mean gives you 0.66—a more honest reflection of the model’s struggles with Recall [56].

This becomes apparent when the dataset is unbalanced. If there are more negatives than positives, metrics like accuracy can be inaccurate and misleading.

Specifity, ROC-AUC and F1-Score provide very good insights into model performance. Specifity keeps false positives under control, ROC-AUC measures the balance between sensitivity and specificity, and F1-Score ensures that neither precision nor recall lags behind. Working together, we can gain detailed insights into model performance.

Equation 8: F1 Score Formula

$$2*\frac{\text{Precision*Recall}}{\text{Precision+Recall}} \tag{1.8}$$

1.3.11. Sentiment Analysis and BERT

Sentiment Analysis is generally related to the emotional states of people on the internet. These shares can be any sharing on the internet.

Since the Web 2.0 boom in the 2000s, sentiment analysis has grown into one of the most exciting branches of natural language processing (NLP). The basic idea? Figure out whether a piece of text is positive, negative, or sometimes neutral. It's like giving computers the power to understand how we feel [57].

A lot of this magic happens thanks to machine learning, and one of the stars of the show is Google's BERT (Bidirectional Encoder Representations from Transformers). BERT is special because it reads text bidirectionally—it looks at sentences both forward and backward to understand context. This might sound obvious, but older models didn't do this—they read text like you'd read a book: left to right. BERT's bidirectional magic means it's fantastic at grasping subtle meanings and nuances that other algorithms might miss [58].

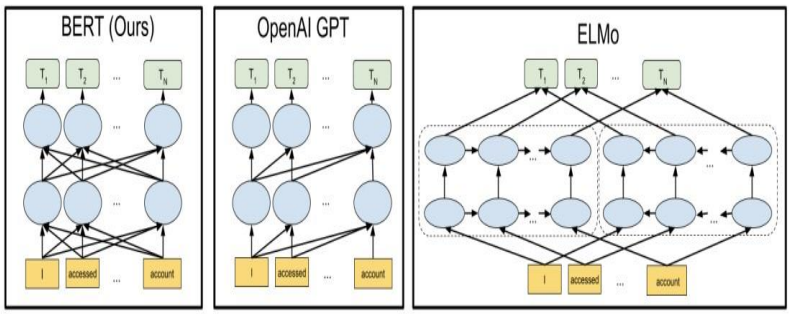

Figure 12: BERT, OpenAI GPT and ELMo

In 2018, Devlin and Chang showed that BERT was a game-changer compared to other transformer models. Unlike OpenAI's GPT-2, which only reads text in one direction, or ELMo, which is sort of bidirectional but not quite as deep, BERT stood out for its robust architecture. Sure, GPT-3 eventually came along with its flashy 175 billion parameters but BERT remains a solid favorite, especially with tools like Hugging Face making it super easy to customize for specific tasks [59].

These days, big companies love using sentiment analysis to figure out what people think about their products. Gone are the days of boring surveys and endless focus groups. Now, brands can scrape social media or user reviews to get instant insights. Say you're running a campaign for a new phone—analyzing tweets, reviews, and blog posts can quickly tell you whether customers are loving it or pointing out flaws [60].

By using sentiment analysis, companies can generate a "satisfaction profile" that paints a clear picture of how their brand is perceived. It's like having a crystal ball for customer feedback, but with a lot more data crunching.

1.4. DATA ANALYSIS WITH PYTHON PROGRAMMING

Python's story begins in 1991 when it was introduced as a versatile programming language. Over the years, it's become a favorite for everything from web development to machine learning. Fun fact: its popularity exploded after 2005, thanks to frameworks like Django, which made building websites a breeze [61].

Although Python is sometimes dismissed as just a "scripting language," that label doesn't do it justice. Sure, it's great for quick scripts and automating tasks, but it's also powerful enough to handle serious, complex applications [62].

Python for Data Science

Python's rise in scientific computing and data analysis is nothing short of remarkable. It's gone from a niche tool to a must-have for data science and machine learning. A big part of its success comes from its simplicity and readability. Let's face it—nobody likes dealing with overly complicated code, and Python makes things refreshingly straightforward.

Debugging: Python's Not-So-Secret Superpower

Python shines in debugging, too. If something goes wrong, it doesn't just crash dramatically (looking at you, C++). Instead, it raises an exception and gives you a helpful stack trace to figure out what went wrong. And sometimes, all you need is a few well-placed print statements to get things back on track—a trick every programmer loves [63].

Python's Libraries and Community

What really sets Python apart for data analysis are its libraries, like pandas for data manipulation and scikit-learn for machine learning. These tools have made Python the go-to language for anyone working with data. And let's not forget its ability to play

nice with legacy systems in C, C++, and FORTRAN, which are still used for heavy-duty computing tasks in research and industry [64].

Solving the “Two-Language Problem”

In many organizations, researchers use one language (like R or SAS) for prototyping and another (like Java or C++) for production. This "two-language problem" is a headache, but Python is bridging the gap. It’s flexible enough for research and powerful enough for production, saving companies a lot of hassle.

That said, Python isn’t perfect. For high-performance tasks, it often relies on lower-level languages like C. But new technologies like Numba, which adds just-in-time (JIT) compilation, are making Python faster and reducing the need for outside help. It’s not perfect, but hey, nothing ever is [65].

1.4.1. Python Programming and Libraries

Python has become the backbone of data analysis, thanks to its robust ecosystem of libraries. Let’s dive into some of the key tools that make Python such a powerhouse for scientific computing and data science [66].

1.4.1.1. NumPy

NumPy is a very important part of numerical computation in Python. If we work with numbers and need more speed, this library is exactly what we need. Some of its useful features are as follows:

- ndarray: A blazing-fast, multi-dimensional array that’s the star of the show.
- Math functions: Important for work that needs to be done in mathematical sequences.
- I/O tools: To read and write data.
- Linear algebra: Mathematical cycles, such as Fourier transforms, are used to solve matrices.
- C API: It allows C program and Python program to work together, which is important for performance.

Compared to Python’s built-in lists and arrays, NumPy is miles ahead in speed and efficiency. So, if you’re crunching numbers, trust me—NumPy is your best friend [67].

1.4.1.2. Pandas

Since pandas hit the scene in 2010, it's been a game-changer for handling structured data. Think of it as Excel on steroids—but way more powerful and programmable. Its two main data structures are:

- DataFrame: A 2D table that's super flexible, like a spreadsheet with extra brainpower.
- Series: A 1D array that acts like a single column in a DataFrame.

Pandas takes NumPy's speed and combines it with database-style functionality, making it perfect for cleaning, slicing, and reshaping data. Need to wrangle messy datasets? Pandas makes it almost too easy [68].

1.4.1.3. Matplotlib

It is used to create graphs in Python. This tool is one of the most important tools for data visualizations. Although it may seem a bit bulky compared to newer libraries, matplotlib is still the preferred choice for customizable and publication-quality visuals [69].

1.4.1.4. Ipython and Jupyter

It lets you tweak and test code on the fly, breaking away from the traditional "edit-compile-run" flow. It's perfect for trial-and-error-heavy tasks like data analysis [70].

1.4.1.5. SciPy

It is like an upgraded version of NumPy. It is used for more advanced scientific calculations. For example:

Integration: Solving differential equations

Optimization: Needed for complex problems

Signal processing: For this type of work.

Together, NumPy and SciPy form a solid foundation for everything from engineering simulations to data analysis [14].

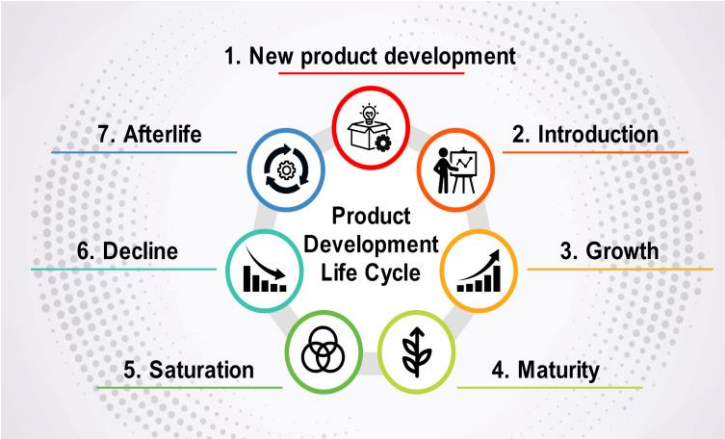

Figure 13: Research Cycle

1.4.1.6. Scikit-learn

It contains many things needed for machine learning. It covers almost everything. Like clustering, changing dimensions, drawing models.

1.4.1.7. Tensorflow and PyTorch

In the world of AI and deep learning, TensorFlow and PyTorch are the reigning champions. By 2022, these libraries had solidified their place as go-to tools for cutting-edge machine learning tasks. Whether you're building neural networks or training complex models, these libraries have you covered [32].

Python's ecosystem is vast, but its true strength lies in how all these tools work together. Whether you're crunching numbers, visualizing trends, or building machine learning models, Python has the tools to get the job done—and then some. Sure, there's always room for improvement (looking at you, matplotlib), but for now, Python's versatility keeps it at the top of the data science food chain.

1.4.2. Data Analysis

At its core, scientific research is about gaining knowledge through a structured process. This systematic journey, often referred to as the research cycle, involves interconnected steps that guide researchers from data collection to insights. Figure 1.1 illustrates this cycle [71].

The Data Analysis Process

Data analysis kicks off with choosing the right dataset—a critical decision that can make or break the analysis. Working with larger data generally means more reliable results. After the datasets are prepared or selected, review and cleaning

processes are performed. Visualization tools like graphs are then used to uncover trends, while modeling helps predict future patterns. Finally, the findings are compiled into reports [72].

1.5. SIMULATION IN METAL CUTTING PROCESS

In industries like automotive, aerospace, and defense, simulating metal forming processes is a game-changer. These simulations let engineers dive into the nitty-gritty of how materials behave under stress, predicting problems before they arise [73].

1. Spotting Problems Early

Stresses and deformations can be simulated without unnecessary and costly trial and error operations, and the design can be changed if necessary based on the results.

2. Boosting Efficiency

Simulations optimize the use of raw materials and tools, slashing costs and improving quality. Imagine identifying energy-saving production tweaks without touching a real machine [74].

3. Testing in Virtual Worlds

Modern software lets you test endless scenarios in a virtual space, skipping costly physical experiments. For instance, during stamping, simulations can refine die designs and ensure proper cavity filling—all before a single machine is turned on.

The Evolution of Simulation Software

Simulation tools have come a long way. Early conferences, like the 1994 “Metal Forming Process Simulation Industry” event, marked the first global effort to standardize computational tools. By the early 2000s, 3D modeling software was taking over, offering remarkable efficiency [66].These tools generally fall into two categories:

• Comprehensive systems: Software like ANSYS.

• Specialized tools: Programs like Deform and QFORM.

ANSYS/LS-DYNA: The Power Player

ANSYS/LS-DYNA is one of the heavyweights in the simulation world. It handles everything from simple bending to multi-layered deformations with ease. But don’t get too excited—it’s not exactly beginner-friendly. The menus and settings can be overwhelming, and running complex simulations often requires a high-powered computer [75].

QFORM: The Specialist

For hot, cold, or warm metal forming, QFORM is the go-to tool. It's incredibly versatile, tackling tasks like forging, pressing, and even electrical metalworking. Need to simulate cooling or bending under gravity? QFORM has you covered. But like any specialized tool, getting the most out of it takes some practice.

1.5.1. Simulation Technologies in Modern Engineering

Simulation technologies have become indispensable in engineering. Industry giants like ANSYS/LS-DYNA and QFORM offer reliable, effective ways to tackle even the most complex problems.

1.5.2. Finite Element Method

1.5.2.1. Introduction to FEM

The Finite Element Method (FEM) has been a cornerstone of engineering analysis since its inception in the 1940s. It all began when Richard Courant introduced the idea of discretizing continuous systems into triangular elements—essentially breaking complex systems into manageable pieces. By 1956, Turner and colleagues refined this by developing the stiffness matrix for triangular elements, and in 1960, Dr. Ray Clough officially coined the term "finite elements" [76].

Fast forward to the 1960s and '70s, FEM was revolutionizing fields like structural analysis, heat transfer, and fluid mechanics. The first practical shell elements (axisymmetric) appeared in 1963, followed by cylindrical and other shell elements in 1969. By the 1990s, FEM had become a go-to computational tool, bolstered by advancements in software and computing power.

Unlike traditional methods, which rely on continuous equations to model systems, FEM breaks objects into smaller, finite elements connected at nodes. It can produce approximate solutions for even complex problems by solving simpler equations. The problem is that it produces problems that are very difficult to solve without using a computer.

This process starts by dividing the system into several areas. Differential equations are used for each area. Variational principles are applied to create a system of algebraic equations. This equation gives us the values we are looking for to solve.

But FEM does not only do this, it also provides flexibility. You can model almost any geometry, no matter how weird or intricate, without oversimplifying shapes into basic geometric forms [77].

1.5.2.2. Benefits and Limitations of FEM

Why FEM is Amazing

It breaks down complex geometries into smaller, easier-to-manage areas, ensuring accurate modeling. Its adaptability allows us to use it for variables. Its strong mathematical and physical foundations also mean you can trust it [78].

But It's Not Perfect

Apart from its strengths, FEM also has weaknesses. It requires high memory and processing power. This method also depends on how well we define the data. If the data is not sufficiently detailed, the results will be far from the truth. Finally, FEM needs validation.

1.5.2.3. Types of Elements in FEM

Of course, different types of engineering problems require different finite element analysis. Some of them are as follows:

3D Beam Element:

Used for three-dimensional structural analysis (a.k.a. space frame elements).

Each node has 12 degrees of freedom (translation and rotation).

Requires input like nodal coordinates, elastic modulus, cross-sectional area, and moment of inertia.

Used for three-dimensional structural analysis (a.k.a. space frame elements).

Constant Strain Triangle (CST) Element:

Connects three nodes with a fixed thickness.

Total of six degrees of freedom.

Stresses are constant throughout, making it suitable for simple stress analyses.

Linear Strain Triangle (LST) Element:

Adds mid-edge nodes for better accuracy.

Each element has six nodes and 12 degrees of freedom.

Captures stress and displacement variations more effectively.

Bilinear Quadrilateral Element:

Perfect for 2D problems.

Four corner nodes with eight degrees of freedom.

Some versions add mid-edge nodes for enhanced accuracy.

Shell Elements:

Ideal for thin structures.

Act like membranes but can't carry vertical loads.

Quadrilateral shells are preferred over triangular ones for better accuracy.

Solid Elements:

Used for modeling 3D solid geometries.

Highly accurate but computationally demanding due to the sheer number of elements involved.

Each of these elements has its niche, allowing engineers to tailor FEM models to specific problems with precision and efficiency.

1.5.2.4. Assumptions and Simplifications in Modeling

Before transitioning to the modeling phase, it is important to briefly discuss certain assumptions and simplifications that will be made during the modeling process [79].

Geometry

The CAD geometry utilized must adequately represent the actual part to ensure accurate analysis.

It is assumed that nonlinear geometric hardening will not significantly influence the system's behavior.

Displacements should remain small enough to ensure a linear solution is valid.

Geometric simplifications may lead to different stress behaviors outside the controlled region, but such errors should not impact the control region.

Only the relevant regions of the part should be modeled, omitting unnecessary areas.

For models using shell elements, the thickness of the part must be significantly smaller than its width and length for the shell idealization to be valid.

The behavior at the junctions of thin surfaces is generally not critical unless within the control region.

To satisfy beam theory, the structure must be sufficiently slender and long.

Aesthetic features that do not affect structural integrity can be disregarded in the model.

Mass variations in the structure due to various effects can be neglected.

If the forces acting on the structure are symmetric about a particular plane, or can be assumed as such, the model can be simplified accordingly.

Material Properties Different materials exhibit different responses under identical conditions. Steel, the most commonly used material in engineering, can produce varying outcomes depending on its alloy composition and the thermal treatments it undergoes. To conduct an accurate and realistic analysis, the material used must be precisely defined. Materials can be isotropic, anisotropic, or orthotropic. Isotropic materials exhibit the same elastic properties in all directions and cross-sections, independent of geometry. Anisotropic materials, on the other hand, have different elastic properties in different directions. Orthotropic materials, a subset of anisotropic materials, possess different elastic properties along mutually perpendicular planes. Most analyses are conducted under the assumption that the material is isotropic and homogeneous. Generally, three fundamental properties of the material are specified: Elastic Modulus (E), Shear Modulus (G), and Poisson's Ratio (ν).

Boundary Conditions

Displacements should remain small enough to ensure the magnitude, direction, and distribution of the load remain constant throughout deformation.

Frictional losses within the system can be considered negligible.

Symmetry can be exploited to reduce and simplify the model.

Connection Elements

Stress concentrations arising from assembly methods such as bolts and welds can be neglected.

Potential loosening of connections over time can be ignored.

Connection areas are often considered as perfect connections.

All welds in the system are assumed to be ideal and flawless.

Any potential damage to connection elements can be disregarded.

1.5.2.5. Considerations in Preparing a Finite Element Model

For the analysis results of the components to be modeled and studied to closely approximate real values, the modeling process itself must be as accurate as possible. There are several key considerations to keep in mind during the modeling phase. First, the mesh structure of the real part should be as uniform and regular as possible. However, in regions where the loading conditions appear critical, a denser mesh may

be permitted. Quadrilateral elements tend to provide more accurate results compared to triangular elements; thus, the use of triangular elements should be minimized. If their use is unavoidable, large-angle triangular elements are recommended. Additionally, overly elongated shell elements should be avoided, and the aspect ratio of element edges should not exceed 1/10 for displacement calculations and 1/5 for stress analysis. For higher-order elements, the distribution of intermediate nodes should be as regular as possible.

If the structure or system being analyzed is symmetric, this symmetry can be leveraged to simplify the modeling process. Instead of modeling the entire structure, only a portion of it needs to be modeled. It is also important to note that irregularities may arise at the junctions where different element types are connected, potentially leading to unexpected results in these regions. Therefore, the sizing of elements used in the model must be handled with great care, ensuring smooth transitions wherever possible. In discontinuous dynamic analyses, there should be consistency between element size, time step, integration method, and vibration duration. Displacement analysis generally requires a less dense mesh structure than stress analysis. However, in analyses that account for geometric and material non-linearities, a denser mesh is necessary. Calculating vibration modes requires a finer mesh than what is needed for determining natural frequencies. In dynamic analyses where high-frequency response values are not critical, the mesh structure used in static analysis can be applied [80].

For anisotropic materials, the Poisson's Ratio should be explicitly defined, and it should be checked whether the theoretical limits of the Elastic Modulus (E) and Shear Modulus (G) are exceeded. In the analysis of complex systems, the entire structure is often initially analyzed with a relatively coarse mesh. The results of this analysis are then used as boundary conditions for a more detailed analysis of the specific region of interest, which is performed with a finer mesh. In this study, vibration analyses conducted using the ANSYS software package considered the above factors when meshing Circular Axis Fixed Section, Parabolic Axis Fixed Section, and Parabolic Axis Variable Section beams. In particular, large-angle triangular elements were used in regions where quadrilateral elements were necessary, particularly contributing positively to the analysis of the Parabolic Axis Variable Section beam when fixed support boundary conditions were applied.

CHAPTER II

OPTIMIZING CUTTING FORCE PREDICTİON USING MACHINE LEARNING: FEM SIMULATIONS, EXPERIMENTAL DATA, AND MODEL ENHANCEMENTS

2.1. OPTIMIZING CUTTING FORCE PREDICTION

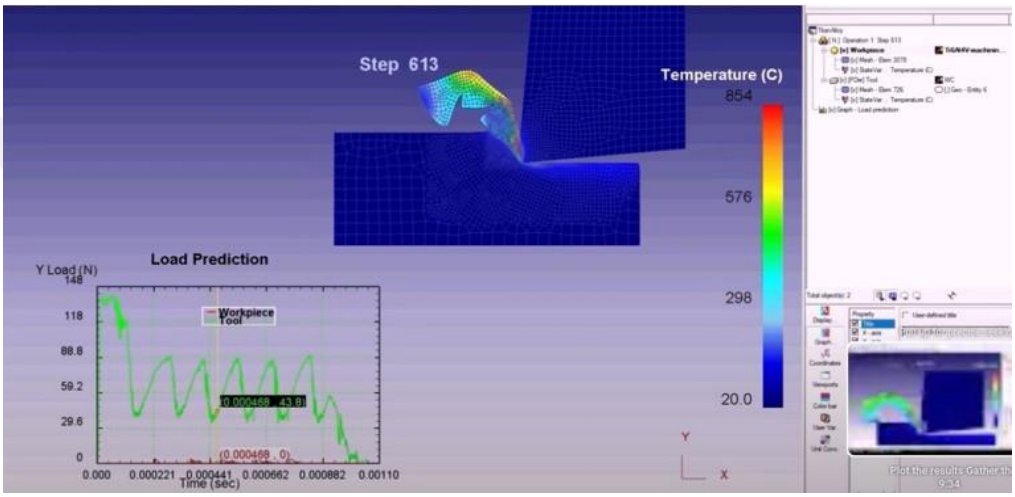

Figure 14: Machining simulation example.

Readily available datasets were also explored, unfortunately, they are limited only to experimental, and not simulation data. Choice was made on the publicly available dataset made out of CNC turning samples. The dataset consisted of various roughness values, accompanied bycutting force components.

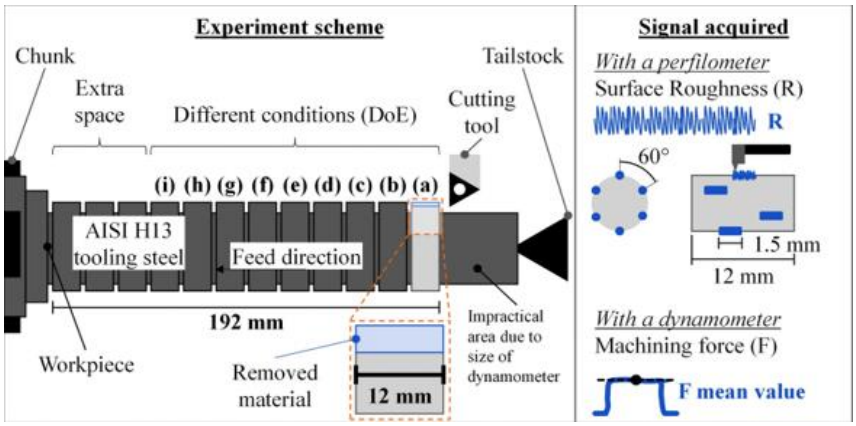

Figure 15: CNC turning dataset measurement acquisition

Scope:

- After the initial assessment, the scope of the project was defined as follows:
- Develop a data parsing code to extract datasets from their input format.
- Organize the available project code and simplify the folder structure. Document theexisting modules.
- Train a machine learning model on both simulation and experimental dataset. Achievesatisfactory accuracy on the test dataset (which will be split from the original data).
- Create an evaluation and visualization framework.
- Generate and deliver a detailed report on the process and outcomes.

Data mining:

Input simulation dataset consisted of 10 folders and a tabular data sheet. Folders contained all theoutput of the executed simulations, with databases files and various exports. Temperature data was contained in .MSG files. Tabular data contained following attributes *Depth Of Cut*, *Feed Rate*, *Length Of Cut, Nofsimnum*, *Step No.* and *Time*. All the simulation .MSG files were parsed and a custom code for extrapolating relevant temperature information was created. Tabular data was also imported, and several redundant features that do not add information were identified: *Time* (as it represents only the time of of the simulation step), *Step No*., (also does not contain information and it is only applicable in the simulation environment), and finally, *Nofsimnum*, as it consists only of 1s and therefore does not contain information. Mentioned features were removed, and temperature data was merged with the tabular data. The rest of the features from the tabular dataset were not deemed particularly useful, as they do not provide significant amount of information. It was decided to leave them in the dataset nonetheless, as their contribution is more than 0. Temperature data was not available for all the tabular data entries, soa lot of those had to be removed. Raw simulation dataset consisted in the end of 1103 entries.

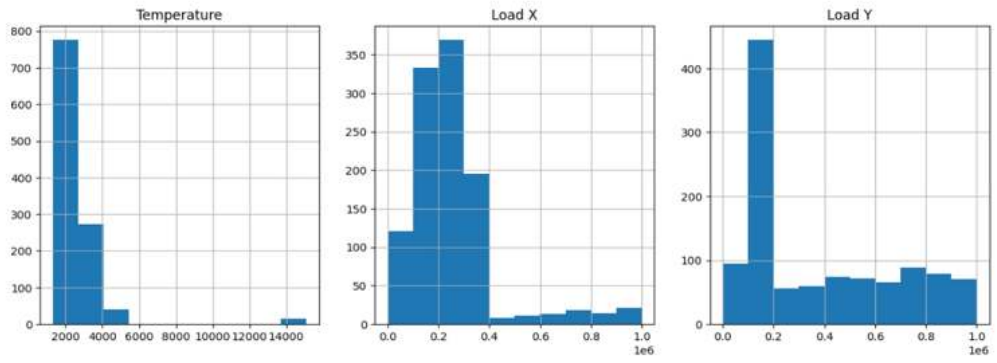

Figure 16: Raw distributions of simulation dataset Temperature feature and output variables.

What can be see from the figure 3, is that both the Temperature feature and the output variables have skewed distributions, which could cause problems during training process, as it would hinder models ability to generalize. To mitigate that, a BoxCox transformation was applied on the data, to achieve a more Gaussian distribution. Furthermore, outliers had been removed from the dataset using both Interquartile Range (IQR) method (3 iterations) and Z-score method (1.75threshold). Finally, data had been subjected to min-max normalization, to make it applicable for training a regression model.

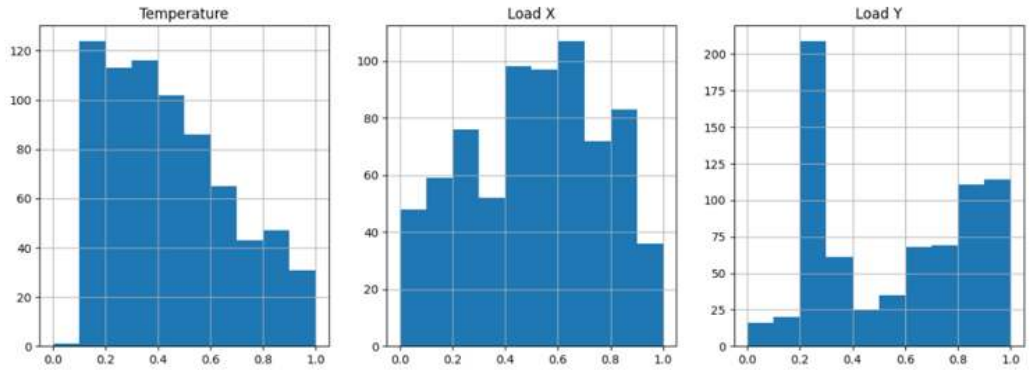

Figure 17: Cleaned, transformed and normalized distributions of simulation dataset Temperaturefeature and output variables.

Interdependency among Temperature feature and output variables was further explored by theKernel Density Estimation (KDE) plot.

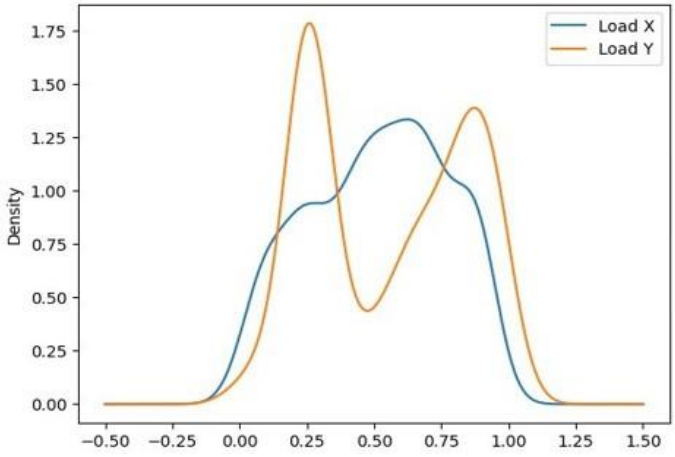

Figure 18: KDE plot of the Temperature – Output variables relation

Finally, correlation matrix was computed and visualized as a heatmap, for all the features andoutput variables.

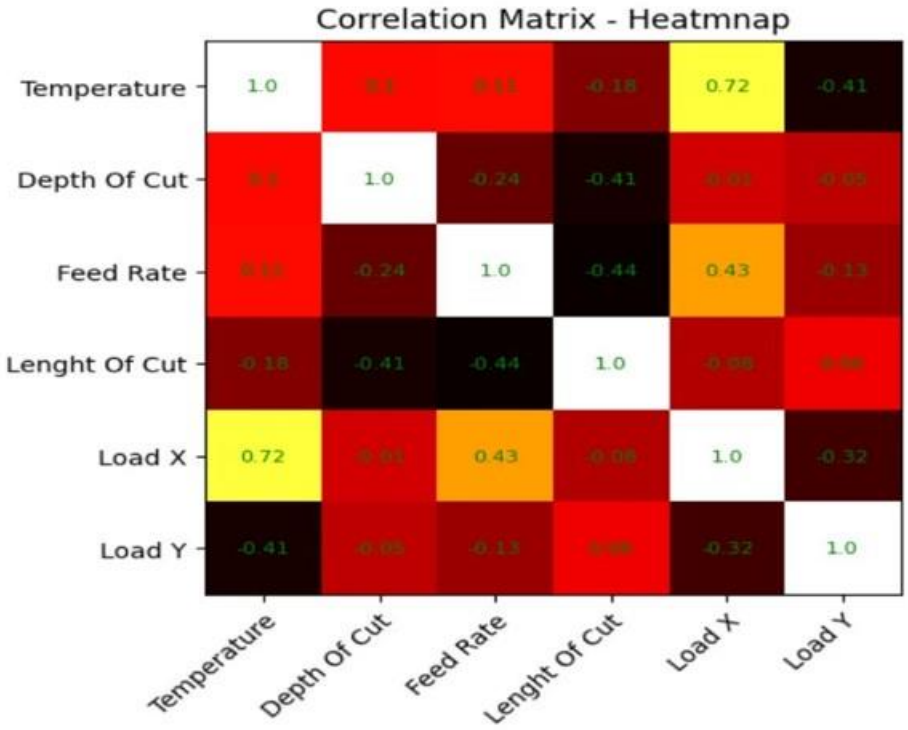

Figure 19: Correlation matrix of the simulation dataset.

What can be concluded from the conducted data mining procedure for the simulation dataset, is that the initial distributions had been greatly improved. Temperature feature is, as expected, mostdescriptive. The issue remains however, that Load Y is still not normally distributed, which is also nicely reflected in its correlation

coefficients: Temperature to Load X has a correlation coeff. of 0.71, and Temperature to Load Y has -0.41.

Experimental dataset, based upon roughness data, was consisted of 2 independent experiments, in total consisting of 612 entries. Upon observing the initial status, it had been decided to pursuethe model that would be capable of predicting resultant force (*F*) from the values of the following roughness parameters: *Ra*, *Rz*, *Rsk*, *Rku*, *Rsm* and *Rt*. The rest of the available featureswere removed. Due to the fact that the experiment measured roughness and cutting forces in discrete steps, there was a lot of duplicated data. After the removal of duplicated entries, the remaining dataset consisted of only 102 entries.

The dataset had been subjected to the IQR outlier removal (single iteration) and simple logarithmic transformation, to achieve a more normal distribution. Finally, the data had beennormalized. Following plots are showing the status after the described preparation steps.

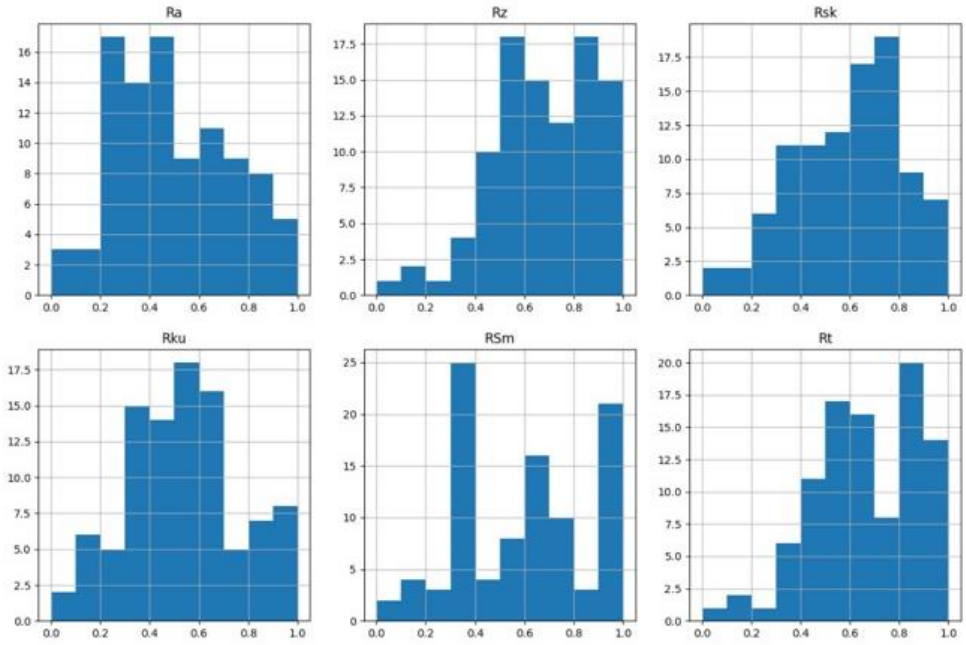

Figure 20: Distribution of the roughness dataset features – after cleanup and transformation

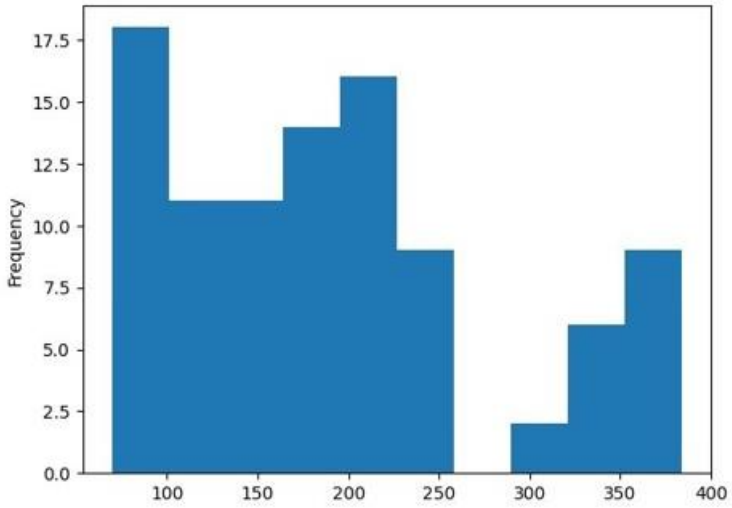

Figure 21: Distribution of the roughness dataset output variable (F) – after cleanup and transformation.

Finally, a correlation matrix had been plotted to access the descriptive power of each feature,visible on figure 22

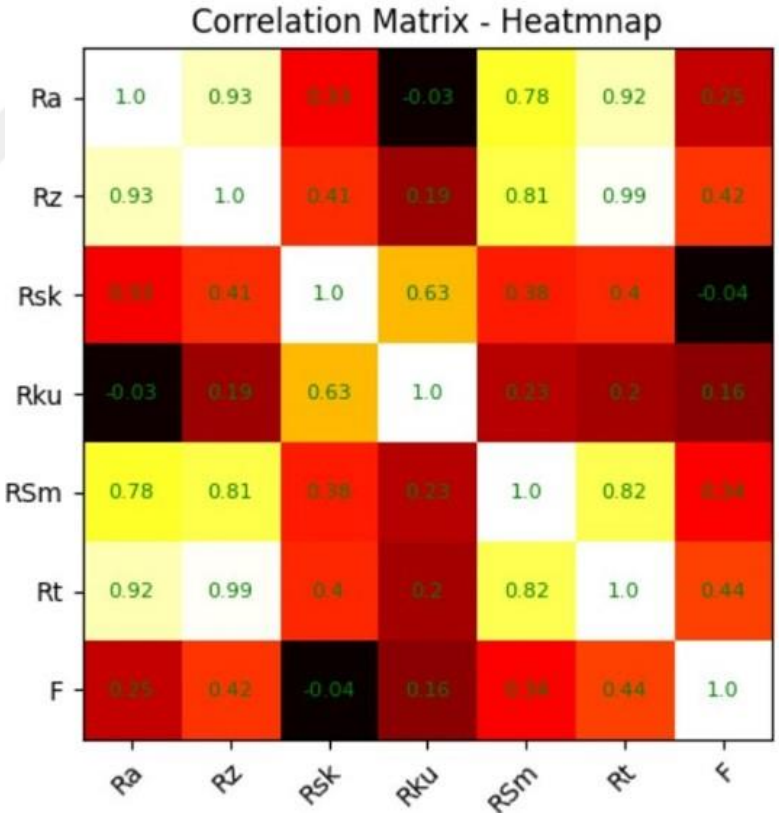

Figure 22: Correlation matrix of the roughness dataset.

Due to the lack of descriptive power of *Rku* and *Rsk* features, it had been decided to removethem from the training dataset.

Model training and evaluation:

First conducted experiments on the simulation dataset pinpointed the main challenges, which is the lack of descriptive power of the Temperature feature, and in general, small amount of available data. Therefore, multiple different architectures were tested, varying from simple MLPs(up to 128 neurons in a layer) to encoder-decoder style neural networks, all with varying results. Experiments were conducted with dropout layers, different activation functions (CELU, ReLU, Softplus), loss functions (Huber, L1, MSE), batch normalization, different gradient optimizers and batch sizes. Putting weight on the features was also attempted, but it caused explosion of validation loss. In total, roughly 50 experimental models were trained. Best results were achievedfor a classical MLP 8x16x32x64 architecture, with Adam optimizer, LeakyReLU activation function (with 0.02 slope), batch size of 8 and adaptive learning rate (0.0001 until 0.18 epochloss and 0.00001 afterwards). MSE loss was used, and it converged at 0.013 MSE.

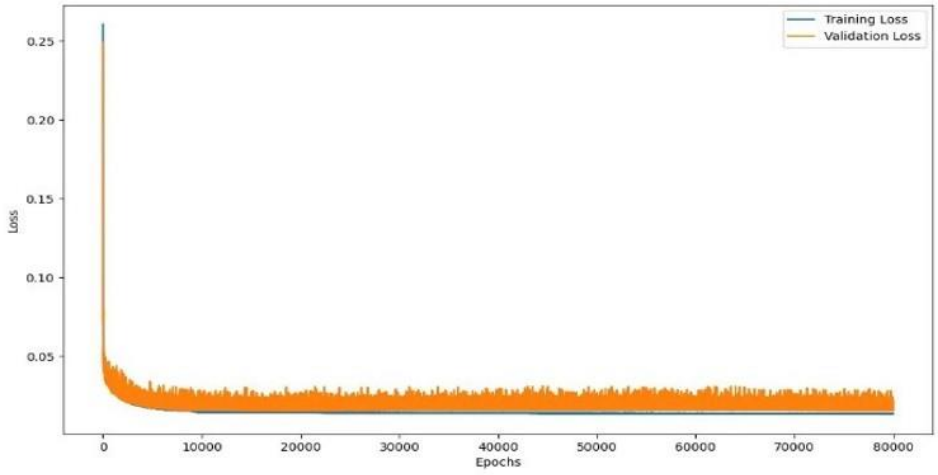

Figure 23: lr_1_e_m4_b_8_e_80000_32_64_LreLU_2me2_addaptive_lr loss plot.

The model example inference is given on the figure below, while it achieved average Mean Average Error (MAE) for Load X predictions of 7123.957 N and 113864.016 N for Load Y predictions.

```
Predicted X Load:, 286862.03
Predicted Y Load:, 117355.17
Actual X Load:, 288796.12
Actual Y Load:, 121823.04
Error X: 1934.0938
Error Y: 4467.867
-----------------
Predicted X Load:, 281215.56
Predicted Y Load:, 84143.57
Actual X Load:, 280669.12
Actual Y Load:, 106769.03
Error X: 546.4375
Error Y: 22625.46
```

Figure 24: lr_1_e_m4_b_8_e_80000_32_64_LreLU_2me2_addaptive_lr average inference.

Similar experiments were done on the roughness dataset side as well. Finally, best results were achieved for a rather small MLP model (4x8x16 architecture), with ReLU loss function, MSE loss and Adam optimizer, also with a adaptive learning rate (0.0001 before 0.085 threshold, and 0.00001 after). It was trained on a batch size of 8, for 8000 epochs.

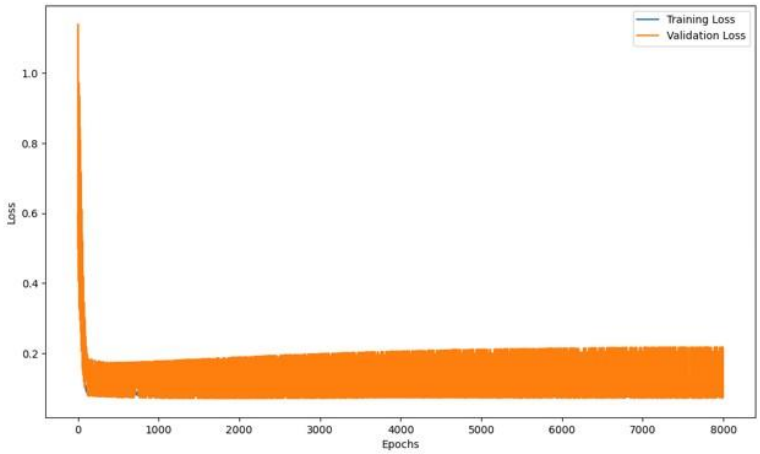

Figure 25: lr_1_e_m4_b_8_e_8000_roughness_adapt_lr loss plot

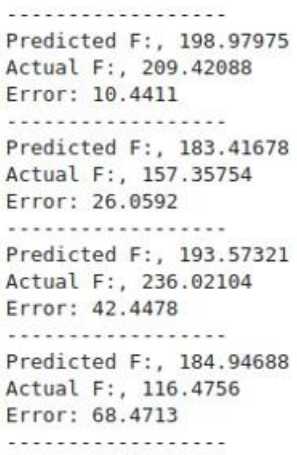

Figure 26: lr_1_e_m4_b_8_e_8000_roughness_adapt_lr average inference.

Model achieved average MAE of 64.6501 N.

Executed work achieved agreed milestones and produced multiple models with targeted performance, as well as robust data mining and pre-processing pipeline, capable of adapting to new datasets and ensuring the further progress of the project. Work also established a solid development framework for further development of machining cutting force prediction models, while the achieved results demonstrated its capability to extract the maximum out of available data. Resulting architectures and approaches correspond with state of the art.

There is room for improvement. Simulation datasets would be preferably extended to include at least one more state variable, or to include exact temperature measurements (in discrete points). In this way, more variability would be included into the dataset, leading presumably to better results. Roughness dataset would have to also be extended with extra measurement data in more intermediate steps, and perhaps with the inclusion of cutting tool radius (as a continuously changed variable). This all would have to be aligned with the end targets of the project, as the final goal is to presumably produce a production ready model, capable of making real time predictions on the CNC parts itself. A unified temperature – roughness (or tool radius roughness) model could also be a good solution, as both of the variables are possible to be measured during the machining process itself A decision of what parameters are actually available for measurements is therefore vital. Further improvements are possible on the model architecture as well, as more elaborate experimental framework could lead to a more optimized architecture and combination of hyperparameters, leading to superior performance.

Improvements – additional accuracy of simulation predictions Following the internal discussions at the client site after the submission of the first project deliverables, it was requested to increase the accuracy of the model. To tackle the issue, it has been decided to increase the size of the dataset by creating additional simulations. Literature survey and previous consultations with software supplier revealed an option of point based temperature measurements. Those were to be extracted in 7 distinctive points in each simulation (1 in the tool and 6 in the workpiece, number P1 to P7), as visible on the figure below.

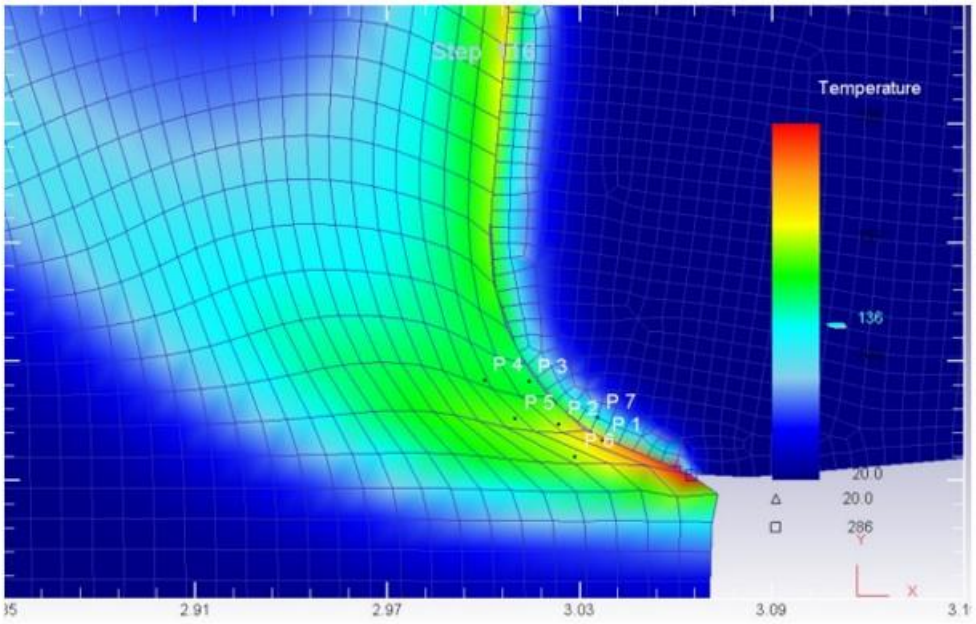

Figure 27: Point Measurements

New point measurements were extracted for all the old simulations. New simulations were also created. To conduct this, a DoE approach was used to define multiple simulation experiments, varying different process parameter factors, given in the table 1. Extra measurements were done for all the simulations & merged with the existing data (workpiece Temperature, Depth Of Cut, Feed Rate, Length Of Cut). Temperature data, although an output from simulations, were to be used as an input to ML model. Data mining framework had also been extended to accommodate for the new point based temperature measurements, which are now exported in a .TXT format. A parser code for the new files had ben written, isolating row by row of temperature data. Environmental temperature had been removed, as it was constant, and therefore hadn't provided any information that was of predictive value. Newly extrapolated

point temperatures had been merged with the previously generated data, which consisted of global temperatures and values of process parameters. A largest overlap between all three sets was found, result in a updated dataset of 1001 entries.

New dataset was also subjected to the same cleaning (outlier removal) and transformation (Box Cox) procedures, described in the previous “Data Mining” chapter, all with the goal to achieve better correlation between input features and output Load X and Load Y variables. It shall be emphasized that only new point based temperatures had been subjected to Box Cox transformation, while all the variables (both input and output ones) had been subjected to outlier removal. Various thresholds had been used for Z-score outlier removal, given in the table 2. Data was in the end normalized (min-max normalization).

Variable	Threshold
P1	2.0
P2	2.0
P3	2.0
P4	2.0
P5	2.0
P6	2.25
P7	2.0
Temperature	2.0
Load X	1.5
Load Y	1.5

Z-score outlier removal threshold parameters.

Following figures visualize the state of the dataset after the outlier removal and all the transformations.

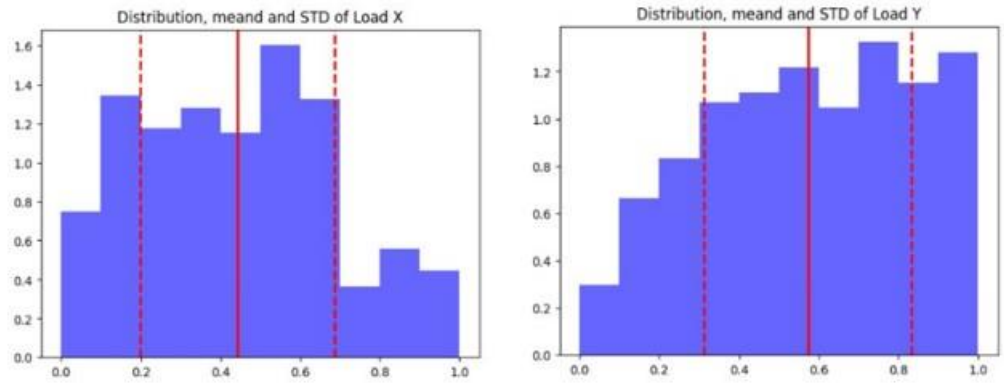

Figure 28: Distribution, mean and standard deviation of the output variables – after cleaning.

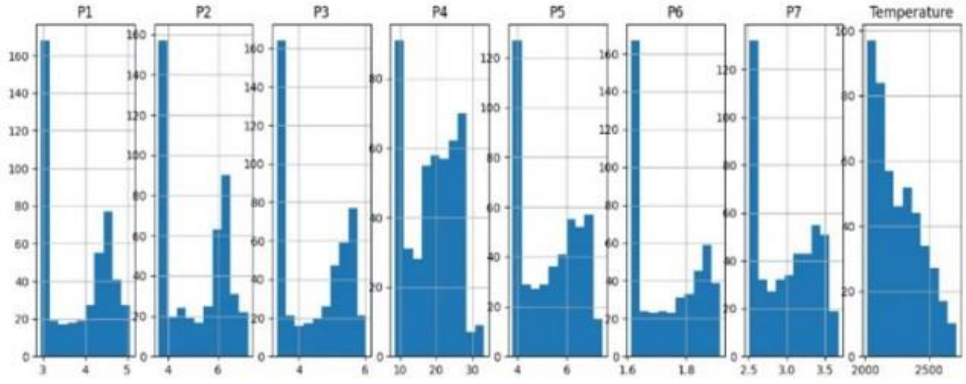

Figure 29: Distribution of the temperature features – after cleaning and transformations the features have on the output variables. It was done fitting a linear regression model.

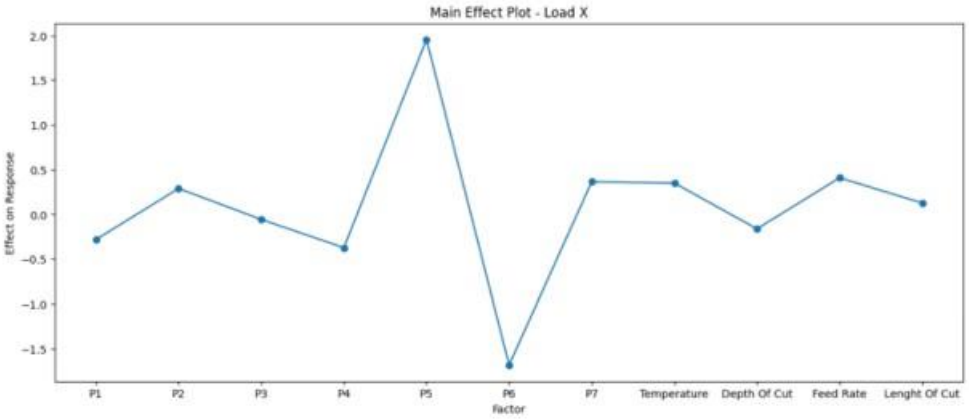

Figure 30: DoE effect plot – Load X.

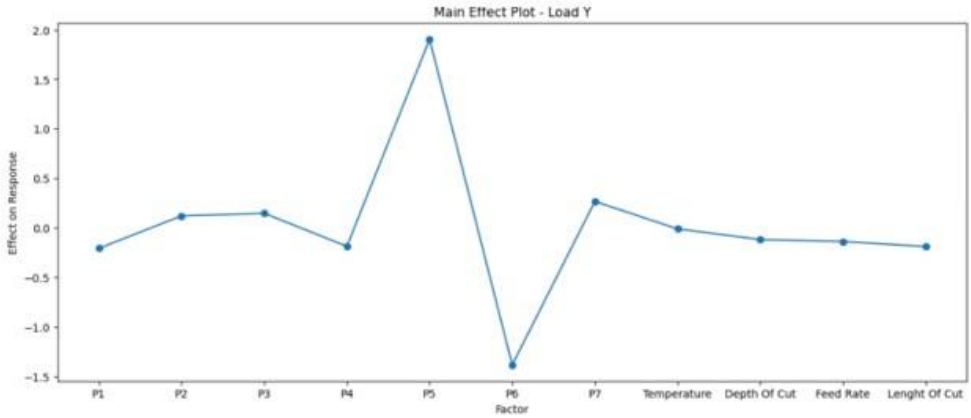

Figure 31: DoE effect plot – Load Y

What can be seen from the effect plots is that one can expect a relatively high influence of the P5 and P6 temperature points to the cutting force size, which could be interesting to investigate further. Finally, another correlation matrix had been computed and plotted, given on the New dataset correlation matrix.

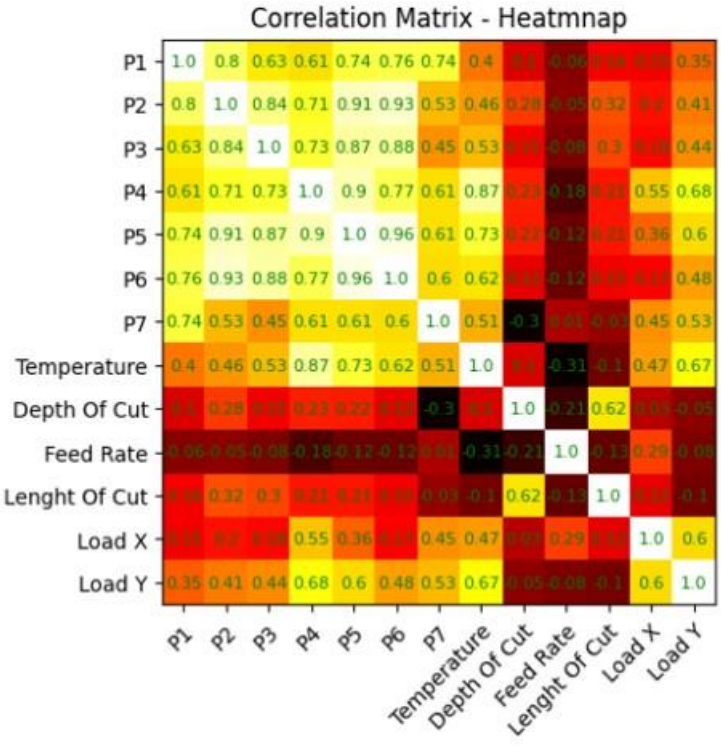

Figure 32: New dataset correlation matrix.

Here, one could definitely notice a much higher level of predictive power of a new dataset. What followed was re-training of the model using the new and merged dataset. Several iterations of training experiments were done again, and the following

architecture and hyperparamters combination showed the best results. A MLP 11x16x32x64 architecture was developed, using Adam optimizer, LeakyReLU activation function (with 0.02 negative slope), batch size of 8 and adaptive learning rate (0.0001 until 0.18 epoch loss and 0.00001 afterwards). The model was trained for 15000 epochs.

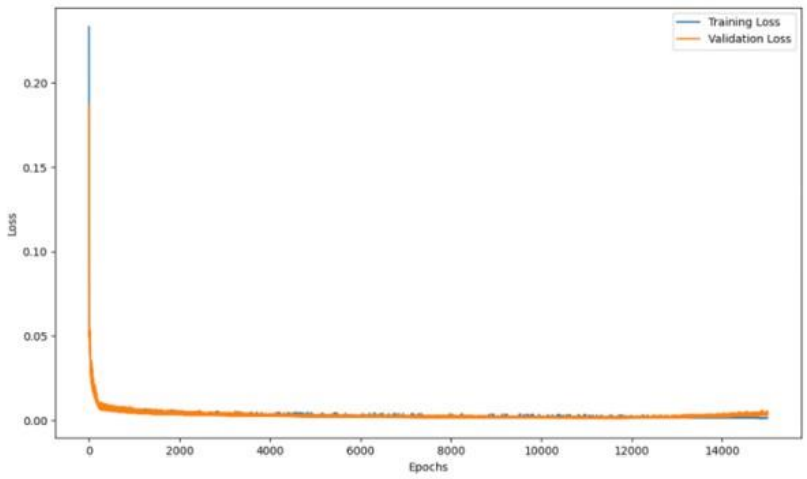

Figure 33: Loss plot for the newly developed model.

Model had been tested on the previously generated test dataset. Following figures are visualizing the results of the model and comparing them to the ground truth.

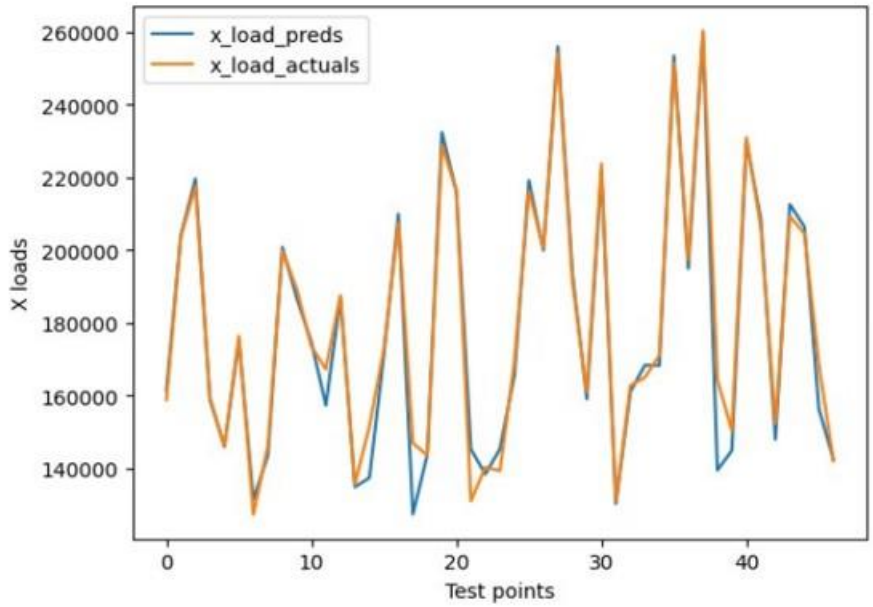

Figure 34: Prediction vs. ground truth comparison – Load X.

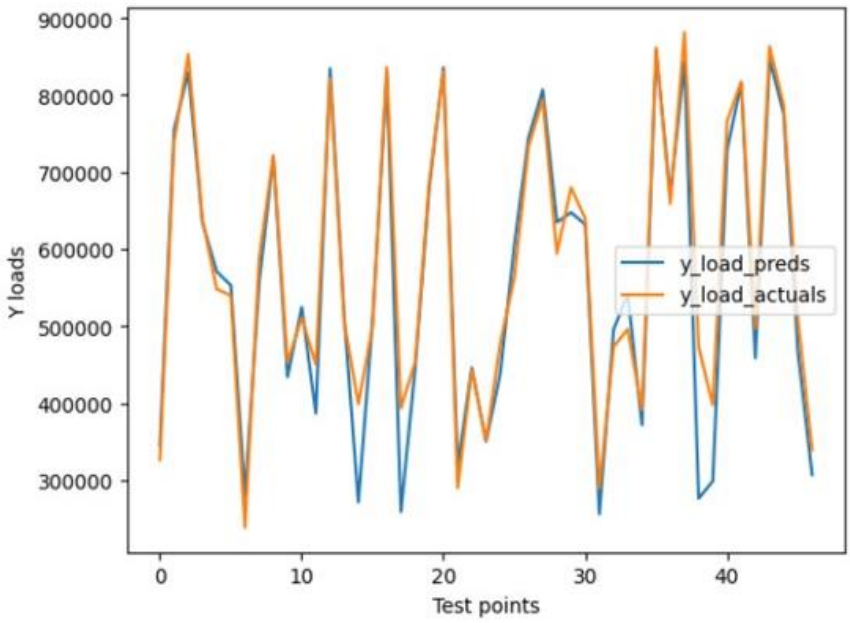

Figure 35: Prediction vs. ground truth comparison – Load Y.

What can be seen is a significant level of agreement of the results with the ground truth, albeit a large range of load values, which clearly indicates superior performance of the new model. MAE error had been computed separately for the Load X and Load Y predictions, totaling to roughly 3785 N and 30860 N, respectively. Compared to the average value of Load X and Load Y, the computed errors corresponds to 1.57 % and 7.2 % of the average value of their respected variables, again showcasing a significant improvement in the models performance.

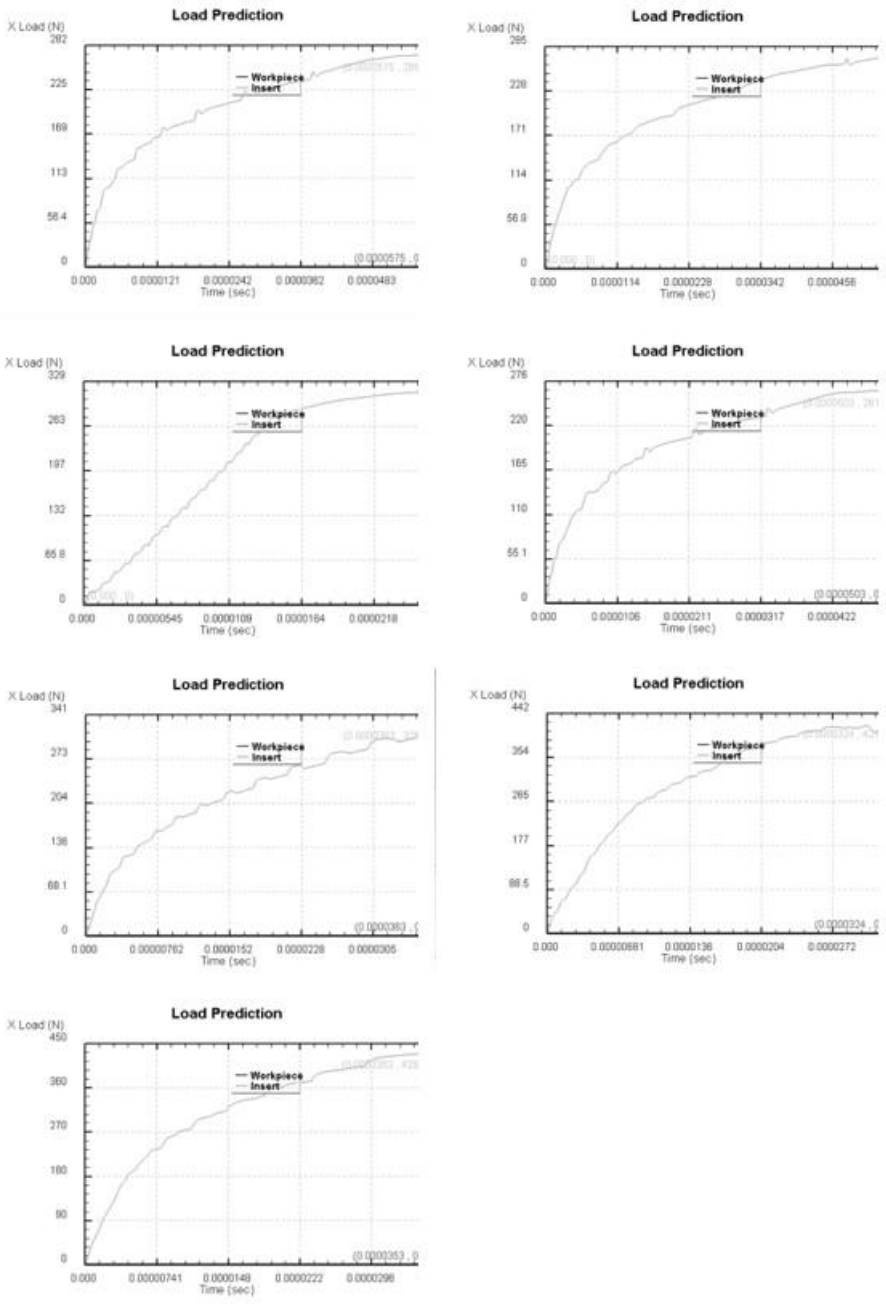

Figure 36: Graphics of Predictions

First 2 stages of the project had the goal of developing a machining cutting forces prediction machine learning framework, using process parameters as features. For this purpose, 2 datasets and 2 models were created. First one based on simulation data exported from Deform 2D software, and second one based on real experimental data. Following the presentation of the stage 2 results, a positive feedback was received and a list of additional requirements / corrections requests had been submitted, as follows:

Experiments

- Explain why the architecture 11-16-32-64-2 was used and compare it with an alternative such as 11-16-32-16-2 to analyze potential differences.
- Investigate how different epoch values (5,000 / 10,000 / 15,000) impact model performance and present the findings in a table.
- Include units in the DOE table for better clarity.
- Create a table showing how different variables, epoch counts, and model architectures affect training duration.

Data analysis & visualization

- Analyze why Load X is very low in rows 137-138-139 but suddenly increases in rows 140-141 in the Data Mining section.
- Identify the reason behind similar fluctuations at test points 18 and 38 in Results X Load and Results Y Load sections and suggest possible solutions.
- Explore the possibility of visualizing cutting forces more effectively.
- Ensure that units are added to all relevant graphs for better readability and interpretation.

Deform 2D & CNC process

- Extend our dataset by placing an additional measurement point on the detached chip and tabulating the results.
- Provide an expert opinion on the type of chips produced in the simulation.

In the following sections, enhancement requests will be addressed and elaborated upon.

Experiments:

First, experimental section of the project enhancement request was tackled. A series of experiments was done, covering a range of NN training parameters and architectures, recording multiple factors. Subsequently, detailed analysis was done. To evaluate the effect different number of epochs, batch sizes and learning rates have on the performance of the model (both training time and inference quality), it was decided to first conduct a series of experiments on a single model architecture, one that proved to be most successful in the previous stage of the project, MLP 11x16x32x64x2. Number of epochs was varied between 15000, 10000 and 5000, batch sizes were varied between 8, 16 and 32. Finally, 2 distinct learning rates were tested, 0.0001 and 0.001. For each experiment, independent evaluation was done, extrapolating loss plots and load predictions comparisons to the ground truth (attached as appendix). Finally, for

each experiment, a Mean Average Error (MAE) as a percentage of the average load value in X and Y direction was computed, to serve as a main accuracy metric.

Table 3: Experiments done using MLP 11x16x32x64x2 architecture

ID	Epoch	Batch Size	Learning Rate	Training Time, min	MAE X, %	MAE Y, %
1	15000	8	0,0001	17	1,9338	6,438
2	10000	8	0,0001	11	1,918	7,822
3	5000	8	0,0001	5	2,667	6,8074
4	15000	16	0,0001	11	2,3303	9,0807
5	10000	16	0,0001	7	2,0688	6,0041
6	5000	16	0,0001	3	3,4097	10,1427
7	15000	32	0,0001	9	2,439	7,2016
8	10000	32	0,0001	5	3,3272	7,4982
9	5000	32	0,0001	2	4,0002	9,2947
10	15000	8	0,001	17	1,9192	4,1896
11	10000	8	0,001	11	1,123	5,2683
12	5000	8	0,001	5	1,5001	4,8017

What can be seen from the experimental data is that the number of epochs and the batch size have the most dominant effect on the training time - larger batches and lower number of epochs significantly speeding up the training. To quantify the effect that different training parameters have on accuracy, average MAE as % of average load was computed for all the main experimental factors, visualized on the figure 1.

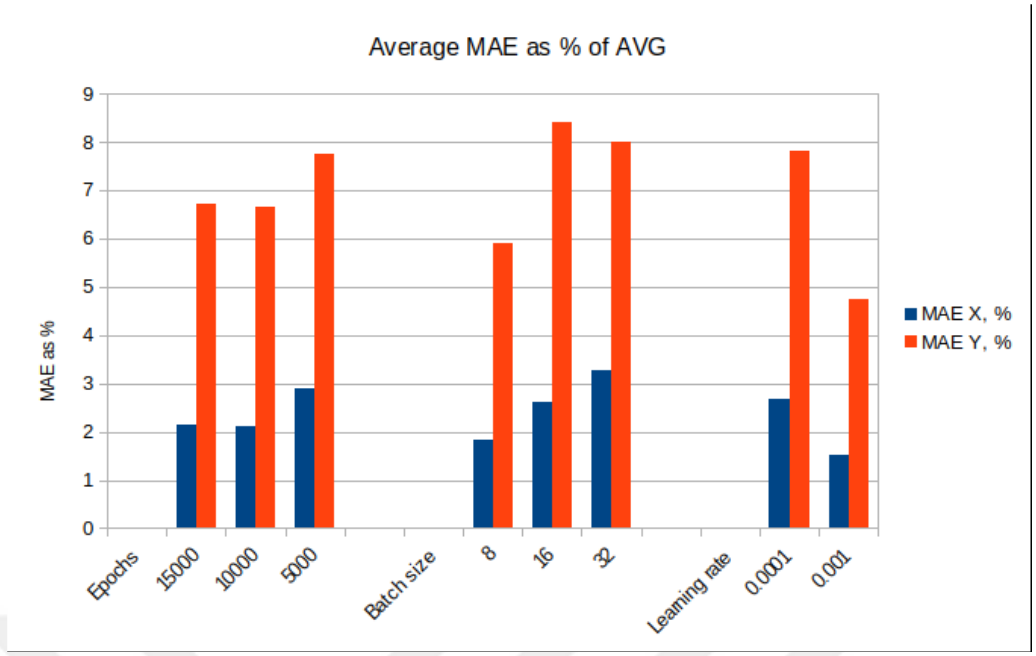

Figure 37: Average MAE as a % of mean load in the respected direction for different experiment parameters.

What can be seen from the graph is that connection between increased accuracy and the size of the parameters is not completely proportional nor inversely proportional. However, it can be said that increasing the number of epochs increases accuracy (as the model converges better) and that smaller batches in this case result in better performance (due to the limited size of the dataset). A surprising conclusion that also can be made is that lower learning rate results in superior performance, possibly because of the adaptive threshold which was applied during training.

The best performing model in the experimental run was a model under ID 10, trained for 15000 epochs, using a batch size of 8 and a learning rate of 0.001. Total training time equaled to 17 min, using NVIDIA RTX 3060 training rig.

Furthermore, 3 experiments using different architectures and similar parameters as the model under ID 10 were subsequently done, with the goal of quantifying how different architectures effect the performance, results being reported in the following tables.

Table 4: Experiments done using MLP 11x16x32x16x2 architecture.

ID	Epoch	Batch Size	Learning Rate	Training Time, min	MAE X, %	MAE Y, %
1	15000	8	0,0001	17	2,2223	7,471
2	15000	8	0,001	18	1,3652	5,2106

Table 5: Experiments done using MLP 11x16x16x2 architecture.

ID	Epoch	Batch Size	Learning Rate	Training Time, min	MAE X, %	MAE Y, %
1	15000	8	0,0001	16	3,946	8,3865
2	15000	8	0,001	16	1,3302	6,0003

Table 6: Experiments done using MLP 11x16x32x64x128x2 architecture.

ID	Epoch	Batch Size	Learning Rate	Training Time, min	MAE X, %	MAE Y, %
1	15000	8	0,0001	18	1,7762	5,8869
2	15000	8	0,001	19	0,8195	4,0187

What can definitely be seen is that reducing the size of the model in this case worsened the performance, while larger architecture had a positive effect to the accuracy, with model MLP 11x16x32x64x128x2 yielding the best performance in the whole project. Model was trained for 15000 epochs, using a batch size of 8 and a learning rate of 0.001. Total training time equaled 19 min, using NVIDIA RTX 3060 training rig. Its results are visualized on the figures below.

Table 7: DOE

Runs	Depth of Cut, mm	Feed Rate, mm/rev	Length of Cut, mm
V1	0,2	0,2	1,5
V2	0,5	0,2	1
V3	0,2	0,5	1,5
V10	0,3	0,2	1,5
V11	0,45	0,15	1,6
V12	0,3	0,2	1
V13	0,3	0,1	1,5
V14	0,4	0,15	1,5
V15	0,5	0,1	1,5
V16	0,2	0,2	1,5
V17	0,35	0,25	1,4
V18	0,4	0,3	1,2
V19	0,25	0,35	1,5
V20	0,3	0,2	1,5
V21	0,45	0,25	1,3
V22	0,35	0,15	1,6
V23	0,4	0,1	1,5
V24	0,5	0,2	1,4
V25	0,3	0,25	1,5
V26	0,2	0,3	1,2
V27	0,45	0,2	1,5
V28	0,5	0,15	1,5

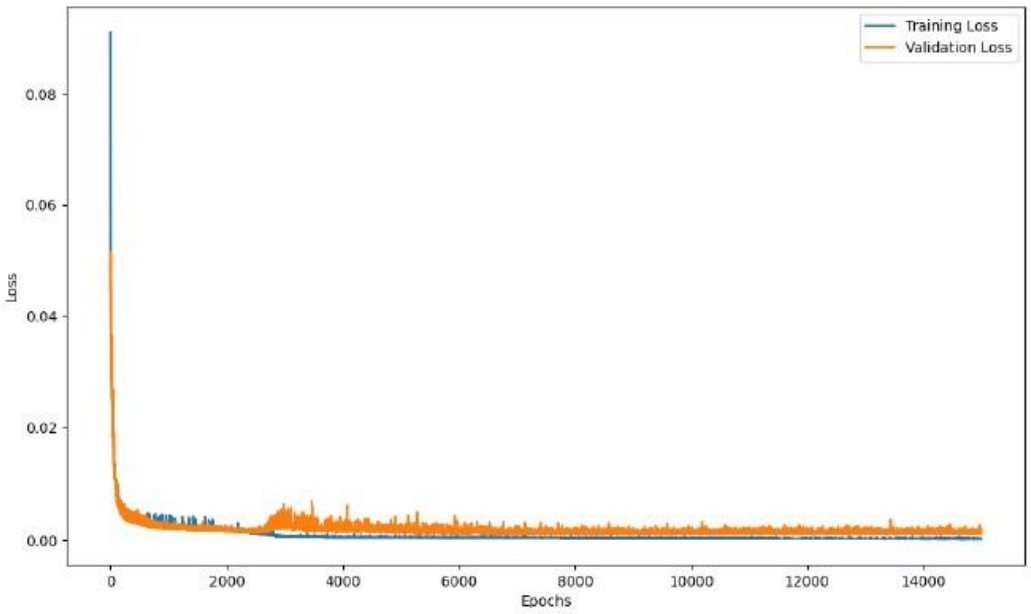

Figure 38: Training loss function of MLP 11x16x32x64x128x2 model.

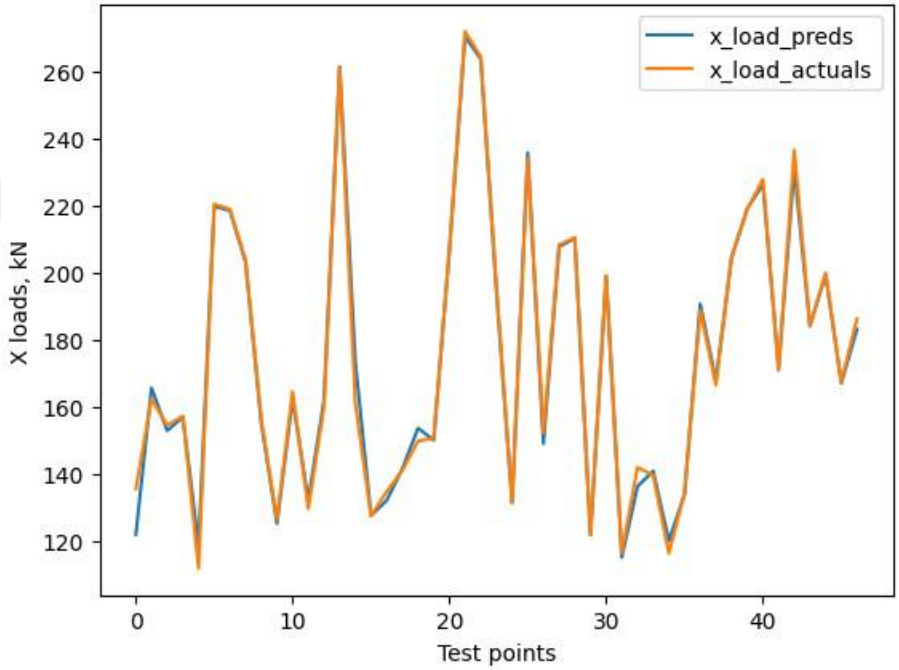

Figure 39: Comparison of predicted load and ground truth load in X direction - MLP

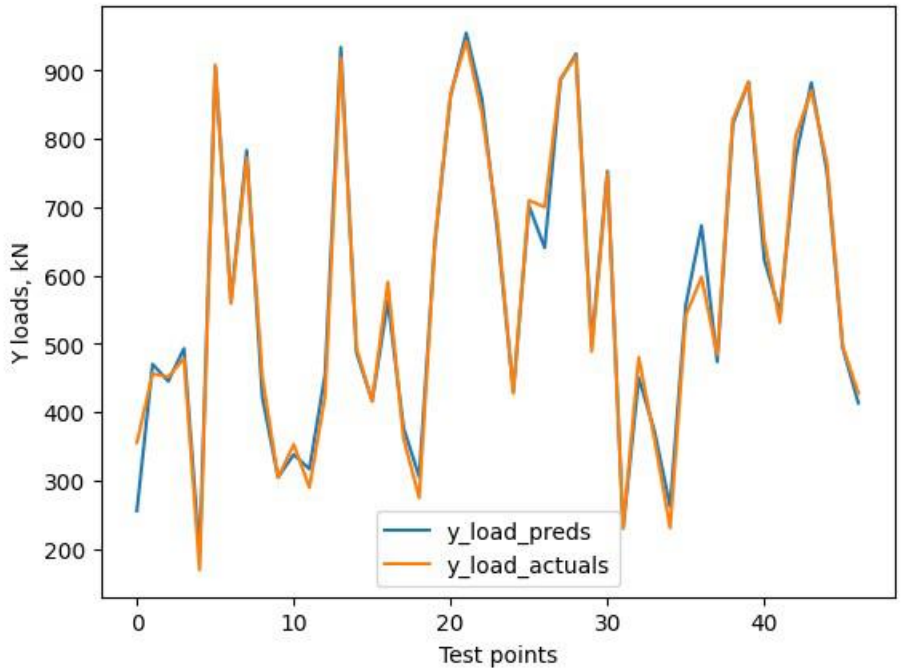

Figure 40: Comparison of predicted load and ground truth load in Y direction - MLP 11x16x32x64x128x2 model.

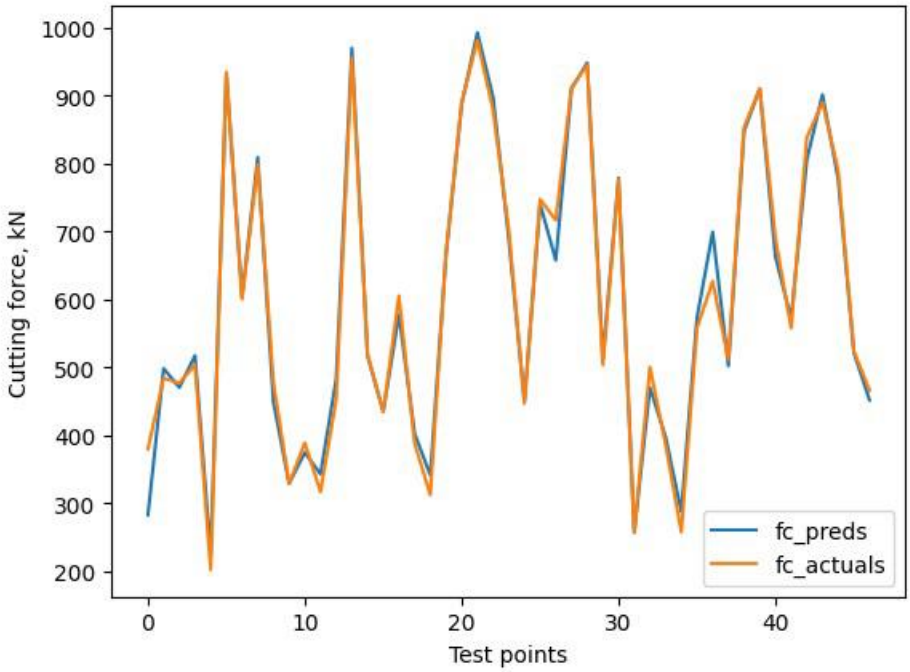

Figure 41: Comparison of predicted load and ground truth resultant load - MLP 11x16x32x64x128x2 model.

Data Analysis & visualization:

To elaborate on the sudden load increase in rows 140 and 141 in the Data Mining section, it is first vital to point that this refers to data coming from the simulation V28.

135	189.928345	131.888397	167.989777	162.20311	194.410629	233.438583	211.112976	0.2	0.15	1.5	276142	152181
136	195.931564	137.492783	173.452026	165.540207	201.399063	235.703125	220.393692	0.2	0.15	1.5	275952	150.49
137	201.566269	143.248016	178.922745	168.873917	207.984116	239.756226	228.805054	0.2	0.15	1.5	276.44	149892
138	206.995926	149.125748	184.453262	172.243011	214.377823	245.576355	237.134949	0.2	0.15	1.5	283.09	158612
139	212.306198	155.102509	190.027023	175.652039	220.487366	251.163971	244.561188	0.2	0.15	1.5	288.01	168046
140	217.115204	161.144989	195.534058	179.065292	226.215302	255.730225	251.512024	0.2	0.15	1.5	289989	172258
141	221.612885	167.193192	200.993622	182.477844	231.617218	259.476257	258.036163	0.2	0.15	1.5	289955	172252

Figure 42: Final rows of V28 simulation raw data.

These rows are the final rows of the simulation data. Upon observing other simulations from the dataset, the same behavior was not seen in all the cases.

107	158.549149	114.168442	144.330215	154.781479	178.323654	133.602844	164.356979	3066.096695	0.25	0.15	0.8	249167	120102
108	164.349121	121.968407	150.67746	160.564514	190.599869	143.531693	175.087769	3124.068112	0.25	0.15	0.8	252779	124762
109	170.225937	129.705154	157.069351	166.45842	197.048737	153.072632	185.519531	3173.970285	0.25	0.15	0.8	256464	129464
110	176.673477	137.34787	163.639572	172.697662	202.636154	162.328491	195.865585	3222.218036	0.25	0.15	0.8	258515	135262
111	183.576126	144.960587	170.479324	179.304306	208.804047	171.306015	206.025284	3269.137891	0.25	0.15	0.8	260886	139539
112	191.049652	152.644424	177.826584	186.479141	217.287521	180.581863	217.129807	3322.034579	0.25	0.15	0.8	261031	142823
113	198.337616	160.516129	185.367157	193.728333	226.794449	190.027191	228.502701	3386.81646	0.25	0.15	0.8	261199	145788
114	204.97139	168.551529	192.906113	200.644196	234.519547	198.74675	238.146591	3451.467837	0.25	0.15	0.8	262313	146266
115	211.214066	176.542847	200.443649	207.338486	240.927399	207.058365	247.139664	3522.455275	0.25	0.15	0.8	262385	145114
116	217.187531	184.39212	207.820404	213.744476	246.56102	215.066284	255.683929	3602.613237	0.25	0.15	0.8	261364	142.56

Figure 43: Final rows of a different simulation raw data.

However, V28 was not an isolated case and several more simulations did indeed demonstrated similar stochastic behavior in the final steps of the simulation. This of course lead to a well founded doubt that this is due to the chip detachment which presumably happened in these cases. During cutting, plastic deformation occurs along the highest shear stress plane where the stress exceeds the shear strength of the material. The workpiece material which forms a chip deforms plastically before the chip's removal, which happens due to different physical effects which will be further elaborated upon. To properly discuss the sudden drop in the cutting forces which can be observed in some simulations, one has take into account the underlying algorithm that is used for computing cutting forces in the discrete nodal points of the FEM mesh (2D in this case). Simulations that had been done in the scope of this project are utilizing Newton-Raphson method, which is an iterative numerical technique that is used to successively find better approximations of the roots of a real-valued non-linear function, usually in the f(x) = 0 form. It takes an initial guess and approximates the value of the function at that guess using its tangent line. The next approximation is where the tangent line intersects the x-axis.

In the context of CNC cutting force computations, the non-linearity of the function comes from the displacements (or deformations) in the material (which can be non linear due to material properties), tool geometry and cutting conditions. These

have large impact to the resultant cutting forces. Upon closer inspection of the simulation, one can definitely see that no chip detachment happened (figure 45).

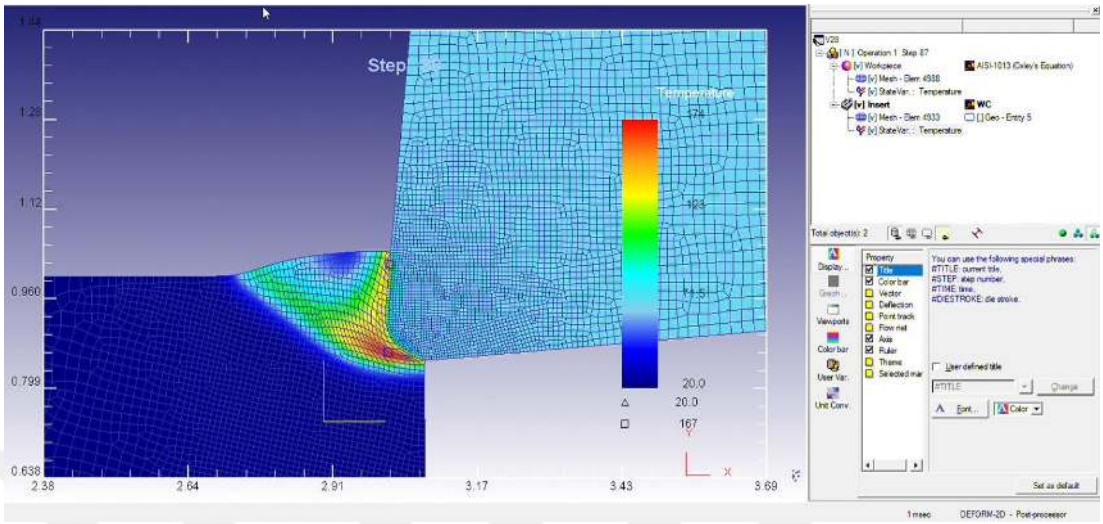

Figure 44: V28 simulation, roughly at ¾ of the simulation length – chip detachment is not visible & probable.

That being said, several possible explanations can be given for a sudden drop in the cutting forces:

- Local yield of the material
- Brief loss of contact or change of contact conditions between the tool and the workpiece-local chip deformation.

A second pattern that was brought to attention was a major difference between predicted load and ground truth loads in the model prediction results reported in the previous stage, visible on the figure 46.

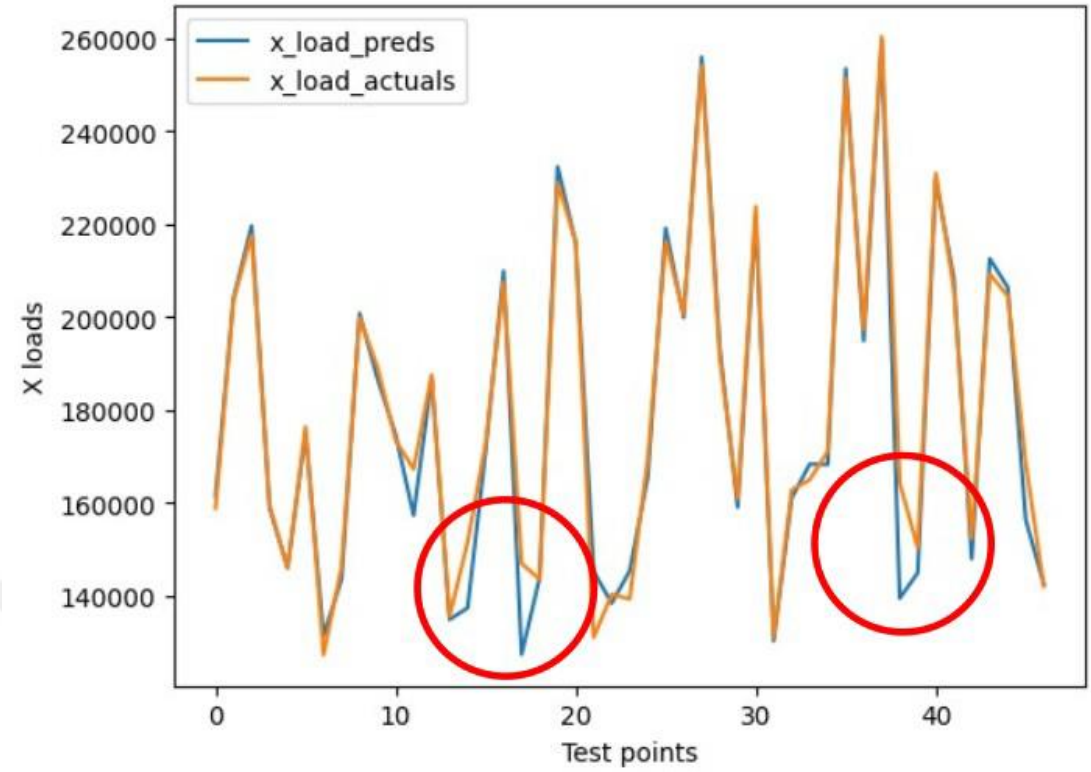

Figure 45: Major difference between predicted and ground truth load – previous stage results.

For these however, the explanation is rather straightforwards, being simply the lack of capability of the model to make accurate prediction in the particular data segment. This lack of capacity can be traced to the fact that the underlying training dataset is still quite small, and the model simply generalizes across the unknown segments, which leads to it not making completely accurate predictions in them. Improvements would involve training the model on more granulated and bigger datasets.

FEM & CNC process: Cutting & abrasive processes - basic theoretical background Cutting and abrasive processes are part of the main group of separation processes. They are mainly used for metallic materials and are fundamental steps in the production processes in machine and vehicle manufacturing, in the aerospace industry, in equipment and drive technology, in biomedical engineering and in many other industrial sectors. They provide accuracies within the ISO tolerance grades of IT2 to IT10. Cutting and abrasive processes represent manufacturing by means of material separation, particularly of material removal. Material in the form of chips is mechanically removed from a raw part/workpiece by one cutting edge (in turning), several cutting edges (in milling) or numerous cutting edges (in grinding). Cutting

process parameters (number of cutting edges, their shape and position in relation to workpiece) is known and describable, while in the case of abrasive processes, only statistical parameters are known regarding geometry and the number of the cutting edges in the abrasive material. In cutting processes, a cutting edge penetrates into the workpiece material. The relative movement between the tool and the workpiece can be described by means of the primary motion (cutting motion) with the cutting speed vc and the feed motion with the feed speed vf (figure 10).

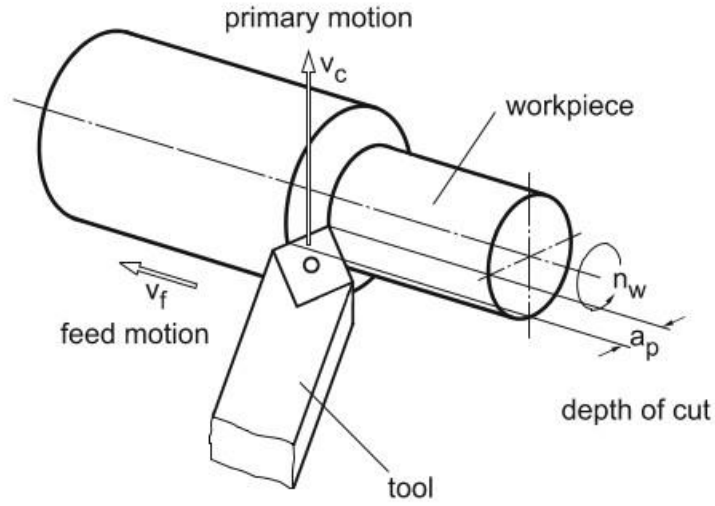

Figure 46: Turning process parameters

When observing these processes from a system point of view, we can distinguish between input and output variables, which is a fundamental premise we need to make to start modeling the process. Input variables can be split into system variables (like static and dynamic stiffness of the machine tool, temperature response, workpiece shape, structure and chemical composition, etc.) and manipulated variables (cutting speed, feed speed, clamping force, etc.). The output variables are divided into process and effect variables. Process variables like resultant forces, powers, temperatures in the chip formation zone, vibrations caused by the process and acoustic emissions are only perceivable during the actual process. Effect variables can be measured at the workpiece (deviations in dimension, shape and position, micro-geometry, influence on the external surface zone), at the tool (wear), at the machine tool (temperature rise, wear) and in the cutting fluids (temperature rise, contamination and chemical properties). The following four criteria are used for evaluating a process and define the machinability:

- Resultant force
- Tool wear
- Surface properties of the workpiece
- Chip form

Chip formation: In cutting and abrasive processes, the cutting edge penetrates into the workpiece material, which is thus plastically deformed and slides off along the rake face of the cutting edge. This is called chip formation. We can assume that the deformation is two-dimensional. The two-dimensional deformation is only disturbed at the edges of the cross section of the undeformed chip, at the free surface and in front of the cutting edge corner, as there is material flow at an angle towards the orthogonal plane, which is caused by linkage to the undeformed material by the free surface, respectively. Undeformed chip cross section and cutting edge are visible on the figure 48.

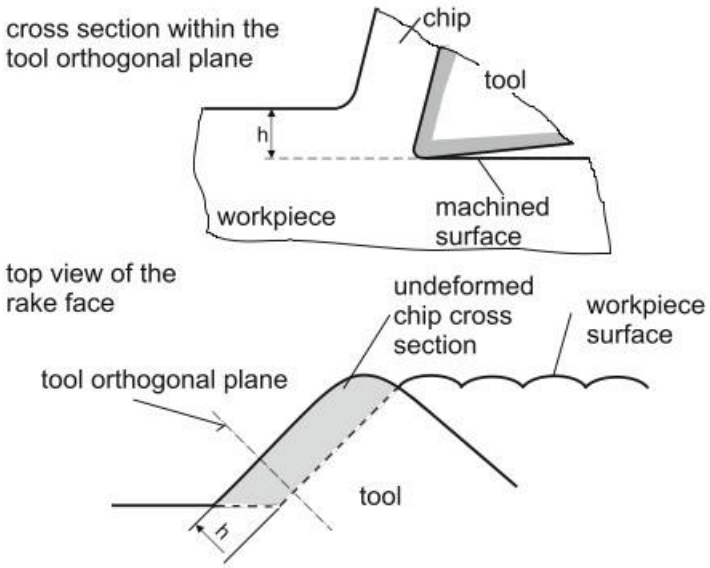

Figure 47: Chip formation schematics

Chip analysis:

To perform analysis on the chip types that are the most probable to occur in the simulations that were done to generate an input dataset, a basic requirement was to extrapolate most influential chip formation parameters from it. Given the theoretical background laid in the previous chapters and the available input parameters, it was deduced that those are workpiece material and the average feed rate used in the simulations. In the scope of this project, most of the Deform 2D simulations had been

made using mostly AISI-1013 as a workpiece material, and Ti-6Al-4V as tool material, with only exception being the simulation v15 which used 15-5 stainless steel for workpiece and TITANIUM-TYPE-2 for tool. Given the predominant usage of AISI 1013, it was decided to proceed with it as a referent material for the chip analysis. Observing at the DOE table used to generate the parameters of the simulations, one can easily compute that the average feed rate used in the project was around 0.2 mm/rev. FEM has strong modeling capabilities for the described chip-related phenomena, as it can track any material point on the FEM mesh and get the temperature evolution over time – visible on the figure 16.

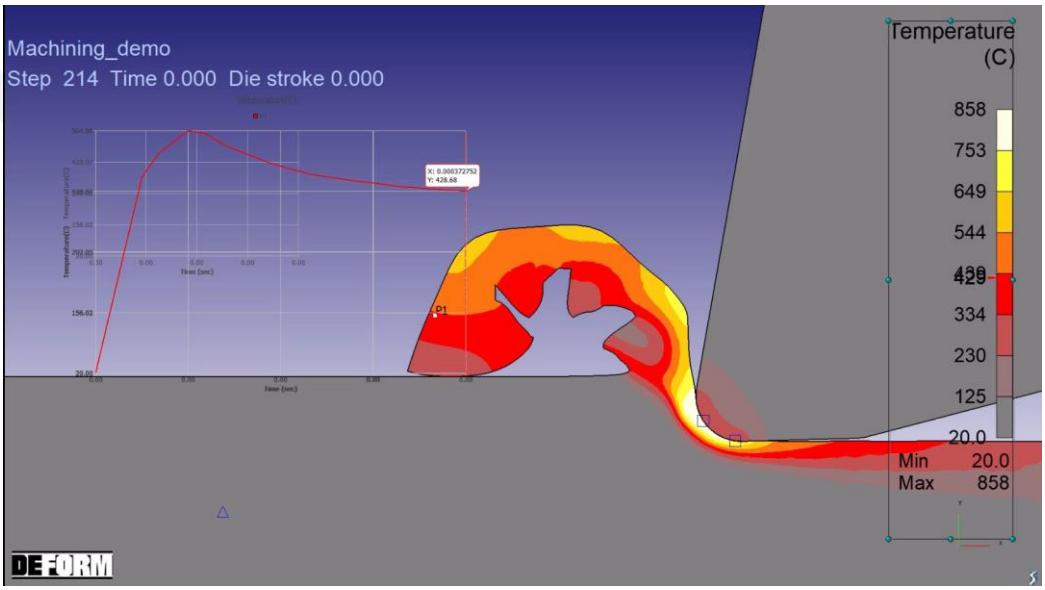

Figure 48: FEM example with point -temperature measurement on the chip.

Usually, chip detachment is not modeled in Deform 2D as it is not conventionally needed, but one could potentially model the fracture and separate the chip and then run a separate simulation to model the cooling to room temperature, if required.

Process monitoring & future potential:

From the conducted literature survey, accumulated theoretical knowledge and conducted experiments, one can draw conclusions about the importance of predicting relevant process parameters when it comes to cutting machining processes. Knowing the cutting forces in advance can help optimizing the costs of the process with valuable input to the design of the clamping system and tools. It can also greatly reduce the costs, as a steady influx of information about the cutting forces can induce preventive

change of the worn-out tools, instead of them being changed after noticing that they had been producing poor quality parts. Finally, knowledge about the chip type and morphology is important for the tool design, design of the workpiece space, safety and for the reliability of the process. To ensure the maximum benefits are reaped from, accurate parameter predictions are required, which are based upon high-quality data. These can be provided by either dedicated simulations or via process monitoring solutions, which provide real-time feed directly from the factory floor. Process monitoring does not only help with detection of non-conformities but also with their anticipation and prevention, as well as real time information. Accumulated data can also be used to perform process trend analysis, often providing valuable production insights. Process monitoring systems start with the measurement data acquisition. An example of a multisensor deployment on a single CNC center is visible on the figure 17, emphasizing the potential of a multi-modal data sources which are capable of inspecting the process from various angles.

Figure 49: Example of a multi-sensor CNC process monitoring acquisition setup

Signals coming from these sensors are to be subjected to a series of post-processing steps, ranging from filtering, frequency domain transformation and feature selection, before being ready to be used for either trend analysis, dashboard

visualization or prediction model training - using basic and advanced ML methods. Correlation of the surface and dimensional measurements with the data collected during process run-time can help linking the process monitoring output with part quality, drastically increasing the value of a process monitoring solution.

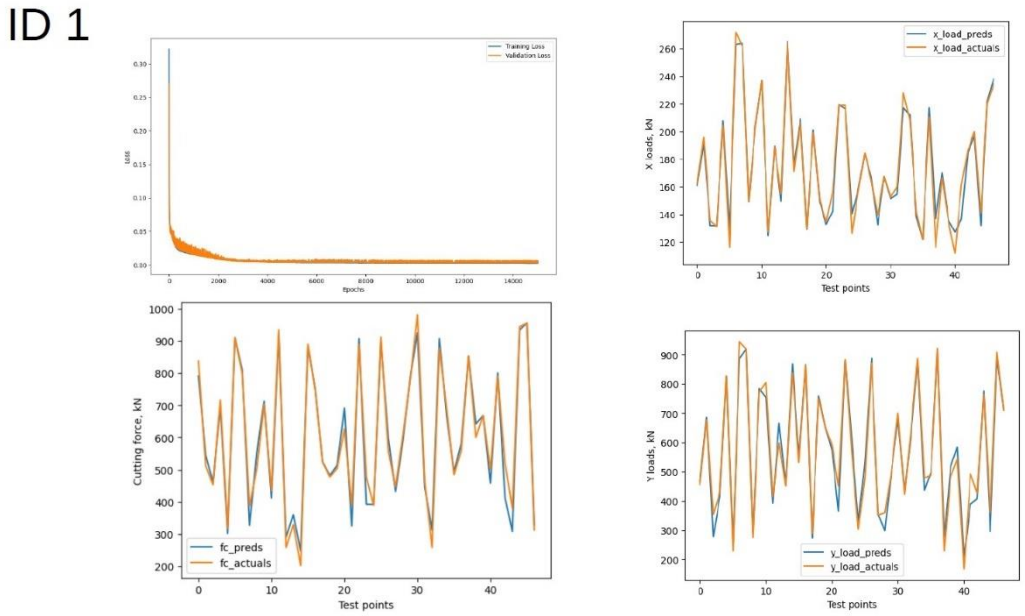

Figure 50: MLP 11x16x32x64x2 ID1

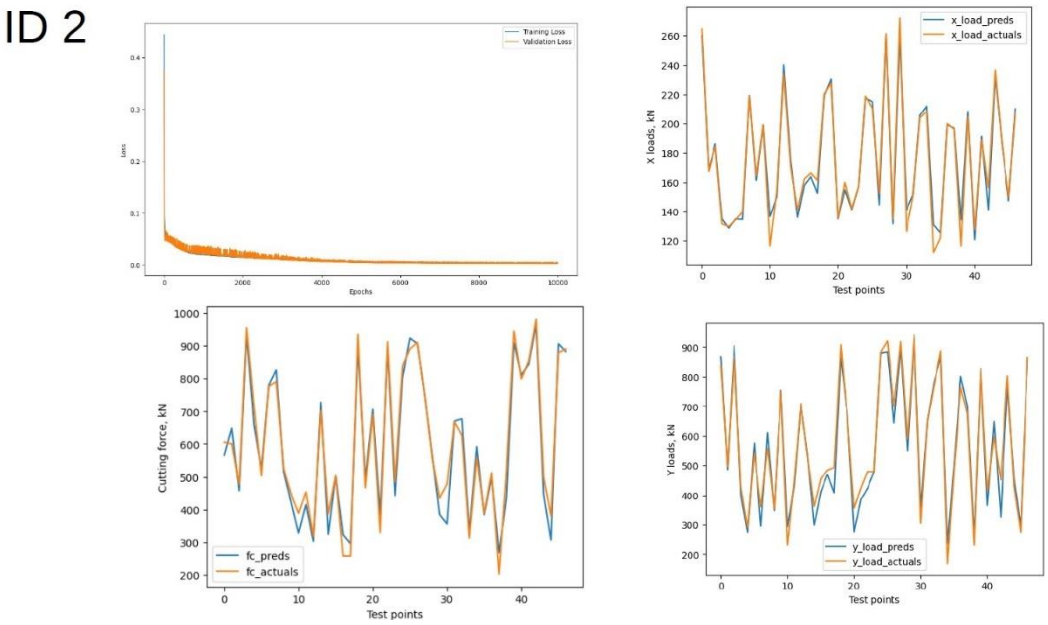

Figure 51: MLP 11x16x32x64x2 ID2

ID 3

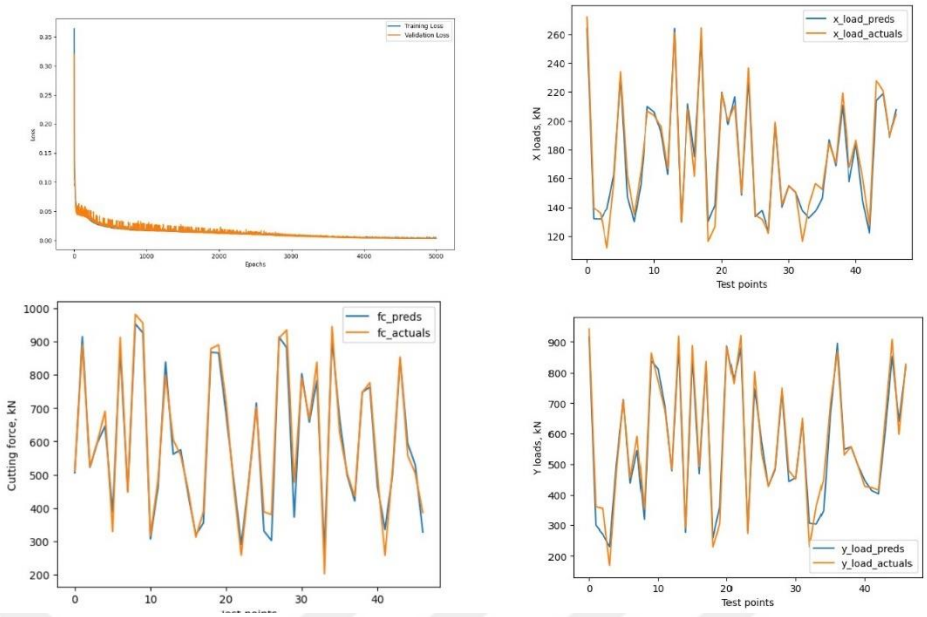

Figure 52: MLP 11x16x32x64x2 ID3

ID 4

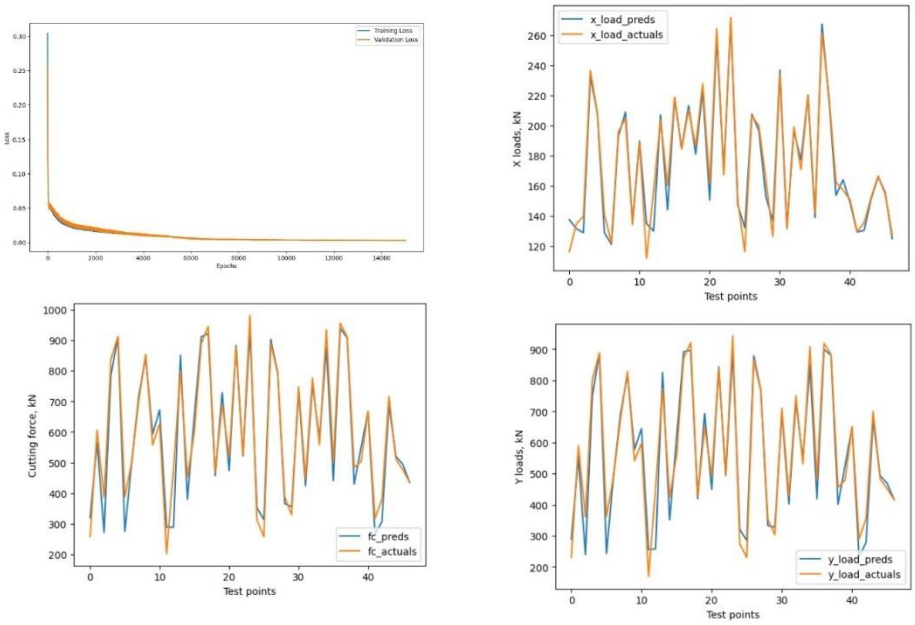

Figure 53: MLP 11x16x32x64x2 ID3

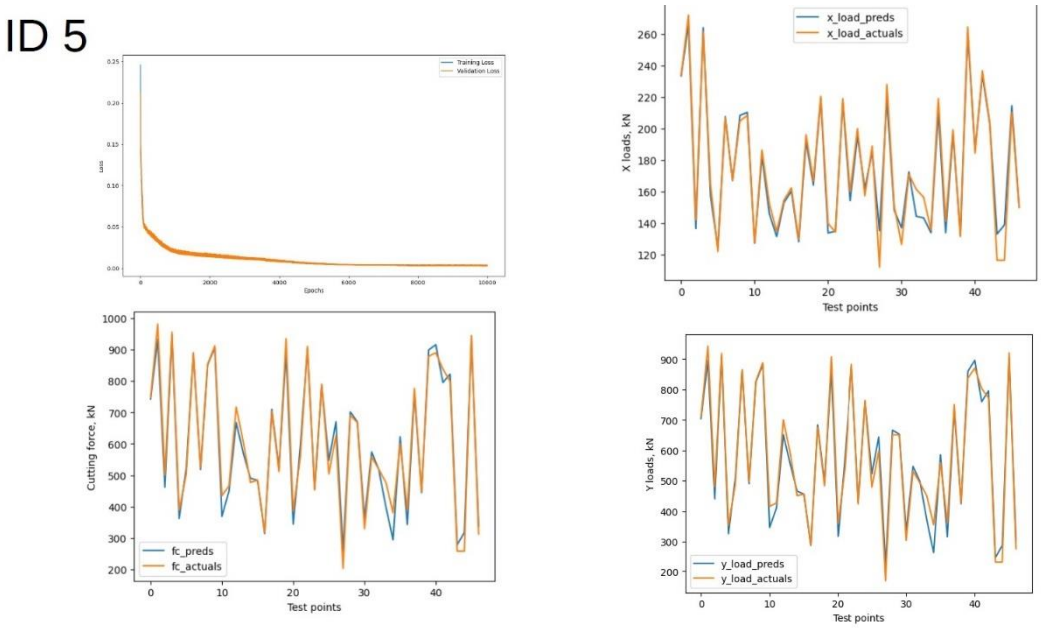

Figure 54: MLP 11x16x32x64x2 ID5

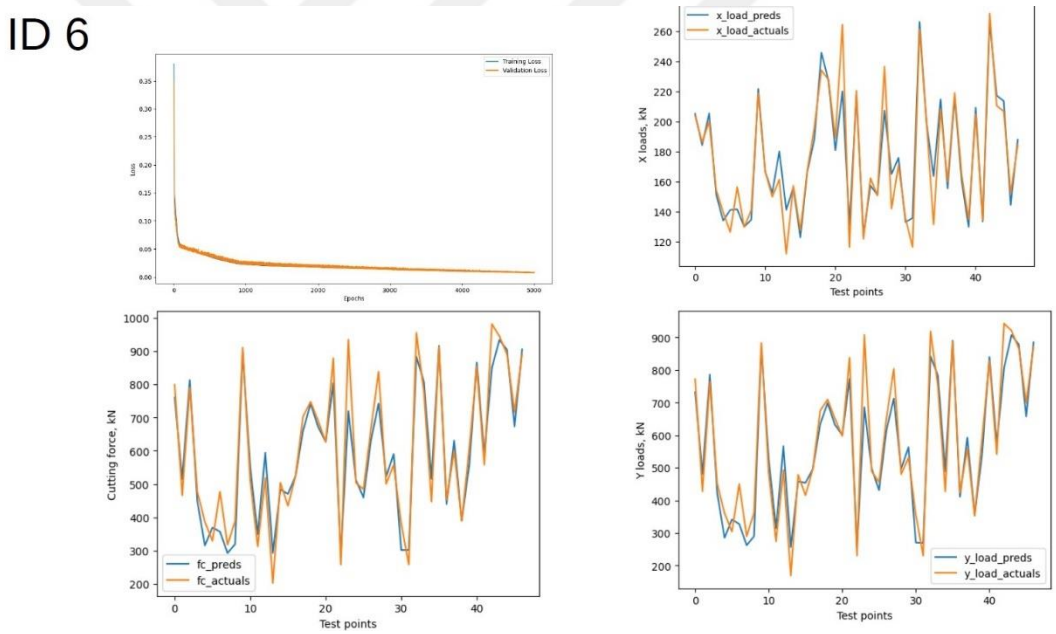

Figure 55: MLP 11x16x32x64x2 ID6

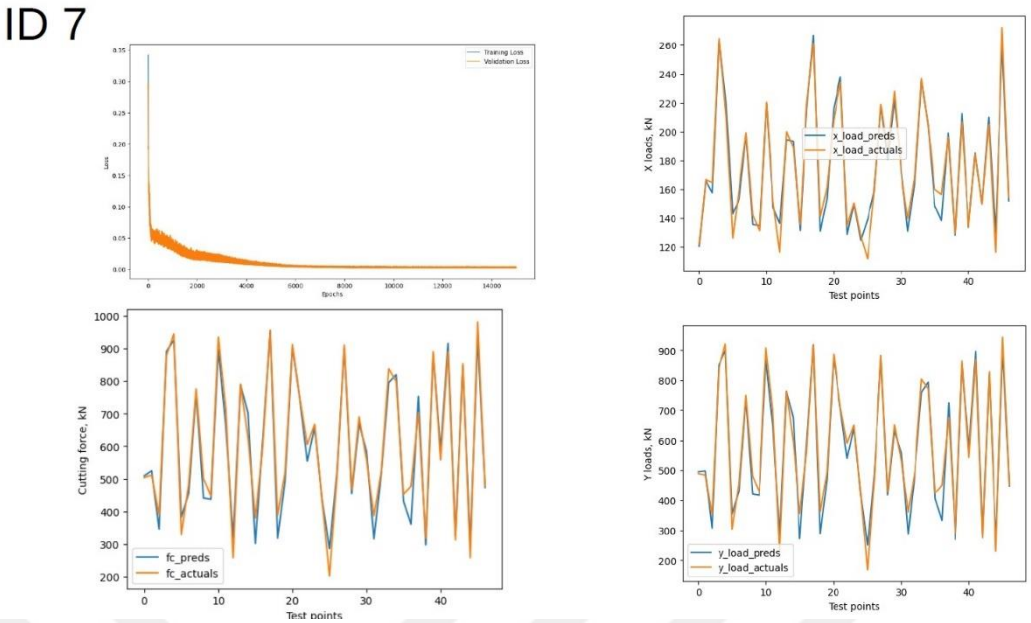

Figure 56: MLP 11x16x32x64x2 ID7

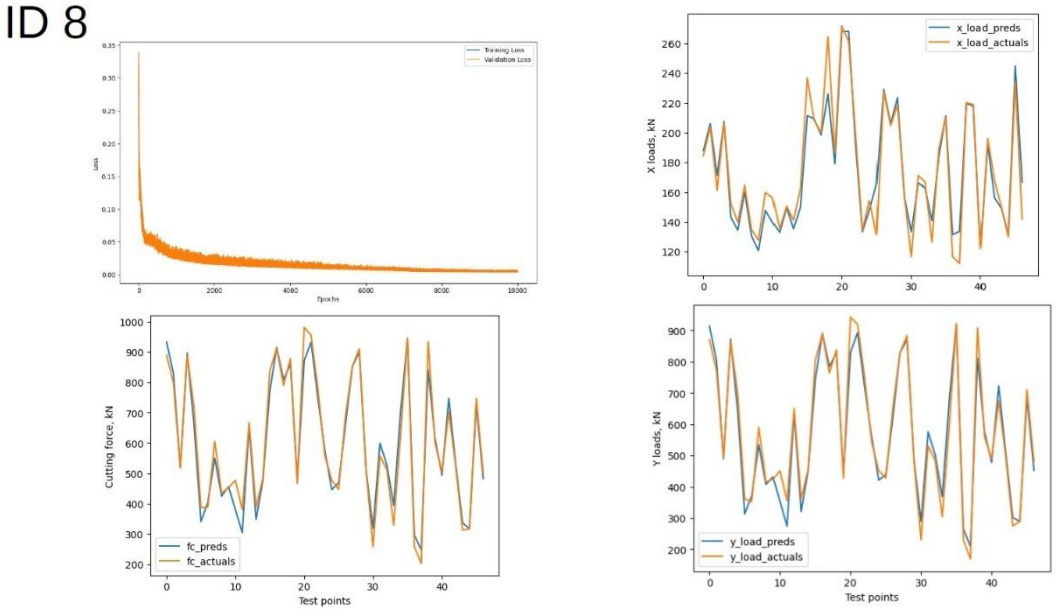

Figure 57: MLP 11x16x32x64x2 ID8

ID 9

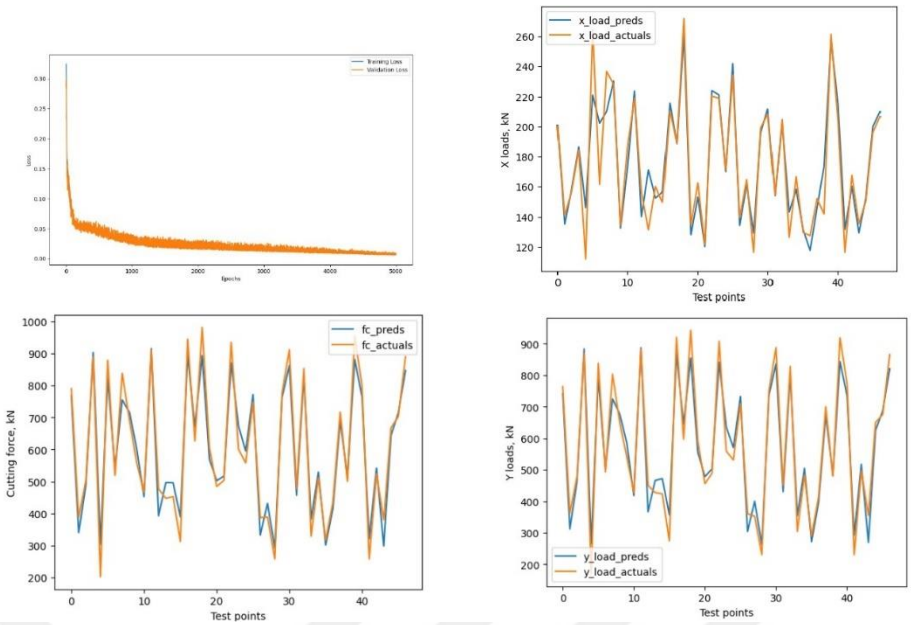

Figure 58: MLP 11x16x32x64x2 ID9

ID 10

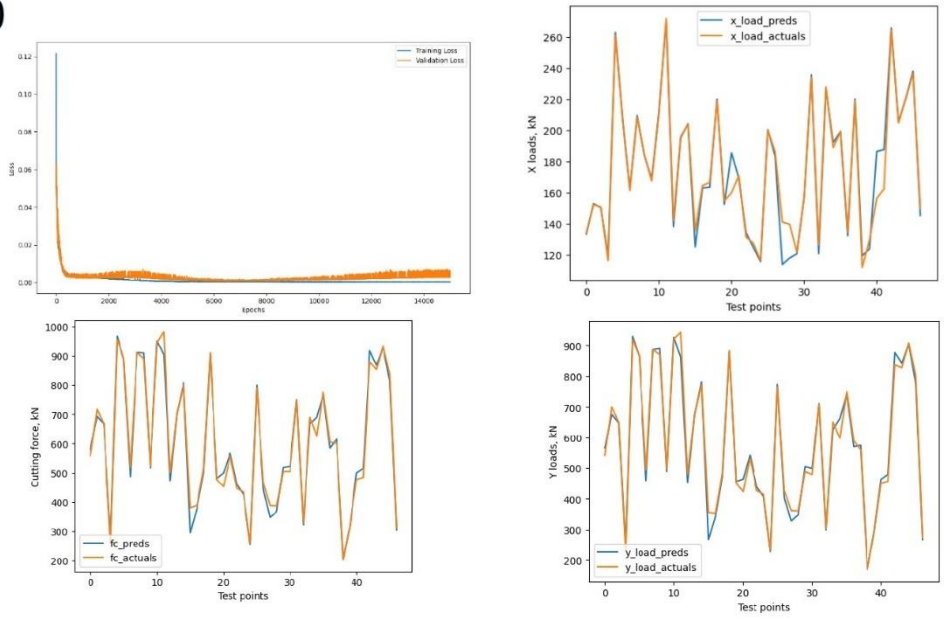

Figure 59: MLP 11x16x32x64x2 ID10

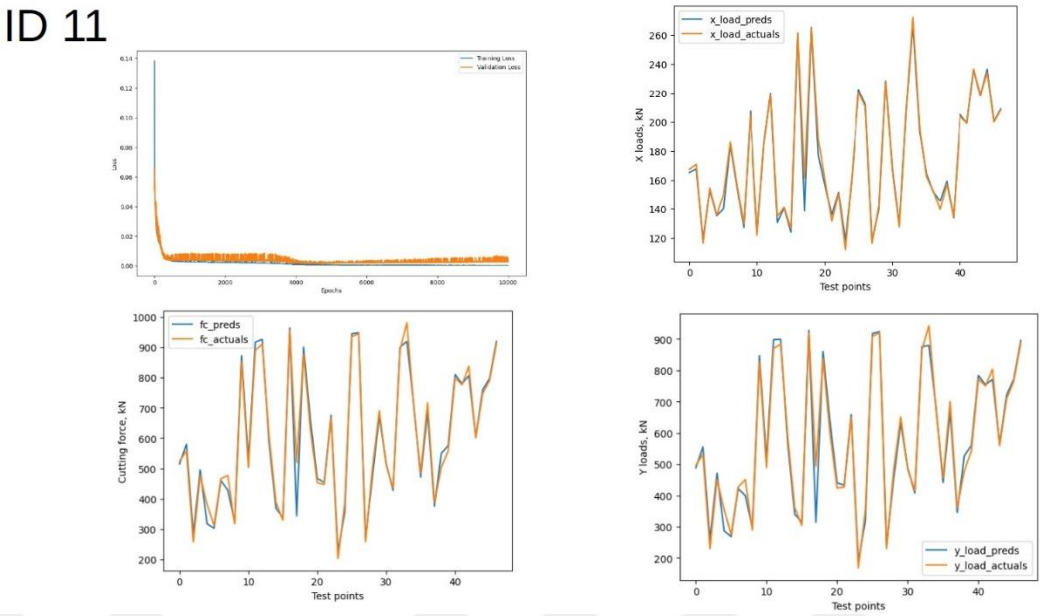

Figure 60: MLP 11x16x32x64x2 ID11

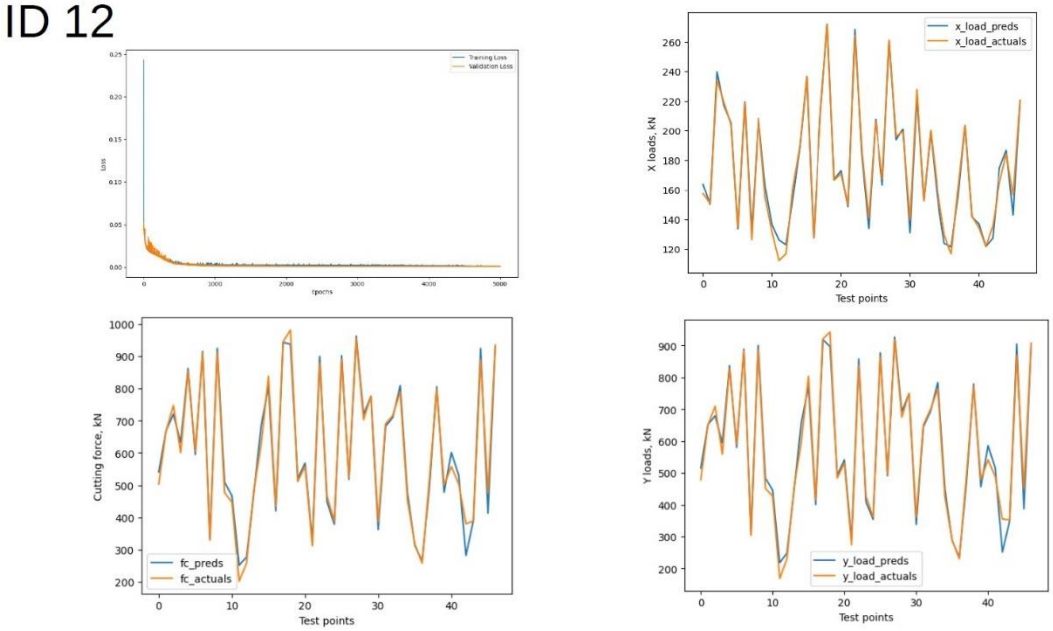

Figure 61: MLP 11x16x32x64x2 ID12

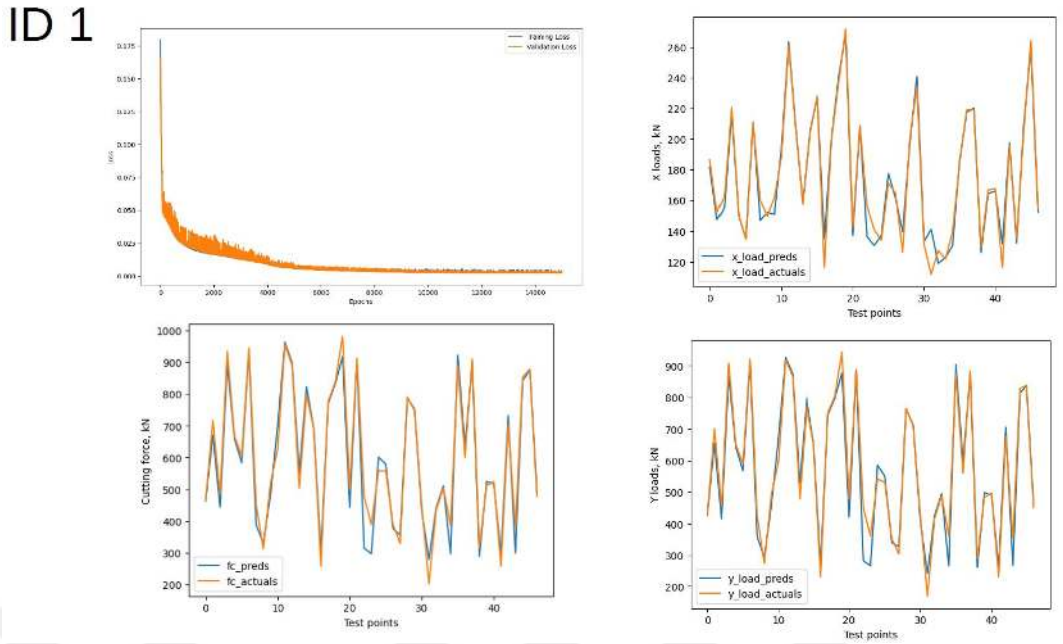

Figure 62: MLP 11x16x32x16x2 ID1

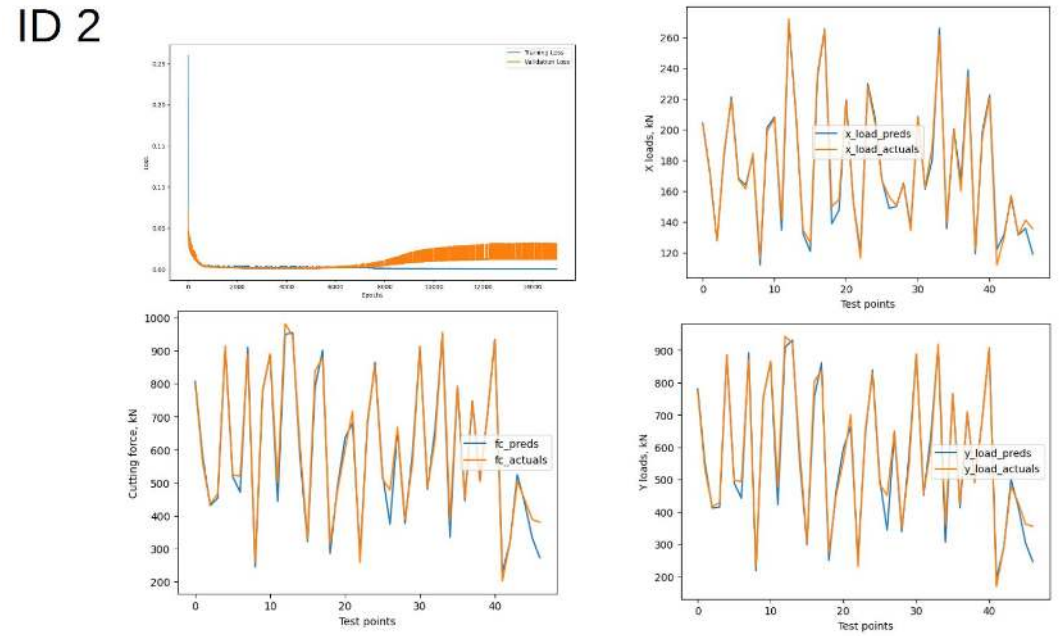

Figure 63: MLP 11x16x32x16x2 ID2

ID 1

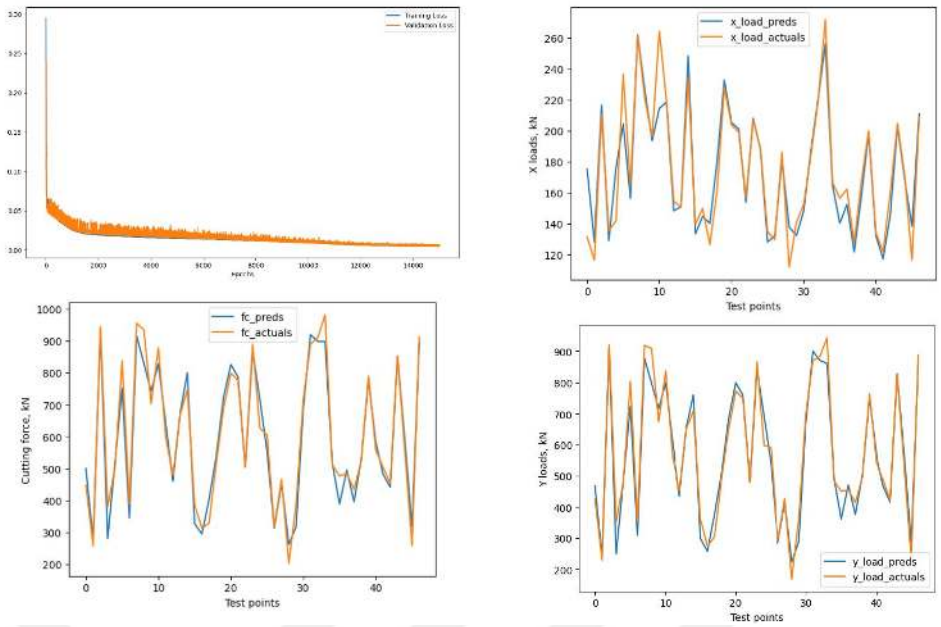

Figure 64: MLP 11x16x16x2 ID1

ID 2

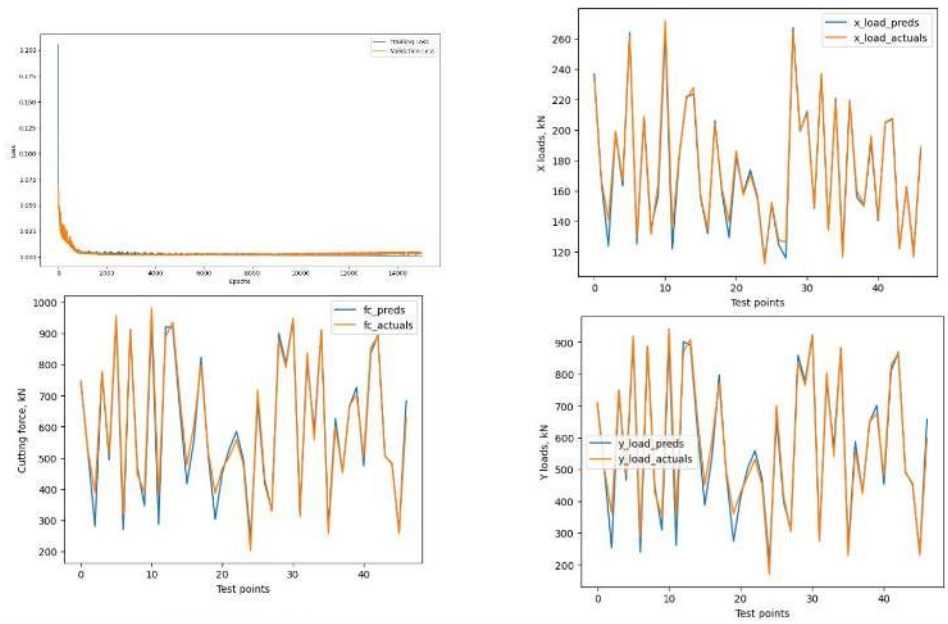

Figure 65: MLP 11x16x16x2 ID2

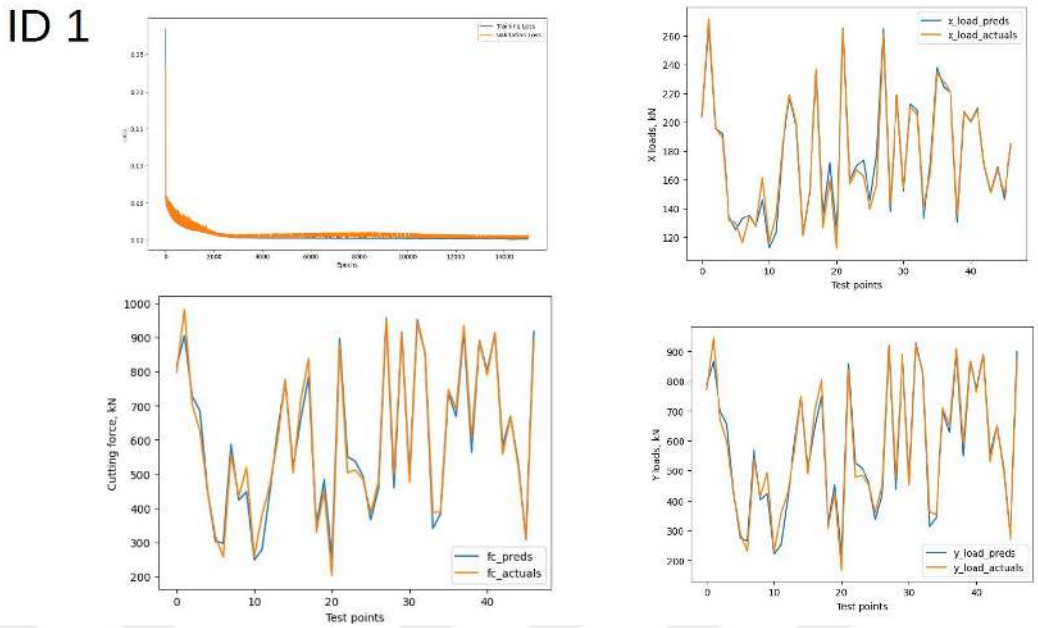

Figure 66: MLP 11x16x32x64x128x2 ID1

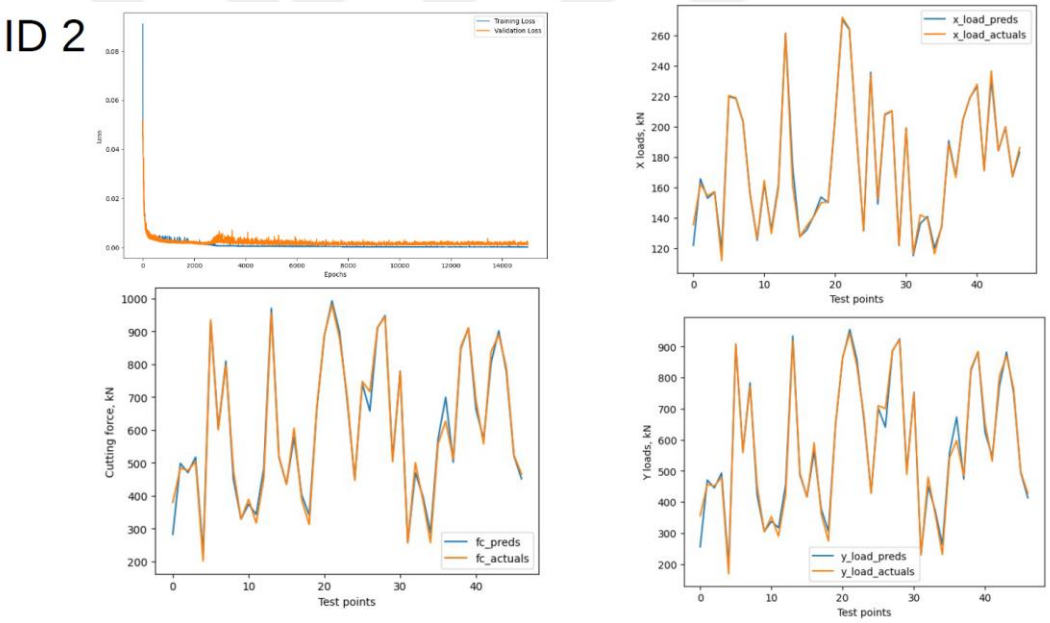

Figure 67: MLP 11x16x32x64x128x2I D2

Table 8: Key Contributions and Feature Directions

Category	Contributions of the Thesis	Suggestions for Future Work
Machine Learning	Developed an MLP-based model	Alternative deep learning models (CNN, RNN)
Data Processing	Integrated FEM and experimental data	Expand dataset for better generalization
Feature Selection	Identified key influential	Utilize additional sensors for new features
Outlier Handling	Applied IQR and Z-score for data cleaning	Test alternative normalization techniques
Industrial Use	Evaluated real-world feasibility with sensor data	Develop a real-time prediction system

CHAPTER III
CONCLUSION

While writing the thesis, our focus was on analyzing the data obtained from FEM and using it for cutting force estimation. By comparing some learning methods, we proved that these methods can provide innovative contributions and are effective in the process of estimating the cutting force.

Our studies have revealed that our algorithms used for machine learning have the potential to increase accuracy for prediction, optimize parameters such as cutting speed and wear temperature, and increase efficiency.

At the same time, it allowed the analyses to be processed and made meaningful. For this reason, it helped to develop a model that adapts to variable process parameters. With this study, the importance of machine learning-focused and data-centric approaches was emphasized.

If we focus on future improvement studies, we need to focus on developing data and all parameters affecting the model. In addition, parallel application of such models with industrial systems can create new development opportunities for process controls. Collecting the data used directly from the production site can trigger much more accurate learning results.

In conclusion, this thesis provides a machine learning-based framework for cutting force prediction, offering practical solutions for process optimization and tool management. The findings highlight the transformative role of machine learning in modern manufacturing applications, paving the way for smarter and more efficient production.

REFERENCES

[1] TÜRKMENOĞLU Cumali and TANTUĞ A. Cüneyd (2014), "Sentiment Analysis in Turkish Media", *International Conference on Machine Learning (ICML)*, pp. 1-11, Beijing.

[2] HEATON Jeff (2015), *Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks*, Heaton Research Inc., Chesterfield.

[3] LOSHCHILOV Ilya and HUTTER Frank (2019), "Decoupled Weight Decay Regularization", *7th International Conference on Learning Representations*, pp. 1-18, New Orleans.

[4] DANCKER Jonte (2024), *A Brief Introduction to Recurrent Neural Networks | Towards Data Science*, https://towardsdatascience.com/a-brief-introduction-to-recurrent-neural-networks-638f64a61ff4, DoA. 4.12.2024.

[5] MARKOPOULOS Angelos P., KARKALOS Nikolaos E. and PAPAZOGLOU Emmanouil L. (2020), "Meshless Methods for the Simulation of Machining and Micro-machining: A Review", *Archives of Computational Methods in Engineering*, Vol. 27, pp. 831–853, DOI: 10.1007/s11831-019-09333-z.

[6] SCHMIDT Robin M. (2019), "Recurrent Neural Networks (RNNs): A gentle Introduction and Overview", *CoRR*, Vol.abs/1912.05911 DOI: 10.48550/arXiv.1912.05911.

[7] KYUNGHYUN Cho, BART van Merrienboer, CAGLAR Gulcehre, DZMITRY Bahdanau, FETHI Bougares, HOLGER Schwenk and YOSHUA Bengio (2014), "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation", *In*, *Proceeding of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Eds. Alessandro Moschitti, Bo Pang, Walter Daelemans, pp.1724-1734, Association for Computational Linguistics, Doha, DOI: 10.3115/v1/D14-1179.

[8] MASHOOD NASIR Inzamam, ATTIQUE KHAN Muhammad, ALHAISONI Majed, SABA Tanzila, REHMAN Amjad and IQBAL Tassawar (2020), "A Hybrid Deep Learning Architecture for the Classification of Superhero Fashion Products: An Application for Medical-Tech Classification", *Computer Modeling in Engineering and Sciences,* Vol. 24, No 3, pp. 1017-1033.

[9] KOLUKISA Burak, DEDETURK Bilge Kağan, HACILAR Hilal and GÜNGÖR Vehbi Çağrı (2024), "An Efficient Network Intrusion Detection Approach Based on Logistic Regression Model and Parallel Artificial Bee Colony Algorithm", *Computer Standards & Interfaces*, Vol. 89, p.103808, DOI: 10.1016/j.csi.2023.103808.

[10] DENG Ben, YANG Minghui, ZHOU Lin, WANG Haowei, YAN Rong, PENG Fangyu (2019), "Smoothed Particle Hydrodynamics (SPH) Simulation and Experimental Investigation on the Diamond Fly-Cutting Milling of Zirconia Ceramics", *Proc CIRP,* Vol. 82, pp. 202–207, DOI: 10.1016/j.procir.2019.04.001.

[11] CUNNINGHAM Padraig, CORD Matthieu, and DELANY Sarah Jane (2008), *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*, Springer, Berlin.

[12] WASILEWSKA Monika and GOLENKO Bartomiej (2019), "Convolutional Neural Network Based Vehicle Classification", *2019 Signal Processing Symposium*, pp. 295-299, Krakow DOI: 10.1109/SPS.2019.8882050.

[13] BALID Walid, TAFISH Hasan and REFAI Hazem H. (2018), "Intelligent Vehicle Counting and Classification Sensor for Real-Time Traffic Surveillance", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 19, Issue 6, pp. 1784-1794 DOI: 10.1109/TITS.2017.2741507.

[14] IWATA, Motoi, ITO Atsushi and KISE Koichi (2014), "A Study to Achieve Manga Character Retrieval Method for Manga Images", *2014 11th IAPR International Workshop on Document Analysis Systems*, pp. 309-313, Tours.

[15] KOLUKISA Burak, GÖRMEZ Yasin and AYDIN Zafer (2023), "A Transfer Learning Approach for Skin Cancer Subtype Detection", *In, 4th International Conference on Artificial Intelligence and Applied Mathematics in Engineering ICAIAME 2022*, Eds. Jude Hemant, Tuncay Yiğit, Utku Köse, Umut Güvenç, pp. 337-347, Springer, Cham, DOI: 10.1007/978-3-031-31956-3_28.

[16] Pınar A., Çolak C., and Gültürk E. (2023), "Evaluation of Performance Metrics in Heart Disease by Machine Learning Techniques", *Journal of Health and Medical Sciences*, Vol 6, Issue 2, pp. 1-10.

[17] CHANG Jianlong, WANG Lingfeng, MENG Gaofeng, XIANG Shiming and PAN Chunhong (2018), "Vision-Based Occlusion Handling and Vehicle Classification for Traffic Surveillance Systems", *IEEE Intelligent Transportation Systems Magazine*, Vol. 10, Issue 2, pp. 80-92, DOI: 10.1109/Mits.2018.2806619.

[18] FAWCETT Tom (2006), "An Introduction to Roc Analysis", *Pattern Recognition Letters*, Vol. 27, No 8, pp. 861–874.

[19] ÜNVER Özkan and GAMGAM Hamza (2008), *Uygulamalı Temel İstatiksel Yöntemler*, Ankara Birlik Kitabevi, Ankara.

[20] CHOLLET François (2021), *Python ile Derin Öğrenme*, Buzdağı Yayınevi, Ankara.

[21] CHEN Tianqi and GUESTRIN Carlos (2016), "Xgboost: A Scalable Tree Boosting System", *In, Proceedings of The Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, Association for Computing Machinery, New York, DOI: 10.1145/2939672.2939785.

[22] NIR Oron, RAPOPORT Gal and SHAMIR Ariel (2022), "Cast: Character Labeling in Animation Using Self-Supervision by Tracking", *Computer Graphics Forum*, Vol. 41, No 2, pp. 135-145.

[23] KORKMAZ Mehmet Erdi and GÜNAY Mustafa (2018), "Finite Element Modelling of Cutting Forces and Power Consumption in Turning of Aisi 420 Martensitic Stainless Steel", *Arabian Journal for Science and Engineering*, Vol. 43, pp. 4863–4870, DOI: 10.1007/S13369-018-3204-4.

[24]. KASONGO Sydney M. and SUN Yanxia (2020), "Performance Analysis of Intrusion Detection Systems Using A Feature Selection Method On The Unsw-Nb15 Dataset", *Journal Of Big Data*, Vol. 7, No 1, DOI: 10.1186/S40537-020-00379-6.

[25] NGUYEN Nhu-Van, RIGAUD Christophe and BURIE Jean Christophe (2019), "Comic Mtl: Optimized Multi-Task Learning for Comic Book Image Analysis", *International Journal on Document Analysis and Recognition (Ijdar)*, Vol. 22, No 3, pp. 265-284.

[26] CENGIZ Enes (2020), *Yapay Zeka Tabanlı Drone Optimizasyonu* (Master's Thesis), Gazi University, Ankara.

[27] ALZAHRANI Abdulsalam O. And ALENAZI Mohammed J. F. (2021), "Designing A Network Intrusion Detection System Based on Machine Learning for Software Defined Networks", *Future Internet*, Vol 13, No 5, Article Number 111, DOI: 10.3390/fi13050111.

[28] BAYKAL Elif, DOĞAN Hülya, ERCİN Mustafa Emre, ERSÖZ Şafak and EKİNCİ Murat (2020), "Transfer Learning with Pre-Trained Deep Convolutional Neural Networks for Serous Cell Classification", *Multimedia Tools and Applications*, Vol. 79, pp. 21–22, DOI: 10.1007/S11042-019-07821-9.

[29] CANU Sergio (2023), *Flatten and Dense Layers*, https://Pysource.Com/2022/10/07/Flatten-And-Dense-Layers-Computer-Vision-With-Keras-P-6/, DoA. 5.12.2024.

[30] TANG An, TAM Roger, CADRIN-CHENEVERT Alexandre, GUEST Will, CHONG Jaron, BARFETT Joseph, CHEPELEV Leonid, CAIRNS Robyn, MITCHELL J. Ross, CICERO Mark D., POUDRETTE Manuel Gaudreau, JAREMKO Jacob L., REINHOLD Caroline, GALLIX Benoit, GRAY Bruce and GEIS Raym (2018), "Canadian Association of Radiologists White Paper on Artificial Intelligence in Radiology", *Canadian Association of Radiologists Journal*, Vol. 69, pp. 120-135.

[31] BHATTACHARYYA Bijot and DOLOI Biswanath (2020), "Machining Processes Utilizing Chemical and Electrochemical Energy", *Mod Mach Technol*, DOI: 10.1016/B978-0-12-812894-7.00005-0.

[32] KATYAL Kamaldeep, DUTTA Maitreyee and GRANJAL Jorge (2020), "Towards A Secure Internet of Things: A Comprehensive Study of Second Line Defense Mechanisms", *IEEE Access*, Vol 8, pp. 127272–127312, DOI: 10.1109/Access.2020.3005643.

[33] BARKANA Buket D., SARIÇİÇEK İnci and YILDIRIM Burak (2017), "Performance Analysis of Descriptive Statistical Features in Retinal Vessel Segmentation via Fuzzy Logic, Ann, Svm, and Classifier Fusion", *Knowledge-Based Systems*, Vol. 118, pp. 165-176.

[34] ÖZKESİCİ Muhammed Yasir and YILMAZ Selmi (2021), "Oral ve Maksillofasiyal Radyolojide Yapay Zeka", *Sağlık Bilimleri Dergisi*, Vol. 30, No 3, pp. 346-351, DOI: 10.34108/Eujhs.1040476.

[35] QIN Xiaoran, ZHOU Yafeng, HE Zheqi, WANG Yongtao and TANG Zhi (2017), "A Faster R-CNN Based Method for Comic Characters Face Detection", *In, 2017 14th Iapr International Conference on Document Analysis and Recognition (Icdar)*, Vol. 1, pp. 1074-1080, IEEE, DOI: 10.1109/ICDAR.2017.178.

[36] BOTTORE Marco, CHARA B. Dalla and DEFLORIO Francesco Paolo (2013), "Wireless Sensor Networks for Traffic Monitoring in A Logistic Centre", *Transportation Research Part C: Emerging Technologies*, Vol. 26, pp. 99-124, DOI: 10.1016/J.Trc.2012.06.008.

[37] RANI Manisha, GAGANDEEP (2021), "Employing Artificial Bee Colony Algorithm for Feature Selection in Intrusion Detection System", *8th International Conference on Computing for Sustainable Global Development*, pp. 496-500, New Delhi.

[38] HAYKIN Simon (1998), *Neural Networks: A Comprehensive Foundation*. Prentice Hall, Canada.

[39] DONG Honghui, WANG Xuzhao, ZHANG Chao, HE Ruisi, JIA Limin and QIN Yong (2018), "Improved Robust Vehicle Detection and Identification Based on Single Magnetic Sensor", *IEEE Access*, Vol. 6, pp. 5247-5255 DOI: 10.1109/Access.2018.2791446.

[40] PEIXOTO Francke (2023), *A Simple Overview of Multilayer Perceptron (MLP)*, https://www.analyticsvidhya.com/blog/2020/12/mlp-multilayer-perceptron-simple-overview, DoA. 7.12.2024.

[41] KOCHENDERFER Mykel J. and WHEELER Tim A. (2019), *Algorithms for Optimization*, The Mit Press, England.

[42]. ZOU Zhengxia, CHEN Keyan, SHI Zhenwei, GUO Yuhong and YE Jieping (2023), "Object Detection in 20 Years: A Survey", *Proceedings of the IEEE*, Vol. 111, No 3, pp. 257-276.

[43] LOPEZ MARTIN Manuel, CARRO Belen, SANCHEZ-ESGUEVILLAS Antonio and LLORET Jaime (2019), "Shallow Neural Network with Kernel Approximation for Prediction Problems in Highly Demanding Data Networks", *Expert Systems with Applications*, Vol. 124, pp. 196-208, DOI: 10.1016/J.Eswa.2019.01.063.

[44] XU Shuyuan, WANG Jun, SHOU Wenchi, NGO Tuan, SADICK Abdul-Manan and WANG Xiangyu (2021), "Computer Vision Techniques in Construction: A Critical Review", *Archives of Computational Methods in Engineering*, Vol. 28, pp. 3383-3397.

[45] BREIMAN Leo (2001), "Random Forests", *Machine Learning*, Vol. 45, No 1, pp. 5–32, DOI: 10.1023/A:1010933404324.

[46] BENGIO Yoshua (2012), "Practical Recommendations for Gradient-Based Training of Deep Architectures", *In, Neural Networks: Tricks of the Trade*, Second Edition, Eds. Grégoire Montavon, Geneviève B. Orr, Klaus-Robert Müller, pp. 437-478, Springer.

[47] JAMALI Mohammad Ali Jabraeil, BAHRAMI Bahareh, HEIDARI Arash, ALLAHVERDİZADEH Parisa and NOROUZI Fardah (2020), *Towards the Internet of Things*, Springer, Berlin, DOI: 10.1007/978-3-030-18468-1.

[48] LIN Tsung-Yi, GOYAL Priya, GIRSHICK Ross, HE Kaiming and DOLLAR Piotr (2020), "Focal Loss for Dense Object Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 42, No 2, pp. 318-327, DOI: 10.1109/Tpami.2018.2858826.

[49] ZUO Chao, QIAN Jiaming, FENG Shijie, YIN Wei, LI Yixuan, FAN Pengfei, HAN Jing, QIAN Kemao and CHEN Qian (2022), "Deep Learning in Optical Metrology: A Review", *Light: Science and Applications*, Vol. 11, No 1, pp. 1-54.

[50] HO Hoang Nam, RIGAUD Christophe, BURIE Jean-Christophe and OGIER Jean-Marc (2013), "Detecting Recurring Deformable Objects: An Approximate Graph Matching Method for Detecting Characters in Comics Books", *In, Graphics Recognition. Current Trends and Challenges*, Eds. Bart Lamiroy, Jean-Marc Ogier, pp. 122-134, Springer, Berlin.

[51] MARTIN Scott (2023), *What's The Difference Between A CNN And An RNN*, https://www.edge-ai-vision.com/2018/09/whats-the-difference-between-a-cnn-and-an-rnn, DoA. 6.12.2024.

[52] DANISH Mohd, GINTA Turnad Lenggo, RANI Ahmad Majdi Abdul, CAROU Diego, DAVIM J. P., RUBAIEE Saeed and GHAZALI Sami (2019), "Investigation of Surface Integrity Induced on AZ31C Magnesium Alloy Turned Under Cryogenic and Dry Conditions", *Procedia Manufacturing*, Vol. 41, pp. 476–483, DOI: 10.1016/J.Promfg.2019.09.035.

[53] MEFTAH Souhail, RACHIDI Tajjeeddine and ASSEM Nasser (2019), "Network Based Intrusion Detection Using the Unsw-Nb15 Dataset", *International Journal of Computing and Digital Systems*, Vol. 8, No 5, DOI: 10.12785/Ijcds/080505.

[54] DEDETÜRK Bilge Kağan and AKAY Bahriye (2020), "Spam Filtering Using A Logistic Regression Model Trained by an Artificial Bee Colony Algorithm", *Applied Soft Computing Journal*, Vol. 91, p. 106-229, DOI: 10.1016/J.Asoc.2020.106229.

[55] NIESLONY P., KROLCZYK Grzegorz M., WOJCIECHOWSKI Szymon, CHUDY Roman, ZAK K., MARUDA R. W. (2018), "Surface Quality and Topographic Inspection of Variable Compliance Part After Precise Turning", *Applied Surface Science,* Vol. 434, pp. 91–101.

[56] AFUWAPE Afeez Ajani, XU Ying, ANAJEMBA Joseph Henry and SRIVASTAVA Gautam (2021), "Performance Evaluation of Secured Network Traffic Classification Using A Machine Learning Approach", *Computer Standards & Interfaces*, Vol. 78, p. 103-545, DOI: 10.1016/J.Csi.2021.103545.

[57] MURUGAN Pushparaja and DURAIRAJ Shanmugasundaram (2017), "Regularization and Optimization Strategies in Deep Convolutional Neural Network", *CoRR*, Vol. abs/1712.04711, DOI: 10.48550/arXiv.1712.04711.

[58] NASTESKI Vladimir (2017), "An Overview of The Supervised Machine Learning Methods", *Horizons B.*, Vol 4, pp. 51-62, DOI: 10.20544/Horizons.B.04.1.17.P05.

[59] NGUYEN Nhu-Van, RIGAUD Christophe and BURIE Jean-Christophe (2017), "Comic Characters Detection Using Deep Learning", *2017 14th Iapr International Conference on Document Analysis and Recognition (Icdar)*, pp. 41-46, Kyota.

[60] NGUYEN Nhu-Van, RIGAUD Christophe and BURIE Jean-Christophe (2018), "Digital Comics Image Indexing Based on Deep Learning", *Journal Of Imaging*, Vol. 4, No 7, p. 89.

[61] OYEBODE Oluwaseun and IGHRAVWE Desmond Eseoghene (2019), "Urban Water Demand Forecasting: A Comparative Evaluation of Conventional and Soft Computing Techniques", *Resources*, Vol. 8, No 3, Article Number 156, DOI: 10.3390/Resources8030156.

[62] AKAY Bahriye and KARABOĞA Dervis (2012), "A Modified Artificial Bee Colony Algorithm for Real-Parameter Optimization", *Information Sciences*, Vol. 192, pp. 120-142, DOI: 10.1016/J.Ins.2010.07.015.

[63] ÖZKAN Mustafa and KAR Görkem (2022), "Türkçe Dilinde Yazılan Bilimsel Metinlerin Derin Öğrenme Tekniği Uygulanarak Çoklu Sınıflandırılması", *Mühendislik Bilimleri ve Tasarım Dergisi*, Vol. 10, No 2, pp. 504-519.

[64] WAN Min, YE Xiang-Yu, YANG Yun and ZHANG Wei-Hong (2017), "Theoretical Prediction of Machining-Induced Residual Stresses in Three-Dimensional Oblique Milling Processes", *Int J Mech Sci*, Vol. 133, pp. 426–437, DOI: 10.1016/J.Ijmecsci.2017.09.005.

[65] YANAGISAWA Hideaki, YAMASHITA Takuro and WATANABE Hiroshi (2018), "A Study on Object Detection Method from Manga Images Using CNN", *2018 International Workshop on Advanced Image Technology (Iwait)*, pp. 1-4, Chiang Mai.

[66] DANISH Mohd, GINTA Turnad Lenggo, HABIB Khairul, RANI Ahmad Majdi Abdul and SAHA Bidyut Baran (2019), "Effect of Cryogenic Cooling on The Heat Transfer During Turning of AZ31C Magnesium Alloy", *Heat Transfer Engineering,* Vol. 40, No. 12, pp. 1023–1032.

[67] HARRIS Charles R., MILLMAN K. Jarrod, VAN DER WALT Stéfan J., GOMMERS Ralf, VIRTANEN Pauli, COURNAPEAU David, WIESER Eric, TAYLOR Julian, BERG Sebastian, SMITH Nathaniel J., KERN Robert, PICUS Matti, HOYER Stephan, VAN KERKWIJK Marten H., BRETT Matthew, HALDANE Allan, FERNÁNDEZ DEL RÍO Jaime, WIEBE Mark, PETERSON Pearu, GÉRARD-MARCHANT Pierre, SHEPPARD Kevin, REDDY Tyler, WECKESSER Warren, ABBASI Hameer, OLIPHANT Travis E. (2020), "Array Programming with Numpy", *Nature,* Vol. 585, pp. 357-362, DOI: 10.1038/S41586-020-2649-2.

[68] AIZAWA Kiyoharu, FUJIMOTO Azuma, OTSUBO Atsushi, OGAWA Toru, MATSUI Yusuke, TSUBOTA Koki, IKUTA Hikaru. (2020), "Building A Manga Dataset "Manga109" with Annotations for Multimedia Applications", *IEEE Multimedia*, Vol. 27, No 2, pp. 8-18.

[69] ESPINOSA Roberto, GACIA-SAIZ Diego, ZORRILLA Marta, ZUBCOFF Jose Jacobo and MAZON Jose Norberto (2019), "S3mining: A Model-Driven Engineering Approach for Supporting Novice Data Miners in Selecting Suitable Classifiers", *Computer Standards & Interfaces*, Vol. 65, PP. 143-158 DOI: 10.1016/J.Csi.2019.03.004.

[70] GEORGE Jobin, MARY Leena and RIYAS K. S. (2013), "Vehicle Detection and Classification from Acoustic Signal Using ANN and KNN", *2013 International Conference on Control Communication and Computing*, pp. 436-439, India, DOI: 10.1109/Iccc.2013.6731694.

[71] KIM Hayeon, LEE Eun-Cheol, SEO Yongseok, IM Dong-Hyuch and LEE In-Kwon (2020), "Character Detection in Animated Movies Using Multi-Style Adaptation and Visual Attention", *IEEE Transactions on Multimedia*, Vol. 23, pp. 1990-2004.

[72] SARKAR Dipanjan, BALI Raghav and GHOSH Tamoghna (2018), *Hands-On Transfer Learning with Python: Implement Advanced Deep Learning and Neural Network Models Using Tensorflow and Keras*, Packt Publishing, Birmingham.

[73] SHARMA Neha, SHARMA Reecha and JINDAL Neeru (2021), "Machine Learning and Deep Learning Applications-A Vision", *Global Transitions Proceedings*, Vol. 2, No 1, pp. 24-28.

[74] MALATO Gianluca (2021), *Hyperparameter Tuning. Grid Search and Random Search | Your Data Teacher*, https://www.yourdatateacher.com/2021/05/19/hyperparameter-tuning-grid-search-and-random-search, DoA. 9.12.2024.

[75] ISMAIL Asmida, AHMAD Siti, SOH Azura Che and HASSAN Mohd Khair (2019), "Improving Convolutional Neural Network (CNN) Architecture (Minivggnet) with Batch Normalization and Learning Rate Decay Factor for Image Classification", *The International Journal of Integrated Engineering*, Vol. 11, No 4, pp. 51-59.

[76] QURESHI Ayyaz-UI-Haq, LARIJANI Hadi, MTETWA Nhamoinesu, JAVED Abbas and AHMAD Jawad (2019), "RNN-ABC: A New Swarm Optimization Based Technique for Anomaly Detection", *Computers*, Vol. 8, No 3, Article Number 59, DOI: 10.3390/Computers8030059.

[77] RANI Manisha, GAGANDEEP (2022), "Effective Network Intrusion Detection by Addressing Class Imbalance with Deep Neural Networks Multimedia Tools and Applications", *Multimedia Tools and Applications*, Vol. 81, No 6, pp. 8499-8518, DOI: 10.1007/S11042-021-11747-6.

[78] ASBORNO Magdalena I., BURRIS Colling G. and HERNANDEZ Sarah (2019), "Truck Body-Type Classification Using Single-Beam Lidar Sensors", *Transportation Research Record*, Vol. 2673, No 1, pp. 26-40, DOI: 10.1177/0361198118821847.

[79] BAECK Thomas, FOGEL D. B., MICHALEWICZ Zbigniew (2018), *Evolutionary Computation 1: Basic Algorithms and Operators*, CRC Press, Florida.

[80] BALASARASWATHI Veeran Ranganathan, SUGUMARAN Muthukumarasamy and HAMID Yasir (2017), "Feature Selection Techniques for Intrusion Detection Using Non-Bio-Inspired and Bio-Inspired Optimization Algorithms", *Journal of Communications and Information Networks*, Vol. 2, No 4, pp. 107-119, DOI: 10.1007/S41650-017-0033.