



POLITÉCNICA



**UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
AERONÁUTICA Y DEL ESPACIO
GRADO EN INGENIERÍA AEROESPACIAL**

TRABAJO FIN DE GRADO

**Validation Methods for Industrial Machine Learning
Problems**

AUTOR: Pablo YESTE BLESA

ESPECIALIDAD: Ciencias y Tecnologías Aeroespaciales

TUTOR DEL TRABAJO: Javier DE VICENTE BUENDÍA

Febrero de 2024



POLITÉCNICA



Resumen

Este trabajo presenta un proceso de validación diseñado para abordar los desafíos asociados a la certificación de modelos de redes neuronales en entornos industriales, con un enfoque particular en su aplicación en la industria aeroespacial. El proceso propuesto se enfoca en problemas de regresión comúnmente tratados mediante redes neuronales artificiales, poniendo especial énfasis en la fiabilidad y seguridad de la validación. A través de su aplicación práctica utilizando un modelo subrogado industrial, se demuestra la eficacia del proceso en la validación de métodos industriales en situaciones reales, específicamente en el ámbito del cálculo estructural y la predicción de modos de fallo en estructuras aeronáuticas. Finalmente, se examina el potencial de futuros desarrollos para ampliar la aplicabilidad del proceso a diversos dominios industriales y explorar paradigmas de aprendizaje alternativos, resaltando su capacidad para impulsar la innovación y la eficiencia en diferentes sectores industriales.

Abstract

This work presents a validation pipeline tailored to address the challenges of certifying neural network models within industrial contexts, with a focus on its application in the aerospace industry. The proposed pipeline targets regression problems commonly encountered with artificial neural networks, offering a structured approach to ensure reliability and safety. Through practical application using an industrial surrogate model, the efficacy of the pipeline is demonstrated in addressing real-world challenges, particularly in structural calculation and failure mode prediction in aeronautical structures. Additionally, future directions for extending the pipeline's applicability to diverse industrial domains and exploring alternative learning paradigms are discussed, highlighting its potential for driving innovation and efficiency across industries.

Contents

Resumen	iii
Abstract	v
Nomenclature	ix
Introduction.	xix
1 General concepts on supervised learning for industrial applications	1
1.1 Mathematical description of ANNs	1
1.2 Overfitting. Bias-variance trade-off	4
1.3 Applicability	6
2 The industrial problem	11
3 The validation pipeline	15
3.1 Overview	15
3.2 Data: size, features, dimensionality reduction	19
3.3 Train-test split	21
3.3.1 Geometrical analysis	21
3.4 Global error quantification	33
3.5 Distributed error quantification.	35
3.6 Distributed error quantification: conditioning the error distribution on the input space	47
3.6.1 Visualization of error as a function of categorical (discrete) input variables	48
3.6.2 Bias detection and quantification (1D)	50
3.6.3 Bias detection and quantification (2D)	54
3.6.4 Visualization of error as a function of numerical (continuous) input variables	56
3.6.5 Bias detection and quantification	56
3.7 Distributed error quantification: conditioning the error distribution on the output space	60
3.8 Uncertainty model	67

3.8.1	Global uncertainty model based on $P(E)$	68
3.8.2	Local uncertainty model based on $P(E Y)$	70
3.8.3	Local uncertainty model based on $P(E X)$	73
3.8.4	Full uncertainty model (FUM)	77
4	Conclusions and research outlook	81

Nomenclature

Acronyms		y	Output vector, prediction vector
ANN	Artificial neural network	\mathbf{y}_t	Ground true vector
ANOVA	Analysis of Variance	\mathcal{F}	Surrogate model
CI	Confidence interval	e	Residue
gESD	generalised Extreme Studentized Distribution [test]	m	Output dimensionality
K-S	Kolmogorov-Smirnov [fitness test]	n	Input dimensionality
MAE	Mean absolute error	S_X^*	Set of input vectors in dataset "/*"
MSE	Mean squared error	S_Y^*	Set of output vectors in dataset "/*"
PCA-99	Principal component analysis, explained variance=99%	S^{Full}	Full dataset
PDF	Probability density function	S^{Test}	Test dataset
RMSE	Root mean squared error	S^{Train}	Training dataset
Mathematics		x_i	Feature i of input vector \mathbf{x}
\mathbf{x}	Input vector, feature vector	y_i	Prediction i of output vector \mathbf{y}

List of Figures

2	Research fields in which Machine Learning represents an outbreak in technological development. [12]	xx
1.1	Smallest cube enclosing a set of 20 randomly generated points in a 3-dimensional euclidean space.	9
1.2	Convex hull enclosing the set of fig. 1.1. The convex hull's volume is always smaller than or equal to the volume of the hypercube.	9
2.1	Typical fuselage section of an aeroplane with semi-monocoque configuration. The main structural elements are highlighted in the image. Image edited under Creative Commons License from the original [26].	12
2.2	Surrogate model input-output data flux.	13
3.1	Validation pipeline overview.	16
3.2	Location of the train-test split assessment in the overall validation pipeline	22
3.3	Input [numeric] variables distributions double histogram. Only the first four input variables have been plotted.	30
3.4	Input categorical variables distributions double histograms	31
3.5	Output variables distributions double histograms	31
3.6	Requirements of the geometrical analysis	32
3.7	Scatter plot of ground true (x axis) against predicted values (y axis) of output variable "RF Net Tension", with the correspondent R^2 coefficient. In this case, $R^2 = 1,000$ indicates a perfect fit of predicted to ground true values. Mismatches due to underestimating failure risk are labelled in red, while those due to overestimating failure risk are labelled in blue. Similar graphs can be computed for every pair $\{y_j, \hat{y}_j\}$ of features in the output variables $\mathbf{y}_t = \{y_{t1}, y_{t2}, \dots, y_{tm}\}$. Plots for the rest of the six output variables of MS-S18 show analogous results.	34
3.8	Location of section 3.4 in the validation pipeline.	35
3.9	Double histogram depicting ground true values and model's predictions for the six output variables of MS-18.	37

3.10 Empirical residue distribution sampled from the test set, for each of the six output variables of MS-S18. x -axis limits have been truncated to $\mu \pm 3\sigma$, where the most part of the error lies. Histograms have been appropriately binned for a correct visualization.	39
3.11 Cumulative error distributions corresponding to the PDFs showed (as binned histograms) in Figure 3.10.	39
3.12 Graphical comparison of the p-values resulting from the K-S test for the MS-18 data.	41
3.13 (Binned) histograms depicting the distribution of the variable $z = \gamma + \sinh \frac{e-\xi}{\lambda}$, where $e \sim JohnsonSU(\gamma, \delta, \xi, \lambda)$ is the sampled residue. It can be shown that z converges to a normal distribution[41].	41
3.14 Box diagram showing the relative position of section 3.6 in the complete validation pipeline.	47
3.15 Box and whisker plots depicting the residue conditioned to two different categorical variables. Figure 3.15(a): "Frame". Figure 3.15(b): "Stringer". Clearly, outliers can be appreciated in both cases ("Fr69-Fr70" in Figure 3.15(a), and various stringers in Figure 3.15(b)).	49
3.16 Quantification of linear trend for input categorical variables "Stringer" (left) and "Frame" (right). Variables have been numerically encoded following a logical order (<i>e.g.</i> in the right image, six integers in the x axis represent the six possible values of variable Frame). The residue of output variable "RF Net Tension" has been plotted against the categorical variable (y axis). In this case, no linear trend is appreciated in neither of the two input variables (no bias present).	54

3.17 Scatter plots of residue against a particular numerical input variable. Left: Input var. FU.0420.26. A linear trend is appreciated, suggesting the model makes worse predictions as input load "FU.0420.26" is larger. Right: input var. FU.0420.25. No linear trend is appreciated, although severe heteroscedasticity can be observed. Note the vertical bar arrangement, showing particular input loads for which dispersion is unusually large. In view of the former results, the engineer may decide to investigate the underlying causes both for the linear trend in the left and for heteroscedasticity on the right. With the information, model and data boosting may be performed. More training data could be decided to be sampled in the range of [20000,60000] of FU.0420.26, for instance, and some regularization technique could be tried for the loss function in order to penalise inputs triggering high variance observed in the right.	56
3.18 Relative position of section 3.7 in the complete validation pipeline.	60
3.19 Goodness of fit of the error distribution (test set previously filtered) to some parametrised distributions. The test is performed individually for the output variables (Reserve Factors) displayed on top of each image. The goodness of fit is assessed with a 1-variable K-S test. The test set is filtered using the x axis, rejecting every point whose output \hat{y} is larger than x . If the p-value resulting from the K-S test in the filtered test set is greater than 0.05, the vertical slice at position x is labelled green (red if $p\text{-value} < 0.05$, grey if less than a threshold number of points –namely, 30– are present in the filtered test set).	62
3.20 Line plot of the p -value from the true-predicted distributions under a 2 sample K-S test. Distribution similarity is rejected if $p\text{-value} < 0.05$	64
3.21 Violin plots showing the absolute value residue for every bin.	65
3.22 Sliced-bar with p -values from the binned error under a K-S test to assess the goodness of fit to several parametrised distributions.	66
3.23 Global Uncertainty Model: error coverage. The coverage is higher than 95% for both output variables –for convenience just two output variables are depicted– what means the calibration criteria is successfully met.	70
3.24 Local Uncertainty Model based on $P(E \mathbf{Y})$: Scatter plot of the residue as a function of the predicted output \hat{y} , for output variables "RF Pure Compression" and "RF Shear Panel Failure".	71

3.25 Local Uncertainty Model based on $P(E \mathbf{Y})$: Violin plots of the residue, for binned output variables "RF Pure Compression" and "RF Shear Panel Failure"	71
3.26 LOUM coverage in the validation set. For both output variables coverage is above the minimum 95% threshold.	73
3.27 LIUM: Scatter plot of the residue of "RF Forced Crippling" as a function of the input x (left: x is "FU.0410.16". Right: x is "FU.0410.24").	74
3.28 Violin plots showing the residue of output variable "RF Forced Crippling". (a): Numerical inputs (binned). (b) and (c): categorical inputs.	75
3.29 LIUM coverage on the evaluation set. Residue of the output variable "RF Forced Crippling". The grey area represents the uncertainty CI calculated by the LIUM for the corresponding bin.	78
3.30 Conceptual visualization of FUM as a conservative merge between GUM (whose bootstrap interval [P2.5,P97.5] is represented by the two straight horizontal lines), LOUM and LIUM (whose bootstrap intervals [P2.5,P97.5] are represented indistinctively by the red and blue curves).	79

List of Tables

3.1	Output of algorithm 1	25
3.2	Voxel reference table	26
3.3	Points outside voxel hypercube	27
3.4	<code>reqs_results_table</code> . The first and the last requirements compare absolute figures of points at some isolation level with the overall number of points. Whereas the second and third requirements compare the difference in number of points between two consecutive levels with the total number of points.	27
3.5	P-values of the input variables. The test performed is a 2-sample Kolmogorov-Smirnov or a 2-sample χ^2 test, depending on the data being numerical (first 19 rows) or categorical ("dp", "Frame" and "Stringer").	29
3.6	P-values of the output variables distributions	30
3.7	Resulting p -values from the 2-sample K-S test of goodness of fit between ground true (\mathbf{y}_t) and predicted (\mathbf{y}) distributions. The null hypothesis H_0 that both distributions conform is rejected if the p -value is smaller than 0.05.	38
3.8	P-values of the K-S test comparing the empirical sample of $P(e)$ to the theoretical distributions indicated in each column. The null hypothesis H_0 (the empirical distribution has been sampled from the one figuring in a given column) is rejected when $p - \text{value} < 0.05$	40
3.9	Outlier detection taking $e \sim JohnsonSU$ as H_0 . N.B. for this table the usual requirements for classifying a point as an outlier ($z \in \sigma \pm 3\mu$ for the z-score and significance $1 - a \geq 95\%$ for the gESD) have been softened here for illustration purposes to $z \in \sigma \pm 1.2\mu$ and $a = 0.45$ for both tests, respectively.	42
3.10	Summary of bootstrapped error statistics. For the median, the Wilson-score[44] is used for computing the confidence interval.	45
3.11	Bootstrapped percentiles (1 st , 5 th , 10 th , 90 th , 95 th and 99 th) of the residue distribution, calculated with a 95% confidence interval using Wilson-score.	46

3.12 P-values results from the one-way ANOVA test. The red labelled cells show that variable "dp" shows bias for the residual distribution of "RF Net Tension", as well as variable "Frame" for the residual distribution of "RF Forced Crippling" and variable "Stringer" for the residual distributions of "RF Net Tension" and "RF Pure Compression". For the rest of the table, p-values greater than 0.05 indicate that H_0 cannot be rejected with a statistical confidence of at least 95%.	51
3.13 Z-score results for biased categorical variables. One table is generated for each pair biased categorical input variable-output variable.	52
3.14 2D bias quantification: Results from mean and variance z-score tests on biased Stringer-Frame input pairs (bias detection has been performed through 1-way ANOVA). For the input value combinations showed in the columns, either mean or variance (or both) show bias (weak: $z\text{-score} < 3$. Strong: $z\text{-score} > 3$). For the combination of Frame Fr67-Fr69 and Stringer Str07, there are not enough test points to compute the z -score variance test.	55
3.15 Bias detection and quantification on numerical input variables (the results for just five input variables are shown here – in columns–): P-value (statistical significance of the hypothesis that the variable is biased), Pearson coefficient, and slope of the best linear fit are shown in the first three rows. The last row shows the message "Biased" in case the p -value > 0.05 , "NO" otherwise. Data from this table corresponds to residue of the output variable "RF Column Buckling". Analogous tables exist for the rest of the output variables.	57
3.16 1-way ANOVA test results (p-values) for binned numerical input variables. For the output variable "RF Forced Crippling", bias is found in the input variables "FU.0420.25" and "FU.0430.25". Similarly, for the output variable "RF Net Tension", bias is found in the input variable "FU.0430.15".	58
3.17 Bias quantification in binned FU.0430.15 input variable. Columns represents bins showing bias. The same quantification methods employed for categorical variables (z -score for mean and variance outlier detection) have been used.	59
3.18 Summary of binned input variables bias quantification after binning the numerical variables and performing one-way ANOVA and z -score tests to every bin.	59

3.19 Binnarization of the test set. The test set has been divided into ten bins according to output values. Extremes of the six intervals –one for each Reserve Factor– defining the bins are shown below.	63
3.20 Global Uncertainty Model: Confidence Interval (mean and extremes) for output variables "RF Forced Crippling" (left) and "RF Column Buckling" (right). Header showing $[y_{min}, y_{max}]$ in the calibration set.	69
3.21 Global Uncertainty Model. The six confidence intervals (bootstrapped) defining the GUM alongside their coverage –in the calibration set– are showed.	69
3.22 Local Output Uncertainty Model: Bootstrapped CIs for each of the ten bins into which the output variable "RF Forced Crippling" has been binned. The mean of each CI is also given, as well as the bin extrema $[\hat{y}_{min}, \hat{y}_{max}]$ for every bin (in the first row).	72
3.23 LOUM coverage. Coverage in the validation set is displayed in the first row, whereas the coverage bootstrapped CIs are displayed in the second row.	73
3.24 LIUM: Bootstrapped CIs for binned input variable "FU.0430.26" (bins' extrema displayed as column names). Residue corresponds to output variable "RF Shear Panel Failure".	76
3.25 LIUM: Bootstrapped CIs for categorical input variable "dp" (possible values that the variable can take are displayed as column names). Residue corresponds to "RF Shear Panel Failure" output variable.	76

Introduction.

In recent years, artificial neural networks have become ubiquitous across industries, fundamentally reshaping operations and decision-making processes. Their integration into various sectors has heralded a new era of innovation and efficiency. From optimizing flight trajectories^[1] to enhancing predictive maintenance^[2–4], artificial neural networks have emerged as indispensable tools, enabling organizations to unlock insights and drive transformative outcomes. As the aerospace sector embraces this technological revolution, the adoption of neural networks underscores a strategic imperative to harness the power of machine learning in pursuit of greater precision, reliability, and safety. Particularly in the aerospace industry, the integration of ML models presents both tremendous opportunities and formidable challenges (vid. Figure 2). As the demand for advanced predictive analytics and automation escalates, so too does the necessity for rigorous statistical validation methodologies. The burgeoning reliance on ML algorithms for critical decision-making processes necessitates a paradigm shift towards comprehensive validation pipelines. Amidst this transformation, air-safety authorities have intensified their demands for stringent validation and verification processes^[5, 6] to ensure the safety and reliability of ML-driven systems deployed within aerospace environments. Yet, industry leaders have only recently begun to confront the complexities of certifying ML models^[7–11], prompting the initiation of discussions around the development of guidelines and a roadmap for design assurance, especially concerning network-related technologies. This pressing need underscores the imperative for collaborative efforts within the industry to establish robust validation frameworks that not only meet regulatory standards but also address the evolving challenges posed by ML integration.

In the forthcoming chapters, this work delineates a comprehensive validation pipeline tailored to address the challenges of certifying neural network models, with a special focus in the aerospace industry. Chapter 1 serves as the foundational framework, focusing on the class of problems amenable to the proposed validation methodology. Principally, the validation pipeline targets regression problems, acknowledging their prevalence in industrial settings employing artificial neural networks (ANNs). While the proposed pipeline aspires to versatility, accom-

DISCIPLINE	MACHINE LEARNING APPLICATION
Aerodynamics	Optimizing the design and operation of the aircraft. Wing design. Flight control. Aircraft performance optimization.
Aeroacoustics	Noise reduction. Aeroacoustic optimization. Flight path optimization. Noise prediction.
Combustion	Optimizing the combustion process in aircraft engines. Combustion optimization. Emission prediction. Fuel optimization. Predictive maintenance.
Structural health monitoring	Enabling early detection and diagnosis of structural damage or degradation. Predictive maintenance. Damage detection. Structural health prognosis. Structural design optimization.

Figure 2: Research fields in which Machine Learning represents an outbreak in technological development. [12]

modating a broad spectrum of problems, this work primarily centers on regression problems encountered with ANNs in industry.

Within [chapter 1](#), a formal exposition of these regression problems is provided, elucidating key concepts essential to subsequent chapters. Notably, the discussion encompasses vital notions within the context of surrogate models, including overfitting, applicability region of a surrogate model, etc. These concepts serve as recurring themes throughout the ensuing chapters, underpinning the methodology and rationale behind the proposed validation pipeline.

Moreover, to exemplify each facet of the validation pipeline, rigorous testing is conducted employing an industrial surrogate model. This surrogate model encapsulates real-world complexities inherent in a significant problem domain: structural calculation and failure mode prediction in aeronautical structures. The utilization of this surrogate model facilitates a practical illustration of the validation pipeline’s efficacy, demonstrating its applicability to high-stakes aerospace applications and showcasing its potential to enhance safety and efficiency within the industry.

Transitioning to [chapter 2](#), a concise introduction to the industrial problem domain and the surrogate model utilized for validation purposes is provided. This chapter serves to contextualize the subsequent discussions on validation methodologies within the specific industrial context, laying the groundwork for a detailed exploration of the validation pipeline’s application.

[Chapter 3](#) constitutes the core of this work, delving into the validation pipeline in depth. Employing a directed graph to encapsulate its operational framework, this chapter offers a comprehensive overview of the validation process. Each component of the pipeline is scrutinized in detail, with dedicated sections elucidating their roles and contributions to the overarching validation methodology. By systematically dissecting the validation pipeline, this chapter elucidates its intricacies, offering insights into its design rationale and practical implementation within aerospace contexts.

Finally, the concluding chapter provides a synthesis of the findings, offering insights into the efficacy of the proposed validation pipeline and outlining avenues for future research. Additionally, an "outlook research" propounds potential directions for advancing validation methodologies in response to evolving industry demands and technological advancements. Through these concluding reflections, this work underscores the significance of collaborative efforts in establishing robust validation frameworks and driving innovation within the aerospace sector.

General concepts on supervised learning for industrial applications

In the dynamic landscape of industrial applications, the integration of supervised learning techniques has emerged as a pivotal force in addressing complex challenges, particularly within the aerospace industry. This chapter delves into the foundational concepts of industrial-oriented supervised learning, focusing on its profound implications for solving regression problems inherent in various aerospace endeavors. From optimizing fuel consumption[13] to enhancing structural integrity[14], numerous industrial dilemmas can be effectively framed as regression tasks, where supervised learning algorithms offer unparalleled potential for both solvability and computational efficiency. Within the realm of industry-oriented supervised learning, understanding the mathematical background of these methodologies lays the groundwork for harnessing their transformative impact on real-world industrial challenges, particularly within aerospace domains.

1.1 Mathematical description of ANNs

Mathematically, the problems we refer to in this chapter consist of finding the explicit shape of a function

$$\mathcal{F}_t : S_X \rightarrow S_{Y_t} \quad (1.1)$$

$$\mathcal{F}_t(\mathbf{x}) = \mathbf{y}_t \quad (1.2)$$

with $S_X \subset \mathbb{R}^n$ and $S_Y \subset \mathbb{R}^m$; *i.e.* the function \mathcal{F}_t takes as input a vector $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$, and outputs a vector $\mathbf{y}_t = \{y_{t1}, y_{t2}, \dots, y_{tm}\}$. The input vector could represent, for instance, a collection of loads applied to some part of the fuselage structure of an aeroplane, whereas the output vector may represent the probability of failure of that structure under the loads contained in \mathbf{x} . The problem at hand would therefore be finding the function \mathcal{F}_t which maps load configurations to failure probabilities. By categorizing the task of finding such function \mathcal{F}

2 Chapter 1. General concepts on supervised learning for industrial applications

as a "regression problem", we mean usually $n \gg m$. Note that finding the explicit shape of \mathcal{F}_t might be a difficult task when \mathcal{F}_t is a strongly non-linear function, especially if $n \gg 1$.

Although regression problems can be tackled by several means, in this work we will focus on artificial neural networks or ANNs[15]¹. From this approach, the starting point is a parametrized function \mathcal{F} with which we shall approximate \mathcal{F}_t (which is taken as the ground true –unknown– function that conveys the physics of the regression problem²), so that we have:

$$\mathcal{F} : S_X \rightarrow S_Y \quad (1.3)$$

$$\mathcal{F}(\mathbf{x}; W) = \mathbf{y} \quad (1.4)$$

Where W is the set of parameters defining the explicit shape of \mathcal{F} . In this equation, the absence of subscript "t" accounts for the difference between our parametrized function and the real function we are trying to find, the one from eq. (1.1). N.B. the domain set S_X is the same for both \mathcal{F}_t and \mathcal{F} , which is not the case for the codomain.

When speaking about regression problems, the input vector \mathbf{x} is usually called the "feature vector", while the output vector \mathbf{y}_t is usually called –in a descriptive way– the "prediction vector". Similarly, n is called the "feature" or the "input" dimension, and m is called the "output" or the "prediction" dimension.

We shall denote \mathcal{F} 's dependency on W with a subscript: \mathcal{F}_W . Mathematically, the explicit shape of \mathcal{F}_W can adopt a wide variety of expressions. Perhaps the most common architecture is the so-called Multi Layer Perceptron (MLP)[15], which basically consists of a series of iterative affine transformations (defined by the parameters W), the first of which is applied to the elements of \mathbf{x} and non-linear functions (the rectified linear unit function or ReLU, the sigmoid function and the hyperbolic tangent being the most widely used) applied to each iteration's result.

The approach now is to define a distance $\mathcal{L} = \|\mathcal{F}_t - \mathcal{F}\|$ and treat the regression problem as the task of minimizing \mathcal{L} :

$$\mathcal{F}^* = \operatorname{argmin}(\mathcal{L}) \quad (1.5)$$

From a practical point of view, the information available for defining \mathcal{L} consists of a (large³)

¹When using an ANN to solve a regression problem, we shall refer to the ANN as a "'surrogate model" as well.

²Indeed, \mathcal{F}_t is just a conceptual instrument we use to associate pairs of input-output values $(\mathbf{x}_i, \mathbf{y}_i)$.

³By large, we mean $|S^{\text{Full}}| \gg n$ (cfr. eq. (1.6)).

set of duplets $(\mathbf{x}, \mathbf{y}_t)$ ⁴ representing empirical samples from the feature and the prediction space coming from experimental data, simulations, or any other source. Such set is called S^{Full} :

$$S^{\text{Full}} = \{\mathbf{x}_i, \mathbf{y}_{ti}\}_{i=1}^{N_{\text{Full}}} \quad (1.6)$$

The procedure is to split Full into a training and a test set, so that

$$S^{\text{Train}} \cup S^{\text{Test}} = S^{\text{Full}} \quad (1.7)$$

$$S^{\text{Train}} \cap S^{\text{Test}} = \emptyset \quad (1.8)$$

$$|S^{\text{Train}}| > |S^{\text{Test}}| \quad (1.9)$$

\mathcal{L} (which is called the "loss function") can now be defined as

$$\mathcal{L} = \mathcal{L}(W) \quad (1.10)$$

$$\mathcal{L}(W) = E(W) + \lambda R(W) \quad (1.11)$$

where

$$E(W) = \|S_{Y_t}^{\text{Train}} - S_Y^{\text{Train}}\| \quad (1.12)$$

and

$$R(W) = \|W\| \quad (1.13)$$

for an arbitrary norm $\|\cdot\|$. In eq. (1.11), λ is just an arbitrary scaling factor. For the explicit dependence of E on W , recall eq. (1.4)⁵.

\mathcal{L} is called the "loss function". We can see from eq. (1.11) that it combines the effect of two terms: the error term (eq. (1.12)) accounts for the mismatch between actual data in \mathcal{F} 's codomain and our parametrized function's predictions, whereas the regularization term (eq. (1.13)) imposes a restriction on the set W (which is done mainly for numerical stability reasons). The exact definition of $E(W)$ and $R(W)$ is an arbitrary choice. Common used error functions are:

⁴N.B. the lack of hat here, meaning data has been sampled from the domain and codomain of \mathcal{F} , S_X and S_Y , respectively.

⁵N.B. S_Y^{Test} is not available beforehand; it is generated by applying \mathcal{F}_W to S_X^{Test} .

- The Mean Square Error (MSE):

$$E(W) = \frac{1}{N_{\text{Train}}} \sum_{i=1}^{N_{\text{Train}}} (\mathbf{y}_{ti} - \hat{\mathbf{y}}_i)^2 \quad (1.14)$$

- The Mean Absolute Error (MAE):

$$E(W) = \frac{1}{N_{\text{Train}}} \sum_{i=1}^{N_{\text{Train}}} |\mathbf{y}_i - \hat{\mathbf{y}}_i| \quad (1.15)$$

For the regularization term, a common choice is the L_2 norm.

Let us rewrite now the optimization problem defined in eq. (1.5) the following way:

$$W^* = \operatorname{argmin}\{L(W)\} \quad (1.16)$$

The algorithm commonly used to find W^* is gradient descent:

$$W(k+1) = W(k) - \eta \nabla_W \mathcal{L}(W) \quad (1.17)$$

Where, again, η is an arbitrarily defined scaling factor which receives the name of "learning rate".

A plethora of different algorithms have been tried for ANN training, but the majority of them shares the basics with gradient descent. Amongst common techniques, descending learning rates and stochastic gradient descent are perhaps the most widely used. Descending learning rate just means λ decreases as successive loops of eq. (1.17) are completed. Stochastic gradient descent (cfr. [15] for further details) applies the loss function, eq. (1.11) to a batch that has been randomly sampled from S^{Train} instead of the whole set.

1.2 Overfitting. Bias-variance trade-off

An important phenomenon to care about during training of ANNs is **overfitting**⁶. When aiming to solve any regression problem with ANNs, one must decide on the architecture (by architecture we mean the explicit definition of \mathcal{F}_W). One example of possible network architectures are the aforementioned Multi Layer Perceptrons. An important characteristic of the

⁶For more extensive treaty on this topic, cfr. [15]

network is the size of the set W . Popular wisdom in machine learning says the more complex, non-linear the function \mathcal{F}_t is, the larger the parameter set W needs to be in order to properly capture all the features in \mathcal{F}_t . One can build an analogy between a regression problem with ANNs and a polynomial interpolation problem. If \mathcal{F}_t is supposed to be a polynomial⁷, then the objective of minimizing the loss function as defined in eq. (1.14) is equivalent to of finding a polynomial that interpolates through the points $\{(\mathbf{x}_i, \mathbf{y}_{ti})\}_{i=1}^{N_{\text{Train}}}$ in S^{Train} (as the loss would vanish to zero when all the points be fitted by virtue of eq. (1.12)). In this case W would just represent the set of coefficients $\{c_0, c_1, \dots, c_{N_W}\}$ that defines the polynomial \mathcal{F}_W . The minimum degree of the polynomial that solves this interpolation problem is clearly $N_{\text{Train}} \equiv |S^{\text{Train}}|$. Now, if we set $N_W < N_{\text{Train}}$, obviously we would not be able to fit all points in the training set as the degree of the interpolating polynomial would not be large enough to solve the problem, and the minimum of the loss function as defined in eq. (1.11) would be presumably high. When this occurs, it is said that the model \mathcal{F}_W has **high bias**. Due to the lack of sufficient trainable parameters in W , the model is not capable of learning all the intricacies and patterns in the training set. Our model is just not complex enough for the regression task we aimed to solve.

On the other hand, setting $N_W \gg N_{\text{Train}}$ can lead to the so-called overfitting. In industrial applied regression problems, data in S^{Full} comes from experiments or numerical simulations, as has been mentioned before. This type of data inherently encompasses various sources of noise, which may originate from sensors, numerical instabilities within simulations, signal interference, among other factors. Now take for instance that the target function \mathcal{F}_t is a polynomial of degree $N_1 \ll N_{\text{Train}}$. This means our training set is redundant as we have sampled N_{Train} points from a polynomial of degree N_1 . If data was noise free, our interpolation problem would be determined by just N_1 points in the test set and coefficients $\{c_{N_1+1}, c_{N_1+2}, \dots, c_{N_W}\}$ would be identically zero. But presence of noise deviates points in S^{Train} from the actual polynomial \mathcal{F}_t , and thus makes the interpolation problem non redundant, *i.e.* the coefficients $\{c_{N_1+1}, c_{N_1+2}, \dots, c_{N_W}\}$ would be different from zero and our surrogate model \mathcal{F}_W will have a higher degree than the actual \mathcal{F}_t , as it would try to fit the noise in S^{Train} . This is called overfitting. Clearly this is undesirable from all points of view, namely:

- Computational resources are invested in fitting irrelevant noise.
- By fitting the noise in the training set, the model loses generalization capability, *i.e.*

⁷Usually there are absolutely no guarantees that $\hat{\mathcal{F}}_W$ is a polynomial. Here we are just assuming it for illustration purposes.

although the loss \mathcal{L} in the training set might be very low due to the model correctly fitting every point in that set, performance in the test set drops due to the noise patterns the model has learnt in the training set not being present at the test set. This will cause a mismatch between the training and the test loss (low and high, respectively). When this occurs, we refer to it as the model having **high variance**.

It is clear from all the above reasons that during the training phase it is paramount to achieve an equilibrium between bias and variance, *i.e.* between configuring a model complex enough to capture all the patterns and correlations in the data and avoiding overfitting.

There exists several regularization techniques to avoid overfitting, mainly early stopping and random neuron deactivation (dropout)[15].

Bias-variance trade-off will be a present idea throughout the validation pipeline. One of the goals of the validation is to prevent overfitting and to provide mechanisms to detect it.

1.3 Applicability

Another important concept which will be later used in chapter 3 is the applicability region of a surrogate model.

The applicability region can be intuitively defined as the set of the input and output values for which the model's prediction can be trusted. Applicability region is a refined idea of the *Operational Design Domain* of a model.

For a model \mathcal{F} to be applicable to a certain new point \mathbf{x} :

- \mathbf{x} needs to be inside the *input* applicability region of \mathcal{F} .
- The prediction $\mathbf{y} = \mathcal{F}(\mathbf{x})$ needs to be inside the *output* applicability region of \mathcal{F} .

Knowing whether a new point is either inside or outside applicability is a binary decision problem which can be addressed *a priori* (filtering) or be learned *a posteriori* (model boosting). In other words, the binary classification can be addressed from an unsupervised (geometric) learning approach, or a supervised learning approach:

A. Supervised applicability classifiers

Supervised applicability classifiers are based on exogenous criteria. Examples of them include:

- **Operational design domains.** Based on engineer/scientist expertise, only certain combinations and range of features make physical sense. These combinations are used to define a (geometrical) applicability region.
- **Error-filtered convex hull.** Initially the whole set under testing is in applicability range. After filtering those test cases whose prediction error exceeds a tolerance, the applicability region is defined as the convex hull of the remaining subset. The convex hull can be simply defined as the smallest convex set containing the data[16] (vid. Figure 1.2).
- **Error-labelled classifiers.** Initially the whole test set is in applicability range. After labelling those test cases whose prediction error exceeds a tolerance as outside applicability, different binary classifiers are trained on the full dataset.

B. Unsupervised applicability classifiers

On the other hand, geometrical applicability classifiers are based on the premise of ANN interpolation capabilities. It has been widely discussed (see *e.g.* [17–19]) that ANN’s performance relies on their interpolation capabilities, and that no extrapolation capabilities outside their applicability region should be assumed.

Subsequently, from a geometrical approach to the applicability classification problem, we shall define the input applicability region as the geometrical region in the input space where the model is known to work in interpolating regime.

There are plenty of definitions for the interpolating region of a given dataset. Some authors define it as the smallest hypercube enclosing the data[20] (vid. fig. 1.1) although this may seriously challenge interpolation due to isolated points falling into the interpolation region under this definition. Many authors (vid. [21, 22]) define the interpolation region as the *convex hull* of the training data, *i.e.* :

Definition 1. [23] Standard interpolation occurs for a sample \mathbf{x} whenever this sample belongs to the convex hull of a set of samples $S_X^{Train} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$.

Recently, the authors from [23] have pointed out that, due to the so-called curse of dimensionality, extrapolation outside the training convex hull always ends up taking place if the input dimension is sufficiently large. The curse of dimensionality[15, pp. 17-18] can be illustrated by the fact that the unitary n -sphere’s volume asymptotically diminishes to 0 as n increases. In

the end, the effect is that, as the number of dimensions increase, more data is needed in order for the model to work in interpolating regime, as is showed by Theorem 1.3.1.

Theorem 1.3.1. [24] Given a d -dimensional dataset $S_X = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with i.i.d. samples uniformly drawn from a hyperball, the probability that a new sample \mathbf{x} is in interpolation regime (recall definition 1) has the following asymptotic behaviour:

$$\lim_{d \rightarrow \infty} p(\mathbf{x} \in \text{Hull}(\mathbf{X})) = \begin{cases} 1 & \iff N > d^{-1}2^{\frac{d}{2}} \\ 0 & \iff N < d^{-1}2^{\frac{d}{2}} \end{cases} \quad (1.18)$$

The main answer to the argument of [23] is that interpolation does not occur in the ambient space, but in a latent, low-dimensional space of the input data[25]. This leads to a new definition of interpolation alternative to definition 1.

Interestingly, the authors in [25] go beyond demonstrating that interpolation occurs (at least in a latent representation of the ambient space). They unveil two more crucial conditions for optimal model performance:

- The training and testing data should share the same cumulative distribution. In situations where this is impossible, matching the tail distribution becomes essential.
- The testing points should not be isolated within the dataset. This means they do not constitute extreme outliers in the probability distribution function of the training points, that testing and training data follow the same cumulative distribution (or the same tail distribution whenever the former cannot be fulfilled).

Examples of unsupervised applicability classifiers include:

- **Hypercube range classifier.** The applicability region is defined as the boundary of the hypercube whose edges are the ranges $[min, max]$ of each input variable in the training set. In practice, this amounts to checking if each numerical feature X_i of the test point lies inside the range spanned by the training subset.
- **Input space convex hull classifier.** The applicability region is defined as the boundary of the convex hull of the training set.

- **Isolated region detection through geometrical analysis.** Even if a point is inside the convex hull of the training set, it can be very far away from other points, thus generalization can be challenged. Isolated region detectors address this by checking the statistical vicinity of train points and comparing it to the distance of a given test point to the closest training point.

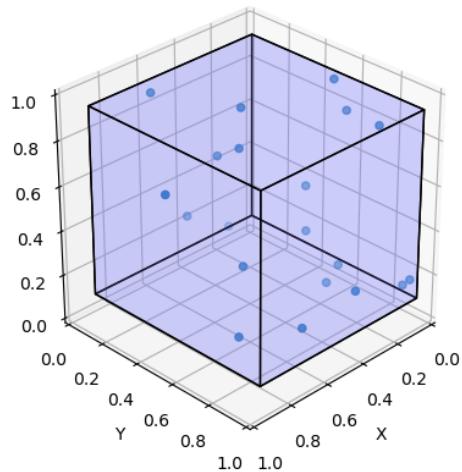


Figure 1.1: Smallest cube enclosing a set of 20 randomly generated points in a 3-dimensional euclidean space.

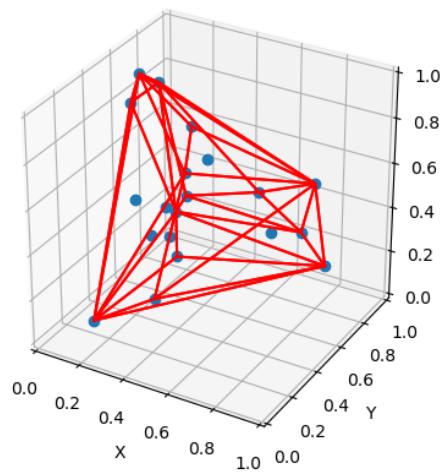


Figure 1.2: Convex hull enclosing the set of fig. 1.1. The convex hull's volume is always smaller than or equal to the volume of the hypercube.

The industrial problem

The whole validation pipeline proposed in the next section is illustrated with realistic data from a real industrial problem of particular interest to the aerospace sector: stress analysis and failure prediction in aircraft fuselage structures. Results from statistical tests, graphs and tables are provided alongside the theoretical background at each step. This chapter is thus dedicated to introduce the industrial problem used for illustration of results.

The industrial problem at hand is an application of ANNs in aircraft stress engineering. In aircraft engineering, the typical structural configuration of airplanes involves a semimonocoque design, characterized by a combination of internal framing and an external covering (known as "skin"). This configuration has been used almost exclusively in the aviation industry since the II World War, owing to its ability to provide strength and durability while minimizing weight, which is crucial for aviation applications.

The main elements of this structural configuration include (vid. [Figure 2.1](#)):

- **Skin:** The outermost layer of the aircraft, forming the aerodynamic surface. It bears the aerodynamic loads and distributes them across the structure. The skin is typically made of lightweight yet robust materials such as aluminium or composite materials.
- **Frames:** Internal structural components that run perpendicular to the longitudinal axis of the aircraft, providing shape and rigidity to the fuselage. Frames are spaced along the length of the fuselage and are connected to the skin, forming the structural framework.
- **Stringers:** Longitudinal structural members that run parallel to the longitudinal axis of the aircraft. They are attached to the frames and provide additional reinforcement to the skin, helping to distribute loads and maintain structural integrity.

Together, the skin, frames, and stringers form a robust yet lightweight structure that can withstand the stresses and loads experienced during flight. This configuration is essential for ensuring the safety and performance of aircraft.

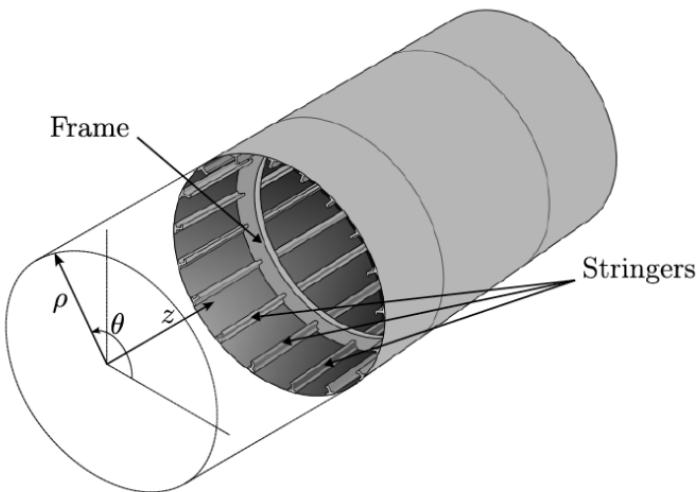


Figure 2.1: Typical fuselage section of an aeroplane with semi-monocoque configuration. The main structural elements are highlighted in the image. Image edited under Creative Commons License from the original [26].

For typical aircraft fuselage panel design, the dominant form of stiffened post buckling failure under shear loading is forced crippling[27]. This occurs when the shear buckles in the panel skin force the attached stiffener (stringers) flanges to deform out-of-plane. There are other failure modes related to buckling, tension, and compression.

The surrogate model presented here is the "MS-S18" model, whose aim is to predict Reserve Factors (*RF*) that quantify failure likelihood for regions of the aircraft subject to different loads happening in flight maneuvers (vid. Figure 2.2). There are six possible failure modes, whose names are "Forced Crippling", "Column Buckling", "In Plane", "Net Tension", "Pure Compression", and "Shear Panel Failure".

Input data consists of $n = 22$ features (loads applied to a specific region of the aircraft and different maneuver specs). These variables are either numerical continuous variables (the magnitudes of the applied loads) or categorical variables. These are variables which can only take a limited set of values (*e.g.* boolean variables, integer variables, etc.) There are three categorical variables alongside 19 numerical ones in MS-S18's dataset. Their names are "dp", "Frame", and "Stringer"¹. They contain geometrical information about the region where the loads (whose magnitudes are described by the 19 numerical variables) are being applied. For instance, "Frame" and "Stringer" identify structural elements of the fuselage (their values are

¹Variables' names are an arbitrary decision and completely irrelevant to this work. Nevertheless, they are provided here for illustration.

string variables like "Fr64-Fr65" and "Str08", referring to the interstitial region between fuselage frames no. 64 and no. 65 and the stringer no. 8, respectively).

Output data consists of 6 reserve factors that quantify the stress failure likelihood:

$$\mathbf{y} = (RF_1, RF_2, RF_3, RF_4, RF_5, RF_6)$$

where $RF_i \in [0, 5]$ In this (logarithmic) scale, 0 means extreme risk and 5 means risk extremely low.

This study prioritizes assessing the efficacy of the developed validation tool, not optimizing the neural network model itself. Our focus remains solely on its inputs –the 22 aforementioned features– and its outputs –the 6 reserve factors representing 6 distinct failure modes–. This simplification allows us to isolate and evaluate the performance of the validation tool without introducing confounding variables related to the specific neural network design (*i.e.*, without explicitly specifying the network's architecture and its parameters, W).

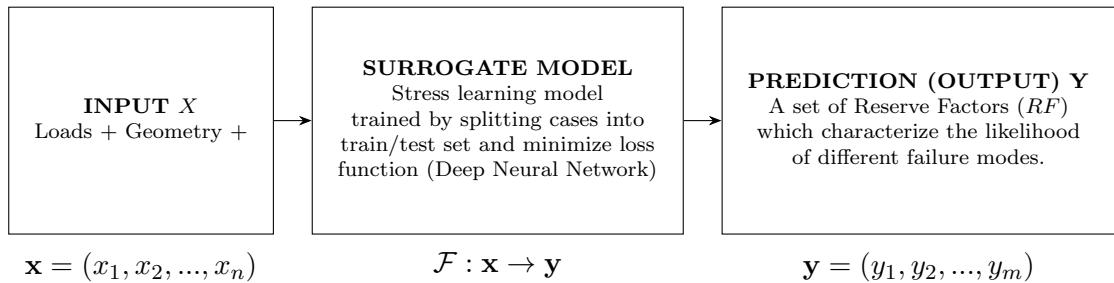


Figure 2.2: Surrogate model input-output data flux.

The validation pipeline

3.1 Overview

In Figure 3.1 the proposed validation pipeline is depicted in the form of a directed graph. Each block of the diagram represents a different station in the pipeline, which consists of a comprehensive statistical validation part, whose ultimate product is the "uncertainty model" labelled block, plus the "XAI" block and a section dedicated to "model and data boosting", which receives the outputs of the statistical validation and the uncertainty model, and interacts with "Data" and "Model definition and training" blocks in an effort to leverage information collected for the statistical validation to enhance model performance.

This work is exclusively focused on the statistical validation of the model. This comprises the vertical streamline starting in the block "Data" and finishing in " $P(E)$ ", " $P(E|X)$ " and " $P(E|Y)$ " blocks, with the "Uncertainty model" block as the final outcome of the whole validation, which in turn is a tool that uses all the information of previous blocks to predict *when* and *how much* the model can be trusted. The whole validation loop can be described in terms of the following blocks:

- 1. Data: size, features, dimensionality reduction.** The first milestone in the validation pipeline is related to data acquisition and manipulation prior to dataset splitting into train and test sets, *i.e.* its focus is the set

$$S^{\text{Full}} = \{\mathbf{x}_i, \mathbf{y}_{ti}\}_{i=1}^{N_{\text{Full}}}$$

Focusing on the set size and input dimensionality requirements, as well as encoding algorithms for categorical variables and leveraging dimensionality reduction of data. All of this is developed in detail in section 3.2.

- 2. Train-test split** In supervised learning applications, the dataset is typically divided into the training and the test sets. Keeping different sets for each task is fundamental in

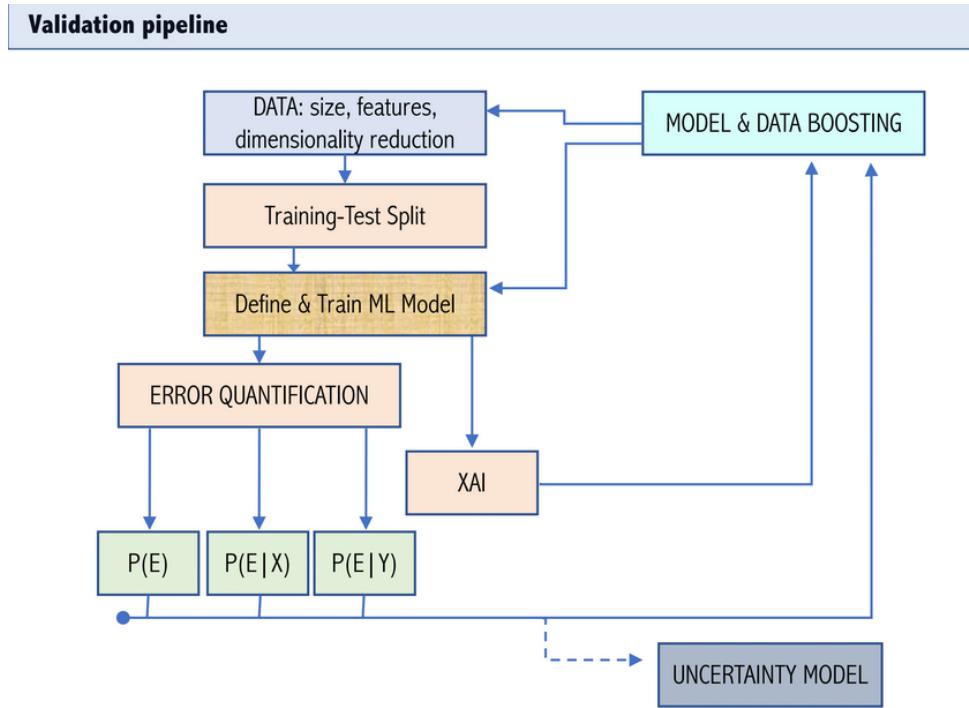


Figure 3.1: Validation pipeline overview.

order to prevent high variance (cfr. section 1.2). In the model validation loop, the first question that should be asked, even before training the model, is whether the dataset split is appropriate for training (vid Figure 3.2).

Section 3.3 delves deeper in the tests and mechanisms proposed for assessing train-test split adequateness and fixing possible issues.

3. **Define and train the ML model** This station deals with the surrogate model architecture and size in terms of trainable parameters, as well as model training. There exist a plethora of different ANN architectures in the literature, ranging from the Multi-Layer Perceptron (section 1.1) to more exotic setups such as Convolutional Neural Networks or CNNs[28], widely used for image recognition tasks, or Recurrent Neural Networks or RNNs[29], used for temporal dependent problems.

As has been widely discussed in chapter 1, the model size in terms of the trainable parameter set size plays an important role in the bias-variance trade-off problem. Carefully tuning this hyperparameter is key in achieving low error.

AS with the model size, the training length plays a similar role. When training with the algorithm from eq. (1.17), a complete tour of the training dataset is called an "epoch".

If the model is trained for too long (a very large number of epochs) overfitting may start appearing. This can be easily detected by plotting the training and the test loss against the number of epochs. When the two start diverging, training should be stopped as test loss is increasing due to overfitting. Mechanisms such as early-stopping[30] or dropout (cfr. section 1.2) can help preventing over-training.

4. **Global error quantification** After having defined and train the surrogate model, it is the turn of assessing its performance. In other words, error measuring. The first step in this direction are global-error figures based in point-wise error metrics. Further discussion is provided in [section 3.4](#).
5. **Distributed error quantification:** $P(E)$. When the error distribution is non-Gaussian (which is very common) this block becomes mandatory as global error metrics don't provide enough information. Moreover, this block deals with error outlier detection and is the first building block of the uncertainty model. More details are provided in [section 3.5](#).
6. **Error distribution conditioned on the input space:** $P(E|X)$. The second milestone in the uncertainty model building journey is reconstructing the local error distribution conditioned on input data. Furthermore, another incentive for taking this effort is that dataset boosting needs information of the error as a function of the input variables, in order to optimise data collection in [section 3.2](#). More discussion on this topic can be read in [section 3.6](#).
7. **Error distribution conditioned on the output space:** $P(E|Y)$. A basic scatter plot of predicted against true values in [section 3.4](#) usually tells us that the error is not independent from predicted values at all. This is also the final building block of the uncertainty model. More details are given in [section 3.7](#).
8. **Uncertainty model** This block represents the ultimate achievement in the statistical validation part. Making use of the knowledge gathered in previous sections, an extremely powerful tool is developed: an uncertainty model which tells us in probabilistic terms *when* and *how much* the model's predictions can be trusted. The true power of this concept resides in its applicability not only to the test set, but to new, unseen predictions of the model. This tool operates by predicting the model's error over new predictions based on the distributed error information gained in previous sections.

9. **XAI** The issue of explainability of machine learning (ML) models is gaining increasing attention among industry stakeholders, particularly those involved in the integration of artificial neural networks (ANNs). This concern is particularly pronounced in sectors such as aerospace[5, 6], where trustworthiness and reliability are crucial throughout the manufacturing chain. ANNs, with their complex architectures and training algorithms, often yield results that are challenging to interpret, adding to the urgency of addressing explainability concerns.
10. **Model and data boosting** This block acts as a hub which receives interpretable information from the blocks dedicated to the statistical analysis, and outputs feedback to those blocks directly involved with dataset and model building (and training, in the last case). Model and data boosting block so does close the validation loop, connecting the *analysis* blocks which output information about an existing model or dataset, with the *construction* blocks which make that dataset and model exist. This configuration is intended to leverage the statistical analysis to not only address certification tasks, but also improve the model performance overall.

In the upcoming sections, the primary emphasis will be on the statistical validation of the model, rendering the 9th and 10th blocks beyond the scope of this study. However, while the initial block pertaining to data is not strictly within the realm of statistical validation, a succinct overview is provided in section 3.2. This overview serves to furnish insightful context for the subsequent chapters.

3.2 Data: size, features, dimensionality reduction

For industrial regression models such as the MS-S18, data may contain noise to some uncertainty levels, as discussed in [chapter 1](#). This changes the perspective from data duplets of the shape

$$(\mathbf{x}, \mathbf{y})_j = ((0.25, -0.95, \dots, -0.02); (0.1, 0.4, \dots, 0.7))$$

to random variables \mathbf{x} and \mathbf{y} whose cumulative probability functions define confidence intervals in which finding x and y has a 95% probability, *i.e.* elements of S^{Full} have now the shape

$$(\mathbf{x}, \mathbf{y})_j = (([0.21, 0.29], [-0.98, -0.93], \dots, [-0.05, 0.01]); ([0.0, 0.3], [0.3, 0.5], \dots, [0.67, 0.73]))$$

Now, for training an ANN, it data needs to be uncertainty free. The approach is to take p samples from the random variables $(\mathbf{x}, \mathbf{y})_j$ inside their confidence intervals. It is possible now to combine the sampled tuples to obtain p^2 new tuples. This is indeed one method of data augmentation¹.

One primary characteristic of S^{Full} is the number of features in the input data, n (cfr. [chapter 1](#)). Ideally, input data should contain as many features as needed to faithfully convey all the information about the physics of the regression problem at hand (so the more features, the more information the input vector \mathbf{x} conveys). But it is also important for training purposes that the training set uniformly –to some extent– samples the hypercube defined by the input variable ranges, otherwise extremely isolated points would produce bad performance in uncovered regions (recall applicability region from [chapter 1](#)). As n increases, the required size of S^{Train} to uniformly sample the training set hypercube increases exponentially (to see this, think about how the hypervolume of an n -hypercube increases as n does so). This effectively imposes a limit to the input dimensionality n . Data augmentation might help when it is required to enlarge n but augmenting the training set size with traditional methods (experimental data, numerical simulations, etc.) is too expensive.

Data encoding can also play an important role at this station. Input data can be either numerical or categorical, as described in [chapter 2](#). Due to the way ANNs’ are parametrized (cfr. [section 1.1](#)), it is required that all input variables be numerical. This can be achieved by one-hot encoding categorical variables² although other encoding algorithms may be optimal in

¹For a more detailed description of the concept of data augmentation, cfr. [31]

²The basic idea behind **one-hot encoding** is to represent each unique category of a categorical variable as a binary vector, where each element of the vector corresponds to a category and is set to 1 if the observation

computational terms. As an illustration, for the model MS-S18 there exists a logical way of encoding frames and stringers taking advantage of the circular symmetry involving the stringer configuration around frames. This makes it possible to uniquely identify each stringer with a single value corresponding to the stringer's θ coordinate in the reference system displayed in Figure 2.1. Similarly, each frame can be encoded with a single value corresponding to the frame's z coordinate in the same reference system. Encoding algorithms such as the one here described can be really useful for dimensionality reduction and data interpretability, but unluckily there is no way to systematically find such an algorithm for a given dataset, and this work must be completely carried out by humans.

For last, input dimensionality may be significantly reduced with a principal component analysis or PCA[32]. This analysis finds an orthonormal axis system whose axis are oriented in the directions which data variance is maximum. In many cases, data variance can be described up to a high percentage of its total value projecting it on a truncated PCA axis system. This happens when data is correlated in a way that effectively reduces "degrees of freedom". When this happens, n can be reduced (in some cases by large amounts) without losing critical information of the input data.

belongs to that category, and 0 otherwise. Therefore, for a categorical variable with n distinct categories, one-hot encoding will generate a vector of length n with a single value of 1 and the rest of the values as 0.

3.3 Train-test split

The aim of this section is to propose a consistent method to evaluate to which extent the dataset split fulfils the necessary conditions for the subsequent surrogate model \mathcal{F} to be appropriately trained on S^{train} and tested on S^{test} (vid. Figure 3.2). As has been previously discussed, this basically means S^{test} be in the applicability region of S^{train} , in order for the training-validation loop of the model to be coherent.

Typical figures for the train-test split ratio are 80%-20%. When the model being trained is very large, a third dataset (the validation set) may be needed for comparing different hyperparameter configurations, in which case the split is typically done at 60%-20%-20% for the train, test, and validation sets, respectively[15, pp. 20-21].

With the dataset split into the train, evaluation, and test sets, the standard training and validation loop is as follows: the model is trained to "fit" the data in the training set. After a certain amount of training, its performance is measured in the evaluation set. The test set is used to compare the performance of different hyperparameter configurations. Note that the NN is always evaluated with data not previously "seen" during training, and as such cannot develop any bias for the training set (although bias for the validation or test sets may exist).

For the case of MS-S18, we suppose we have a dataset which has been split into a training set

$$S^{\text{train}} = \{(\mathbf{x}^{\text{tr}}, \mathbf{y}^{\text{tr}})_i\}_{i=1}^{N_{\text{training}}}$$

and a test set

$$S^{\text{test}} = \{(\mathbf{x}^{\text{te}}, \mathbf{y}^{\text{te}})_i\}_{i=1}^{N_{\text{test}}},$$

3.3.1 Geometrical analysis

To address the applicability classification problem, points in the test set extremely far away from the training set (effectively outside applicability) need to be detected. If found, that would mean the train-test split is incorrect.

There is a wide range of criteria for classifying a point inside or outside applicability. The proposed validation loop makes such analysis following a hierarchical flow. For each point in the test set a decision is made based on where that test set point lies with respect to the training set.

Because categorical variables are usually related to different physical conditions, the first

step is to condition the analysis of each test point only with respect to the training subset of points whose categorical variables have exactly the same values than the test point under analysis. This conditioning on categorical variables defines, for each test point, a *voxel*, *i.e.* a region in the input space defined by the precise values of each categorical variable. At this point the first requirement comes into play: the number of training samples inside such voxel needs to be large enough (*e.g.* larger than zero).

Then, inside the voxel the ranges of each feature X_i of the training subset are computed, and then the maximum ranges are checked and defined. Their cartesian product defines the voxel's hypercuboid volume associated to the numerical features. It is checked whether the test point is inside or outside the hypercube. In practice, this amounts to checking if each numerical feature X_i of the test point lies inside the range spanned by the training subset. If all numerical features fall inside these ranges, the test set falls inside the training subset hypercuboid, otherwise, it falls outside. The points lying outside such hypercuboid are outside applicability and are flagged. The "hypercube requirement" is that such percentage of points outside applicability is zero, or very close to it.

If the test point under analysis is found to be outside the hypercuboid, the analysis fin-

Training – Test split

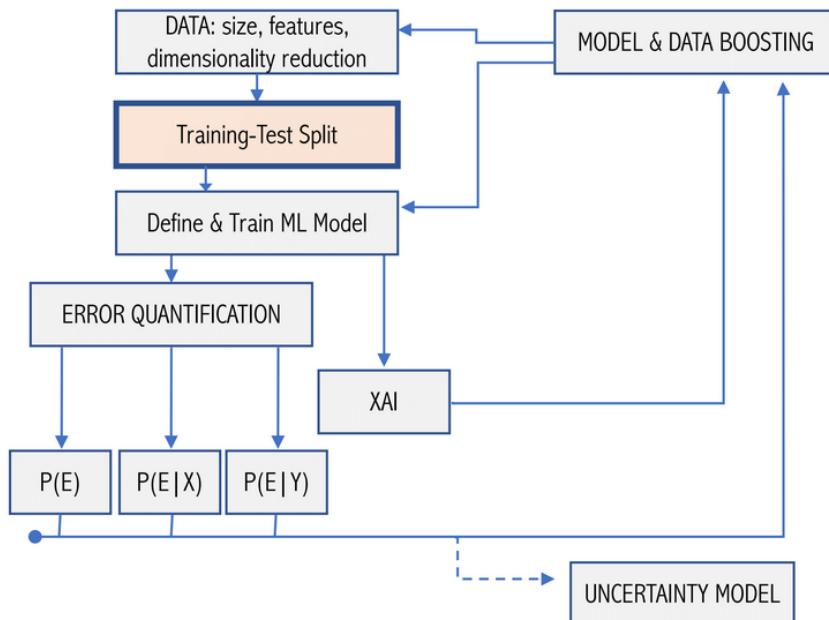


Figure 3.2: Location of the train-test split assessment in the overall validation pipeline

ishes and the loop moves to the next point. Otherwise, the second step is to check whether this point not only lies inside hypercuboid, but further, whether it also lies inside the convex hull spanned by the training subset in a PCA99³ projection. This hull has a smaller volume than the hypercuboid. If the test point under analysis is found to be outside the convex hull in the PCA99 projection, the analysis finishes and the loop moves to the next point.

Otherwise, then the third step is to check whether this point (not only lies inside hypercuboid and CH_{PCA99}, but also) lies inside the convex hull spanned by the training subset in ambient space. The volume of this hull is typically smaller than the one of the PCA99 projection by virtue of the curse of dimensionality, and if the test point lies inside this hull, interpolating properties of the surrogate model will suggest that the model is not require to generalize outside the region where it cannot generalize.

The step of the PCA projection is motivated by the curse of dimensionality, which makes every test point be in extrapolating regime with respect to the training data when the convex hull of the raw, unprocessed training data is used in a high-dimensional input space, as pointed out by [23]. The work of [25] effectively shows how in such cases, a low-dimensional space is more useful for applicability classification.

Each step provides different level of evidence of whether the point under analysis was adequately located, where the best is that it lies inside the convex hull of the corresponding voxel in ambient space, and the worst is that it lies outside the hypercuboid.

An additional check is to analyse whether p-hacking is happening, *i.e.* whether the test set point is "too close" to an actual training point (or indeed equal to a training point), what would falsely induce low test error of the model. For this, we measure the distance of the test point to the closest point of the training subset. This information is later used to assess different aspects of potential p-hacking and its impact on the decision of train-test split correctness.

The former procedure for classifying the applicability of test points is summarised in [algorithm 1](#).

Output of [algorithm 1](#) is depicted in [Table 3.1](#). This information has to be processed to complete [Table 3.4](#). The first column identifies the test point inside the test set. The next three columns show information about the corresponding voxel: existence, identification and size. Columns 5 to 7 show information related to pointwise distances inside the voxel. The last three columns show the relative position of the test point inside the voxel (inside the hyper-

³A truncated PCA projection in which the selected components explain at least 99% of the whole variance of the data.

Algorithm 1: Train-test split geometrical analysis

Data: $S^{\text{train}}, S^{\text{test}}$, voxel_size_req

Result: inside_hypercube, inside_PCA99, inside_ambient, avg_train_dist, min_test_dist

- 1 Initialize empty lists: test_voxels, train_voxels $\leftarrow []$;
- 2 Initialize boolean arrays:
inside_hypercube, inside_PCA99, inside_ambient $\leftarrow [\text{False}, \dots, \text{False}]_{1 \times \text{len}(S_{\text{test}})}$;
- 3 Initialize arrays for distances: avg_train_dist, min_test_dist $\leftarrow [\text{NaN}, \dots, \text{NaN}]_{1 \times \text{len}(S_{\text{test}})}$;
- 4 **foreach** $\mathbf{x} = (\mathbf{x}_{\text{numerical}}, \mathbf{x}_{\text{categorical}})$ in S^{test} **do**
- 5 Compute test_voxel by imposing $\mathbf{x}_{\text{categorical}}$;
- 6 Append test_voxel to test_voxels;
- 7 **end**
- 8 **foreach** $\mathbf{x} = (\mathbf{x}_{\text{numerical}}, \mathbf{x}_{\text{categorical}})$ in S^{train} **do**
- 9 Compute train_voxel by imposing $\mathbf{x}_{\text{categorical}}$;
- 10 Append train_voxel to train_voxels;
- 11 **end**
- 12 $i \leftarrow 0$;
- 13 **foreach** test_voxel in test_voxels **do**
- 14 Get train_voxel by imposing $\mathbf{x}_{\text{categorical}}$;
- 15 Compute main nearest neighbour distance inside train_voxel:
 $\bar{d}_{\text{train}}(\text{train_voxel}, \text{train_voxel})$;
- 16 avg_train_dist[i] $\leftarrow \bar{d}_{\text{train}}$;
- 17 **if** size(train_voxel) \geq voxel_size_req **then**
- 18 **foreach** \mathbf{x} in test_voxel **do**
- 19 Compute minimum nearest neighbour distance: $d_{\text{test}}(\mathbf{X}, \text{train_voxel})$;
- 20 min_test_dist[i] $\leftarrow d_{\text{test}}$;
- 21 **if** \mathbf{x} not in hypercube **then**
- 22 **break**;
- 23 **else**
- 24 inside_hypercube[i] $\leftarrow \text{True}$;
- 25 **if** \mathbf{x} not in CH_PCA99 **then**
- 26 **break**;
- 27 **else**
- 28 inside_PCA99[i] $\leftarrow \text{True}$;
- 29 **if** \mathbf{x} in ambient_hull **then**
- 30 inside_ambient[i] $\leftarrow \text{True}$;
- 31 **end**
- 32 **end**
- 33 *i* $\leftarrow i + 1$;
- 34 **end**
- 35 **end**
- 36 **end**

cube/convex hull in PCA99/convex hull in ambient space). In this algorithm, if the test point is found to be outside the hypercube (CH_{PCA99}), then its position with respect to CH_{PCA99} and $\text{CH}_{\text{ambient}}$ ($\text{CH}_{\text{ambient}}$) is not computed in the sake of computational efficiency.

To better understand how the voxels are computed, a reference of their categorical vari-

Table 3.1: Output of [algorithm 1](#).

	Vox. exists	Vox. ID	Vox. size	Min. test to train vox. dist.	Min. train to train vox. dist.	Avg. train to train vox. dist.	Inside vox. hypercube	Inside vox. CH (PCA99)	Inside vox. CH (ambient)
0	True	729	10	4.2222	2.3103	4.9535	False	NaN	NaN
1	True	703	10	6.8161	3.5068	6.2151	True	False	NaN
2	True	211	10	3.2955	1.6819	4.4763	False	NaN	NaN
3	True	799	8	NaN	NaN	NaN	NaN	NaN	NaN
4	True	531	6	NaN	NaN	NaN	NaN	NaN	NaN
...
1995	True	102	8	NaN	NaN	NaN	NaN	NaN	NaN
1996	True	536	11	3.3157	2.8551	4.4765	True	False	NaN
1997	True	770	6	NaN	NaN	NaN	NaN	NaN	NaN
1998	True	634	10	1.1911	1.8775	5.0745	True	False	NaN
1999	True	277	10	4.8021	3.1150	5.6874	True	False	NaN

2000 rows \times 9 columns

ables is given in [Table 3.2](#). Points inside each voxel all have the same categorical values, which effectively act as a unique identifier for the voxel. We see each voxel represents a different stringer section between two concrete frames, plus the categorical variable "np" which can take the values either 0 or 0.9.

Data in the column "inside vox. hypercube" of [Table 3.1](#) has been rearranged for the sake of interpretability in [Table 3.3](#). This table shows the points which lie outside the training voxel hypercube for every voxel where such points exist. Similar tables could be arranged with the number of points belonging to the hypercube but not to the convex hull in a PCA99 projection, or with those points belonging to the convex hull in a PCA99 projection but not in ambient space.

This information is used to check the following parameters and requirements, which are stored in [Table 3.4](#):

Table 3.2: Voxel reference table

Voxel
Voxel ID
0 (0.0, Fr68-Fr69, Str40)
1 (0.9, Fr66-Fr67, Str36p)
2 (0.0, Fr69-Fr70, Str29)
3 (0.9, Fr69-Fr70, Str35)
4 (0.0, Fr65-Fr66, Str01)
...
827 (0.9, Fr64-Fr65, Str21)
828 (0.9, Fr66-Fr67, Str32)
829 (0.0, Fr64-Fr65, Str26p)
830 (0.0, Fr64-Fr65, Str29p)
831 (0.0, Fr67-Fr68, Str07)

751 rows × 1 columns

- Whether there are enough training samples inside each training subset is compared with `voxel_size_req`.
- The percentage of test points inside the ranges of the training samples is compared with `hypercube_req`.
- The number of points inside the hypercube but outside `CH_PCA99` is compared with `CHMP_PCA99_negative_req`.
- The number of points inside the hypercube and `CH_PCA99` but outside `CH_ambient` is compared to the requirement `CHMP_ambient_negative_req`.
- The total number of points inside `CH_ambient` is compared to `CHMP_ambient_abs_req`.
- All these requirements are stored in [Table 3.4](#).

Having analysed the isolation level of test points, it is necessary to check whether p-hacking is taking place. That is, whether test points are unrealistically close to train points, thus making the model evaluation flawed. This is measured using the Mann-Whitney U test[33]. In this test, the initial hypothesis H0 is that both distributions are the same. H1 is that the distribution underlying test-training distances is stochastically less than distribution underlying training-training

Table 3.3: Points outside voxel hypercube

Voxel ID	Voxel	Num. of test points belonging to voxel	Num. of test points outside voxel's hypercube
729	(0.0, 'Fr66-Fr67', 'Str28')	2	2
703	(0.0, 'Fr65-Fr66', 'Str29')	3	2
211	(0.0, 'Fr66-Fr67', 'Str40')	4	3
219	(0.0, 'Fr64-Fr65', 'Str35')	4	3
413	(0.9, 'Fr68-Fr69', 'Str05p')	5	4
...
561	(0.9, 'Fr69-Fr70', 'Str43')	2	1
436	(0.0, 'Fr64-Fr65', 'Str40p')	1	1
276	(0.0, 'Fr64-Fr65', 'Str31')	1	1
109	(0.0, 'Fr69-Fr70', 'Str04')	2	1
446	(0.9, 'Fr69-Fr70', 'Str21')	1	1
104 rows × 3 columns			

Table 3.4: `reqs_results_table`. The first and the last requirements compare absolute figures of points at some isolation level with the overall number of points. Whereas the second and third requirements compare the difference in number of points between two consecutive levels with the total number of points.

	Desired rate (equal or less)	Obtained rate
Hypercube rate	0.1000	0.5780
CHMP PCA99 negative rate	0.1000	0.3880
CHMP ambient negative rate	0.1000	0.0340
CHMP ambient abs rate	0.1000	0.3880

distances, which would involve risk of p-hacking. H_0 is rejected only with a 95% confidence. For instance, for the dataset of MS-18, the resulting p-value is $p = 0.9793$, thus the null hypothesis is not rejected and the dataset can be assumed to be free of p-hacking.

3.3.1.1 Non-geometrical analysis

To help (the engineer in charge) make a decision about the dataset split, another approach is developed, complementary to the geometrical analysis carried out so far. This approach consists of focusing on the statistical distributions of the train and the test datasets. A proper train-test split is characterised by similar statistical distributions of both sets (test and training)[25]. When this cannot be achieved, a softer requirement is that each of the feature variables X_i need to have reasonably similar marginal distributions in the training and the test set.

The following analysis checks whether, for each individual input feature, the distribution of the training and test set is reasonably similar. This is also checked for the output variables (the ground true ones, thus this is independent from the surrogate model). Finally, the splitting can be done locally, *i.e.* region by region, by binning each numerical variable. The implementation of this analysis tackles each input variable x_i independently and for each input variable, it compares the training and the test set such that:

- If the variable x_i is ‘categorical’: both (train and test) categorical frequency distributions are plotted (Figure 3.4), a 2-sample χ^2 test[34, p. 431] is performed, and the p-value of such test is introduced in Table 3.5.
- If the variable x_i is numerical: both (train and test) numerical frequency distributions are plotted (Figure 3.3), a 2-sample Kolmogorov-Smirnov test[34, p. 454] is carried out, and the p-value of such test is introduced in the same table.

The null hypothesis for both tests is that both train and test distributions are the same. H_0 is only rejected with a 95% confidence.

As stated earlier, applicability is not only a matter of the *input* space, but also of the *output* space. The 2-sample Kolmogorov-Smirnov test has been performed again on each of the six output variables. The p-values and their statistical distributions are depicted in Table 3.6 and Figure 3.5, respectively.

Table 3.5: P-values of the input variables. The test performed is a 2-sample Kolmogorov-Smirnov or a 2-sample χ^2 test, depending on the data being numerical (first 19 rows) or categorical ("dp", "Frame" and "Stringer").

p-values	
	p-value
factors	0.97409
FU.0410.14	0.08896
FU.0410.15	0.77249
FU.0410.16	0.53206
FU.0410.24	0.31840
FU.0410.25	0.54828
FU.0410.26	0.58953
FU.0420.14	0.19180
FU.0420.15	0.50018
FU.0420.16	0.58953
FU.0420.24	0.54015
FU.0420.25	0.34948
FU.0420.26	0.74838
FU.0430.14	0.16084
FU.0430.15	0.17195
FU.0430.16	0.51601
FU.0430.24	0.55646
FU.0430.25	0.42458
FU.0430.26	0.66518
dp	0.99969
Frame	1.00000
Stringer	1.00000

3.3.1.2 Train-test split. Conclusions

The proposed validation loop for the train-test split focuses, on the one hand, on the applicability classification problem, which makes use of a geometrical analysis which classifies test points at different levels of isolation. A series of requirements are defined based on the proportions of such isolated points. Plus, p-hacking is checked. The output of this analysis is showed in Figure 3.6.

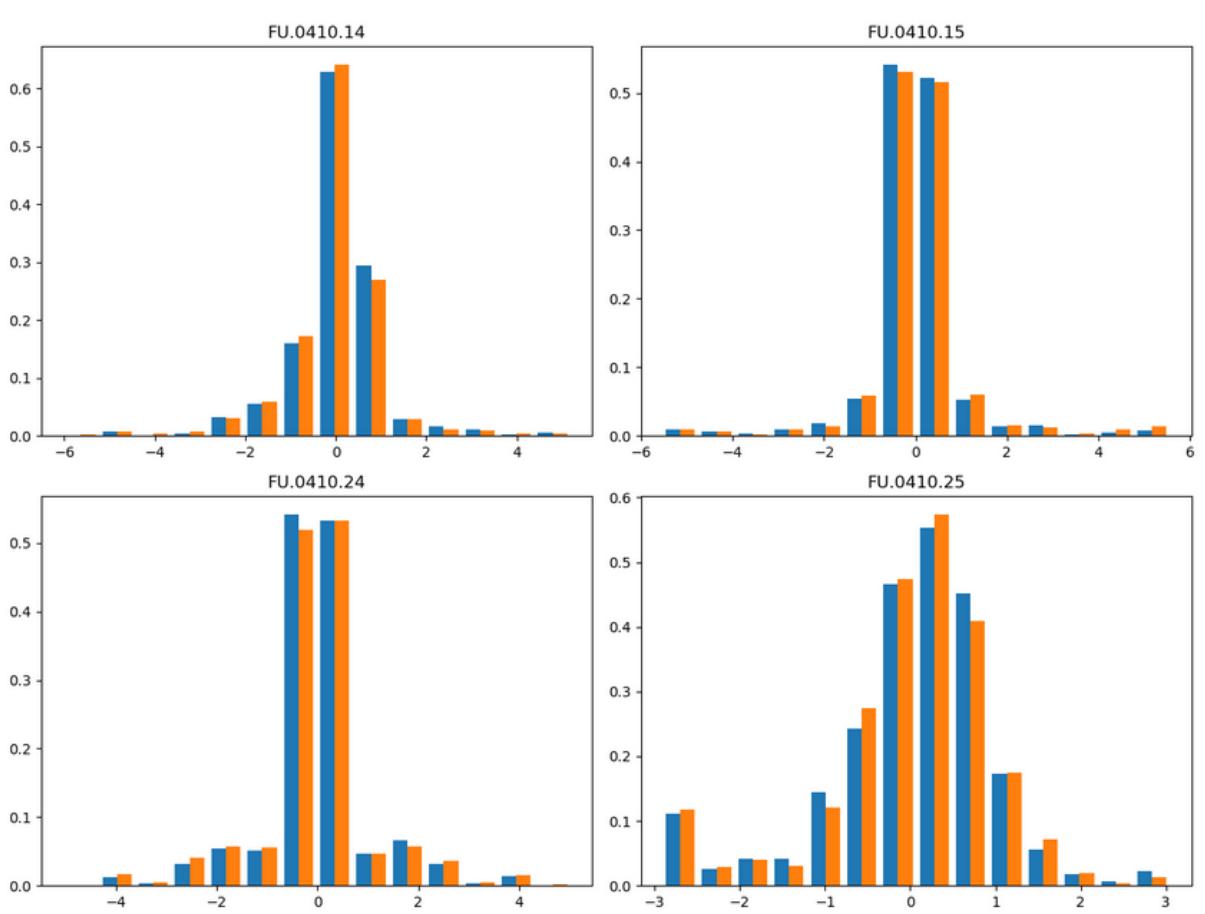


Figure 3.3: Input [numeric] variables distributions double histogram. Only the first four input variables have been plotted.

Table 3.6: P-values of the output variables distributions

p-values	
	KS
RF Forced Crippling	1.00000
RF Column Buckling	1.00000
RF In Plane	1.00000
RF Net Tension	0.24668
RF Pure Compression	1.00000
RF Shear Panel Failure	0.99995

On the other hand, the distributions of both input and output variables in the training set are checked to be reasonably similar to that of the test set. The main outputs are the p-values shown at Table 3.5 and Table 3.6.

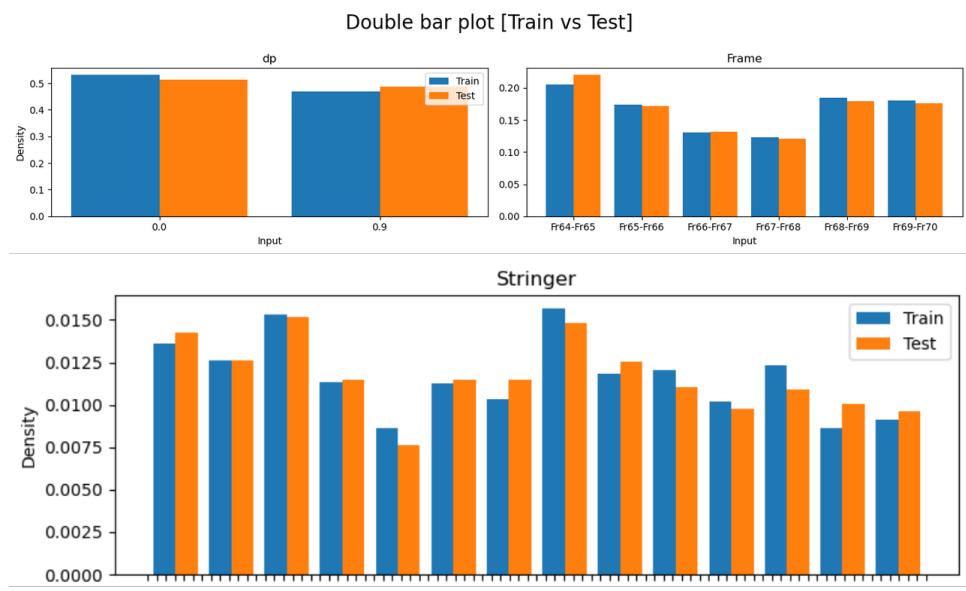


Figure 3.4: Input categorical variables distributions double histograms

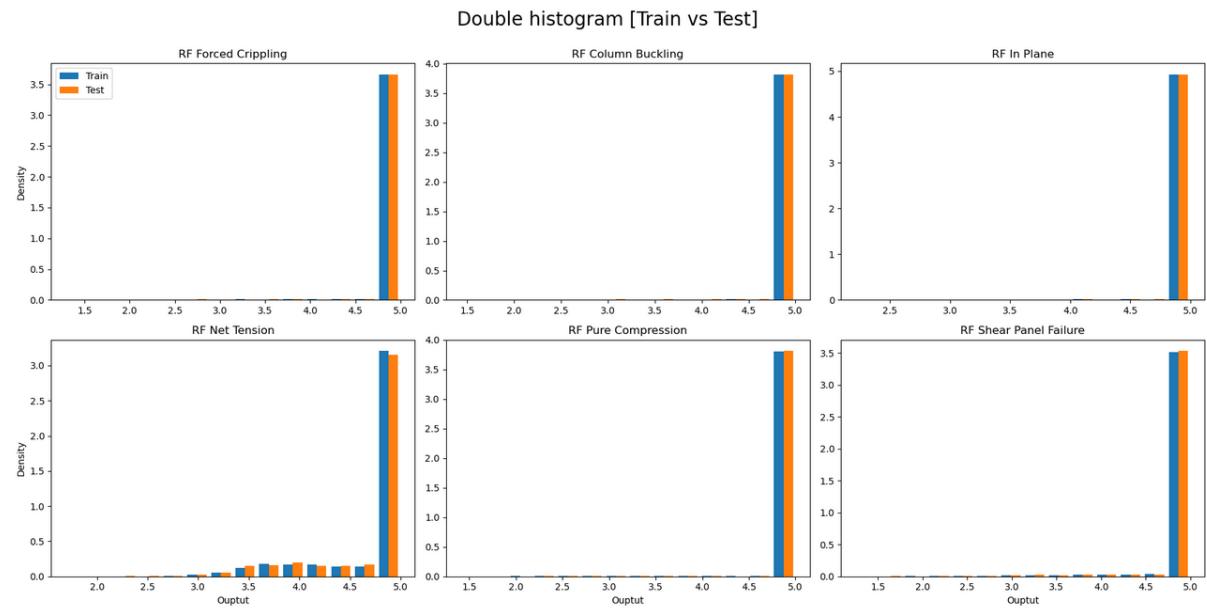


Figure 3.5: Output variables distributions double histograms

Of course, any given dataset may meet some of the requirements, but not all (as is the case with the MS-S18 dataset). The engineer in charge should evaluate the ensemble output and decide whether it is necessary to redo the split completely or partially, depending on data availability.

Requirement check		
Type	Check	
Variable Compatibility Check	Optional	True
Voxel Size	Optional	None
Voxel Hypercube Rate	Optional	False
CHMP PCA99 (negative)	Optional	False
CHMP ambient (negative)	Optional	True
Mann Whitney U test pvalue	Optional	True

Figure 3.6: Requirements of the geometrical analysis

3.4 Global error quantification

This section corresponds to box labelled "Error quantification" in Figure 3.1.

After running the model, the simplest analysis of its performance consists of measuring the aggregated error of predictions against ground true values over the whole test set. Different metrics and criteria can be adopted for this task. Common error measures are the Mean Absolute Error or MAE (cfr. eq. (1.15)) and the Root Mean Square Error or RMSE (which is just the root square of eq. (1.14)). These metrics account for the distance between points representing ground true values (\mathbf{y}_t) and model's predictions (\mathbf{y}), taken as the L_1 norm (MAE) or the square of the L_2 norm (RMSE) of the distances. In many industrial applications of machine learning, difference between ground true and predicted values can be more or less important depending on whether that difference is due to *overestimating* or *underestimating*. Take, for instance, the case of MS-S18 model, whose predictions are a measure of the probability of failure of aeronautical structural components. Clearly, underestimating the risk is much more dangerous than overestimating it. For cases such as this one, a useful measure of the error is the **residue**. The residual error of a given point i is measured as

$$\mathbf{e}(i) = \mathbf{y}_t(i) - \mathbf{y}(i) \quad (3.1)$$

The drawback of the residual error is that, when using it as an aggregated indicator for the whole test set, residues can cancel out. A null MAE or RMSE account for a perfectly fitted model (ground true values and predictions are equal). That is not the case for the residual error. In the following sections, unless specifically specified, the metric of choice for measuring the error is the residue (Equation 3.1), although different error metrics are also supported from an implementation point of view, one of the most common alternatives being the absolute value of the residue, $|\mathbf{e}|(i) = |\mathbf{y}_t(i) - \mathbf{y}(i)|$.

An interesting scalar metric for the global performance of the model is the coefficient of determination R^2 [35] of the scatter distribution of \mathbf{y}_t vs \mathbf{y} . R^2 is a measure of the goodness of fit of predicted to ground true values. Illustration of this is provided in Figure 3.7. The most important conclusion of Figure 3.7 is that the error shows to be heteroscedastic. This means that the variance of the error is not constant along some variable range (in this case, that variable is the output variable named "RF Net Tension"). In Figure 3.7 we can clearly see that dispersion grows with the output value. Cfr.[36, p. 374]. This is shown by the fact that dispersion is

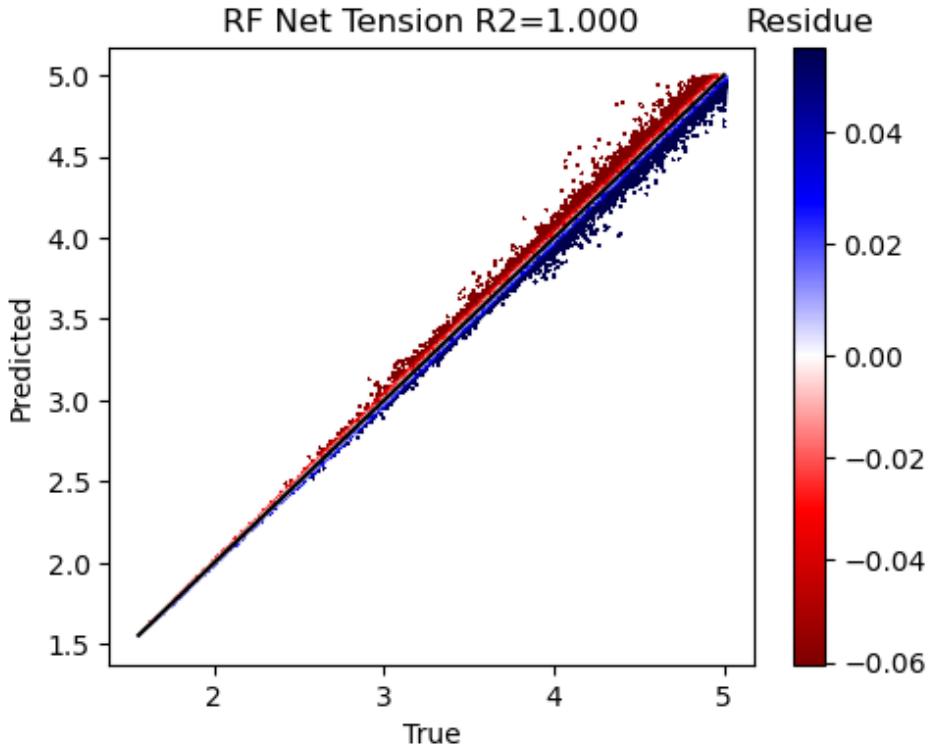


Figure 3.7: Scatter plot of ground true (x axis) against predicted values (y axis) of output variable "RF Net Tension", with the correspondent R^2 coefficient. In this case, $R^2 = 1,000$ indicates a perfect fit of predicted to ground true values. Mismatches due to underestimating failure risk are labelled in red, while those due to overestimating failure risk are labelled in blue. Similar graphs can be computed for every pair $\{y_j, \hat{y}_j\}$ of features in the output variables $\mathbf{y}_t = \{y_{t1}, y_{t2}, \dots, y_{tm}\}$. Plots for the rest of the six output variables of MS-S18 show analogous results.

higher for higher Reserve Factors (interestingly, this is the desired behaviour, as lower Reserve Factors implicate higher failure risk, and thus precision is more important in the region of low Reserve Factors). This property makes it necessary to study the error distribution, as well as the error distribution conditioned on input and output space ($P(e)$, $P(e|X)$ and $P(e|Y)$, resp.) as is discussed in the following sections.

3.5 Distributed error quantification.

This section corresponds to box labelled " $P(E)$ " in Figure 3.8. From now on, the terms "error" and "residue" (as defined in the previous section) will be used as synonyms.

Beyond global error statistics, the probability density function or PDF of the residue,

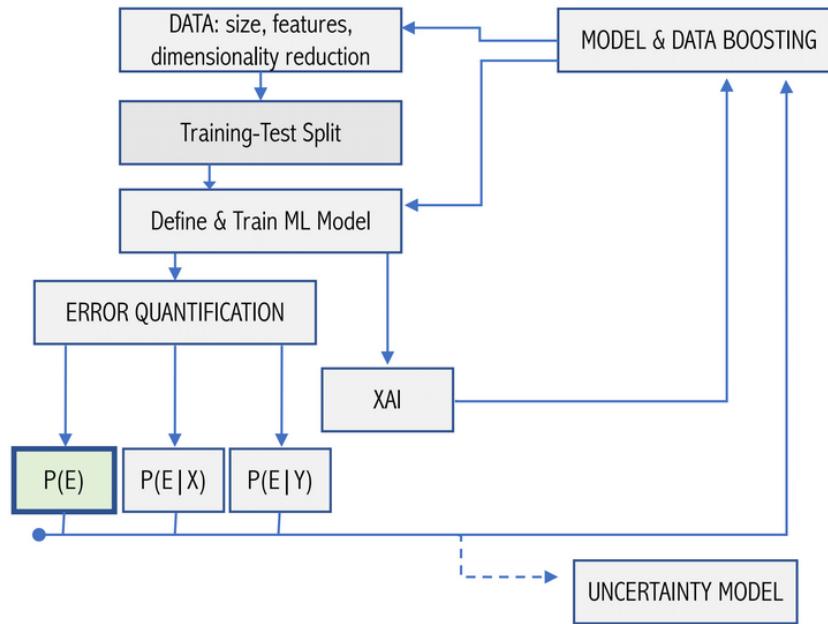


Figure 3.8: Location of section 3.4 in the validation pipeline.

$P(e)$, is of great importance for validation purposes. The main goal of this section is finding the analytic expression of $P(e)$ and computing important statistics of it, and, when the first is not possible, computing the important statistics from the empirical distribution of the residue (sampled from test set points) with the method of bootstrapping⁴. There are three main reasons for studying $P(e)$:

A. Non gaussian error distribution

It is common in industry-applied problems to find situations in which the error between ground-truth values and predictions coming from a regression model (let it be a surrogate ANN, some parametric regressor, etc.) which is supposed to fit any function describing a complex sys-

⁴Vid. following paragraphs

tem follows absolutely non-Gaussian distributions (see *e.g.* [37–40]). Amongst the reasons for this, the most frequent are, on the one hand, non-homogeneous data sampling in the training set (leading to uncovered regions and isolated points) which can cause poor model performance due to non-interpolation-regime operation, and on the other hand, the inherent difficulty encountered at predicting outputs for specific input configurations due to strongly non-linear physics or governing equations (mathematically this manifests in the form of high gradients). When the error is non-Gaussian, concentration statistics such as MAE or RMSE stop being informative. In such case, a comprehensive analysis on $P(e)$ is more adequate.

B. Outlier detection

Outliers are strange events in a population sampled from a known PDF, in the sense that it is not expected to find them, or that their position is far away from expected. Outlier detection helps identifying strange phenomena which the engineer in charge could decide to investigate. Imagine, for instance, that certain residue e_x was systematically sampled from the test set with an unusual frequency, compared with similar values. In this case, it would be necessary to assess whether this high frequency is statistically expectable from the residue's PDF or not. If e_x was found to be an outlier, determining the underlying reason triggering such high frequency would help with model boosting.

Outlier detection relies on knowing $P(e)$. The probability of finding a residue larger than a given magnitude x is measured as $p_{>x} = \int_x^\infty P(e) de$. If we find some x for which $p_{>x} << 1$, all samples of the residue $e > x$ would be classified as outliers. This simple idea lies behind standard outlier-detection methods such as the z-score and the gESD (which are later discussed).

C. Uncertainty measuring

The marginalised distribution of $P(e)$ is the first step in the journey towards building an **uncertainty model**. This is the ultimate milestone of the whole validation pipeline, since it provides precise information about *how much* and *when* the model's predictions are trustworthy. This is the whole point of [section 3.8](#). The uncertainty model relies on the marginalised distribution of the residue for building confidence intervals which embed the model's error with a given statistical confidence level.

A simple plot can help us have a first intuition for the cause of subsequent results presented in this section. In [Figure 3.9](#) ground true values and model's predictions are represented

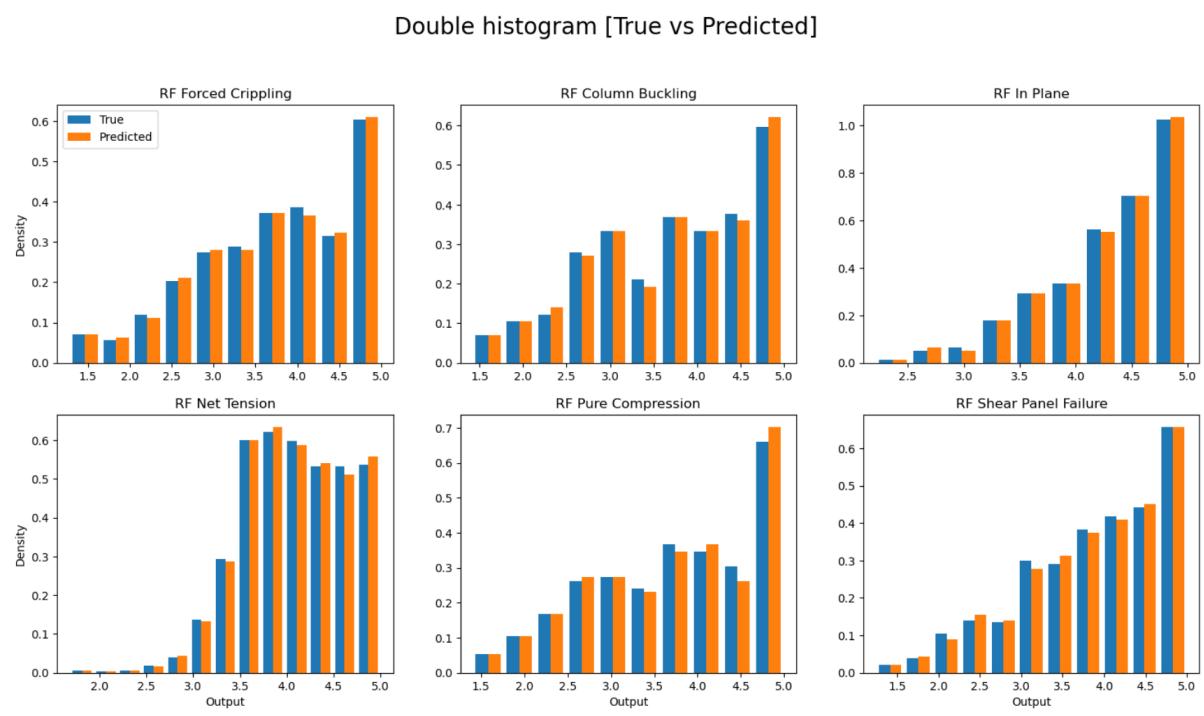


Figure 3.9: Double histogram depicting ground true values and model's predictions for the six output variables of MS-18.

in a double histogram for MS-18.

If the model accurately predicts results along the whole output variables range (or equivalently it is not heteroscedastic), we would expect both the true and the predicted results to come from the same (unknown) distribution (this statement is taken as the null hypothesis H_0). We can test this with a (2 sample) goodness-of-fit test like the Kolmogorov-Smirnov. Once again, the null hypothesis is rejected only at a 95% confidence level (*i.e.* if the p-value derived from the K-S test is lower than 0.05). The corresponding p-values of the K-S test for the six output variables of MS-18 are depicted in Table 3.7.

In Table 3.7 we can see how the K-S test rejects the null hypothesis in some cases despite distributions in Figure 3.9 looking very similar. This is due to the K-S test sensitivity to the size of datasets.

As it has been previously mentioned, outlier detection and uncertainty models both rely on the PDF of the residue, $P(e)$. The main goal of this section is finding the analytical definition of $P(e)$ ⁵, and measuring important statistics of it. This is addressed with a focus similar to that followed in Figure 3.9 and Table 3.7, but instead of assessing the fitness of the predictions (\hat{y})

⁵This might not always be possible. When it is not, non-parametrical bootstrapping is given as an alternate solution (vid. next paragraphs).

Table 3.7: Resulting p -values from the 2-sample K-S test of goodness of fit between ground true (\mathbf{y}_t) and predicted (\mathbf{y}) distributions. The null hypothesis H_0 that both distributions conform is rejected if the p -value is smaller than 0.05.

Hypothesis Tests Results (p-value)	
	KS
RF Forced Crippling	0.03763
RF Column Buckling	0.11906
RF In Plane	0.03428
RF Net Tension	0.60624
RF Pure Compression	0.08367
RF Shear Panel Failure	0.03510

distribution to the ground true values (y) distribution, we try to assess the goodness of fit of the empirical residue distribution to some well-known distributions. For reference, the (empirical) error distribution of the six output variables is depicted in Figure 3.10, as well as the corresponding cumulative distributions, given in Figure 3.11.

Under the assumption that $P(e)$ can be described by some well-known parametric distribution, we use the 1-sample K-S test (coupled with a minimal-squares based optimizer to find the optimal set of parameters for each distribution) to compare distributions of Figure 3.10 to the following distributions:

- Normal:

$$f(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- Laplace:

$$f(x|\mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$$

- Cauchy:

$$f(x|x_0, \gamma) = \frac{1}{\pi\gamma \left[1 + \left(\frac{x-x_0}{\gamma}\right)^2\right]}$$

- JohnsonSU:

$$f(x|\gamma, \delta, \lambda, \xi) = \frac{\delta}{\gamma\sqrt{2\pi}} \frac{1}{x+\xi} \exp\left(-\frac{1}{2}(\gamma + \lambda \log(x+\xi))^2\right)$$

P-values from the K-S test are given in Table 3.8. As we can see, the normal distribution does

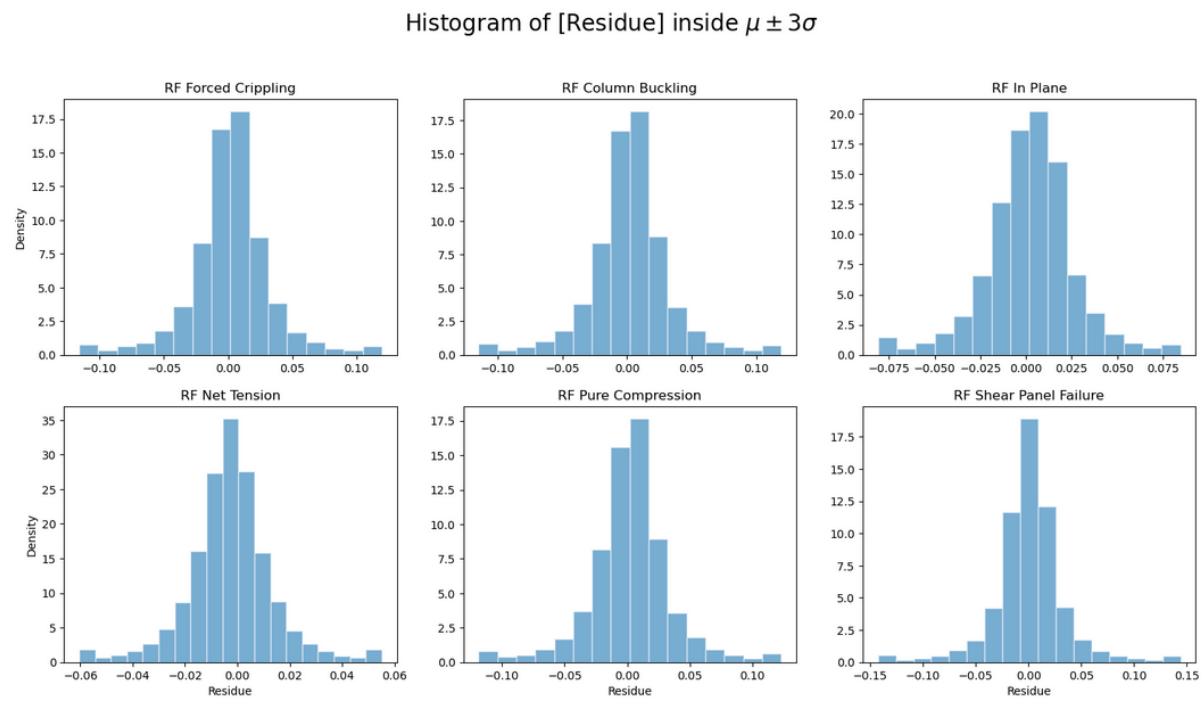


Figure 3.10: Empirical residue distribution sampled from the test set, for each of the six output variables of MS-S18. x -axis limits have been truncated to $\mu \pm 3\sigma$, where the most part of the error lies. Histograms have been appropriately binned for a correct visualization.

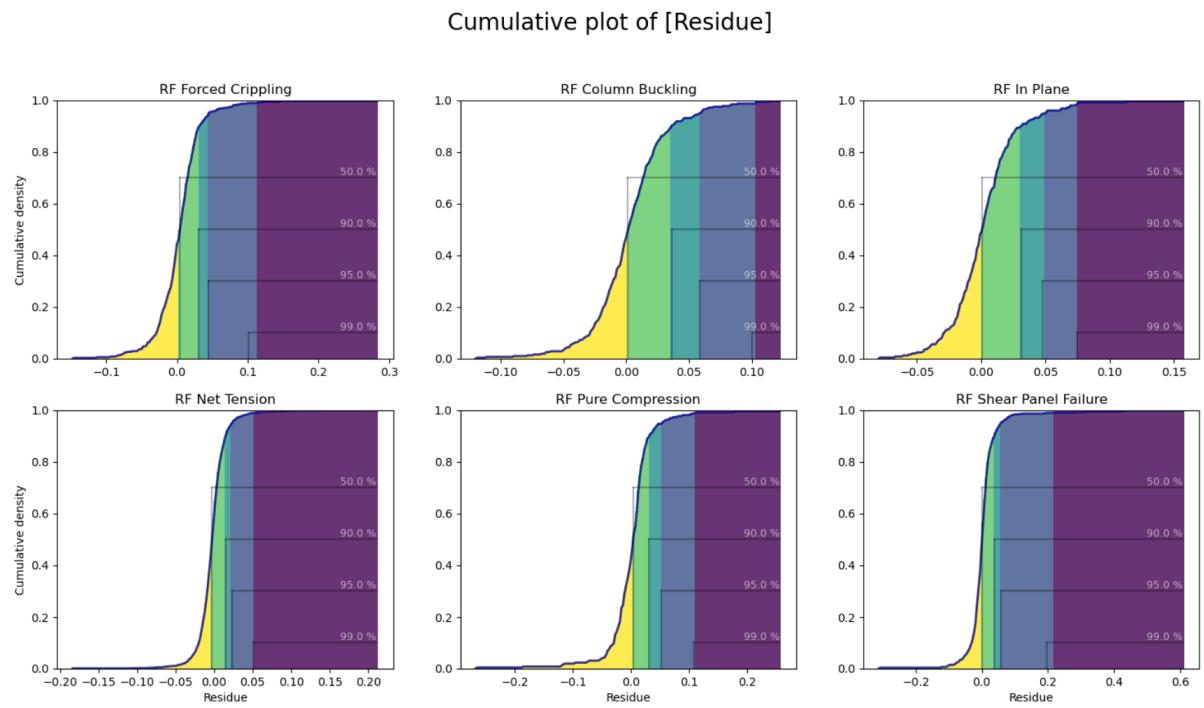


Figure 3.11: Cumulative error distributions corresponding to the PDFs showed (as binned histograms) in Figure 3.10.

not fit any of the output variables' error. While Laplace and Cauchy distributions' p-values from the K-S test are above the 0.05 threshold in five of the six output variables, they are well below the obtained p-values for the JohnsonSU[41] distribution (vid. Figure 3.12). In fact, when augmenting the dataset size from the illustrative-sized employed here (10,000 items) to a more realistic 800,000 items, neither of Laplace nor Cauchy distributions passes the test (their p-values drop to near-zero orders of magnitude). This happens due to the K-S sensibility to the size of data, which makes the test more strict when the datasets are large (as would be expected). We conclude that the Laplace and Cauchy distributions just pass the K-S anecdotally for the unrealistically small dataset size which has been used for illustrative purposes, and we also conclude that the only theoretical distribution (of the list which has been checked) that fits the MS-18's error distribution is the JohnsonSU.

Provided that there exists a simple transformation of the JohnsonSU distribution's random variable (vid. Figure 3.13) that converges to a normal distribution, one can, with the information obtained from the K-S test (that is, assuming the error data comes from a JohnsonSU distribution) apply standard outlier detection methods to a transformed variable $z \sim \mathcal{N}(0, 1)$, like the z-score (every point located outside $\mu \pm 3\sigma$ is considered to be an outlier) and the generalised Extreme Studentized Deviate[42] (gESD), thus fulfilling the aims described at the beginning of this section concerning outlier detection (vid. Table 3.9).

Results shown in Table 3.7 rise some concerns about the method followed until now.

Table 3.8: P-values of the K-S test comparing the empirical sample of $P(e)$ to the theoretical distributions indicated in each column. The null hypothesis H_0 (the empirical distribution has been sampled from the one figuring in a given column) is rejected when $p - \text{value} < 0.05$.

	Fit to each output-metric			
	Norm	Laplace	Cauchy	Johnsonsu
RF Forced Crippling	0.004	0.138	0.138	0.613
RF Column Buckling	0.043	0.328	0.435	0.954
RF In Plane	0.005	0.691	0.691	0.616
RF Net Tension	0.000	0.060	0.001	0.843
RF Pure Compression	0.001	0.053	0.438	0.791
RF Shear Panel Failure	0.000	0.002	0.453	0.716

The immediate concern that arises is what would happen if we were unable to find a theoretical distribution that fits some variable's error distribution with a statistically significant

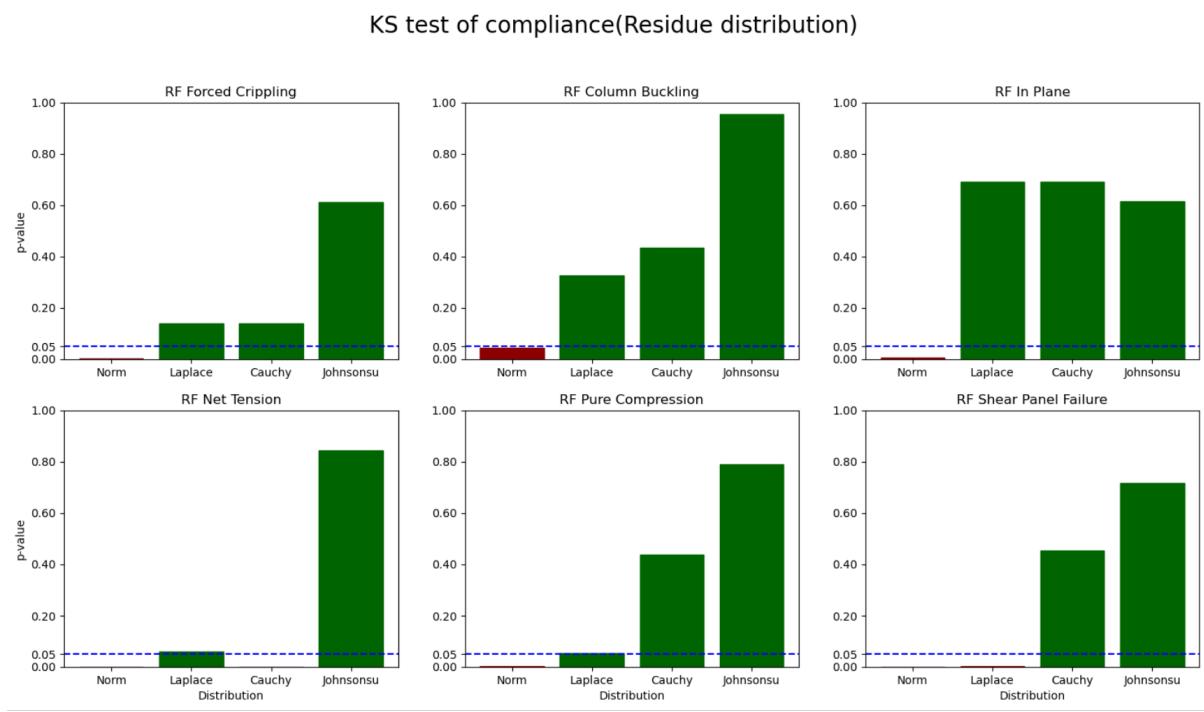


Figure 3.12: Graphical comparison of the p-values resulting from the K-S test for the MS-18 data.

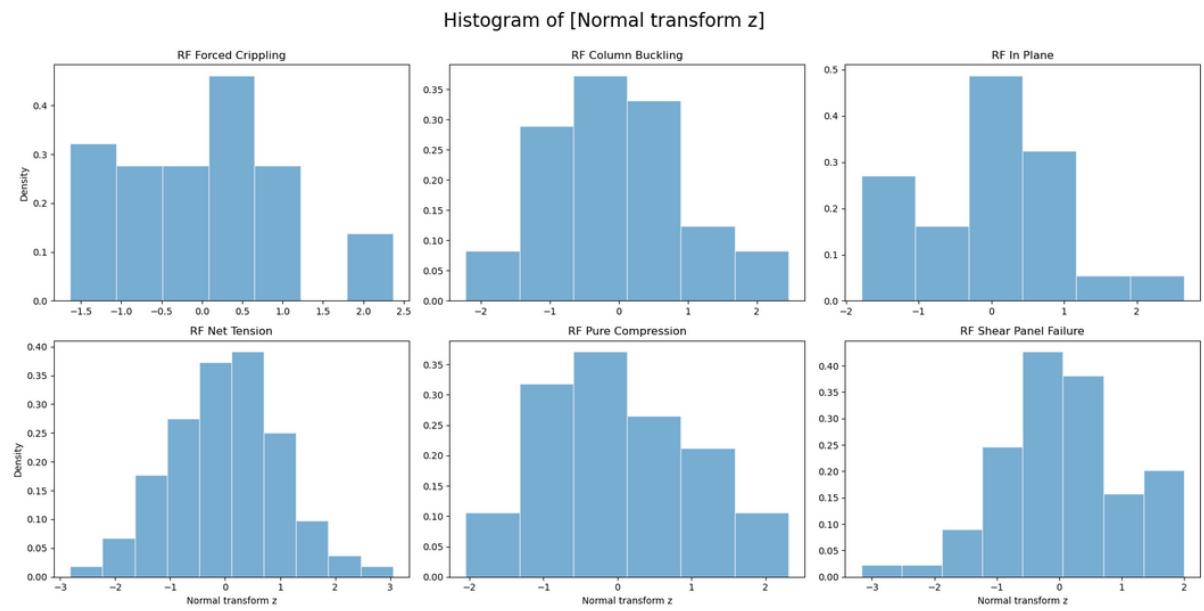


Figure 3.13: (Binned) histograms depicting the distribution of the variable $z = \gamma + \sinh \frac{e-\xi}{\lambda}$, where $e \sim JohnsonSU(\gamma, \delta, \xi, \lambda)$ is the sampled residue. It can be shown that z converges to a normal distribution[41].

confidence level. In fact, the most common statistical distributions (normal, Laplace, Cauchy) do not properly fit the MS-18 error distribution (when using an industrial-sized dataset, not

Table 3.9: Outlier detection taking $e \sim JohnsonSU$ as H_0 . N.B. for this table the usual requirements for classifying a point as an outlier ($z \in \sigma \pm 3\mu$ for the z-score and significance $1 - a \geq 95\%$ for the gESD) have been softened here for illustration purposes to $z \in \sigma \pm 1.2\mu$ and $a = 0.45$ for both tests, respectively.

	Outliers assuming JohnsonSU distribution					
	RF Forced Crippling	RF Column Buckling	RF In Plane	RF Net Tension	RF Pure Compression	RF Shear Panel Failure
st.normaltest pvalue	0.64	0.86	0.44	0.76	0.95	0.44
Total number of cases	38	31	25	279	26	69
Outliers outside 1.5 stds	5 / 13.16%	4 / 12.90%	3 / 12.00%	36 / 12.90%	4 / 15.38%	8 / 11.59%
Lower bound z=-1.5	-0.04	-0.04	-0.01	-0.03	-0.07	-0.04
Upper bound z=1.5	0.04	0.04	0.03	0.02	0.06	0.04
GESD outliers (a=0.45)	0-25 / 65.79%	0-1 / 3.23%	0-1 / 4.00%	0-270 / 96.77%	0-18 / 69.23%	0-1 / 1.45%
GESD lower bound	0.00	-0.09	-0.02	0.00	0.01	-0.10
GESD upper bound	0.01	0.04	0.04	0.00	0.02	0.07

the one employed for illustration here), and we've had to rely on the (rather exotic) JohnsonSU distribution. Without a parametrized distribution that properly fits the error, an uncertainty model cannot be computed. Recall building an accurate uncertainty model is the main motivation for this section. Computing empirical statistics of the empirical distribution of the residue (obtained from test set points) is possible, but assuming the empirical statistics are the same than the true, theoretical ones is not possible. To solve this obstacle, an alternative method for calculating informative statistics of the error's distribution that do not rely on knowing the parametrized analytic expression of it is therefore needed. The method provided here is known as non-parametric bootstrapping[43] and the main concept behind it is showed in [algorithm 2](#). This algorithm:

1. Samples N points with replacement from the original population S . N.B. replacement makes the new and the original populations (possibly) different.
2. Statistic x is calculated in the new population.

3. Steps 1 and 2 are repeated $M \gg 1$ times, giving a collection of x 's (called X).
4. If M is sufficiently large, X converges to a Gaussian population. The bootstrapped CI for the statistic x with a 95% confidence is bounded by the percentiles 2.5% and 97.5% of X .

Algorithm 2: Non-parametric bootstrapping

Data: Population $S = \{S_1, S_2, \dots, S_N\}$ with unknown PDF.

Result: Statistic x 's CI

```

1 Initial ize list:  $CI = [0]_{1 \times 2};$ 
2 Initialize list:  $X = [0]_{1 \times M};$ 
3 for  $i = 1, \dots, M \gg 1$  do
4    $S_i \leftarrow C_S(N, N);$ 
5    $X(i) \leftarrow x_{S_i};$ 
6 end
7  $CI(1) \leftarrow P_X^{2.5\%};$ 
8  $CI(2) \leftarrow P_X^{97.5\%},$ 
```

Some informative statistics of the error distribution are given in [Table 3.10](#). The statistics are computed twice, once in the empirical distribution of $P(e)$ sampled from the outputs of $\mathcal{S}^{\text{test}}$, and the other one in the form of bootstrapped CIs.

To better understand the utility of $P(e)$ for building an uncertainty model, the simple idea behind these models is presented here, although it is further discussed in [section 3.8](#).

In [Table 3.11](#), some quantiles of the error distribution are presented. For reference, they are computed as empirical statistics of the empirical error distribution, and using bootstrapping (in this case their confidence intervals are given instead). The simplest uncertainty model which can be built with this information assumes that, for future samples of the residue, the quantiles of [Table 3.11](#) will still hold true (*i.e.*, the empirical and the theoretical quantiles coincide). For instance, we would assume that, according to [Table 3.11](#), for future samples the error of variable "RF Forced Crippling" will belong to the interval $[-0.0432, 0.0448]$ (defined by percentiles 5th and 95th) with a frequency equal to 90%. If we wanted to soften the assumption that the empirical and the theoretical quantiles are the same, we could use the bootstrapped quantiles instead. That way, with a 95% confidence we could claim that the error of "RF Forced Crippling" variable will belong to $[-0.0548, 0.0586]$ with a frequency of 90% as a minimum, given that we now from bootstrapping that, with a 95% confidence, the true 5th quantile belongs to the range $[-0.0548, -0.0353]$ and the true 95th quantile belongs to $[0.0379, 0.0586]$.

Of course, the uncertainty model described in the last paragraph can be fine-tuned using additional information about the error distribution. If we found $P(e)$ to be heteroscedastic, we could benefit from conditioning our uncertainty model to certain regions of the input (or the output) space. This is the main motivation for section 3.6, section 3.7 and section 3.8.

Table 3.10: Summary of bootstrapped error statistics. For the median, the Wilson-score[44] is used for computing the confidence interval.

Confidence level at 95%. BS=Confidence Interval (percentile bootstrap), WS=CI (wilson-score)										
	count	min	mean	(BS) median	(WS) median	std (BS) std	IQR (BS) IQR	kurtosis (BS)	skewness (BS)	max skewness
RF Forced Crippling	383	-0.15	0.0032	0.0062	0.0041	0.0068	0.034	0.039	0.035	14. 24. 2.7 1.4 3.1 -0.74 0.28
RF Column Buckling	319	-0.12	0.0027	0.0061	0.0014	0.0047	0.031	0.034	0.035	2.7 3.8 1.6 0.22 0.81 -0.33 0.12
RF In Plane	252	-0.08	0.0020	0.0059	0.00065	0.0028	0.027	0.031	0.030	5.0 8.3 0.84 1.1 1.9 0.13 0.16
RF Net Tension	2794	-0.18	-0.0026	-0.0019	-0.0028	-0.0022	0.018	0.02	0.016	17. 25. 6.2 0.72 2.0 -0.75 0.21
RF Pure Compression	265	-0.27	0.0019	0.0073	0.0040	0.0077	0.041	0.052	0.029	0.034 0.025 15. 21. 5.1 -0.31 2.4 -2.9 0.26
RF Shear Panel Failure	695	-0.31	0.0038	0.0081	0.00073	0.0026	0.053	0.067	0.028	0.031 0.026 44. 63. 20. 4.0 5.9 0.037 0.61

Table 3.11: Bootstrapped percentiles (1st, 5th, 10th, 90th, 95th and 99th) of the residue distribution, calculated with a 95% confidence interval using Wilson-score.

Confidence level at 95%. BS=Confidence Interval (percentile bootstrap), WS=CI (wilson-score)														
	1%	(WS) 1%	5%	(WS) 5%	10%	(WS) 10%	median	(WS) median	90%	(WS) 90%	95%	(WS) 95%	99%	(WS) 99%
RF Forced Crippling	-0.0834	-0.0712	-0.0432	-0.0353	-0.0286	-0.0253	0.00405	0.00676	0.0309	0.0389	0.0448	0.0586	0.101	0.147
	-0.115	-0.0548	-0.0361	-0.0286	-0.0246	-0.0246		0.0000785		0.027		0.0379		0.0768
RF Column Buckling	-0.0782	-0.0605	-0.0441	-0.0337	-0.0283	-0.0373	0.00139	0.00472	0.0362	0.0453	0.0582	0.0656	0.100	0.113
	-0.111	-0.0509	-0.0449	-0.0306	-0.0284	-0.0203		0.000646		-0.00138		0.0442		0.0686
RF In Plane	-0.0586	-0.0449	-0.0386	-0.0449	-0.036	-0.0337		0.00281	0.0310	0.0418	0.0481	0.0664	0.0747	0.158
	-0.0798	-0.0449	-0.0386	-0.0449	-0.036	-0.0337		0.00220		-0.00220		0.0233		0.0664
RF Net Tension	-0.0506	-0.0444	-0.0277	-0.0261	-0.0198	-0.0187		-0.00280		-0.00224		0.0158		0.0596
	-0.0633	-0.0633	-0.0277	-0.0298	-0.0213	-0.0213				-0.00329		0.0135		0.0437
RF Pure Compression	-0.118	-0.0549	-0.0428	-0.0352	-0.0289	-0.0244		0.00400		0.00772		0.0407		0.259
	-0.267	-0.074	-0.074	-0.074	-0.0389	-0.0389				0.00124		0.0314		0.0683
RF Shear Panel Failure	-0.0994	-0.0775	-0.0498	-0.0392	-0.0307	-0.0258		0.000728		0.00259		0.0376		0.316
	-0.123	-0.0578	-0.0578	-0.0578	-0.0361	-0.0361				-0.00108		0.0299		0.0902

3.6 Distributed error quantification: conditioning the error distribution on the input space

This section corresponds to box labelled " $P(E|X)$ " in Figure 3.14. In section 3.5, the marginalised distribution of the residue $P(e)$ was discussed. This section aims at extracting useful information by conditioning the error distribution to the input variables' space, which mathematically can be denoted by $P(e|X)$. Amongst the motivations for this, we count in the first place that this section will help building our uncertainty model (vid. section 3.8) by refining the methods described in the previous section for uncertainty prediction with new information from $P(e|X)$. Furthermore, model and data boosting (vid. Figure 3.14) can both benefit from the analysis performed in this section. Knowledge of the error distribution conditioned to the input space can help identify those regions where either:

- The dataset is properly representing the region, but model training is deficient.
- Dataset quality is not sufficient for proper training according to the specified performance requirements.

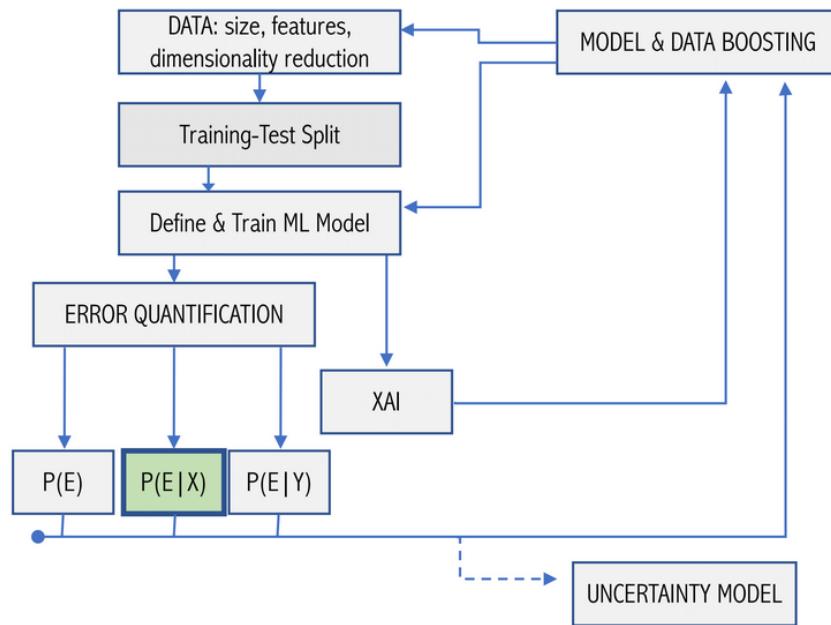


Figure 3.14: Box diagram showing the relative position of section 3.6 in the complete validation pipeline.

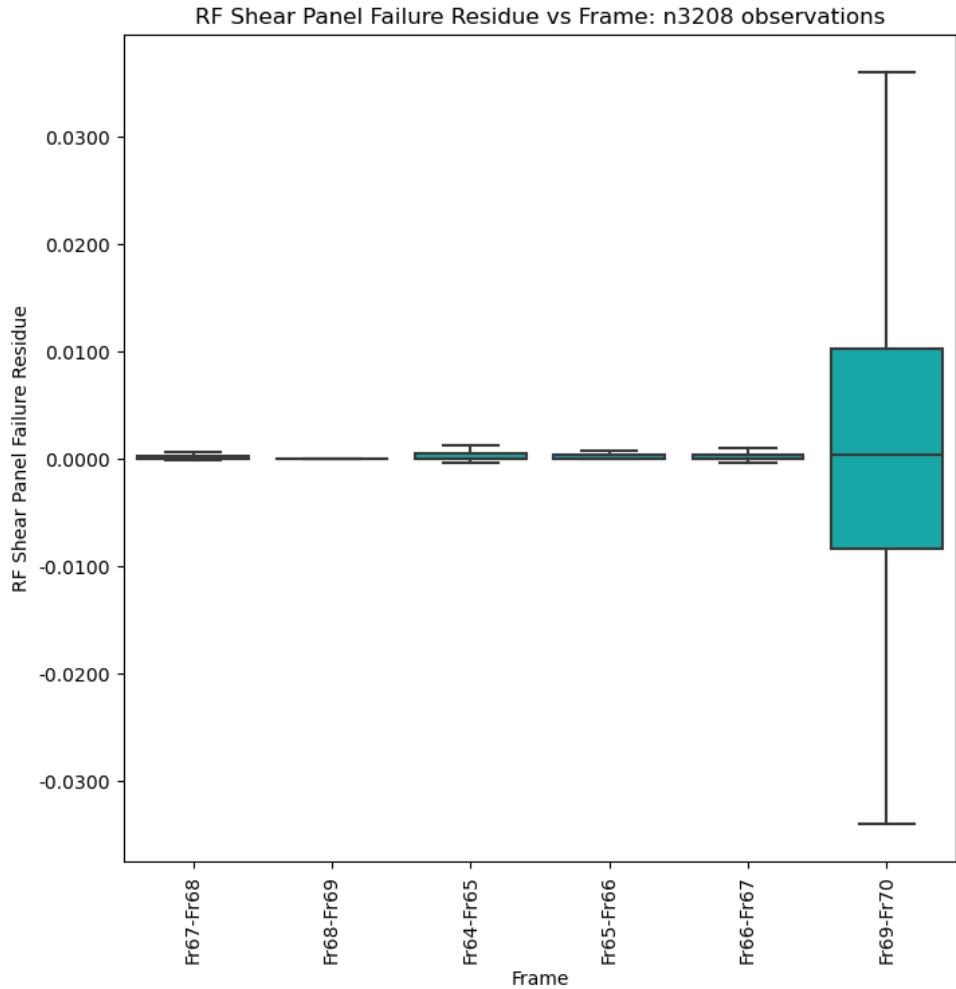
In the first case, training of the model can be reinforced by several means (longer training, hyperparameter tuning, transfer learning, etc.) In the second case, resampling data in deficient regions, or artificial data enhancing techniques such as data augmentation?? are recommended.

The task of recomposing $P(e|X)$ needs to be addressed in a computationally efficient way, given the input space dimensionality m [recall the input features vector is $\mathbf{x} = (x_1, x_2, \dots, x_m)$] can be very large. For the case of MS-S18, for instance, the input space's dimensionality is $m = 31$ (28 numerical inputs plus 3 categorical). With industrial sized datasets (typically reaching up to millions of data points), recomposing $P(e|X)$ becomes unaffordable. The easiest way of overcoming this issue is by binning the input space according to a certain criteria. If we denote the binned input space by $x^B = \{x_1^B, x_2^B, x_3^B \dots x_r^B\}$ where x_j^B represents a certain bin, and there are a total of r bins, then we can substitute the task of recomposing $P(e|X)$ by that of recomposing $P(e|X^B)$. Although simple, this idea allows for an effective computational cost reduction.

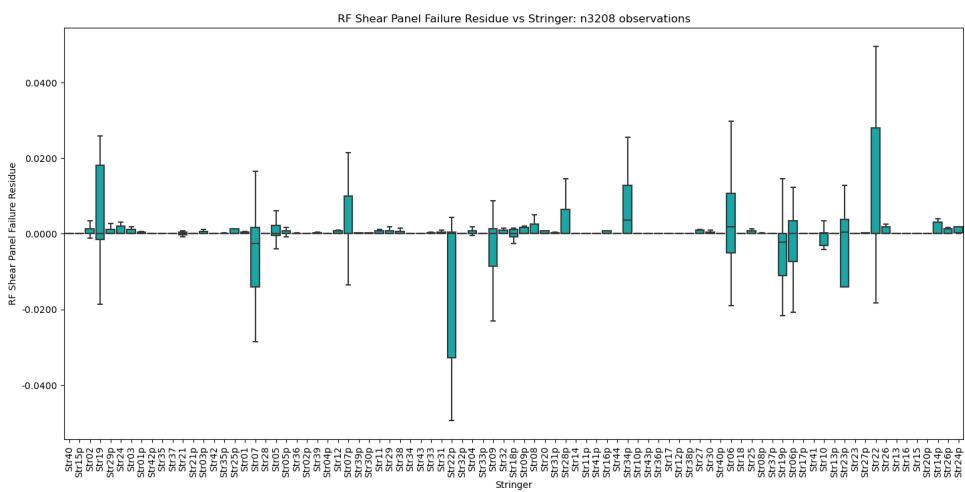
3.6.1 Visualization of error as a function of categorical (discrete) input variables

The immediate criteria for binning the input space is using categorical variables. These variables do in essence bin the input space as they can only take certain (or rather discrete) values. As mentioned in [chapter 2](#), for MS-S18 there are three categorical variables enclosing geometrical information: "Frame", "Stringer", and "dp". [Figure 3.15](#) plots, for each categorical input variable, a box plot of the error as a function of the category.

The box plot layout reports the median, Inter-quantile Range and whiskers. Outliers showing anomalous dispersion of the residue could indicate geometrical configurations (*e.g.* the one defined by frames no. 69-70 and stringer no. 26) where training has been inefficient. The engineer in charge could inspect this zone and conclude, for instance, that the physics at play in that region involve exploding gradients which induce strong non-linearities.



(a) Error conditioned to the categorical variable "Frame"



(b) Error conditioned to the categorical variable "Stringer"

Figure 3.15: Box and whisker plots depicting the residue conditioned to two different categorical variables. Figure 3.15(a): "Frame". Figure 3.15(b): "Stringer". Clearly, outliers can be appreciated in both cases ("Fr69-Fr70" in Figure 3.15(a), and various stringers in Figure 3.15(b)).

3.6.2 Bias detection and quantification (1D)

After the visual inspection carried out in Figure 3.15, a statistical test is needed to actually check whether the observed residue for some frames or stringers is unexpected enough to be considered an "outlier". This is, we need to check whether anomalies shown in Figure 3.15 actually are statistically significant or not. To this end, the following procedure is proposed:

A. Detection of error bias in single categories (ANOVA)

The initial method of choice to detect whether a given input categorical variable shows any bias is the one-way ANOVA test[45]. This method consists on the following steps⁶:

- We fix a categorical variable i under analysis.
- The method scatter-plots the residual error versus the category of x^i , for all n samples of the test set.
- ANOVA makes then an hypothesis test, where the null hypothesis H_0 is that the mean of the (theoretical distribution) error for each of the categories is the same. The ANOVA test rejects H_0 with a certain confidence if the p -value of the test is smaller than a certain level.
- The output of the method is just the p -value, if this is smaller than 0.05, then H_0 is rejected at 95% confidence and we say that the categorical variable x^i shows bias.

In Table 3.12, results of the ANOVA test from MS-S18 are shown. The test shows bias in all the three categorical input variables, but for different output variables each one.

⁶N.B. ANOVA in principle requires data in each category to be normally distributed and homoscedasticity (variances in different categories are similar, cfr.[36, p. 374]). It is important to remark these conditions are sometimes not met.

Table 3.12: P-values results from the one-way ANOVA test. The red labelled cells show that variable "dp" shows bias for the residual distribution of "RF Net Tension", as well as variable "Frame" for the residual distribution of "RF Forced Crippling" and variable "Stringer" for the residual distributions of "RF Net Tension" and "RF Pure Compression". For the rest of the table, p-values greater than 0.05 indicate that H_0 cannot be rejected with a statistical confidence of at least 95%.

1-way ANOVA (p-value)						
	RF Forced Crippling	RF Column Buckling	RF In Plane	RF Net Tension	RF Pure Compression	RF Shear Panel Failure
dp	0.60142	0.35218	0.86072	0.00000	0.68189	0.90204
Frame	0.02972	0.69550	0.24502	0.22363	0.39069	0.59135
Stringer	0.32703	0.24055	0.28428	0.00062	0.04613	0.83613

B. Quantification of error bias in single categories test no. 1: based on error mean outlier

This analysis is to be done only if results from the one-way ANOVA test show bias in at least one categorical variable. Under this assumption, the former test has identified a number of categorical variables that show bias. Here we quantify such bias by checking whether there are certain categories which are "outliers", *i.e.* categories where the error mean is substantially different than for the rest of categories. We check this using z-score[46], which consists on the following steps: we initially construct the mean error e in each category. If the categorical variable has s categories, then we have a vector (e_1, e_2, \dots, e_s) . We then z-score this vector to build

$$\left(\frac{e_1 - \langle e \rangle}{\sigma(e)}, \frac{e_2 - \langle e \rangle}{\sigma(e)}, \dots, \frac{e_s - \langle e \rangle}{\sigma(e)} \right),$$

where

$$\langle e \rangle = \frac{1}{s} \sum_{i=1}^s e_i; \quad \sigma(e) = \sqrt{\frac{1}{s} \sum_{i=1}^s (e_i - \langle e \rangle)^2}$$

We make use of the rule of thumb that category i shows **weak bias** if

$$1 < \frac{|e_i - \langle e \rangle|}{\sigma(e)} \leq 3,$$

whereas category i shows **strong bias** if

$$\frac{|e_i - \langle e \rangle|}{\sigma(e)} > 3.$$

The test output is shown in Table 3.13, where for each categorical variable that was previously identified as having bias, a sub-table showing **only** the specific categories that either show weak or strong bias is displayed.

Table 3.13: Z-score results for biased categorical variables. One table is generated for each pair biased categorical input variable-output variable.

Frame with RF Forced Crippling as output			dp with RF Net Tension as output		
	Fr66-Fr67	Fr69-Fr70		0.000000	0.900000
N	4	1	N	15	2
mean	-0.014281	0.008480	mean	0.004450	-0.005974
Z score	1.174359	1.770030	Z score	1.000000	1.000000
Bias	Weak	Weak	Bias	Weak	Weak

Stringer with RF Net Tension as output					Stringer with RF Pure Compression as output	
	Str06	Str06p	Str31	Str39		Str07
N	1	2	1	1	N	1
mean	-0.022004	0.028592	-0.026716	0.035710	mean	0.178702
Z score	1.225013	1.331218	1.463067	1.690886	Z score	3.086849
Bias	Weak	Weak	Weak	Weak	Bias	Strong

C. Quantification of error bias in single categories test no. 2: based on error variance outlier

This analysis is to be done only if results from the one-way ANOVA test show bias is found in at least one categorical variable.

The procedure is similar as before, but here we investigate whether for certain categories the dispersion (not the mean) of the samples in a given category is substantially different than for the rest of categories. For instance, experience in MS-S18 suggests that the main source of error bias in categorical variables comes from the fact that different categories have different variance. This is quantified by doing a z-score analysis on category variances: if the categorical variable under analysis has s categories, then we have a vector of variances (v_1, v_2, \dots, v_s) , where

v_i is the variance of the error for all samples in category i . We then z-score this vector to build

$$\left(\frac{v_1 - \langle v \rangle}{\sigma(v)}, \frac{v_2 - \langle v \rangle}{\sigma(v)}, \dots, \frac{v_s - \langle v \rangle}{\sigma(v)} \right),$$

where

$$\langle v \rangle = \frac{1}{s} \sum_{i=1}^s v_i; \quad \sigma(v) = \sqrt{\frac{1}{s} \sum_{i=1}^s (v_i - \langle v \rangle)^2}$$

We make use of the rule of thumb that category i shows ***weak variance bias*** if

$$1 < \frac{|v_i - \langle v \rangle|}{\sigma(v)} \leq 3,$$

whereas category i shows ***strong variance bias*** if

$$\frac{|v_i - \langle v \rangle|}{\sigma(v)} > 3$$

The analysis outputs, for each categorical variable that was previously identified as having bias, a table showing only the specific categories that either show weak or strong bias according to this second method.

D. Quantification of error bias in single categories test no. 3: based on identification of linear trend

This test is only applicable to categorical variables for which specific categories have a natural ordering (mathematically, we say there is a canonical geometric embedding for the categorical variable). Such identification needs to be done "by hand" by the engineer in charge.

Such embedding exists when the categorical variable is related for instance to a geometrical location in the plane (for instance the Frames and Stringers are categorical –in so far they are discrete– but they correspond to regions of the plane along a Cartesian and a radial axis, so there is a natural ordering). In those cases, it makes sense to investigate linear trends as these are interpretable.

The method just proceeds to fit a linear model (with just one explanatory variable). If the slope is above a certain threshold and if the fit is good, then we can be confident the linear trend is genuine, and the slope can be used to quantify the linear bias and as a source for uncertainty.

For each categorical variable that has a natural ordering and was previously detected as having bias, the test fits a linear model and outputs the result of this model.

If such model has a slope (statistically) significantly different from zero, then this categorical variable is flagged as having a linear trend.

Illustration of this test is provided in Figure 3.16.

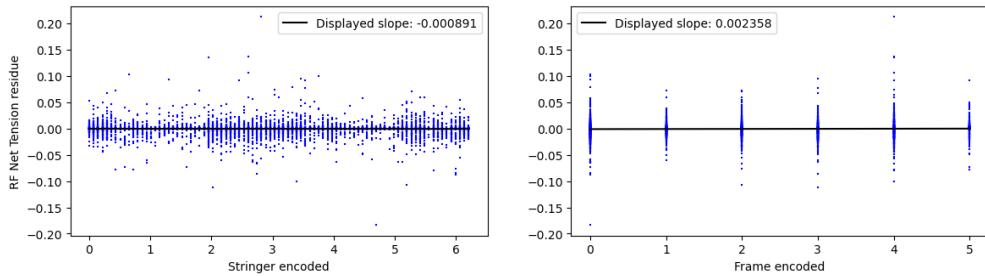


Figure 3.16: Quantification of linear trend for input categorical variables "Stringer" (left) and "Frame" (right). Variables have been numerically encoded following a logical order (*e.g.* in the right image, six integers in the x axis represent the six possible values of variable Frame). The residue of output variable "RF Net Tension" has been plotted against the categorical variable (y axis). In this case, no linear trend is appreciated in neither of the two input variables (no bias present).

3.6.3 Bias detection and quantification (2D)

In the previous section we considered the specific effect (bias) of individual input variables on the error. However, it can be the case that such bias is enhanced when groups of input variables are considered together. The method of choice to analyze whether two independent (input) categorical variables have an effect on the error means is the 2-way ANOVA test[47]. In general the method is not very informative except when the combination of input variables is itself interpretable. This usually requires the external feedback of the engineer. For instance, the engineer can know *a priori* that certain combination of input variables play a synergistic role, go together, etc. For instance, in MS-S18, the categorical variables **Frame** and **Stringer** are geometric locations and thus, together, provide a location of the specific region of the plane that the configuration analyses. One can thus perform a 2-way ANOVA to analyze the presence of bias accordingly. If such bias exists, then bias quantification tests described in subsection 3.6.2 can be applied.

A. Bias detection (2D)

For the pair of (previously chosen) categorical variables, the test performs 2-way ANOVA as described above and outputs the p -value. If this p -value is smaller than 0.05, we conclude that

there exists bias at 95% confidence.

B. Bias quantification (2D)

Suppose we have two categorical variables A and B , where A has q categories $A = (A_1, A_2, \dots, A_q)$ and B has r categories $B = (B_1, B_2, \dots, B_r)$. Suppose also that a 2-way ANOVA concluded that the error is biased on the combined effect of A and B . We then can build all the pairs (A_i, B_j) and interpret each of them as a single category of this "block categorical variable", *i.e.* we have now qr categories. We can subsequently apply the bias quantification tests 1 and 2 described above, applied to the "block categorical variable".

- If bias has been detected, the "block categorical variable" with qr categories is built and a table with qr columns and 2 rows is defined (for test no. 1 and test no. 2 results).
- Bias quantification test no. 1 (error mean, see above) is performed on the block categorical variable, and fills up in the table the first row for those columns that show either weak or strong bias.
- Bias quantification test no. 2 (error variance, see above) is performed on the block categorical variable, and fills up in the table the second row for those columns that show either weak or strong bias.

Results from this tests are illustrated in [Table 3.14](#).

Table 3.14: 2D bias quantification: Results from mean and variance z -score tests on biased Stringer-Frame input pairs (bias detection has been performed through 1-way ANOVA). For the input value combinations showed in the columns, either mean or variance (or both) show bias (weak: z -score < 3 . Strong: z -score > 3). For the combination of Frame Fr67-Fr69 and Stringer Str07, there are not enough test points to compute the z -score variance test.

	Fr64-Fr65:Str38	Fr67-Fr68:Str07
# samples	2	1
1-way ANOVA - ZScore(mean)	Weak	Weak
1-way ANOVA - ZScore(var)	Strong	-

3.6.4 Visualization of error as a function of numerical (continuous) input variables

In this section the same basic analysis is performed as in subsection 3.6.2, although conditioning the error on numerical instead of categorical input variables.

In this section we plot, for each numerical input variable, a scatter plot of the error as a function of the input numerical variable at analysis. From the scatter plot and its linear fit, we expect to (with human intervention) identify bias in the variables which present it, and use visual information for model and data boosting. Illustrative results of two input variables are given in Figure 3.17.

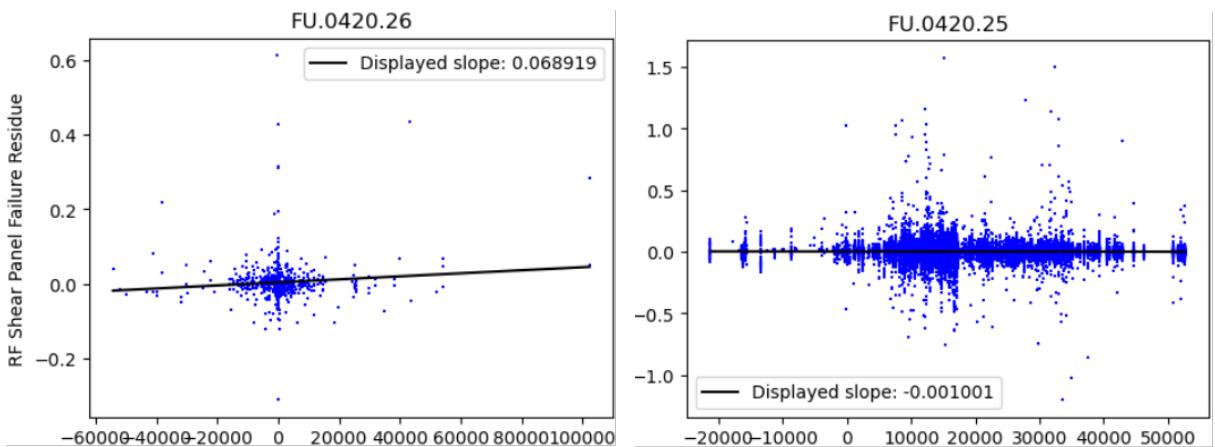


Figure 3.17: Scatter plots of residue against a particular numerical input variable. Left: Input var. FU.0420.26. A linear trend is appreciated, suggesting the model makes worse predictions as input load "FU.0420.26" is larger. Right: input var. FU.0420.25. No linear trend is appreciated, although severe heteroscedasticity can be observed. Note the vertical bar arrangement, showing particular input loads for which dispersion is unusually large. In view of the former results, the engineer may decide to investigate the underlying causes both for the linear trend in the left and for heteroscedasticity on the right. With the information, model and data boosting may be performed. More training data could be decided to be sampled in the range of [20000, 60000] of FU.0420.26, for instance, and some regularization technique could be tried for the loss function in order to penalise inputs triggering high variance observed in the right.

3.6.5 Bias detection and quantification

The goal of this section is to flag statistically significant bias found in output variables, in order to provide mathematical assurance to engineers when interpreting results from Figure 3.17. Similar to categorical variables, we use different tests for bias detection and for bias quantification. As the input variables are now numerical, instead of ANOVA and z-score we use:

A. Detection of linear trends for individual numerical variables

For each input variable, the code fits a linear model that can identify a linear trend of the error as a function of the input variable.

The test outputs (vid. Table 3.15):

- The p -value that accounts for whether such type of bias indeed exists.
- The Pearson correlation coefficient.
- The slope of the best linear model fit.

Table 3.15: Bias detection and quantification on numerical input variables (the results for just five input variables are shown here – in columns–): P-value (statistical significance of the hypothesis that the variable is biased), Pearson coefficient, and slope of the best linear fit are shown in the first three rows. The last row shows the message "Biased" in case the p -value > 0.05 , "NO" otherwise. Data from this table corresponds to residue of the output variable "RF Column Buckling". Analogous tables exist for the rest of the output variables.

	factors	FU.0410.14	FU.0410.15	FU.0410.16	FU.0410.24
p-value	0.600663	0.625659	0.037808	0.480859	0.059544
Pearson coeff.	0.029417	0.027417	-0.116347	-0.039608	0.105612
Slope (normalized)	0.013192	0.027099	-0.141826	-0.019688	0.068905
Linear trend	NO	NO	Biased	NO	NO

B. Discretizing continuous variables

The process is to bin each input numerical variable and thus treat them as categorical, so that one can perform one-way ANOVA followed by bias quantification tests no. 1 (mean outlier) and test no. 2 (variance outlier) former discussed. By default the number of bins is equal to 10.

For each input variable:

- The input variable is binned.
- The steps depicted in sections subsection 3.6.2 are applied.

Results are shown in Table 3.16, Table 3.17 and Table 3.18.

Table 3.16: 1-way ANOVA test results (p-values) for binned numerical input variables. For the output variable "RF Forced Crippling", bias is found in the input variables "FU.0420.25" and "FU.0430.25". Similarly, for the output variable "RF Net Tension", bias is found in the input variable "FU.0430.15".

	1-way ANOVA (p-value)					
	RF Forced Crippling	RF Column Buckling	RF In Plane	RF Net Tension	RF Pure Compression	RF Shear Panel Failure
factors_binned	0.36995	0.57707	0.16276	0.32759	0.72817	0.67178
FU.0410.14_binned	0.14541	0.60928	0.47117	0.86966	0.16037	0.72052
FU.0410.15_binned	0.47235	0.07028	0.23893	0.19848	0.69785	0.36203
FU.0410.16_binned	0.99209	0.29325	0.30693	0.59764	0.86850	0.11857
FU.0410.24_binned	0.28251	0.37036	0.06887	0.30212	0.50172	0.10088
FU.0410.25_binned	0.84519	0.27807	0.35407	0.71585	0.99200	0.10885
FU.0410.26_binned	0.97916	0.23816	0.63145	0.18726	0.59593	0.54611
FU.0420.14_binned	0.11827	0.54753	0.44448	0.69463	0.13006	0.87985
FU.0420.15_binned	0.27700	0.05152	0.19852	0.27864	0.86905	0.30744
FU.0420.16_binned	0.96355	0.29429	0.34697	0.53042	0.88578	0.10234
FU.0420.24_binned	0.28161	0.36135	0.06327	0.32091	0.60557	0.12283
FU.0420.25_binned	0.03686	0.09300	0.46097	0.78027	0.13730	0.83867
FU.0420.26_binned	0.56289	0.15438	0.85899	0.28868	0.77835	0.78597
FU.0430.14_binned	0.20706	0.64043	0.65843	0.16471	0.25147	0.57201
FU.0430.15_binned	0.34776	0.20721	0.70087	0.01776	0.68061	0.87189
FU.0430.16_binned	0.29934	0.05751	0.33328	0.78752	0.76107	0.21693
FU.0430.24_binned	0.50895	0.18564	0.79205	0.09649	0.68746	0.90018
FU.0430.25_binned	0.03176	0.07608	0.30233	0.82106	0.70637	0.73546
FU.0430.26_binned	0.75006	0.18252	0.25337	0.45661	0.99750	0.39317

Table 3.17: Bias quantification in binned FU.0430.15 input variable. Columns represents bins showing bias. The same quantification methods employed for categorical variables (z-score for mean and variance outlier detection) have been used.

FU.0430.15_binned with RF Net Tension as output				
	3	4	8	9
N	269	284	281	299
mean	-0.003908	-0.005275	-0.001238	-0.001153
Z score	1.004073	2.079140	1.096293	1.163220
Bias	Weak	Weak	Weak	Weak

Table 3.18: Summary of binned input variables bias quantification after binning the numerical variables and performing one-way ANOVA and z-score tests to every bin.

RF Forced Crippling		
	FU.0420.25_binned	FU.0430.25_binned
1-way ANOVA - ZScore(mean)	Weak	Weak
1-way ANOVA - ZScore(var)	Weak	Weak

RF Net Tension		
	FU.0430.15_binned	
1-way ANOVA - ZScore(mean)	Weak	
1-way ANOVA - ZScore(var)	Weak	

3.7 Distributed error quantification: conditioning the error distribution on the output space

This section corresponds to box labelled " $P(E|Y)$ " in Figure 3.18.

Figure 3.7 shows an important fact about the error distribution: dispersion is not homogeneous in the full range of the output variables –in the case of MS-S18, all the six Reserve Factors belong in the range (1, 5)–, *i.e.* error is heteroscedastic. This means that the residue is dependent on the prediction of the model, which has important implications in our validation approach. First, some statistical tests rely on the assumption of homoscedasticity. Second and most important, heteroscedasticity is a first indicator which tells us the need to carry out the analysis of the error distribution conditioned on the output space, $P(e|Y)$. Characterising the error in different regions of the output space is the final building block for our complete uncertainty model (vid. section 3.8).

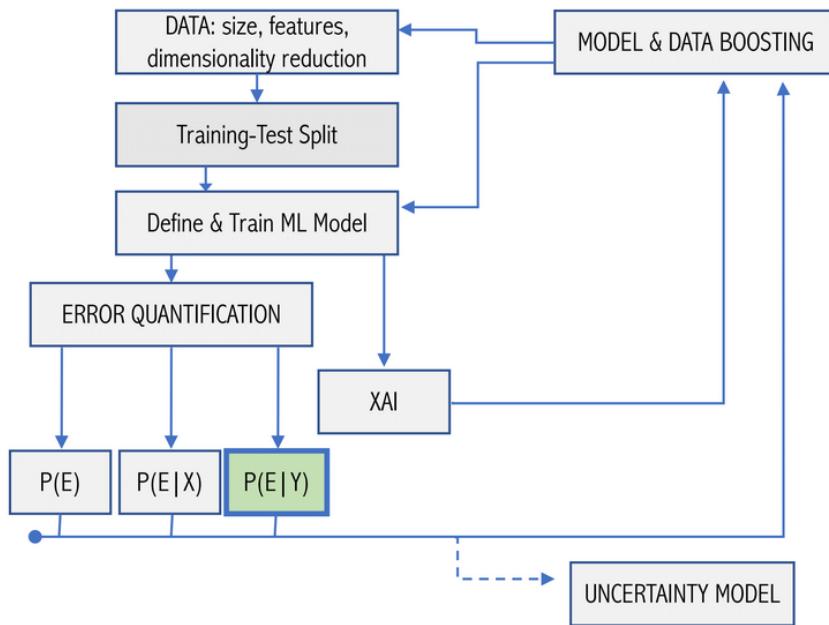


Figure 3.18: Relative position of section 3.7 in the complete validation pipeline.

The first step in the reconstruction of $P(e|Y)$ is to try to fit the error distribution to some well known theoretical distribution. But instead of just repeating what was done in section 3.5, now we add an extra variable to the equation: the prediction of the model y (representing the k -th Reserve Factor from the output vector \mathbf{y}). Instead of trying to fit the whole error distri-

bution, we filter the test set by establishing $y < y_0$ and try to fit the filtered test set to one of the well known distributions discussed in [section 3.5](#) (Normal, Laplace, Cauchy, JohnsonSU). In [Figure 3.19](#) four of the six output variables –for convenience– of MS-S18 are put under a 1-variable K-S test to assess the goodness of fit of the error to the mentioned distributions. For this figure, the absolute value of the residue has been used instead of the plain residue. We can observe that, while all four theoretical distributions start fitting $P(e)$ when $y_0 < 1.5$, soon the p-value drops down indicating a mismatch between the empirical error and Cauchy, Laplace, and Normal distributions. Only the JohnsonSU remains a good fit, but finally diverges as well when $y_0 \approx 3$. The conclusion is that only the JohnsonSU distribution is found to be a good match for $P(e|Y)$, but only when filtering the test set up to moderate values of y_0 . This suggests presence of extreme phenomena in the error distribution, which in fact is in accordance with the heteroscedastic behaviour observed in [Figure 3.7](#), showing dispersion increases with the output y .

By filtering the test set using a maximum allowable value of predictions \hat{y}_0 , we have reached to interesting conclusions. The question now is, why filtering the test set just the particular way we have done it? The choose was motivated by the intuition from [Figure 3.7](#) that dispersion increases with the prediction value. A logical way of filtering the test set in order to find local information of the error distribution is binning it. For the next part, we bin the test set in 10 bins according to their output values, as described in [Table 3.19](#).

For each bin, a 2-sample K-S test comparing the true against the predicted distribution is performed. Such test is only performed if the number of points inside each bin is greater than a minimum threshold (set at 30 points). Results (only four output variables are displayed for convenience) are shown in [Figure 3.20](#). There we can see a clear tendency: while for low output values the two distributions conform well, the p-value eventually drops for higher bins. This confirms the information of [Figure 3.7](#) and [Figure 3.19](#) that the model predicts very well when the output is in the low range, while it has more difficulties predicting higher Reserve Factors. This idea is reinforced by looking at [Figure 3.21](#), where violin plots of the residue for every bin have been plotted. In this figure, it can be clearly appreciated that the (absolute) residue dispersion is dependent with the model’s prediction, growing as \hat{y} does so. As previously discussed, this is the desired behaviour, as low RF indicate high failure risk, whereas high reserve factors indicate low risks. It is therefore preferable to accumulate variance at the low RF extreme of the spectrum.

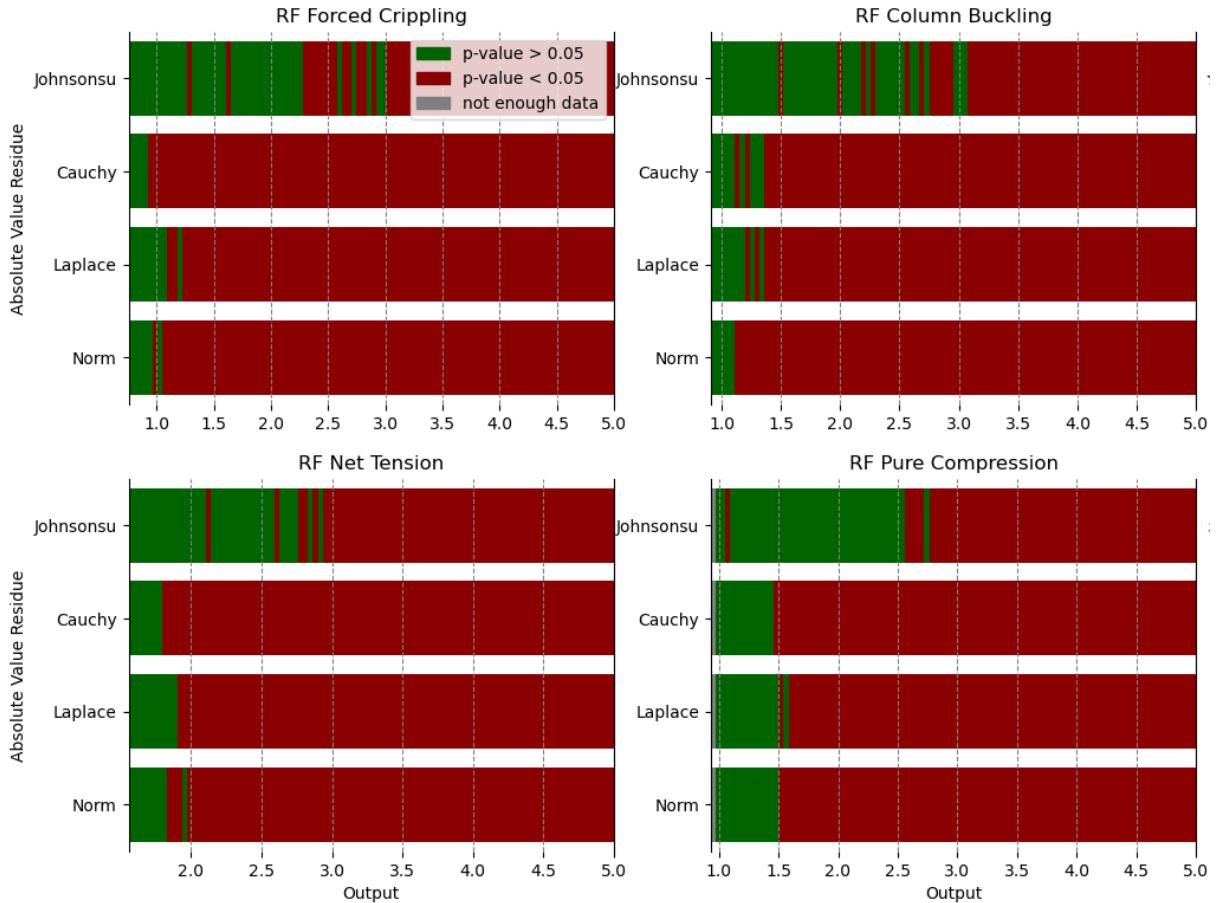


Figure 3.19: Goodness of fit of the error distribution (test set previously filtered) to some parametrised distributions. The test is performed individually for the output variables (Reserve Factors) displayed on top of each image. The goodness of fit is assessed with a 1-variable K-S test. The test set is filtered using the x axis, rejecting every point whose output \hat{y} is larger than x . If the p-value resulting from the K-S test in the filtered test set is greater than 0.05, the vertical slice at position x is labelled green (red if $p\text{-value} < 0.05$, grey if less than a threshold number of points –namely, 30– are present in the filtered test set).

In Figure 3.22 the error distribution of each individual bin is put under a K-S test to assess the goodness of fit to one of the aforementioned theoretical distributions. Results show similar conclusions to Figure 3.19: only the JohnsonSU makes a good match, even though the p -value drops to under 0.05 for certain bins (note the last bin never passes the test).

Finally, we conclude by pointing out that the binned output space can be subject to the same bias detection and quantification tests (1-way ANOVA, z -score for median, z score for variance, linear fit) which were discussed in subsection 3.6.2 for categorical input (and binned numerical) variables.

Table 3.19: Binnarization of the test set. The test set has been divided into ten bins according to output values. Extremes of the six intervals –one for each Reserve Factor– defining the bins are shown below.

	Bins limits									
	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5	Bin 6	Bin 7	Bin 8	Bin 9	Bin 10
RF Forced Crippling	(0.754, 1.179)	(1.179, 1.603)	(1.603, 2.028)	(2.028, 2.453)	(2.453, 2.877)	(2.877, 3.302)	(3.302, 3.726)	(3.726, 4.151)	(4.151, 4.575)	(4.575, 5.001)
RF Column Buckling	(0.909, 1.318)	(1.318, 1.727)	(1.727, 2.136)	(2.136, 2.545)	(2.545, 2.954)	(2.954, 3.364)	(3.364, 3.773)	(3.773, 4.182)	(4.182, 4.591)	(4.591, 5.001)
RF In Plane	(2.01, 2.309)	(2.309, 2.608)	(2.608, 2.907)	(2.907, 3.206)	(3.206, 3.505)	(3.505, 3.804)	(3.804, 4.103)	(4.103, 4.402)	(4.402, 4.701)	(4.701, 5.001)
RF Net Tension	(1.556, 1.901)	(1.901, 2.245)	(2.245, 2.589)	(2.589, 2.934)	(2.934, 3.278)	(3.278, 3.622)	(3.622, 3.967)	(3.967, 4.311)	(4.311, 4.656)	(4.656, 5.001)
RF Pure Compression	(0.933, 1.34)	(1.34, 1.747)	(1.747, 2.153)	(2.153, 2.56)	(2.56, 2.967)	(2.967, 3.373)	(3.373, 3.78)	(3.78, 4.187)	(4.187, 4.593)	(4.593, 5.001)
RF Shear Panel Failure	(0.854, 1.269)	(1.269, 1.683)	(1.683, 2.098)	(2.098, 2.512)	(2.512, 2.927)	(2.927, 3.342)	(3.342, 3.756)	(3.756, 4.171)	(4.171, 4.585)	(4.585, 5.001)

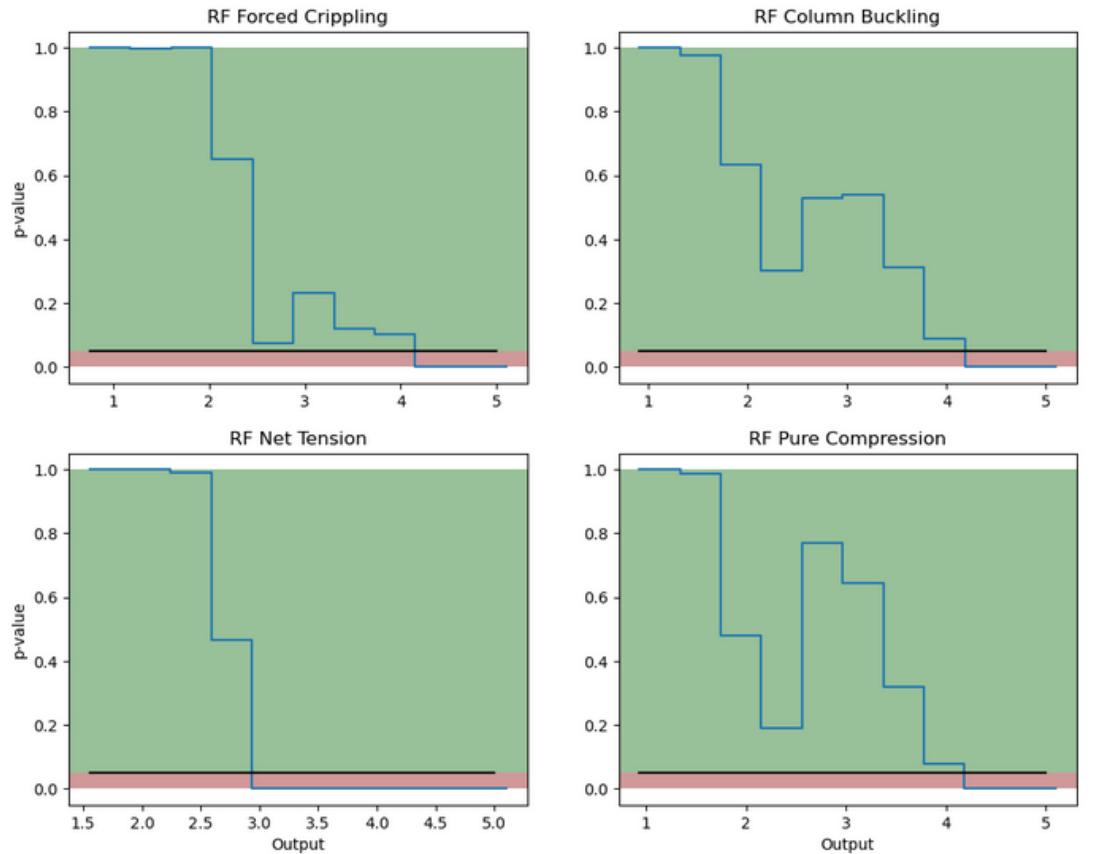


Figure 3.20: Line plot of the p -value from the true-predicted distributions under a 2 sample K-S test. Distribution similarity is rejected if p -value < 0.05 .

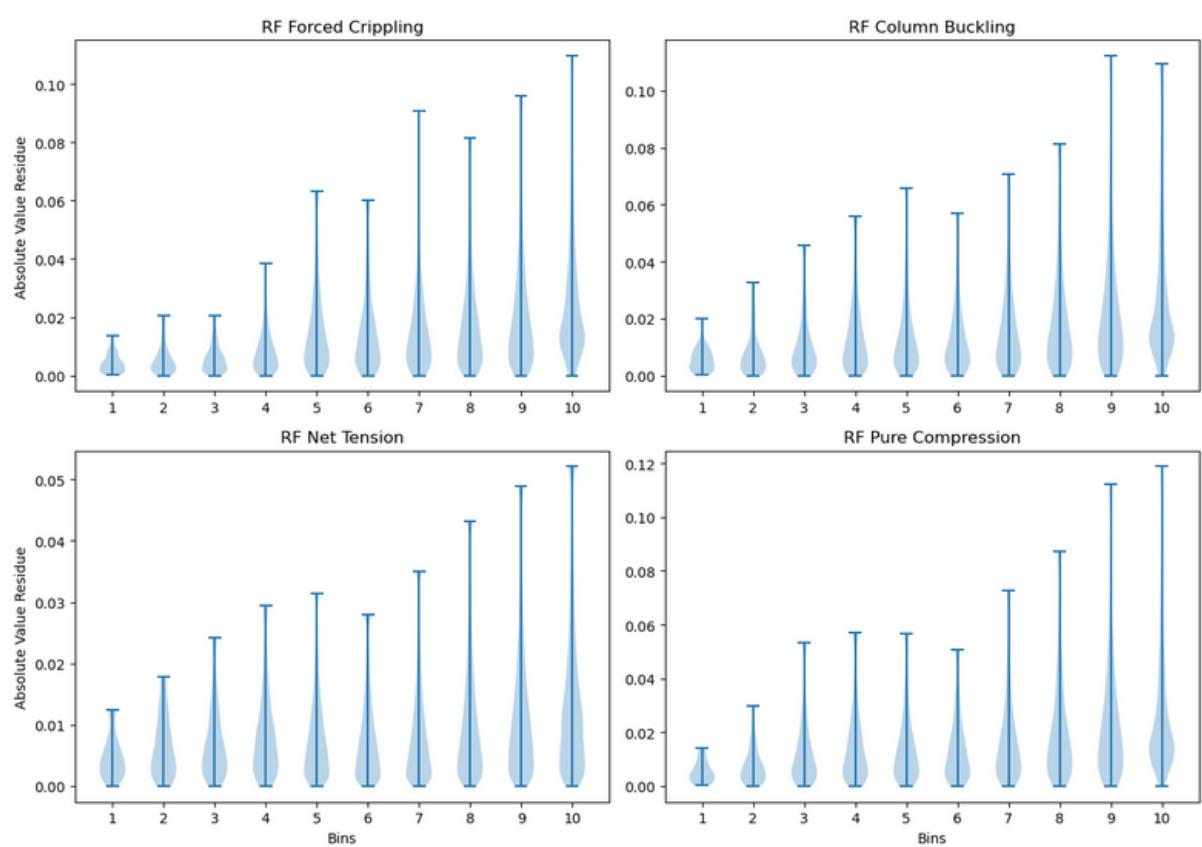


Figure 3.21: Violin plots showing the absolute value residue for every bin.

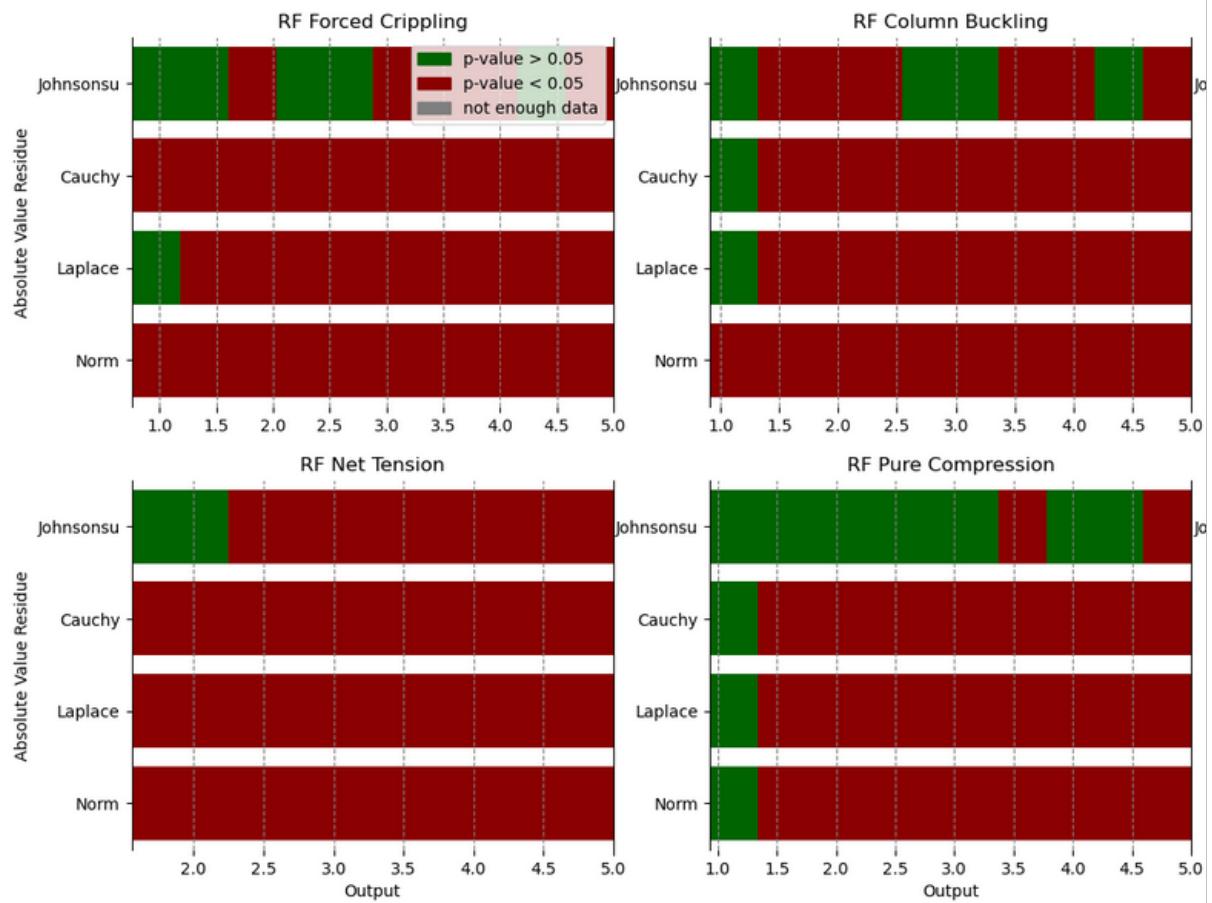


Figure 3.22: Sliced-bar with p-values from the binned error under a K-S test to assess the goodness of fit to several parametrised distributions.

3.8 Uncertainty model

This section corresponds to box labelled "Uncertainty model" in Figure 3.1. It is envisioned with the following question in mind:

Given a new configuration with input parameters $\mathbf{x} = (x_1, x_2, \dots, x_m)$ and prediction $\mathbf{y} = (y_1, y_2, \dots, y_q)$, where $\mathbf{y} = \mathcal{F}(\mathbf{x})$, what is the expected uncertainty (error) associated to the prediction \mathbf{y} ?

To the aim of answering this question, the main tasks are:

- To build an **uncertainty model**.
- To **validate** such model.

The expected uncertainty translates into (for instance) $q = 6$ confidence intervals $(CI_1, CI_2, \dots, CI_6)$, such that the k -th predicted reserve factor y_k has a confidence interval $CI_k = [y_k^{\min}, y_k^{\max}]$. Other types of uncertainty quantifications are possible (standard deviation, credible interval, etc).

Importantly, there are various sources of uncertainty that need to be combined:

1. **Global error distribution** (cfr. section 3.5). This is, the distribution of error in the test set, $P(E)$.
2. **Local-output error distribution** (cfr. section 3.7). This is, the distribution of the error in the test set, conditioned to a specific region of the prediction, $P(E|Y)$.
3. **Local-input error distribution** (cfr. section 3.6). This is, the distribution of the error in the test set, conditioned to a specific region of the input space (error bias characterization), $P(E|X)$.

Our uncertainty model will combine contributing factors (1), (2) and (3) into an effective uncertainty.

Each part (1-3) of the process will need to be **validated**. To this aim, we split the test set into a calibration set \mathcal{E}^{cal} and a test set⁷ $\mathcal{E}^{\text{test}}$ (80/20 random split), estimate the uncertainty model on the calibration set and evaluate it by computing its **coverage** (*i.e.* the percentage of

⁷Not to be confused with $\mathcal{S}^{\text{test}}$. Here we denote by "test set" the empirical error distribution sampled from

configurations whose true output belongs to the confidence interval region) in the test set.

3.8.1 Global uncertainty model based on $P(E)$

After a sanity check that the dataset under analysis (here, the full calibration set) fulfils the minimum size requirements (threshold is set at 300 points for each output variable \mathbf{Y}_i) we proceed with:

A. Global Uncertainty Model (GUM) computation

The interval $[P2.5, P97.5]$ (or in general, the interval $[a, b]$ if other percentiles are chosen) of $P(E)$ is estimated in the calibration set, where the lower bound P2.5 and the upper bound P97.5 are not raw estimates but the mean of a bootstrap analysis of the lower and upper percentiles respectively. The bootstrapping procedure is implemented as described in [algorithm 3](#). This procedure is repeated for all m output variables. Illustration of the GUM is provided in [Table 3.20](#).

Algorithm 3: Bootstrapped uncertainty CI

Data: Calibration set \mathcal{E}^{cal}
Result: Bootstrapped 95% CI [P2.5,P97.5]

- 1 Initialize empty arrays $P_{2.5}$ and $P_{97.5}$;
- 2 **for** $i = 1$ **to** 100 **do**
- 3 Build calibration set i by randomly sampling (with replacement) a total number of points equal to the size of the calibration set;
- 4 Compute lower ($P_{2.5}^{(i)}$) and upper ($P_{97.5}^{(i)}$) percentiles;
- 5 Add $P_{2.5}^{(i)}$ to $P_{2.5}$;
- 6 Add $P_{97.5}^{(i)}$ to $P_{97.5}$;
- 7 **end**
- 8 $P_{2.5_mean} \leftarrow \text{mean}(P_{2.5})$;
- 9 $P_{97.5_mean} \leftarrow \text{mean}(P_{97.5})$;
- 10 **return** $P_{2.5_mean}$ and $P_{97.5_mean}$;

B. Global Uncertainty Model (GUM) evaluation

The (bootstrapped) error coverage is estimated in the test set. This is done by computing the percentage of points in the test set whose error lies inside the chosen uncertainty model. The bootstrap procedure is carried out as described in [algorithm 4](#).

Table 3.20: Global Uncertainty Model: Confidence Interval (mean and extremes) for output variables "RF Forced Crippling" (left) and "RF Column Buckling" (right). Header showing $[y_{min}, y_{max}]$ in the calibration set.

[0.75, 4.99]		[0.9069, 4.99]	
Mean	0.001859	Mean	0.002292
CI	[-0.5366, 0.1018]	CI	[-0.5339, 0.1076]

Algorithm 4: Bootstrapped coverage

Data: Uncertainty model (CI), test set $\mathcal{E}^{\text{test}}$

Result: Mean and confidence interval of error coverage

- 1 Initialize empty array *error_coverages*;
 - 2 **for** $i = 1$ **to** 100 **do**
 - 3 Build resampled test set i by randomly sampling (with replacement) a total number of points equal to the size of the test set;
 - 4 Compute error coverage in resampled test set i ;
 - 5 Add error coverage to *error_coverages*;
 - 6 **end**
 - 7 Sort *error_coverages* in increasing order;
 - 8 Calculate mean, 2.5th percentile, and 97.5th percentile of *error_coverages*;
 - 9 **return** Mean, 2.5th percentile, and 97.5th percentile;
-

The output is the bootstrap error coverage and its 95% confidence interval. The criteria for a correctly calibrated error coverage is that it is correctly calibrated if $(b-a)\%$ (by default, 95%) lies inside the bootstrapped error coverage confidence interval.

This procedure is repeated for all m output variables. For a graphical interpretation of results, cfr. Figure 3.23. The six GUM's CIs are given in Table 3.21.

Table 3.21: Global Uncertainty Model. The six confidence intervals (bootstrapped) defining the GUM alongside their coverage –in the calibration set– are showed.

Coverage table						
	RF Forced Crippling	RF Column Buckling	RF In Plane	RF Net Tension	RF Pure Compression	RF Shear Panel Failure
Coverage	98.89	99.07	98.92	98.9	99.12	99
CI	[98.61, 99.13]	[98.77, 99.31]	[98.57, 99.21]	[98.8, 98.99]	[98.8, 99.37]	[98.8, 99.17]

the points belonging to $\mathcal{S}^{\text{test}}$.

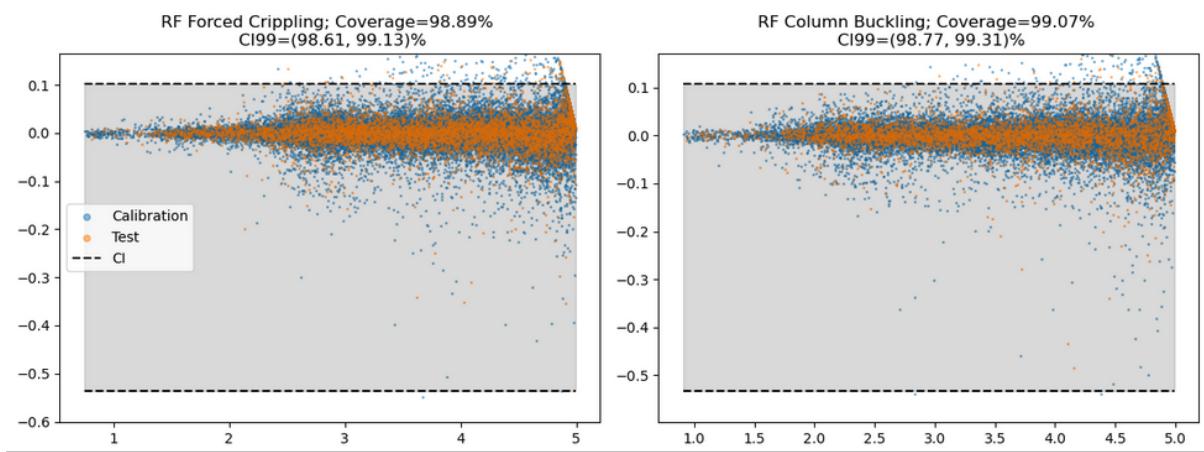


Figure 3.23: Global Uncertainty Model: error coverage. The coverage is higher than 95% for both output variables –for convenience just two output variables are depicted– what means the calibration criteria is successfully met.

3.8.2 Local uncertainty model based on $P(E|Y)$

A. Output variable binning

In order to be able to computationally afford this analysis, the output space \mathbf{Y} must be binned. Detail of the binning is given in [Table 3.22](#).

In the first place, a scatter plot of the error of each point in the calibration set is plotted for illustration as a function of the true output y_t in [Figure 3.24](#).

The calibration set is then partitioned into a total of equispaced n_{bins} bins according to the range of the true output y_t .

For illustration, a total of n_{bins} violin plots (one per bin, concatenated in a single plot where the "x axis" is the bin number) is plotted in [Figure 3.25](#), where each violin plot describes the error density of the points in the respective bin.

B. Local Output Uncertainty Model (LOUM) computation

First, a sanity check that the number of points in each bin is above a certain threshold (namely, 300 points) assures statistical robustness for this next part.

For those bins for which the minimum-size requirement is not met, the bin is given the Global Uncertainty Model (GUM).

For those bins for which the minimum-size requirement is met, the code computes an uncertainty model in terms of the bootstrap [P2.5, P97.5] (or a general [a,b]) using [algorithm 3](#),

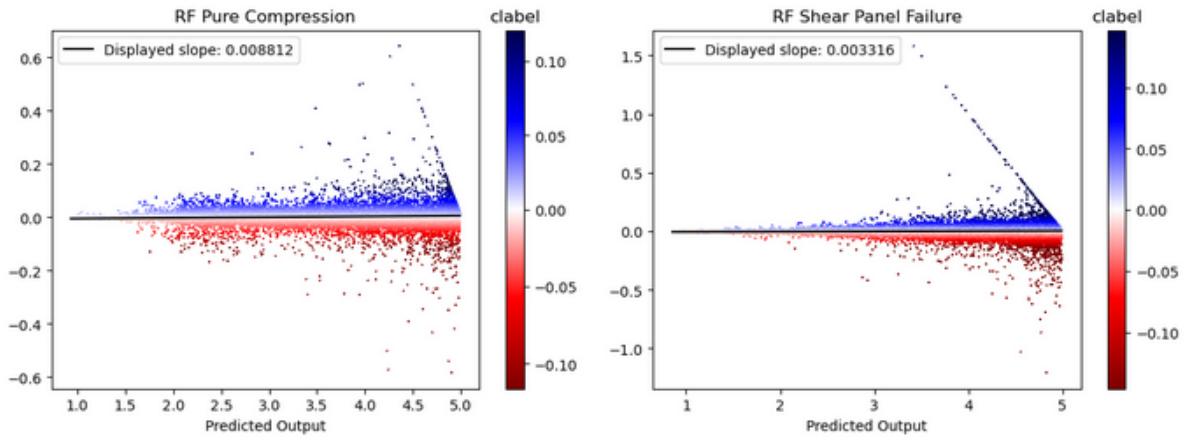


Figure 3.24: Local Uncertainty Model based on $P(E|\mathbf{Y})$: Scatter plot of the residue as a function of the predicted output \hat{y} , for output variables "RF Pure Compression" and "RF Shear Panel Failure".

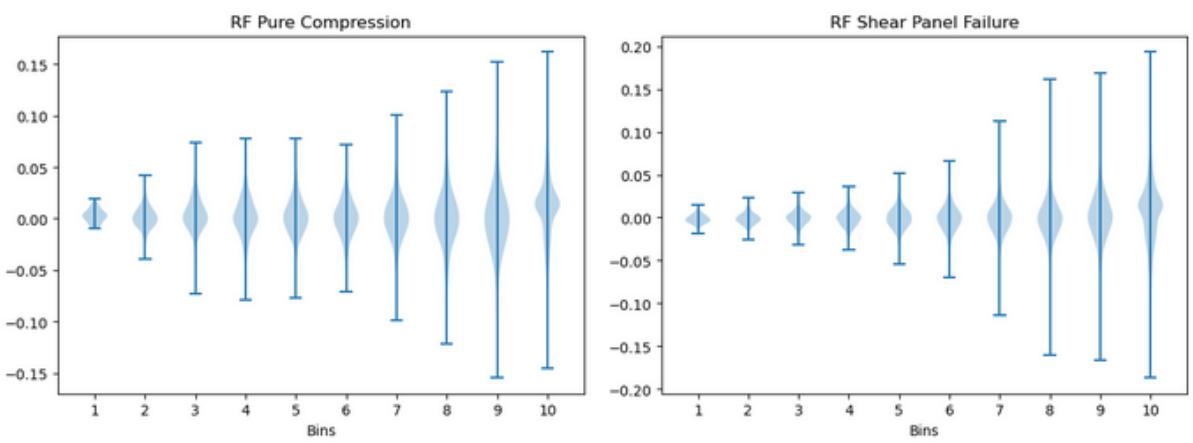


Figure 3.25: Local Uncertainty Model based on $P(E|\mathbf{Y})$: Violin plots of the residue, for binned output variables "RF Pure Compression" and "RF Shear Panel Failure".

and assigns to the bin the resulting local uncertainty model (vid. Table 3.22).

This procedure is repeated for all m output variables.

C. Local Output Uncertainty Model (LOUM) evaluation

The bootstrapped error coverage in the test set associated to the LOUM vector is computed with algorithm 4, with the nuance that for each point in the test set, we need to find, based on the point's error, to which bin of the output variable it belongs, and then it is checked whether such error is within the corresponding bootstrapped interval [P2.5, P97.5].

The output is the bootstrap error coverage and its 95% confidence interval. The criteria for a correctly calibrated error coverage is that it is correctly calibrated if (b-a)% (by default,

Table 3.22: Local Output Uncertainty Model: Bootstrapped CIs for each of the ten bins into which the output variable "RF Forced Crippling" has been binned. The mean of each CI is also given, as well as the bin extrema $[\hat{y}_{min}, \hat{y}_{max}]$ for every bin (in the first row).

Input: RF Forced Crippling; Output: RF Forced Crippling									
	[0.75, 1.174]	[1.174, 1.598]	[1.598, 2.022]	[2.022, 2.446]	[2.446, 2.87]	[2.87, 3.294]	[3.294, 3.718]	[3.718, 4.142]	[4.142, 4.566]
Mean	-0.000885	-0.0001975	0.0001137	0.001315	0.0004612	-0.0002439	0.0006223	0.001116	0.001307
CI	[-0.02095, 0.01634]	[-0.03995, 0.02743]	[-0.06692, 0.02547]	[-0.1572, 0.05091]	[-0.2539, 0.07597]	[-0.1987, 0.0708]	[-0.4765, 0.09805]	[-0.4279, 0.1017]	[-0.321, 0.1269]

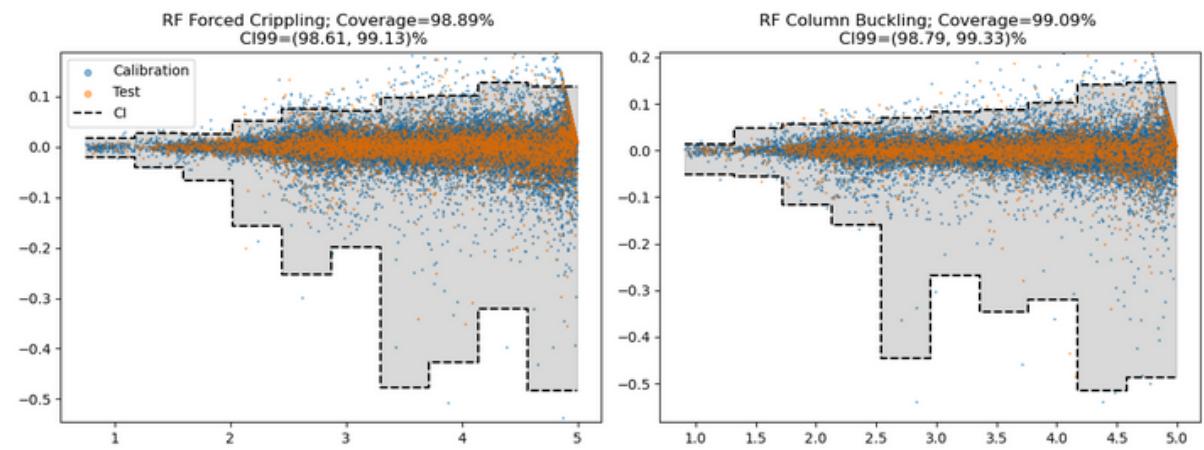


Figure 3.26: LOUM coverage in the validation set. For both output variables coverage is above the minimum 95% threshold.

95%) lies inside the bootstrapped error coverage confidence interval.

This procedure is repeated for all m output variables.

A scatter plot of the error of each point in the calibration set is displayed in Figure 3.26, along with the uncertainty model interval (the upper and lower percentiles emphasized in blue and red respectively). This LOUM shows up as a piece-wise constant upper curve and a piece-wise constant lower curve. LOUM coverage for all the six Reserve Factors can be found in Table 3.23.

Table 3.23: LOUM coverage. Coverage in the validation set is displayed in the first row, whereas the coverage bootstrapped CIs are displayed in the second row.

Input: dp; Output: RF Forced Crippling		
	0.000000	0.900000
Mean	0.001506	0.002707
CI	[-0.5033, 0.09891]	[-0.5199, 0.1104]

3.8.3 Local uncertainty model based on $P(E|X)$

A. Input variable binarization

In order to be able to computationally afford LIUM calculation, it is needed to bin the input space in the first place. Details of the resulting bins can be consulted in Table 3.24

For illustration, a scatter plot of the error of each point in the calibration set as a function of x_i is displayed in Figure 3.27 (just two input numerical and two categorical variables have

been selected for convenience).

For each continuous input variable x_i the calibration set is partitioned into a total of equispaced n_{bins} bins according to the range of input variable x_i .

For illustration, in Figure 3.28(a) n_{bins} number of violin plots (one per bin, concatenated in a single plot where the "x axis" is the bin number) are plotted, where each violin plot describes the error density of the points in the respective bin.

Binning is exclusive to continuous numerical variables. For illustration, in Figure 3.28(b) and Figure 3.28(c) m_i violin plots (one per each possible value that the corresponding input variable x_i can take) are plotted, where each violin plot describes the error density of the points whose input variable x_i takes the same value.

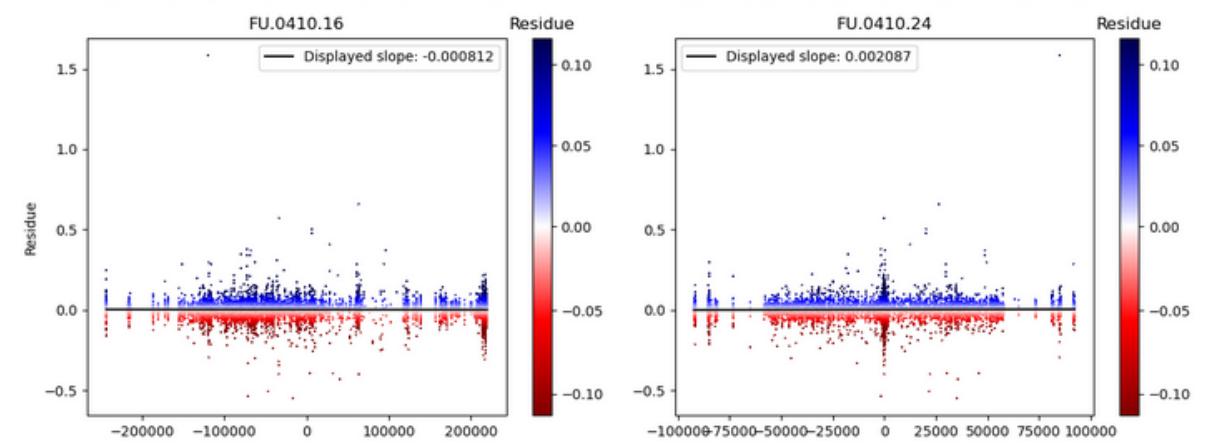


Figure 3.27: LIUM: Scatter plot of the residue of "RF Forced Crippling" as a function of the input x (left: x is "FU.0410.16". Right: x is "FU.0410.24").

B. Local Input Uncertainty Model (LIUM) computation

For each continuous input variable x_i that has been discretized into n_{bins} bins:

- It is checked whether the number of points inside each bin fulfils the minimum bin size (set at 300 points) requirement.
- For those bins for which the minimum bin size requirement is not met, the bin is given the Global Uncertainty Model (GUM).
- For those bins for which the minimum bin size requirement is met, an uncertainty model is computed in terms of the bootstrapped interval [P2.5, P97.5] (or a general $[a,b]$) with

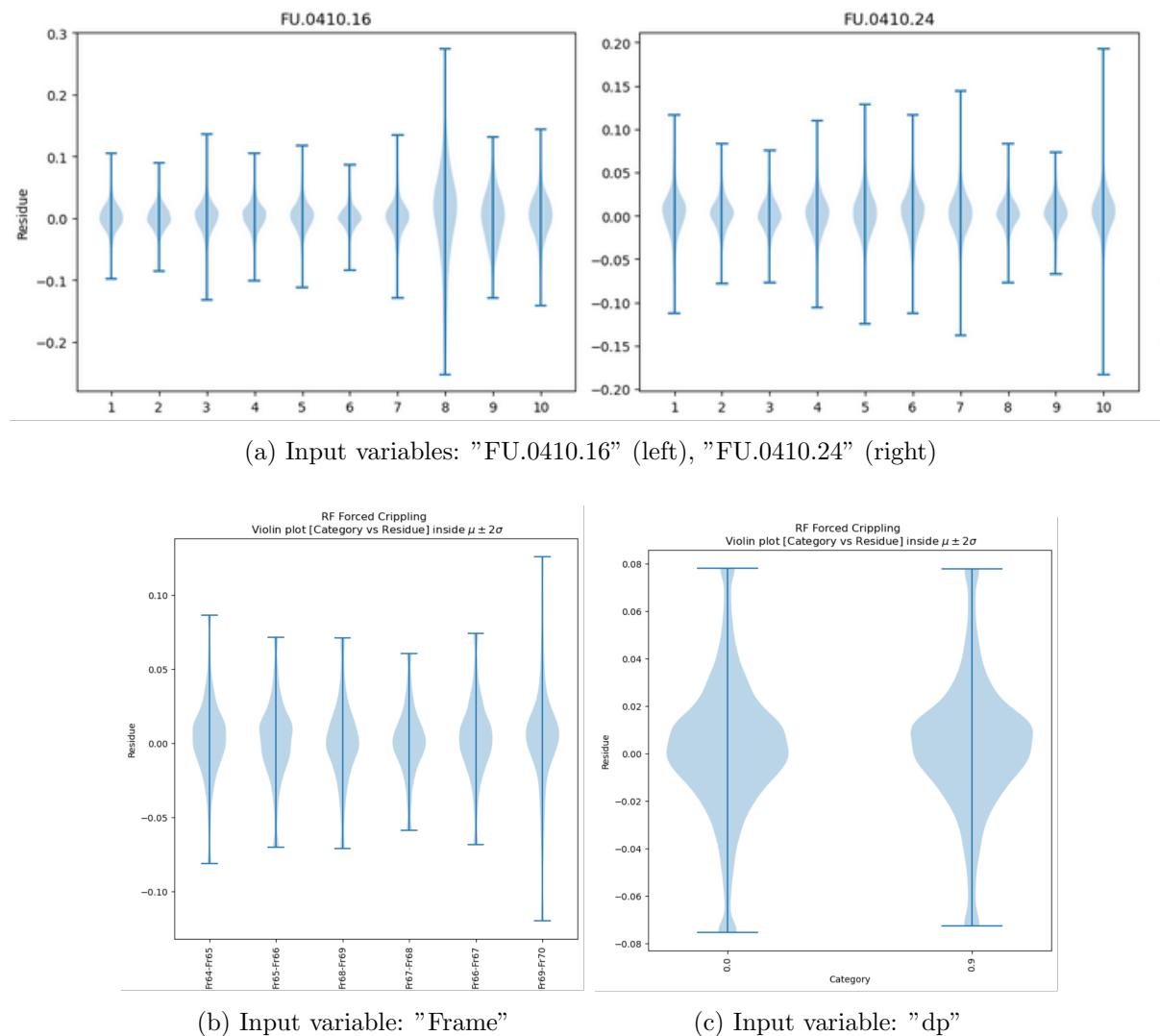


Figure 3.28: Violin plots showing the residue of output variable "RF Forced Crippling". (a): Numerical inputs (binned). (b) and (c): categorical inputs.

algorithm 3, and the resulting local uncertainty model is assigned to the bin.

- The output (illustration of it is provided in Table 3.24) is the resulting vector of bootstraps [P2.5, P97.5] associated to the input variable x_i .

For each categorical input variable x_i :

- The code checks whether the number of points inside each category fulfils the minimum bin size requirement (the minimum bin size requirement is just a name, note nevertheless that categorical variables have not been binned).

Table 3.24: LIUM: Bootstrapped CIs for binned input variable "FU.0430.26" (bins' extrema displayed as column names). Residue corresponds to output variable "RF Shear Panel Failure".

Input: FU.0430.26; Output: RF Shear Panel Failure								
	[-1.656e+04, -1.325e+04]	[-1.325e+04, -9938]	[-9938, -6625]	[-6625, -3313]	[-3313, 0]	[0, 3313]	[3313, 6625]	[6625, 9938]
Mean	-0.003271	0.007046	-0.001071	0.001169	0.001253	0.0003371	-0.0001739	0.002927
CI	[-0.3023, 0.1606]	[-0.179, 0.1424]	[-0.2358, 0.1031]	[-0.6774, 0.09636]	[-0.6403, 0.1062]	[-1.116, 0.1228]	[-0.2585, 0.1092]	[-0.1927, 0.175]

Table 3.25: LIUM: Bootstrapped CIs for categorical input variable "dp" (possible values that the variable can take are displayed as column names). Residue corresponds to "RF Shear Panel Failure" output variable.

Input: dp; Output: RF Forced Crippling		
	0.000000	0.900000
Mean	0.001506	0.002707
CI	[-0.5033, 0.09891]	[-0.5199, 0.1104]

- For those categories for which the minimum bin size requirement is not met, the category is given the Global Uncertainty Model (GUM).
- For those categories for which the minimum bin size requirement is met, the code computes an uncertainty model in terms of the bootstrap [P2.5, P97.5] (or a general [a,b]) with [algorithm 3](#), and assigns to the category the resulting local uncertainty model.
- The output (illustration of it is provided in [Table 3.25](#)) is the resulting vector of bootstraps [P2.5, P97.5] (with length equal to the number of categories of the categorical variable x_i) associated to the input variable x_i .

This procedure is repeated for all m output variables.

C. Local Input Uncertainty Model (LIUM) evaluation

For each input variable:

- The bootstrap error coverage in the test set associated to the LIUM vector is computed with [algorithm 4](#), with the nuance that for each point in the test set, we need to find, based

on the point's error, to which bin (or category) of the input variable it belongs, and then check whether such error is within the corresponding bootstrapped interval [P2.5, P97.5].

- The output is the bootstrap error coverage and its 95% confidence interval. The criteria for a correctly calibrated error coverage is that it is correctly calibrated if (b-a)% (by default, 95%) lies inside the bootstrapped error coverage confidence interval.

This procedure is repeated for all m output variables.

For illustration, a scatter plot of the error of each point in the calibration set is displayed in [Figure 3.29](#), along with the uncertainty model interval (the upper and lower percentiles emphasized in blue and red respectively). This LIUM shows up as a piece-wise constant upper curve and a piece-wise constant lower curve.

3.8.4 Full uncertainty model (FUM)

The FUM is a conservative uncertainty model that consistently takes, for a given point of the test set and its associated k -th reserve factor y , (\mathbf{x}, y) , the largest uncertainty:

$$\text{FUM}((\mathbf{x}, y)) = \max\{\text{GUM}, \text{LOUM}(y), \text{LIUM}(x_1), \text{LIUM}(x_2), \dots, \text{LIUM}(x_n)\}$$

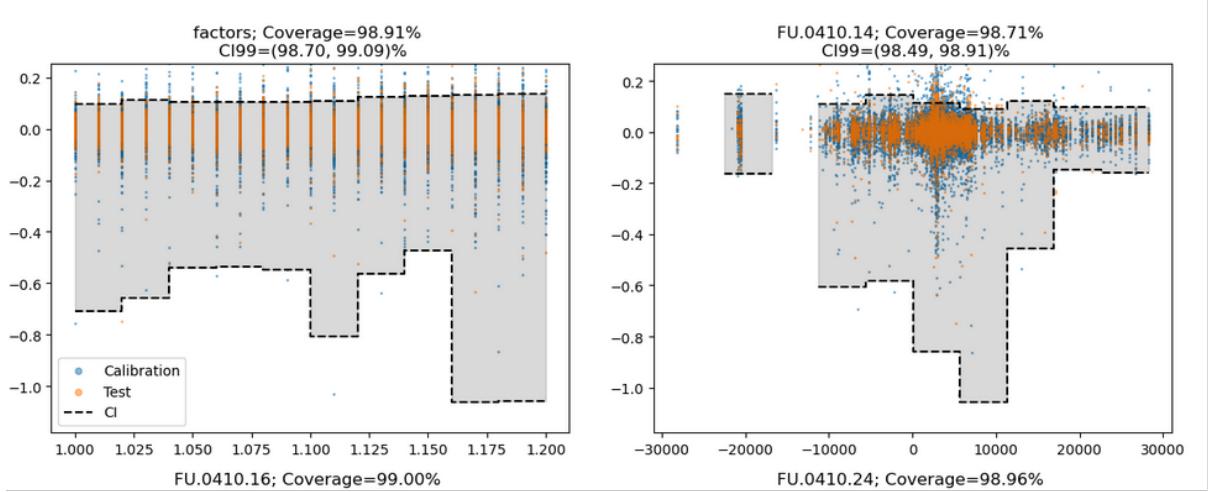
Accordingly (vid. [Figure 3.30](#)),

- For each point in the test set with input variables (x_1, \dots, x_n) and output prediction y , the the larger out of all the intervals in the set

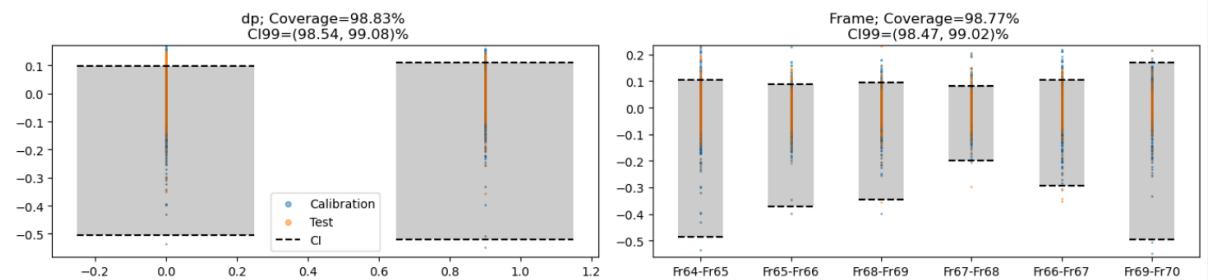
$$\{\text{GUM}, \text{LOUM}(y), \text{LIUM}(x_1), \text{LIUM}(x_2), \dots, \text{LIUM}(x_n)\}$$

is selected and it is checked whether the error associated to this point lies inside the corresponding interval.

- The error coverage is computed, accordingly.
- The bootstrap error coverage is computed, as per previous sections, and the final output is the bootstrap error coverage and its 95% confidence interval, along with a markdown message about whether the FUM is well calibrated.



(a) Numerical inputs: "FU.0410.14" (left), "FU.0410.15" (right). White slices correspond to bins which have not fulfilled the minimum bin size requirements. For such bins, the GUM is used instead of LIUM.



(b) Categorical inputs: "dp" (left), "Frame" (right)

Figure 3.29: LIUM coverage on the evaluation set. Residue of the output variable "RF Forced Crippling". The grey area represents the uncertainty CI calculated by the LIUM for the corresponding bin.

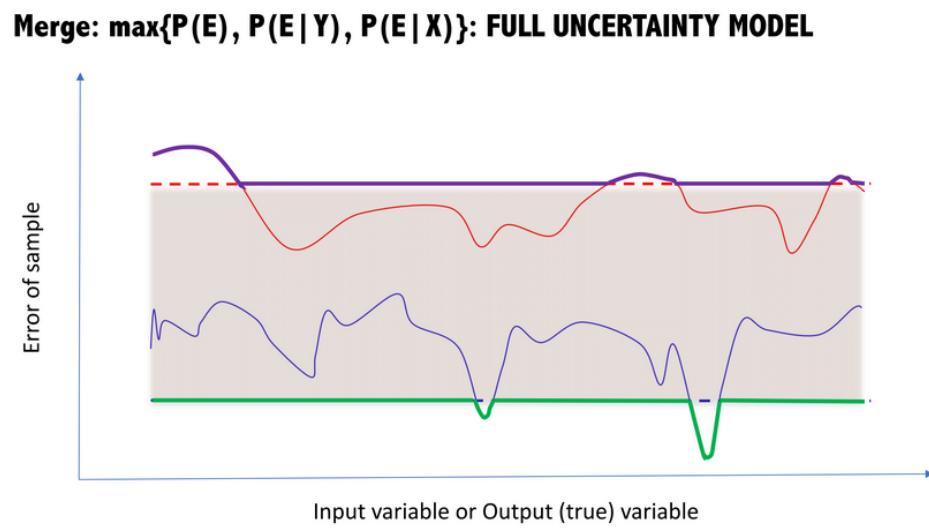


Figure 3.30: Conceptual visualization of FUM as a conservative merge between GUM (whose bootstrap interval [P2.5,P97.5] is represented by the two straight horizontal lines), LOUM and LIUM (whose bootstrap intervals [P2.5,P97.5] are represented indistinctively by the red and blue curves).

Conclusions and research outlook

This work presents a validation pipeline tailored to address the challenges inherent in certifying neural network models across various industrial domains. While the focus has been emphasized on its applications within the aerospace industry, it's crucial to underscore the pipeline's inherent versatility and adaptability to a wide range of industrial problems and solution methodologies.

Versatility Across Industries: Although the validation pipeline showcased in this work has been exemplified within the aerospace sector, its principles and methodologies are applicable to diverse industrial contexts. The structured approach to addressing regression problems with neural networks offers a universal framework that can be adapted to different domains.

Practical Utility: Through the practical application of the validation pipeline using an industrial surrogate model, this work demonstrates its efficacy in addressing real-world challenges. By illustrating its application to structural calculation and failure mode prediction in aeronautical structures, the pipeline's potential to enhance safety and efficiency becomes apparent, regardless of the specific industry context.

Flexibility and Scalability: Chapter 3 elucidates the comprehensive nature of the validation pipeline, emphasizing its flexibility and scalability to accommodate diverse problem domains and evolving technological landscapes. By providing a detailed examination of each component's role, the pipeline offers a customizable framework that can be tailored to meet the specific needs and challenges of different industries.

Future Directions: The potential applications of the validation pipeline presented here are vast and promising, extending beyond the confines of the aerospace industry. Future developments could explore leveraging the capabilities of physics-informed neural networks or PINNs[48]. Furthermore, there is an opportunity to explore direct prediction of aerodynamic loads, showcasing the versatility of the pipeline in tackling complex multidimensional problems across various industrial sectors.

While this work primarily focuses on supervised learning paradigms, the scope can be ex-

panded to encompass other learning methodologies. For instance, avenues for exploration include the application of reinforcement learning[49] techniques to refine and optimize decision-making processes within the validation pipeline. Additionally, the incorporation of collective learning approaches holds promise for leveraging distributed intelligence and collaborative problem-solving across diverse datasets and domains.

In essence, the future development and application of the validation pipeline are poised to unlock new frontiers in industrial innovation and efficiency. By embracing emerging technologies and methodologies, such as PINNs and reinforcement learning, and fostering collaboration across disciplines, researchers and industry practitioners can continue to push the boundaries of what is possible in the realm of neural network validation and application.

In conclusion, this advocates for the widespread adoption of rigorous validation methodologies and the responsible utilization of neural networks across industrial domains. By embracing the principles of the proposed validation pipeline and fostering collaboration among industry stakeholders, organizations can ensure the reliability, safety, and transformative potential of neural network-driven systems in an ever-evolving industrial landscape.

Bibliography

- [1] Y. Xu et al. “Machine-Learning-Assisted Optimization of Aircraft Trajectories Under Realistic Constraints”. In: *Journal of Guidance, Control, and Dynamics* (2023), pp. 1–12 (cit. on p. [xix](#)).
- [2] B. Shukla, I.-S. Fan, and I. Jennions. “Opportunities for explainable artificial intelligence in aerospace predictive maintenance”. In: *PHM Society European Conference*. Vol. 5. 1. 2020, pp. 11–11 (cit. on p. [xix](#)).
- [3] P. Adhikari, H. G. Rao, and M. Buderath. “Machine learning based data driven diagnostics & prognostics framework for aircraft predictive maintenance”. In: *Proceedings of the 10th International Symposium on NDT in Aerospace, Dresden, Germany*. 2018, pp. 24–26 (cit. on p. [xix](#)).
- [4] P. Korvesis. “Machine learning for predictive maintenance in aviation”. PhD thesis. Université Paris Saclay (COmUE), 2017 (cit. on p. [xix](#)).
- [5] E. Force and A. Daedalean. “Concepts of design assurance for neural networks (codann) ii”. In: *Concepts of Design Assurance for Neural Networks (CoDANN)*. EASA, 2021 (cit. on pp. [xix](#), [18](#)).
- [6] E. A. I. Roadmap. “EASA Concept Paper: First usable guidance for Level 1 machine learning applications”. In: (2021) (cit. on pp. [xix](#), [18](#)).
- [7] A. Henderson, S. Harbour, and K. Cohen. “Toward Airworthiness Certification for Artificial Intelligence (AI) in Aerospace Systems”. In: *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*. IEEE. 2022, pp. 1–10 (cit. on p. [xix](#)).
- [8] J.-G. Durand, A. Dubois, and R. J. Moss. “Formal and Practical Elements for the Certification of Machine Learning Systems”. In: *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*. IEEE. 2023, pp. 1–10 (cit. on p. [xix](#)).
- [9] K. Dmitriev, J. Schumann, and F. Holzapfel. “Toward certification of machine-learning systems for low criticality airborne applications”. In: *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*. IEEE. 2021, pp. 1–7 (cit. on p. [xix](#)).

- [10] H. El Mir and S. Perinpanayagam. “Certification of machine learning algorithms for safe-life assessment of landing gear”. In: *Frontiers in Astronomy and Space Sciences* 9 (2022), p. 896877 (cit. on p. [xix](#)).
- [11] S. Paul et al. *Assurance of Machine Learning-Based Aerospace Systems: Towards an Overarching Properties-Driven Approach*. Tech. rep. United States. Department of Transportation. Federal Aviation Administration, 2023 (cit. on p. [xix](#)).
- [12] S. Le Clainche et al. “Improving aircraft performance using machine learning: a review”. In: *Aerospace Science and Technology* (2023), p. 108354 (cit. on p. [xx](#)).
- [13] N. Hong and L. Li. “A data-driven fuel consumption estimation model for airspace redesign analysis”. In: *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*. IEEE. 2018, pp. 1–8 (cit. on p. [1](#)).
- [14] S. Pfingstl et al. “Warped Gaussian processes for predicting the degradation of aerospace structures”. In: *Structural Health Monitoring* 22.4 (2023), pp. 2531–2546 (cit. on p. [1](#)).
- [15] S. Marsland. *Machine Learning: An Algorithmic Perspective*. 2nd ed. Boca Raton, USA: Chapman & Hall/CRC, 2015 (cit. on pp. [2](#), [4](#), [6](#), [7](#), [21](#)).
- [16] F. P. Preparata and M. I. Shamos. “Convex Hulls: Basic Algorithms”. In: *Computational Geometry: An Introduction*. New York, NY: Springer New York, 1985, pp. 95–149. ISBN: 978-1-4612-1098-6. DOI: [10.1007/978-1-4612-1098-6_3](https://doi.org/10.1007/978-1-4612-1098-6_3). URL: https://doi.org/10.1007/978-1-4612-1098-6_3 (cit. on p. [7](#)).
- [17] D. Barrett et al. “Measuring abstract reasoning in neural networks”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 511–520. URL: <https://proceedings.mlr.press/v80/barrett18a.html> (cit. on p. [7](#)).
- [18] B. M. Lake and M. Baroni. “Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks”. In: *CoRR* abs/1711.00350 (2017). arXiv: [1711.00350](https://arxiv.org/abs/1711.00350). URL: [http://arxiv.org/abs/1711.00350](https://arxiv.org/abs/1711.00350) (cit. on p. [7](#)).
- [19] D. Saxton et al. “Analysing Mathematical Reasoning Abilities of Neural Models”. In: *CoRR* abs/1904.01557 (2019). arXiv: [1904.01557](https://arxiv.org/abs/1904.01557). URL: [http://arxiv.org/abs/1904.01557](https://arxiv.org/abs/1904.01557) (cit. on p. [7](#)).

- [20] T. Ebert, J. Belz, and O. Nelles. “Interpolation and extrapolation: Comparison of definitions and survey of algorithms for convex and concave hulls”. In: *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. IEEE. 2014, pp. 310–314 (cit. on p. 7).
- [21] W.-Y. Loh, C.-W. Chen, and W. Zheng. “Extrapolation errors in linear model trees”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.2 (2007), 6–es (cit. on p. 7).
- [22] P. Klesk. “Construction of a Neurofuzzy Network Capable of Extrapolating (and Interpolating) With Respect to the Convex Hull of a Set of Input Samples in R”. In: *IEEE Transactions on Fuzzy Systems* 16.5 (2008), pp. 1161–1179. DOI: [10.1109/TFUZZ.2008.924337](https://doi.org/10.1109/TFUZZ.2008.924337) (cit. on p. 7).
- [23] R. Balestriero, J. Pesenti, and Y. LeCun. “Learning in high dimension always amounts to extrapolation”. In: *arXiv preprint arXiv:2110.09485* (2021) (cit. on pp. 7, 8, 23).
- [24] I. Bárány and Z. Füredi. “On the shape of the convex hull of random points”. In: *Probability theory and related fields* 77 (1988), pp. 231–240 (cit. on p. 8).
- [25] L. Bonnasse-Gahot. “Interpolation, extrapolation, and local generalization in common neural networks”. In: *arXiv preprint arXiv:2207.08648* (2022) (cit. on pp. 8, 23, 28).
- [26] Tosaka. *Airframe. 4 Types*. [https://commons.wikimedia.org/wiki/File:Airframe_\(4_types\).PNG](https://commons.wikimedia.org/wiki/File:Airframe_(4_types).PNG). Accessed 22-08-2022. 2009 (cit. on p. 12).
- [27] P. Bijlaard. “On the Buckling of Stringer Panels Including Forced Crippling”. In: *Journal of the Aeronautical Sciences* 22.7 (1955), pp. 491–501 (cit. on p. 12).
- [28] Z. Li et al. “A survey of convolutional neural networks: analysis, applications, and prospects”. In: *IEEE transactions on neural networks and learning systems* (2021) (cit. on p. 16).
- [29] L. R. Medsker and L. Jain. “Recurrent neural networks”. In: *Design and Applications* 5.64-67 (2001), p. 2 (cit. on p. 16).
- [30] L. Prechelt. “Early stopping-but when”. In: *Neural Networks: Tricks of the trade*. Springer, 2002, pp. 55–69 (cit. on p. 17).
- [31] L. Taylor and G. Nitschke. “Improving deep learning with generic data augmentation”. In: *2018 IEEE symposium series on computational intelligence (SSCI)*. IEEE. 2018, pp. 1542–1547 (cit. on p. 19).

- [32] H. Hotelling. “Analysis of a complex of statistical variables into principal components.” In: *Journal of educational psychology* 24.6 (1933), p. 417 (cit. on p. 20).
- [33] B. Rosner and D. Grove. “Use of the Mann–Whitney U-test for clustered data”. In: *Statistics in medicine* 18.11 (1999), pp. 1387–1400 (cit. on p. 26).
- [34] R. Velez Ibarrola and A. Garcia Perez. *Calculo de probabilidades y Estadistica Matematica*. 1st ed. Madrid, Spain: Universidad Nacional de Educacion a Distancia, 1994 (cit. on p. 28).
- [35] D. Zhang. “A coefficient of determination for generalized linear models”. In: *The American Statistician* 71.4 (2017), pp. 310–316 (cit. on p. 33).
- [36] J. D. Jobson. *Applied multivariate data analysis: regression and experimental design*. Springer Science & Business Media, 2012 (cit. on pp. 33, 50).
- [37] G. Chen, J. R. Gott, and B. Ratra. “Non-Gaussian Error Distribution of Hubble Constant Measurements”. In: *Publications of the Astronomical Society of the Pacific* 115.813 (2003), p. 1269 (cit. on p. 36).
- [38] P. Pernot, B. Huang, and A. Savin. “Impact of non-normal error distributions on the benchmarking and ranking of Quantum Machine Learning models”. In: *Machine Learning: Science and Technology* 1.3 (2020), p. 035011 (cit. on p. 36).
- [39] D. Smyl et al. “Learning and correcting non-Gaussian model errors”. In: *Journal of Computational Physics* 432 (2021), p. 110152 (cit. on p. 36).
- [40] L. Chai et al. “Using generalized Gaussian distributions to improve regression error modeling for deep learning-based speech enhancement”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.12 (2019), pp. 1919–1931 (cit. on p. 36).
- [41] M. C. Jones and A. Pewsey. “Sinh-arcsinh distributions”. In: *Biometrika* 96.4 (2009), pp. 761–780 (cit. on pp. 40, 41).
- [42] B. Rosner. “Percentage points for a generalized ESD many-outlier procedure”. In: *Technometrics* 25.2 (1983), pp. 165–172 (cit. on p. 40).
- [43] B. Efron. “Bootstrap methods: another look at the jackknife”. In: *Breakthroughs in statistics: Methodology and distribution*. Springer, 1992, pp. 569–593 (cit. on p. 42).
- [44] E. B. Wilson. “Probable inference, the law of succession, and statistical inference”. In: *Journal of the American Statistical Association* 22.158 (1927), pp. 209–212 (cit. on p. 45).

- [45] T. K. Kim. “Understanding one-way ANOVA using conceptual figures”. In: *Korean journal of anesthesiology* 70.1 (2017), pp. 22–26 (cit. on p. 50).
- [46] B. R. Kirkwood and J. A. Sterne. *Essential medical statistics*. John Wiley & Sons, 2010 (cit. on p. 51).
- [47] Y. Fujikoshi. “Two-way ANOVA models with unbalanced data”. In: *Discrete Mathematics* 116.1-3 (1993), pp. 315–334 (cit. on p. 54).
- [48] M. Raissi et al. “Deep learning of vortex-induced vibrations”. In: *Journal of Fluid Mechanics* 861 (2019), pp. 119–137 (cit. on p. 81).
- [49] L. P. Kaelbling, M. L. Littman, and A. W. Moore. “Reinforcement learning: A survey”. In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285 (cit. on p. 82).