applied sciences

MDPI

Article

AI-Assisted Creep Time Prediction Using Creep Strain Curves of AISI 316 Austenitic Stainless Steel: Effects of Data Transformation and Hyperparameter Optimisation

Arsalan Nazim [1,*], Andrea Tonti [2] and Elisabetta Gariboldi [1]

[1] Department of Mechanical Engineering, Politecnico di Milano, 20156 Milano, Italy; elisabetta.gariboldi@polimi.it
[2] Istituto Nazionale per l'Assicurazione contro gli Infortuni sul Lavor, 00143 Rome, Italy; a.tonti@inail.it
* Correspondence: arsalan.nazim@polimi.it

Featured Application

The proposed AI framework can be used to predict creep time behaviour of austenitic stainless steel employed in high-temperature applications such as boilers, heat-exchangers and steam piping systems, thereby supporting creep life assessment.

Abstract

High-temperature structural components are susceptible to creep deformation, which can ultimately lead to failure. In this work, an AI-based framework was developed capable of predicting the creep time of 316 austenitic stainless steel. Here, creep time refers to both the time to reach specific strain levels and the time to rupture. However, the scope of the present work is limited to rupture-time prediction, while the application of the framework to strain-level prediction will be reported in future work. The dataset consisted of creep strain curves from four heats, including both rupture and non-rupture curves. Random Forest (RF), Gradient Boosting (GB), Extreme Gradient Boosting (XGB), Support Vector Regressor (SVR), Gaussian Process Regressor (GPR), and Neural Network (NN) were employed. The effects of square-root and cube-root transformations on data distribution and model learning behaviour were analysed using model learning curves. An Optuna (version 4.3.0)-based hyperparameter tuning strategy was employed. The cube-root transformation improved the learning performance of SVR, GPR, and NN, whereas RF, GB, and XGB remained unaffected. Learning curves revealed mild overfitting for RF, GB, and XGB, and very minimal overfitting for SVR, GPR, and NN. NN achieved the best predictive performance ($R^2 = 0.92$, RMSE $= 0.195$, deviation factor of 1.57). The findings demonstrated that the combined useof creep strain curves, data transformation, learning curve guided model selection, and rigorous hyperparameter tuning can improve the prediction accuracy under a limited dataset.

Keywords: machine learning; creep time prediction; austenitic stainless steels; Optuna; learning curves

Academic Editor: Nicanor Cimpoesu

Received: 4 June 2026
Revised: 16 June 2026
Accepted: 17 June 2026
Published: 23 June 2026

1. Introduction

316 austenitic stainless steel (ASS) is widely used in high-temperature components, such as heat exchanger tubes and other components in power, chemical, or petrochemical plants, owing to its combination of excellent mechanical strength and outstanding corrosion resistance [1,2]. A low-carbon, nitrogen-alloyed variant, 316LN, offers improved

performance through reduced carbide precipitation and enhanced solid-solution strengthening [3]. Still, 316LN steels remain susceptible to creep deformation during prolonged service under loading at elevated temperatures [4]. Creep, the time-dependent plastic strain under constant stress, progresses through primary, secondary, and tertiary stages before culminating in the final rupture of components [5,6], which should be avoided in structural parts.

In the past, several strategies have been developed to assess component life under creep conditions. Most of them are related to rupture times under given temperature and stress conditions. These include time–temperature parameter (TTP) models, such as the Larson–Miller parameter [7], Manson–Haferd method [8], Sherby–Dorn parameter [9], and Monkman–Grant relationship [10], as well as phenomenologically or mixed physical/phenomenological based models such as the Theta Projection Method [11] and BJF model [12]. However, both model categories have critical limitations. TTP-based approaches strongly depend on the availability and quality of experimental data and typically consider only the constant stress and temperature adopted as creep testing conditions, conditions for which model lifetime predictions will also be provided. Therefore, the role of additional important features, such as the actual chemical composition and information on the initial microstructure (resulting from processing and heat treatments) and its evolution during service, is neglected. On the other hand, physically based models require good knowledge of creep processes and parameter fitting. Their parameters are often alloy-specific and valid only under limited operating conditions, which limits their wide applicability [13–17].

In recent years, Artificial Intelligence (AI) has appeared as a strong alternative to conventional creep life prediction models. AI algorithms can capture complex, nonlinear relationships directly from data, potentially improving prediction accuracy [18]. Bhardwaj et al. [1,2] demonstrated the capability of Deep Neural Networks (DNNs) and ensemble models to predict the creep and fatigue life of 316 ASS, achieving accuracies above 95%. Baraldi et al. [6] further combined an Artificial Neural Network (ANN) with a phenomenological model for 316LN steels, highlighting the potential of hybrid approaches. Similarly, Zhang et al. [19] and Wei et al. [16] showed that machine learning algorithms such as Random Forest (RF), Support Vector Regressor (SVR), and Gaussian Process Regressor (GPR) consistently outperformed conventional time–temperature parameters in predicting rupture life for heat-resistant austenitic alloys. In addition to austenitic stainless steels, comparable success has been reported in other alloy systems. Chai et al. [15] applied regression and kernel-based methods to 9Cr–1Mo steels. Qin et al. [18] demonstrated the use of Convolution Neural Network (CNN) and Support Vector Machine (SVM) for classifying creep regimes, and Sakurai et al. [20] applied RF, Extreme Gradient Boosting (XGB), SVR, and ANN for ferritic steels.

Despite these advances, several gaps remain. For instance, most previous investigations have primarily focused on rupture-time prediction using datasets based on rupture time and operational parameters such as stress and temperature. Furthermore, existing investigations predominantly rely on logarithmic transformations due to their physical basis, rooted in Arrhenius-type equations [21,22]. However, the role of transformation in machine learning is mainly to reduce the data skewness. In fact, ML models tend to perform better when the data is more symmetric or more Gaussian-like [23]. Thus, beyond logarithmic transformations, investigating the influence of other power transformations, such as the square-root and cube-root, on dataset distribution (skewness) and model performance is crucial. To the best of the author's knowledge, use of such alternative power transformation is not well documented, especially in the creep-related literature. Moreover, a systematic evaluation of how these transformations affect model learning

behaviour, particularly through learning curve analysis, is also less explored. Additionally, hyperparameter optimisation has largely been restricted to conventional techniques such as grid and random search, which often fail to fully utilise the potential of other ML optimisation strategies.

Furthermore, a common challenge across creep-domain investigations is the limited data availability of long-term experimental data. This motivates the development of robust data-driven modelling approaches. Similar efforts have been implemented in other high-temperature engineering domains. For instance, Fan et al. [24] recently employed reduced-order modelling combined with multi-objective optimisation for high-temperature engineering applications, highlighting the broader applicability of data-driven models under data availability limitations.

To address these gaps, the present work proposes an AI-based framework in which several AI models are trained to predict the creep time, where creep time refers to the time to reach specific strain levels and the time to reach final rupture, using creep strain curves of 316 austenitic stainless steel. However, at present, only the results corresponding to rupture time are reported and discussed. The application of the present framework to predict time to reach specific strain levels will be reported separately in future work.

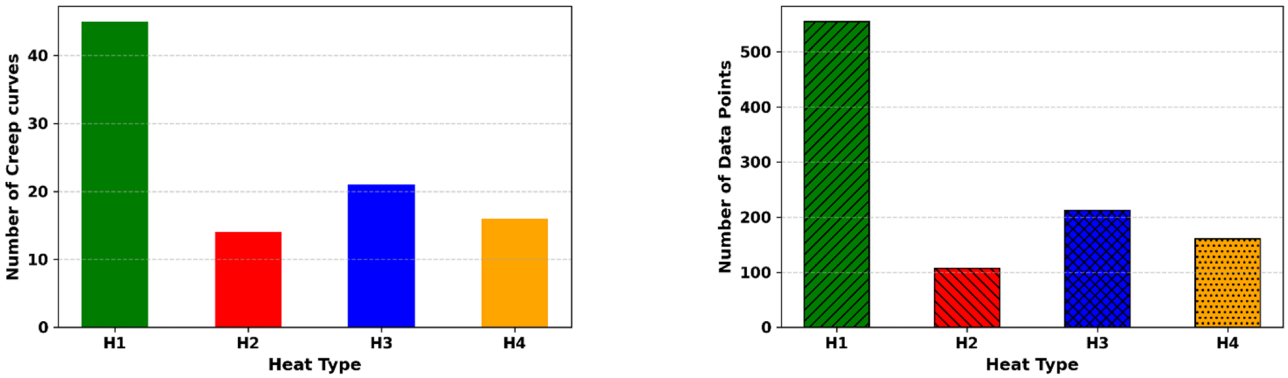
More broadly, the use of complete signal histories as inputs to data-driven models are receiving increased attention. The recent work of Qiao et al. [25] on digital-twin utilising the full signal histories is one of the example. In the present work, full creep strain curves are employed rather than only the rupture time as the model input. Several machine learning models, including Random Forest, Gradient Boosting, Extreme Gradient Boosting, Support Vector Regression, Gaussian Process Regression, and Neural Network, were employed. Alternative power transformations, including the square-root and cube-root, are systematically analysed to quantify their influence on model learning behaviour and predictive accuracy. For advanced hyperparameter optimisation, the Optuna framework [26] was used to achieve more efficient model tuning. Model performances were evaluated using the commonly employed statistical parameters in machine learning such as Coefficient of Determination (R^2), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

2. Materials and Methods

2.1. Data Description

The present study focuses on austenitic stainless steel, AISI 316 grade (18Cr13NiMo). Creep strain data were available from the European Creep Collaborative Committee (ECCC) and were the same ones on which previous assessments of creep models were applied, providing a more conventional analytical creep description [27]. From the complete dataset of 98 creep curves, 3 curves were excluded, since only a single data point was present for each curve. These single data points correspond to the isolated rupture measurement and hence lag the associated creep curve information. Of the remaining 95 curves, 73 were associated with specimens that reached the final fracture. The curves that remained ongoing at the time of data extraction are classified as non-rupture curves. These curves are particularly valuable because they provide information on the early stages of creep, specifically the primary and secondary regimes.

The tests cover temperatures from 500 °C to 700 °C, while the stress (MPa) range is about 2 orders of magnitude and the time (hours) range is about 3 orders of magnitude. Moreover, the dataset comprises four heats: H1 (green), H2 (red), H3 (blue), and H4 (orange). The distribution of creep curves per heat is shown in Figure 1a. H1 yields the most curves (44), followed by H3 (21), H4 (16), and H2 (14). In addition, the available data points per heat are reported in Figure 1b. On average, H1 provides the most rows per test (543), whereas H2 provides the fewest (107).

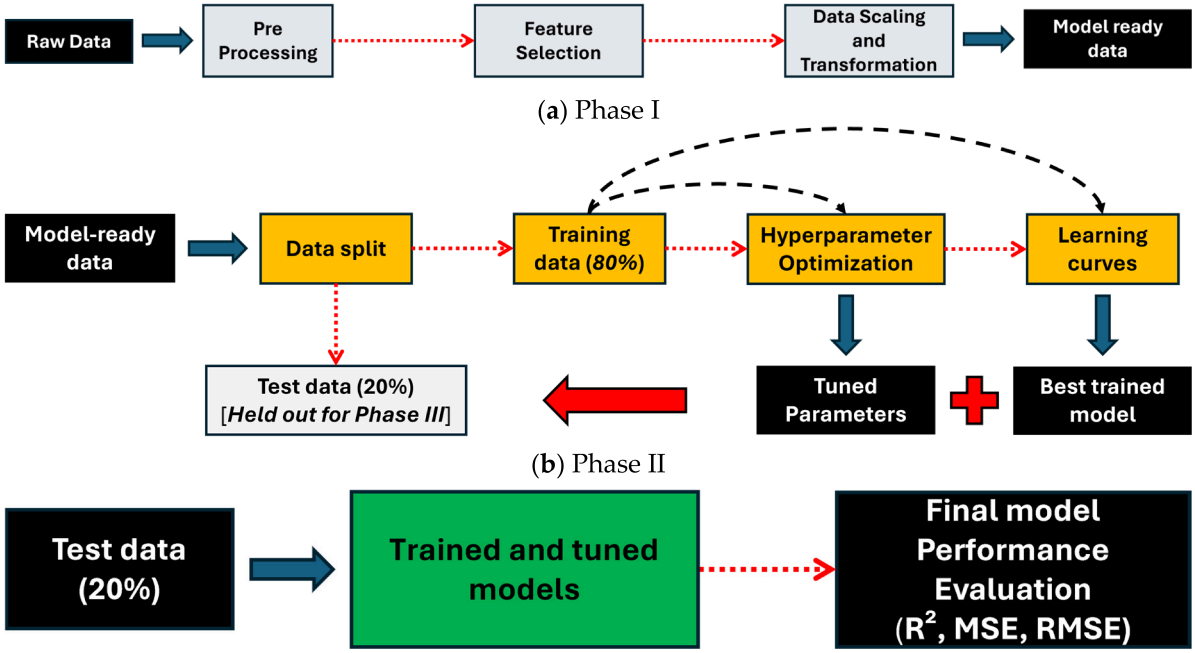


(**a**) (**b**)

Figure 1. Distribution of the dataset across heats: (**a**) number of creep curves per heat; (**b**) number of available time–strain rows per heat. To improve the graphical difference between the two distribution classes, different hatch patterns are included in (**b**) corresponding to different heat.

2.2. Methodological Pipeline

The overall methodology adopted in this study is summarised in Figure 2. In the following subsections, each stage of the pipeline is described in detail.



(**a**) Phase I

(**b**) Phase II

(**c**) Phase III

Figure 2. Methodology pipeline for rupture-time prediction. (**a**) Phase I: Data setup and feature selection. (**b**) Phase II: Model training and hyperparameter optimisation. (**c**) Phase III: Creep time prediction.

2.2.1. Phase I—Dataset Setup and Feature Selection

The creep data used in this study were originally available as time–strain rows for each creep test performed at a particular stress and temperature. Afterwards, the dataset was reorganised into a tabular format, where each row represents a single measurement point along a creep curve. The columns include identifiers such as heat, creep curve type, test condition (true stress in MPa and temperature in K), and recorded outputs (time in hours and true strain in percent). In general, most conventional machine learning algorithms are not inherently capable of handling missing or duplicate entries within the dataset [23,28].

In the current dataset, a proper check confirmed that no missing values were present in the total 1023 rows of the dataset.

The next step involved feature selection. Feature selection aims to identify the input variables that are most important to the target variable. Among the commonly used methods, the Pearson Correlation Coefficient (PCC) is widely employed to quantify the strength of a linear relationship between two variables (input and features). The correlation coefficient ranges from −1 to +1, where values close to +1 or −1 indicate strong positive or negative correlations, respectively. Such methods are mainly useful when dealing with a large number of potential input features [23]. However, in the present study, the dataset comprises only a few variables, including stress, temperature, strain, heat information, and curve type, which serve as the model inputs and have a direct and physical significance with the output, creep time. Furthermore, since no microstructural, chemical, or other data are available for the investigated austenitic stainless steel, they have not been considered in the present investigation.

Heats (H1–H4) representing the categorical data were encoded as numerical data using One-Hot Encoding (OHE) [23]. After the application of OHE, the categorical heat variable was transformed into a binary vector format (e.g., Heat 1 represented as). Similarly, to distinguish data from rupture or non-rupture creep curves, a binary flag variable (is_rupture) [23] was introduced, allowing the model to identify whether a data point corresponds to a rupture or non-rupture test. All numerical input features and the target variable were normalised to the range [0, 1] using the Min–Max scaling technique, as expressed in Equation (1). Data scaling is necessary to remove bias and variance that may be caused by variables with different physical units and magnitudes (e.g., MPa, Kelvin, millimetres [23].

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

where X is the original unscaled value, X_{scaled} is the normalised value, and, based on the training dataset, X_{min} and X_{max} represent the minimum and maximum values of the feature, respectively.

Data transformations (the final step of Phase I) are then widely employed to bring the distributions of variables closer to the Gaussian distribution (minimal skewness). Generally, variables with a Gaussian distribution tend to improve model performance [23,29]. Logarithmic transformation has been predominantly used for data transformation in the field of creep behaviour [28,30,31], and it also follows the classical representation of creep strength plots, where a logarithmic scale is adopted for the time axis, and in some cases also for the stress axis. In the present work, in addition to the logarithmic transformation, two alternative power-based transformations were investigated: the square-root ($\sqrt{x}$) and the cube-root ($\sqrt[3]{x}$). The impact of these transformations was systematically analysed in terms of their ability to reduce skewness in the data, and afterwards their effect on model learning performance, analysed by their learning curves.

2.2.2. Phase II—Model Training and Hyperparameter Optimisation

Phase II begins by splitting the 95 curves into training and testing curves, respectively. In the present work, a quite conventional 80:20 split ratio was adopted [29]. The data rows (829) originating from the 76 training curves were used for the model training, and the data rows (194) corresponding to the 19 test curves were used for testing and assessing the generalisation performance of the model.

To compensate for the issue of data imbalance (see Figure 1a,b), stratified sampling [20] was incorporated into the data-splitting procedure. Stratification was performed at the curve level based on the heats (Heat 1, Heat 2, Heat 3, and Heat 4) to ensure that samples

from each heat were proportionally represented in both the training and test sets. In fact, this approach aims to reduce bias in model learning caused by uneven heat distribution. It is important to highlight that stratification could alternatively have been performed by combining heats and curve type (is_rupture flag). However, the distribution of non-rupture curves was non-uniform across the heats. For instance, Heat 2 and Heat 4 contained only two non-rupture curves each. Under such conditions, combined stratification would have resulted in several heat-curve type subgroups containing very few datasets. This will potentially increase the risk of unstable train–test partitions, resulting in reduced representativeness of individual heats within each subgroup.

Nevertheless, since curve status (is_rupture flag) was not considered for the stratification process, slight differences in the distribution of rupture and non-rupture curves may exist between the training and testing data subsets. To provide a quantitative assessment of the resulting distribution, Table 1 summarises the number of rupture and non-rupture curves in each subset. The training subset contained 60 rupture and 16 non-rupture curves, while the testing subset contained 13 rupture and 6 non-rupture curves. Although slight differences exist between the two subsets, these variations are consistent with the adopted heat-based stratification methodology and the limited availability of non-rupture curves for Heat 2 and Heat 4. Therefore, some level of evaluation bias cannot be entirely ruled-out, and the reported performance metrics should be interpreted in the context of this chosen methodological approach of the present work.

Table 1. Quantitative comparison of rupture and non-rupture curve distribution in training and testing subsets.

Subset	Rupture	Non-Rupture	Total Curves
Training	60	16	76
Testing	13	6	19

To make the models training more robust, training was performed using a 5-fold cross-validation (CV) strategy, following the initial test-train split strategy. The CV strategy has been discussed in the subsequent Section 2.4. and the process is schematically represented in Figure 3. Model behaviour during training was monitored using learning curves, which provided information about model convergence, overfitting, and model generalisation. Based on the learning curves, the most appropriate models were selected for the final prediction on the test dataset.

2.2.3. Phase III—Creep Time Prediction

In the final phase, selected models were used to model the creep time on the unseen 20% test dataset. The generalisation abilities of the models were assessed using three standard performance metrics: the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and the Coefficient of Determination (R^2), defined by Equations (2)–(4) [29].

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

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4}$$

where y_i represents the actual values, $\hat{y}_i$ denotes the predicted values, $\bar{y}$ is the mean of the actual values, and n is the total number of data points.

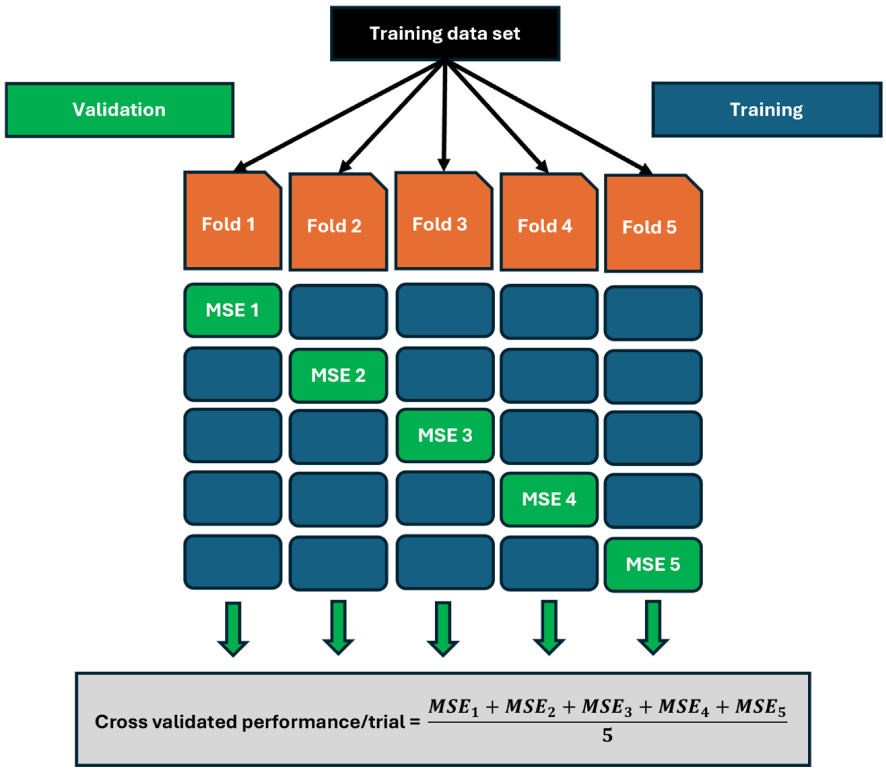


Figure 3. Depiction of the applied 5-fold cross-validation procedure. The training dataset is divided into five equal subsets. In each iteration, 4 folds (blue) were used for training, while the remaining fold (green) was used for validation. The final performance is computed as the average of the mean squared errors across the five validation runs (MSE_1–MSE_5).

2.3. Models

The present investigation employs three classes of machine learning models: (i) Tree-based ensemble models (Random Forest (RF), Gradient Boosting (GB), Extreme Gradient Boosting (XGB)), (ii) Kernel-based regressors (Support Vector Regressor (SVR), Gaussian Process Regressor (GPR)), and (iii) Neural Network (NN). A detailed explanation of the models, along with their respective hyperparameters, is provided in Appendices A and B.

2.4. Hyper Parameter Optimisation (HPO)

In contrast to the model parameters that a model learns automatically during training (such as the weights in neural networks), hyperparameters must be defined before the training process begins. Selecting appropriate hyperparameters is crucial for determining model performance and thus requires careful tuning. In fact, a well-designed and systematic HPO can improve the model's accuracy and generalisation by several times [32].

In previous studies, mainly related to creep behaviour of materials, several researchers [18,33,34], employed Grid Search (GS) and Random Search (RS) for hyperparameter optimisation. GS searches over all possible hyperparameter combinations within a predefined search space, making it computationally intensive. On the other hand, RS randomly samples hyperparameter combinations and evaluates the model's performance for each set. While RS is faster than GS as it does not evaluate every possible combination, both methods can still be computationally challenging. They may fail to identify the very optimal regions of the search space [32].

In the present work, HPO was performed using the Optuna framework (version 4.3.0) [26], a state-of-the-art Python library. Optuna consists of three essential components: (i) a search space, which defines the range of selected hyperparameters. The candidate hyperparameters and their ranges were based on the recommendations reported in [35,36]; (ii) an objective function, evaluates model performance during the optimisation process; and (iii) a search strategy, which guides the selection of optimal hyperparameters [26].

For each model class, a trial corresponds to one complete evaluation of the objective function using a specific combination of hyperparameters. In this study, the number of trials for each model class was initially fixed to 20, progressively increasing to 100. After 100 trials, no further improvement in the objective function was noticed. Eventually, the number of 100 trials yielded a good trade-off between required optimisation accuracy and the computational feasibility. During each trial of Optuna, the training dataset was first divided into five equal folds, which were reused for all trials in turn. Figure 3 illustrates the application of this process. Within each trial, a 5-fold CV procedure was applied. For instance, in each iteration, four folds were used for model training, with the remaining fold functioning as the validation set. In fact, the model was trained five times per trial, each time using a different fold as the validation fold.

The final performance of a trial was computed as the average validation error (MSE) across the five folds. Based on this averaged MSE, Optuna revised its internal probabilistic model using a Bayesian optimisation strategy [37] coupled with the Tree-structured Parzen Estimator (TPE) sampler [38]. This mechanism enabled Optuna to propose new hyperparameter combinations that were more likely to improve model performance in successive trials. Furthermore, the Optuna-based search strategy enabled computational resources to be focused on the most promising areas of the hyperparameter search space, resulting in better convergence and effective optimisation efficiency than conventional GS and RS strategies [35].

2.5. Training and Learning Curves (LC)

Learning curves (LC) provide fundamental understanding into three essential aspects of model behaviour: (i) whether the available amount of data is sufficient for effective model training, (ii) whether the model performance stabilises with increasing data, and (iii) whether the model exhibits signs of overfitting or underfitting, critical issues that needs to be dealt, before using the models for the final prediction on the test dataset.

In this study, LC were constructed for the selected machine learning models (RF, GB, XGB, SVR, and GPR) to systematically investigate the effect of training set size on model generalisation. Additionally, LC were generated separately for different data transformation strategies in order to assess whether, and to what extent, the choice of transformation $(\log(x), \sqrt{x}, \sqrt[3]{x})$, influences model accuracy. Importantly, the present approach allowed the identification of the best-performing transformation strategies that result in the maximum reduction of skewness and later analysed the effect of this skewness reduction on the model learning behaviour.

For each model–transformation combination, LC were constructed by progressively increasing the size of the effective training data subset from 10% to 100% of the available training data in each cross-validation split. To avoid ambiguity, we distinguish between the terms “training data” and “training data subset”. The training data comprises 80% of the complete dataset (i.e., 829 out of 1023 dataset) and is reserved for model development. During internal 5-fold CV, this training data is further partitioned so that approximately 80% (663 dataset) is used for model fitting in each fold. At the same time, the remaining portion serves as the validation fold. Consequently, the maximum effective training size in the LC corresponds to this per-fold training portion rather than the full 828 samples.

Notably, the test set, comprising the remaining 20% of the original dataset (194), is never used during either model training or validation.

The LC are therefore generated by progressively increasing the size of this training data subset, from 10% of the 663 dataset up to 100%. At each training fraction (e.g., 10%, 20%, . . ., 100%), model performance was evaluated using 5-fold CV.

In particular, at a specified fraction, the selected training data subset is partitioned into five equal-sized subsets, called folds. In each iteration, one fold is used as the validation set, and the remaining four are used for model training. This procedure is repeated 5 times, with each fold serving once as the validation set. The corresponding training and validation R^2 scores are recorded for each split and thereafter averaged to obtain the mean performance at that training fraction.

The final training and validation performance at each fraction (e.g., 10%, 20%, . . ., 100%) is then obtained by averaging the R^2 scores across all folds. Moreover, the standard deviation ($\pm 1\sigma$) across folds is computed and visualised as a band in the LC. As training progresses, the width of the band region indicates the model's stability for a given data partition.

3. Results & Discussion

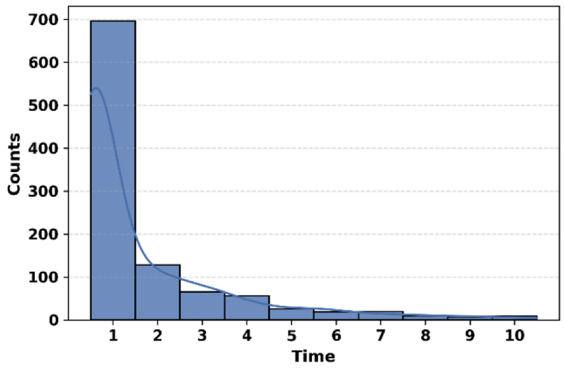
3.1. Data Transformation Strategies

The following section examines the effect of power transformations ($\sqrt{x}$ and $\sqrt[3]{x}$) on reducing the skewness of the target variable, time (h), as well as the two numerical input variables, true strain (%) and true stress (MPa). The distributions of the untransformed variables are shown in Figure 4a–c. All distributions are visualised using histograms, where the *y*-axis shows the count of observations and the *x*-axis shows the variable intervals. For each variable, the range between its minimum and maximum values was divided into ten equal-width bins.

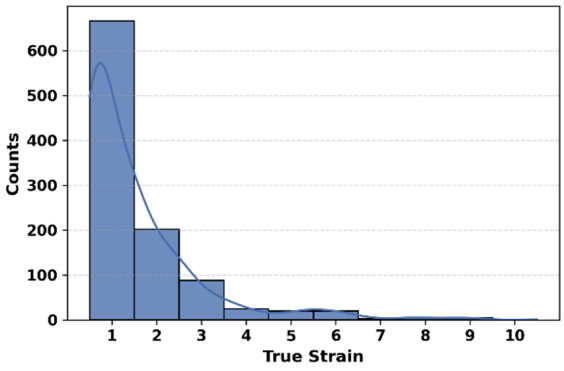
As evident from Figure 4a,b, the untransformed time and true strain distributions exhibit pronounced right-skewed behaviour, with skewness values of 2.193 and 2.563, respectively. This behaviour is expected in creep data, as most observations tend to cluster at lower values, with a limited number of large-magnitude values extending the right tail. Such heavy-tailed distributions lead to high skewness and strong asymmetry. In contrast, the untransformed true stress distribution shows comparatively mild asymmetry, with a skewness of 0.786, closer to the ideal Gaussian value of 0.

Figures 5–7 summarise the effects of the different transformation strategies on the distributions of the variables. On the other hand, Figure 8 further completes this analysis by presenting a skewness profile plot that shows the skewness values computed for both the original, untransformed data and the transformed datasets. This representation provides a quantitative assessment of the degree to which each transformation reduces distributional asymmetry.

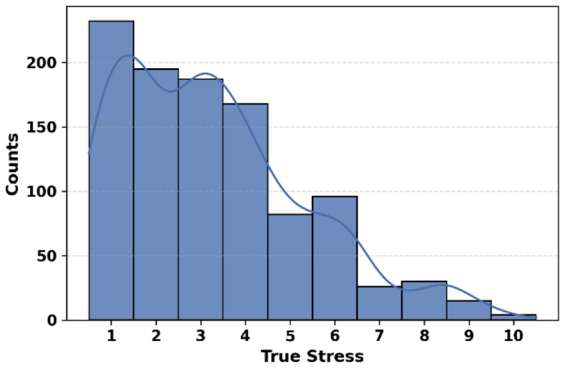
Figure 5a–c summarises the effects of the $\log(x)$ transformation on the variable distributions. For consistency, the exact interval strategy adopted for the original untransformed data was applied to the transformed variables, that is, dividing the range between the minimum and maximum transformed values into 10 equal intervals. This uniform-interval approach allows a direct, visible comparison of how each transformation transforms the underlying distributions.



(**a**) Original Time Distribution

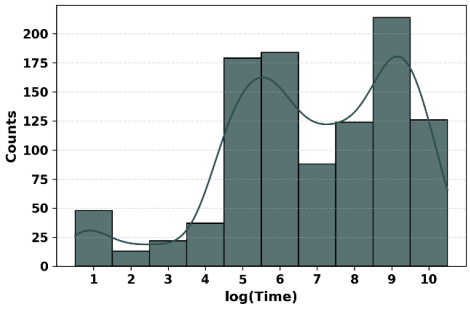


(**b**) Original True Strain Distribution

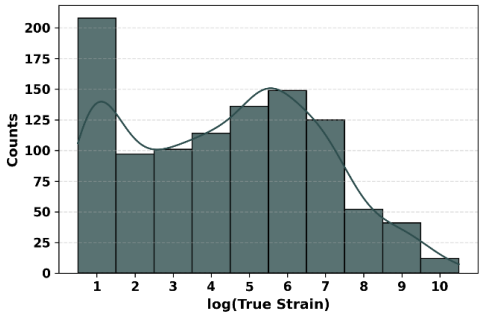


(**c**) Original True Stress Distribution

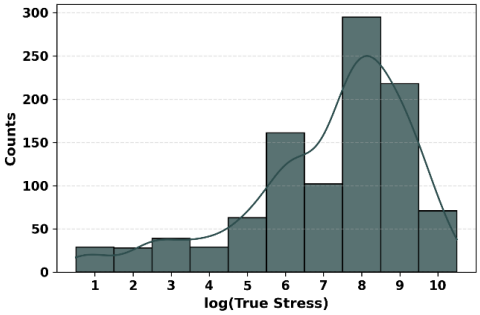
Figure 4. Frequency distributions of the untransformed variables: (**a**) time, (**b**) true strain, and (**c**) true stress. The histograms represent the count of observations within equal-width intervals spanning the respective minimum and maximum values of each variable.



(**a**) Time (log Transformation)

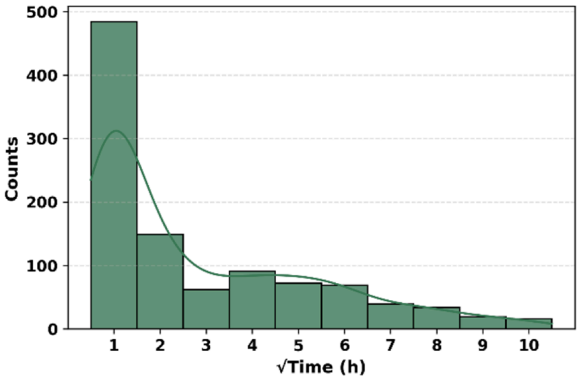


(**b**) True Strain (log Transformation)

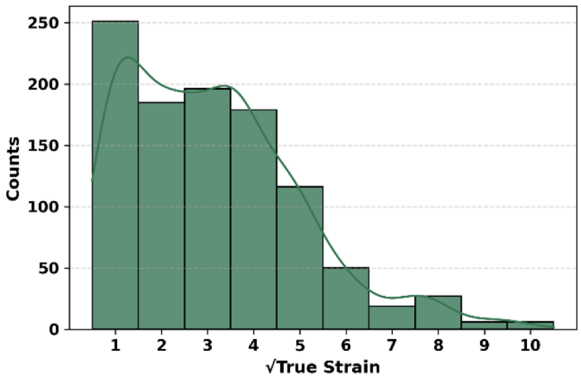


(**c**) True Stress (log Transformation)

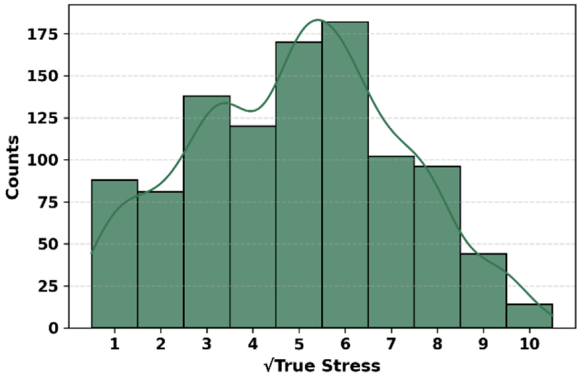
Figure 5. Frequency distributions of the $\log(x)$ transformed variables: (**a**) time (log transformation), (**b**) true strain (log transformation) and (**c**) true stress (log transformation). The histograms represent the count of observations within equal-width intervals spanning the respective minimum and maximum values of each transformed variable.



(**a**) Time (square-root Transformation)



(**b**) True Strain (square-root Transformation)

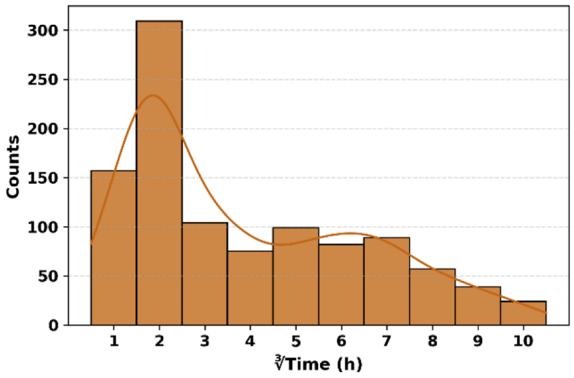


(**c**) True Stress (square-root Transformation)

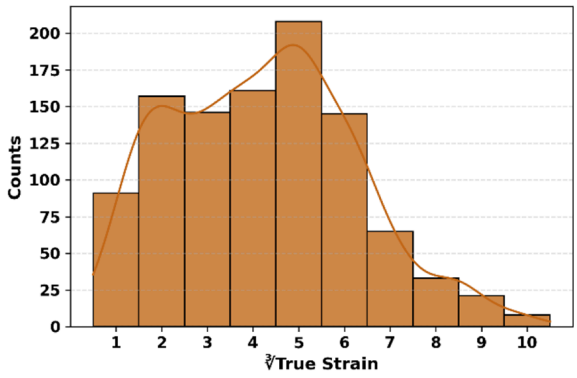
Figure 6. Frequency distributions of the $\sqrt{x}$ transformed variables: (**a**) time (square-root transformation), (**b**) true strain (square-root transformation), and (**c**) true stress (square-root transformation). The histograms represent the count of observations within equal-width intervals spanning the respective minimum and maximum values of each transformed variable.

Compared to the untransformed variables, the $\log(x)$ resulted in a noticeable decrease in skewness for all studied features, i.e., time, true strain, and true stress. As noted earlier, the original variables displayed strong positive skewness, with values of +2.193 for time, +2.563 for true strain, and +0.786 for true stress. Importantly, after the logarithmic transformation, the skewness values decreased for all variables, reaching approximately -1. This demonstrates a considerable change in distribution shape. In addition, the $\log(x)$ transformation reduced the initial right skewness, leading to an overcorrection and distributions that were relatively left-skewed. In fact, the final transformed values of all the variables were outside the preferred skewness range of -0.5 to +0.5 [39], considered in the present investigation, as highlighted by the shaded region in Figure 8. These results imply that while the $\log(x)$ is highly effective in reducing skewness, it does not yield optimal Gaussianity in the present scenario.

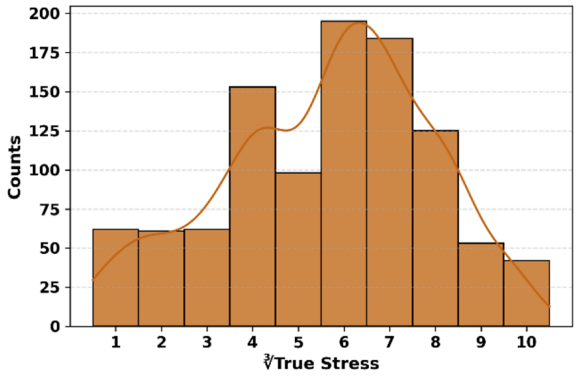
On the other hand, $\sqrt{x}$ transformation (Figure 6a–c) resulted in a near-Gaussian symmetry for true stress, with a final skewness value of -0.033, corresponding to a reduction of nearly 96% compared to the original skewness. However, both time and true strain transformed to positive skewness of around +1, placing them well outside the acceptable range of -0.5 to +0.5. Interestingly, the response of $\sqrt{x}$ transformation for time and true stress was qualitatively opposite to that achieved by the log(x) transformation, in which both variables were overcorrected towards negative skewness.



(**a**) Time (cube-root Transformation)



(**b**) True Strain (cube-root Transformation)



(**c**) True Stress (cube-root Transformation)

Figure 7. Frequency distributions of the $\sqrt[3]{x}$ transformed variables: (**a**) time (cube-root transformation), (**b**) true strain (cube-root transformation), and (**c**) true stress (cube-root transformation). The histograms represent the count of observations within equal-width intervals spanning the respective minimum and maximum values of each transformed variable.

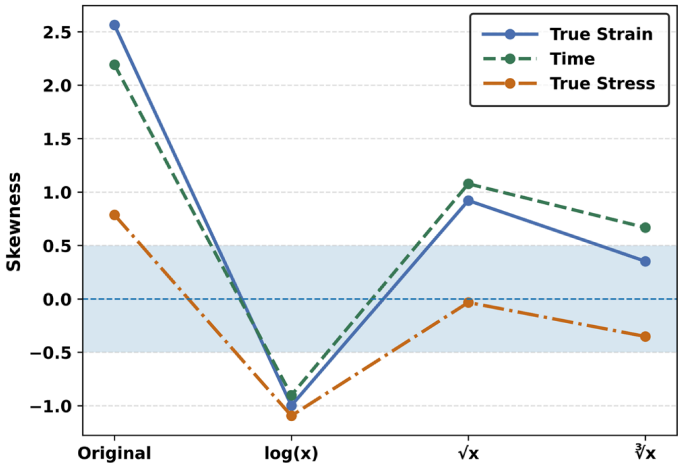


Figure 8. Effect of transformation on the original skewness of true strain, time and true stress. The blue-shaded band between +0.5 and −0.5 skewness indicates the near-symmetric region [39], corresponding to a data distribution close to Gaussian behaviour.

In contrast to the $\sqrt{x}$ transformation, the $\sqrt[3]{x}$ transformation (Figure 7a–c) produces a more balanced reduction across all the investigated variables. In fact, leading both true strain and true stress within the acceptable skewness band, with final skewness values of +0.353 and −0.352, respectively, indicating almost a near-symmetric band distribution. Meanwhile, time skewness reduced to +0.668, representing a significant correction of 70%, although it remains slightly above the ideal Gaussian range. Overall, $\sqrt[3]{x}$ transformation

demonstrates strong skewness reduction without overcorrection. Consequently, it offers the best overall performance in normalising the distributions, with all variables either within or close to the Gaussian symmetry range.

It is quite evident from the earlier discussion that, from a purely Gaussian distributional perspective, the $\sqrt[3]{x}$ transformation emerges as the most consistent approach, as it simultaneously reduces skewness across all variables and aligns them closer to Gaussian-like behaviour. Before proceeding to understand the effect of selcted transformation on the model learning behaviour, it is pertinent to highlight one important consideration. As evident from Figure 8, at a given time, we have applied the same transformation to all variables. However, it is acknowledged that a different transformation to individual variables could have been applied. For instance, at the variable level, for true stress, $\sqrt{x}$ transformation resulted in almost zero skewness (Figure 8). Nonetheless, such variable-specific transformation strategies were not adopted in the present work. Instead, a uniform transformation was selected based on its overall impact on skewness and the model's learning performance. This was, in turn, done to facilitate systematic model comparison and reduce the additional complexity associated with applying such a strategy.

3.2. Transformation Effects on Model Learning

The present section examines the effect of the different transformation strategies on the learning behaviour of the models employed. The effects of the transformation, along with the no transformation (NT) case, are summarised in Figures 9 and 10, respectively. Figure 9 reports the 5-fold CV R^2 values (at the full validation dataset) for each model class, under the different applied transformation. As illustrated in Figure 9, RF and GB, were almost insensitive to the type of transformation applied. In fact, RF and GB maintain nearly constant performance, with R^2 values of 0.82 and 0.86, respectively. In contrast, XGB shows a modest dependence on the applied transformation. For instance, while its performance under the NT case is comparatively lower (R^2 of 0.76) than that of RF and GB, but the application of transformations leads to noticeable improvements, with R^2 increasing to 0.84 for $\sqrt[3]{x}$ transformation.

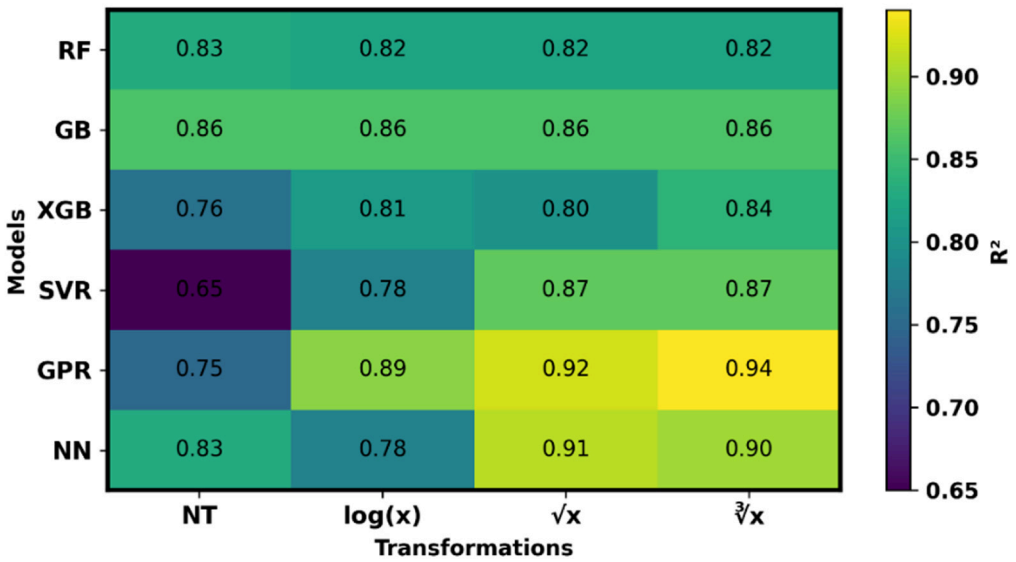


Figure 9. Validation R^2 at full training size for each model (rows) under different target transformations: NT (no transformation), $\log(x)$, $\sqrt{x}$, and $\sqrt[3]{x}$ (columns). The colour scale follows a sequential scheme, where darker blue–purple tones indicating lower validation performance and brighter yellow–green tones indicating higher values.

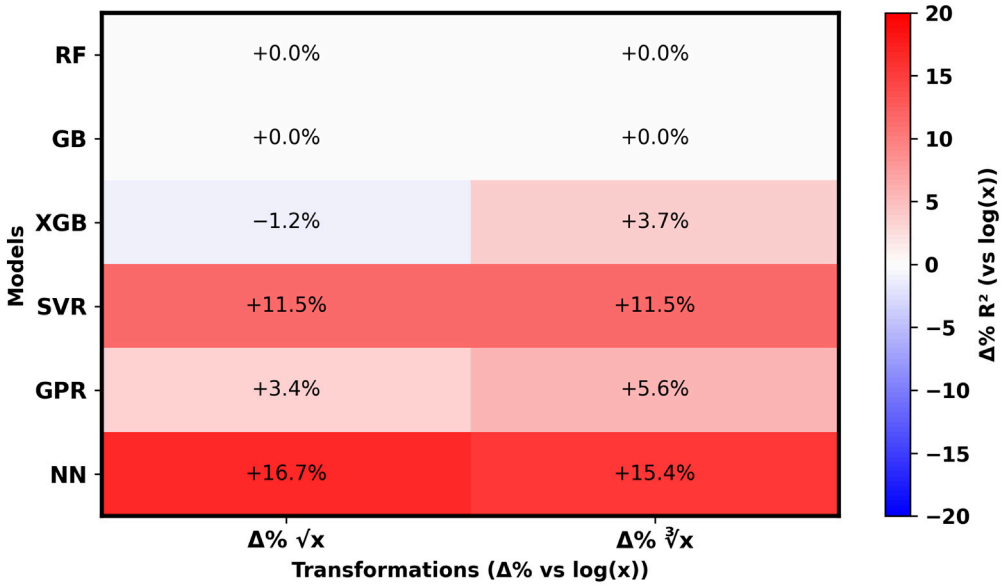


Figure 10. Relative change in validation performance (Δ% R^2) for each model (rows) when using the $\sqrt{x}$ or $\sqrt[3]{x}$ transformations, computed with respect to the $\log(x)$ baseline (columns). The colour scale depicts the percentage gain or loss relative to the one achieved with logarithmic transformation for each class of model. The red tone signifies the increase while the blue tone represents the decrease and white being the neutral.

The general behaviour of tree-based models remains quite stable across transformations, confirming their insensitivity to the applied transformation. This observation aligns with the commentary reported in [23]. In contrast, kernel-based models (SVR and GPR) and the NN are highly sensitive to the chosen transformation. As shown in Figure 9, the performance of SVR improves dramatically from an R^2 of 0.65 in the NT case to 0.85 under both the $\sqrt{x}$ and $\sqrt[3]{x}$ transformations. Furthermore, a slight improvement of 0.13 in R^2 was observed with the $\log(x)$ transformation.

GPR, following similar trends to SVR, shows a sharp rise in R^2 from 0.75 (NT case) to 0.89 with the $\log(x)$ transformation, reaching even higher values for the $\sqrt{x}$ and $\sqrt[3]{x}$ transformations (0.92 and 0.94, respectively). NN does not benefit from all the transformations, achieving an R^2 value of 0.91 under the $\sqrt{x}$ transformation and 0.90 under the $\sqrt[3]{x}$ transformation. This behaviour suggests that, as with kernel-based models, NN also benefit from smoother, less skewed target distributions, which facilitate more stable gradient-based optimisation and improved generalisation.

Evidently, the alternative transformation strategies employed in the present investigation outperform the traditionally used $\log(x)$ transformation. However, to quantify the model performance gains, Figure 10 presents the relative changes in validation performance (Δ% R^2) obtained using the $\sqrt{x}$ and $\sqrt[3]{x}$ transformations, compared with the $\log(x)$ reference baseline. As evident from Figure 10, no change in performance was observed for RF and GB, whereas for XGB, application of $\sqrt{x}$, a slight depreciation (-1.2%) in performance yields an improvement of +3.7%.

Both $\sqrt{x}$ and $\sqrt[3]{x}$ result in identical performance gains of +11.6% for SVR. In the case of GPR, although both transformations enhance performance, the $\sqrt[3]{x}$ transformation produces a larger improvement (+5.6%), indicating a superior effectiveness for this model. Conversely, for the NN model, the $\sqrt{x}$ transformation provides the largest benefit, resulting in a performance increase of 16.7%. Nevertheless, despite these slight model-specific differences, the $\sqrt[3]{x}$ transformation delivers the most consistent performance improvement overall.

Based on the combined findings of Sections 3.1 and 3.2, several important conclusions can be drawn. First, the selection of transformation represents the main objective of the pre-processing stage and a critical step in the machine learning pipeline, apart from tree-based models (RF, GB, and XGB). Therefore, this choice should not be random. Rather, for the given dataset and models employed, the correct choice of transformation should be guided by both the resulting distributional characteristics (Gaussianity) plus the corresponding impact on model learning performance. As discussed earlier, in the present investigation, $\sqrt[3]{x}$ transformation yields near-Gaussian distributions for the examined variables (time, true strain, and true stress) and simultaneously delivers the most consistent learning performance across the majority of models. The results indicate that an appropriate choice of transformation enhances predictive performance. In particular, SVR, GPR, and NN achieved the highest validation accuracy when Gaussian-like transformed variables were used. In contrast, tree-based models (RF, GB, and XGB) showed limited sensitivity to transformation and maintained consistent performance.

3.3. Learning Curves (LC): Generalisation and Model Selection

In this section, the role of LC in identifying the most suitable models for the creep time prediction is investigated. The LC illustrate the evolution of both training and validation performance as a function of training subset size.

They provide key insights into model convergence behaviour (overfitting versus underfitting), generalisation ability (the gap between training and validation performance), and the model stability (fold-to-fold cross-validation performance). Together, these aspects enable an informed and systematic selection of the optimal predictive model [40]. Figure 11a–e illustrates the LC for each model class, where the black and red curves represent the training and validation performance, respectively. The respective shaded region indicates the standard deviation across the five cross-validation folds. Across all models, the training performance remains consistently high, with R^2 values exceeding 0.80 even for the smallest training subset and approaching 1.0 at the full training data subset. This indicates that during the training phase, models can capture the complex underlying relationships in the data.

In contrast, validation performance improves rapidly with increasing training set size, then gradually saturates, particularly beyond 400 samples. This behaviour is much more prominent for kernel-based models. For instance, in SVR and GPR (Figure 11d,e), the validation performance reaches a plateau roughly around 400 samples, indicating that further data addition beyond this threshold will only provide borderline improvement in validation performance. This suggests that these models can extract most of the appropriate underlying information from the available data, rather early during the training phase.

For SVR, the low validation performance at a smaller training size rises to ≈0.87 on the full dataset, with its train–validation gap restricted to ≈0.04, indicating strong generalisation abilities. In addition, GPR achieves good performance, reaching a validation R^2 of ≈0.94 at the largest training size, accompanied by a comparably small generalisation gap (≈0.03). This small gap between the training and validation curves indicates minimal overfitting and strong generalisation behaviour. Significantly, although both SVR and GPR demonstrate strong generalisation abilities, GPR narrowly outperforms SVR, both in terms of high validation performance and low generalisation gap.

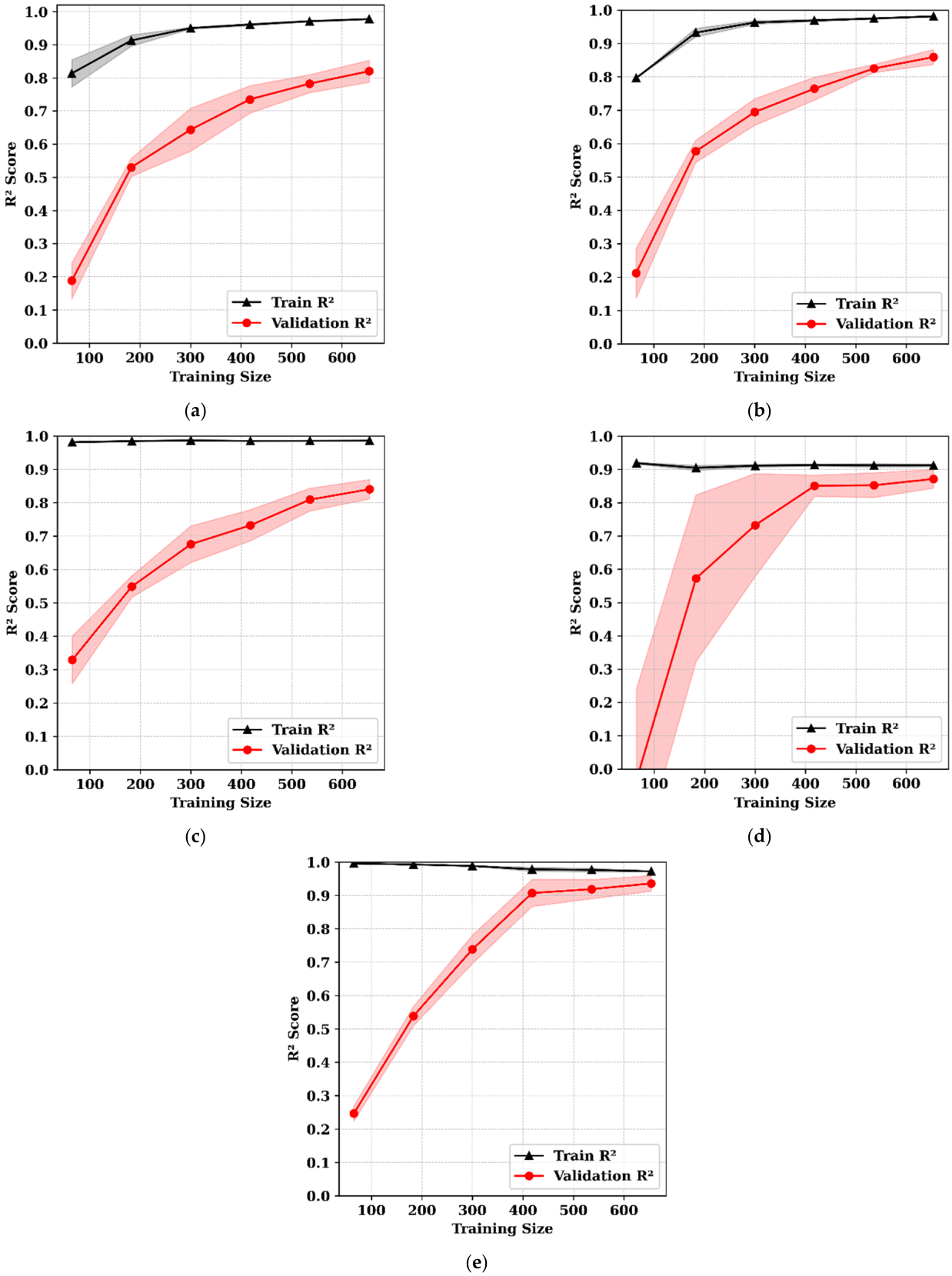
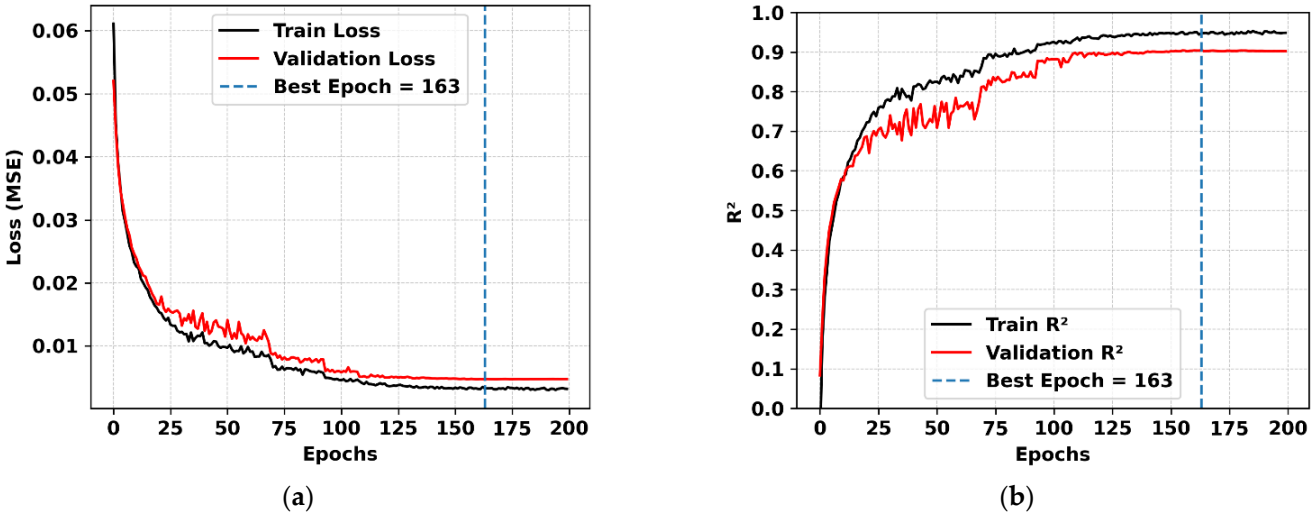


Figure 11. Learning curves for (**a**) RF, (**b**) GB, (**c**) XGB, (**d**) SVR, and (**e**) GPR models, depicting the evolution of training (black) and validation (red) performance as a function of training dataset. All curves are obtained using the selected $\sqrt[3]{x}$ transformation. The red shaded band represents the standard deviation of the validation scores across 5-fold cross-validation.

In contrast, in Figure 11a–c, the tree-based models do not reach a plateau even beyond 600 samples. Their validation performance continues to improve with increased training samples, suggesting that tree-based models can still benefit from additional data, making them data hungry, compared to kernel-based models. Among them, RF reaches a validation R^2 of ≈0.82 at the largest training size, with a generalisation gap of ≈0.16, nearly four to five times higher than observed in SVR (≈0.04) and GPR (≈0.03) respectively. This larger gap reflects the mild overfitting, compared to the minimal overfitting, observed in kernel-based models. Observing the similar trends of RF, GB improves steadily, but to a slightly higher validation performance of ≈0.86, accompanied by a lower gap of ≈0.12, almost 25% lesser than RF, however still higher than kernel-based models, reflecting the mild overfitting again. Lastly, XGB performs comparably, showing a similar gap of ≈0.15, slightly lower than that of RF but higher that of GB. Among the tree-based models, GB, due to its lowest gap (≈0.12), outperforms the counterparts, RF and XGB. Nevertheless, due to the persistent gap between the training and validation curves, tree-based models are not considered reliable and are excluded from the final model selection.

Another key feature of the learning curves is the fold-to-fold stability (shaded bands in Figure 11a–e). RF, GB, and XGB deliver consistent performance across the folds, indicated by the narrow shaded region. This suggests that tree-based models show uniform stability over the folds. In contrast, for SVR, the initial-middle wide fold-to-fold stability progressively narrows, demonstrating that SVR stabilises as more data become available. On the other hand, GPR shows a fine region of fold-to-fold stability, depicting consistent performance across both stability and generalisation.

Figure 12a,b reports the learning curves of the NN. Unlike tree- and kernel-based models, whose performance is evaluated as a function of training set size, NN learning curves are analysed as a function of epochs, where each epoch corresponds to one complete pass over the entire training dataset during the optimisation process [24]. The MSE curve represents the objective function, minimised during training and provides a direct measure of the prediction error magnitude. A decreasing MSE with the increase in epochs indicates refined learning, while divergence between training and validation MSE would suggest overfitting [29].



(**a**) (**b**)

Figure 12. Learning curves for the NN model: (**a**) training (black) and validation (red) R^2, and (**b**) training and validation loss (MSE) as functions of epochs. The dashed vertical line denotes the optimal epoch (163), selected based on validation performance.

As shown in Figure 12a, both training and validation loss curves display a sharp vertical decline within 20 epochs, reflecting fast error reduction as the NN learns the underlying patterns in the data. Furthermore, both training and validation MSE curves remain overlapping throughout the training process, with only a small, stable gap between the two curves. Nevertheless, no apparent deviation is observed even at subsequent epochs (after 150 epochs). The absence of any divergence is a strong indicator of exceptional learning, which is highly stable and has effective generalisation [29], confirming the absence of overfitting. In addition, the best performance is achieved at epoch 163, highlighted by a dashed vertical blue line (Figure 12a,b), indicating the minimum loss (MSE).

Figure 12b further validates the MSE performance by showing the evolution of the R^2 performance. Both training and validation R^2 values increase rapidly in the early epochs, followed by a gradual saturation toward a plateau. At epoch 163, validation performance reached 0.90 with a generalisation gap of ≈0.04, comparable to SVR (≈0.04) and GPR (≈0.03), indicating strong generalisation with minimal overfitting. Broadly, NN depicts a learning performance consistent with the kernel-based models, along with SVR and GPR, is identified as a reliable candidate for the final rupture-time prediction task.

Table 2 summarises the results of the above discussion. The model selection criterion was implemented based on three complimentary aspects: (i) validation performance (R^2), (ii) train–validation R^2 gap (quantitative measurement of overfitting), and (iii) convergence behaviour (plateau reached or not).

Table 2. Summary of model selection criteria. Note: The reported Test R^2 values correspond to the rupture-time prediction task using the test dataset containing both rupture and non-rupture curves.

Models	Validation R^2	Train–Validation Gap	Plateau Reached	Test R^2
RF	0.82	0.16	No	0.50
GB	0.86	0.12	No	0.64
XGB	0.84	0.15	No	0.44
SVR	0.87	0.04	Yes	0.90
GPR	0.94	0.03	Yes	0.87
NN	0.90	0.04	Yes	0.92

As evident from Table 2, among the tree-based models, GB emerged as the most competitive candidate, depicting a validation performance ($R^2 = 0.86$), which is comparable to that of SVR ($R^2 = 0.87$). However, GB does show a substantially larger train–validation gap of 0.12 compared to 0.04 of SVR. Furthermore, unlike SVR, GPR, and NN, the validation performance of GP did not reach a clear plateau within the investigated training size range, suggesting that additional training data may still improve its performance. Similar observations were made for RF and XGB. Hence, considering all three criteria, SVR, GPR, and NN emerged as the most suitable candidates and therefore were selected for the final prediction task.

It is pertinent to note that this model selection strategy is further supported by the performance on the independent test dataset. As shown in Table 2, all three tree-based models (RF, GB, and XGB) depicted lower test performance when applied to rupture-time prediction using the dataset containing both rupture and non-rupture curves. This observation independently validates the conclusions drawn from the learning curve analysis and reinforces the robustness of the proposed methodology of the present work. The test performance of the selected SVR, GPR, and NN models is discussed in detail in Section 3.5.

3.4. Hyperparameter Optimisation

The hyperparameters tuned by Optuna for each model class are summarised in Table 3. The search space for parameter tuning was guided by the recommendations from [35,36].

It is pertinent to note that the optimisation results illustrated in Table 3 define only the internal structure and learning behaviour of the respective models. These hyperparameters do not contain information about, nor can they be used to reproduce, the confidential creep rupture dataset employed in this investigation. A detailed explanation of the tuned hyperparameters and their functional roles is provided in Appendices A and B.

Table 3. Selected and tuned key hyperparameter values of the selected model, optimised for time predictions using the Optuna framework.

Model	Hyperparameters	Search Range	Tuned Values
RF	n_estimators	100–500	475
	max_dept	5–15	15
	min_samples_split	2–20	3
	min_samples_leaf	2–10	2
	max_features	sqrt, log2, None	sqrt
	bootstrap	True, False	False
GB	n_estimators	100–250	236
	learning_rate	0.001–0.1 (log scale)	0.083
	max_dept	4–10	8
	min_samples_split	10–20	20
	min_samples_leaf	10–20	10
	subsample	0.1–0.9	0.85
	max_features	sqrt, log2, None	log2
XGB	n_estimators	50–300	187
	learning_rate	0.001–0.1 (log scale)	0.061
	max_dept	4–10	7
	min_child_weight	1–10	3
	gamma	0–5	0.0005
	subsample	0.2–0.8	0.73
	colsample_bytree	0.5–1.0	0.88
	L1(reg_alpha)	1×10^{-6}–10	0.0437
	L2(reg_lambda)	1×10^{-6}–10	0.0102
SVR	C	1×10^{-2}–300 (log scale)	298.78
	epsilon	1×10^{-2}–10	0.0168
	kernel	rbf, sigmoid	rbf
GPR	kernel	rbf, Matérn	Matérn
	constant_value	0.1–6 (log scale)	0.510
	length_scale	0.01–6 (log scale)	0.109
	alpha	1×10^{-5}–1×10^{-1} (log scale)	0.031
NN	units_1	32,512	352
	units_2	32,512	416
	dropout	0–0.5	0.0165
	L2_reg	1×10^{-6}, 0.01 (log scale)	1.318×10^{-6}
	learning_rate	1×10^{-4}, 1×10^{-2} (log scale)	0.00154
	optimizer_name	adam, rmsprop, nadam	nadam
	activation function	not tuned	ReLU
	batch size	16, 32, 64	16

In the present work, the optimal RF configuration selected 475 estimators, suggesting that a relatively large ensemble size is required. The maximum tree depth reached the upper limit of the search space (15), suggesting that deeper trees were needed to capture the complex and highly nonlinear relationships among stress, temperature, and strain. Relatively small values of min_samples_split = 3 and min_samples_leaf = 2 allowed fine-grained partitioning of the feature space, enabling the model to learn localised patterns

while keeping a minimum number of samples per node. Also, the optimised configuration set bootstrap = False, meaning that each tree was trained on the full dataset rather than on bootstrap-resampled subsets. This choice ensures that all available data contribute to training each tree, which can be beneficial in small-data regimes. In this case, diversity within the ensemble was primarily maintained through random feature subsampling max_features = "sqrt", which introduces variability in the split selection process [36].

For the GB model, the number of boosting steps converged to 236, almost half that of those in the RF (475). Unlike RF, which counts on a large ensemble of independent trees, GB builds trees successively, with each new tree focusing on correcting the residual errors of the previous ensemble [36]. The fact that GB converges with fewer estimators suggests a more efficient learning process and a compact model structure, which is further confirmed by its maximum depth of 8 (lower than that of RF). The learning_rate was optimised to a relatively low value of 0.083. A smaller learning rate allows the model to make gradual updates to the residuals, thereby reducing the risk of overfitting.

On the other hand, parameters such as min_samples_split = 20 and min_samples_leaf = 10, both higher than those observed for the RF, further reflect the formation of less deep-branched trees. As a result, the GB model prefers simpler, better generalised tree structures that reduce the tendency to overfit local changes in the training data. Also apparent from the low generalisation gap (≈0.12), in fact, the lowest among all the tree-based models. The subsample, optimised to 0.85, shows that the model benefited from a randomly selected portion of the data, where each tree is trained on roughly 85% of the available data. Finally, the tuned max_features = log2 suggests that the model achieved the best performance when a smaller subset of features was considered at each split.

For the XGB model, the optimised number of estimators (187), learning rate (0.061), and maximum depth (7) indicate a level of model complexity comparable to that of the GB model, with both relying on a gradual, stage-wise learning process. However, unlike GB, XGB incorporates two more regularisation terms to control model complexity. These are L1 (reg_alpha = 0.0437) and L2 (reg_lambda = 0.0102), which both add the penalties, making the model less complex. Like gamma, min_child_weight also restricts the number of splits at each tree [36]. Both parameters contribute to improved generalisation on small, heterogeneous datasets such as the present creep dataset. The subsample (0.73) and colsample_bytree (0.88) parameters were optimised to values below 1, indicating that each tree was trained using only a subset of samples and features.

In the case of kernel-based models (SVR and GPR), the relatively high value of the regularisation parameter C = 298.78 indicates that the SVR model assigns a strong penalty to training errors, thereby favouring a close fit to the data. Along with C, the parameter epsilon = 0.0168 represents the width of the insensitive zone around the regression function, within which prediction errors are not penalised [35]. The small value of epsilon implies a narrow tolerance region, forcing the model to penalise even for small deviations. The combination of a high C and a low epsilon suggests that the model is configured to achieve high precision, while still maintaining controlled regularisation. This balance promotes accurate fitting without inducing overfitting, as confirmed by strong validation performance (R^2 = 0.87) and a low generalisation gap. Another key component of kernel-based models is the choice of kernel function. In the present work, the SVR kernel was tuned to the radial basis function (rbf), which is well-suited for capturing nonlinear relationships [35].

In contrast to the SVR kernel, the GPR model was tuned to a Matérn kernel, which defines the shape of the fitted function and is known to be less smooth than the rbf kernel [41]. This property makes the Matérn kernel more natural for physical phenomena such as creep. Therefore, selecting the Matérn kernel is a physically significant option for the present situation. The parameter constant_value, which controls the overall ver-

tical scale of the predicted function, was tuned to a moderate value of 0.510 [41]. This indicates that the GPR model allows moderate output fluctuations, suggesting that the underlying relationship is neither too oscillatory nor too flat, a behaviour that is consistent with typical creep data. The optimised length_scale of 0.109 further indicates that the model's predictions vary moderately with slight changes in the input features [41]. In fact, it allows the capture of localised nonlinear trends without introducing excessive complexity. The parameter alpha was tuned to 0.031, corresponding to a low noise level added to the kernel matrix [41], thereby preventing overfitting and enhancing the model's generalisation capability.

Finally, for the NN model, the activation function, which determines how the weighted input at each neuron is converted into an output signal, introducing nonlinearity into the network [36], has not been tuned. In fact, the Rectified Linear Unit (ReLU) was adopted, as it is the most widely used activation function in creep-domain analysis. Nevertheless, all remaining major hyperparameters were tuned. Specifically, the number of neurons in the two hidden layers was set to 352 and 416, respectively.

The dropout rate, which serves as a regularisation mechanism, randomly deactivates a fraction of neurons. It was optimised to a very small value (≈1.60%). This indicates that strong regularisation was not required for the present NN configuration. This behaviour can be attributed to the limited network depth and the small learning rate (0.00154). In addition, constrains the magnitude of weight updates and encourages stable convergence. Moreover, the optimised L2_reg (regularisation coefficient) of 1.318×10^{-6} provides an extra, yet mild penalty on large weights, further supporting the idea that only weak regularisation was necessary. Among the tested optimisers (Adam, RMSprop, and Nadam), the Nadam optimiser emerged as the most effective. Lastly, the batch size, which represents the subset of the entire dataset used to determine the gradients for the network weights, was tuned to a lower value of 16. In general, a low batch size value leads to slightly higher run time but can result in better learning performance [36].

3.5. Creep Time Predictions: Rupture-Time Results

In the last phase (Phase III) of the machine learning pipeline, the best-performing models, identified from the learning curve analysis, which include SVR, GPR and NN were employed to predict the creep time on the test dataset. As anticipated before, creep time includes both the time to reach specific strain levels and time to rupture. However, at this stage, only the results for the rupture time are presented. Furthermore, including creep curves for both rupture and non-rupture type is an important highlight of the present investigation. Therefore, the effect of including or excluding the non-rupture creep curves on the mode's performance in predicting the rupture time is also investigated.

Figure 13 illustrates this comparison: Column I corresponds to the full dataset containing both types of curves; on the other hand, Column II presents the case of only rupture curves in the dataset. For excluding the non-rupture curves from the full dataset, the is_rupture flag was used, and all the corresponding non-rupture creep curves were removed from the entire dataset. For each scenario, the predicted rupture times are compared with the corresponding original values using log–log plots, normalised by the maximum rupture time. Since the predictions were compared on the basis of original time values, the applied preprocessing steps of Phase I, such as Min–Max scaling and applied $\sqrt[3]{x}$ transformation were removed, restoring the time (target variable) to the original physical units (hours).

I: Rupture and Non-Rupture curves included in dataset

II: Only Rupture Curves included in the dataset

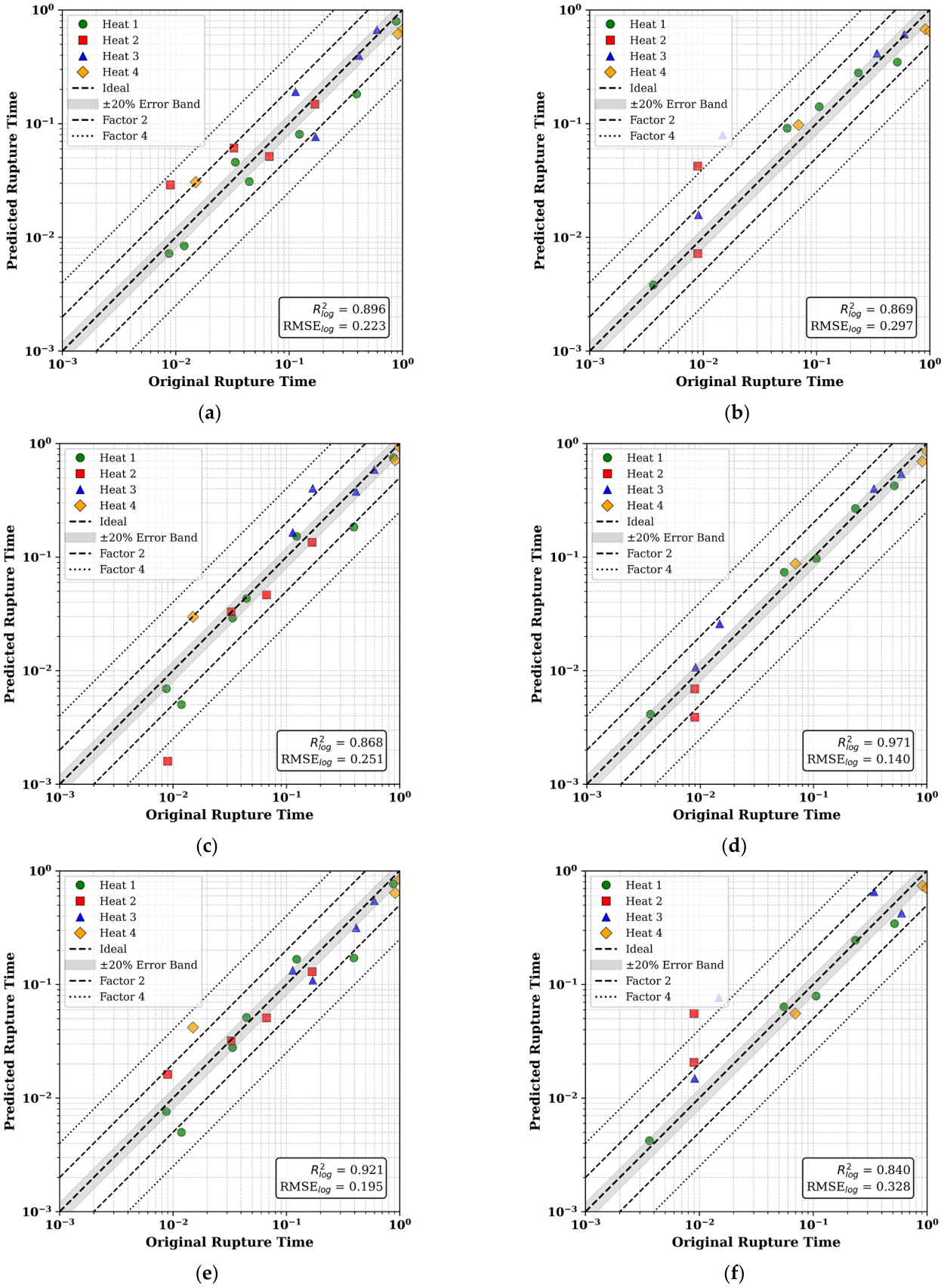


Figure 13. Predicted versus original rupture time using log–log plots. For confidentiality reasons, both original and predicted rupture times were divided by the maximum rupture time in each figure.

Column I represents the case where both rupture and non-rupture curves are included, while Column II corresponds to the case with only rupture curves. Data points are colour-coded according to the four material heats considered in the present investigation. The dashed black line represents the ideal case of perfect prediction, while the grey shaded region denotes the ±20% error band. Factor-of-two and factor-of-four reference lines are also included. Global model-level performance metrics (R^2 and RMSE), computed in the logarithmic domain, are reported in each subfigure. (**a**) SVR. (**b**) SVR. (**c**) GPR. (**d**) GPR. (**e**) NN. (**f**) NN.

The model performance is examined using the R^2 and RMSE, both computed in the logarithmic domain. Importantly, both R^2 and RMSE are computed at the global level, considering all heats together to assess the overall predictive model performances, as well as at the per-heat level to examine the model predictive performance specific to individual heats. The results of per-heat analysis (performed only for the case of the full dataset, Column I) are summarised in Table 4. It is worth noting that only RMSE values are reported for the per-heat analysis, as the R^2 metrics become less stable and reliable when computed on very small sample sizes.

Table 4. Per-heat rupture-time prediction performance (RMSE in the logarithmic domain) for each selected model in the case of the full dataset (rupture and non-rupture curves combined).

Models	SVR	GPR	NN
Heat 1	0.172	0.193	0.209
Heat 2	0.292	0.357	0.152
Heat 3	0.198	0.189	0.120
Heat 4	0.206	0.206	0.277

The results of Column I are discussed first. As evident from Figure 13e, the NN exhibits the highest predictive accuracy of (R^2 = 0.92) along with the lowest RMSE of 0.195, indicating good agreement between the predicted and original rupture times. The majority of NN predictions closely followed the ideal trend line, with nearly all data points lying within the ±20% error band. Furthermore, the inclusion of factor 2 and factor 4 lines further highlights the robustness of the model, as almost all data points remain within the factor 2 bound. Only one data point each from Heat 1 (green coded) and Heat 4 (yellow coded) slightly exceeds the factor 2 limit but still falls well within the factor 4 bounds. On a global level, the NN predictions deviates from the experimental rupture times by an average factor of ≈1.57 ($\approx 10^{RMSE}$), confirming the high reliability of NN model for creep rupture time predictions. Sakurai et al. [20] reported a slightly lower RMSE of 0.14, corresponding to an average factor of 1.38, for commercial creep-resistant steels. The slightly higher prediction error observed in the present study can be attributed to the limited dataset and the absence of additional features, such as microstructural information.

From an engineering perspective, the obtained prediction errors (RMSE of 0.195 and deviation factor of 1.57) indicate that the NN can provide reasonable estimates of creep rupture life. Practically, the predictions on average will be within a factor of ≈1.57 of the experimental creep rupture time, demonstrating the potential applicability of the present NN for preliminary creep life assessment.

Compared to the NN model, SVR (Figure 13a) exhibits a slightly lower predictive accuracy, with an R^2 of 0.90 and a higher RMSE of 0.223. Except for the single data point corresponding to Heat 2 (coded in red), all the predictions lie well within the factor 2 bounds. Furthermore, on a global scale, SVR predictions deviate from the original rupture time, by an average factor of ≈ 1.64, which is only marginally higher than that observed for the NN model. On the other hand, the GPR model (Figure 13c) exhibits the lowest predictive performance among the tested approaches, with an R^2 value of 0.87 and the

highest RMSE of 0.251. At the global level, this corresponds to an average deviation factor of ≈1.78, which is the largest among all considered models. Similar to the SVR results, Heat 2 (red-coded) shows the most pronounced deviations, with one prediction falling outside the factor 4 bounds, while the remaining data points are largely confined within the factor 2 region.

Continuing with the discussion on results presented in Column I. Table 4 presents the per-heat RMSE values. Overall, the NN model demonstrates the most consistent behaviour, achieving the lowest RMSE values for Heat 2 (0.152) and Heat 3 (0.120), indicating strong predictive capability across varying data subsets. In contrast, SVR and GPR exhibit higher variability, particularly for Heat 2, where RMSE increases to 0.292 and 0.357, respectively, confirming the larger deviations observed in the corresponding prediction plots.

The observed differences can mainly be attributed to the non-uniform distribution of creep curves across the heats. For instance, Heat 1, owing to the highest number of creep curves (44), demonstrates relatively low RMSE values across all models. Respectively, Heat 3, with 21 curves, also shows good predictive performance, particularly for the NN model. On the other hand, Heat 2 and Heat 4 have fewer curves, leading to decreased model robustness. For Heat 4, despite the limited data availability, all models maintain comparable RMSE values, suggesting reasonable generalisation within this subset. However, the consistently higher errors for Heat 2 across all models highlight the combined effects of greater variability in creep behaviour for this heat. Taken together, the per-heat analysis supports the superior robustness of the NN model, followed by SVR, while GPR shows comparatively weaker generalisation, particularly for heats with limited data. This further underscores the importance of balanced data distribution for reliable rupture-time predictions.

In the column II of Figure 13, Figure 13b,d,f illustrate the effect of the removal of non-rupture creep curves from the full dataset on the predictive performance for each model. As evident from the comparison between Figure 13a,b, the SVR experienced only a slight reduction in model performance. The R^2 decreases from 0.896 to 0.869 (≈3% reduction), while the RMSE increases from 0.223 to 0.297, corresponding to a moderate increase of 33%. The relatively robust performance of SVR against the reduction in the dataset can be attributed to the underlying working mechanism of SVR, which relies on support vectors, a subset of data points of the entire dataset [42].

On the other hand, NN (Figure 13f) suffered from a major impact of the non-rupture creep curves removal, evident by the low R^2 of 0.84 (reduction of ≈9%) and RMSE of 0.328 (sharp increase of ≈68%), depicting the substantial loss in predictive accuracy. The reason for this behaviour of NN lies in its structure, which typically requires large and diverse datasets, making it data hungry by nature [43,44]. In fact, the removal of non-rupture creep curves resulted in the approximate loss of 35% dataset.

Unlike the SVR and NN, the GPR (Figure 13c) benefited largely from the exclusion of non-rupture creep curves, evident by the sharp rise in R^2 from 0.87 to 0.97 (increase of ≈11%) and decline in RMSE from 0.251 to 0.140, leading to a substantial decrease of 44%. One possible explanation for this behaviour can be attributed to the reduction in data heterogeneity following the removal of non-rupture creep curves, which, in turn, might have limited the performance of the Matérn kernel (Section 3.4 and Table 3), hypertuned by Optuna. In general, Matérn kernels provide flexibility in mapping relationships with varying smoothness. However, single kernel performance could be affected under the heterogeneous dataset [45].

This interpretation is further partly supported by the comparison between the validation and held-out test performance of 0.94 (Figure 9) and 0.87 (Figure 13c), respectively, computed on the full dataset. In contrast to this behaviour, SVR and NN maintained

consistency from validation to held-out test conditions. Although this observation may suggest that GPR is comparatively more sensitive to the changes in the dataset composition. However, additional experimental investigation, including dedicated kernel comparisons and ablation studies, would be required to verify the hypothesis and the underlying causes of such GPR behaviour.

Overall, the results indicate that the influence of non-rupture curve on rupture-time prediction is model dependent and is not uniform across the machine learning approaches considered in the present work.

4. Conclusions

This study developed an AI-based framework for predicting creep time, referring to the time required to reach specific strain levels and the time to rupture. However, the results for time to rupture predictions are presented only at this stage. Furthermore, the framework is limited to four heats of austenitic stainless steel, grade 316, within specific stress–temperature regimes, thereby restricting its direct applicability to other alloys or broader service conditions.

- The cube-root transformations resulted in higher validation performance for SVR, GPR, and NN.
- The tree-based models (RF, GB, and XGB) were insensitive to the choice of transformation.
- Learning curve analysis revealed mild overfitting for the tree-based models, while SVR, GPR, and NN demonstrated minimal overfitting.
- During time to rupture prediction, the NN model achieved the highest overall predictive accuracy (R^2 = 0.92), followed by SVR (0.90) and GPR (0.87).
- The per-heat evaluation (individual heat performance) revealed that SVR provided the most accurate predictions for Heats 1 and 4, and NN performed best for Heats 2 and 3.
- Importantly, the effect of incorporating the non-rupture creep curves on the respective model performance in predicting the rupture time was not uniform across the models. It depends mainly on the choice of model, like SVR and NN, which benefited from their incorporation, while GPR performance was hindered.

Author Contributions: Conceptualisation, A.N., A.T. and E.G.; methodology, A.N. and E.G.; software, A.N.; validation, A.N., A.T. and E.G.; data curation, A.N.; writing—original draft preparation, A.N.; writing—review and editing, A.N., A.T. and E.G.; supervision, E.G.; funding acquisition, A.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was financially supported by the agreement between the Istituto Nazionale per l'Assicurazione contro gli Infortuni sul Lavoro (INAIL) and Politecnico di Milano for the funding of doctoral research activities.

Data Availability Statement: The data used in this study are confidential and therefore cannot be made available.

Acknowledgments: The authors acknowledge the European Creep Collaborative Committee (ECCC) for supplying the creep curve data used to develop and evaluate the performance of the diverse AI-based models shown in this work. The authors also sincerely thank INAIL (Istituto Nazionale per l'Assicurazione contro gli Infortuni sul Lavoro), Italy, for its financial support.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
ML	Machine Learning
HPO	Hyperparameter Optimisation
CV	Cross-Validation
LC	Learning Curves
RF	Random Forest
GB	Gradient Boosting
XGB	Extreme Gradient Boosting
SVR	Support Vector Regressor
GPR	Gaussian Process Regressor
NN	Neural Network

Appendix A. Models

Appendix A.1. Random Forest

Random Forest (RF), introduced by [46], is an ensemble learning method that combines multiple decision trees (DT) [29]. While a single DT (Figure A1a) is easy to interpret, it can often suffer from severe overfitting, limiting its ability to generalise to unseen data [47]. RF addresses this limitation by constructing a forest of trees (number of independent trees) on random subsets of both the training data and the feature space, thereby introducing diversity among trees [48]. Figure A1b illustrates the schematic of RF, where bootstrapping (sampling with replacement) is used to generate multiple sub-datasets, and each tree is trained in parallel. The "random" aspect arises from both random sampling of data and random selection of features at each split, ensuring diversity among trees. The final prediction is obtained by averaging the outputs of all trees in regression tasks using bagging (bootstrap aggregating). This ensemble approach reduces variance, improves robustness, and allows RF to handle large datasets effectively [48].

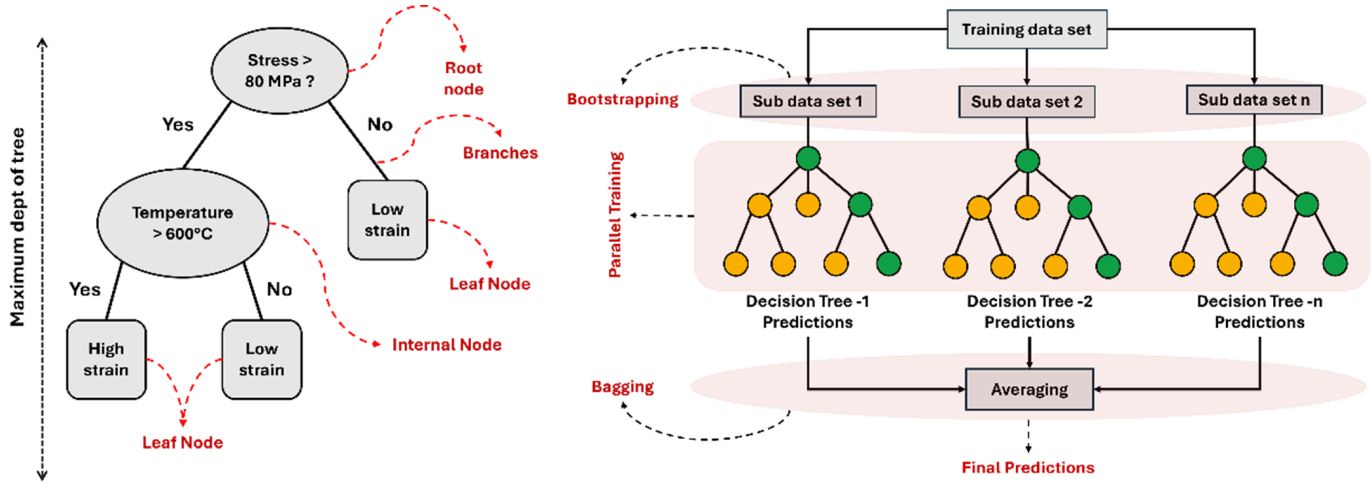


Figure A1. (**a**) Schematic representation of a single decision tree (DT); (**b**) Random Forest (RF).

Appendix A.2. Gradient Boosting

Gradient Boosting (GB) [49] is a boosting ensemble method that combines multiple weak learners to build a strong predictive model. These weak learners are typically shallow decision trees trained (not deep) sequentially, where each new tree aims to fix the errors of

the previous tree. Figure A2 illustrates the schematic of GB. The process begins with a first decision tree fitted to the training data; the residuals (differences between observed and predicted values) then become the target for the next tree. This iterative procedure continues until the maximum number of trees (boosting steps) is reached. The final prediction is obtained as an additive combination of all trees, scaled by a learning rate that controls the contribution of lowering the weights of each tree and thereby reducing overfitting [36,50].

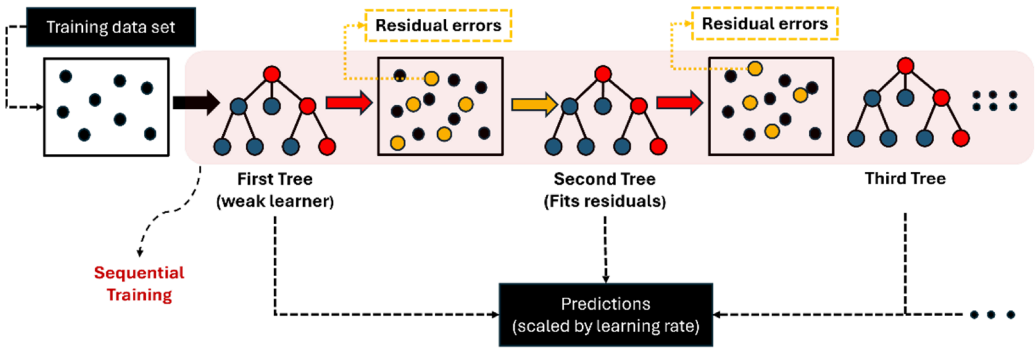


Figure A2. Schematic representation of the Gradient Boosting process.

Appendix A.3. Extreme Gradient Boosting

Extreme Gradient Boosting (XGB) [51] builds on the concepts of the Gradient Boosting (GB) framework. However, with further improvements in speed, efficiency, and generalisation. Unlike standard GB, XGB incorporates advanced features such as built-in regularisation (L1 and L2) to reduce overfitting.

Appendix A.4. Support Vector Regressor

Support Vector Regression (SVR) [42] uses the concept of Support Vector Machines (SVM) and extends it to the regression tasks. Figure A3 illustrates a schematic of the 1D SVR model, where the line (or hyperplane in a multidimensional space) represents the predicted function.

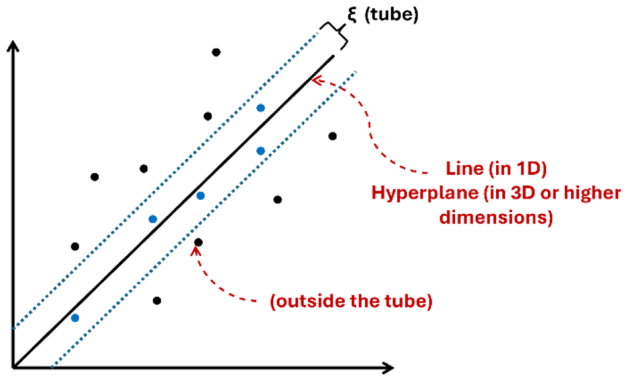


Figure A3. A schematic representation of the one-dimensional Support Vector Regression (SVR) model, where only the data points lying outside the ε-tube (in black) are considered for determining the predictions (adapted from [52]).

SVR describes a tolerance region, known as the ε-tube (epsilon), where small prediction errors are considered acceptable and do not significantly influence the model. In fact, instead of trying to predict every data point, SVR uses the data points outside the tube (black data points in Figure A3) to build the model. This prevents the SVR from overfitting to small noise and deviations within the dataset. Moreover, enabling it to focus on the critical data points, which are far from the main line or hyperplane [52]. Apart from

the parameter ε, the kernel and cost (C) parameters play an important role in defining the performance of SVR. Kernels transform the input data into a higher-dimensional feature space, where the relationship becomes approximately linear, even if it appears highly nonlinear in the original space. Some of the common kernel types include linear, polynomial, radial basis function (RBF), and sigmoid. Parameters such as cost parameter (C), L1, and L2 act as regularisation terms, controlling the balance between the required model complexity and the degree to which deviations larger than ϵ are tolerated [36].

Appendix A.5. Gaussian Process Regressor

Gaussian Process Regression (GPR) [53] is a non-parametric model, meaning it does not assume any predefined function between the input (feature) and output (target) variables. Instead, GPR assumes that there could be an infinite number of smooth functions that might explain or fit the data. Each of these possible functions is assigned a probability based on how well it matches the experimental data. In fact, to predict new conditions, GPR assesses how similar the new point is to the points in the training data. This similarity is measured using a kernel or covariance function, which makes the predictions smooth and consistent, so that similar inputs yield similar outputs [53].

Appendix A.6. Neural Network (NN)

Neural networks (NN) are motivated by the structure of biological neurons [54–56]. A single artificial neuron takes multiple inputs, applies weights and a bias, and passes the weighted sum through an activation function to generate an output. NN are formed by interconnecting many such neurons into layers, generally comprising an input layer, hidden layers, and an output layer. Figure A4 illustrates the schematic of the NN used in the present investigation, where stress, temperature, strain, heat, and curve type (rupture or non-rupture) serve as input features. During the feed-forward phase, as input signals pass through the network, predictions are made at the output. However, during the training phase, a phenomenon known as backpropagation is activated. This iteratively adjusts the weights and biases, minimising error and improving model accuracy over time [56,57].

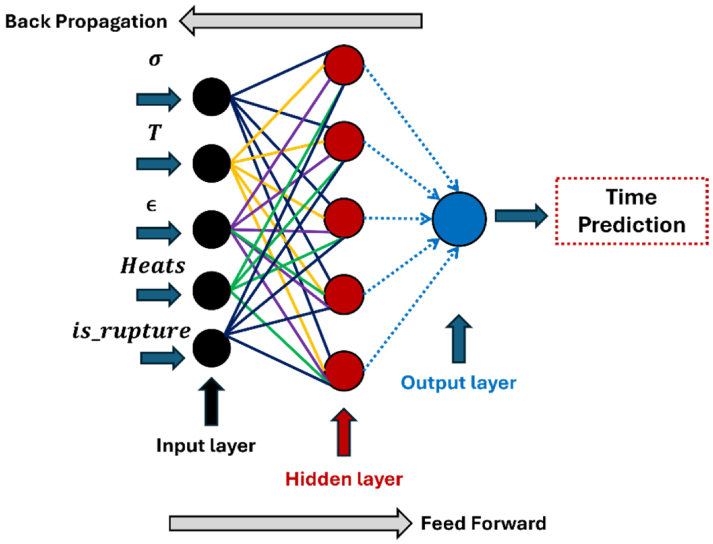


Figure A4. Schematic of a neural network (NN) for time prediction used in the present investigation, showing input features (σ, T, ϵ, *heats*, *is_rupture*), hidden layers, and the output layer.

Appendix B. Description of Model Hyperparameters

Model	Hyperparameters	Description
RF	n_estimators	Total number of independent trees in the ensemble model
	max_dept	Represents the depth of the tree from the root node (first) to the leaf node (terminal node), controlling the maximum number of splits along any path
	min_samples_split	Minimum number of samples required to split an internal node
	min_samples_leaf	Minimum number of samples required at a leaf node (end node of the tree)
	max_features	Number of randomly selected and considered for finding the best split (the choice is made at each node)
	bootstrap	Determined whether sampling of data is done with replacement (True) or without replacement (False) for each tree
GB/XGB	n_estimators	Number of boosting steps (like number of trees in RF)
	learning_rate	Shrinkage step which controls the lowering of the weights in each boosting step
	max_dept	Same as that for RF
	min_child_weight	Like max_dept and gamma (restricts the number of splits of each tree)
	gamma (XGB)	Minimum loss reduction required to make a split (regularisation parameter)
	subsample	Fraction of data used for training at each boosting step
	colsample_bytree	Fraction of features (the choice is made once for each tree)
	reg_lambda (XGB)	L1 regularisation coefficient (controls the strength of penalty)
	reg_alpha (XGB)	L2 regularisation coefficient (controls the strength of penalty)
SVR	kernel	Real valued symmetrical function (linear, polynomial, radial basis and sigmoid are some of the common kernel functions)
	epsilon	Width of the margin (ε-tube) around the regression line within which errors are ignored, controlling models' sensitivity
	C	Regularisation parameter that controls model complexity by weighting the penalty for constraint violations
GPR	kernel	Function defining similarity points (radial basis, Matérn)
	constant_value	Scaling factor controlling the overall signal magnitude of the Gaussian process
	length_scale	Determines how smooth or flexible the fitted function is, smaller value fits the data more closely
	alpha	Noise variance term added to the kernel matrix for numerical stability and regularisation, preventing overfitting
NN	units_1 & units_2	Number of neurons in hidden layer 1 and hidden layer 2
	activation	Determines how the input to each node is mathematically transformed into its output for each layer, governing the nonlinear behaviour of the network
	dropout	Randomly deactivates a fraction of neurons during training to prevent overfitting (regularisation parameter of NN)
	L2_reg	L2 regularisation coefficient (controls the strength of penalty)
	learning_rate	Controls the size of weight updates during training, determining how quickly or slowly a neural network learns from errors
	optimizer_name	Algorithm that updates the network weights to minimise loss (Nadam, Adam)

References

1. Bhardwaj, H.K.; Shukla, M. A unified creep and fatigue life prediction approach for 316 austenitic stainless steel using machine and deep learning. *Fatigue Fract. Eng. Mater. Struct.* **2024**, *47*, 3444–3463. [CrossRef]
2. Bhardwaj, H.K.; Shukla, M. Machine Learning-Based Improved Creep Life Prediction of 316 Austenitic Stainless Steel with Add-on Chemical and Microstructural Features. *J. Mater. Eng. Perform.* **2025**, *34*, 18978–18996. [CrossRef]
3. Lee, Y.S.; Kim, D.W.; Lee, D.Y.; Ryu, W.S. Effect of grain size on creep properties of type 316LN stainless steel. *Met. Mater. Int.* **2001**, *7*, 107–114. [CrossRef]
4. Narula, P.; Kumar, P.A.; Vanaja, J.; Reddy, G.V.P.; Rao, G.V.S.N. Machine learning assisted prediction of creep data of India specific reduced activation ferritic martensitic steel. *Mater. Today Commun.* **2023**, *35*, 106165. [CrossRef]
5. Holdsworth, S. Creep-ductility of high temperature steels: A review. *Metals* **2019**, *9*, 342. [CrossRef]
6. Baraldi, D.; Holmström, S.; Nilsson, K.-F.; Bruchhausen, M.; Simonovski, I. 316L(N) Creep Modeling with Phenomenological Approach and Artificial Intelligence Based Methods. *Metals* **2021**, *11*, 698. [CrossRef]
7. Larson, F.R.; Miller, J. A Time-Temperature Relationship for Rupture and Creep Stresses. *Trans. ASME* **1952**, *74*, 765–775. [CrossRef]
8. Manson, S.S.; Haferd, A.M. *A Linear Time-Temperature Relation for Extrapolation of Creep and Stress-Rupture Data*; NACA: Boston, MA, USA, 1953.
9. Orr, R.L.; Sherby, J.E.; Dorn, O.D. *Correlations of Rupture Data for Metals at Elevated Temperatures*; American Society of Mechanical Engineers (ASME): New York, NY, USA, 1953.
10. Monkman, F.C.; Grant, N.J. An Empirical Relationship Between Rupture Life and Minimum Creep Rate in Creep-Rupture Tests. In *Proc of the ASTM*; ASTM International: West Conshohocken, PA, USA, 1956; Volume 56, pp. 593–620.
11. Evans, R.W.; Wilshire, B. *Creep of Metals and Alloys*; The Institute of Metals: London, UK, 1985.
12. Bagley, R.L.; Jones, D.I.G.; Freed, A.D. Renewal creep theory. *Metall. Mater. Trans. A* **1995**, *26*, 829–843. [CrossRef]
13. Xiang, S.; Chen, X.; Fan, Z.; Chen, T.; Lian, X. A deep learning-aided prediction approach for creep rupture time of Fe–Cr–Ni heat-resistant alloys by integrating textual and visual features. *J. Mater. Res. Technol.* **2022**, *18*, 268–281. [CrossRef]
14. Zhou, C.L.; Yuan, R.H.; Liao, W.J.; Yuan, T.H.; Fan, J.K.; Tang, B.; Zhang, P.X.; Li, J.S.; Lookman, T. Creep rupture life predictions for Ni-based single crystal superalloys with automated machine learning. *Rare Met.* **2024**, *43*, 2884–2890. [CrossRef]
15. Chai, M.; He, Y.; Li, Y.; Song, Y.; Zhang, Z.; Duan, Q. Machine Learning-Based Framework for Predicting Creep Rupture Life of Modified 9Cr-1Mo Steel. *Appl. Sci.* **2023**, *13*, 4972. [CrossRef]
16. Wei, L.; Wang, S.; Hao, W.; Huang, J.; Qu, N.; Liu, Y.; Zhu, J. Prediction of High-Temperature Creep Life of Austenitic Heat-Resistant Steels Based on Data Fusion. *Metals* **2023**, *13*, 1630. [CrossRef]
17. Zhang, S.; Wang, L.; Zhu, S.P.; Deng, X.; Fu, S.; Luo, C.; Dong, Y.; Yan, D. Physics-informed neural network for creep-fatigue life prediction of Inconel 617 and interpretation of influencing factors. *Mater. Des.* **2024**, *245*, 113267. [CrossRef]
18. Qin, Q.; Zhang, Z.; Long, H.; Zhuo, J.; Li, Y. Prediction of creep properties of Co–10Al–9W superalloys with machine learning. *J. Mater. Sci.* **2024**, *59*, 4571–4585. [CrossRef]
19. Zhang, X.; Yao, J.; Wu, Y.; Liu, X.; Wang, C.; Liu, H. A Method for Predicting the Creep Rupture Life of Small-Sample Materials Based on Parametric Models and Machine Learning Models. *Materials* **2023**, *16*, 6804. [CrossRef] [PubMed]
20. Sakurai, J.; Demura, M.; Inoue, J.; Yamazaki, M. Creep Life Predictions by Machine Learning Methods for Ferritic Heat Resistant Steels. *ISIJ Int.* **2023**, *63*, 1786–1797. [CrossRef]
21. Jan, M.B.; Chai, M. Machine learning approaches for creep rupture life prediction of metallic materials: A comprehensive review. *Int. J. Press. Vessel. Pip.* **2026**, *219*, 105690. [CrossRef]
22. Yang, T.X.; Dou, P. Prediction of creep rupture life of ODS steels based on machine learning. *Mater. Today Commun.* **2024**, *38*, 108117. [CrossRef]
23. Brownlee, J. *Data Preprocessing with Machine Learning*; Machine Learning Mastery: Vermont, Australia, 2020.
24. Fan, Y.; Zhang, H.; Ding, Y.; Wu, Y.; Guo, J.; Li, F. Multi-objective optimization based on reduced order model for VHTR design. *Ann. Nucl. Energy* **2026**, *237*, 112460. [CrossRef]
25. Qiao, Z.; Ning, S.; Gai, Y.; Xie, C. A digital twin guided physical-virtual denoising method for early fault detection of rolling element bearings. *Mech. Syst. Signal Process.* **2026**, *249*, 114108. [CrossRef]
26. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A Next-generation Hyperparameter Optimization Framework. In Proceedings of the KDD'19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 2623–2631. [CrossRef]
27. Holdsworth, S.R.; Askins, M.; Baker, A.; Gariboldi, E.; Holmström, S.; Klenk, A.; Ringel, M.; Merckling, G.; Sandstrom, R.; Schwienheer, M.; et al. Factors influencing creep model equation selection. *Int. J. Press. Vessel. Pip.* **2008**, *85*, 80–88. [CrossRef]

28. Wang, J.; Fa, Y.; Tian, Y.; Yu, X. A machine-learning approach to predict creep properties of Cr–Mo steel with time-temperature parameters. *J. Mater. Res. Technol.* **2021**, *13*, 635–650. [CrossRef]
29. Géron, A. *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow*, 2nd ed.; O'Reilly Media: Santa Rosa, CA, USA, 2019.
30. Wang, C.; Wei, X.; van der Zwaag, S.; Wang, Q.; Xu, W. From creep-life prediction to ultra-creep-resistant steel design: An uncertainty-informed machine learning approach. *Acta Mater.* **2025**, *292*, 121073. [CrossRef]
31. Chen, Y.; Zhao, Y. Machine learning-based predictions and analyses of the creep rupture life of the Ni-based single crystal superalloy. *Sci. Rep.* **2023**, *14*, 20716. [CrossRef]
32. Shekhar, S.; Bansode, A.; Salim, A. A Comparative study of Hyper-Parameter Optimization Tools. In Proceedings of the 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), Brisbane, Australia, 8–10 December 2021. [CrossRef]
33. Swetlana, S.; Rout, A.; Singh, A.K. Machine learning assisted interpretation of creep and fatigue life in titanium alloys. *APL Mach. Learn.* **2023**, *1*, 016102. [CrossRef]
34. Han, H.; Li, W.; Antonov, S.; Li, L. Mapping the creep life of nickel-based SX superalloys in a large compositional space by a two-model linkage machine learning method. *Comput. Mater. Sci.* **2022**, *205*, 111229. [CrossRef]
35. Agrawal, T. *Hyperparameter Optimization in Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2021. [CrossRef]
36. Bartz, E.; Bartz-Beielstein, T.; Zaefferer, M.; Mersmann, O. *Hyperparameter Tuning for Machine and Deep Learning with R: A Practical Guide*; Springer: Berlin/Heidelberg, Germany, 2023. [CrossRef]
37. Snoek, J.; Larochelle, H.; Adams, R.P. Practical Bayesian optimization of machine learning algorithms. *Adv. Neural Inf. Process. Syst.* **2012**, *4*, 2951–2959.
38. Bergstra, J.; Bardenet, R.; Bengio, Y.; Kégl, B. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24 (NIPS 2011)*; NeurIPS Proceedings: San Diego, CA, USA, 2011; Volume 2011, pp. 1–9.
39. Hatem, G.; Zeidan, J.; Goossens, M.; Moreira, C. Normality Testing Methods and the Importance of Skewness and Kurtosis in Statistical Analysis. *BAU J.-Sci. Technol.* **2022**, *3*, 7. [CrossRef]
40. Mohr, F.; van Rijn, J.N. Learning curves for decision making in supervised machine learning: A survey. *Mach. Learn.* **2024**, *113*, 8371–8425. [CrossRef]
41. Rasmussen, C.E.; Williams, C.K.I. *Gaussian Processes for Machine Learning*; The MIT Press: Cambridge, MA, USA, 2006; Volume 7.
42. Drucker, H.; Surges, C.J.C.; Kaufman, L.; Smola, A.; Vapnik, V. Support vector regression machines. *Adv. Neural Inf. Process. Syst.* **1997**, *1*, 155–161.
43. Ceungh, H.L.; Uvdal, P.; Mirkhalaf, M. Augmentation of scarce data—A new approach for deep-learning modeling of composites. *Compos. Sci. Technol.* **2024**, *249*, 110491. [CrossRef]
44. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; The MIT Press: Cambridge, MA, USA, 2019; Volume 29, ISBN 3463353563306. Available online: www.deeplearningbook.org (accessed on 1 January 2026).
45. Pan, Y.; Zeng, X.; Xu, H.; Sun, Y.; Wang, D.; Wu, J. Evaluation of Gaussian process regression kernel functions for improving groundwater prediction. *J. Hydrol.* **2021**, *603*, 126960. [CrossRef]
46. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
47. Mienye, I.D.; Jere, N. A Survey of Decision Trees: Concepts, Algorithms, and Applications. *IEEE Access* **2024**, *12*, 86716–86727. [CrossRef]
48. Khan, M.Y.; Qayoom, A.; Nizami, M.S.; Siddiqui, M.S.; Wasi, S.; Raazi, S.M.K.U.R. Automated Prediction of Good Dictionary EXamples (GDEX): A Comprehensive Experiment with Distant Supervision, Machine Learning, and Word Embedding-Based Deep Learning Techniques. *Complexity* **2021**, *2021*, 2553199. [CrossRef]
49. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
50. Baharvand, S.; Ahmari, H. *Application of Machine Learning Approaches in Particle Tracking Model to Estimate Sediment Transport in Natural Streams*; Springer: Dordrecht, The Netherlands, 2024; Volume 38. [CrossRef]
51. Chen, T.; Guestrin, C. XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16), San Francisco, CA, USA, 13–17 August 2016; pp. 785–794. [CrossRef]
52. Kleynhans, T.; Montanaro, M.; Gerace, A.; Kanan, C. Predicting top-of-atmosphere thermal radiance using MERRA-2 atmospheric data with deep learning. *Remote Sens.* **2017**, *9*, 1133. [CrossRef]
53. Williams, C.K.I.; Rasmussen, C.E. Gaussian Processes for Regression. *Adv. Neural Inf. Process. Syst.* **1996**, *8*, 514–520.
54. McCulloch, W.S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **1943**, *5*, 115–133. [CrossRef]
55. Rosenblatt, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychol. Rev.* **1958**, *65*, 386–408. [CrossRef] [PubMed]

56. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning Representations by Back-Propagating Errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
57. Wang, X.; Liu, Y.; Xin, H. Bond strength prediction of concrete-encased steel structures using hybrid machine learning method. *Structures* **2021**, *32*, 2279–2292. [CrossRef]