Computational Engineering Project Report

Machine Learning Assisted Optimization of Hardening Parameters

Crista Ekblom, F. San Koktas and Akin Erdem

Aalto University

COMPUTATIONAL ENGINEERING PROJECT

Aalto University
Computational Engineering Project Report 2023

Machine Learning Assisted Optimization of Hardening Parameters

Crista Ekblom, F. San Koktas and Akin Erdem

Computational engineering project submitted in fulfillment of the requirements for the degree of Bachelor of Science in Technology.
Otaniemi, 15 Dec 2023

Supervisor: Junhe Lian

Aalto University
School of Engineering
Bachelor's Programme in Science and Technology

Author
Crista Ekblom, F. San Koktas and Akin Erdem

Title
Machine Learning Assisted Optimization of Hardening Parameters

School School of Engineering

Degree programme Bachelor's Programme in Science and Technology

Major Computational Engineering **Code** ENG3082

Supervisor Junhe Lian

Advisor Zinan Li

Date 15 Dec 2023 **Pages** 54 **Language** English

Abstract

Understanding and knowing the properties of materials is of huge importance for the development of new products. In certain kind of product development, it is crucial to have knowledge of what the so-called hardening parameters of the used materials are. These dictate how the material behaves and how well it can handle force loading, determining when plastic deformation is to occur. Different variations of the constitutive equations that describe this kind of behavior have been utilised in research with Swift-Voce law being a comprehensive one with seven hardening parameters. The problem of finding the values for these parameters is an inverse problem that has historically been solved with methods that are computationally time consuming (e.g. FEMU and VFM). With the recent rise in popularity of utilising machine learning in different areas of research, it as a tool to solve the inverse problem of finding the hardening parameters has been explored. It is of interest, because machine learning could provide a solution to the long computational times of methods previously used. Therefore, the aim of the research presented in this paper is to develop a machine learning model that is capable of predicting the hardening parameters for the Swift-Voce hardening law. The input data for training and validation constitutes of force-displacement data of steel DP1000. Based on previous literature, initially a neural network model is constructed. For the purpose of comparison of machine learning algorithms, alternative models utilising Random Forest and XGBoost algorithms are also built. The results obtained from these models suggest that machine learning models offer a rapid and computationally efficient approach to gain insights into potential values for hardening parameters. However, it is noteworthy that the accuracy of results may be constrained by the available data set size, emphasizing the need for further investigation and potentially larger data sets for enhanced model precision.

Keywords Machine learning, hardening parameters, inverse method, Swift-Voce

urn https://aaltodoc.aalto.fi

Contents

1 Introduction

This section gives a brief overview of the current research and provides the motivation for the research conducted and presented in this report (Subsection 1.1). Following this, the aim of the research and the specific research questions are presented and elaborated in Subsection 1.2.

1.1 Research overview and motivation

The development of new materials for the utilization of different industries is of great importance. These materials need to have certain kinds of properties to be of use. For instance, the automotive industry requires lightweight metals. This means that having knowledge of the properties of the materials that are potentially to be used is essential for the evaluation of their suitability for the product being developed. As an example, in the context of the automotive industry knowing the properties of the metals used for the building of the car is important when calculating how durable the car is in a crash scenario. Another example is airplanes, where the durability of the material used for the body of the plane is of importance combined with lightweightness. One component of knowing what the properties of a material are, is being informed of what the parameters of so-called constitutive equations for the material are. Constitutive equations in an engineering context describe the relationship between two physical quantities, e.g., the relationship between stress and strain. There are different kinds of constitutive equations used to describe this kind of relationship and these kind of equations are also known as hardening laws. A known hardening law that is used to describe the isotropic hardening is the so-called Swift-Voce law, which is an expansion of the Swift and Voce hardening laws.

If the parameters for these equations are not known, they can be found utilising an inverse method if, i.e., force-displacement data is available for the material specimen. Traditionally, the parameters for these laws have been identified with methods that require significant computational time. Examples of these methods are Constitutive Equation Gap Method (CEGM), Virtual Fields Method (VFM) and Finite Element Model Updating (FEMU). To make the parameter identification process faster, applying machine learning methods to solve the inverse problem has become more prominent in recent research. Machine learning has developed fast in the past decade and there are many algorithms suitable for the solving of the inverse problem that calibration of the parameters is.

Therefore, it is of interest to attempt to develop a machine learning model for the purpose of the identification of the hardening parameters and compare how this performs against more traditional methods of solving the inverse problem. This will be the focus of the research conducted and presented in this report. The following section goes in-depth about the aim and the specific research questions that shall be answered by the research conducted and presented in this paper.

1.2 Project aim and scope

The aim of the research presented in this report is to develop a machine learning-based model that can solve the inverse problem and calibrate the hardening parameters. The model is supposed to be developed in a way that it could be universally utilised for any kind of material. The model shall be trained with experimental data generated with the help of the Abaqus software.

A data set was provided to the project team in the beginning of the project, which consisted of different kinds of geometries in different temperatures for the steel material DP1000 with data of the specimens' forces and displacement. This provided data is meant to be used for the training and validation of the model. Finally, the developed model is to be compared with more traditional methods of solving the inverse method and evaluated how well it performs against those. Based on these, a few research questions are formulated to seek answers to. These are as follows:

- Can a machine learning model be developed to calibrate the hardening parameters for steel DP1000?
- How do the results of the developed machine learning model compare with a more traditional method of solving an inverse problem?

2 Theoretical Background

This section reviews the theoretical background of the research conducted and presented in this report. Section 2.1 discusses the hardening laws utilized in the research for this paper, following a brief introduction to what inverse methods are in Section 2.2. Section 2.3 gives a short account of what machine learning entails. Finally, Section 2.4 reviews the literature related to the machine learning-assisted optimization of hardening parameters.

2.1 Hardening Laws

Hardening laws are type of constitutive equations, which are used to describe the relationship between two physical quantities. Specifically, a hardening law (or rule) "describes the relationship of stress–strain to increase further the amount of stress to produce additional strain after having reached the elastic limit" [1]. There are different kinds of hardening rules. These differ based on what kind of hardening is desired to be investigated. Examples are kinematic and strain hardening. The hardening laws utilized in the research conducted for this paper are the Swift, Voce and Swift-Voce hardening laws. These laws can be used to describe the expansion of yield surface or in other words isotropic hardening [2]. These are each explained briefly in the following subsections.

Swift

The Swift hardening law is one of the most used models to describe flow stress-strain behavior in sheet metal forming simulations [3]. The law has three parameters and can be described by the below equation:

$$\sigma(\varepsilon) = C_1 \cdot (C_2 + \varepsilon)^{C_3}$$

Voce

The Voce hardening law differs slightly from the Swift law, but has similarly three parameters apart from the strain. The law can be described with the below equation:

$$\sigma(\varepsilon) = C_1 + C_2 \cdot (1 - \exp(-C_3 \cdot \varepsilon))$$

Swift-Voce

The Swift-Voce hardening law is an extension of both the Swift and Voce. This hardening law combines both laws into one by including a weight parameter. This means that the law has in total of seven parameters. Swift-Voce law can be seen below:

$$\sigma(\varepsilon) = C_1 \cdot \sigma_{\text{Swift}}(\varepsilon) + (1 - C_1) \cdot \sigma_{\text{Voce}}(\varepsilon)$$

2.2 Inverse Method

The inverse method is a technique where the input parameters for a model are estimated (with uncertainty) by comparing the model output magnitudes with experimental data [4]. This is the opposite of a direct or forward problem, where the parameters are known, and the predictions of observations can be predicted through the help of, e.g., physical laws. The inverse methods can be utilized for instance for the calibration of identifying parameters for flow curves of materials with describing the relationship between, e.g., stress and strain. This is also what is the aim of the research conducted in this paper. Other applications where inverse problems are present are for example computerized tomography, seismic imaging, electron microscopy, and magnetic resonance imaging [5]. Figure 1 displays the difference between the forward and the inverse problem and clearly shows how these two types of problems are opposites.

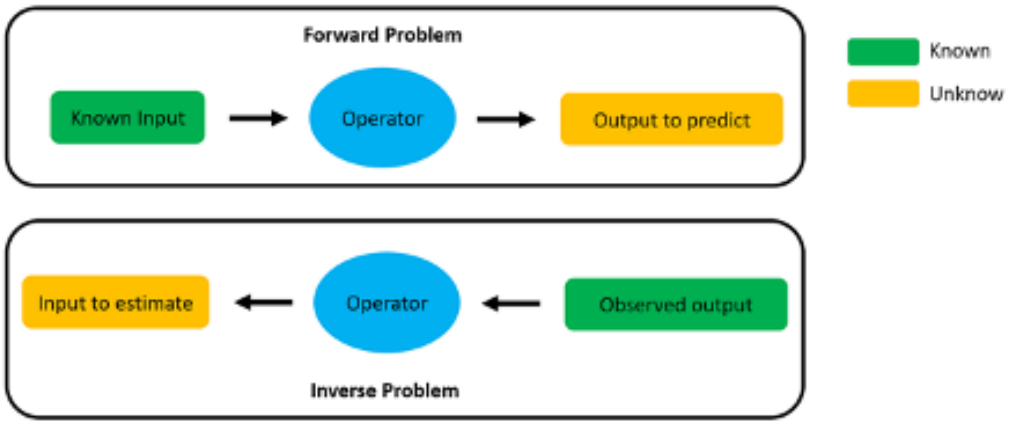

Figure 1. *Inverse Problem vs Forward Problem [6].*

2.3 Machine Learning

In recent years machine learning (ML) has become something of a hot topic and is currently being utilized and applied in many fields of research [7]. Terms like artificial intelligence and data mining have been used interchangeably with it. The concept entails the development of a computer system that learns and improves from data without having to be explicitly programmed. In simple terms, an algorithm is fed some data for training. The algorithm builds some mathematical model and gives an output. How the training process is done and whether outputs are known or not depends on the type of algorithm. This simplified process is illustrated by Figure 2.

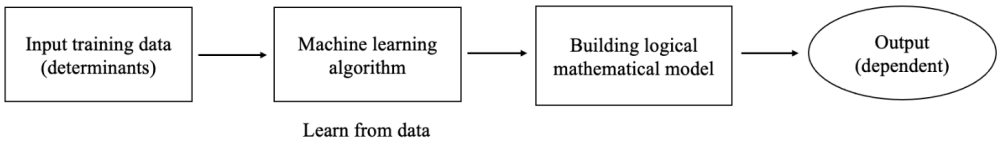

Figure 2. *Machine Learning Simplified.*

Machine learning algorithms can be classified into different types based on the learning process of the model. The four major categories are unsupervised learning, semi-supervised learning, supervised learning, and reinforcement learning [7]. Figure 3 displays these categories and what type of machine learning algorithms fall under each category with examples of applications. For the purpose of this project, the machine learning algorithm called Artificial Neural Network (ANN) is very much relevant. For the importance of understanding this crucial concept, the following subsection is dedicated to explaining this.

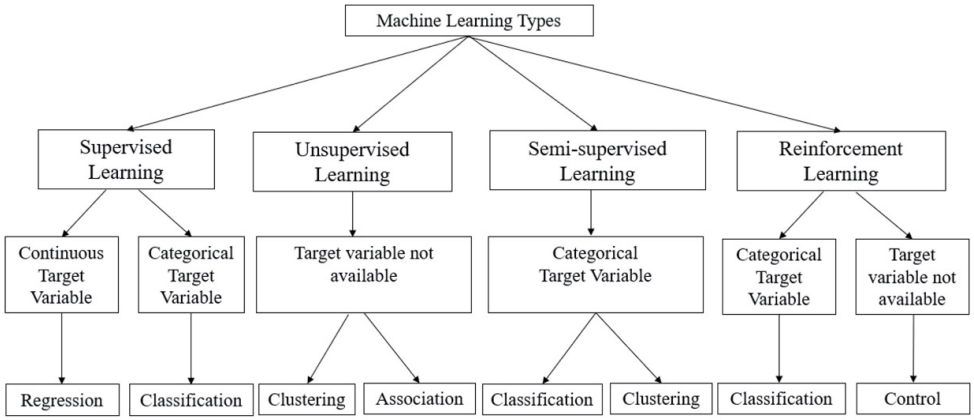

Figure 3. *Machine Learning Categorized [8].*

Artificial Neural Networks

Artificial Neural Networks (ANN) are a branch of machine learning algorithms. What is special about this type of machine learning algorithms is that they are constructed in a way that attempts to mimic the functionality of a human brain. They are also the basis of so-called deep learning algorithms. Figure 4 illustrates a simple structure of an ANN algorithm. This has the so-called input layer, following three hidden layers and the output layer. The algorithm consists of nodes. These mirror the nodes that can be found in the human brain. The different nodes can be assigned a different weight and threshold. The weights help to determine whether a variable is significant and the thresholds make sure that a node is not activated if not required. These are a part of something called the activation function, which determines whether a specific node is activated or not.

Another concept that is important for the function of an ANN algorithm is the loss (or cost) function. The loss function is essentially a function that compares the target and predicted output values. Thereby, it measures how well the neural network models the training data used for the model. Different kinds of loss functions can be utilized depending on the context which the developed model is used for. Some typically used examples of these are Mean Squared Error (MSE), Mean Absolute Error (MAE) and Binary-Cross Entropy [9].

All the so-called parameters mentioned in this section, which affect the structure of the ANN algorithm, are collectively known as hyperparameters. There are two different kinds of hyperparameters - model parameters and hyper-parameters [10]. Model parameters are the kind of hyperparameters that can be initialized and updated through the data learning process. Examples of these in the context of ANN are weights and nodes. Hyper-parameters on the other hand cannot be directly estimated from data learning and must be set before training a ML model because they define the architecture of the model. Examples of these for an ANN algorithm are the learning rate and activation function. The learning rate is a tuning parameter that determines the pace at which an algorithm updates or learns the values of a parameter estimate and is an important hyperparameter in the ANN context [11]. Hyperparameter tuning is the process of finding the best possible combination of hyperparameters for a ML model to get the most optimally functioning model. This is a process that was also crucial for the research conducted in this paper, which is discussed further in the following section about the methodology.

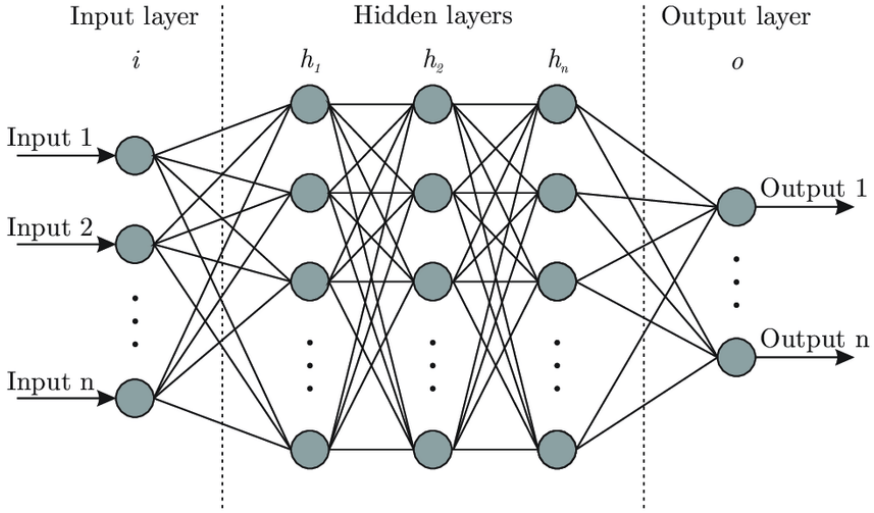

Figure 4. *Structure of a artificial neural network [12].*

2.4 Literature Review

Knowing the mechanical properties of materials is of increased importance – especially with the development of new materials for the use of e.g., automotive and aerospace industries [13]. To identify for example the hardening parameters of a material methods like the Constitutive Equation Gap Method (CEGM), Virtual Fields Method (VFM), and Finite Element Model Updating (FEMU) have been utilized to solve the inverse problem where the displacement field has been obtained and is used as input [14]. The FEMU method is illustrated in Figure 5 below and the principle of it is to iteratively update the material hardening parameters of a finite element model in order to minimize a functional based on measured fields and simulated fields with an adequate minimization algorithm [15]. The problem with all these methods just mentioned is that they cannot handle complex heterogeneity or thickness variations in the material and the cost of computations can become very high [14]. This is why with the fast development of machine learning, applying ML methods for the calibration of hardening parameters has become more prominent in research in recent years. The motivation for this has to do with decreased cost and the ability to have the calibration done faster than with other types of methods used for solving inverse problems [14].

Lourenço et al. [16] describe different scenarios in metal forming applications where machine learning algorithms can be utilized. One of the applications they discuss, is the application of machine learning for parameter identification with inverse modeling. A case study is conducted where an ANN-based algorithm is developed to find the parameters for Swift's hardening law and Hill48 anisotropic criterion. What is remarked by the authors is that the reliability of the developed

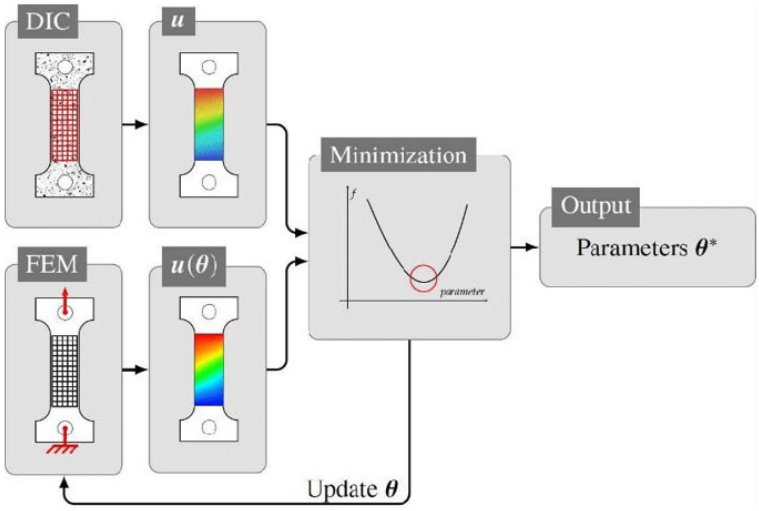

Figure 5. *Process of FEMU [15].*

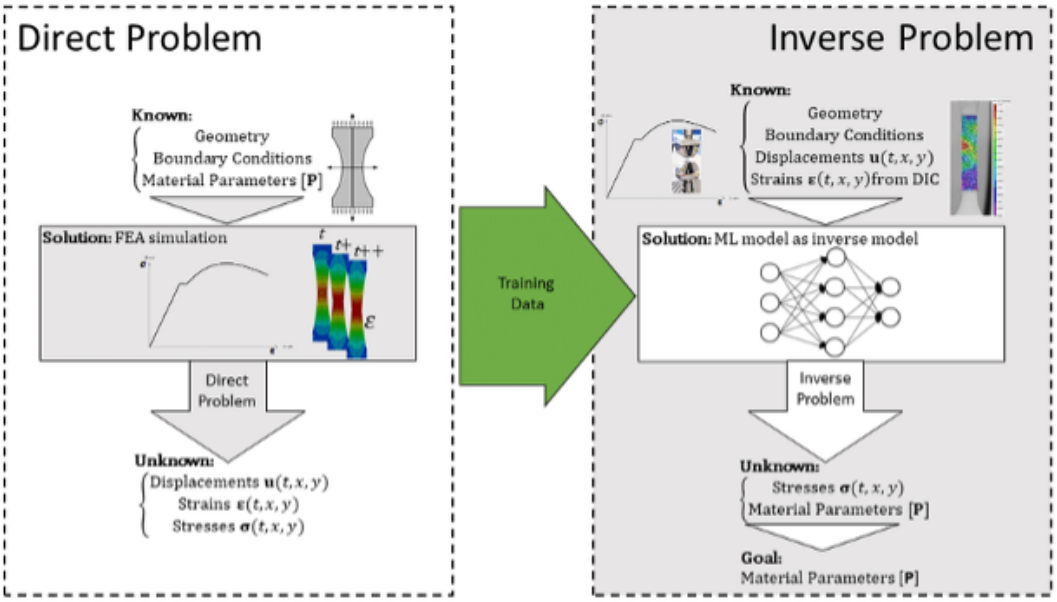

Figure 6. *ML as an inverse problem for calibration of parameters [16]*

model is highly dependent on the quantity and quality of the training data used. The process of this case study conducted by Lourenço et al. can be seen illustrated in Figure 6.

Yao et al. [17] conduct a similar kind of process by identifying the parameters for the Swift hardening law for 6061 aluminum alloy with the help of a backpropagation neural network algorithm. Data from different specimens with different geometries of the researched aluminum alloy were used for the developed neural network model. It was found that the model could predict quite well the parameters for the material in comparison to an experimental-numerical hybrid model. The significant benefit was the decrease in required computational time

where CPU time for the machine learning model was 192.5 h in comparison to the experimental-numerical hybrid method where this was 296.7 h.

Zhang et al. [14] develop a machine-learning model for the mechanical characterization of heterogeneous membranes. They compare their results with the results obtained by traditional FEMU. A lightweight Fully Connected Neural Network (FCNN) is utilized to solve the inverse problem where the goal is to identify Young's modulus for the material. It is found that the FCNN model can predict the value as accurately as the FEMU methodology, but the computational time is 6 orders of magnitude less than for the FEMU. The prediction ability is seen to improve in the future by the authors with the improvement of the structure to the used neural network algorithm making the use of this kind of methodology for the calibration of the material parameters even more compelling.

Neural network-based algorithms are often favored in research as the choice of a machine learning algorithm for parameter identification as demonstrated by the already mentioned examples. Others that have used a similar type of methodology in developing a neural network model are Lin et al. [18], Guo et al. [19], Wang et al. [20], Chen et al. [21] and Hajari et al. [22]. Liu et al. [23] review the use of ANN in constitutive modeling for composite materials and state that the reasons why this type of machine learning algorithm is used widely in research for material behavior have to do with the fact that these algorithms perform well with growing data, are able to approximate complex nonlinear relations and there are many advanced open-source libraries for this type of algorithms (e.g. Tensorflow and PyTorch).

An alternative machine learning algorithm to an ANN-based one for material parameter identification is presented by Bastos et al. [24]. They use the Extreme Gradient Boosting (XGBoost) model for the parameter calibration to find the Swift law parameters. XGBoost is an algorithm that is decision-tree-based and utilizes so-called boosting in its learning. Another example of a decision-tree-based algorithm is Random Forest, which is essentially a less improved version of an XGBoost algorithm that utilizes bagging instead of boosting. Both of these algorithms are categorized under the larger umbrella of supervised learning [25].

The model developed by Bastos et al. [24] is tested with multiple different sizes of datasets. It is found that the results improve with the increased number of samples in the training dataset, but that 1500 sample set already gives good results from the model. To improve the computational time of the model since the time required for training was huge, a so-called Principal Component Analysis

(PCA) was conducted to reduce the dimensionality of the data set used for the training.

Utilizing machine learning in material parameter identification is still a relatively new development. Improvements to the models are to be expected with the further development of machine learning methods. At current ANN-based models are dominant within the field of parameter identification for constitutive models, although as Bastos et al. [24] demonstrate, alternative methods can be deployed. The main objective of the research presented in this paper is to develop an ANN-based algorithm to identify the hardening parameters for DP1000 steel sheets with different geometries. The following section details the methodology for the research conducted.

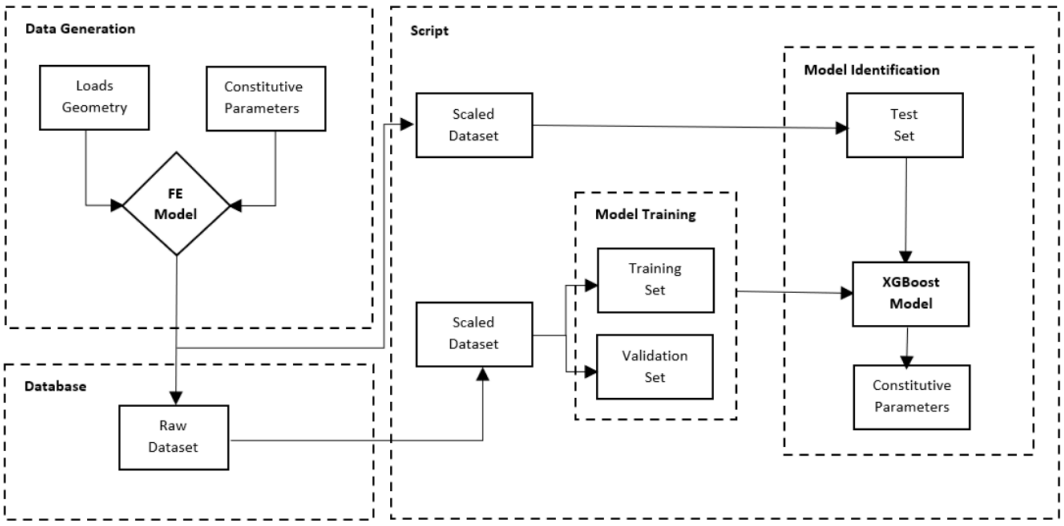

Figure 7. *Framework that Bastos et al. used for their XGBoost model [24]*

3 Methodology

This section presents first in Section 3.1 the initial workflow for the research. Following this Section 3.2 describes the data utilized. Section 3.3 discusses the phases in the configuration of the developed machine learning models with bullet points. Section 3.4 elaborates on Section 3.3 by providing a detailed explanation of every phase. Sections 3.5 and 3.6 discuss alternative models for the neural network model, namely Random Forest and XGBoost.

3.1 Initial workflow

Initially, a substantial amount of time was used to understand the research problem. This proved to be quite challenging for the project team, due to the members having inadequate background knowledge on the topic. What provided to be helpful for the understanding of the problem and what the team should strive to achieve, was conducting the literature review on the topic during the first weeks of the project work. Additionally, investigating and understanding the structure of the provided data proved also to be a bit of a challenge. The data is discussed in more detail in the following Section 3.2. The support from the advisor was an additional help in improving the teams overall understanding of the data and the problem at hand.

After conducting the literature review, understanding the problem statement and formulating the research questions, the initial choice of a machine learning algorithm was made. The choice was made to develop an artificial neural network model as this was supported by the literature. Because the team lacked previous knowledge of how to do this, a dummy ANN model was developed just to understand how this kind of modelling works. The more detailed discussion of the model development can be found in Section 3.3. In this section both the development of the dummy model and the actual model for the project is presented. Figure 8 below illustrates the initial workflow for the project.

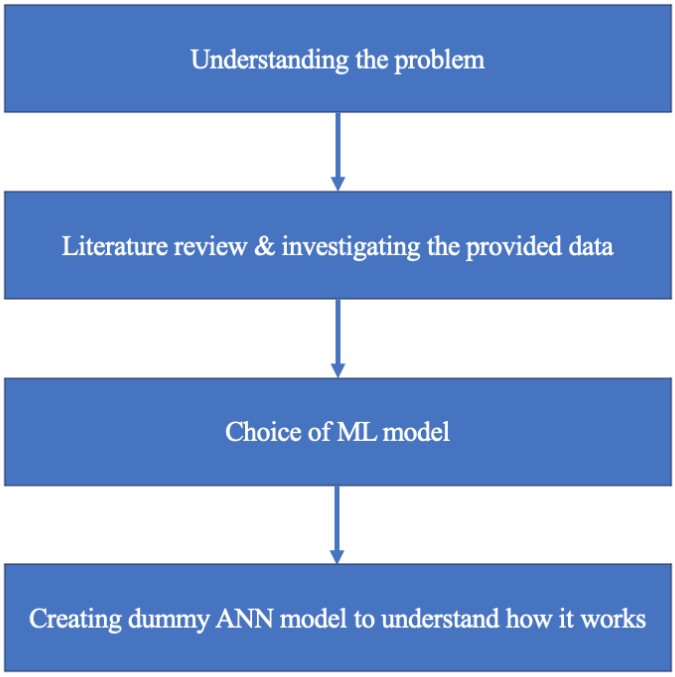

Figure 8. *Initial workflow of the project work*

3.2 Data

The project team was provided with force-displacement data based on random hardening parameters which have been generated through ABAQUS simulations. Furthermore, experimental force-displacement data for all geometries, which were used to compare the project team's obtained predicted results, were also provided. Both the simulation and experimental data were given separately for 25°C and 400°C. The only exception was that experimental force-displacement data for NDB20 was not provided for 400°C.

```
data for training/
  (hardening parameters) initial guesses/
  (hardening parameters) iteration guesses (400C)/
  25C experimental force-displacement curves/
  400C experimental force-displacement curves/
```

Each of the four subfolders contained data for CHD6, NDBR6, NDBR20, and NDBR50 geometries, primarily comprising extensive `.npy (NumPy)` and `.csv` files. Cleaning the data, particularly the `.npy` files, posed certain challenges.

NumPy files in the "(hardening parameters) initial guesses" folder contained the data needed to train the developed model. To gain insight into the .npy file

structures, arrays were printed and inspected. These arrays followed a consistent pattern, starting with c1, c2, c3, c4, c5, c6, and c7 values representing hardening parameters, followed by dictionaries for force and displacement. This pattern recurred consistently throughout the data.

Working with the data stored in both .csv and .npy files presented challenges due to discrepancies in the number of columns and rows across the files that were processed. To address this, the following steps were undertaken:

1. The standardized .npy files were converted into .csv format to enhance compatibility with the local environment, such as Visual Studio. The .csv files provide a more accessible means of visualization and analysis.

2. The hardening parameter file was set up with 7 columns (c1-c7) for individual parameters. In the file for force-displacement (f/d) data, to keep it simple and straightforward two columns were used: one for force and one for displacement.

3. The number of rows across the x and y files were balanced by duplicating data points within the file containing hardening parameters.

With an equal number of rows in both data sets, data analysis proceeded. The hardening parameters (c1 to c7) were placed as the output, while (force and displacement) were placed as the input.

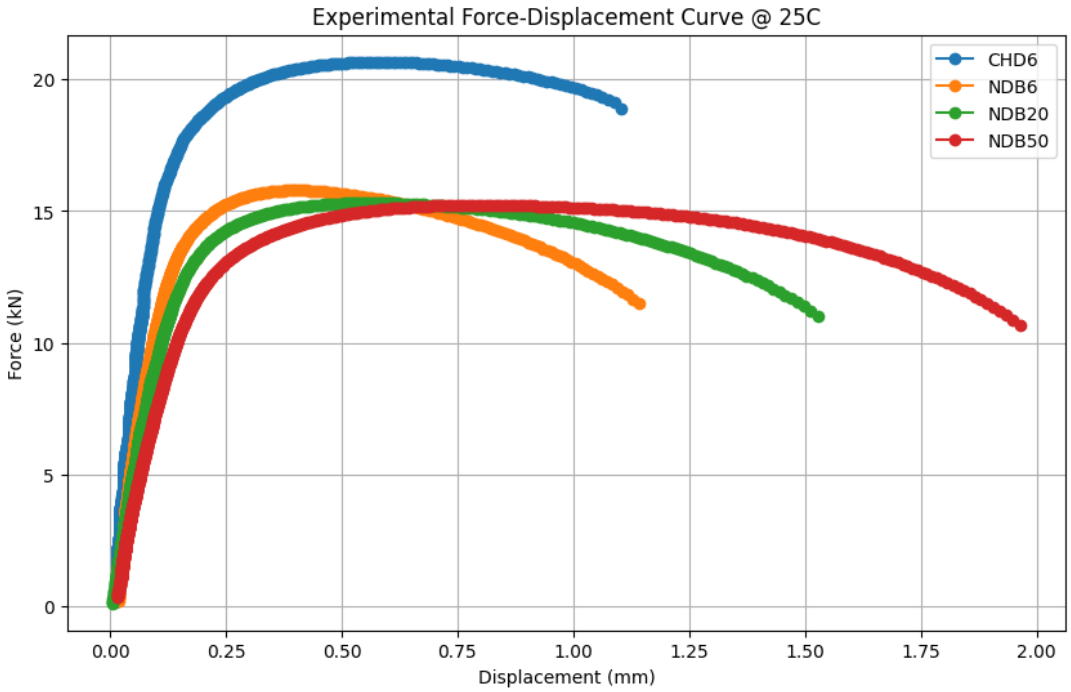

Figure 9. *Force-displacement curves of provided experimental data (25°C)*

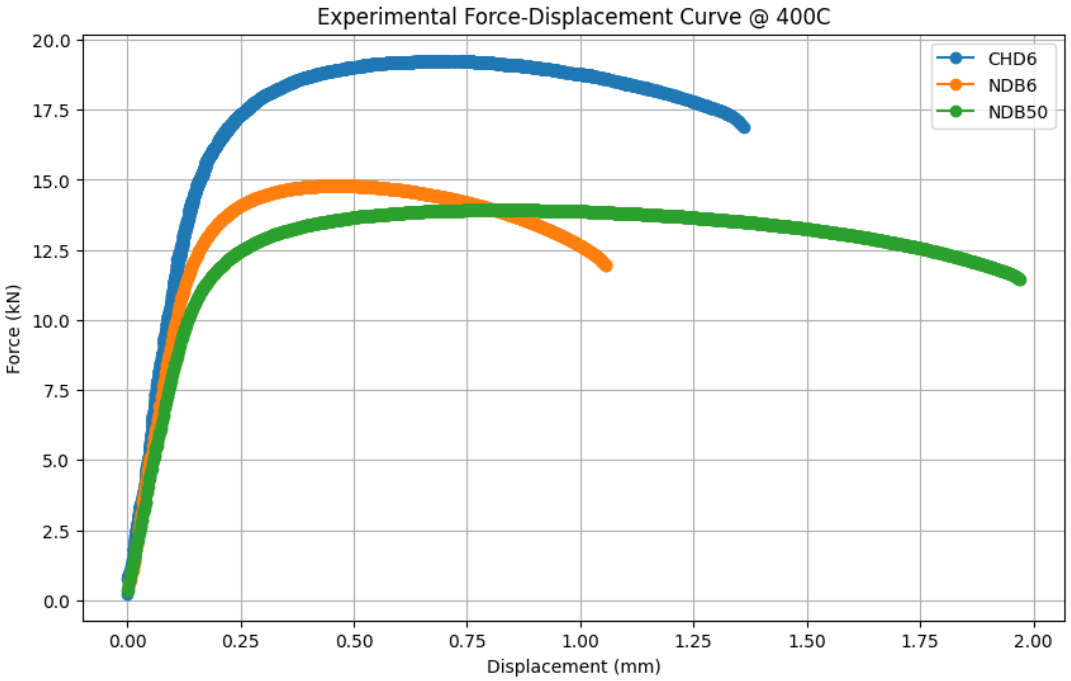

Figure 10. *Force-displacement curves of provided experimental data (400°C)*

3.3 Configuration of the Model

As already discussed in Section 3.1, a dummy neural network model was initially built to understand how this kind of algorithm is built and how it works. In other words, the project process followed an iterative approach, with each phase building upon the lessons learned from the preceding one. After the dummy neural

network model had been built, the work for developing a neural network model was initiated. Figure 11 illustrates the different phases of development. Following this, the meaning of phases is shortly explained in bullet points. A more thorough explanation of each phase is presented in Subsection 3.4.

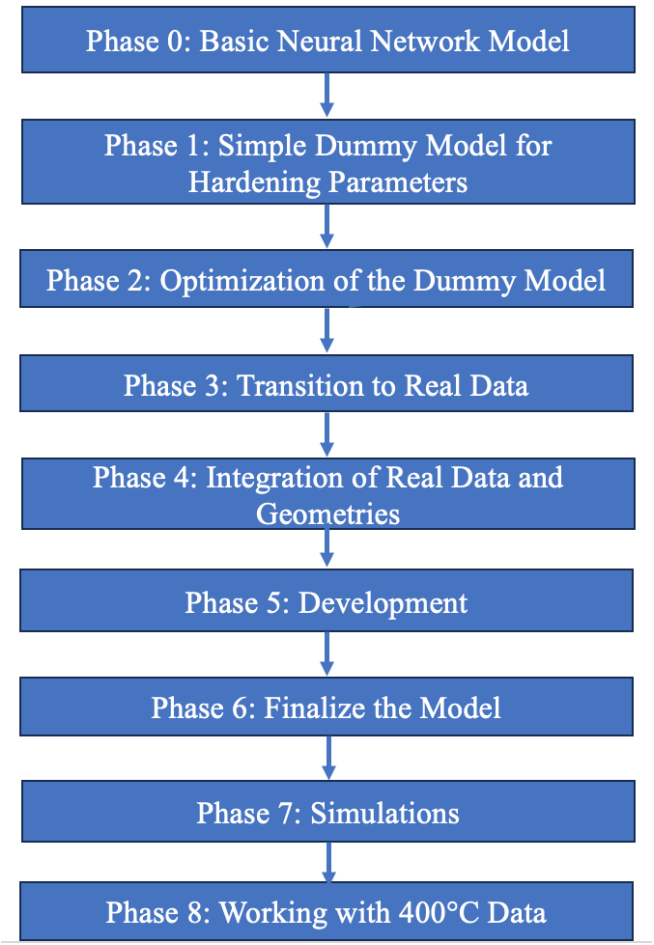

Figure 11. *The different phases of model development.*

Phase 0: Basic Neural Network Model

- In the initial phase, the primary objective was to grasp the fundamentals of neural network architecture.
- A simple neural network model was established, designed not for actual parameter prediction, but rather to acquire foundational knowledge about constructing neural networks.
- Random data was used, and the focus was on recording training data, predictions, and loss dynamics throughout 1000 epochs.

Phase 1: Simple Dummy Model for Hardening Parameters

- A simple dummy model was built to understand neural networks better and predict hardening parameters for different shapes.
- Dealing with complex data sets prompted the project team to stick with using random data.
- Training loss was checked, hardening parameters in different shapes compared, different learning rates' effects tested, loss changes over time monitored and prediction accuracy evaluated.

Phase 2: Optimization of the Dummy Model

- Using the insights gained from the initial dummy model, the architecture was fine-tuned and its performance enhanced.
- By refining the model, an improvement in training loss was observed, enhancing its predictive accuracy.

Phase 3: Transition to Real Data

- In the next phase, the transition from synthetic data to real, authentic data was made.
- A multi-output regression approach was used.
- To enhance visualization, Savitzky-Golay smoothing was applied to the data curves.
- The training loss derived from real data was tracked.

Phase 4: Integration of Real Data and Geometries

- The real-data model was built on and it was adapted to handle different geometries.
- Training loss was monitored across 1000 epochs and hardening parameters were predicted for each geometry.
- Learning rates were explored, and the convergence behavior of loss for different geometries was examined.

Phase 5: Development

- Experimentation with different network architectures: Different numbers of hidden layers and neurons were explored.
- Experimentation with different loss functions: The simplest MSE Loss function was discovered to work the best.
- Experimentation with all the hyperparameters, e.g., the learning rate was conducted.
- The activation function was changed from Relu to Leaky ReLu to address the "dying ReLu" problem.
- The input data structure was changed from different input files for each geometry to one input file that has the data for all geometries.
- The prediction structure was changed so that instead of predicting hardening parameters separately for each geometry, a single set of parameters were to be predicted that are applicable to all geometries.
- Principal component analysis (PCA) was implemented but the team realized that the model often gave better results without it so this was taken out.
- Two different data sets for training and testing were implemented. Early stopping was used in the training set to monitor validation loss. The project team began making the predictions on the test set (data that was previously untouched by the model).
- Scaling and inverse scaling were implemented.
- L2 regularization was implemented but later took it out because the model was better without it.
- R-squared score calculation was implemented to assess how well the model captures the data.
- Skorch wrapper was implemented for the simplification.

Phase 6: Finalize the Model

- Prediction interval for c4 and c6 was implemented based on the literature data to get better-looking stress-strain curves.

- Training and validation loss tracking were implemented to later use for visualization.
- The model was run 10 times. Among these runs, the predictions with the lowest validation error from the run with the highest R-squared score were recorded.

Phase 7: Simulations

- Stress and strain data were gained with Swift-Voce law based on the predicted hardening parameters.
- Based on this stress-strain data, .odb files were generated containing force-displacement data by using ABAQUS simulations in CSC.
- The .odb files were postprocessed and .rpt files were retrieved containing force-displacement data of each geometry.
- The .rpt files were converted into .csv files.
- The simulated force-displacement data was plotted and these curves were compared with the experimental data.

Phase 8: Working with 400°C Data

- All the described phases were repeated for 400°C data.

3.4 Explanation of Phases

Phase 0: Basic Neural Network Model

The work was initiated by generating a data set for the neural network to be developed, using an 'axis' variable for input and a specific mathematical function to create the corresponding output. The neural network had an input layer, two hidden layers, and an output layer designed for regression. The model was compiled with the 'ADAM' optimizer and used the 'mean squared error' as the loss function. After initiating training for 100 epochs, where the model adjusted its parameters to minimize errors, the model was tested by making predictions using the trained model.

Phase 1: Simple Dummy Model for Hardening Parameters

In Phase 1 of the project, the same methods that had been utilized before were stuck to - the structure of the neural network, how the data was generated, and how training and predictions were made. However, this time, the model included 4 sets of random f/d data (for 4 geometries) as the input as well as one set of random hardening parameters data as the output. Through back-propagation, the neural network was adjusted based on gradients, tracking loss values during training. After training, the neural network was used to predict hardening parameters for each input set, obtaining predictions with random data. Finally, "c1" to "c7" parameters were visualized across all shapes using a bar chart.

Phase 2: Optimization of the Dummy Model

The neural network was fine-tuned by training it repeatedly with a supervised learning approach. The goal was to minimize the Mean Squared Error loss by adjusting the model's parameters. Once trained, the model made predictions, and its performance was assessed by comparing and visualizing the results.

Phase 3: Transition to Real Data

Unlike the earlier dummy model that used synthetic data, this model was designed to use real input data, which was provided by the project advisor.

Phase 4: Integration of Real Data and Geometries

While Phase 3 operated on real data, it had a limitation in that it did not take into account the variations across different geometries. In contrast, the Phase 4 model has been designed to address this gap. It considers the data from all four geometries. Moreover, varying learning rates were experimented with for each geometry, and how these different learning rates impacted the training loss for these geometries was visualized.

Phase 5: Development

This stage, spanning around 2 months, was the longest phase of the project work. Refining the model was the main focus for it to become better at predicting. The next section details the specific improvements that were made during this time.

Neural Network Architecture Different ways of setting up the network were attempted, with testing various numbers of hidden layers and neurons meaning that hyperparameter tuning was employed. The best-performing setup turned out to be 3 hidden layers with 256, 256, and 128 neurons. Importantly, this configuration is also fast on the CPU, making it computationally efficient.

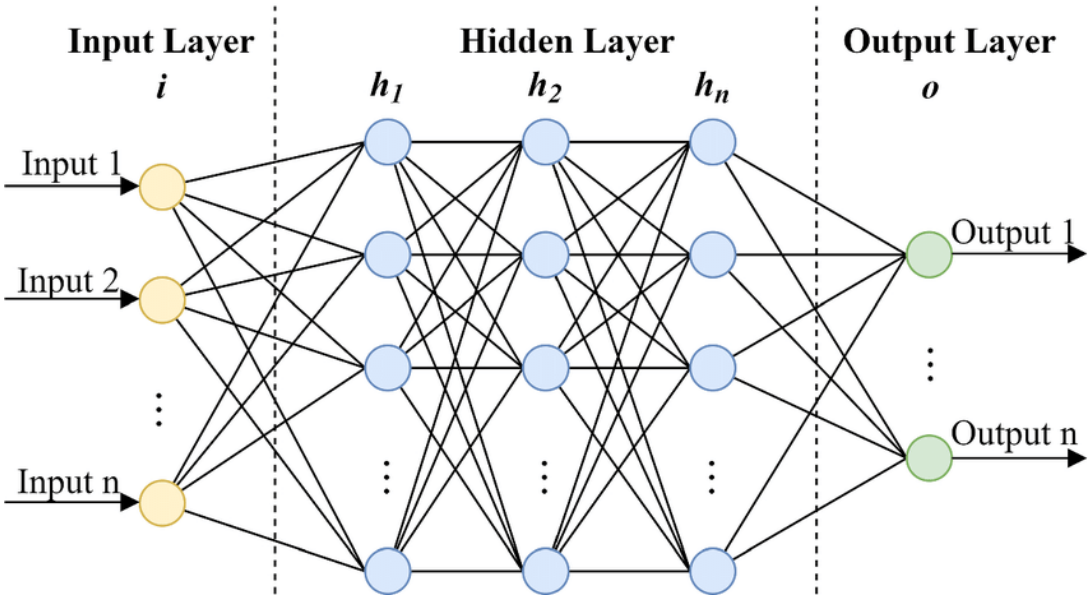

Figure 12. *A neural network architecture that is similar to ours [26]*

Loss Function Different loss functions were experimented with and it was discovered that the simplest Mean Squared Error (MSE) loss function worked best for the model. MSE measures the average squared difference between predicted values and actual values. In the context of this specific project, using the MSELoss function from the neural network library of Torch, it was established that this approach effectively minimized errors during training, contributing to the model's overall success. The MSE function can be seen below:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Learning Rate A learning rate (lr) of 0.0005 was found to work best for the developed model. The learning rate represents the size of steps the model takes during training to minimize errors. A smaller learning rate, like 0.0005, often helps the model converge more effectively, especially in complex data sets.

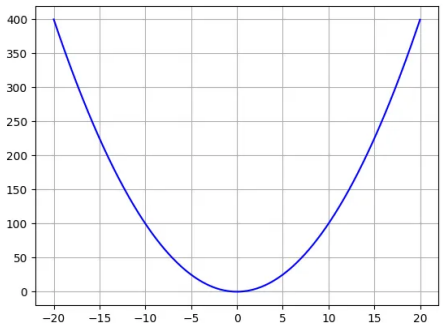

Figure 13. *MSE Loss function graphed [27]*

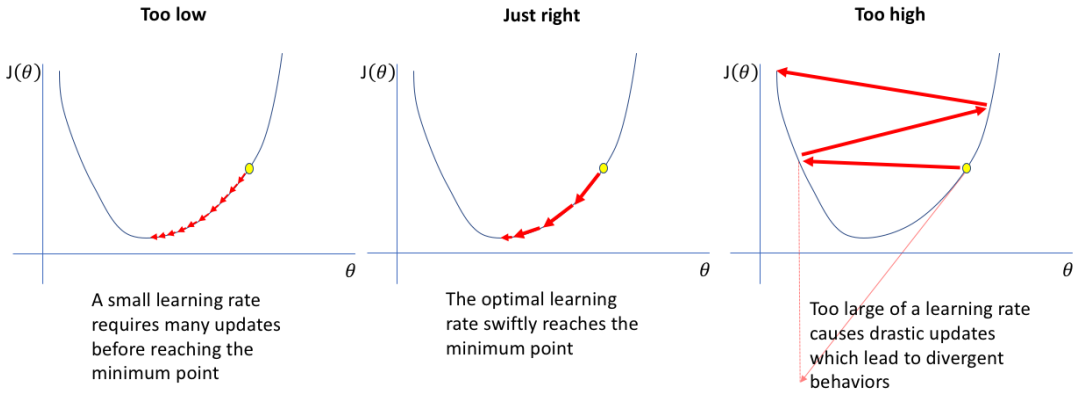

Figure 14. *Impact of learning rate illustrated [28]*

Activation Function The team switched from using ReLU (Rectified Linear Unit) to Leaky ReLU to fix the "dying ReLU" problem, where some neurons become inactive during training. Leaky ReLU avoids this by allowing small, non-zero values for negative inputs. Although Sigmoid and Swish activation functions were also attempted, these did not work well for the model, so Leaky ReLU was the one finally used for better performance.

Input Data Input data structure was changed from different input files for each geometry to one input file that has the data for all geometries. In other words, instead of using four files with two columns each, one file with eight columns was used.

Prediction Structure The way the model predicts was modified. Instead of predicting hardening parameters individually for each geometry, the changed was done for predicting a single set of parameters that applies to all geometries.

PCA Principal component analysis (PCA) was incorporated, a technique used to simplify the complexity in high-dimensional data while retaining trends and patterns. Initially, the team found that two principal components were sufficient

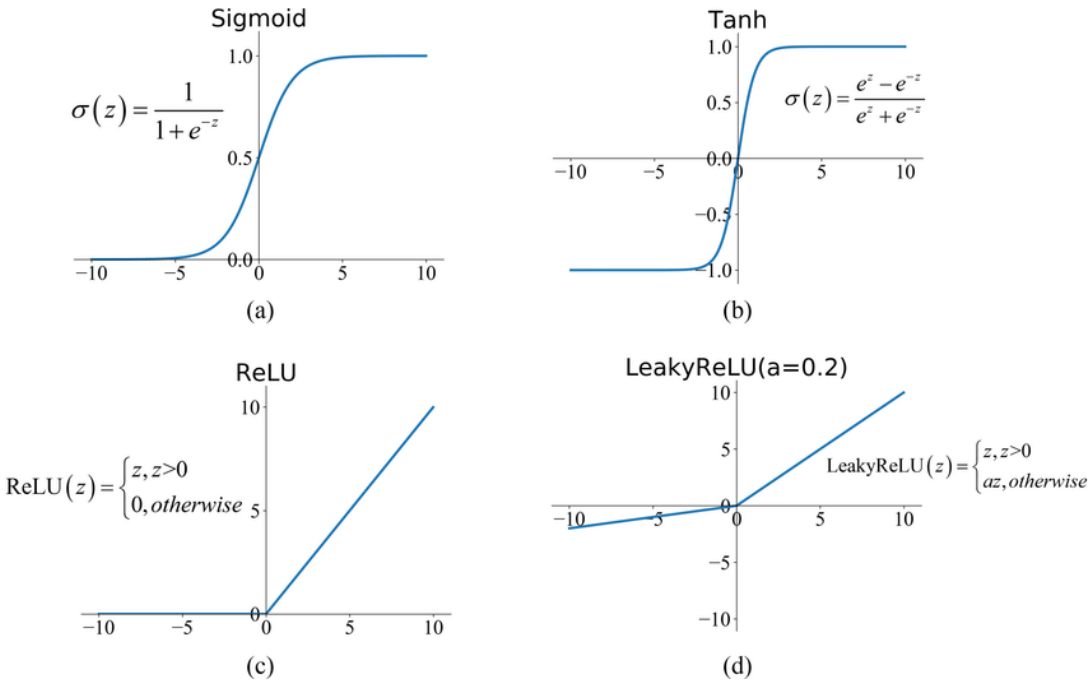

Figure 15. *Commonly used activation functions [29]*

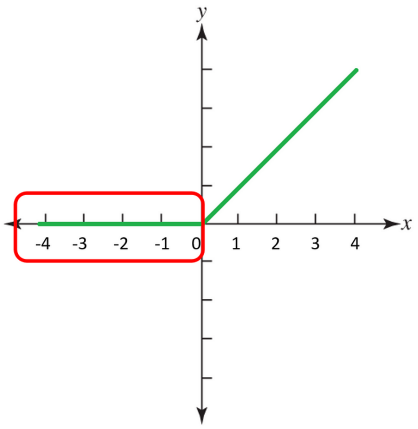

Figure 16. *Dying ReLu problem illustrated [30]*

for the input data. However, upon observing that the model often achieved better results without PCA, the decision to exclude it from the process was taken.

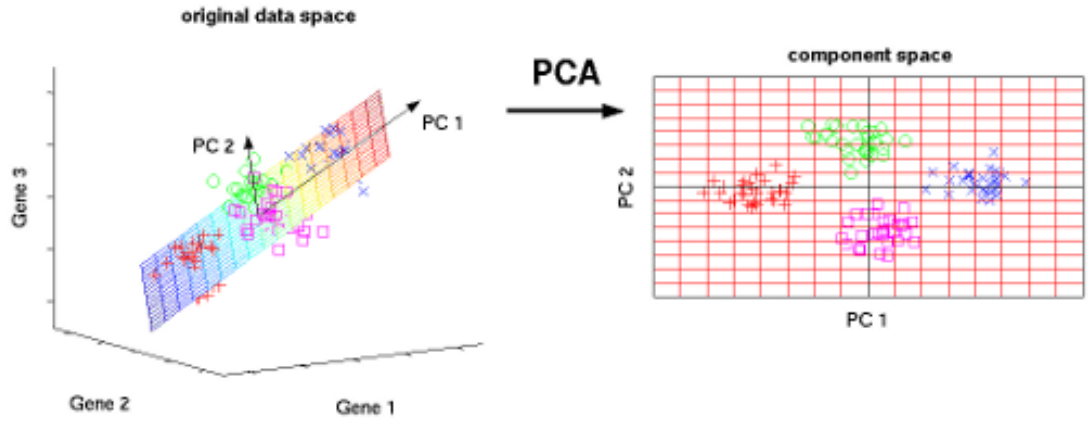

Figure 17. *Impact of PCA [31]*

Different Data Sets Two data sets were used: a training set and a testing set. The training set was to teach the model. Even though there was no separate validation set, the training set implicitly handled the validation by using an early stopping mechanism. If the validation loss did not improve for 20 iterations, the loop stopped. After training, predictions were made on the test set to see how well the model works in the real world.

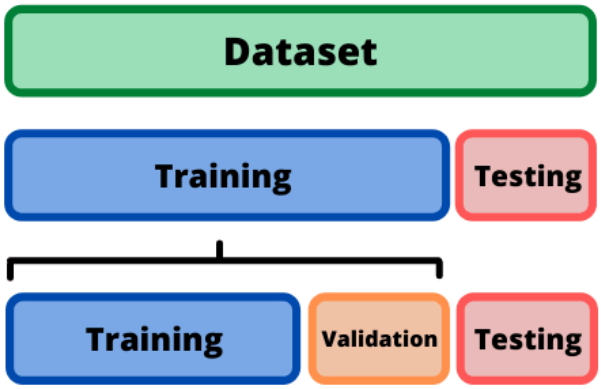

Figure 18. *Framework used for datasets [32]*

Scaling MinMaxScaler from sklearn was used to scale and unscale the data. Scaling ensures that all features have similar numerical ranges, preventing any one feature from overshadowing others during model training. This helps the developed model perform optimally by treating each feature more equally.

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

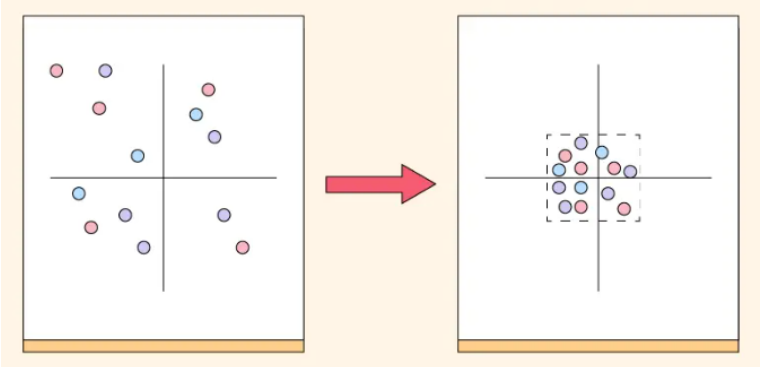

Figure 19. *Impact of feature scaling [33]*

R-squared Scoring R-squared score calculation was implemented to evaluate the performance of the model in capturing the data. The R-squared score pro-

vides insights into how well the developed neural network is able to explain the variability in the data, serving as a valuable metric for model assessment and validation.

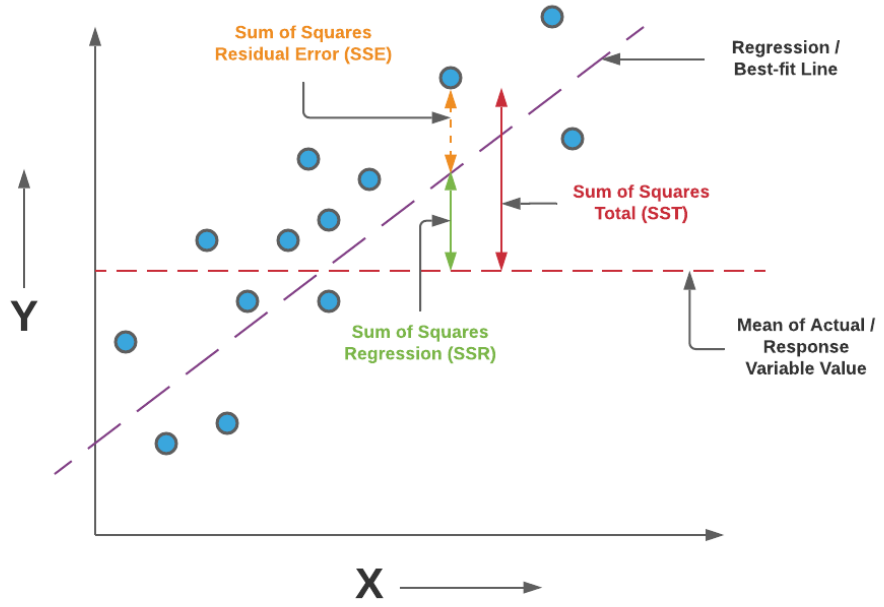

Figure 20. *Illustration of metrics used to calculate R-squared score [34]*

Regularization Initially, experimentation with L2 regularization, a technique adding a penalty term to the loss function based on squared model weights, was done. However, the team found the model performed better without it. Instead, it was decided to rely on Leaky ReLU to naturally prevent overfitting with its built-in negative slope. This, along with early stopping, served as effective regularization without the need for extra L2 regularization.

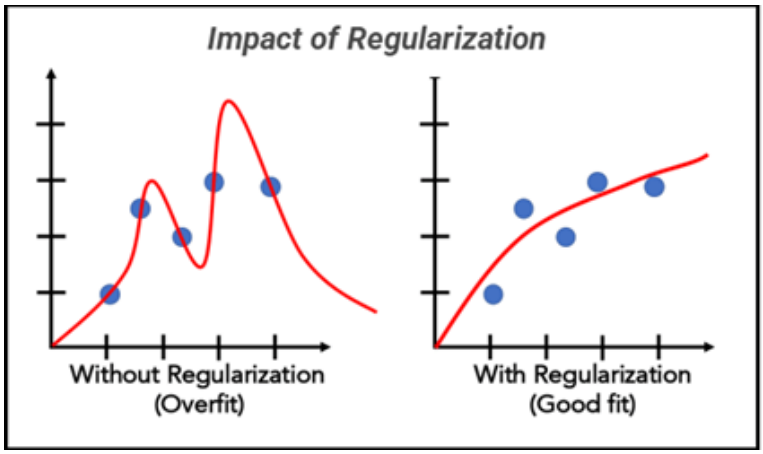

Figure 21. *Impact of regularization [35]*

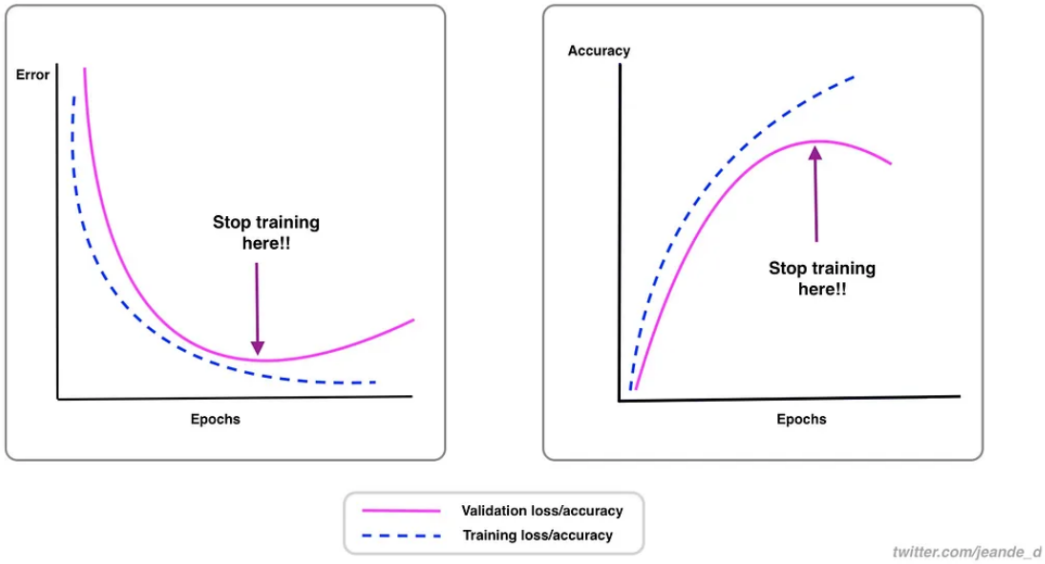

Figure 22. *Early stopping mechanism illustrated [36]*

Skorch wrapper By implementing the Skorch wrapper, the integration of the neural network was streamlined into scikit-learn workflows. This simplifies the overall process, making it more convenient to incorporate neural networks seamlessly into the machine-learning tasks.

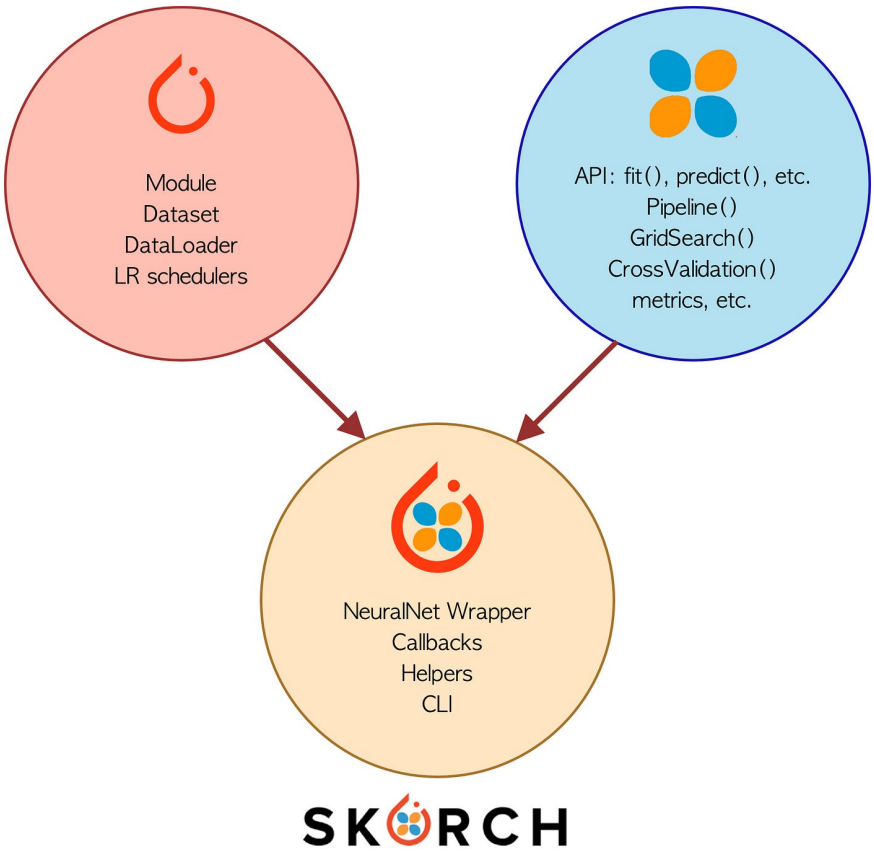

Figure 23. *Schematic diagram of Skorch Wrapper [37]*

Phase 6: Finalize the Model

Prediction Interval Acknowledging the significant impact of c4 (hardening exponent) on the stress-strain curve, a range derived from literature data was intentionally included. Small changes in c4, even by 0.05, can result in significant shifts in the curve. Likewise, recognizing the crucial role of c6 in determining the yield point, similar to c2, a specific range was set based on data found in the literature. The specified ranges are as follows:

```
fourth_variable_range = [0.04, 0.17]
sixth_variable_range = [700, 1500]
```

Mulkey's report [38] indicated that the lowest hardening exponent for martensitic steel can be as low as 0.04. Likewise, papers by Huang et al. [39] and Farabi et al. [40] indicated that for high-strength steels, the highest value for hardening exponent is often around 0.17. Currently, there is very little literature on the range of c6 values that high-strength steel can get. However, by combining the research from Palaperti et al.[41] and He et al. [42], the project team determined that an appropriate range for the c6 parameter is 700 to 1500 MPa.

Loss Tracking A tracking system for training and validation loss during the model's training process was added, intending to use this information for later visualization.

Predictions The model was run 10 times, and among these runs, the predictions from the run with the lowest validation error and the highest R-squared score were selected.

Phase 7: Simulations

Stress and strain data were obtained with Swift-Voce law based on the predicted hardening parameters. This stress-strain data was used within the 'Material_DP1000_Mises.inp' file, which was provided by the project advisor. The team also had .inp files of the geometries (chd6.inp, ndb6.inp, ndb20.inp, ndb50.inp) containing information related to the material properties and geometry. The .inp files of geometries imported the 'Material_DP1000_Mises.inp' file, which included the stress-strain data. For NDB50, the 'ZinanCSCBatchExample.sh' file was used, which imported the ndb50.inp file in order to submit a job by using 'sbatch' command. As the output of this job, the ndb50.odb file was obtained. The same process

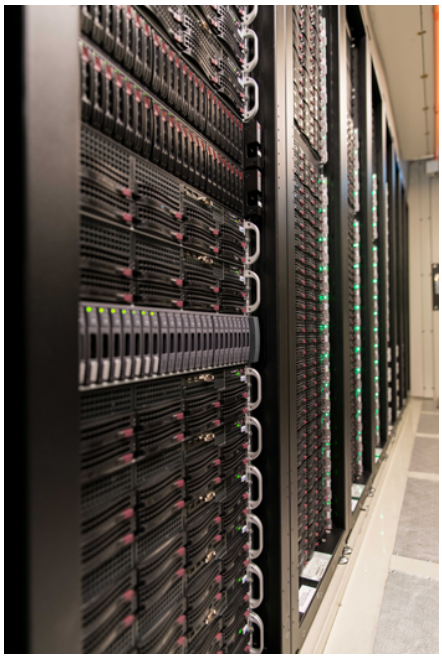

Figure 24. *Puhti supercomputer that was used for the simulations [43]*

was repeated for the other three geometries as well. The completion time of jobs varied from geometry to geometry. Most jobs had a duration of more than an hour.

The .odb files of all geometries were sent to the project advisor. She provided the project team the corresponding post-processing script. By using this post-processing script, force and displacement data were extracted from the .odb files in the form of .rpt. To use the post-processing script, Abaqus license access from CSC support was requested by the project team. The necessary license was issued by CSC support and they were very helpful throughout the whole process. Having acquired the Abaqus license, the Abaqus module was imported and the post-processing script was run. Then the .rpt files were converted into .csv files. Finally, the simulated force-displacement data was visualized, making it possible to make a comparison with the corresponding experimental curves.

ABAQUS simulations were the longest for geometry CHD20. It took more than 2 hours and a .odb file was gained that was much bigger than the .odb file of the other geometries (2+ GB). Probably due to the large size of this file, 300 force-displacement data points were obtained after post-processing for NDB20, while 100 data points were retrieved for other geometries. Post-processing took about 5 minutes for all geometries. Post-processing was conducted on the compute node shell of CSC, by utilizing four cores. Since the team used ABAQUS module during post-processing, the ABAQUS license from CSC support was requested as mentioned previously. After getting the license, it was loaded within the shell and the post-processing was conducted. Simulations to get the .odb file were conducted

on the terminal within the group folder.

Here are the commands that were used during the simulations and post-processing:

```
# to submit the job
$ sbatch TheBatchFile.sh

# the following commands should be run on compute node shell to avoid
    errors
# to go to the correct directory containing .odb files
$ cd /scratch/projectDirectory/GroupFolder/

# to load abaqus 2022
$ module load abaqus/2022

# to run the post-processing script
$ abq2022 cae noGUI=ppScript.py
```

Phase 8: Working with 400°C Data

All the different phases as described previously were repeated for 400°C data. The same scripts for extracting, mining, and formatting data were used. The same machine learning models (but only with changed input data) were utilized. The same simulation process that was done for 25°C was repeated, with the only difference being that parameter predictions were different since different input data had been used for the models.

3.5 Alternative Models

In addition to the developed neural network model, the project team decided to look into the possibility of utilizing alternative machine learning algorithms for the purpose of predicting the hardening parameters. Apart from Bastos et al. [24] not much research on alternative methods apart from neural networks has been previously conducted. Therefore, it is of interest to look into the prediction capabilities of different kinds of models and that is why the team developed a model based on the Random Forest algorithm and a model based on the XGBoost algorithm. What these actually entail and how these were developed is detailed in the following subsections.

Random Forest

Random Forest is a method that works like a team of decision-makers. It uses many small decision trees to make predictions, and then it combines all their ideas to give a more accurate result. A Random Forest model was implemented by the project team to compare the findings with the developed neural network model. The same data files as the neural network model utilized were used and the data set was split into training and testing sets. To emphasize parameter importance, a weighted MSE Loss function was used. Employing a Random Forest Regression model with 100 trees, it was trained on the training set. The random state "42" was used to ensure reproducibility. It is important to note that "RandomForestRegressor" from Scikit-Learn was utilized. Subsequently, predictions were made on the test set. Evaluating the model, a custom loss, mean squared error, and R-squared were computed. Lastly, visual scatter plots comparing actual and predicted values for each parameter, accompanied by linear regression lines were created. The goal was to develop a robust model for predicting parameters (c1 to c7) based on the provided features.

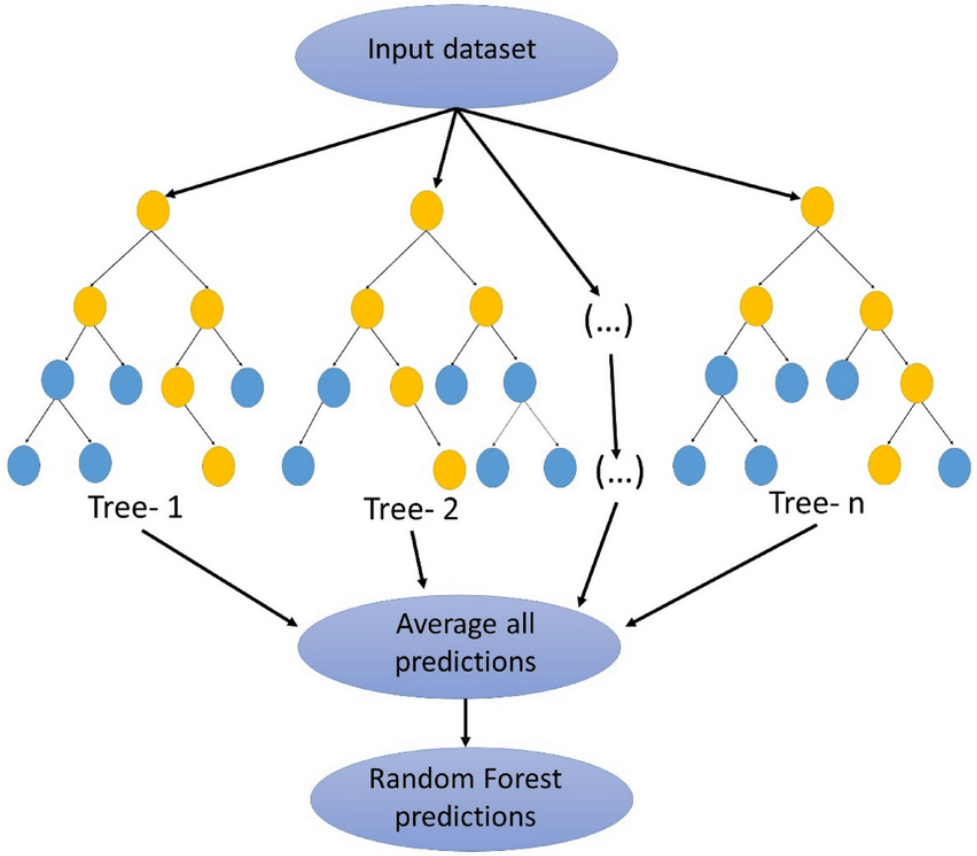

Figure 25. *Random Forest diagram [44]*

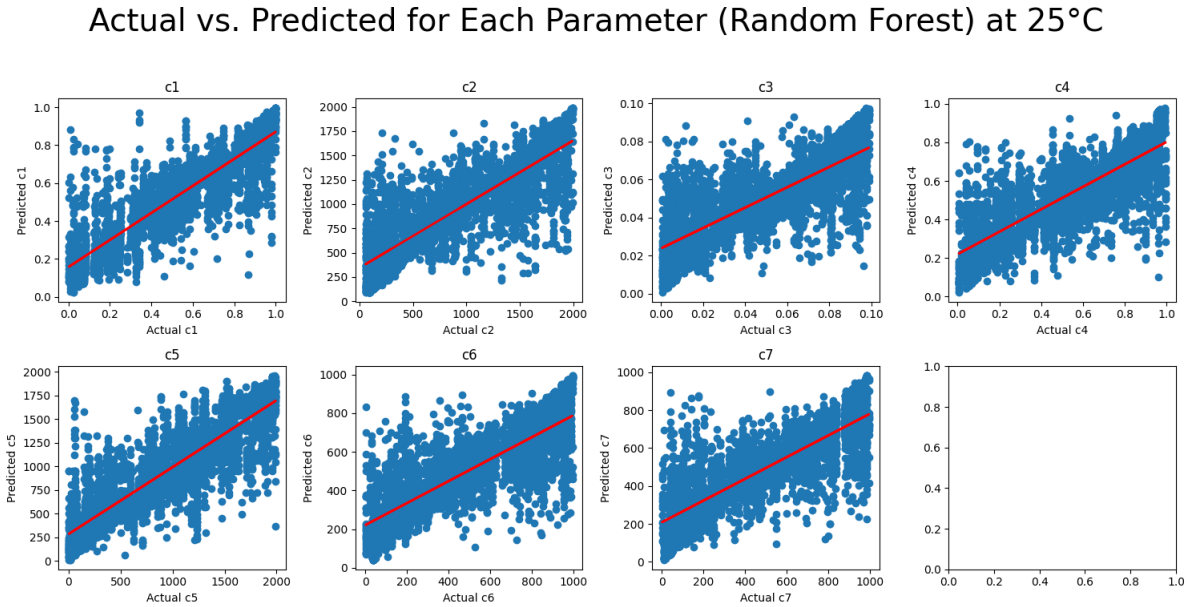

Figure 26. *Best fit lines for every parameter constructed with random forest model. (25C)*

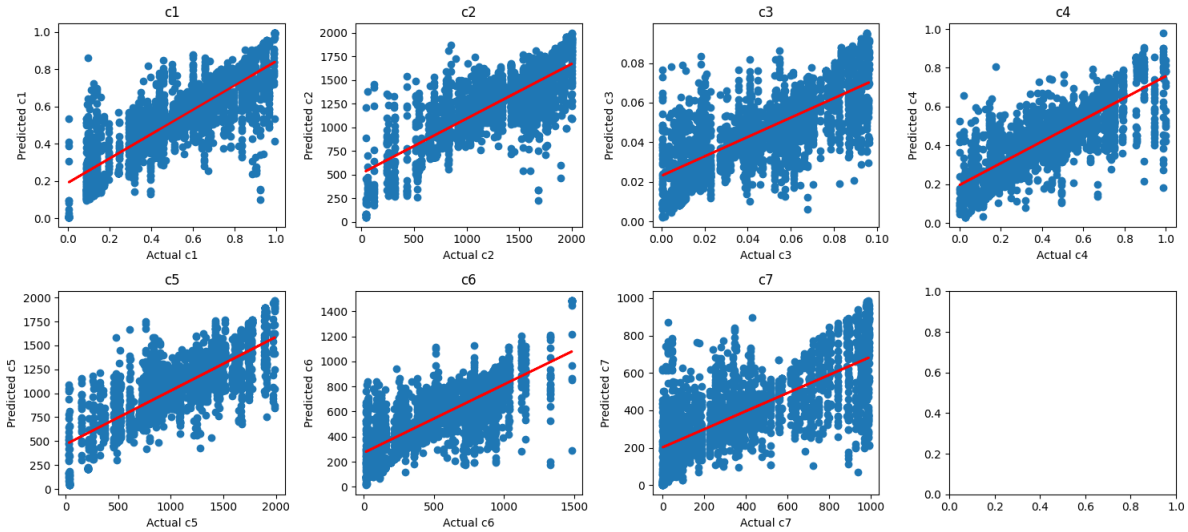

Figure 27. *Best fit lines for every parameter constructed with random forest model. (400C)*

XGBoost

XGBoost is a technique that is similar to Random Forest. In Random Forest, each prediction tree "decides" separately, and their decisions are averaged. With XGBoost, the trees work together, learning from each other's mistakes to improve predictions step by step. XGBoost tends to work better in complex problems since it has a mechanism that "learns from the mistakes" and adapts accordingly. To implement XGBoost, the same steps were followed as for the Random Forest. There were only a few changes that were made. Instead of using 100 trees, 500 trees were used as it significantly improved the R2 score. DMatrix from XGB was also used to convert Pandas data frames to the format that XGBoost can understand. The seed was unchanged. With these changes, the XGBoost algorithm was used to obtain alternative predictions.

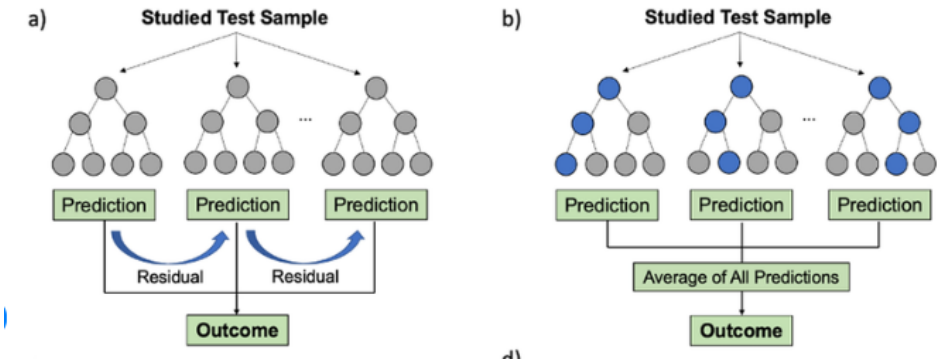

Figure 28. *XGBoost (a) vs. Random Forest (b) [45]*

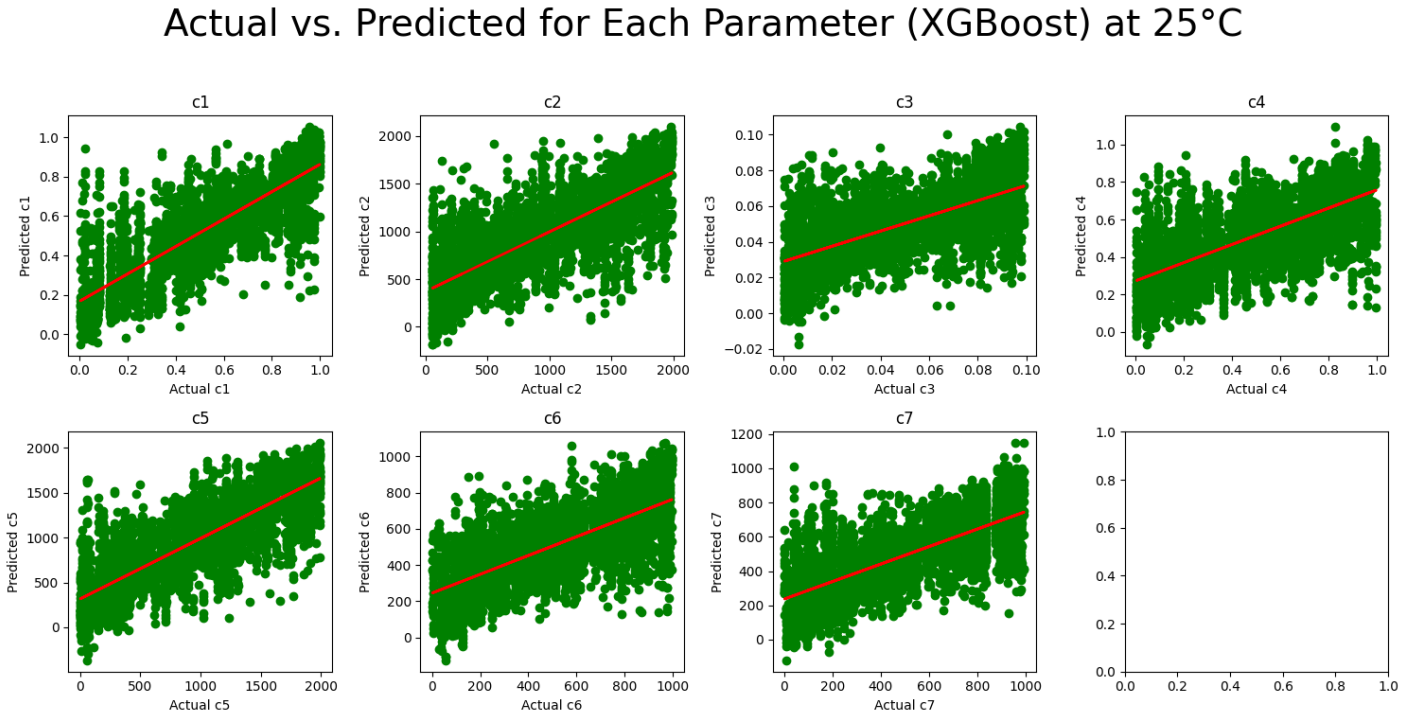

Figure 29. *Best fit lines for every parameter constructed with XGboost model (25C).*

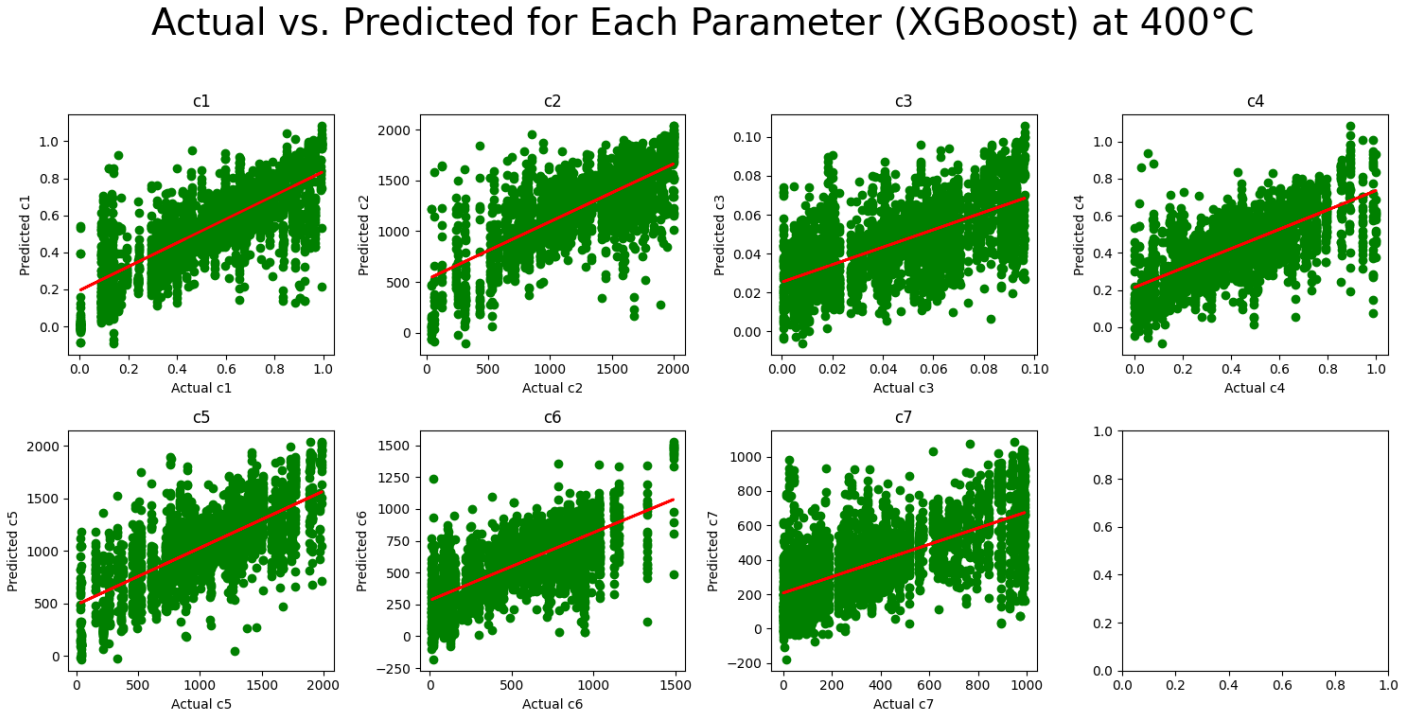

Figure 30. *Best fit lines for every parameter constructed with XGboost model (400C).*

4 Results

This section displays the results obtained from the developed machine learning models. As explained in Subsection 3.4, the approach for development involved selecting predictions with the lowest validation error from the run with the highest R-squared value, ensuring a comprehensive data representation. Based on these considerations, the results for the hardening parameters and the corresponding R-squared and loss for the predicting model are as follows:

Table 1. *Performance Metrics and Predicted Parameters for ML Models at 25°C*

Name	R-squared	Loss	c1	c2	c3	c4	c5	c6	c7
NN	0.55	0.05	0.67	1276.13	0.04	0.17	1467.29	700.00	347.39
RF	0.67	36507.26	0.74	1265.03	0.05	0.17	1048.29	700.00	367.25
XGB	0.55	47906.77	0.79	970.58	0.05	0.14	1274.28	700.00	297.66

Table 2. *Performance Metrics and Predicted Parameters for ML Models at 400°C*

Name	R-squared	Loss	c1	c2	c3	c4	c5	c6	c7
NN	0.46	0.04	0.50	1406.27	0.04	0.17	856.24	802.43	456.02
RF	0.58	33685.11	0.46	1532.02	0.03	0.17	849.99	802.43	452.70
XGB	0.48	40554.44	0.51	1561.77	0.04	0.17	744.17	898.77	521.70

Note that in the tables above:

- NN corresponds to the Neural Network Model.
- RF corresponds to the Random Forest Model.
- XGB corresponds to the XGBoost Model.
- Loss for NN models is validation loss.
- Loss for RF and XGB models are MSE loss.
- Values have been rounded to two decimals for better page fit.

The results obtained from the models are further illustrated in different ways in the following subsections. In Subsection 4.1, the hardening parameters for all the developed models are visualized. In Subsection 4.2 the R-squared value of all the models are shown in a way where these can be compared. The individual parameters can be compared by model with each other in the figures found in Subsection 4.3. The training and validation loss of the developed neural network model is visualized in the figure that can be found in Subsection 4.4. Subsection 4.5 shows the flow curves that are achieved from inputting the predicted parameters in the Swift-Voce law. The closer the curve is to the experimental curve the better (the values for this were only available for 25°C). Finally, in Subsection 4.6, the force-displacement curves that could be obtained from the predicted parameters are presented for each model along with the experimental curves that the project team was provided the data with from the beginning. The data for plotting these curves were obtained through the ABAQUS simulations run in CSC detailed in the previous methodology section. What these results actually mean and what kind of conclusions can be drawn from these is expanded and discussed in the following Sections 5 and 6.

4.1 Hardening Parameters for Each Model

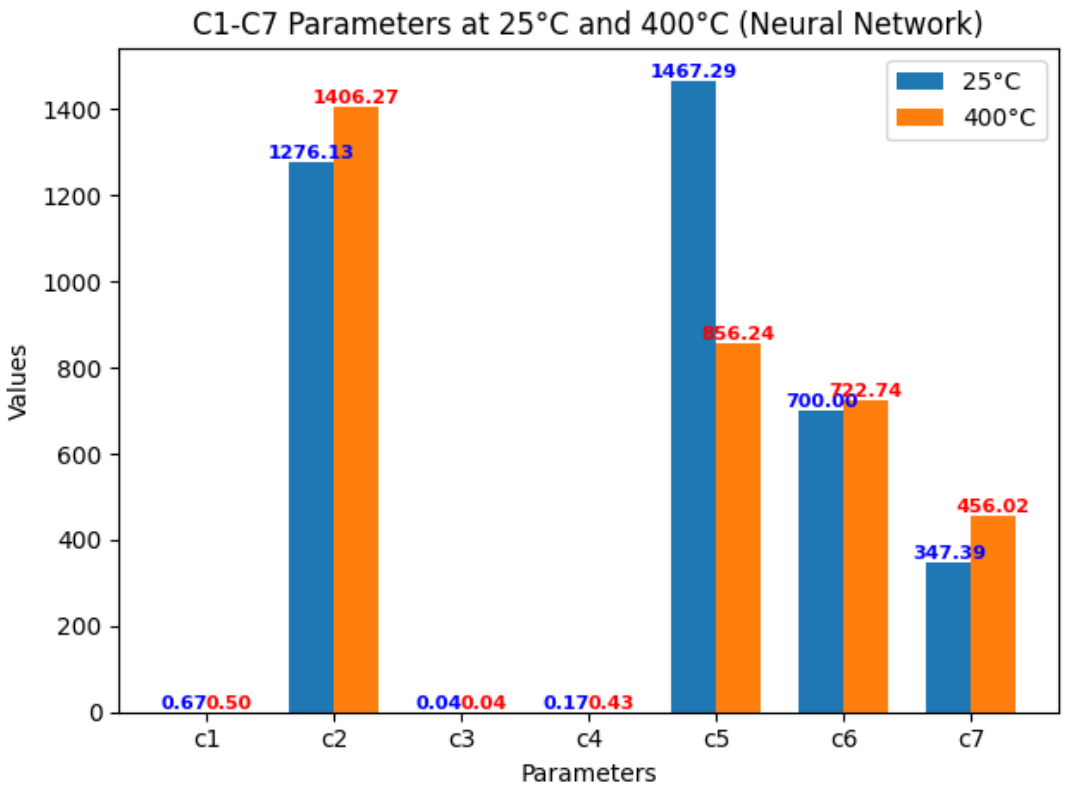

Figure 31. *Predictions by neural network based on 25°C data vs. 400°C data*

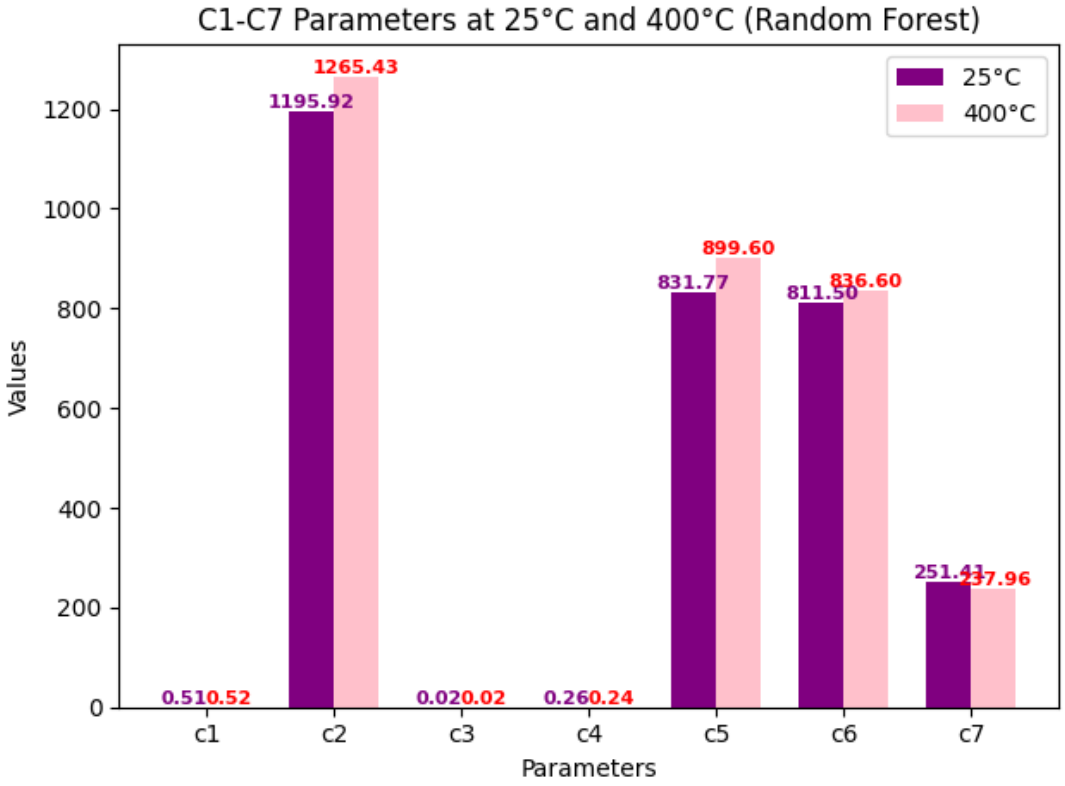

Figure 32. *Predictions by random forest based on 25°C data vs. 400°C data*

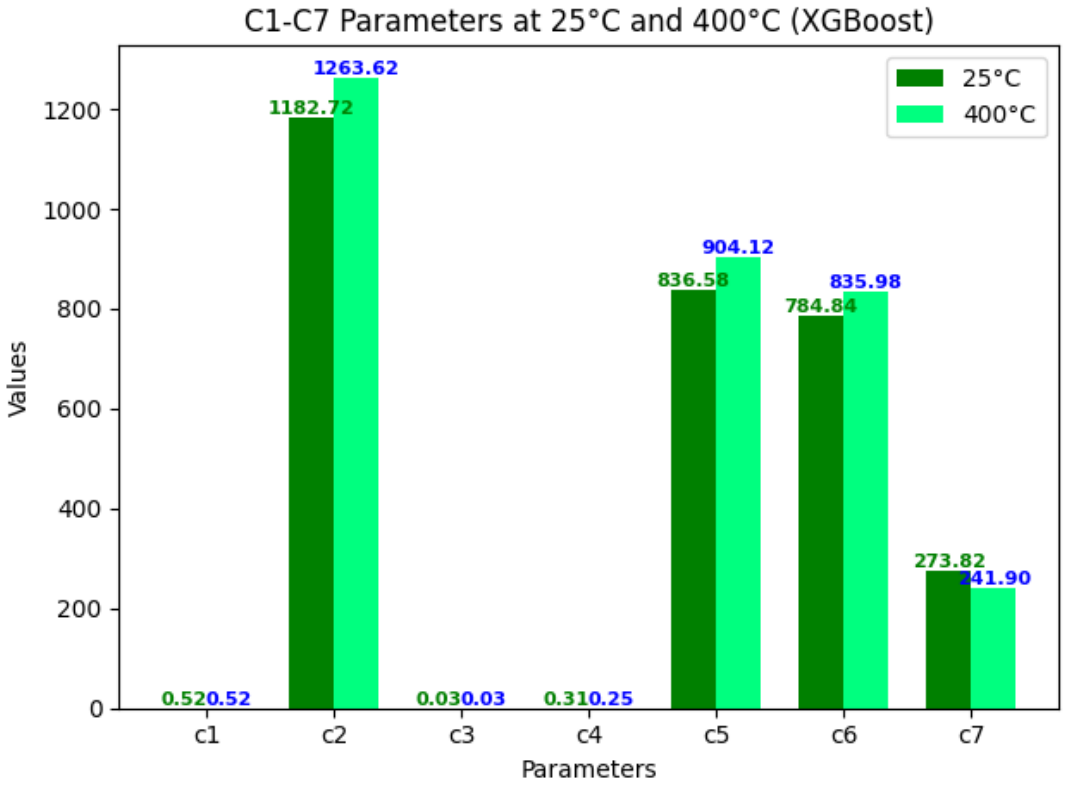

Figure 33. *Predictions by XGboost on 25°C data vs. 400°C data*

4.2 R-squared of Models

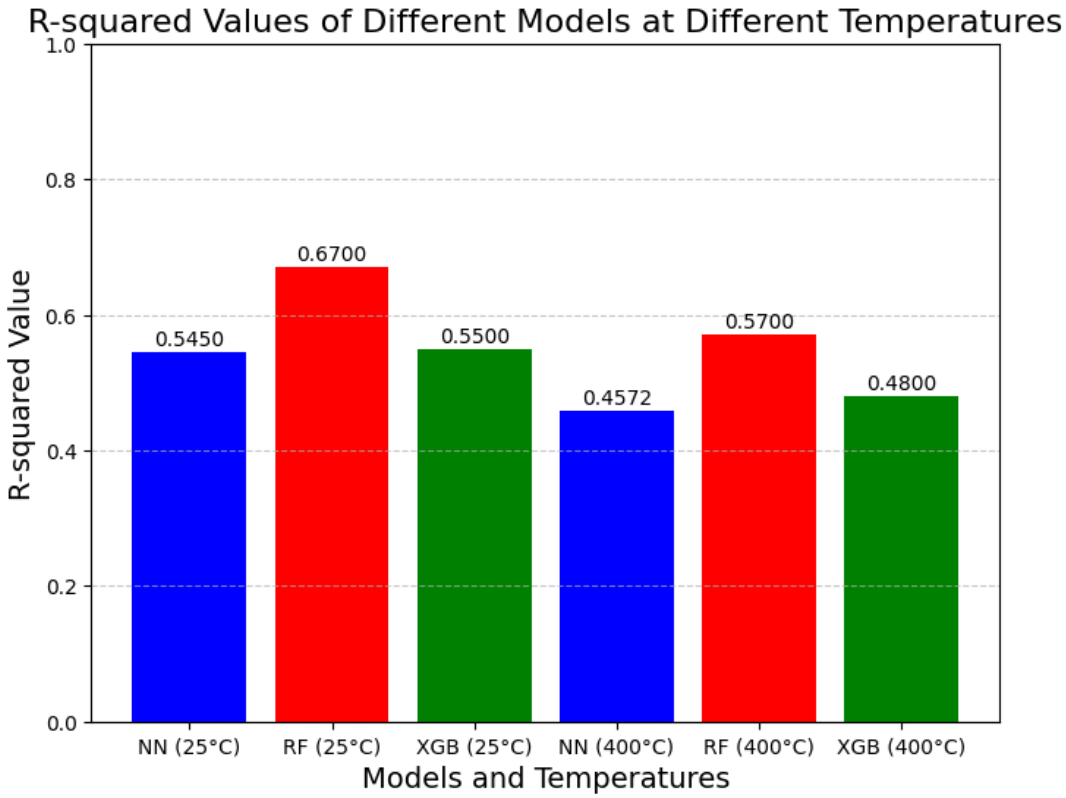

Figure 34. *Comparison of R-squared values across different models at different temperatures*

4.3 Comparison of Parameters Across Models

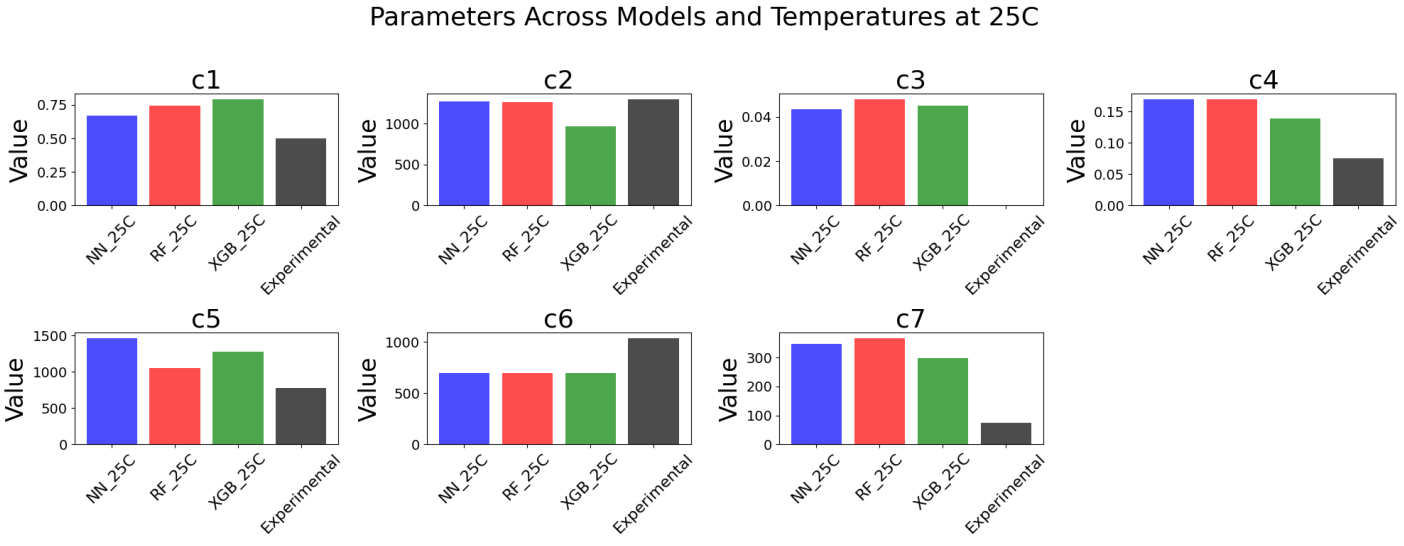

Figure 35. *Parameters across models and temperatures (25°C)*

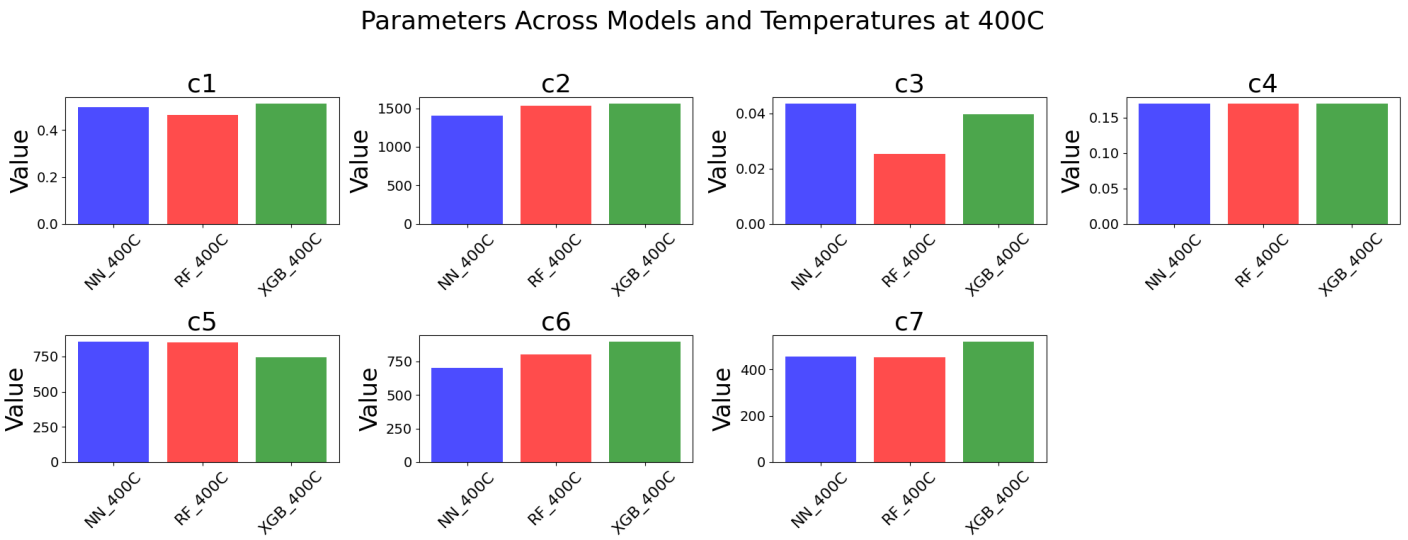

Figure 36. *Parameters across models and temperatures (400°C)*

4.4 Training and Validation Loss of Neural Network Model

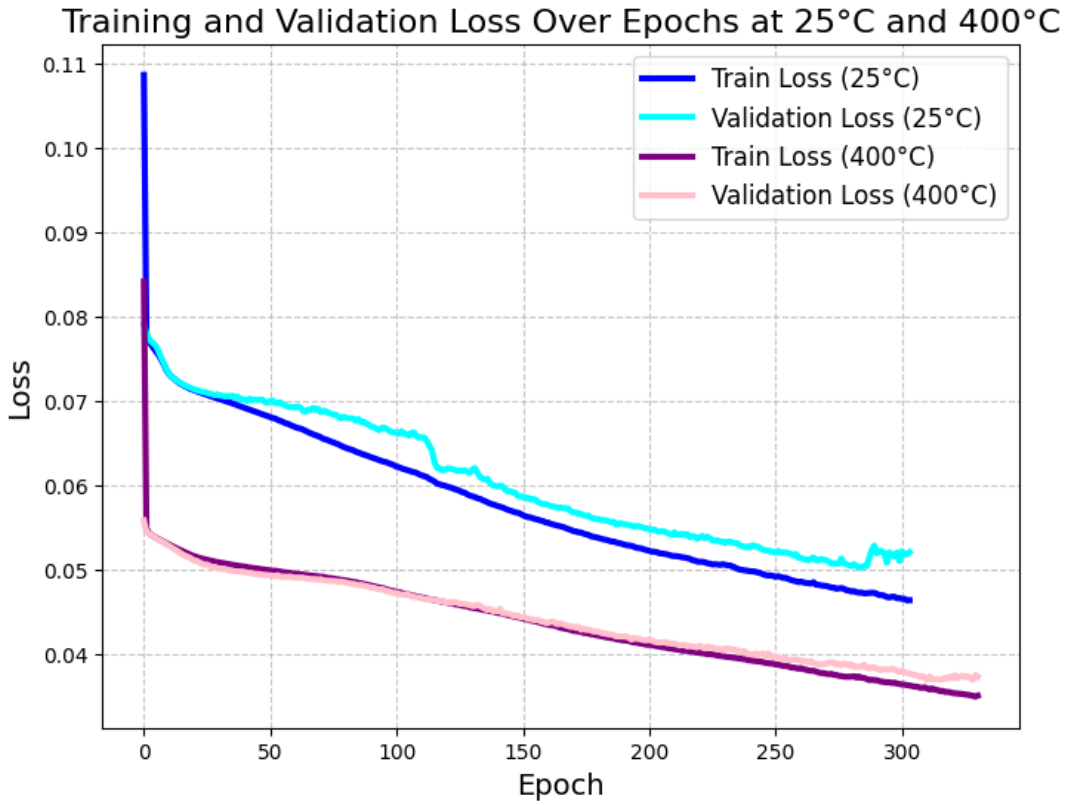

Figure 37. *Loss in the neural network model with two different sets of input data.*

4.5 Flow Curves

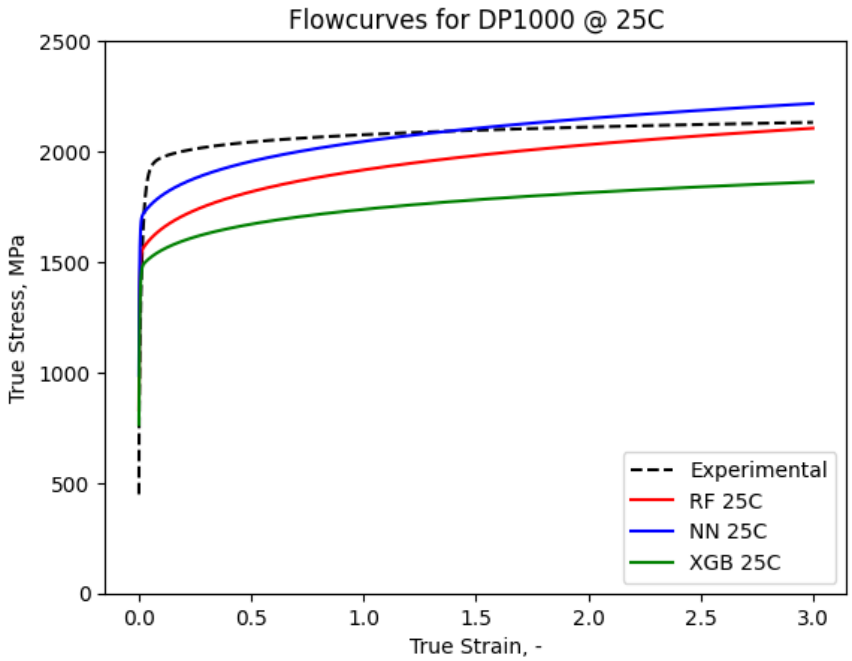

Figure 38. *Flowcurves constructed with the Swift-Voce law by using the predicted hardening parameters (25°C).*

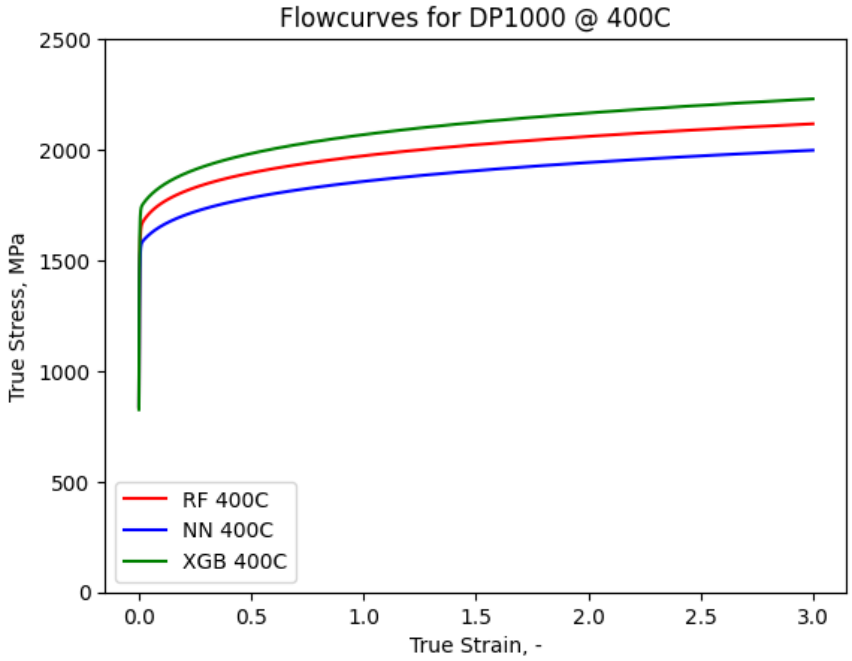

Figure 39. *Flowcurves constructed with the Swift-Voce law by using the predicted hardening parameters (400°C).*

4.6 Force-Displacement Curves Obtained From Predicted Parameters

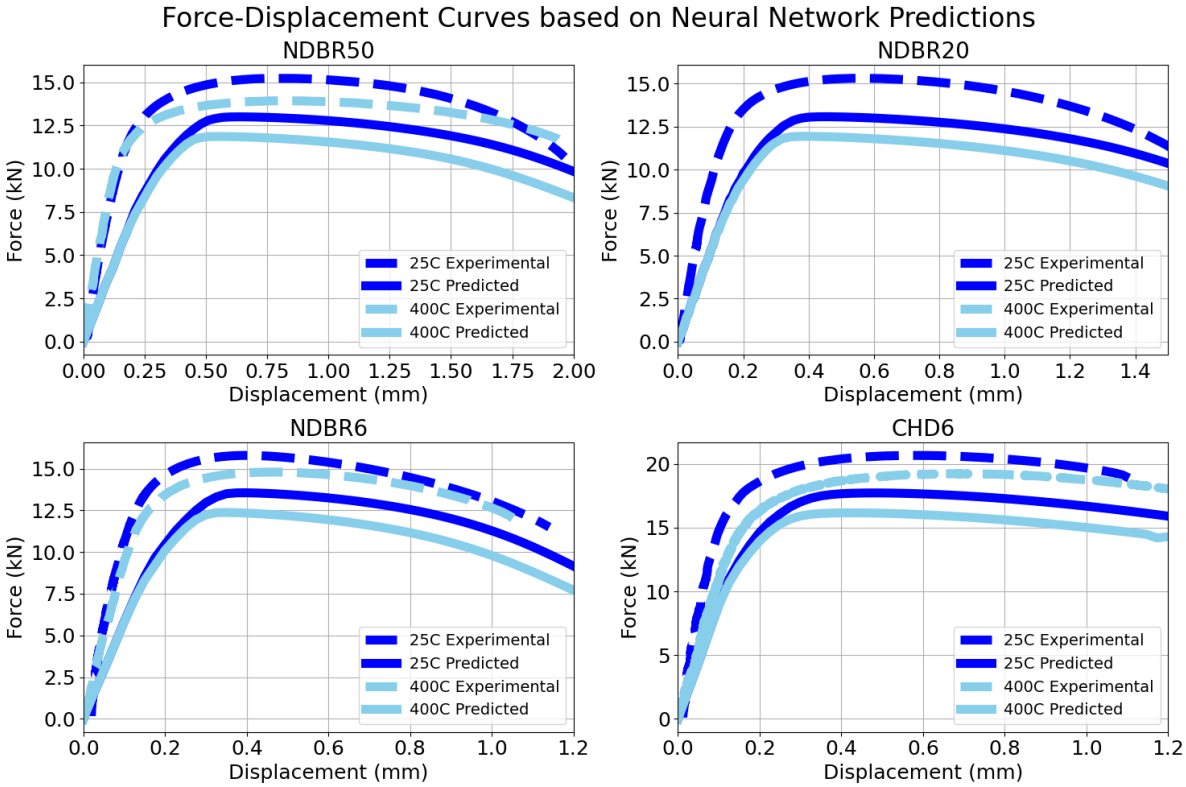

Figure 40. *Force-displacement curves based on the predictions of the neural network model.*

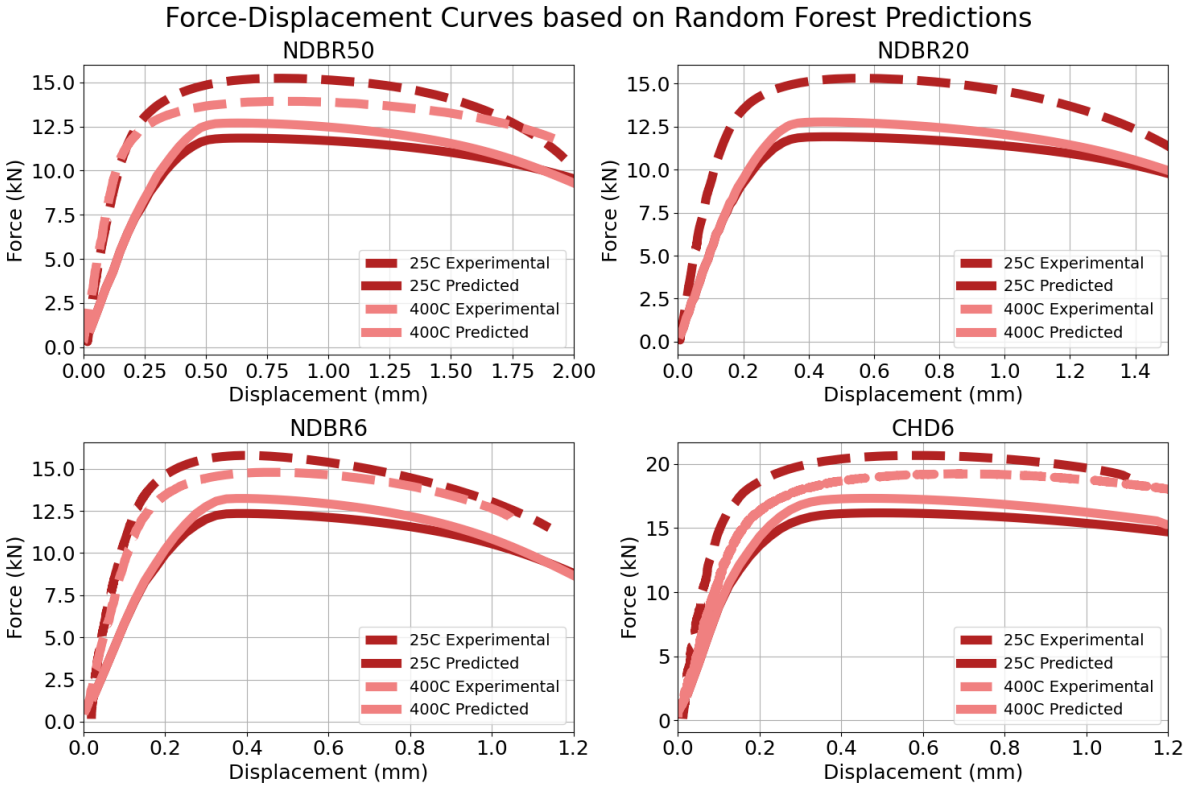

Figure 41. *Force-displacement curves based on the predictions of the random forest model.*

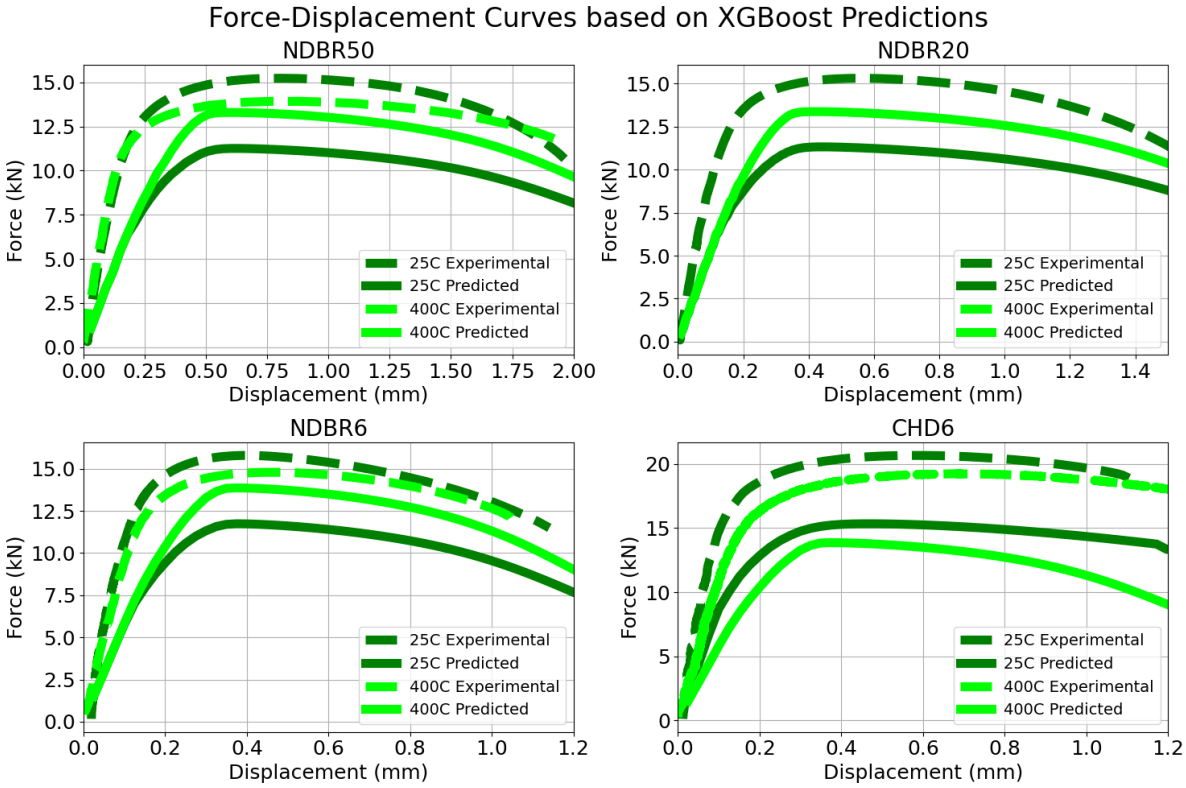

Figure 42. *Force-displacement curves based on the predictions of the XGboost model.*

5 Discussion

This section discusses the results obtained from the different models developed. Subsection 5.1 elaborates on the created ANN model and what the results of this are. Following this, Subsection 5.2 goes over what the results of the additional alternative models were. Subsection 5.3 discusses the validity of all the created models. Finally, Subsection 5.4 gives a brief overview of the effect of temperature on the models with Subsection 5.5 closing the section with a few remarks on the limitations for the use of the developed models.

5.1 ANN model

The original aim of the project was to develop an ANN model that would be able to predict universally any kind of material's hardening parameters for Swift-Voce constitutive equation based on force-displacement input data of the specific material. The process required a lot of experimentation with hyperparameters to find the most optimal model.

It was observed that even very small deviations in certain parameters (e.g. hardening exponent) can result in very different stress-strain curves. This is demonstrated and visualized for the c4-parameter in Figure 42. The high sensitivity of stress-strain curves creates the need to predict extremely accurate (95%+) parameters. Yet, only 100 sets of hardening parameters were provided to the project team, which is definitely not enough to produce a neural network that will yield extremely accurate results. Despite this, the developed model captured over half of the relationships in the data, as evidenced by the R-squared value of 0.55.

The study showed the project team that although using neural networks is a computationally efficient way of predicting hardening parameters, this approach may not yield extremely accurate results. If extreme accuracy is desired, then traditional methods for finding hardening parameters might be a better approach. On the other hand, if the goal is to get an overall idea of the parameter values, using the developed neural network tool might be a quick and easy way to achieve the goal.

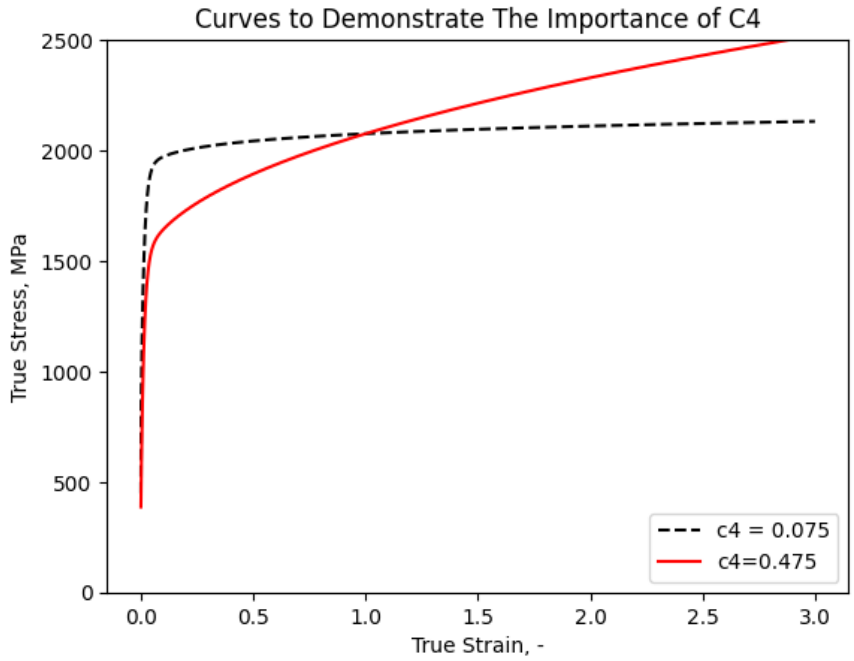

Figure 43. *Figure illustrating the impact of c4 in stress-strain curves when all the other parameters are kept the same.*

5.2 Alternative models

Surprisingly, the Random Forest model gave results with higher R-squared values. For 25°C data, while the Random Forest model consistently got over 0.65 R-squared value, the neural network model averaged at 0.5. This was quite shocking because the Random Forest model was much easier to construct, it ran faster, and it had a more simple structure compared to the neural network model. Despite all these, it gives more stable results and captures the relationship in the data better. In short, the Random Forest might be a more efficient alternative to the neural network. Similar observations were found in the XGBoost model as well. Although the R-squared value was not as high as that of the Random Forest, it was slightly higher than for the neural network model. Similarly to the Random Forest, XGBoost was much easier to construct and ran faster. With that said, neural networks will possibly have a higher R-squared value in more complex data sets provided that there is more input data. For example, if there were 20 parameters to be predicted and 15 input features, the neural network might have performed better compared to Random Forest or XGBoost.

5.3 Validity of models

It is discovered that a higher R-squared value may not necessarily mean better predictions. A model with a high R-squared on the training data may perform

poorly on new, unseen data. This explains why some of the predictions from models with lower R-squared were more accurate compared to models with higher R-squared. Therefore, it's not advisable to rely solely on the R-squared value when evaluating the quality of predictions. A more reliable method for evaluating prediction accuracy is to analyze the stress-strain curves and force-displacement curves derived from the predictions. Predictions by all models have resulted in somewhat acceptable force-displacement and stress-strain curves, although there is room for improvement for these.

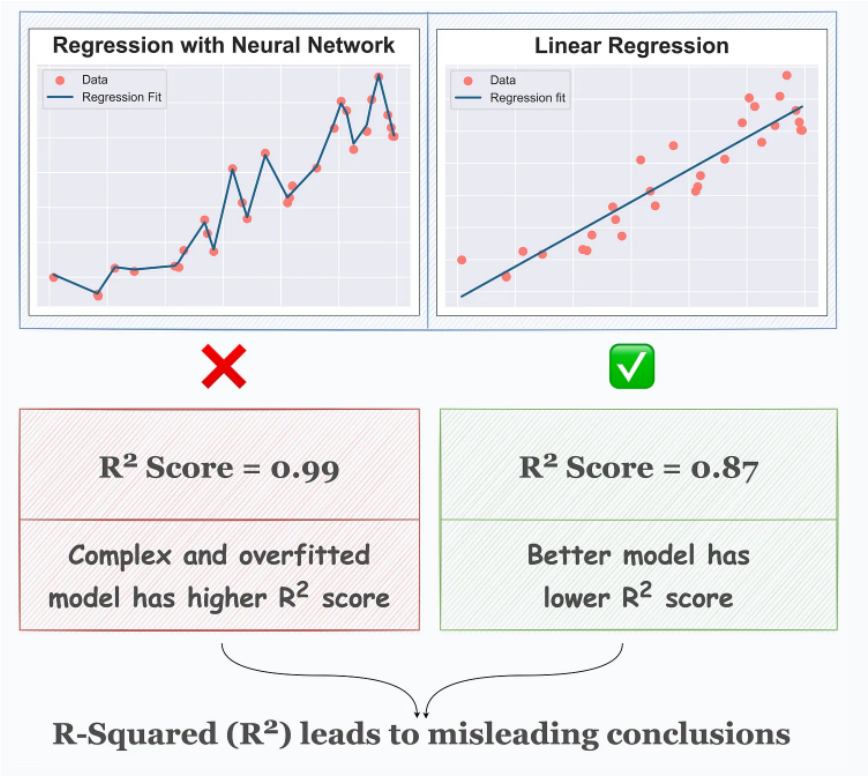

Figure 44. *Diagram showing R-squared score may be misleading. [46]*

The force-displacement curves indicate that:

- For 25°C data, Neural Network predictions have resulted in the most accurate force-displacement curves. Random Forest comes second with a small difference and XGBoost comes in third with a large difference.
- For 400°C data, Random Forest and XGboost predictions have resulted in more accurate force-displacement curves than Neural Network predictions.
  - NDBR50 and NDBR6 curves are slightly more accurate in XGBoost compared to Random Forest.
  - CHD6 curve is way more accurate in the Random Forest model compared to the XGBoost model.

  - It is not possible to comment on the accuracy of NDBR20 as the experimental data for 400°C is not provided.

The stress-strain curves indicate that:

- For 25°C data, Neural Network predictions have resulted in the curve with the best yield point. Random Forest comes second and XGBoost is the last.
  - Although the Neural Network curve has the best yield point, it passes the experimental curve when the strain is about 1.5. It remains slightly above the experimental curve from a strain of 1.5 to 3.0.
  - Although the XGBoost curve has the worst yield point, it has the best radius of curvature among the curves.
  - Random Forest curve becomes the most accurate curve as we go closer to the Strain of 3.0. Towards the end, the Neural Network curve is already past the experimental curve while the Random Forest is significantly close to the experimental curve.
- For 400°C data, XGBoost predictions have resulted in the curve with the best yield point. Random Forest comes in second and Neural Networks is the last.
  - Although the XGBoost curve has the best yield point, it passes the experimental curve when the strain is about 1.25. It remains slightly above the experimental curve from a strain of 1.25 to 3.0.
  - Although the Neural Networks curve has the worst yield point, it has the best radius of curvature among the curves.
  - Random Forest curve becomes the most accurate curve as we get closer to the Strain of 3.0. Towards the end, the XGBoost curve is already past the experimental curve while Random Forest is significantly closer to the experimental curve.

5.4 Effect of Temperature

Another finding is when the model had 400°C data as the input, it yielded worse results. This is likely because the project team was not provided with the f/d data for the NDB20 geometry at 400°C. Thus, the model was trained with data from three geometries, unlike for 25°C, where data for four geometries was utilized. This meant that 400°C data contained 14847 rows of force-displacement data,

while 25°C data contained 20200 rows. Since the model had much less data for training for 400°C, the predictions turned out to be worse.

5.5 Limitations

There are some things to be considered if the developed models are to be used by others. Before using any of the models, it is important to check out the README.md file and make the necessary adjustments. Using the correct file types and structures is necessary for the model to run properly.

6 Conclusion

This section summarizes the project's main findings and thereby gives the concluding remarks. The main findings are listed below:

- The ML models developed offers a fast and computationally efficient method to obtain preliminary insights into potential values for hardening parameters. However, complete reliance on them may not guarantee the utmost accuracy, at least with the amount of input data that was at the team's disposal.
- The R-squared value of the models is directly linked to the quantity of data. To improve data representation and boost the R-squared value, increasing the amount of data is crucial.
  - For each model, the predictions for 400°C had a lower R-squared value since the input data set for 400°C was smaller than that for the data for 25°C.
- A higher R-squared value may not always indicate superior predictions. A more effective approach to assess the prediction quality is to examine the constructed stress-strain curves (flow curves) and force-displacement curves.
  - Both the XGBoost and Random Forest models exhibited higher R-squared values, quicker performance, and simpler construction in comparison to the Neural Network model. Nonetheless, even though the neural network model had a lower R-squared value, its predictions with 25°C data resulted in the most accurate stress-strain and force-displacement curves.
  - The Neural Network showed better performance with a larger data set (25°C) but struggled when there was less input data (400°C). This indicates that the Neural Network relies heavily on having a substantial data set to perform optimally. Unlike NN, XGB and RF performed well even with smaller data sets.
- Even minor deviations in certain parameters, such as the hardening exponent, can lead to significant variations in stress-strain curves.
- Swift-Voce offers a computationally efficient approach for constructing stress-strain curves. However, inaccuracies in predicting the weight parameter can lead to the dominance of either Swift or Voce law.

7 Future Research

This section presents the suggestions for future research. These are summarized in the four points listed below:

- While current literature predominantly employs neural network-based models for predicting hardening parameters, there's a need to explore the potential of alternative algorithms like Random Forest and XGBoost.
- Further research on individual hardening parameters would be of benefit, especially considering the challenge of identifying basic information about certain parameters.
- The project team suggests that ML engineers and materials scientists collaborate on these kinds of projects. ML engineers know how to create effective models, while materials scientists understand the materials. Working together improves model accuracy and makes the outcomes more meaningful.
- To improve the neural network ML model developed and presented in this paper, the project team suggests considering using hyperparameter optimization techniques like grid search or random search. These methods can enhance performance by systematically testing different hyperparameter combinations.

Personal evaluation

7.1 Reflections

San Learning to use CSC, understanding how everything relates to each other, and understanding how machine learning works were the main challenges. This was probably the only project that I worked on for this long and this much in my entire life. Coming up with a relatively accurate model was incredibly challenging but I enjoyed the process. I had no ML experience before this project but, I think I learned a lot of stuff during the process. I realized that I enjoyed ML, and decided to continue with the Macadamia Master's program in Aalto. Thus, in a way, this course will affect the rest of my life. Also, when I first read the requirements of this project, I thought I would never be able to do it. But it turns out that it was doable. So in a way, this course "forced" me to discover my potential.

Crista The project has been demanding in many respects. For me personally, it was a huge challenge to fit together my other commitments with the work required for the project — especially while working full-time along with my studies. Due to this, I feel like my contribution was not as big as I would have liked. I do still feel like I was able to contribute to the work in a meaning full way and learned some new things as well. I found it really interesting to see how a ML model is actually built after gaining a theoretical understanding of ML algorithms from my thesis work. The fact that we were able to construct a working model after being very lost in the beginning of the project is something to be very proud of. In that sense, being part of the positive progress throughout the project has been rewarding and I would like to extend my thanks to my teammates for making this a good experience and also our advisor Zinan who provided help and support. The main take away for me from this project is the increased knowledge in machine learning. This knowledge is definitely something I see I could utilise in my professional life in the future.

Akin When we first started this project, not only did I not know most of the concepts that we were going to work with, but there were many other relevant concepts that I didn't even realize that I didn't know at the time. Learning this many concepts in a short amount of time that I hadn't even heard before was a big struggle. I had to read a lot just to understand what we would be trying to achieve in this project. Understanding the theory was a challenge especially

when combined with the fact that I was taking several courses at the same time. The learning process was very draining but rewarding at the same time. The more I read and learned, the more confident I got that we would be able to do this project successfully. We started to take significant steps as we devoted all our energy into researching and experimenting. Both of my teammates were very collaborative and easy to communicate with. Our advisor, Zinan, was also helpful and supportive whenever we were stuck and needed guidance. 3 months ago, I would have never thought that I would learn this much in a relatively short amount of time. I am still by no means an ML expert, however, this project has helped me to gain skills and knowledge that I never would have expected to gain in only 3 months.

Appendix

Access to Repository and Further Remarks

The repository for this project can be accessed through this link. Data files are not pushed to the main branch as they are huge. The size of the data files exceeds the limit of data we can push to GitHub on the free plan. Files that are not committed to the main branch are the files with the following extensions: .csv, .so, .rpt, .pth, .txt, .npy, .png.

The files used within the CSC (i.e. all the .inp, .sh, .py files), are provided by advisor Zinan Li and edited by the project team. Since the original files do not belong to the project team, these cannot be shared publicly within the team's repository. If you need the aforementioned files for your project, please contact zinan.li@aalto.fi.

Refer to these videos for step-by-step explanations and execution of our code.

References

[1] M. Peksen, "Numerical thermomechanical modelling of solid oxide fuel cells," *Progress in Energy and Combustion Science*, vol. 148, pp. 1–20, 2015.

[2] Y. Li, J. He, B. Gu, and S. Li, "Identification of advanced constitutive model parameters through global optimization approach for dp780 steel sheet," *Procedia Engineering*, vol. 207, pp. 125–130, 2017.

[3] K. Prasad, D. Kumar, H. Krishnaswamy, and D. K. Banerjee, "Uncertainties in the swift hardening law parameters and their influence on the flow stress and the hole expansion behavior of dual-phase (dp600) steel specimens," *Journal of Materials Engineering and Performance*, vol. 32, p. 9206–9220, 2023.

[4] F. Reventos, *Thermal-Hydraulics of Water Cooled Nuclear Reactors*, ch. 3. Woodhead Publishing, 2017.

[5] GIS, "What are inverse problems?," Oct. 2023. *Available at* `http://inverseprobleme.de/?page_id=285&lang=en`. [Accessed 23-10-2023].

[6] S. Guo, "Solving inverse problems with physics-informed deeponet: A practical guide with code implementation," July 2023. *Available at* `https://towardsdatascience.com/solving-inverse-problems-with-physics-informed-deeponet-a-practical-guide-with-code-implementation-27795eb4f502`. [Accessed 9-10-2023].

[7] R. Pugliese, S. Regando, and R. Marini, "Machine learning-based approach: global trends, research directions, and regulatory standpoints," *Data Science and Management*, vol. 4, pp. 19–29, 2021.

[8] J. Fumo, "Types of machine learning algorithms you should know," June 2017. *Available at* `https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861.`. [Accessed 9-10-2023].

[9] V. Yathish, "Loss functions and their use in neural networks." `https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9#:~:text=A%20loss%20function%20is%20a,the%20predicted%20and%20target%20outputs.`, 2022. [Accessed 07-12-2023].

[10] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, no. 20, pp. 295–316, 2020.

[11] A. Rakhecha, "Understanding learning rate." `https://towardsdatascience.com/https-medium-com-dashingaditya-rakhecha-understanding-learning-rate-dd5da26bb6de`, 2019. [Accessed 08-12-2023].

[12] F. Bre, J. Gimenez, and V. Fachinotti, "Prediction of wind pressure coefficients on building surfaces using artificial neural networks," *Energy and Buildings*, vol. 158, 11 2017.

[13] W. Liu, J. Lian, S. Münstermann, C. Zeng, and X. Fang, "Predicition of crack formation in the progressive folding of square tubes during dynamic axial crushing," *International Journal of Mechanical Sciences*, vol. 176, p. 105534, 2020.

[14] Y. Zhang, L. Guo, C. J. Brousse, C.-H. Lee, A. Azoug, H. Lu, and S. Wang, "Machine learning based inverse modelling of full-field strain distribution for mechanical characterization of a linear elastic and heterogeneous membrane," *Mechanics of Materials*, vol. 165, p. 104134, 2022.

[15] L. Petureau, P. Doumallin, and F. Bremand, "Identification of local elastic parameters in heterogeneous materials using a parallelized femu method," *International Journal of Applied Mechanics and Engineering*, vol. 24, no. 4, pp. 140–156, 2019.

[16] R. Lourenço, A. Andrade-Campos, and P. Georgieva, "The use of machine-learning techniques in material constitutive modelling for metal forming processes," *Metals*, vol. 12, no. 3, p. 427, 2022.

[17] D. Yao, S. Pu, M. Li, Y. Guan, and Y. Duan, "Parameter identification method of the semi-coupled fracture model for 6061 aluminium alloy sheet based on machine learning assistance," *International Journal of Solids and Structures*, vol. 254-255, no. 111823, 2022.

[18] Y. Lin, J. Zhang, and J. Zhong, "Application of neural networks to predict the elevated temperature flow behavior of a low alloy steel," *Computational Materials Science*, vol. 43, no. 4, pp. 752–758, 2008.

[19] L. feng Guo, B. cheng Li, and Z. min Zhang, "Constitutive relationship model of tc21 alloy based on artificial neural network," *Transactions of Nonferrous Metals Society of China*, vol. 23, pp. 1761–1765, 2013.

[20] J. Wang, B. Zhu, C.-Y. Hui, and A. T. Zehnder, "Determination of material parameters in constitutive models using adaptive neural network machine learning," *Journal of the Mechanics and Physics of Solids*, vol. 177, no. 105324, 2023.

[21] D. Chen, Y. Li, X. Yang, W. Jiang, and L. Guan, "Efficient parameters identification of a modified gtn model of ductile fracture using machine learning," *Engineering Fracture Mechanics*, vol. 245, no. 107535, 2021.

[22] A. Hajari, M. Morakabati, S. M. Abbasi, and H. Badri, "Constitutive modeling for high-temperature flow behavior of ti-6242s alloy," *Materials Science Engineering A*, no. 681, pp. 103–113, 2017.

[23] X. Liu, S. Tian, F. Tao, and W. Yu, "A review of artificial neural networks in the constitutive modeling of composite materials," *Composites Part B: Engineering*, vol. 224, no. 109152, 2021.

[24] N. Bastos, P. A. Prates, and A. Andrade-Campos, "Material parameter identification of elastoplastic constitutive models using machine learning approaches," *Key Engineering Materials*, vol. 926, pp. 2193–2200, 2022.

[25] A. Lev, "Xgboost versus random forest." `https://www.qwak.com/post/xgboost-versus-random-forest`, 2022. [Accessed 07-12-2023].

[26] V. Balakrishnan, Z. Shi, C. Law, R. Lim, L. Teh, and Y. Fan, "A deep learning approach in predicting products' sentiment ratings: a comparative analysis," *The Journal of Supercomputing*, vol. 78, 04 2022.

[27] G. Seif, "Understanding the 3 most common loss functions for machine learning regression." https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3. [Accessed 04-12-2023].

[28] J. Jordan, "Setting the learning rate of your neural network.." https://www.jeremyjordan.me/nn-learning-rate/. [Accessed 04-12-2023].

[29] J. Feng, X. He, Q. Teng, C. Ren, H. Chen, and Y. Li, "Reconstruction of porous media from extremely limited information using conditional generative adversarial networks," *Physical Review E*, vol. 100, 09 2019.

[30] K. Leung, "The dying relu problem, clearly explained." https://towardsdatascience.com/the-dying-relu-problem-clearly-explained-42d0c54e0d24. [Accessed 04-12-2023].

[31] S. M, "Learn about principal component analysis in details." https://analyticsvidhya.com/blog/2022/03/learn-about-principal-component-analysis-in-details/. [Accessed 03-12-2023].

[32] "How to train and test data like a pro." https://sdsclub.com/how-to-train-and-test-data-like-a-pro/. [Accessed 03-12-2023].

[33] B. Gupta, "Feature scaling in machine learning scaler topics." https://www.scaler.com/topics/machine-learning/feature-scaling-in-machine-learning/. [Accessed 03-12-2023].

[34] C. GOYAL, "The game of increasing r-squared in a regression model." https://www.analyticsvidhya.com/blog/2021/05/the-game-of-increasing-r-squared-in-a-regression-model/. [Accessed 04-12-2023].

[35] Naveen, "What is l1 and l2 regularization in deep learning?." https://www.nomidl.com/deep-learning/what-is-l1-and-l2-regularization-in-deep-learning/. [Accessed 03-12-2023].

[36] J. de Dieu Nyandwi, "Early stopping explained!." https://jeande.medium.com/early-stopping-explained-62eebce1127e. [Accessed 04-12-2023].

[37] F. López, "Skorch: Pytorch models trained with a scikit-learn wrapper." https://towardsdatascience.com/skorch-pytorch-models-trained-with-a-scikit-learn-wrapper-62b9a154623e. [Accessed 03-12-2023].

[38] J. R. Mulkey, "Strain hardening exponent for 17-4 ph stainless steel.," 1 1971.

[39] T. Huang, R. Gou, W. Dan, and W. Zhang, "Strain-hardening behaviors of dual phase steels with microstructure features," *Materials Science and Engineering: A*, vol. 672, pp. 88–97, 2016.

[40] J. M. G. Farabi Bre and V. D. Fachinotti, "Prediction of wind pressure coefficients on building surfaces using artificial neural networks," *Energy and Buildings*, vol. 158, pp. 1429–1441, 2018.

[41] D. P. R. Palaparti, B. Choudhary, and T. Jayakumar, "Influence of temperature on tensile stress-strain and work hardening behaviour of forged thick section 9cr–1mo ferritic steel," *Transactions of the Indian Institute of Metals*, vol. 65, 10 2012.

[42] Y. He, W. Li, M. Yang, Y. Li, X. Zhang, Z. Zhao, P. Dong, J. Ma, and S. Zheng, "Modeling of temperature-dependent ultimate tensile strength for metallic materials," *Journal of Constructional Steel Research*, vol. 191, p. 107184, 2022.

[43] "Supercomputer puhti is now available for researchers." https://www.csc.fi/en/-/supertietokone-puhti-on-avattu-tutkijoiden-kayttoon. [Accessed 04-12-2023].

[44] H. Sahour, V. Gholami, J. Torkman, M. Vazifedan, and S. Saeedi, "Random forest and extreme gradient boosting algorithms for streamflow modeling using vessel features and tree-rings," *Environmental Earth Sciences*, vol. 80, 11 2021.

[45] D. Şen, M. Hüseyinoğlu, and M. Günay, "Prediction of global temperature anomaly by machine learning based techniques," *Neural Computing and Applications*, vol. 35, pp. 1–14, 04 2023.

[46] A. Chawla, "Why r-squared is a flawed regression metric." https://www.blog.dailydoseofds.com/p/why-r-squared-is-a-flawed-regression. [Accessed 03-12-2023].