Approximation Algorithms for Rapid Evaluation and Optimization of Architectural and Civil Structures

by

Stavros Tseranidis

BS, MS in Civil Engineering, Department of Civil Engineering – Polytechnic School

Aristotle University of Thessaloniki, 2014

Submitted to the School of Engineering

in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Computation for Design and Optimization

at the

Massachusetts Institute of Technology

September 2015

Signature of Author:.....

School of Engineering
August 7, 2015

Certified by:.....

Caitlin T. Mueller
Assistant Professor, Building Technology
Thesis Supervisor

Accepted by:.....

Nicolas G. Hadjiconstantinou
Professor, Mechanical Engineering
Co-Director, Computation for Design and Optimization (CDO)

Approximation Algorithms for Rapid Evaluation and Optimization of Architectural and Civil Structures

by

Stavros Tseranidis

Submitted to the School of Engineering
on August 7, 2015 in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computation for Design and Optimization

Abstract

This thesis explores the use of approximation algorithms, sometimes called surrogate modelling, in the early-stage design of structures. The use of approximation models to evaluate design performance scores rapidly could lead to a more in-depth exploration of a design space and its trade-offs and also aid in reducing the computation time of optimization algorithms. Six machine-learning-based approximation models have been examined, chosen so that they span a wide range of different characteristics. A complete framework from the parametrization of a design space and sampling, to the construction of the approximation models and their assessment and comparison has been developed. New methodologies and metrics to evaluate model performance and understand their prediction error are introduced. The concepts examined are extensively applied to case studies of multi-objective design problems of architectural and civil structures. The contribution of this research lies in the cohesive and broad framework for approximation via surrogate modelling with new novel metrics and approaches that can assist designers in the conception of more efficient, functional as well as diverse structures.

Key words: surrogate modelling, conceptual design, structural design, structural optimization

Thesis supervisor: Caitlin T. Mueller

Title: Assistant Professor of Building Technology

Acknowledgements

First and foremost, I would like to sincerely thank my advisor, Professor Caitlin Mueller, for giving me such an interesting topic and guiding me throughout this research. The knowledge I acquired from her through our meetings, discussions and her courses at MIT are extremely valuable to me. She provided me with new insights and perspectives on structural engineering, design, computational approaches and algorithms. Her course titled “Computation for Structural Design and Optimization” has been one of my most meaningful and thought-provoking experiences at MIT. For all that I am truly grateful.

I would like to thank Professor Jerome Connor and Doctor Pierre Ghisbain who gave me the opportunity to be a Teaching Assistant in their courses and trusted me with such an important task. Collaborating with them has been a great honor for me.

I would also like to thank Professor John Ochsendorf for welcoming me to his group and exposing me to new fields in structural engineering and beyond.

Finally, I would like to thank all the students in the Digital Structures Group and the Structural Design Lab. Especially, Nathan Brown for providing me with his datasets, which were crucial for my research.

Table of Contents

List of Mathematical Symbols

SYMBOL	MEANING
$\boldsymbol{X}$	Matrix *n*X*p* containing the input variables' values
$\boldsymbol{Y}$	Vector of dimension n containing the original, actual output values
$\boldsymbol{h}$	Vector of dimension n containing the predicted output values
y_i	Actual output value *i*
h_i	Predicted output value *i*
n	Number of samples; design observations
p	Number of features; explanatory variables

Abbreviations

NN – Neural Networks

RF – Random Forests

RBFN – Radial Basis Function Networks

RBFNE – Radial Basis Function Networks Exact

MARS – Multivariate Adaptive Regression Splines

KRIG – Kriging Regression

MSE – Mean Squared Error

RMSE – Root Mean Squared Error

MRE – Mean Rank Error

TMRE – Top Mean Rank Error

TRE – Top Ratio Error

TFE – Top Factor Error

REM – Rank Error Measures

AAE – Average Absolute Error

MAE – Maximum Absolute Error

RAAE – Relative Average Absolute Error

RMAE – Relative Maximum Absolute Error

MOO – Multi-Objective Optimization

1. Introduction

The engineering design process is a demanding endeavor. It is a mix of analytical tools with human intuition. The design of new structures lies on the same category. It is complex; it involves multiple objectives and numerous parameters. The work on this thesis contributes in this area and has the goal to empower designers to achieve more efficient structures, by using a type of approximation technique called surrogate modelling.

1.1 Structural optimization

To the extent of the feasible and the realistic, optimization is strived for in engineering design. In structural engineering, specifically, optimization is a process that involves many physical constraints, such as construction, material production and transportation, cost, safety and many more. At the same time, in the discipline of architecture, more abstract considerations are prioritized in design; functionality, aesthetics and human psychology just to name a few. Both disciplines are combined in practice, sharing the same goals to produce a single result.

Unlike other engineering disciplines, the optimization objective goals and constraints are not always easily quantified and expressed in equations, but rather require human intuition and initiative to materialize. For this reason, structural optimization has yet to reach its full potential. With modern computers and tools available, this potential is beginning to be realized. There are a few examples where it has been successfully applied to buildings, such as in projects by Skidmore Owings & Merrill (SOM) [1], shown in Figure 1. To illustrate the breadth of designs for braced frame systems, another one by Neil M. Denari Architects is shown in Figure 2 (High Line 23, New York City [2]). While this design is not structurally optimized, its architectural success is closely linked to its structural system and geometry. So, in the structural design optimization, there needs to be a balance between quantitative and qualitative objectives.

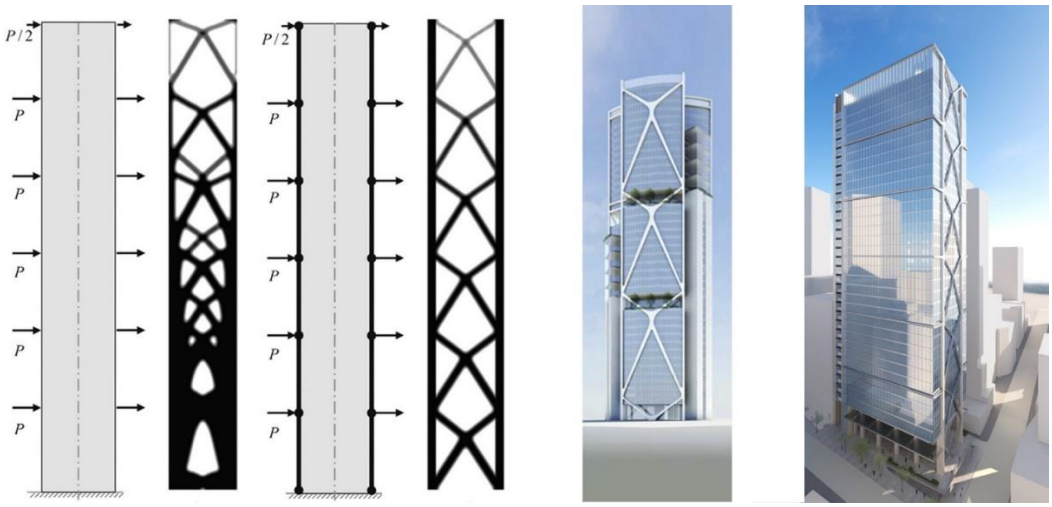


Figure 1: Optimized braced frame system, SOM (Image from [1])

Figure 2: Braced frame system, Denari Architects (Image from [2]*)*

On the one hand is the quantitative aspect of optimization. Structural evaluation simulations as well as energy simulations, which compute the energy consumed by a building, often require significant computational power, increasing with the complexity and size of the project. Thus, optimization algorithms can usually require substantial execution time. To put this in context, for energy simulations included in the case studies, the time required for one sample evaluation was approximately 25 sec, which impedes a free-flowing design process. This fact slows down the design process. On the other hand, because of the qualitative nature of this type of design with the hard-to-quantify considerations as described above, many iterations are required and the result is never likely to come up from a single optimization run. In practice, the combination of slow simulation and problem formulation challenges means that optimization is rarely used in the design of architectural and civil structures. In fact, even quantitatively comparing several design alternatives can be too time-consuming, resulting in poor exploration of the design space and likely a poorly performing design.

1.2 Need for computational speed

The exploration of the design problem and various optimal solutions should ideally happen in real time, so that the designer is more productive. Research has shown that rapid response time can result in significant productivity and economic gains [3]. “Improved individual productivity is perhaps the most significant benefit to be obtained from rapid response time” [3]. The upper threshold for computer response time for optimal productivity has been estimated at 400ms and is commonly referred to as the Doherty threshold [3]. This threshold was originally developed in the 1980s for system response of routine tasks, like typing. Today, however, software users expect similarly rapid response for any interactions with the computer, even those that require expensive calculations like performance simulation. Immediate

response from the computer can benefit not just rote productivity, but also creative thinking. The concept of *flow* is used in psychology to describe "completely focused motivation"; when a person gets fully immersed and productive. It was thoroughly studied by M. Csikszentmihalyi [4] and among the components for someone to experience *flow*, "immediate feedback" is listed.

The first way to implement the "immediate feedback" effect in computer response is by increasing the available computational power. This can be achieved by either increasing the processing power of a computer or by harnessing parallel and distributed computing capabilities. The other way is to use different or improved algorithms. This thesis focuses on the second approach, investigating algorithms that improve computational speed for design-oriented simulation through approximation.

1.3 Surrogate modelling

Among possible approximation algorithms, this thesis considers surrogate modelling algorithms, a class of machine learning algorithms, and their use in making computation faster and allowing for more productive exploration and optimization in the design of buildings.

Machine learning is about creating models about the physical world based only on available data. The data are either gathered though physical experiments and processes, or by computer-generated samples. Those samples are then fit to an approximation mathematical model, which can then be used directly as the generating means of new representative data samples. In surrogate modelling specifically, the data is collected from simulations run on the computer.

Similar techniques are being used successfully in many other engineering disciplines, but have not been studied and applied extensively to the architectural and structural engineering fields. This research investigates these techniques and evaluates them on various related case studies. Focus is given in the development of a holistic framework that is generalizable.

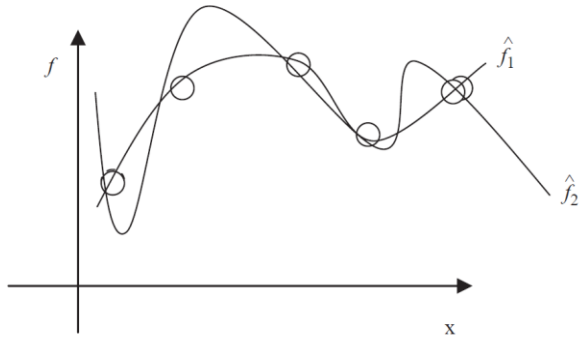


Figure 3: "Many surrogates may be consistent with the data" (Figure from [5]*)*

Figure 3 displays a core concept in surrogate modelling. The circles represent the available data, with many different models being able to fit them. The art in surrogate modelling is to choose the one that will also fit new data well.

1.4 Research question

This thesis considers two key questions:

Is surrogate modelling a viable methodology to use in this field?

The main focus of this thesis is to examine the use of surrogate modelling in the design of architectural and civil structures as a method to rapidly explore the design space and obtain more efficient solutions. It applies it in case study problems and tests its applicability and effectiveness.

How robust is surrogate modelling?

Questions of how good an approximation method is are addressed. Also considered are matters of how accurate a model is and how its error value and variability can be estimated. Methods for this purpose are proposed and applied.

1.5 Organization of thesis

First, a literature review of the existing research in surrogate modelling, its application in structural engineering problems, model types and error assessment methods is outlined in Chapter 2. Chapter 3 is an overview of the main features of the methodology framework used, including assumptions and descriptions of the models used. Sampling and outlier removal techniques are also discussed. Chapter 4 is dedicated to explaining the error assessment and visualization methods used throughout the rest of the thesis. Chapter 5 introduces the proposed method for robust model comparison, whose use is then illustrated through the case studies in Chapters 6 and 7. Specifically, Chapter 6 introduces the case study problems, the parameters examined and the analysis assumptions, while Chapter 7 presents all the numerical results obtained for comparing model performances and assessing them individually for all the case study problems. The original contributions, findings and future considerations are summarized in Chapter 8. Finally, in the Appendix, a sample of the outputs of the developed MATLAB framework is presented for one case study (Airport terminal) and for one performance score approximation (Energy Overall).

2. Background

To avoid a computationally expensive simulation, one approach is to construct a physical model that is simpler and includes more assumptions than the original. This process is very difficult to automate and generalize and requires a high level of expertise and experience in the respective field. A more general approach is to substitute the analytical simulation with an approximate model (surrogate) that is constructed based purely on data. This approach is referred to as data-driven or black-box simulation. The reason is that the constructed approximation model is invariant to the inner details of the actual simulation and analysis. The model has only "seen" data samples that have resulted from an "unknown" process, thus the name black-box. This thesis addresses data-driven surrogate modelling.

The two main areas in which surrogate modelling can be applied are for optimization and design space exploration. Specifically, an approximation (surrogate) model can be constructed as the main evaluation function for an optimization routine or just in order to explore a certain design space in its entirety, better understand variable trade-offs and performance sensitivity. For optimization, it is often used when there are more than one optimization objectives, thus called multi-objective optimization (MOO), and the computational cost of computing them is significant.

2.1 Surrogate modelling on structural designs

Several years ago, when the computational power was significantly less than that of today, one available today, scientists started to explore the possibility of adapting approximation model techniques in intensive engineering problems. One of the first attempts of this kind in the field of structural engineering by Schmit and Miura [6] in a NASA report in 1976. A review of the application of approximation methods in structural engineering was published by Barthelemy and Haftka in 1993 [7]. The methods explored in this review paper are response surface methodology (RSM) as well as neural networks (NN). It was mentioned that more methods will emerge and the practice is going to expand. In fact, today, although the computational power has increased exponentially from twenty years ago, the engineering problems that designers face have also increased dramatically in scale and therefore surrogate modelling has been studied and applied extensively.

Hajela and Berke wrote a paper in 1992 [8] solely dedicated to an overview of the use of neural networks in structural engineering problems. They mentioned that this approximation technique could be useful in the more rapid evaluation of simulations such as non-linear structural analysis. Neural networks and approximation models have still great potential in this field today, when non-linear structural analysis is very frequently performed. Researchers have been using approximation algorithms in the structural engineering field for various problems such as for the dynamic properties and response evaluation and optimization of structures [9], for seismic risk assessment [10] and for energy MOO simulations [11]. Energy simulations are extensively examined in this thesis, since they are usually extremely expensive computationally, and at the same time their use and importance in building and infrastructure design is increasing today.

The use of approximation algorithms in conceptual architectural/structural design was very interestingly examined by Swift and Batill in 1991 [12]. Specifically, for truss problems, with the variables being the positions of some nodes of the truss and the objective the structural weight, a design space was sampled and later approximated using neural networks. A representative figure from this paper can be seen in Figure 4, where the initial base structure and the variable ranges are shown in Figure 4a, while the best designed obtained from the neural network approximation is on Figure 4b.

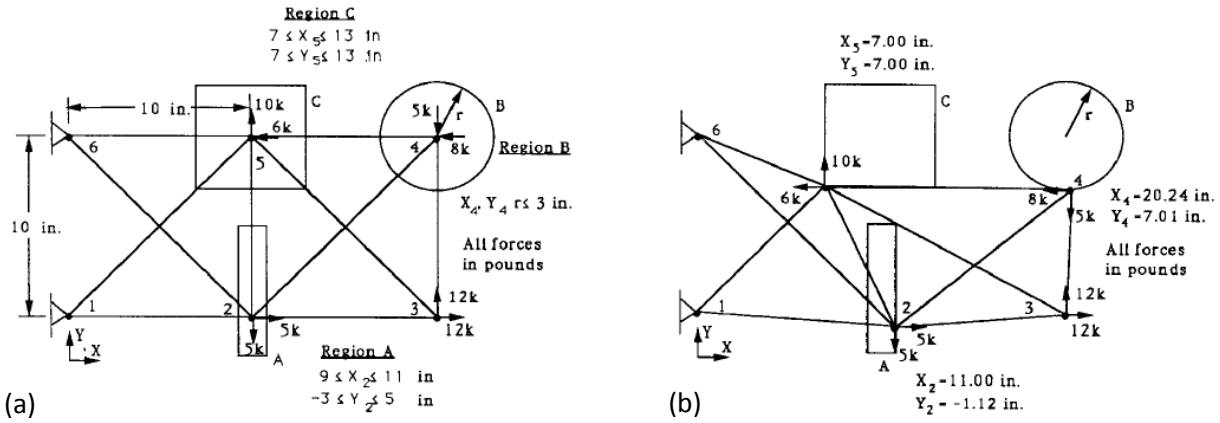


Figure 4: Ten-bar truss (a) design space and (b) best design resulting from NN model (Image from [12])

A similar approach was followed by Mueller [13], with a seven-bar truss problem examined being shown in Figure 6a. The variables were again the positions of the nodes and specifically the vertical nodal positions as shown along with their ranges in Figure 6a. The design space (with the structural weight as the objective score) computed analytically, without approximation is shown in Figure 6b. The approximated design space for different models is shown in Figure 6. Details on the approximation models used later are presented in the following section on model types.

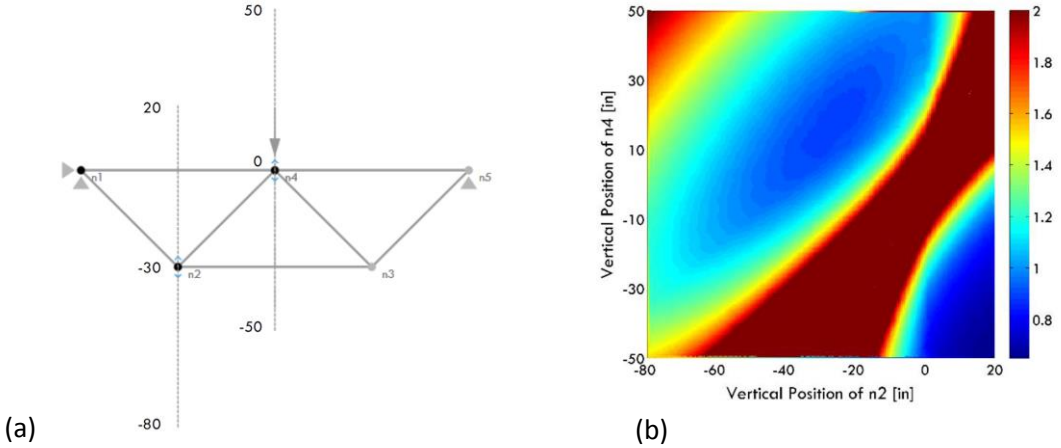


Figure 5: Seven-bar truss (a) variables and (b) analytically computed design space (Image from [13])

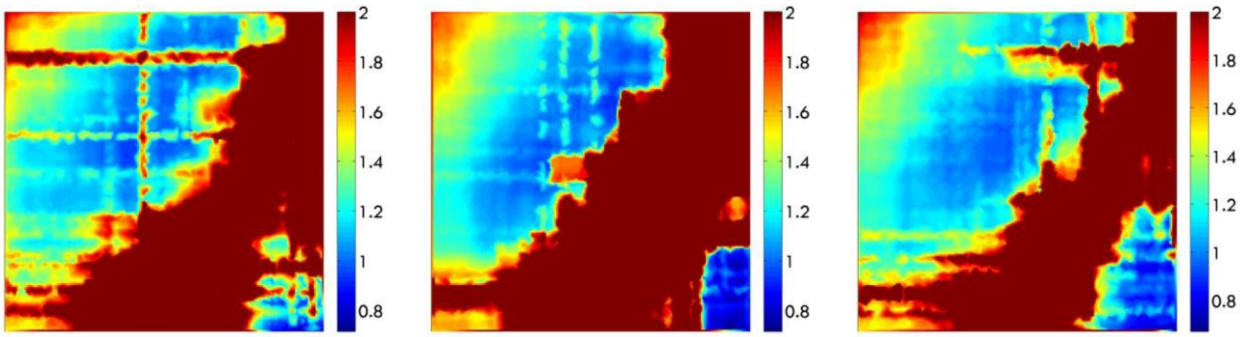


Figure 6: Seven-bar truss approximated design space for different parameters (Image from [13]*)*

It is also worth mentioning that surrogate modelling is being used extensively in the aerospace industry. The basic principles remain the same across disciplines since the methodology relies solely on data. Queipo et al. [5] have made a thorough overview of the common practices of surrogate modelling. They also applied those techniques in an MOO problem from the aerospace industry. Another comprehensive survey of black-box approximation method for use in high-dimensional design problems is included in [14].

There exist attempts of integrating performance evaluation into parametric design in architectural and civil structures. Mueller and Ochsendorf [15] considered an evolutionary design space exploration, Shi and Wang [16] examined performance-driven design from an architect's perspective, while Granadeiro et al. [17] studied the integration of energy simulations into early-stage building design. All of those interesting approaches could benefit by the use of surrogate modelling, which is the main contribution of this thesis.

2.2 Model types

Several methods have been developed over the years to approximate data and have been used in surrogate modelling applications. Very common ones include polynomial regression (PRG) and response surface methodology (RSM) [18], in which a polynomial function is fitted to a dataset using least squares regression. This method has been used in many engineering problems [5].

One of the most widely used surrogate modelling method in engineering problems is Kriging (KRIG). Since it was formally established in the form it is used today [19], it has been applied extensively ([10], [5], [20]). Gano et al. [21] compared Kriging with 2^{nd} order polynomial regression. Chung and Alonso [20] also compared 2^{nd} order RSM and Kriging for an aerospace case study and concluded that both models performed well and are pose indeed a realistic methodology for engineering design.

Another very popular model type are artificial neural networks (referred as NN in this thesis). Extensive research has been performed on this type of model ([5], [8]). Neural networks are greatly customizable and their parameters and architecture are very problem specific.

A special type of neural network is called radial basis function network (RBFN) and was introduced by Broomhead and Lowe [22] in 1988. In this network, the activation function of each neuron is replaced by

a gaussian bell curve function. A special type of RBFN imposes the Gaussian radial basis function weights such that the networks fits the given data with zero error. This is referred to as radial basis function network exact (RBFNE) and its main drawback is the high possibility that the network will not generalize well on new data. Those two types of models, RBFN and RBFNE, are explained in more detail in Chapter 3 as they are studied more extensively in this thesis.

Radial basis functions can also be used to fit high dimensional surfaces from given data. This model type is called RBF [23] and is different from the RBFN model. RBF models can also be referred to as gaussian radial basis function models [5]. Kriging is similar to RBF, but it allows more flexibility in the parameters.

Multivariate adaptive regression splines (MARS) is another type of model. This performs a piecewise linear or cubic multidimensional fit to a certain dataset [24]. It can be more flexible and capture more complex datasets, but requires more time to construct.

Jin et al. [25] performed a model comparison for polynomial regression, Kriging, MARS and RBF models. They used 13 mathematical problems and 1 engineering one to perform the comparisons. Many other references for papers that performed comparisons between those and other models are also included in [25]. An important feature outlined in this paper was that it set five major aspects to compare the models. Those were accuracy, robustness (ability to make predictions for problems of different sizes and type), efficiency (computational time to construct model), transparency (ability of the model to provide information about variable contribution and interaction) and conceptual simplicity. Those issues are examined in following sections in the present thesis.

Finally, the existing MATLAB-based framework SUMO [26], implements support vector machines (SVM) [27] (a model type frequently used for classification), Kriging and neural network models, along with the sampling and has been used in many applications such as RF circuit modelling and aerodynamic modelling [26].

A framework with NN, Random Forests (RF) which have not extensively been applied in structural engineering problems, RBFN, RBFNE, MARS and KRIG models was developed and tested in case study problems in the current thesis to extend the existing research and available methodologies for structural design.

There is a lack of extensive model comparison and their application on problems for structural engineering and building design specifically; this thesis addresses this need to move beyond existing work.

2.3 Error in surrogate models

The most important error required to assess an approximation model is what is called its generalization error. This is more thoroughly examined in the following section about robustness. In the current section, ways to quantify a model's performance on a given set of data are discussed. When the actual performance calculated from the analytic simulation is known for a set of data, and the respective performance from an approximation model is calculated, then the error of the predicted versus the actual performance can be calculated by many different measures. All the following error measures are computed between the actual (represented by y) – and predicted (represented by h) values.

One of the most common ones is R^2, which refers to the correlation coefficient of the actual with the predicted values. A value closer to 1 indicates better fit. This is extensively discussed in this thesis in Chapter 4 about error. Other common measures are the Mean Squared Error (MSE) and its root, the Root Mean Squared Error (RMSE). The Average Absolute Error (AAE) and the Maximum Absolute Error (MAE) are other options, along with Relative Average Absolute Error (RAAE) and the Relative Maximum Absolute Error (RMAE). MAE is generally not correlated with R^2 or AAE and it can indicate whether the model does not perform well only in a certain region. The same holds true for RMAE, which is not necessarily correlated with R^2 or RAAE. However, R^2, RAAE and MSE are usually highly correlated [25], which makes the use of more than one of them somewhat redundant. Gano et al. [21] used R^2, AAE and MAE for the model comparisons they studied, while Jin et al. [25] used R^2, RAAE and RMAE.

The above mentioned error metrics are summarized along with their formulas on Table 1.

	Error metric	Formula
1	MSE	$\frac{\sum_{i=1}^{n}(y_i - h_i)^2}{n}$
2	RMSE	$\sqrt{\frac{\sum_{i=1}^{n}(y_i - h_i)^2}{n}}$
3	R^2	$1 - \frac{\sum_{i=1}^{n}(y_i - h_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$
4	AAE	$\frac{\sum_{i=1}^{n}\lvert y_i - h_i\rvert}{n}$
5	RAAE	$\frac{\sum_{i=1}^{n}\lvert y_i - h_i\rvert}{n \cdot STD(y)}$
6	MAE	$\max(\lvert y_1 - h_1\rvert, \lvert y_2 - h_2\rvert, \dots, \lvert y_i - h_i\rvert)$
7	RMAE	$\frac{\max(\lvert y_1 - h_1\rvert, \lvert y_2 - h_2\rvert, \dots, \lvert y_i - h_i\rvert)}{\mathrm{STD(y)}}$

Table 1: Common surrogate modelling error metrics (y: actual, h: predicted value)

Error measures which provide a more direct and comprehensive quantitative model performance metric are lacking, and some alternative approaches to address this are presented in this thesis. Error measures based on a model's performance on the rank of the samples [13] are also studied. Special focus is also given to the visualization of the results and it is argued that normalization and visualization can have a significant impact in understanding error and are problem specific.

2.4 Robustness in surrogate modelling

As mentioned in the beginning of the previous section, it is crucial for a surrogate modelling application to have an acceptable generalization error. This refers to an error estimate of the model on new data. In this context, new data means data samples that have not been used at any point in the construction of the model. One can realize that this is indeed the most important error required since the rapid generation of accurate new data performance is the main objective of the construction of the surrogate model in the first place.

To estimate the generalization error, several techniques exist. Those are explained in detail in Queipo et al. [5] and Viana et al. [28]. The simplest one is to split a given dataset into train and test data, construct the model with the train data and then compute the error in the test data and take this as an estimate of the generalization error. Another technique is called cross-validation (CV), in which the original dataset is split into k parts and the model is trained with all the parts except one, which is used as the test set of the previous case. The procedure is then repeated until each one of the k sets has served as the test set. By taking the mean of the test set errors, a more robust generalization error estimate is produced. Another advantage is that a measure of this error's variability can be obtained by taking, for example, the standard deviation of the computed k test set errors. If this procedure is repeated the same number of times as the number of samples in the original dataset, which means that only a single sample is used every time as the test set, then this measure is called PRESS and the method leave-one-out cross validation [28]. The last method to obtain a robust generalization error measurement is through Bootstrapping. According to the known definition of the bootstrap (sample with replacement), a certain number of bootstrap samples (datasets) are created as training and test sets. Then the error estimate and its variability calculation procedure is similar to the CV method. For the Bootstrapping method to produce accurate results, a large number of subsamples is usually needed [28].

This thesis attempts a combination of the aforementioned techniques frequently used in the surrogate modelling context with the common practice of machine learning applications (train/validation/test set partition) to obtain a measure of robustness as well as accuracy.

Another way to interpret robustness is to consider it as the capability of the approximation model to provide accurate fits for different problems. This again can be measured by the variance of accuracy and error metrics. However, the scope of this thesis is to examine the deployment of approximation models for case study design problems and not to comment on a model's more general predictive ability regarding its mathematical properties.

Finally, to increase robustness, one could use ensembles of surrogate models in prediction [5]. This means that several models are trained and their results are averaged with a certain scheme to obtain a prediction. Models of this type are Random Forests (RF), which are studied in this thesis and explained in more detail in the next chapter.

2.5 Unmet needs, open questions

As described in the previous sections, the field of surrogate modelling in engineering design is very rich. However, its use in the design of architectural and civil structures is limited compared to other engineering disciplines. There is a need for a comprehensive study of its use in this area to examine whether they can be feasible and realistic in practice. The thread of their use in conceptual structural design was left in 1992 [12] and picked up recently [13]. An extension of the study in this field is necessary, since the advantages that rapid exploration and optimization can have in early stage structural design could be significant. While many model types have been investigated, few have been applied to real problems in this field.

There is also a field within a field in error estimation and visualization of approximation models, which needs to further be explored. Specifically, what other types of error metrics and visualization techniques can be used in surrogate modelling applications are among the questions this thesis will address. A main concept that is addressed and pointed out throughout the thesis is that the prediction results of a model should be visualized instead of just obtaining an error metric value.

Model robustness considerations are also examined thoroughly, proposing a methodology that is focused on approximating data by combining ideas from various surrogate modelling/machine learning contexts and has the goal of broad applicability and scalability in architectural and civil structure design applications.

3. Methodological framework

This chapter outlines in detail the basic components used throughout this research, the proposed methodology and the case studies, all thoroughly explained in the following chapters. In general, the framework developed is based on sampling a design scape in the first place and then constructing and assessing the surrogate models. For the sampling part, the Rhino software and the Grasshopper plugin were used. They are parametric design tools very broadly used in architectural design. As for the surrogate modelling part, the framework and all of the analysis was performed in MATLAB. References to the specific functions and special capabilities of the software are placed in context in the text.

3.1 Surrogate modelling procedure

The basic surrogate modelling procedure consists of three phases; training, validation and testing. A separate set of data is needed for each of those phases. During the training phase, a model is fit into a specific set of data, the training set. The fitting process refers to the construction of the mathematical model; the determination of various weighting factors and coefficients. In the next phase, the trained model is used on a different set of data, the validation set, and its prediction error on this set is computed. The first two steps of training and validation are repeated several times with different model parameters. The model that produced the minimum error on the validation set, is then chosen for the final phase of testing. During testing, another dataset, the test set, is used to assess the performance of the model chosen from the first two steps (minimum validation set error).

The steps are shown schematically in Figure 7. Each model type can have multiple parameters which define it. Those are referred to as nuisance parameters, or simply parameters in the following chapters. Different nuisance parameters can result in different levels of model fit and accuracy. Choosing the best nuisance parameters for a given model type is the goal of the validation phase as described in the previous paragraph. The test phase is for verification of the model's performance on a new set of data, never previously used in the process (training or validation). More details can be found in [27].

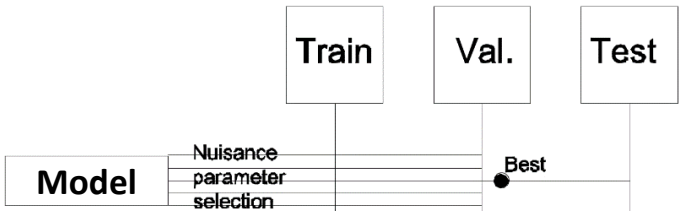


Figure 7: Surrogate modelling procedure

3.2 Model types

The utilization of approximation models, referred to as surrogate models or machine learning algorithms in different disciplines, aims at prediction. A model is essentially a procedure that acts on input data and outputs a prediction of a physical quantity. Previously computed or measured values of the physical quantity at hand, along with the corresponding input variables, are used to create/train the model, which can afterwards be used to make rapid predictions on new data. There are numerous different surrogate modelling algorithms and architectures. In the following section, the models examined in the present thesis are introduced. Lists of the parameters affecting each model which were considered are also included. All the model parameters considered and MATLAB functions used are summarized in Table 10.

3.2.1 Neural Networks (NN)

The human brain consists of billions of neurons connected together. Signals of different intensities are transmitted throughout this network. All the input signals to a neuron are summed together and if the sum exceeds a threshold value, then the specific neuron is triggered. The neural network architecture observed in biological procedures has been studied and has been adapted as a mathematical construct, forming what are known as Artificial Neural Networks. This computational model was firstly proposed in 1943 [29] and has been refined over the years.

The typical single-layer neural network architecture is shown in Figure 8. Multiple hidden layers can be inserted in the architecture.

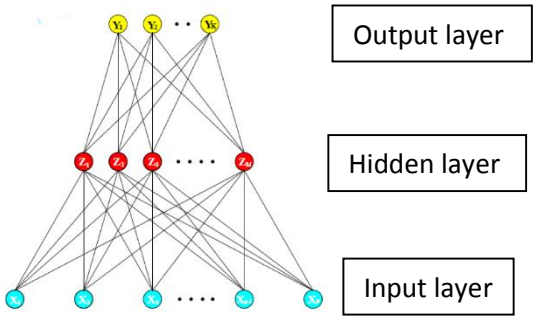


Figure 8: Single-layer neural network architecture (Image from [27]*)*

The procedure to obtain a prediction from a single-layer neural network is the following:

The input vector X is multiplied by the respective weights of the hidden layer connections and the sum is obtained for each neuron. Then at each neuron, this sum is passed through an "activation" function. The procedure is repeated again for the output layer and the results from each output neuron is the prediction vector. Equation 1 describes the calculation procedure in a single neuron.

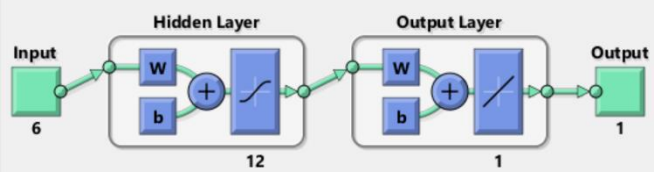
$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X), m = 1, \dots, M$$

Equation 1: Single neuron calculation [27]

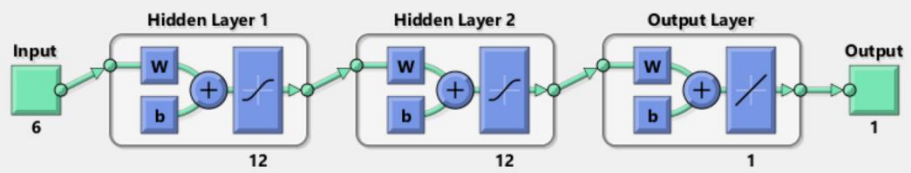
For each neuron the input vector X is multiplied by the respective weights of each connection and the results are summed (with the addition of a constant "bias" term α_{0m}). This sum is then passed through an "activation function" σ(x), which for the results of the present thesis was chosen to be the tan-sigmoid function.

This procedure is made for each neuron in the hidden layer and then all the outputs from that layer are used in the same way as an input vector to the next layer. For the final "output" layer, the "activation" function chosen hereafter was just a linear function with slope 1. The training of the network constitutes the process of determining the weights of the connections so that it performs in a given accuracy on a known training set of X and Y. Many different training, or learning, algorithms exist for neural networks. This type of network is called Feed-Forward and the implementation from MATLAB used in the current thesis is the function `feedforwardnet` [30].

For simplicity and clarity, the output of the network (and any other model examined) was chosen to be always a scalar, thus making the collection of the outputs a vector. This is why Y (the original/actual values) is a vector.



(a)



(b)

Figure 9: Single (a) and double (b) hidden layer neural network architectures used

In Figure 9, the neural network architectures used in the present thesis in MATLAB are shown. There are 6 input variables and a single output. In this figure, there are 12 neurons in each hidden layer.

A common problem with neural networks is overfitting. This means that the model has adapted to the training data with too much precision, but fails to perform equally well to predicting from new observations. To address this the neural network must not be trained to match the training set exactly, but with some tolerance. In the training of the network in MATLAB, independent of the training algorithm, the error of a validation set is computed at each step. If the error on that validation set does not improve more than a threshold for a specified number of training steps (referred to as Maximum validation checks in Table 2), or epochs, then the training stops. This validation set is a subset of the training set passed to the algorithm and is different than the validation set used to choose between the nuisance parameters of the network. The ratio of that internal partitioning ratio of the training set into training and validation is

another parameter of the network's training as summarized in Table 2. The number of neurons was assumed equal for each layer and thus considered as one parameter.

	Parameter
1	Number of neurons
2	Number of layers
3	Training function
4	Maximum validation checks
5	Internal ratio of training data
6	Internal ratio of validation data
7	Internal ratio of test data

Table 2: Neural Network parameters considered

Since for the methodology developed for the present thesis and used in all the case studies, there is a separate test set to assess a model's performance, the "Internal ratio of test data" parameter of Table 2 was set to zero.

3.2.2 Random Forests (RF)

Classification and regression trees (CART) are a type of model that uses sequential splitting of the data in a tree-like structure. The splits are made so that classification or regression error is minimized. For prediction, a sample is passed through the tree and gets the output of the corresponding final-level tree leaf that it results in lying [27]. An extension of the CART model are Random Forests. Random Forests were introduced by Breiman and Cutler [31] in 2001. It is a technique similar to bagging; an ensemble of decision trees. In bagging, not a single tree is grown but several and the most frequent output from each tree is chosen for prediction (classification) or the average of the results from each tree (regression). The main difference of RF from bagging is that each tree split happens on a random subset of the input/explanatory variables and not to all of them. The number of variables to pick for the split is a nuisance parameter of the model called "Number of variables to sample" on Table 3.

As in ensemble tree bagging, the number of trees to grow is also an important parameter of the model. The random forest training algorithm grows the specified number of trees and then uses an average rule to make a prediction from the outputs of all the trees. For each single tree, a bootstrap sample is drawn from the training set. The size of that sample is a nuisance parameter. Then only a subdivision of the input/explanatory variables is used to make a division as described previously, with the division process repeated until a threshold is reached. That threshold could be the minimum number of observations per tree leaf, which is parameter "Minimum leaf observations" on Table 3. Random Forests can be used for either regression or classification, but in the present thesis it has been used only for regression.

Random Forests are averaging many unbiased smaller models (trees), which lowers the variance and thus prevents overfitting.

The implementation class in MATLAB is called `TreeBagger` [32] and the nuisance parameters of the model examined are all outlined on .

	Parameter
1	Number of trees
2	Number of variables to sample
3	Bootstrap sample size ratio
4	Sample with replacement (true/false)
5	Minimum leaf observations

Table 3: Random Forest parameters considered

Random Forests(tm), or RF(tm) is a trademark of Leo Breiman and Adele Cutler.

3.2.3 Radial Basis Function Networks (RBFN)

One can conceptually think of Radial Basis Function Networks (RBFN) as neural networks for which the "activation function" of each neuron is actually a Gaussian radial basis function. The point on which the Gaussian basis function of each neuron is centered is the result of the training process. Essentially, each neuron captures and outputs how similar the input is to the vector on which the neuron is centered at. The standard deviation of each neuron (considered constant for all neurons in this research) determines the spread of influence of that neuron and is a tuning parameter of the network.

This type of network was firstly introduced by [22] and the implementation used for the case studies presented is in MATLAB with the `newrb` function [33]. The nuisance parameters of the implementation of RBFN are shown in Table 4. All the other network parameters were kept at the default values [33].

The training process of the network starts with no neurons. Then the network is simulated for the training set data and neurons are added at each step starting by matching the input vector that had the greatest error on the previous simulation while also adjusting the weights to minimize error overall. The MSE goal threshold that stops the training is a parameter of the network.

A representation of the architecture of a RBFN is shown in Figure 10. The outputs from the network are schematically shown as being 2, but it is again noted that throughout the current manuscript, there is always a single output for all the case studies.

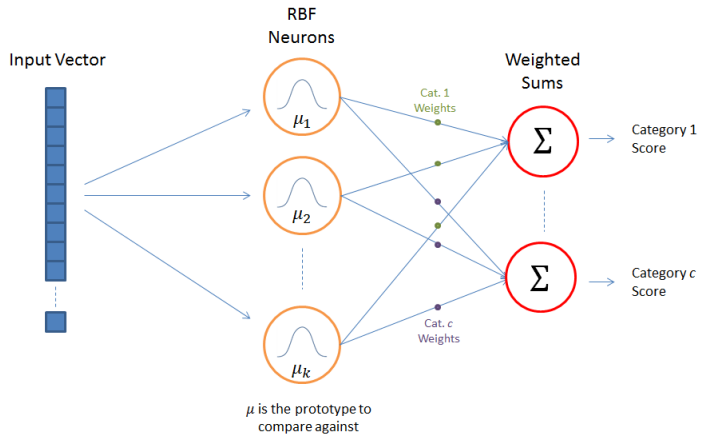


Figure 10: RBFN network architecture (Image from [34])

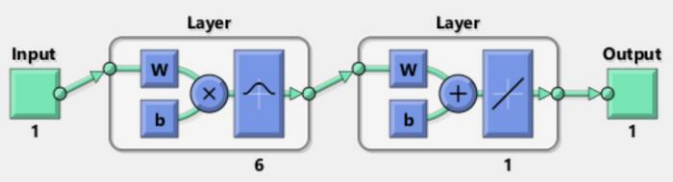


Figure 11: RBFN architecture MATLAB

	Parameter
1	Mean squared error goal
2	Spread of radial basis functions
3	Maximum number of neurons

Table 4: RBFN parameters considered

3.2.4 Radial Basis Function Networks Exact (RBFNE)

A special type of RBFN models, Radial Basis Function Networks Exact (RBFNE) are designed so that they produce zero error on the training set input/output data. The implementation used in the framework was MATLAB's `newrbe` function [35]. For the zero training error to be achieved, a special training process of the network is used. Specifically, the single layer of Gaussian radial basis functions is assigned weights $Xtrain^T$ and its biases are set to 0.826/spread so that all the radial basis functions cross 0.5 at weighted inputs of +/- spread. Then the weights and biases of the output layer are calculated by solving a system of linear equations so that the network matches the outputs of the training set exactly. A more detailed explanation of the algorithm is in [35].

Therefore, the only parameter that RBFNE has is the spread of the radial basis functions of the first layer. Large values of the spread could cause numerical problems and result in the RBFNE not having zero error on the training set. The straightforward training process of RBFNE and the fact that there is only one

parameter to influence it and to be chosen from the validation set error, makes the training of RBFNE very quick as will be observed in later results.

The main advantage of RBFNE is deployment speed. However, the major drawback is the potential overfitting of the training set. The model's performance must be carefully assessed on a completely separate test set to determine whether it could be used for prediction. This effect is illustrated in Figure 12.

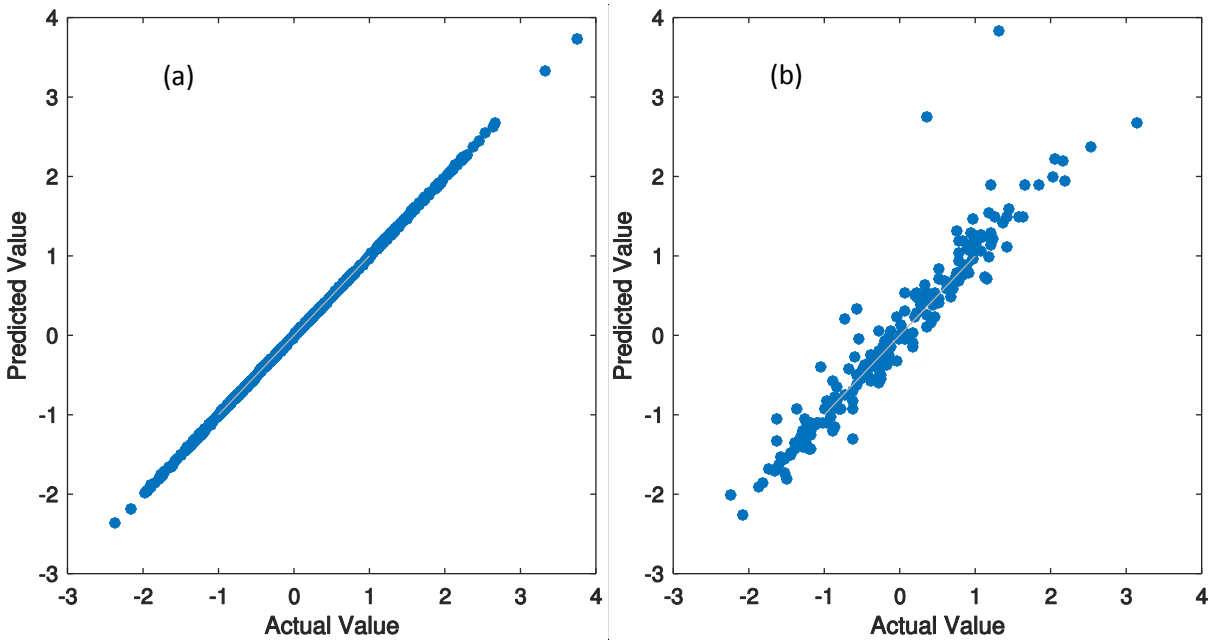


Figure 12: RBFNE (a) Training versus (b) Test performance

	Parameter
1	Spread of radial basis functions

Table 5: RBFNE parameters considered

3.2.5 Multivariate Adaptive Regression Splines (MARS)

MARS is a technique that uses piecewise basis functions in a stepwise training procedure for regression. It was introduced by J.H. Friedman in 1991 [24]. The implementation in MATLAB used in the present thesis can be found on [36].

The basis function types considered are piecewise cubic and piecewise linear. The maximum number of basis functions included in the model was another parameter of the models which was examined here. There exist more parameters which are described in more detail in [24], [27], [36]. For the case studies to follow, the values chosen are outlined for the reproduction of results in Table 7.

	Parameter
1	Piecewise function type (cubic/linear)
2	Maximum number of functions

Table 6: MARS parameters considered

	Parameter	Value
1	Generalized cross-validation (GCV) penalty per knot	3
2	Self Interactions	1 (no interaction)
3	Maximum Interactions	# features * (Self Interactions)
4	Termination threshold	1e-3

Table 7: MARS parameters kept constant

3.2.6 Kriging regression (KRIG)

Kriging is a surrogate modelling method similar to RBFN. A main difference is that in Kriging, the width of the basis functions is allowed to vary for each variable. It is thus more flexible than RBFN. The width of each basis function depends on the correlation of the sample point to which it is centered with the surrounding points and is determined through and optimization routine during the training process.

Kriging was firstly described by Daniel G. Krige in 1951 [37] and Sacks et al. [19]. The method has been implemented by the DACE MATLAB toolbox [38]. The parameters that can be altered are the type of regression functions to use (polynomials of degree 0, 1 or 2) and the correlation function used to adjust the basis functions.

The weights of the basis functions were initialized randomly with values between 0 and 1 at the beginning of the algorithm, provided that in the general case we have no indication of an initial estimate of their value.

	Parameter
1	Degree of polynomial regression function (0/1/2)
2	Correlation function

Table 8: KRIG parameters considered

The different correlation functions examined are outlined in Table 9. Their names are as they appear in the DACE toolbox documentation, where the exact formulas can also be found [38].

	Name
1	EXP
2	GAUSS
3	LIN
4	SPHERICAL
5	CUBIC
6	SPLINE

Table 9: KRIG correlation functions considered

In Figure 13 an example dataset from [38] is shown with different configurations of the parameters to showcase the difference they can make. The black dots are the sampled points used for training the Kriging model and the surfaces are the result of applying the trained model on a fine grid.

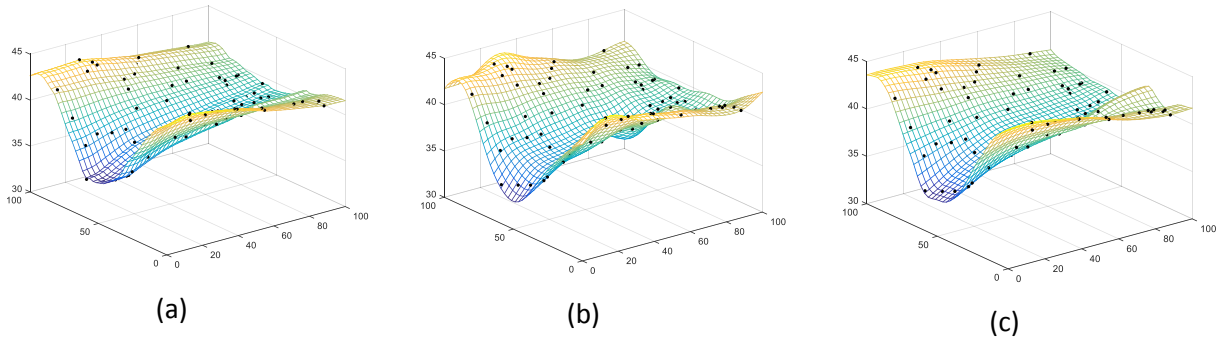


Figure 13: KRIG example; a) regpoly0, correxp b) regpoly1, corrgauss c) regpoly2, corrspline

	NN	RF
1	Number of neurons	Number of trees
2	Number of layers	Number of variables to sample
3	Training function	Bootstrap sample size ratio
4	Maximum validation checks	Sample with replacement (true/false)
5	Internal ratio of training data	Minimum leaf observations
6	Internal ratio of validation data	
7	Internal ratio of test data	
MATLAB	`feedforwardnet` [30]	`TreeBagger` [32]

	RBFN	RBFNE
1	Mean squared error goal	Spread of radial basis functions
2	Spread of radial basis functions	
3	Maximum number of neurons	
MATLAB	`newrb` [33]	`newrbe` [35]

	MARS	KRIG
1	Piecewise function type (cubic/linear)	Degree of polynomial regression function (0/1/2)
2	Maximum number of functions	Correlation function
MATLAB	ARES Lab [36]	DACE Toolbox [38]

Table 10: Summary of models, parameters and MATLAB functions considered

3.3 Normalization

It should be noted that all the datasets used to train and assess the models have been normalized. The normalization scheme used is to make each variable have mean 0 and standard deviation 1. In more detail, once a dataset has been created through simulation, before the partition into train/validation/test set, each input variable column and the output vector was reduced by its mean and then divided by its standard deviation. Normalization is important to bring all the variables in the same range and prevent assigning unrealistic importance and bias towards some variables.

To finally assess the performance of the models, the completely separated test set was used. In order to better comprehend the results and the effect of a model, it was decided to normalize the performance score values (the output for the models) in a manner that the best performing design in the sampled data receives a score of 1 and the rest scale accordingly. For example, a score of 2 performs two times worse than the best design in the test set. A sample test set scatter plot of actual vs. predicted values can be seen in Figure 14 (it is put in context in the "Results" chapter). Apart from the scatter points, two dashed lines are plotted. Those represent the 10% error margins. The score of a point that lies on top of one of those lines was predicted with an error of 10% by the approximation model. The percentage of error is

based on the actual score. Naturally, all the points that lie between these lines have been predicted within a 10% accuracy (this representation is used in [10]). The solid grey line has a slope of 1 and as described in previous sections, represents the ideal exact prediction line. The scale on the two axes is the same.

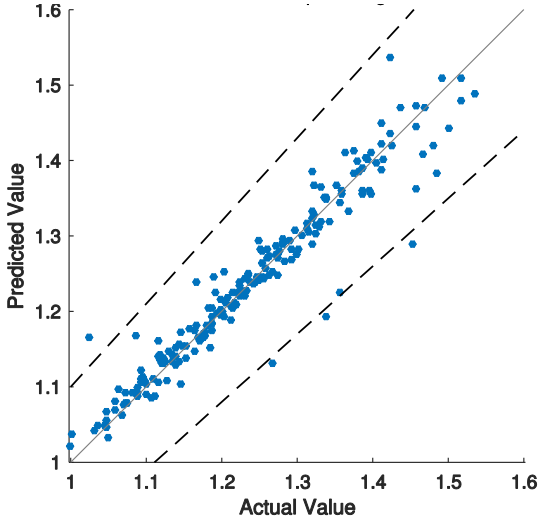


Figure 14: Sample renormalized scatter plot (model: NN)

3.4 Sampling

The first step of any surrogate modelling framework is sampling. It refers to the gathering of a finite number of samples of design space parameters (explanatory variables) with the corresponding objective value (performance measure) for each of those samples. Those samples will be separated into training, validation and test sets for the model development phase.

3.4.1 Grasshopper sampling

For the case studies in the present thesis, the Rhinoceros software [39] was used along with the Grasshopper plugin [40] to generate the parametric design models as well as the objectives. For this reason, a sampling tool was developed for Rhino/Grasshopper since the existing ones were not deemed satisfactory.

The tool includes the following features:

- Sampling design space (Figure 15)
    - Random
    - Grid
    - Latin hypercube
- Evaluating sample objectives
- Storing screenshot for each design

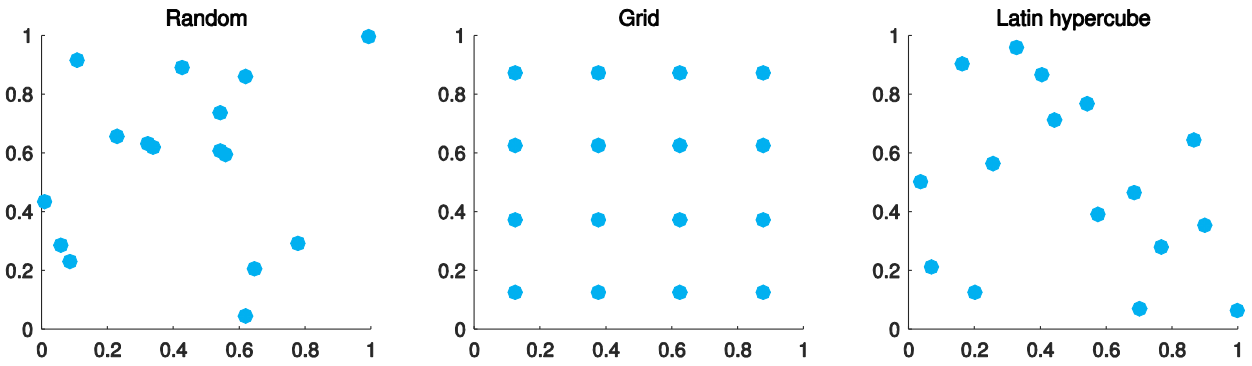


Figure 15: Sampling plans

The three most common sampling plans included in the GH sampling tool are outlined schematically in Figure 15 for 16 samples in two dimensions in the [0,1] range.

The random scheme in this case refers to sampling uniformly random numbers for each variable within a specified range. The grid scheme is very straightforward; the given range for each variable is broken into points with equal distances and then all the sample combinations of those points are taken. The fact that all the combinations have to be sampled, makes the sample size grow geometrically with the number of variables (dimensions) for a given number of equidistant points in each dimension. This is the major drawback of grid sampling. With random sampling, one can take fewer samples than with the grid but there is no guarantee that those samples are going to spread evenly in the design space. The Latin hypercube scheme is a solution to have an evenly spread collection of samples without necessarily having to sample a large number of designs as in the Grid. More details about the Latin hypercube algorithm can be found in [41], which was one of the first papers to study it in depth.

The sampling tool uses the following two existing plug-ins:

- GhPython | [42]
- Hoopsnake | [43]

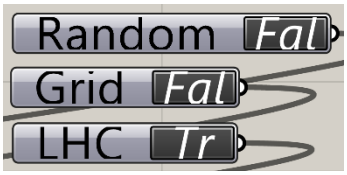
The input sampling parameters of the tool are:

n: total number of samples

k: number of design variables

Sampling scheme: Random/Grid/Latin hypercube

(Only one of the options should be “True”)



Random seed: seed to initialize random number generator, for consistent results

Initially, the sampling is performed for each variable in the range (0,1). There is the capability to impose specific scaling upper/lower bounds for each parameter individually.

The tool has the additional capability to store screenshots of each design sampled. This feature builds on the notion that the framework is to be used in conceptual design, where a visualization of each design is of equal importance as the design parameter data themselves.

The screenshot feature has flexibility over the following parameters as also shown on Figure 16.

- File name: the prefix of the screenshot file names
- Save directory path: the path to which the screenshots will be saved
- File extension: the extension of the file to save the screenshots

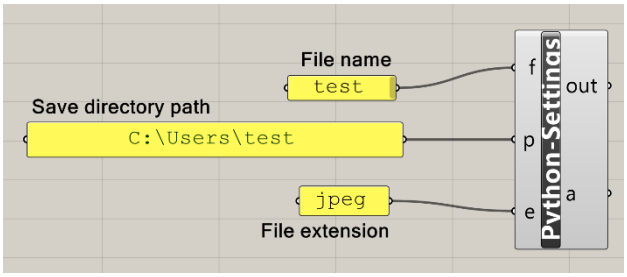


Figure 16: Screenshot tool parameters

3.5 Design example

The first case study, introduced here to illustrate the methodology concepts presented in the following sections, examines a seven (7) bar truss, whose geometry is shown in Figure 17. This is the "initial" geometry of the problem.

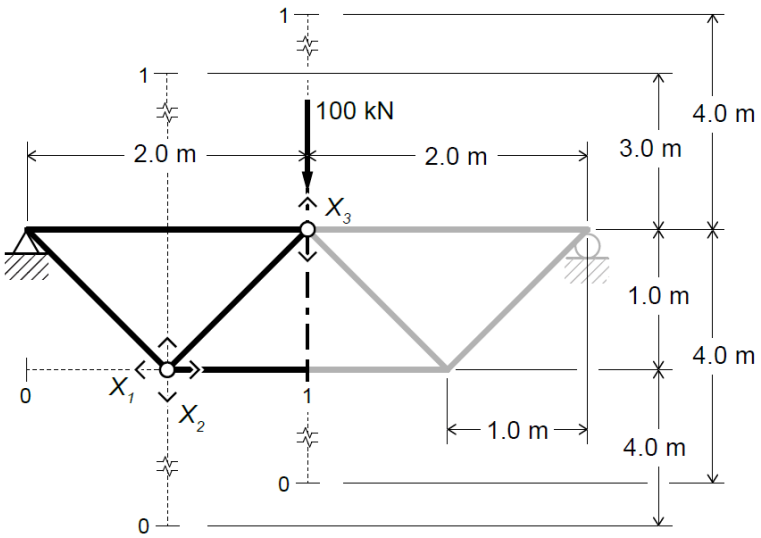


Figure 17: Seven bar truss topology and design variables (Image from [13]*)*

The variables of the problem are x1, x2 and x3 as shown (along with their bounds in Figure 17 [13]). This case study is for research purposes and the methodology is intended for more complicated problems where the structural and/or energy analysis has significant computational needs.

The variables examined were the horizontal – x_1 – and vertical – x_2 – position of one node of the truss as shown in Fig.1 and also the vertical – x_3 – position of the node that lies in the axis of symmetry of truss. It should be noted that the structure is constrained to be symmetric. The design space was sampled based on the three (3) nodal position variables and each design was analyzed for gravity loads (accounting for steel buckling as well, which enhances the nonlinear nature of the problem) and a structural score of the form $\Sigma A_i L_i$ (where A is the section area and L the length of member i) was determined as the objective.

The above problem has been originally addressed and studied and the datasets generated by [13].

Using the developed GH sampling tool described above, data were collected to later fit approximation models from. In Figure 18, 25 designs produced are shown. Each design has a score associated with it, which is used to train an approximation model.

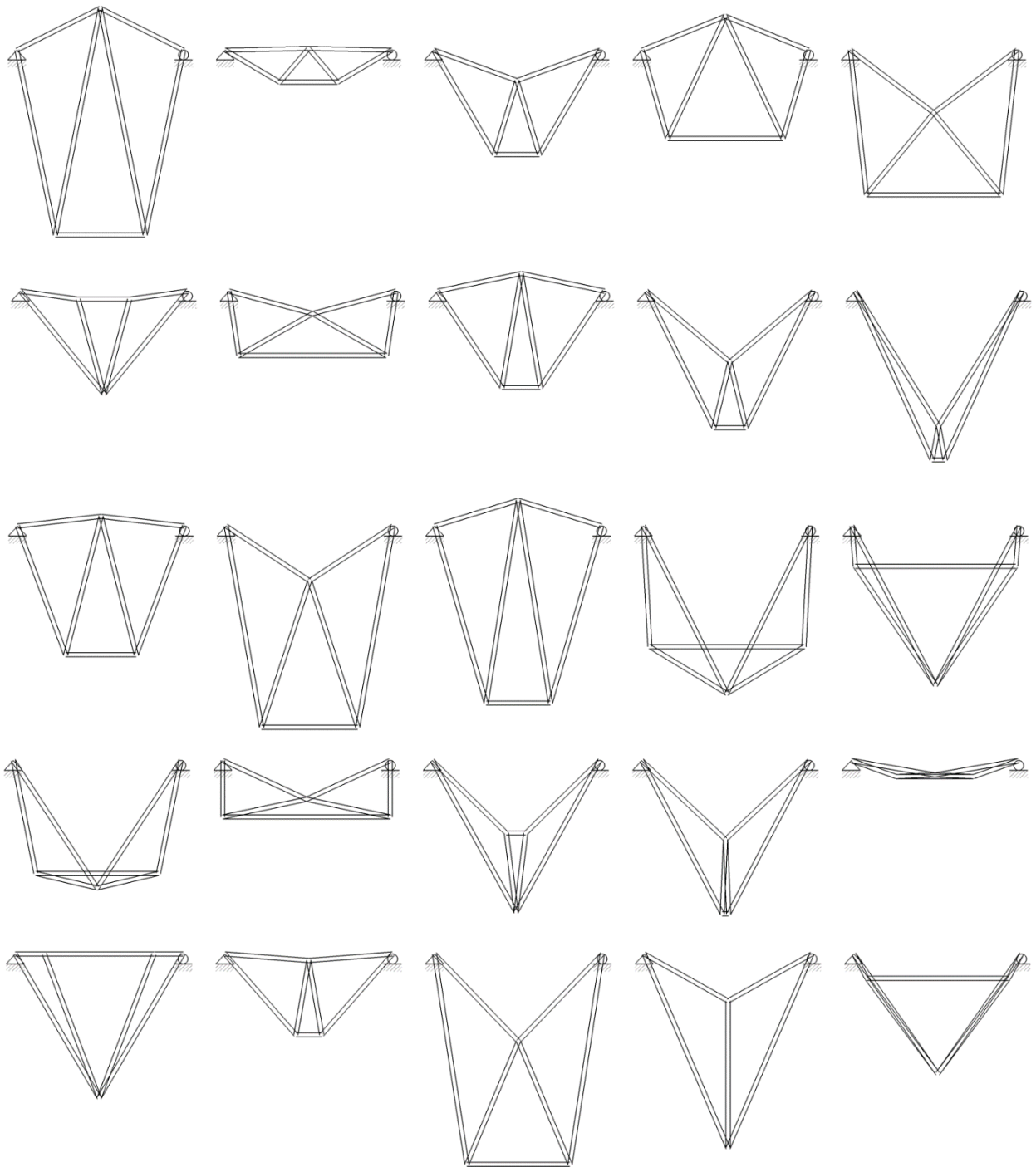


Figure 18: *Design samples of Seven Bar truss generated by GH tool*

3.6 Removing outliers

The presence and the study of outliers is found in many different fields and sciences. Especially in experiment data collection, it is very frequent to discuss this issue and the possibility of removing some samples that showed an extreme behavior. The reasoning in many of those cases is that those samples are not representative of the physical phenomenon studied and have been caused by a malfunction in the equipment or a similar cause. However, that is not the only reason one can consider a certain sample an outlier.

In the more general case, an outlier is a sample that is valid in the physical phenomenon, but is in some way outside the realm of the specific aspects of the phenomenon one desires to study. In structural engineering problems for example, there is usually a threshold in the applied load, displacement or a member's position beyond which equilibrium cannot be achieved and the structure becomes unstable. In this case, one is not interested in studying structural designs that are close to the unstable region, where theoretically equilibrium could be achieved. The effect that this type of outliers on approximation models is illustrated in the seven bar truss example problem (Figure 17) in this section.

The effect of outliers on a model's training and later on its predictive ability can be substantial. The presence of outliers can potentially compromise the performance of a model.

Initially the design space is sampled with a "random" scheme in the broad ranges outlined in Table 11.

Variable	Min	Max
	[m]	[m]
x1	-1	4
x2	-4	4
x3	-4	4

Table 11: Example case study initial sampling ranges

Now if a histogram of the outputs (structural scores) is plotted (Figure 19) we observe that there are some few values that are much larger than the majority of the samples.

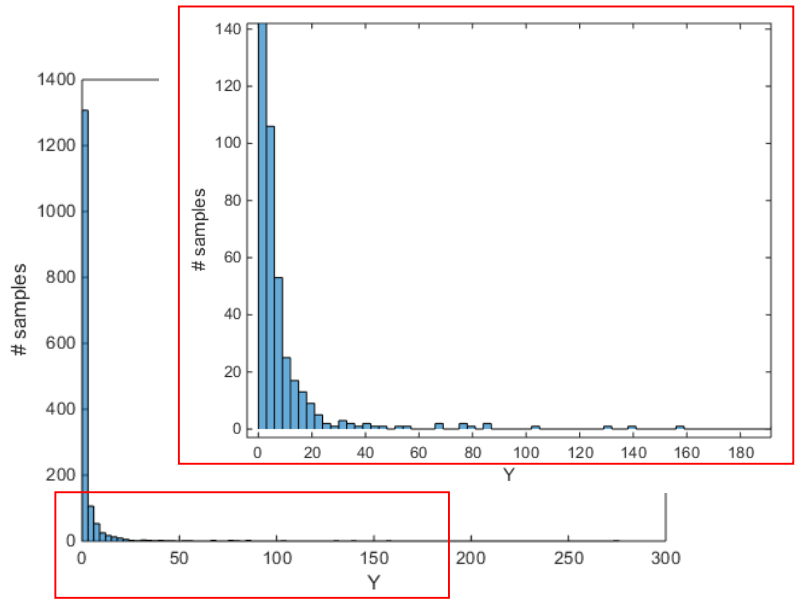


Figure 19: Histogram of outputs (Y) for initial sampling

If for this dataset, a feedforward neural network with a single hidden layer of 6 neurons is trained using 700 training samples then the training fit is displayed in Figure 20a and the performance for a 150-sample test set in Figure 20b. It should be noted that the structural score values are normalized so that the best score has a value of 1.

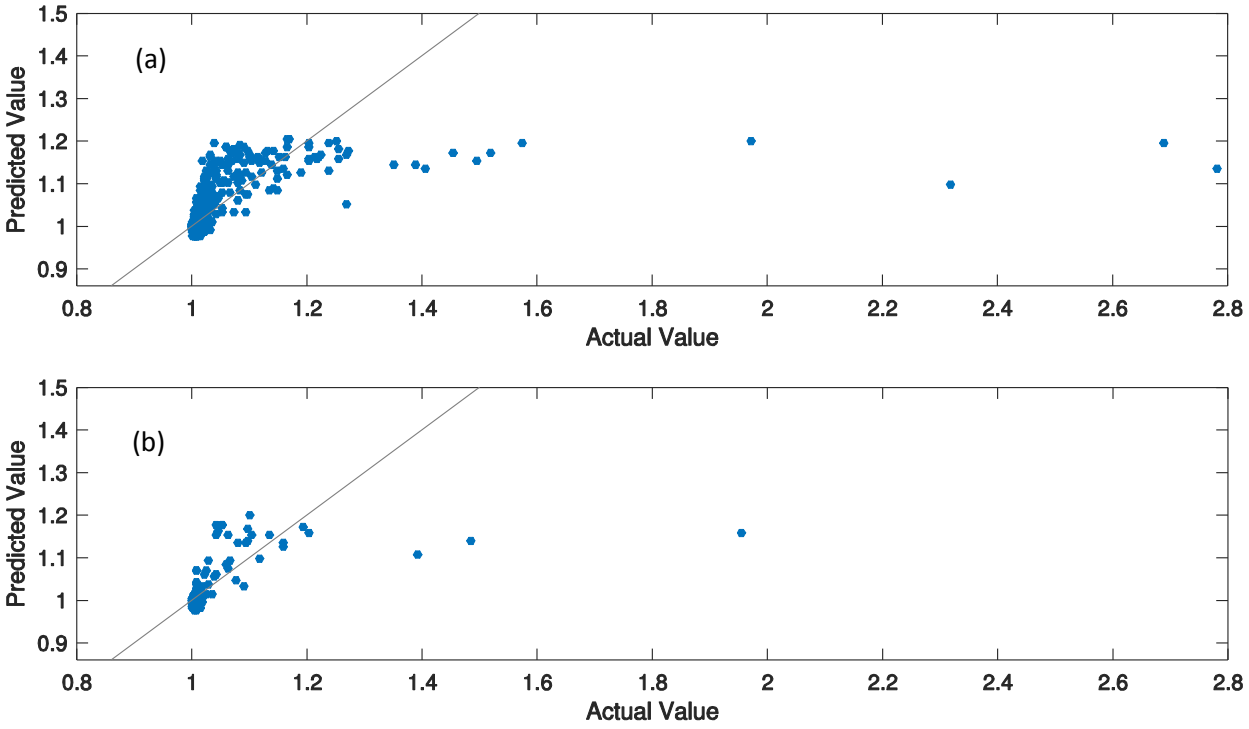


Figure 20: NN performance for dataset containing outliers; (a) Training set (b) Test set

With a closer examination, it can be concluded that the outliers occur because some designs are very close to being structurally unstable (having 3 hinges lying on the same line) and therefore large forces are developed for equilibrium and the structural score surges. Those designs are meaningless in terms of the design space exploration or optimization.

A more rational sampling procedure with more realistic variable ranges that leads in the absence of outliers, can lead to much better approximation performance results. For example, if we modify the sampling ranges and choose those in Table 12, then the same NN as before with 150 training samples and 90 test samples performs much better as seen in Figure 21.

Variable	Min	Max
	[m]	[m]
x1	-1	1
x2	-4	1
x3	-4	1

Table 12: Example case study modified sampling ranges

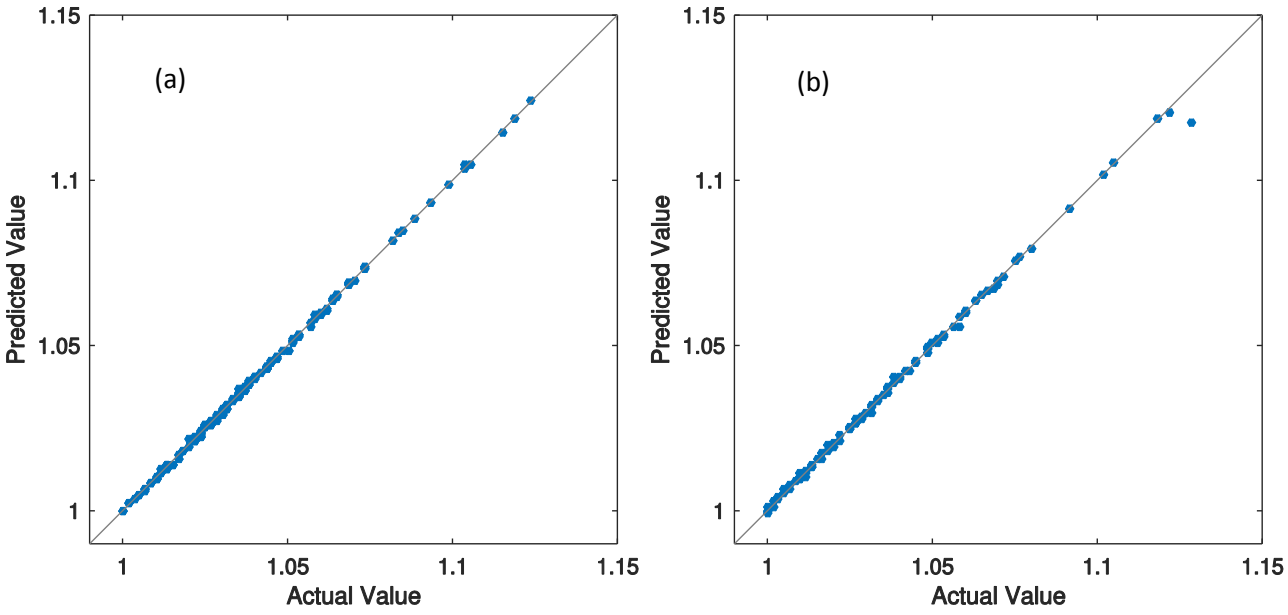


Figure 21: NN performance for modified dataset without outliers; (a) Training set (b) Test set

The designs shown in Figure 18 were generated with a Latin hypercube sampling scheme for the variable ranges in Table 12.

3.7 Summary

This chapter established the groundwork for the next, where new techniques are proposed and case study problems are studied and explained. It started with an overview of the models considered in the developed framework of this thesis, continued by outlining sampling issues and presenting example results from a developed Grasshopper tool for sampling and ended by some preprocessing considerations of the data (outlier removal).

4. Visualization and error

In this section, the crucial issues of visualizing a model's performance and measuring its error is discussed. Several visualization techniques are described and are used in the case studies. An effort was made to describe how to evaluate and understand a model's error through those visualizations. Furthermore, the "flat model" model assessment technique is introduced along with some error metrics based on rank [13].

4.1 Understanding error

When an approximation model is developed, the goal is to make the model "learn" a specific physical process or numerical simulation and be able to make predictions. Therefore, the main evaluation metric of a model is how good it is in making predictions on new data, or in other words, how well it generalizes. The test set has the role of "new" data in a way that it comprises pristine data that were not used at any point in the model's development process (training/validation). The use of a completely separate test set as described before is a very sound way to assess a models performance. In most cases, a test set can be kept when there is an abundance of data at the beginning of the model development process. In case only a relatively small dataset is available (in practice 10-100) then there are techniques such as cross-validation and bootstrap validation to compute a model's generalization error without a separate test set. For the present case studies, the size of the dataset was sufficient and it was decided to use a test set to assess a model's performance.

One of the most frequent ways to visualize a model's performance is to scatter plot the actual values versus the predicted ones from the model (for either the training, validation or test set). In the perfect scenario where the model can perfectly predict the correct values, the points on this plot lie in a straight line with slope 1 and the correlation between actual and predicted values is 1. This type of plot is used the current thesis on the test set data. The more closely the points lie in the slope-1 line, the better the predictive ability of the model.

For the example case study of the seven bar truss as described in Figure 17 if we do a sampling in the range shown in Table 12 from the "initial" geometry, compute a structural score for each sample, partition in train/validation/test sets (Table 13) and train models, we can assess them using the described scatter plots as seen in Figure 22 for RF models (parameters in Table 14) and in Figure 23 for RBFN models (parameters in Table 15).

	# samples
Training set	150
Validation set	50
Test set	90

Table 13: Set sizes

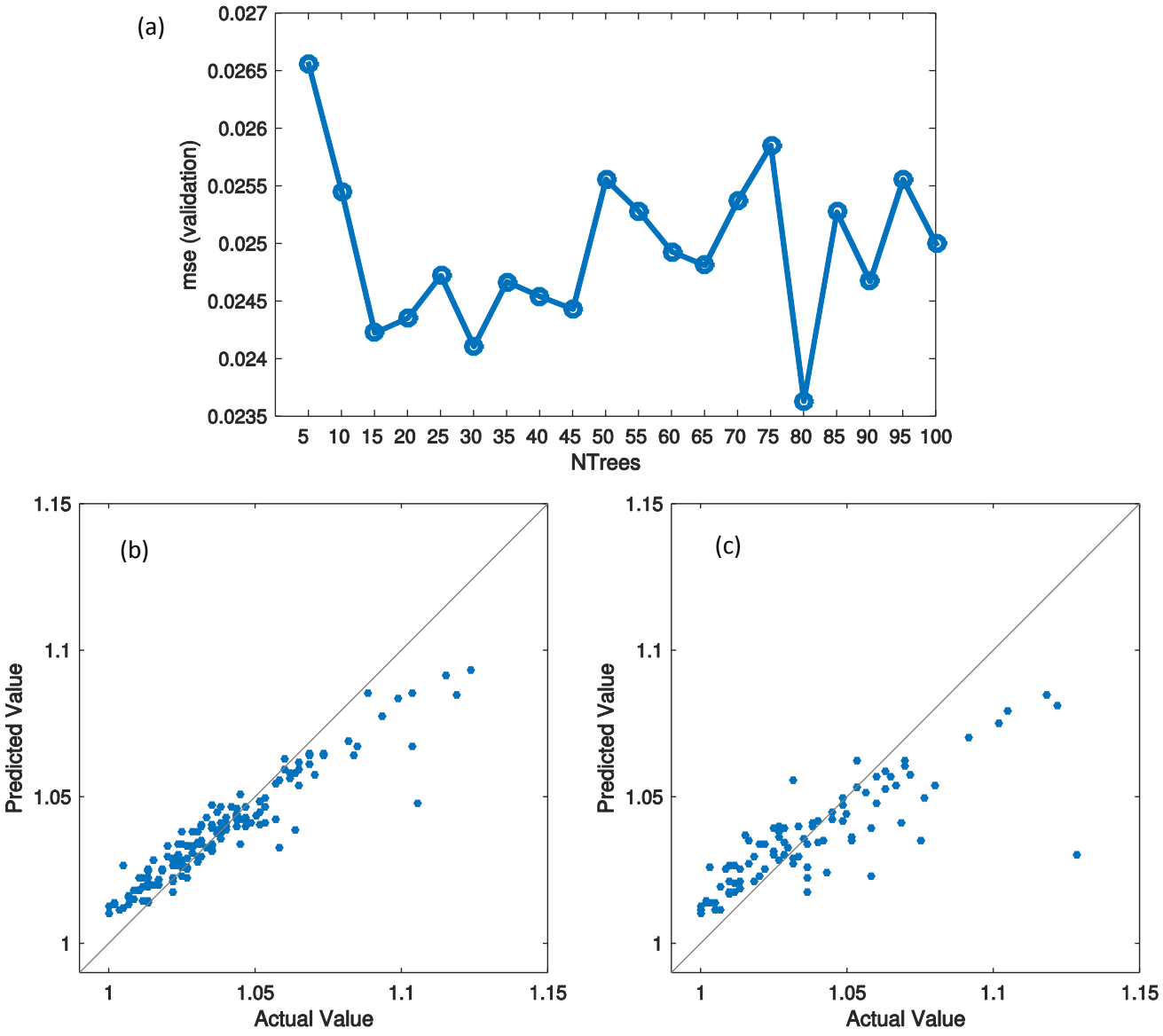


Figure 22: RF model assessment; (a) number of trees selection – validation set (b) best model-80 trees- training set performance (c) best model-80 trees- test set performance

	Parameter	Value
1	Number of trees	5:5:100
2	Number of variables to sample	1
3	Bootstrap sample size ratio	1
4	Sample with replacement (true/false)	True
5	Minimum leaf observations	5

Table 14: RF parameters used in example

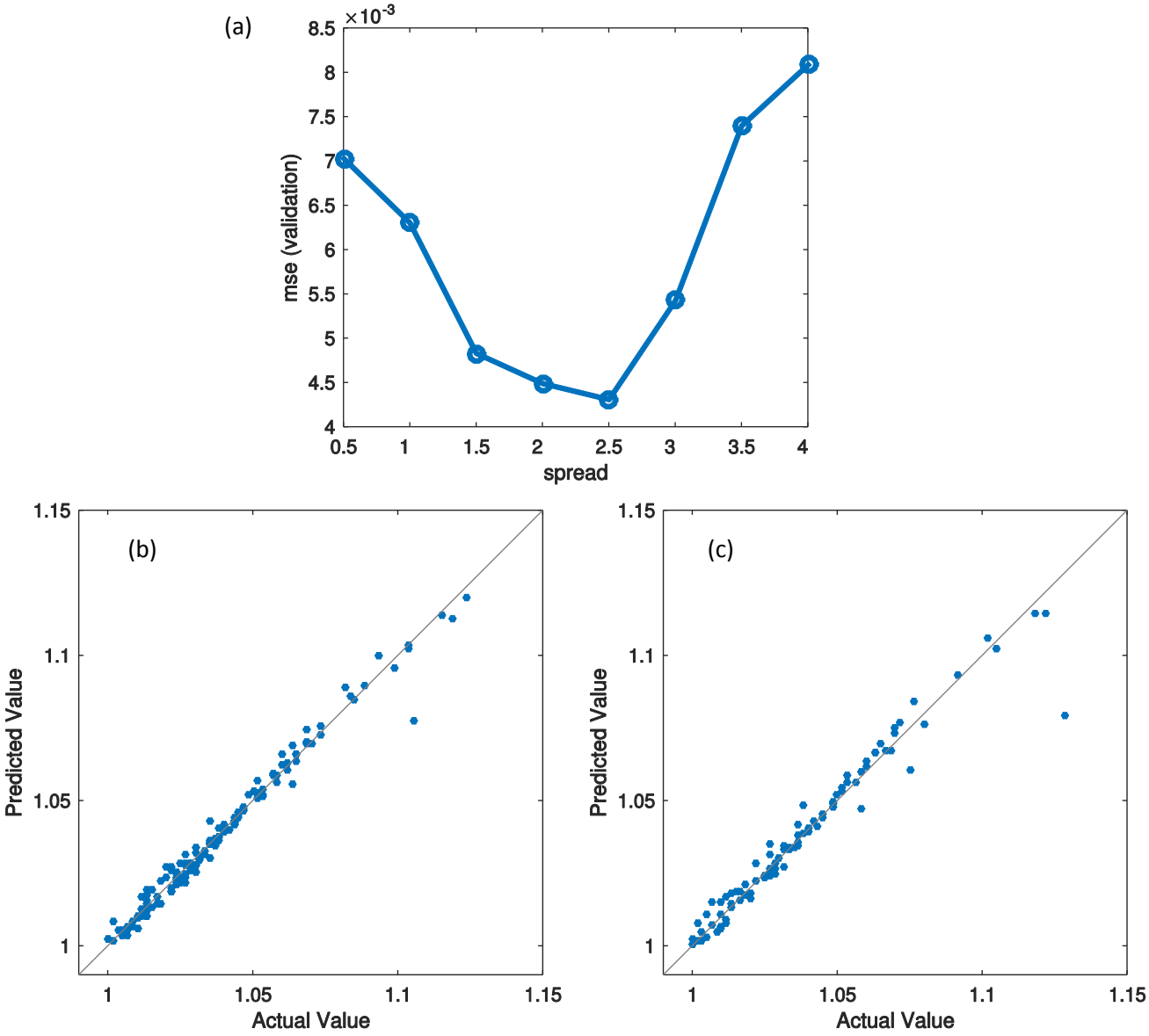


Figure 23: RBFN model assessment; (a) spread selection – validation set (b) best model-2.5 spread- training set performance (c) best model-2.5 spread- test set performance

	Parameter	Value
1	Mean squared error goal	1e-3
2	Spread of radial basis functions	0.5:0.5:4

Table 15: RBFN parameters used in example

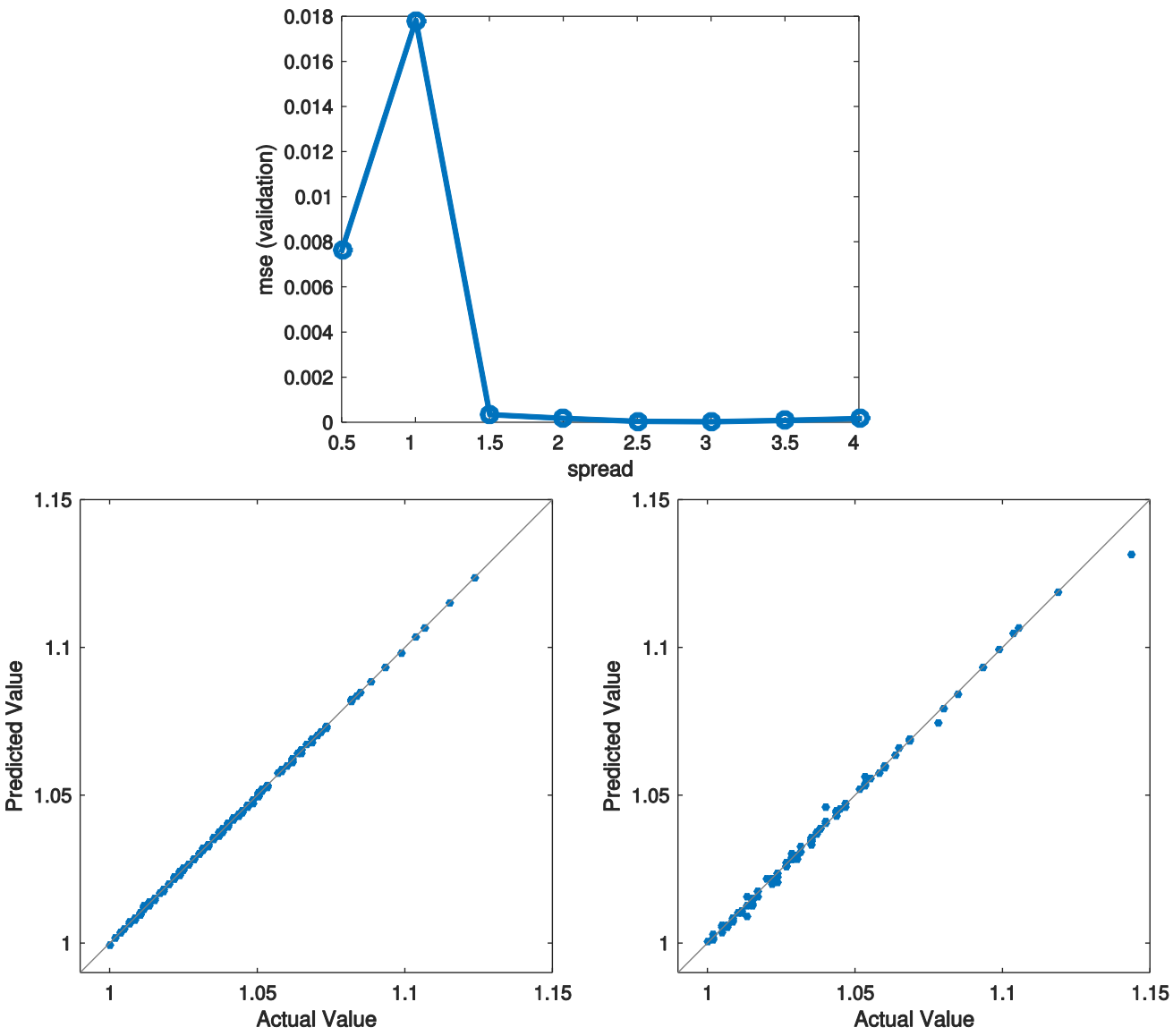


Figure 24: RBFNE model assessment; (a) spread selection – validation set (b) best model-3 spread- training set performance (c) best model-3 spread- test set performance

	Parameter	Value
1	Mean squared error goal	0
2	Spread of radial basis functions	0.5:0.5:4

Table 16: RBFNE parameters used in example

From the above figures, it is evident that the RBFNE model performed better than the RBFN and RF for this example, meaning that the test set scatter points for the Actual vs. Predicted values lie closer to the line of slope 1.

A way to quantify the proximity of the points to the slope 1 line, the correlation of those two variables (Actual and Predicted performance) can be computed and the closer this value is to 1, the better the performance. This correlation metric is often referred to as the R value in statistics.

As an extension to the scatter plot described previously, it is proposed to use scatter plots of each explanatory variable versus the score for both Actual and Predicted score to assess model performance. This plot for the chosen RF model for the earlier example is presented in Figure 25, while the corresponding one for the RBFN model is in Figure 26.

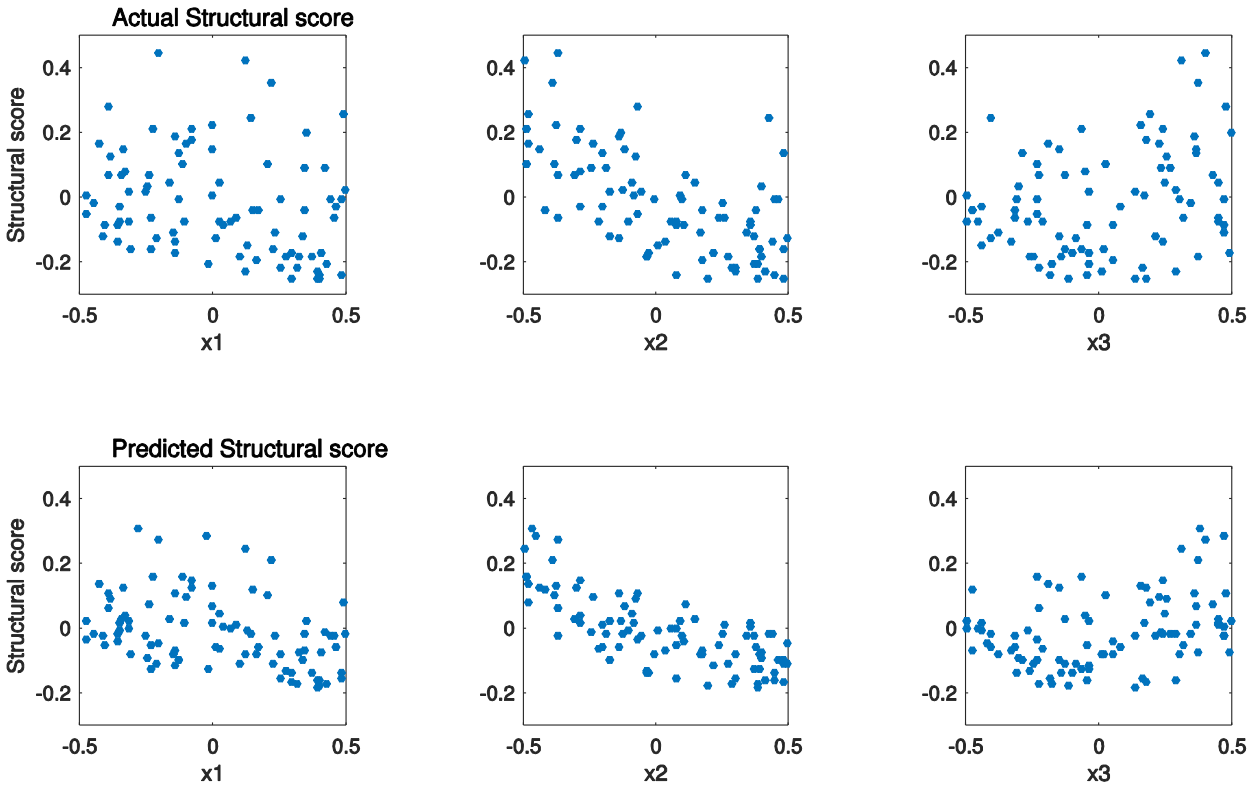


Figure 25: RF variables vs. score scatter plots (Test set)

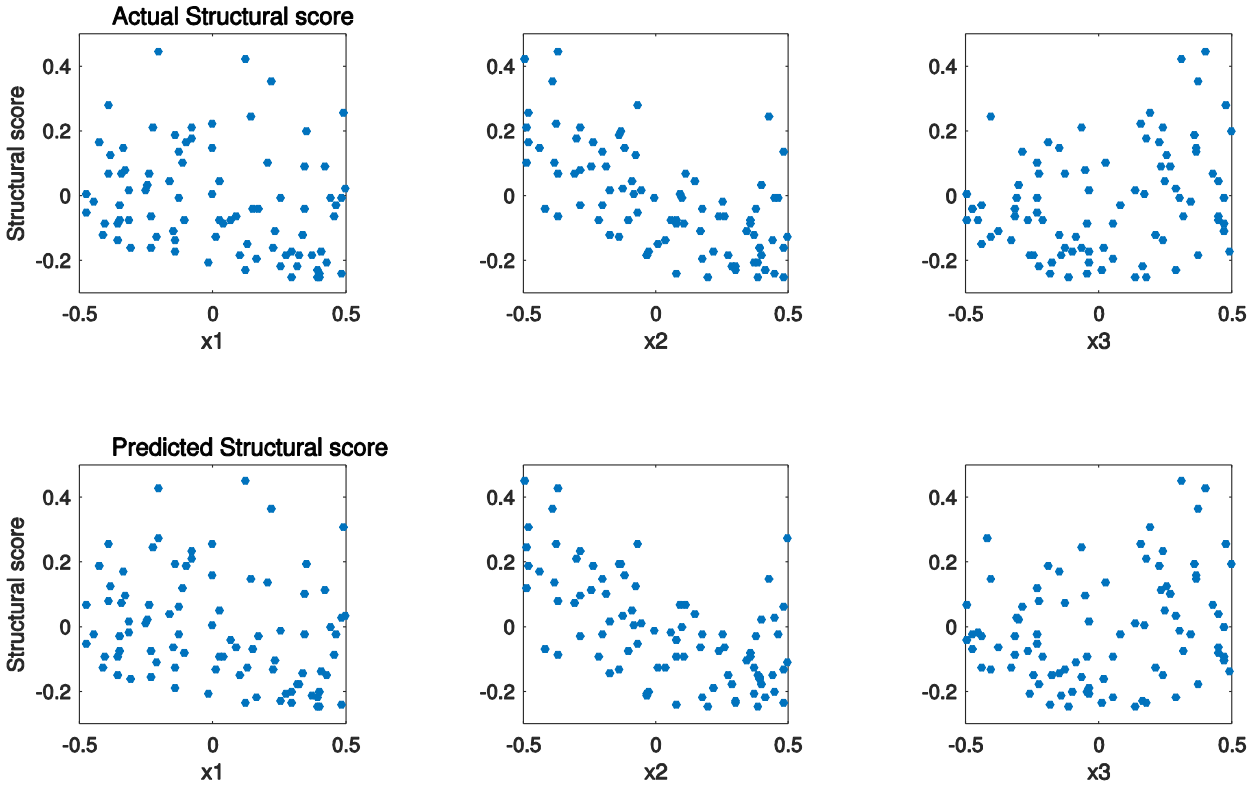


Figure 26: RBFN variables vs. score scatter plots (Test set)

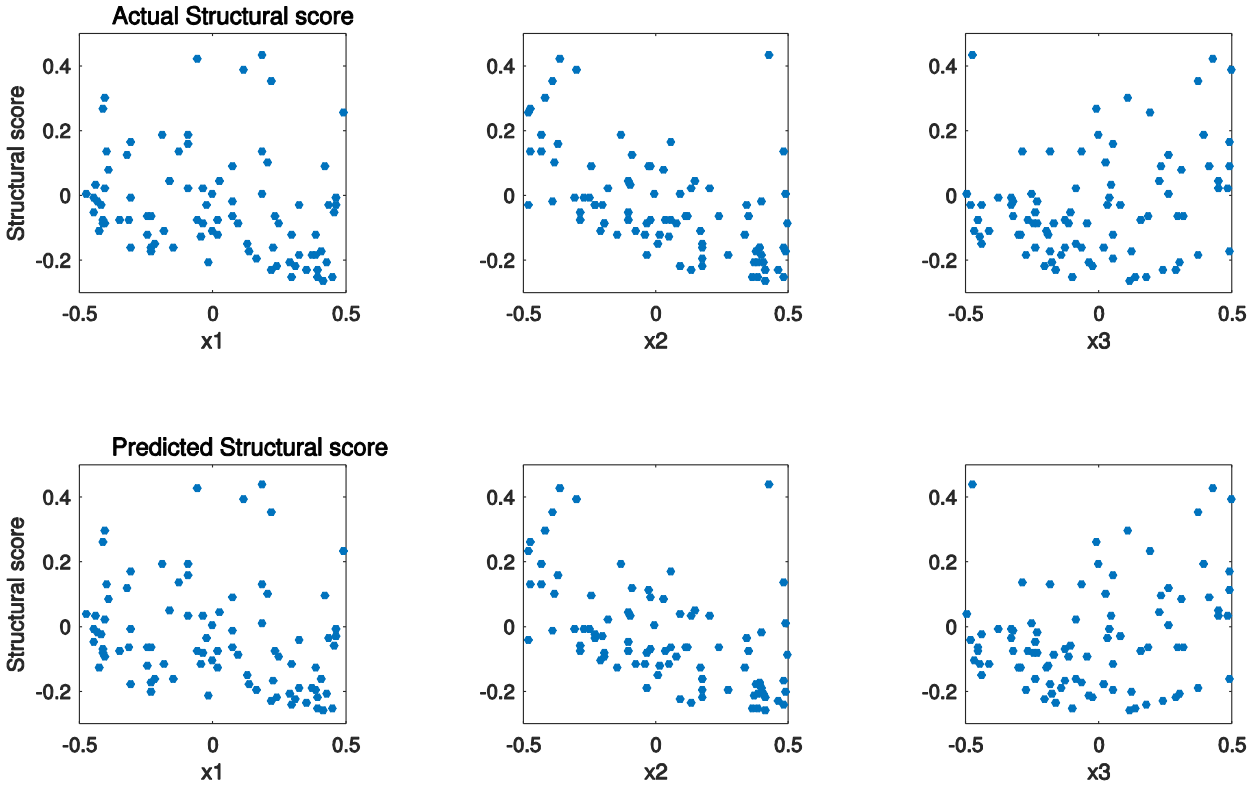


Figure 27: RBFNE variables vs. score scatter plots (Test set)

For this type of variables vs. score scatter plots, the closest the match of the Predicted with the Actual plot, then the best the model performance. When the number of variables becomes large, it becomes difficult to fully visually assess this type of scatter plot. However, those scatter plots could reveal insights about the variable importance and the way the model acts on each one separately.

This type of plot is an attempt to visualize the design space and assess a model performance at the same time in multiple dimension (input variable) problems. When there are only two input (explanatory) variables to consider (always assuming there is a single output score) then comparing 3D scatter plots of the two variables on the X and Y axis and the score on the Z axis for the Actual and Predicted scores could achieve this purpose. For more than two input variables, as is common in practice, the proposed variable vs. score scatter plots are a solution.

4.2 Histograms

Another useful graph to visualize input and output data for surrogate modelling applications is the histogram. It can be used for both the input and the output variables, for the Actual and/or the Predicted values.

For the input variables, a histogram is important to quickly assess the quality of the sampling. For the example of the seven bar truss examined in this chapter, the histograms for each input variable are shown in Figure 28. From this Figure, we can see that the distribution is quite even among all values. This is the desired result for a successful sampling. The slight deviations from the uniform distribution are attributed to the fact that the random sampling scheme was used. The variable ranges used for the sampling are those of Table 12.

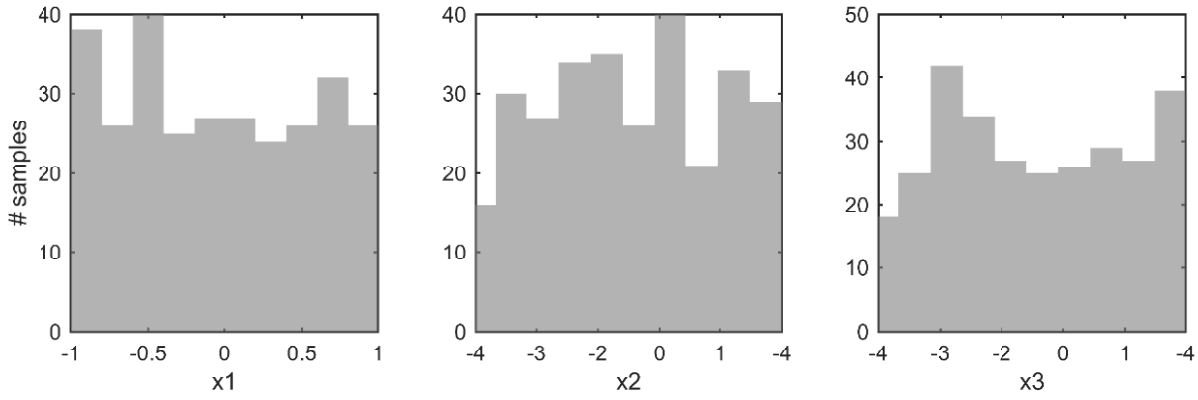


Figure 28: Input variable histogram

As already illustrated in the example with the removal of outliers from the dataset in a previous section, the histogram is also useful for the output variable (score) shown in Figure 29.

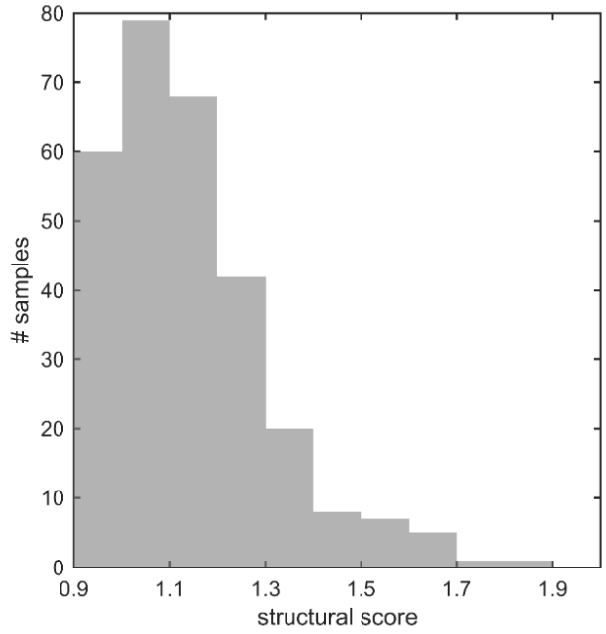


Figure 29: Output score histogram

The histogram of the output values (score) is not necessarily uniform. Its form can be anything and it depends on the interactions between the variables and the sensitivity of the output to each one.

It is useful for identifying if outliers exist in the dataset to take measures against that. Furthermore, it is important to gain an insight on the score ranges and distribution. For example, in the histogram of Figure 29, the majority of the designs have scores up to 20% larger than the "initial" geometry design (base design), which has a score of 1. This normalization scheme, making the base design have a score of 1, was used here to demonstrate that most sampled design perform worse than that and those which perform better only do so at the 10% range. This picture of the "scarcity" of high-performing designs is very important especially in applications where surrogate modelling is used in conjunction with an optimization algorithm. Such an example, where it is very difficult to identify high-performing designs in the design space is presented through a case study in the current thesis.

4.3 "Flat" model

Often it is difficult to comprehend a model's performance in terms of MSE. This is because MSE is a single and somewhat abstract numerical value (possibly with a range). We can compare this value for different model types and across different nuisance parameters for the same model type, but there is lack of a physical intuition on what that number stands for on its own. Specifically, how can it be interpreted qualitatively in terms of the model making good predictions. One wonders how big or small this number

is compared to the values of the scores, how does the normalization scheme affects it, if at all and other similar issues.

To address this, another scalar value for the error is needed to compare to, which is somehow internal to the data themselves without any model effect. One approach to obtain such a number would be to make a random model and calculate its error on the validation set. Specifically, make random predictions and calculate the error. It would be necessary to repeat this process a number of times to obtain an interval to which a t-test could be performed to determine whether a trained model has an effect on the predictions or it can be considered random.

A more simple approach that is proposed here and assists in a more rapid and qualitative evaluation of a model's error is the "flat" model approach. This approach, instead of random predictions, considers a model that predicts a single value for each data point in a set. The proposed value is the arithmetic mean of the dataset. So, the model used to compute the error is a "flat" model that always outputs the same value as shown in Figure 30. It serves as a good benchmark (like the random prediction) since it does not reflect in any way the structure of the data and a surrogate model must surely have a considerably lower error than the "flat" model's in order to be considered adequate.

For the seven bar truss example studied in the current chapter, the model comparison on the test set is shown in Figure 31, where the "flat" model error is the black dashed line at the top of the plot. The y-axis scale is logarithmic and we can observe that some models performed substantially better than the "flat" model, which is the desired effect. To further explain the "flat" model, the scatter plot produced for the test set is included in Figure 30. The predicted value is always the same (mean of Actual values); the "flat" model error is the MSE of the model represented in this scatter plot.

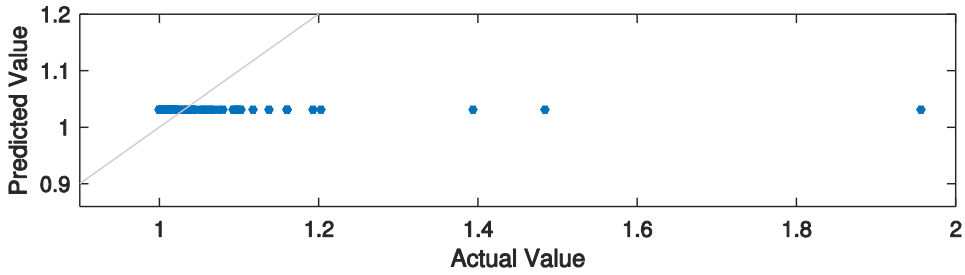


Figure 30: Seven bar truss "flat" model scatter plot – Test set

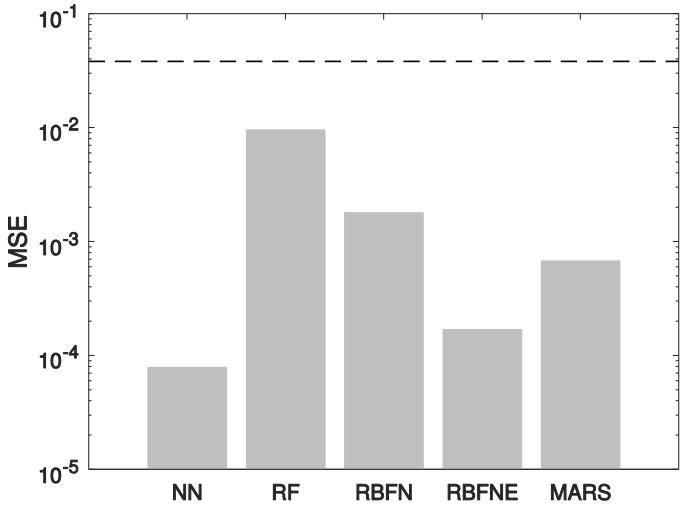


Figure 31: Seven bar truss model comparison with "flat" model error mark – Test set

4.4 Rank error measures

In the context of design and comparing alternatives, the interest in the precision of the performance score value is not great. Instead, the ranking of the different designs with respect to their respective performance is more important. With the goal of integrating this approach in surrogate model assessment, error measures based on rank are examined.

The new Rank-based error metrics described in this section and used to assess the trained models for the examined case study datasets were first introduced [13]. They are briefly described below:

1. Mean Rank Error (MRE)

$$mre = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{r}_i - r_i|}{N/2}$$

$N: total\ number\ of\ samples$

$r_i: actual\ rank$

$\hat{r}_i: predicted\ rank$

2. Top Mean Rank Error (TMRE)

$$tmre = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{r}_i - r_i|}{n/2}$$

$n: number\ of\ top\ performing\ samples\ (user\ specified)$

3. Top Ratio Error (TRE)

$$tre = \frac{n - x}{N}$$

$$x: number\ of\ correctly\ identified\ top\ performing\ samples$$

$$(\#\ predicted\ samples\ in\ actual\ top\ n)$$

The TRE metric "computes how many of the top n samples have been correctly ranked as within the top n designs by the predictive model" [1].

4. Top Factor Error (TFE)

$$tfe = \frac{\hat{r}_{i,max} - n}{N}$$

$$\hat{r}_{i,max}: maximum\ predicted\ rank\ among\ the\ actual\ top\ n\ samples$$

The TFE metric "computes how far out of the top n samples the predictive model has placed its top performers" [1].

For the analysis in the current thesis, the metrics for Mean Rank Error (MRE) and Top Mean Rank Error (TMRE) have been slightly modified to not include the N/2 and n/2 normalization respectively. Therefore, the used metrics were:

$$mre = \frac{1}{N}\sum_{i=1}^{N} |\hat{r}_i - r_i|$$

$$tmre = \frac{1}{n}\sum_{i=1}^{n} |\hat{r}_i - r_i|$$

With this modification, the MRE and TMRE directly reflect by how many rank positions a sample has been "mis-ranked" on average. This is more comprehensible and easily evaluated. The further normalization of N/2 could be useful to compare such error metrics for models with different validation (or test) set sizes. Since for the case studies a fixed number of data points was concerned, this further normalization was decided to not be included in the metrics.

4.4.1 Permutation tests for rank error measures

For the Rank Error Measures, the "flat" model concept introduced in a previous section does not provide a comprehensible way to understand the effect of the model on rank prediction. This is because, it is less comprehensible to assign the same rank to each sample. For this reason, the random prediction paradigm

introduced as well in the previous "flat" model section is used here to determine whether a model has a predictive ability on ranks.

A way to perform permutation testing on the Rank Error Measures is introduced here. The procedure is as follows:

1. Compute the Rank Error Measures (REM) for the test set.

2. Create a random rank permutation, and assume this was the result of the prediction by computing the REM of this result vs the Actual test set rank.

3. Repeat the step two (2) many times. The number of runs for the results included in this section was chosen to be 10,000 random permutation runs.

4. Create the histogram of each REM resulted from the permutations and establish a p-value for the actual model predicted REM.

The null hypothesis for the permutation test is:

$$H_0 = \{the\ predicted\ ranks\ are\ random\}$$

And the H_1 hypothesis is:

$$H_1 = \{the\ predicted\ ranks\ are\ NOT\ random\}$$

The resulting histograms of this process for the example seven bar truss problem for each REM are shown in Figure 32a-d. It should be noted that for the computation of TMRE, TRE and TFE the top 10 designs were accounted for.

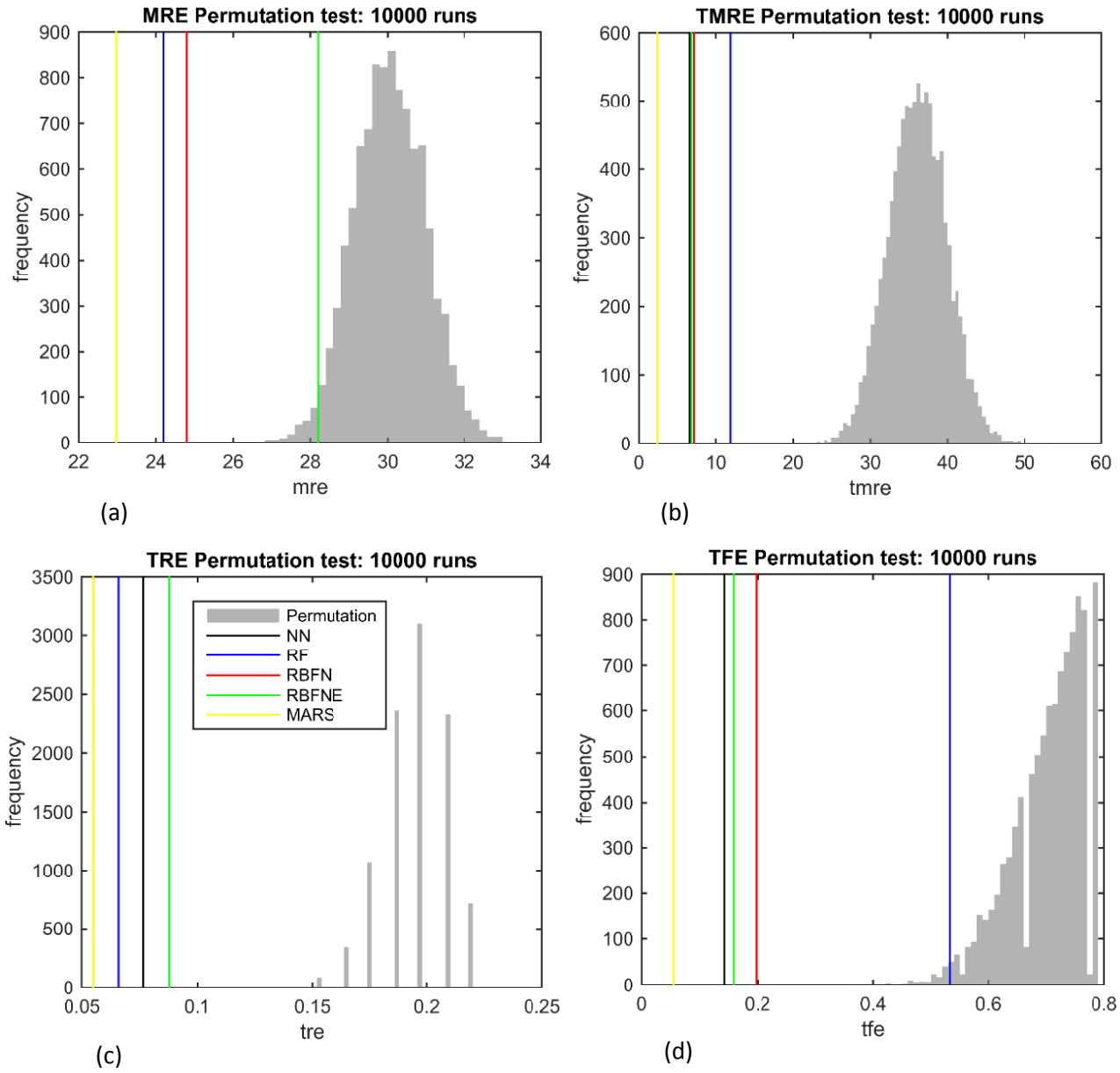


Figure 32: Seven bar truss REM permutation test histograms – (a) MRE (b) TMRE (c) TRE (d) TFE

One can see from the permutation test results on Figure 32a that there is no overlap between the histogram and the test set values for each model, except RBFNE and thus the p-value is very close to zero. This shows that the H0 hypothesis that the trained models have no effect on the MRE can be rejected. Therefore, the models really have an effect on MRE which means they have a rank predictive ability. With the same logic, from Figure 32b we can see that the p-value for the Top Mean Rank Error (TMRE) is zero

for all models, which verifies that the models have a predictive ability on TMRE (top 10 designs). TRE and TFE show similar patterns.

It is interesting to note some possible insights that the permutation test results for Top Ratio Error (TRE) and Top Factor Error (TFE) could reveal. If the p-value for TRE for some model was not zero, while MRE and TMRE approached zero, then this suggests that for that model the predicted results are not accurate in terms of rank. In addition, a non-zero p-value only at the TFE permutation test of one model reveals that for this model there are probably some samples whose predictions are disproportionately larger compared to the other models (because the MRE, TMRE are good). This fact may suggest overfitting. It is interesting and important that the REM, along with the permutation tests, can provide many insights into the behavior of the various models, their individual assessment and their comparison.

4.5 Summary

Techniques for visualizing and measuring error were discussed in this chapter. The importance of reviewing scatter plots of actual vs. predicted values was given special attention. The major contribution was the introduction of a new concept of interpreting error; the “flat” model error. This type of error was extended to also the quantification of a model’s performance on rank error measures by the use of statistical permutation tests.

5. Robust model comparison

A robust model comparison methodology has been developed and is described in detail in this chapter. It was applied in several case study problems, with the results showcased in the next chapters. The motivation for this methodology is to have a way to quantitatively compare the performance of different families of models in approximating the same dataset. An important goal of the methodology is the extraction of an interval of a model's error in addition to the average error value. The way to obtain this interval is not to make a single run of training models on a given train/validation set configuration, but use many configurations and make several runs. At first the structure of a single run is described. Then the process of generating more runs and aggregating them to compare the models is explained. This chapter introduces the main framework that has generated the results of the case studies.

5.1 Single run

In the first place, we assume that there is a single training set and a separate single validation set. A framework was developed in MATLAB to train all the six different types of models discussed here for different nuisance parameters for within each model type as well. The nuisance parameters considered for each model have been extensively described in Chapter 3. The training set (the same for all the models) is firstly used to train the models. Then the mean squared error of those models is computed on the validation set (again the same for all models) and the nuisance parameters with the lowest error is chosen for each model type. This procedure is shown schematically on Figure 33, with the "Best" nuisance parameter model chosen for each model type. One could then use the test set data, apply them to the "Best" models and assess their performance through the visualization techniques described in the previous chapter.

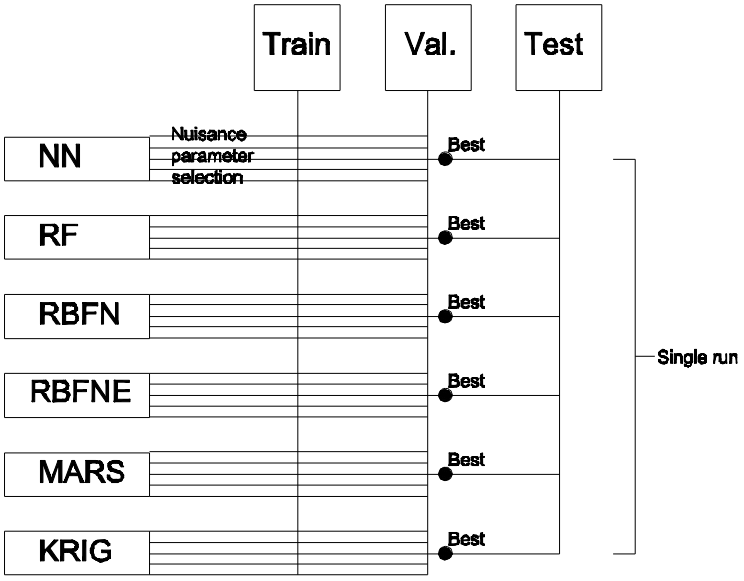


Figure 33: Single run workflow

5.2 Multiple runs

When a single run is carried out, a scalar is produced as the error of each model. However, the error resulting from the test depends in part on the specific data sets used for training and validation. The method proposed here seeks to characterize the effects of variability due to the data set so that the model's general robustness can be understood. One method to obtain a variability measure for the error is cross-validation, which was described briefly in Chapter 3. In that chapter, it was also argued that when an abundance of data is available, it is preferred to choose a completely separate validation set and avoid cross validation. With those two facts considered; 1) abundance of data 2) a need for an error variability measure, a methodology of training the models for several different separated training and validation sets is proposed.

In detail, the aforementioned procedure for the single run is repeated for many different random partitions of the training and validation data pool. This means that the training and validation data are pooled and a certain number of random partitions of these data in training and validation sets with a constant number of samples in each set are generated. The validation set error (MSE) is stored for each different nuisance parameter and every model type. Afterwards, the mean of the MSE for each parameter across all partitions is calculated. Then the nuisance parameter with the minimum arithmetic mean MSE is chosen and the errors for this specific parameter for each partition are studied as the desired measure of the variability. The standard deviation is a metric that can be extracted from this information or just the range and the scatter can be examined. This process is applied for each model and the output is an error measure with the accompanying variability for each of the six models. For one model, the process is schematically shown in Figure 34. In this figure, there were five nuisance parameters considered and five different partition runs were carried out for each parameter. The errors are accumulated, with the thick dashed line showing the arithmetic mean of the runs for each parameter. The lowest-mean-error parameter was chosen; thick black box in the figure.

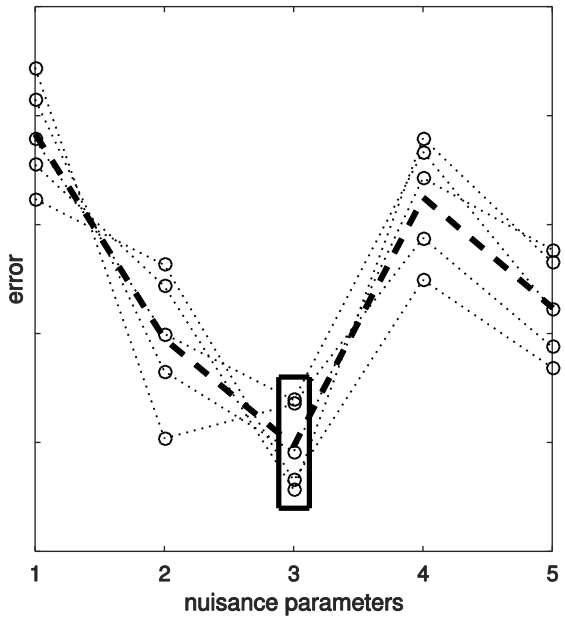


Figure 34: Robust model comparison framework sketch for a single model

The results obtained after this process are summarized in a bar chart to compare the models, which was the original motivation. A bar chart for the seven-bar truss is included in Figure 35. The same chart is included in Chapter 7 where it is put in context. Each bar represents the mean of the error for each model type. The error of each run is also plotted in small black scatter dots to show the variability of the error. The number of the mean is also written inside each bar for easier reading of the whole graph.

A "Flat" model error measure was taken for the validation set for each partition and the mean is also plotted as a dashed straight line in the comparison bar chart, with the actual number also printed at the far right hand side of the line. The y-axis scale is in logarithmic scale to capture large differences in the results.

		NN	RF	RBFN	RBFNE	MARS	KRIG
Mean	[%]	0.2	16.7	1.2	17.9	0.5	0.3
Max	[%]	0.4	25.7	3.4	36.6	0.9	0.8
Min	[%]	0.1	10.4	0.5	6.2	0.3	0.2

Table 17: Seven-bar truss model performance as per-cent of "flat" model error

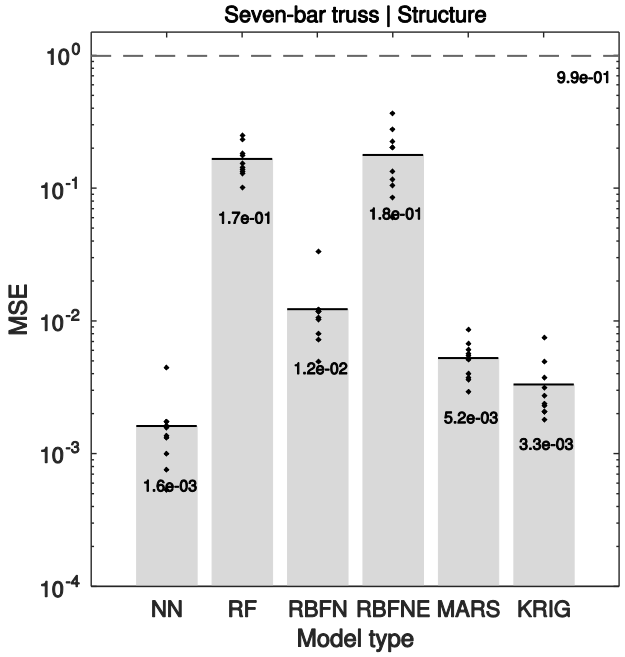


Figure 35: Seven-bar truss comparison bar chart

From Figure 35, it can be observed that some models (NN, RBFN, MARS, and KRIG) performed very well, with RF and RBFNE performing the worst. The performance becomes more evident when someone looks at Table 17 which lists each models mean, maximum and minimum error as a percentage of the "flat" model error. The variability of each model's performance can be visually inspected from the figure by looking at the black scattered dots representing the respective model's error for each run. As another way to quantify the variability one can look at Table 18 which includes the percentage values of the maximum

and minimum error with the respect to the mean one for each model, as well as the standard deviation of the runs' errors.

		NN	RF	RBFN	RBFNE	MARS	KRIG
Max	[% of mean]	276	153	271	204	166	231
Min	[% of mean]	33	62	41	34	56	55
St. Dev	-	0.001	0.048	0.008	0.095	0.002	0.002

Table 18: Seven-bar truss model error variability metrics

It is very interesting to observe that NN has the lowest standard deviation, but has the largest Max value. This is observed from Figure 35, where there seems to be an outlier in the errors. This perfectly illustrates the reason that more partition runs are needed to increase the robustness of an error measure and make a better decision when choosing a model. Particularly, if only a single run was performed and it happened to be on this outlier set, then the eventual model selection could be different. These overlaps of the error metrics for different runs indicate that the model comparison picture could be different if the multiple runs and aggregation was not performed. This is very important, since it has a direct impact to the final model selection for the approximation in the project at hand.

In fact, this can be even more convincingly illustrated by the fact that in Chapter 4, the RBFNE model showed the best performance among the models for an approximation of the same dataset. For the figures in Chapter 4, the training set was set to 150 samples, validation at 50 and test set at 91. For the figures in the current section, the runs were performed with a training set of 180, validation 50 and test 51. The sensitivity of the models on their parameters in the overall performance and the final selection is extremely high. This is why it is of utmost importance that a procedure such the one introduced in this chapter is performed to increase the robustness of the model selection.

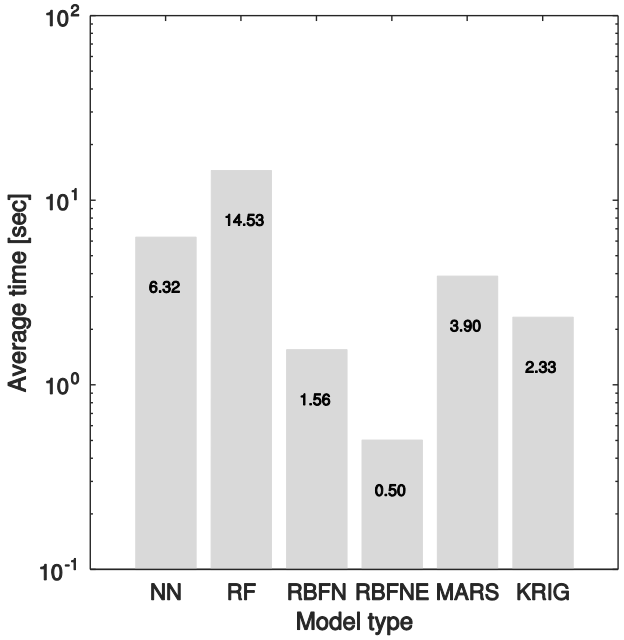


Figure 36: Seven-bar truss model build-time comparison

Figure 36 shows a model construction time comparison bar chart. More specifically, it shows the average time needed to construct a model of each model type for a single run. It should be pointed out that this time includes the validation over several parameters and the eventual selection of the best parameters within each run; the procedure described in the first section of the current chapter for the “Single run”. It is notable that the best performing model in terms of error (NN) required the second longest time to build, while the worst performing (RBFNE) required the least time. And as mentioned before, this is mainly due to the fact that the NN iterates over more parameters and the best-performing network on the validation set is picked, while the RBFNE model only includes a single parameter (spread of the radial basis functions). The fact that the user can choose the number of parameter a model iterates over makes the picture even more complex. Overall, the trade-off between accuracy and time is evident from the above figures.

5.3 Summary

A methodology for a robust model comparison and selection was presented in this chapter. It has been applied to relevant case studies which follow in the next chapters. The intention of this methodology and subsequent framework is not to draw general conclusions regarding model types and their behavior, but instead to implement the concept of trying many different model types and parameters within each model to pick the best performing each time for the specific data available.

6. Case studies

A number of case studies of using approximation algorithms in structural engineering problems have been examined. In this chapter, the context of each case study is presented. The problem description, the parameters considered, the sampling ranges and histograms. The approximation fit and results then follow in the next chapter.

6.1 Case study 1 – Grid truss

6.1.1 Building

For the first case study, a grid structural system was examined. The initial topology/geometry is shown in Figure 37a. The objective was to explore the design space with each node's coordinates being the variable parameters in the exploration. The objective score for each design was the structural weight ΣFiLi, where Fi is the resulting inner force and Li the length of member i. The total score is the sum of the product of Fi and Li for all the members.

The assumptions of the analysis were: (1) hinge supports at all the bottom cord nodes (2) the loads were applied at all the top cord nodes (3) A load was applied in both the horizontal (X) and vertical direction (Y), with the horizontal being twice the vertical (4) the loads and the member lengths were assumed unitless since the structural scores were normalized so that the initial geometry received a score of 1 (5) the displacement method was used to calculate the forces.

A visual of the resulting inner forces of the initial geometry can be seen in Figure 37b, with tension forces indicated in blue and compression in red. It is noted that the members connecting the hinges could have been omitted and serve no purpose. This case study is part of a wider, more general truss generation MATLAB framework and this is why those members were included here.

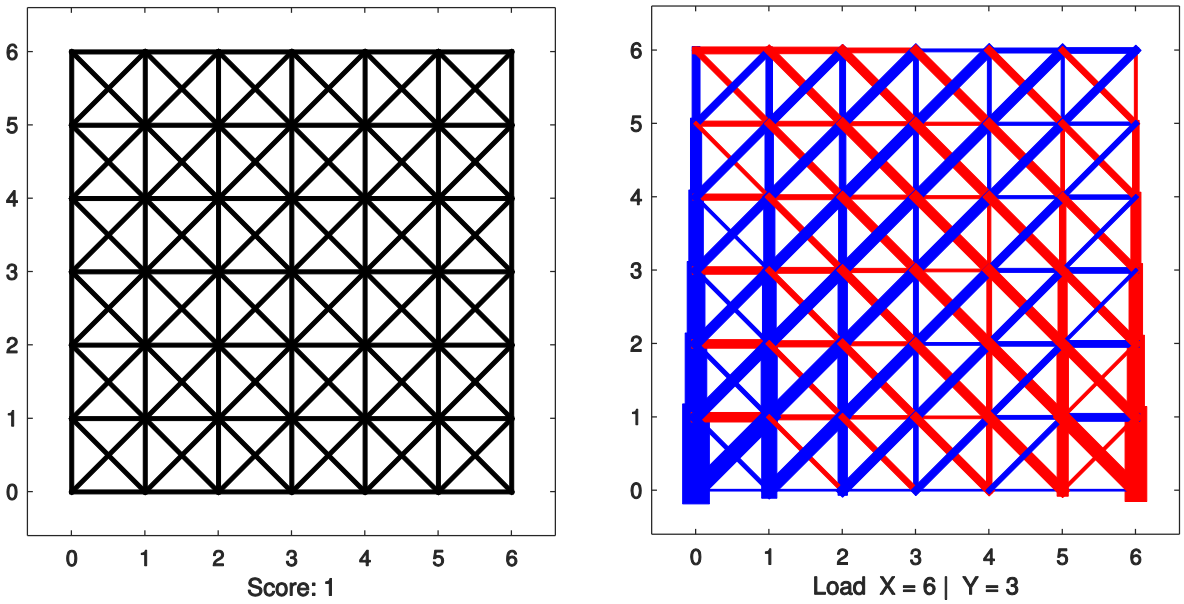


Figure 37: Case study 1: (a) Initial geometry (b) Load paths in initial geometry

The first step to create an approximation model is to sample the design space as has been described in previous chapters. For the sampling of this problem, the variable parameters were the horizontal and vertical deviation of each node from the initial geometry. A constraint was that the hinge bottom cord nodes as well as all the free top cord nodes remained fixed. The sampling range was [-1, 1] with the same dimension used in Figure 37a. The sampling scheme was Latin hypercube. A histogram of the scores obtained from sampling can be seen in Figure 38 for 1,000 and 10,000 samples. One notices that all the sampled scores are above 1, which means that they all perform worse than the original design. This is an indication of how vast the design space is and how scarce high-performing designs are. Indicative sampled designs along with their score can be seen in Figure 39.

This case study showcased the importance of sampling and visualization to better understanding a design space. However, because of the scarcity of high-performing designs, the approximation model development was not pursued.

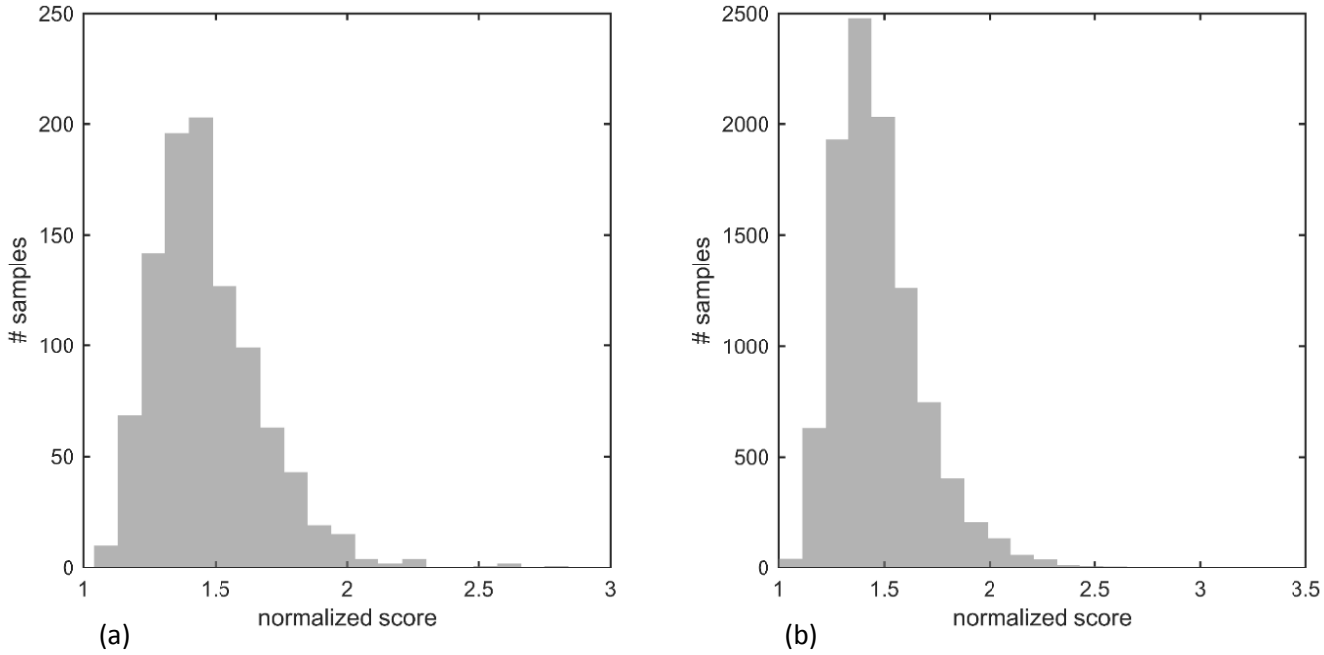


Figure 38: Score histograms for (a) 1,000 samples and (b) 10,000 samples

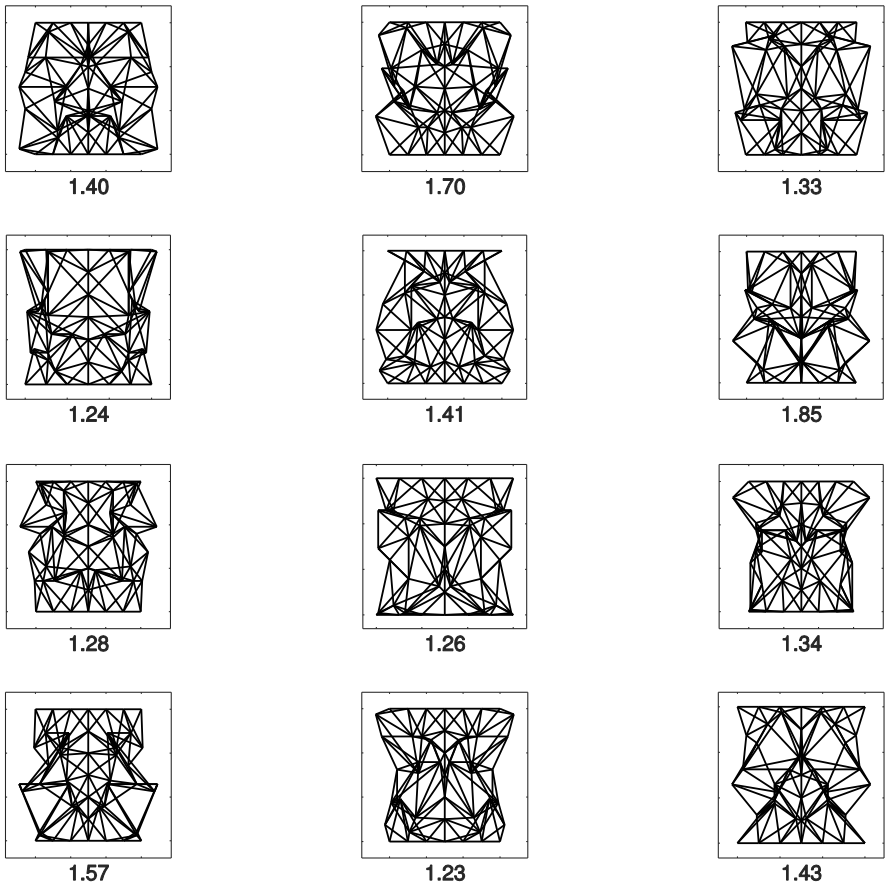


Figure 39: Case study 1-Building sampled designs and their scores

As the grid becomes finer, the structural score evaluation time increases exponentially. This can be seen from the plot of the number of grid modules (assuming square grid) versus single design computation time in Figure 40. Some representative grids with different fidelity levels are shown in Figure 41.

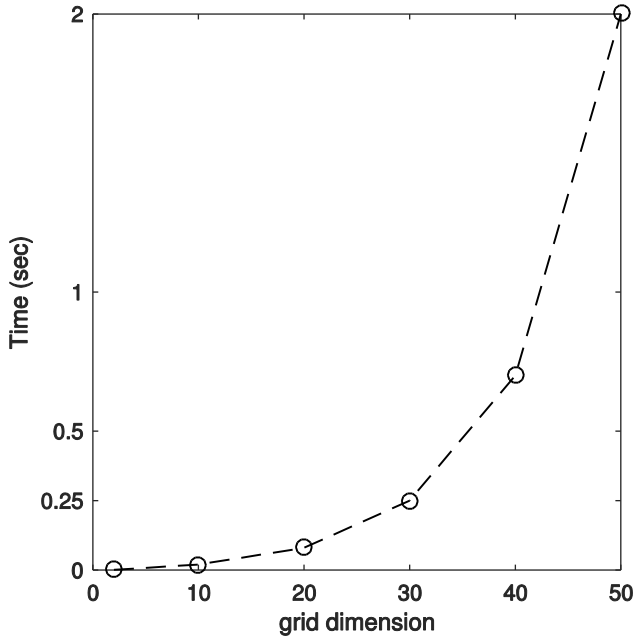


Figure 40: Grid truss fidelity vs. evaluation time

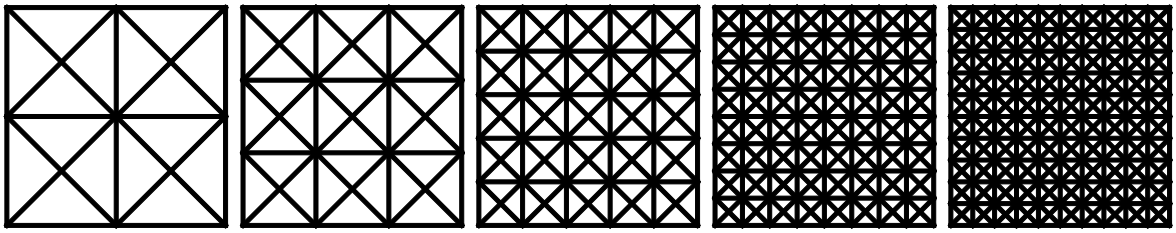


Figure 41: Grids with different fidelity levels

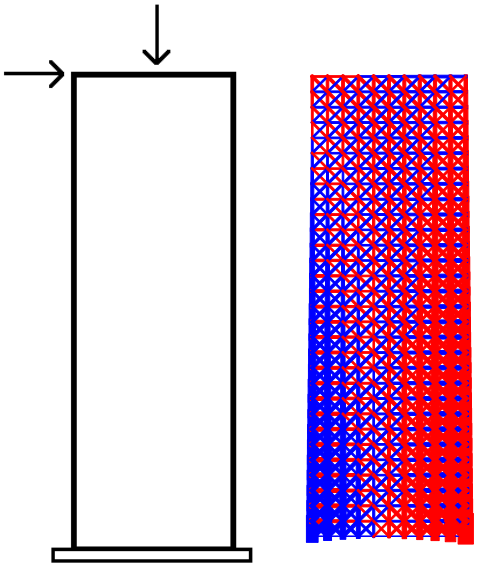


Figure 42: Shear wall simulation

Structural evaluation simulations, become tremendously more expensive computationally when surface finite elements are used to determine the stresses in planar structural elements. The behavior of a planar element such a shear wall could be simulated with the grid framework presented in this section, with a very fine grid to capture the material. For the simulation of the shear wall in Figure 42 (red color represents compression and blue tension), a 10x30 grid was used and the simulation required almost two seconds. So, surrogate modelling could be a choice to potentially explore a shear wall design space more rapidly. The variables in such a case, where the grid is very fine, would not be the coordinates of each individual node, but some more global design variables.

6.1.2 Bridge

The same grid truss framework was applied to a bridge problem, after the building case study of the previous section proved fruitless. The objective and the parameters remained the same as in the previous section, although some of the assumptions have changed and are described below.

The assumptions of the analysis were: (1) hinge supports at the far end nodes of the bottom cord (2) the loads were applied at all the bottom cord nodes (3) A load was applied in only the vertical direction (Y), with no horizontal load (4) the loads and the member lengths were assumed unitless since the structural scores were normalized so that the initial geometry received a score of 1 (5) the displacement method was used to calculate the forces. Each sample evaluation required approximately 2ms.

A visual of the resulting inner forces of the initial geometry can be seen in Figure 43, with tension forces indicated in blue and compression in red.

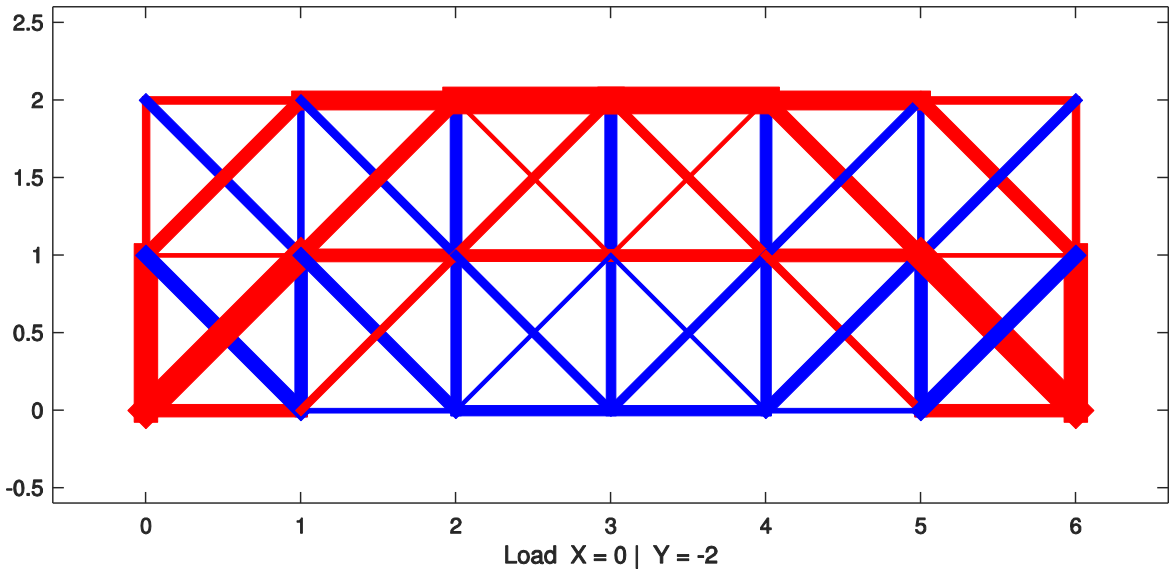


Figure 43: Initial bridge geometry and load paths

For the sampling of this problem, the variable parameters were the horizontal and vertical deviation of each node from the initial geometry. A constraint was that the bottom cord nodes remained fixed, since this cord would potentially serve for the passing of vehicles. This leaves the coordinates of 6 nodes as variables (assuming y-axis symmetry), so in total there are 12 variables. The sampling range was chosen to be [-0.5, 0.5] with the same dimension used in Figure 43. The sampling scheme was Latin hypercube. A histogram of the scores obtained from sampling can be seen in Figure 44 for 10,000 samples.

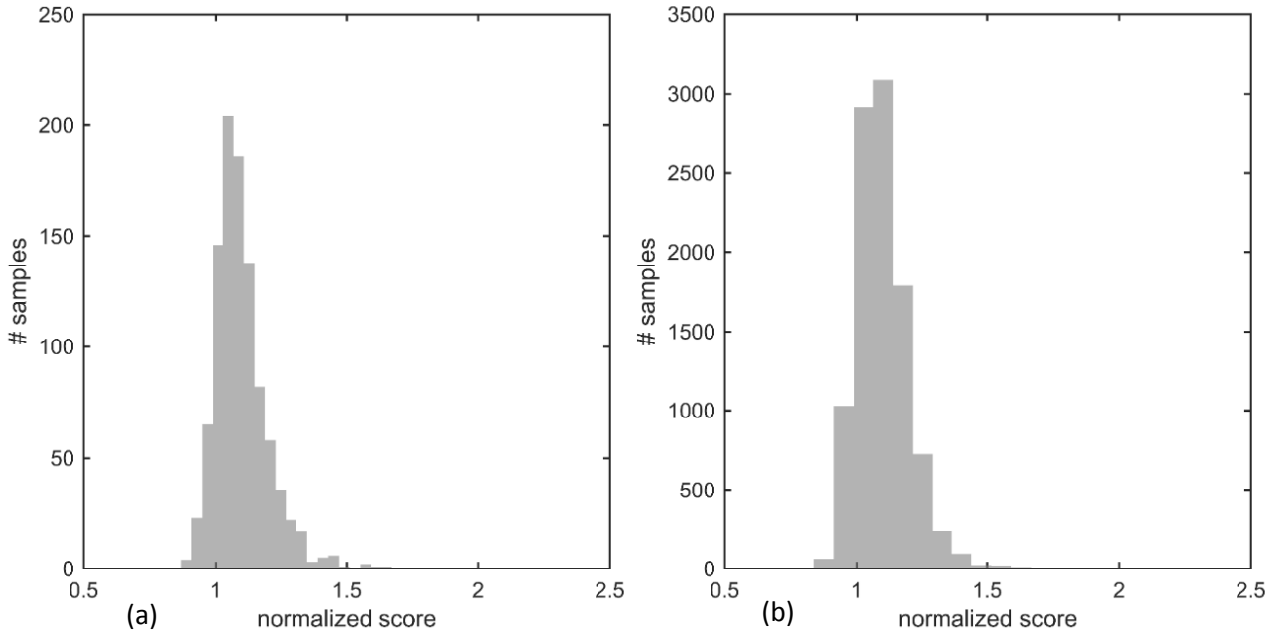


Figure 44: Score histogram for bridge problem for (a) 1,000 (b) 10,000 samples

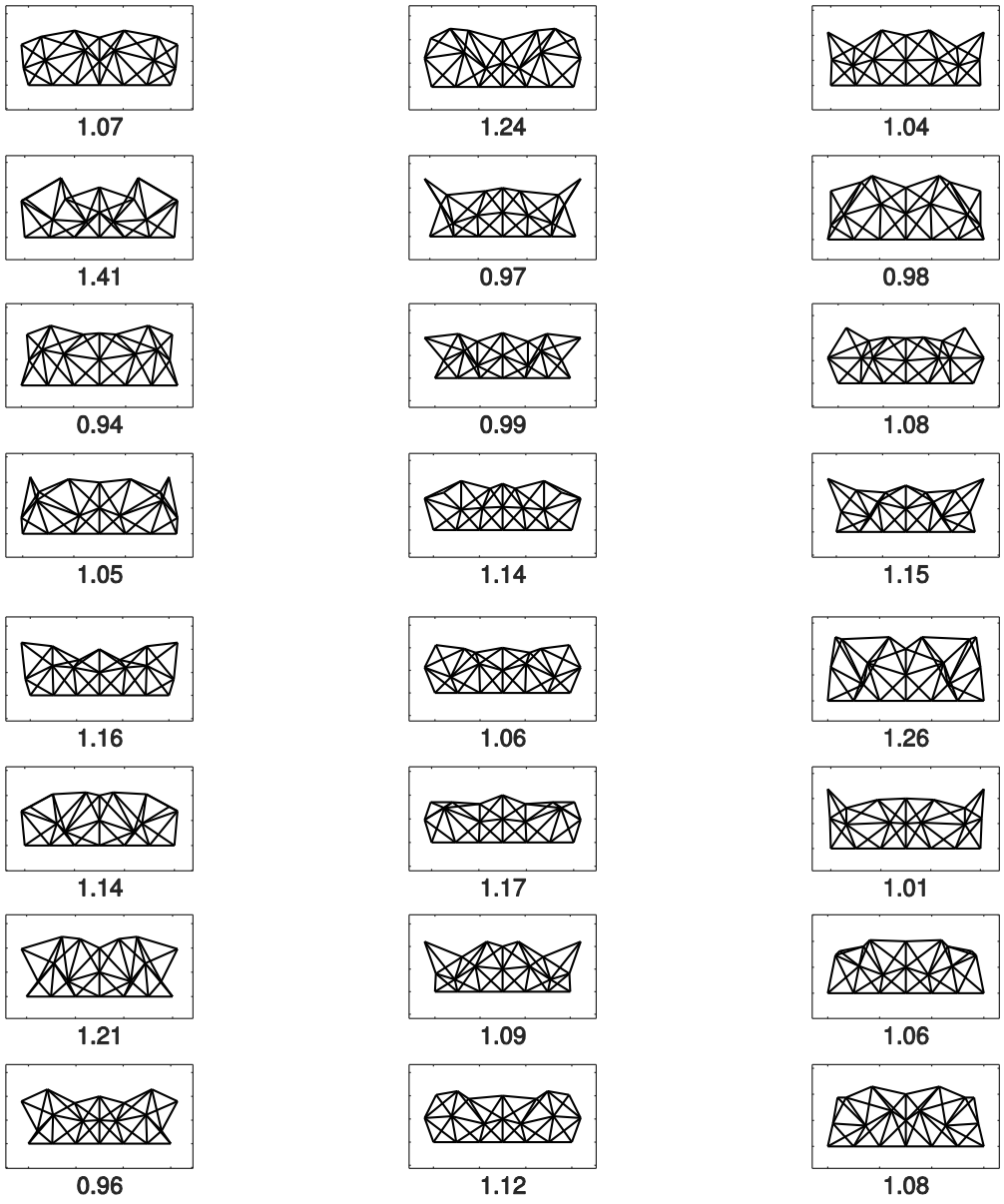


Figure 45: *Case study 1-Bridge sampled designs and their scores*

Figure 45 shows some representative sampled designs and their normalized scores. As is clear Figure 44 and Figure 45, there exist designs that are better-performing compared to the initial base structure (they are the ones with a score lower than 1). Therefore, the approximation framework was applied to this problem and the results are presented in Chapter 7.

Set	# samples
Training	600
Validation	200
Test	200

Table 19: *Case Study 1-Bridge dataset sizes*

6.2 Case study 2 - PI structure

For Case Study 2, the PI shaped structure form shown in Figure 46 was examined.

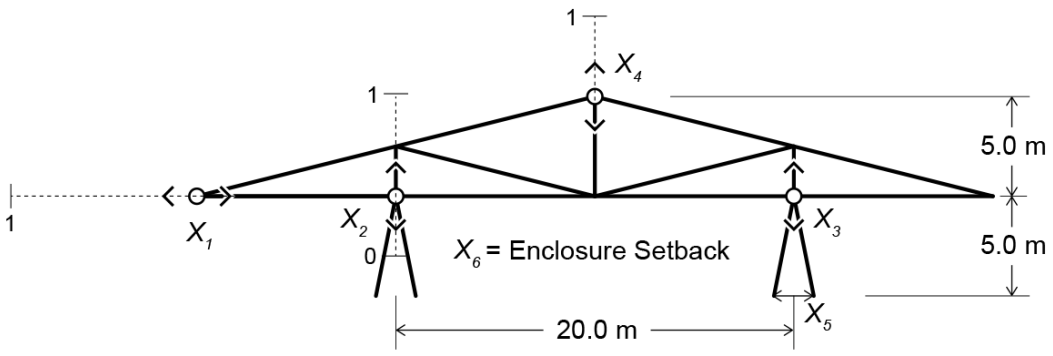


Figure 46: PI structure topology/geometry and variables x1-x6 (Image from Nathan C. Brown)

The variables examined in this case study were (x1) the overhang of the truss outwards, (x4) the position of the top node of the roof truss along the vertical axis. The structure was allowed to be non-symmetrical by considering the positions of the top of the two columns as separate variables (x2) and (x3) as seen in Figure 46. The columns were allowed to be inclined and their inclination was allowed to vary through the varying width of the columns (x5). While the width was set equal for both columns, their final inclination can be different due to the effect of the values of x2 and x3. Finally, the enclosure setback was set as variable (x6). The variables are summarized in Table 20.

Variable	Description
x1	Cantilever length
x2	Vertical position – West column
x3	Vertical position – East column
x4	Vertical position of top node
x5	Column base width
x6	Enclosure setback

Table 20: PI Structure variables

The design space was sampled based on those 6 variables using a Latin hypercube scheme. The variable bounds for the sampling can be seen on Figure 47. The simulation to measure the performance of each design sample was carried out in Rhino/Grasshopper plugins. For the structural performance, the objective was the structural weight and the simulation was performed using the Karamba plugin [44]. For all the energy simulations, the plugin ARCHSIM was used [45]. This plugin connects the parametric model from Rhino, to EnergyPlus [46], an energy analysis program available by the U.S. Department of Energy. This structure was studied in different orientations and climates, which affect the annual energy consumption but not the structural score. More information about this problem, the setup and all the assumptions used in the simulations can be found in [47]. Each energy simulation required approximately 25 sec and each structural 0.2 sec. In this case the evaluation time could be dramatically reduced using surrogate modelling, especially for the energy simulations.

Set	# samples
Training	120
Validation	40
Test	40

Table 21: PI Structure case study dataset sizes

The sampling histograms are shown in Figure 47 and Figure 48 for the Energy Overall and the Structure performance scores respectively. From the input variables' histogram (not shown here) it can be verified that the sampling captured the design space uniformly and from the performance scores histogram we can see that the distribution is similar for all the three sets for both performance scores. This visual verification has been performed for all the objective performance scores but the diagrams are not included here.

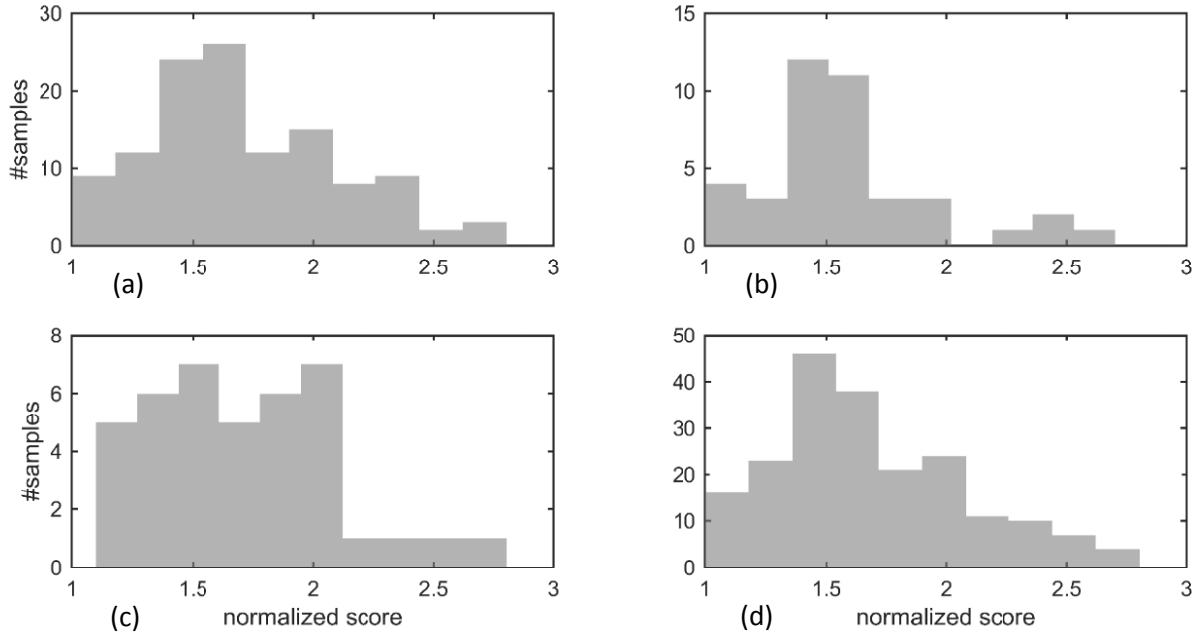


Figure 47: Boston Energy Overall normalized score histograms for (a) Training (b) Validation (c) Test sets (d) All sets

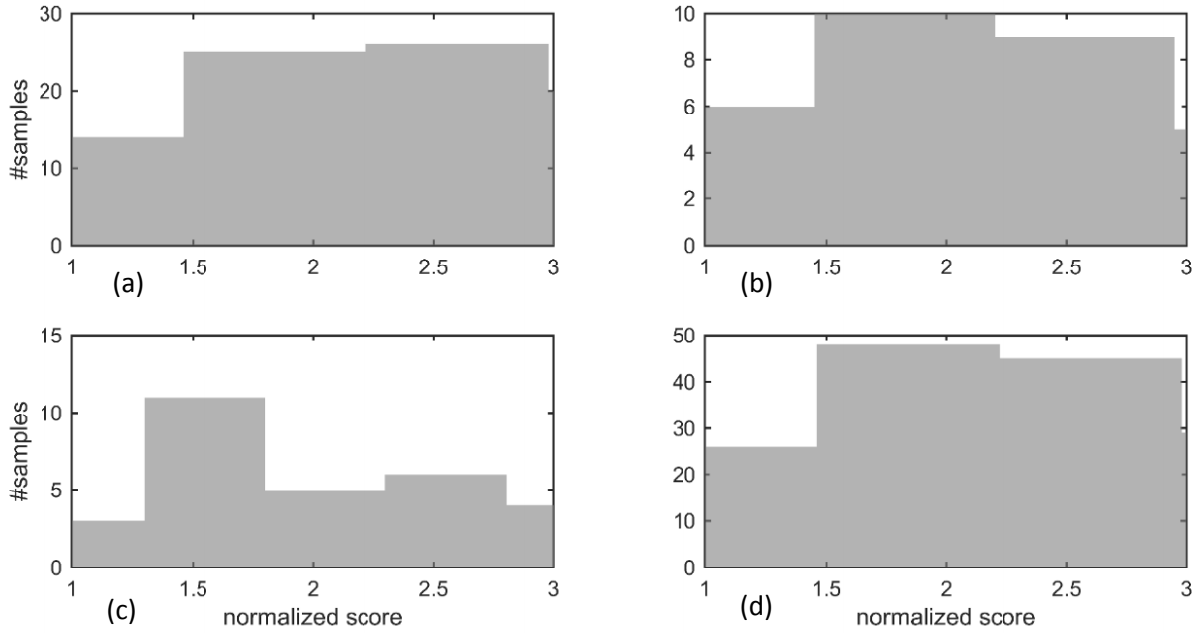


Figure 48: Boston Structure normalized score histograms for (a) Training (b) Validation (c) Test sets (d) All sets

6.3 Case study 3 - Airport terminal

The airport terminal design shown in Figure 50 was the focus of the third case study. This design was inspired by an existing bus terminal shown in Figure 49. The terminal's design was parametrized by 6 variables and included various design objectives. The objectives were (1) the structural weight as described in Case Study 1 (2) the total energy requirements including production and distribution losses (3) the cooling load of the structure as a building in an annual basis (4) the heating load (5) the lighting load and (6) the energy requirements as the sum of the cooling, heating and lighting loads.

The motivation for this case study is that especially the energy simulations to obtain the scores require substantial computational power and therefore one cannot explore the design space in real time. This prevents the designer from fully comprehending the whole design space, realizing its full potential or even rapidly run optimization routines over the selected variables. The use of surrogate modelling could replace the expensive simulation and allow for more deep exploration and optimization over the parameters.

Figure 49: Inspiration for the design case study (Image from [47])

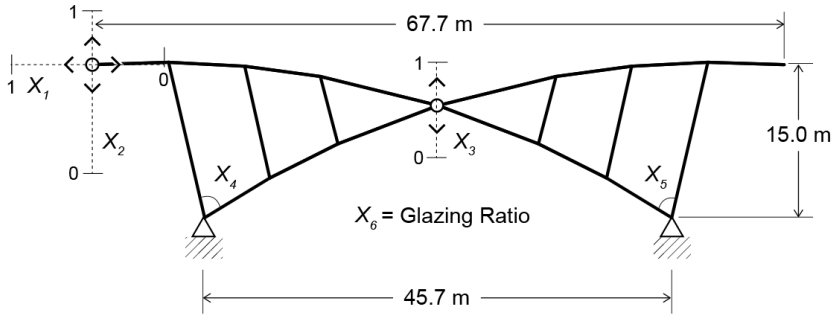


Figure 50: Airport terminal structure topology/geometry and variables x1-x6 (Image from)

The variables for this case study problem for which the structure was parametrized over are: (x1) the horizontal and (x2) the vertical position of the cantilever. The cantilever was chosen to be symmetric for both sides of the structure, (x3) the vertical position of the central node, with its horizontal position always fixed in the middle of the two ground hinge supports, (x4) and (x5) represent the angles of the left and right side respectively of the two truss members joining at the supports and (x6) is the glazing ratio of the windows. The window glazing ratio is important and can affect the energy related scores, since the energy simulation considers sunlight gains for both heating and lighting. Variable summary in Table 22.

Variable	Description
x1	Horizontal position of overhang tip
x2	Vertical position of overhang tip
x3	Vertical position of node on axis of symmetry
x4	Truss inclination 1
x5	Truss inclination 2
x6	Glazing ratio

Table 22: Airport terminal variables

As with the PI structure case study introduced in Section 6.2, the simulation to measure the performance of each design sample was carried out in Rhino/Grasshopper plugins. More information about this

problem, the setup and all the assumptions used in the simulations can be found on [47]. Each design evaluation required approximately 0.2 sec for the structure and 25 sec for the energy, making surrogate modelling an ideal solution.

Because this case study involved energy simulations, a lot of data were collected from simulating the same structure in different climates and orientations. Specifically, the locations considered were Abu Dhabi, Boston and Sydney. They were chosen so that they represent a wide range of different climates. Apart from the location, the orientation was also varied and the corresponding datasets generated. The structure as shown in Figure 8 was placed once so that the horizontal axis was on W-E and once N-S for each location. The N-S orientation datasets are marked as "Rotated" on later figures.

Set	# samples
Training	600
Validation	200
Test	200

Table 23: Airport terminal case study dataset sizes

The histograms for the Energy Overall and Structure performance scores (normalized) the Boston area (W-E orientation) are shown in Figure 52 and Figure 52 respectively for all the sets. It is noted that the normalization was performed on the total pool of training/validation/test data in advance of the random partitioning with the sized seen on Table 23. The histograms for the different sets have a similar distribution, which is desired so that each set is representative of the design space.

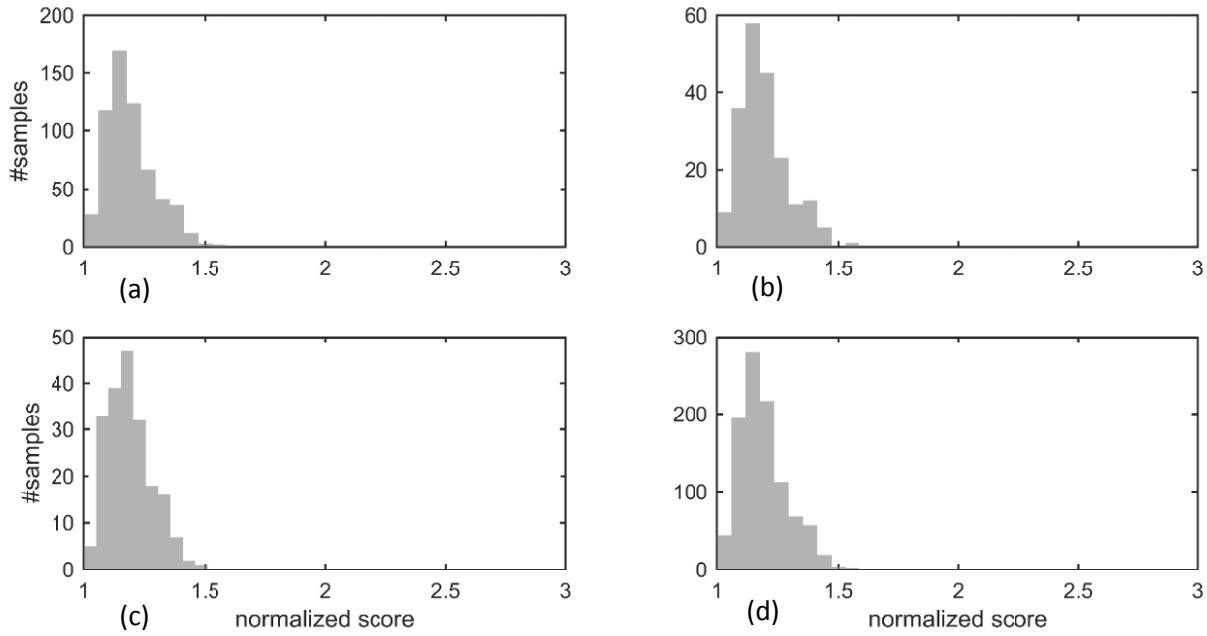


Figure 51: Boston Energy Overall normalized score histograms for (a) Training (b) Validation (c) Test sets (d) All sets

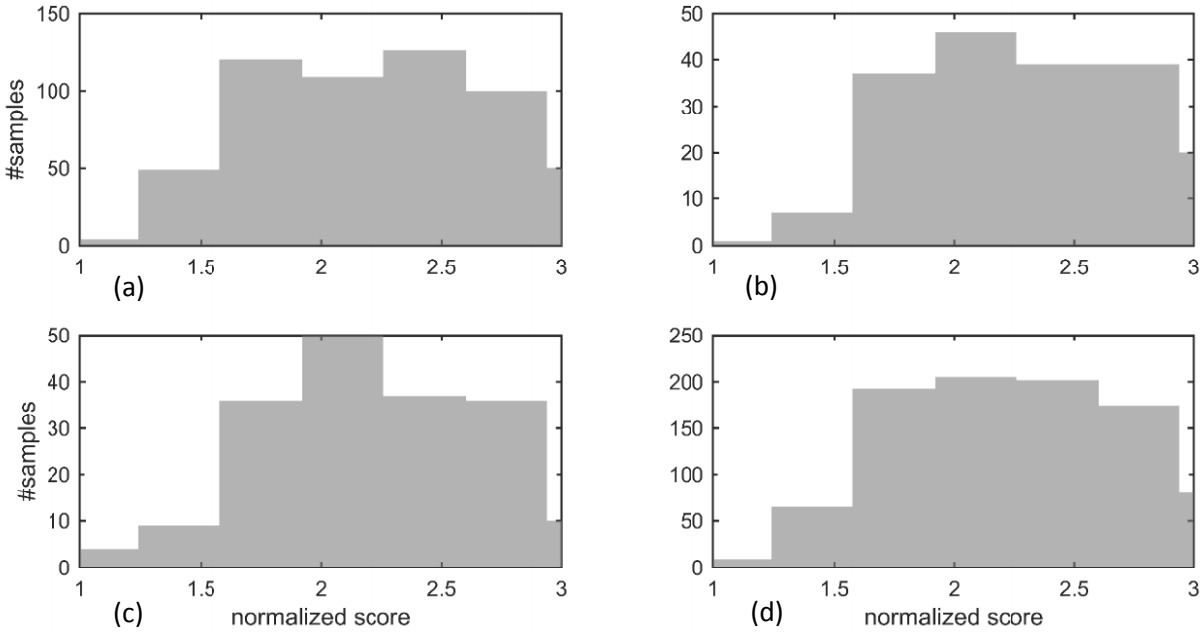


Figure 52: Boston Structure normalized score histograms for (a) Training (b) Validation (c) Test sets (d) All sets

7. Results

For each case study, the data was passed through the robust model comparison framework described in Chapter 5 and the results have been gathered and included in this chapter. For consistency reasons, the same initial framework parameters were set for each case study and for each dataset within. The nuisance parameters examined for each model were set to be the same across the different datasets. One can review those parameters in Appendix A, where the full input/output of the framework is shown for one location and for a single objective dataset (specifically the Energy Overall objective for the Boston dataset for the Airport terminal case study). In the same Appendix, the validation set error is shown for each model and for each nuisance parameter within this model type examined. The parameter with the lowest mean error for the 10 runs was chosen as outlined in Chapter 5.

7.1 Case Study 1 – Grid Truss Bridge

The results of the approximation can be seen in the following figures.

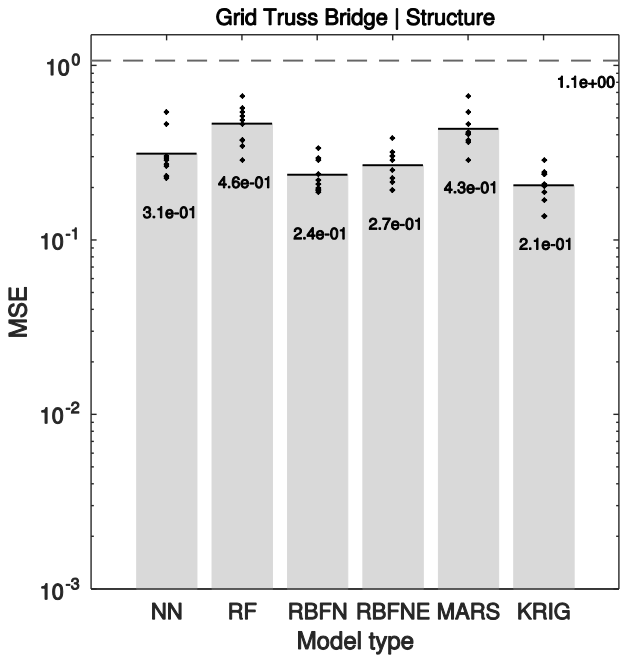


Figure 53: Grid Truss Bridge – model comparison

From the model comparison figure above (Figure 53), it is observed that the models did not achieve a very good fit, with the error quite close to the flat model error. There is a substantial overlap between the runs of all the models, which makes the distinction difficult. The KRIG model was the best-performing,

marginally better than RBFN, which had smaller variability. The scatter plots of Figure 54, show that the predictions for the test set are nearly all marginally inside the ±10% error zone.

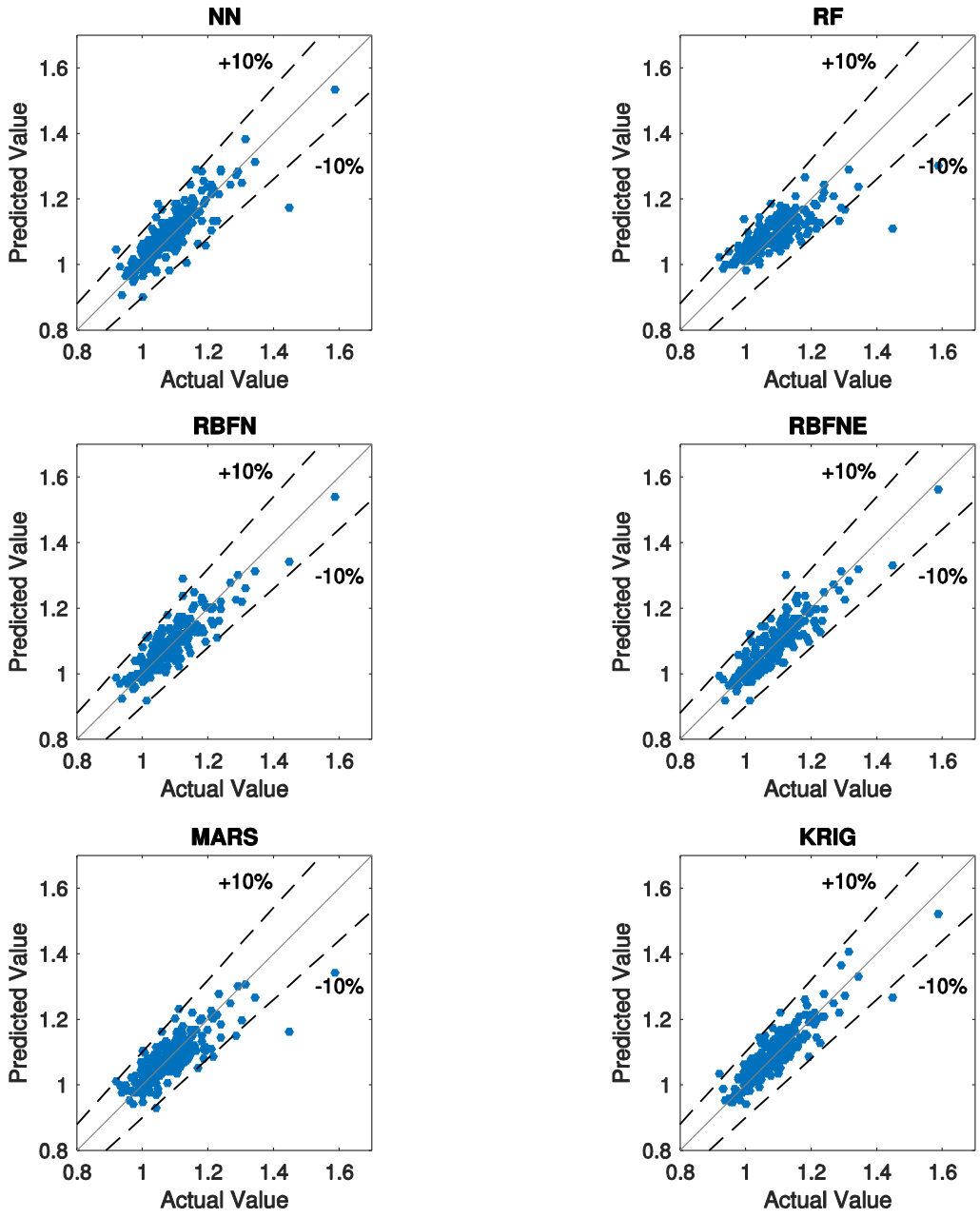


Figure 54: Grid Truss Bridge – performance scatter plots (Test set)

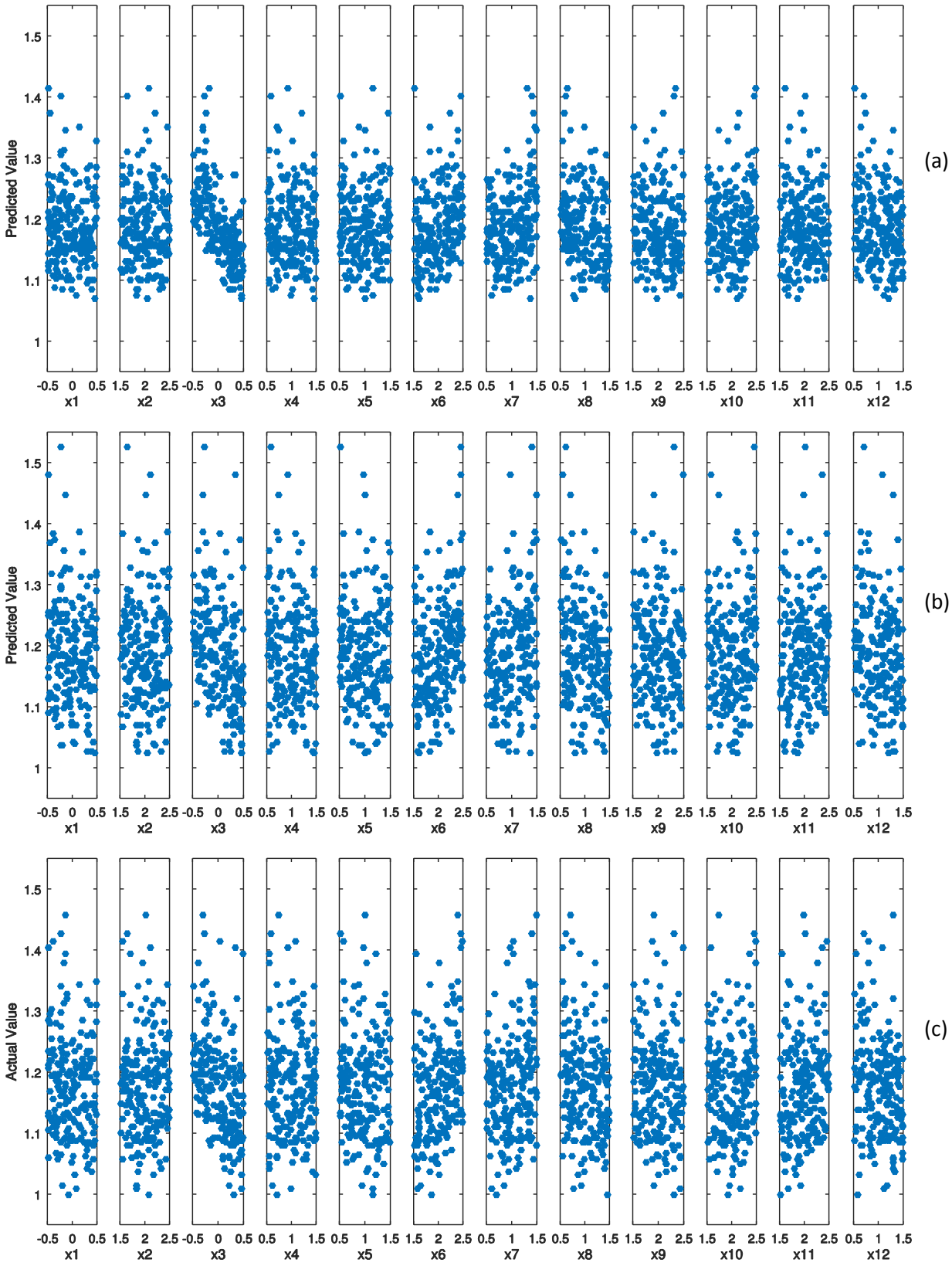


Figure 55: Grid Truss Bridge variable scatter plots (a) RF-worst (b) KRIG-best (c) Actual

Results concerning the rank errors are also presented, for the best-performing model (KRIG). For this model, a permutation rank test was performed and the results are shown in Figure 56. It should be noted that for TMRE, TRE and TFE the number of top performance designs was set to 20, and the permutation test had 1000 iterations.

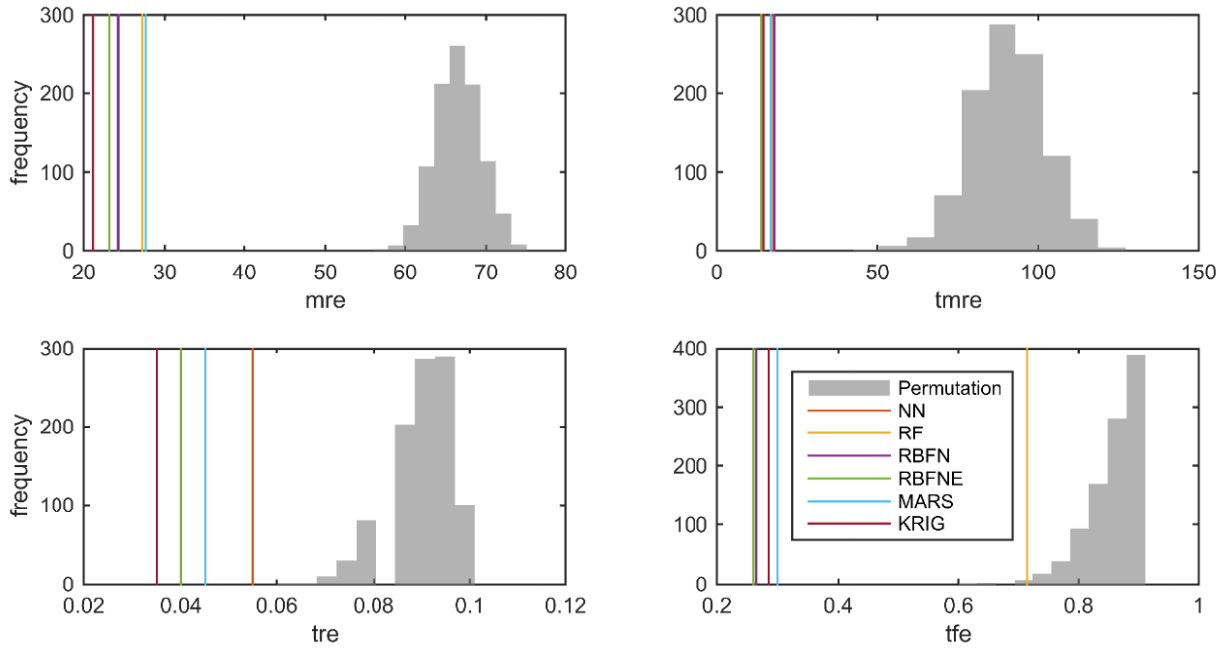


Figure 56: Grid Truss Bridge – permutation rank test (Test set)

The vertical lines for all the model errors do not intersect the histograms, so the p-values are very close to zero and subsequently it is inferred that the models has a predictive ability on ranks. Specifically for the KRIG model that performed best in the mean MSE, the TMRE is 14, from TRE it is calculated that 13 of the actual top 20 designs were predicted in the top 20 and from TFE it can be calculated that the worst prediction of the actual top 20 designs was rank 77. Those results seem to suggest that if someone looks only at the predicted top 20 designs, there is a good probability that they represent the actual top 20, but the rank values themselves are not very accurate.

7.2 Case Study 2 – PI Structure

The locations (for the energy simulations) examined are summarized in Table 24. For each location there the horizontal axis of the structure as shown in Figure 46 is aligned with North-South direction. The datasets examined (same for each location) are summarized in Table 25. Regarding this table, the Cooling, Heating and Lighting are the respective energy loads required annually. More details of the simulation can be found on the references in Chapter 6. The Heating+Cooling+Lighting is the sum of those three loads,

while for the Energy Overall dataset, special coefficients and formulas have been further applied to each individual load to account for the required production and transmission of the energy in addition to the building's requirements. Lastly, the Structure dataset refers to the structural weight score.

	Location	Comments
1	Boston	Moderate climate. Requires both cooling and heating
2	Sydney	Primarily cooling, moderate heating requirements

Table 24: Locations considered for PI Structure

	Performance score dataset
1	Energy Overall
2	Structure
3	Heating+Cooling+Lighting
4	Cooling
5	Heating
6	Lighting

Table 25: Datasets for each location for PI Structure

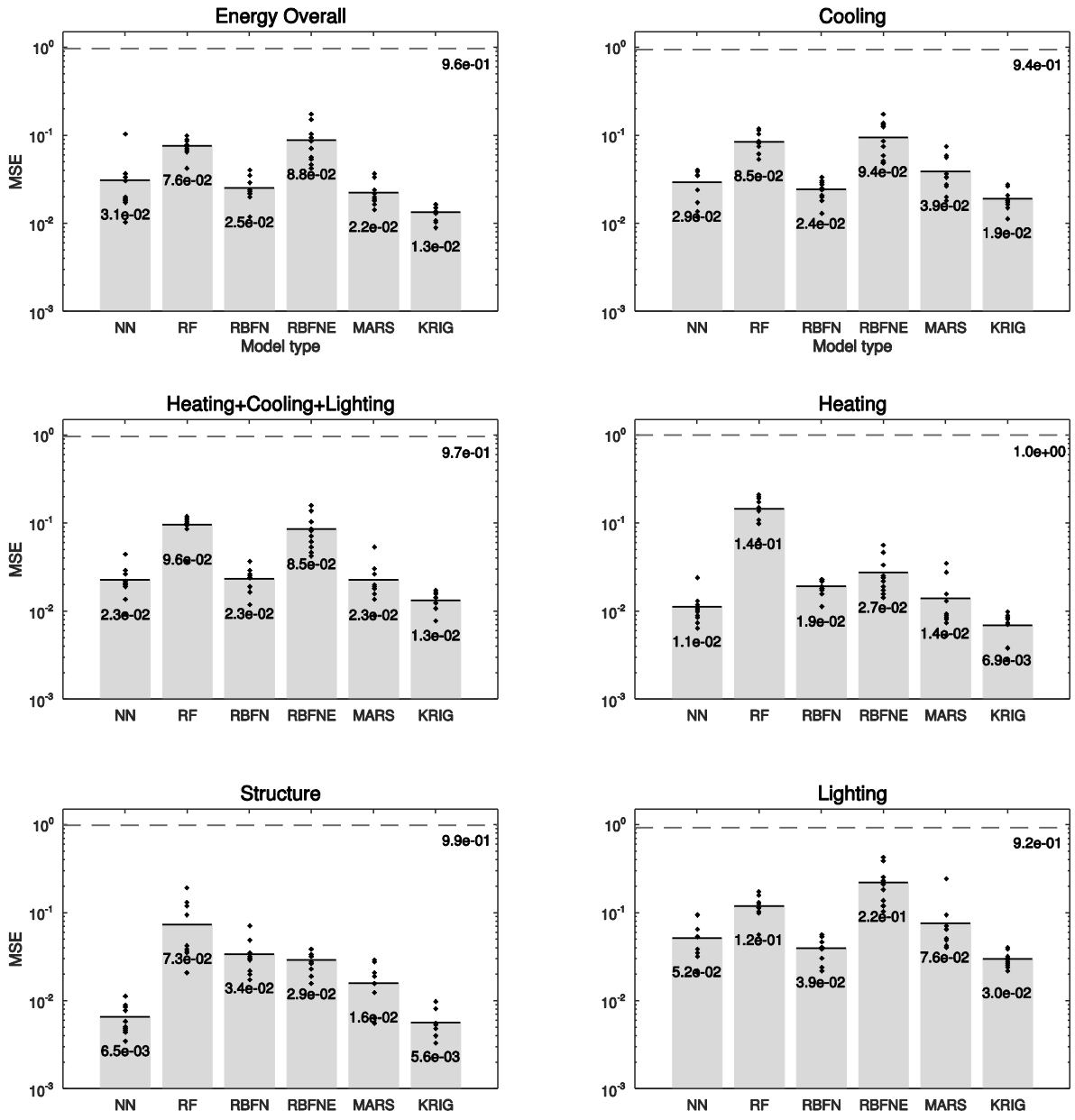


Figure 57: PI Structure – Boston – model comparison

From Figure 57 it can be observed that the approximation was very good for all the performance metrics with models having different performances within each performance metric. The approximation has been at least one order of magnitude better than the "flat" model error. One can notice that the relative performance of the models for each performance dataset are approximately the same. The KRIG models performed best in all performance scores, with the NN models also performing very well. At the same time, the RF and RBFNE models performed the worst.

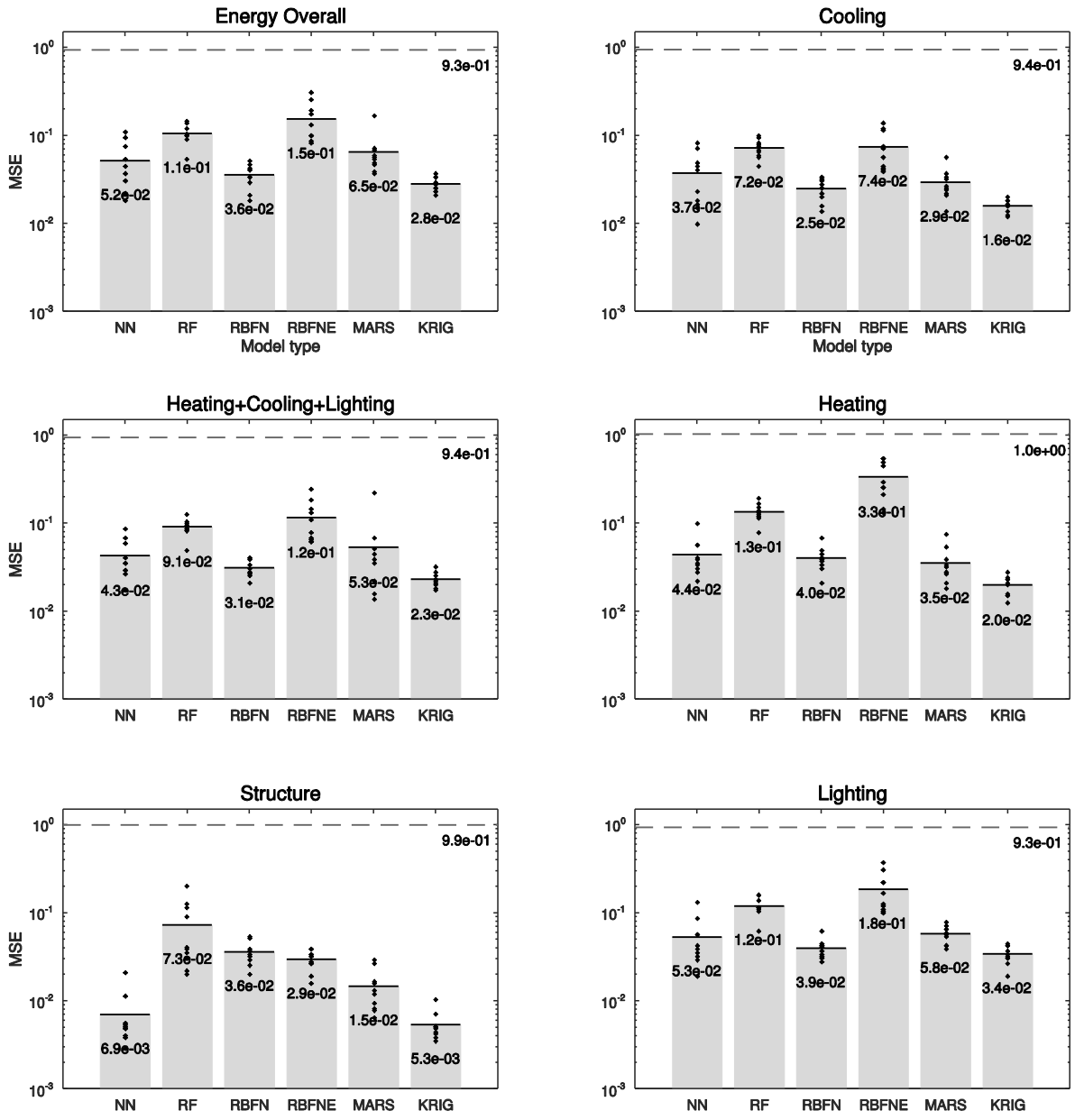


Figure 58: PI Structure – Sydney – model comparison

From Figure 58 it can be observed that the model error performance is very similar to the Boston model comparison. Again, the best model error is at least one order of magnitude better than the "flat" model and for the Structure is nearly three orders of magnitude for the NN and KRIG models. The structure scores are the same as the Boston runs, since the location does not influence the structural weight score. It could have influence through different construction standards, but that was not considered in the analysis. It is included in this figure, for consistency reasons.

All the Predicted vs. Actual performance scatter plots are included in Appendix B and C. In majority, the predicted scores lie in the ±10% error regions. The ones for the Energy Overall of Boston and Structure for

which the models performed the best are also included below for visual comparison (Figure 59 and Figure 61). In addition, the performance vs. variables scatter plots are also included for Energy Overall of Boston for RBFNE (worst performance) and KRIG (best performance) in Figure 60.

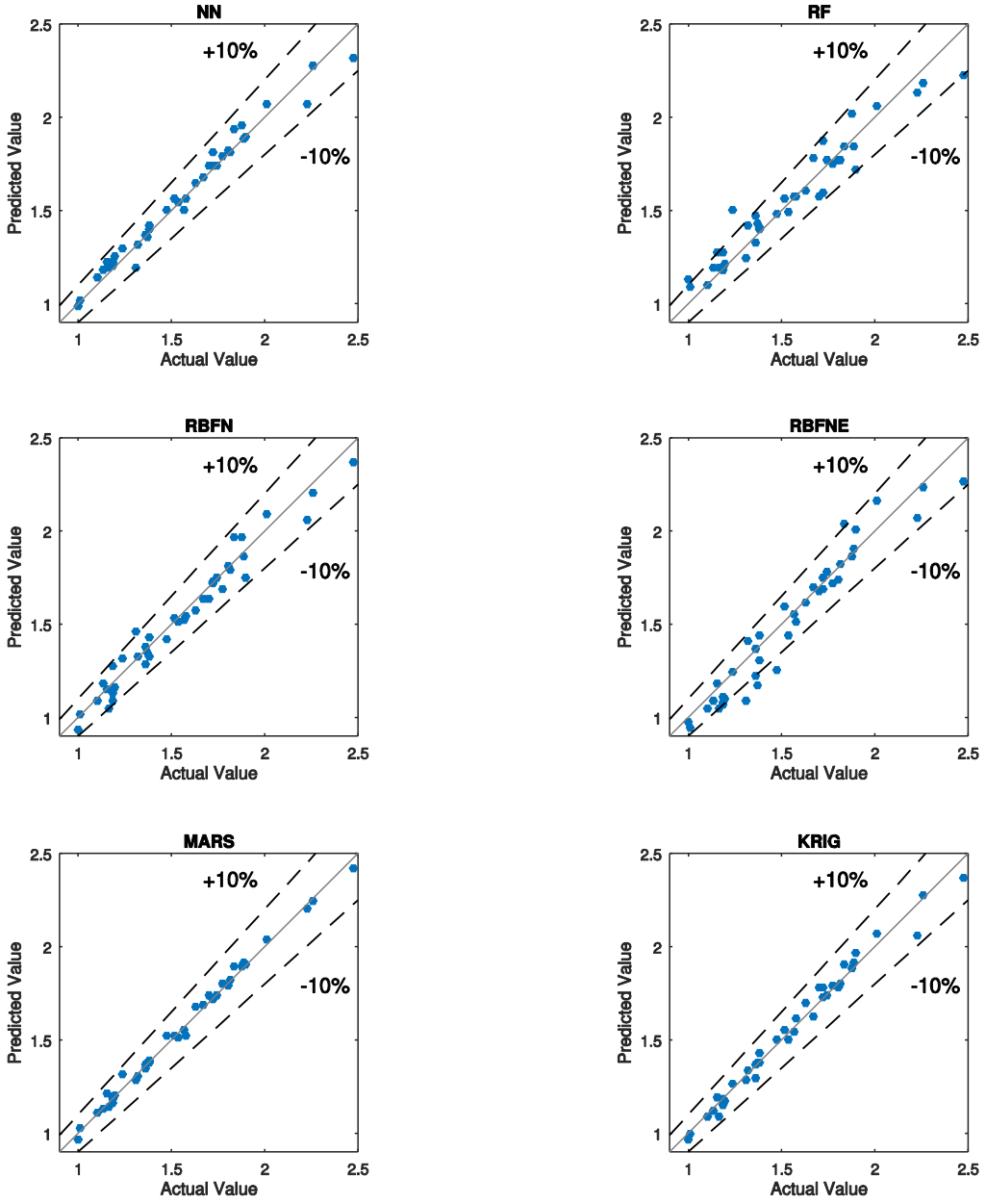


Figure 59: PI Structure – Boston – Energy Overall performance scatter plots

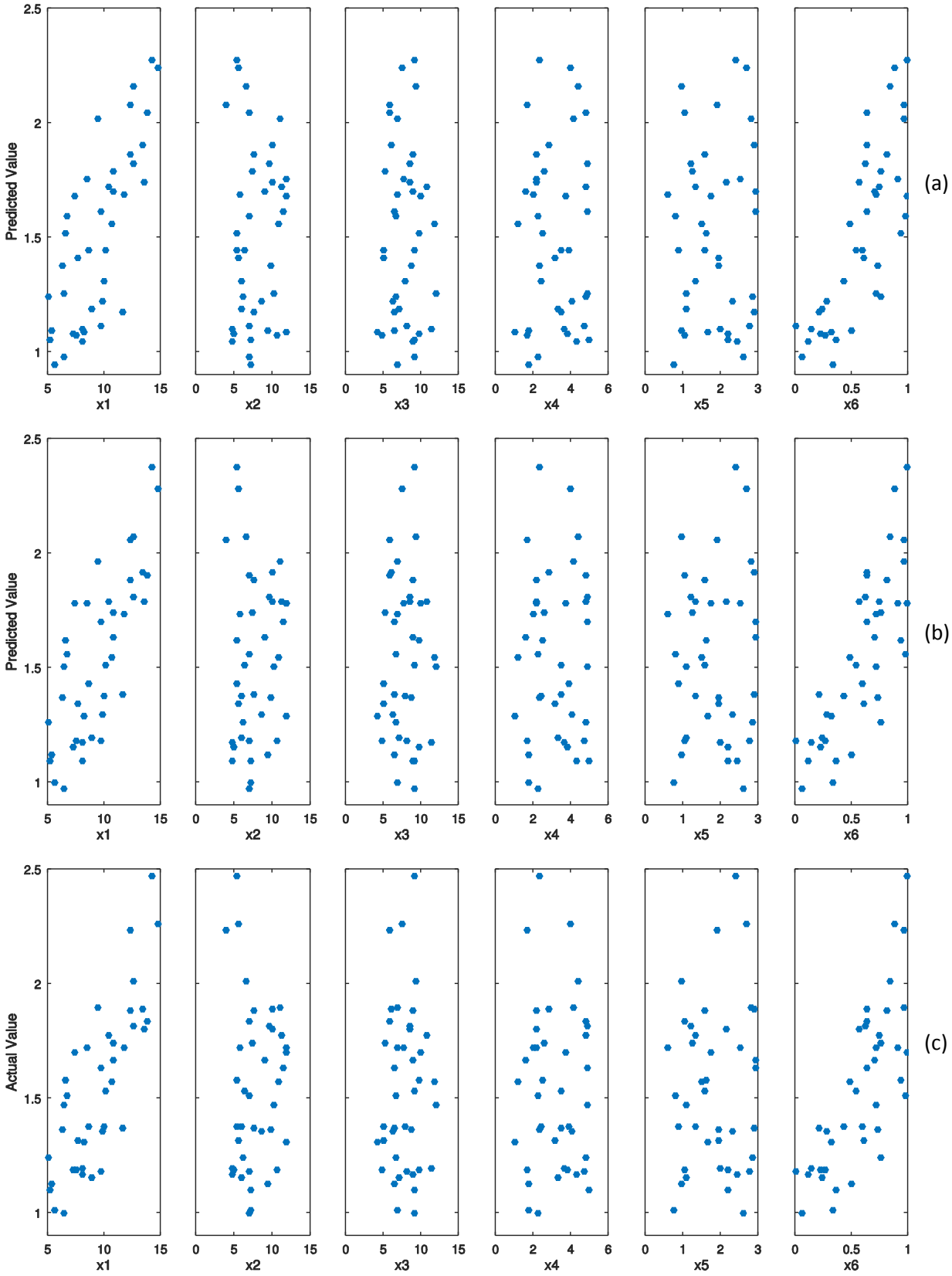


Figure 60: PI Structure–Boston–Energy Overall scatter plots (a) RBFE-worst (b) KRIG-best (c) Actual

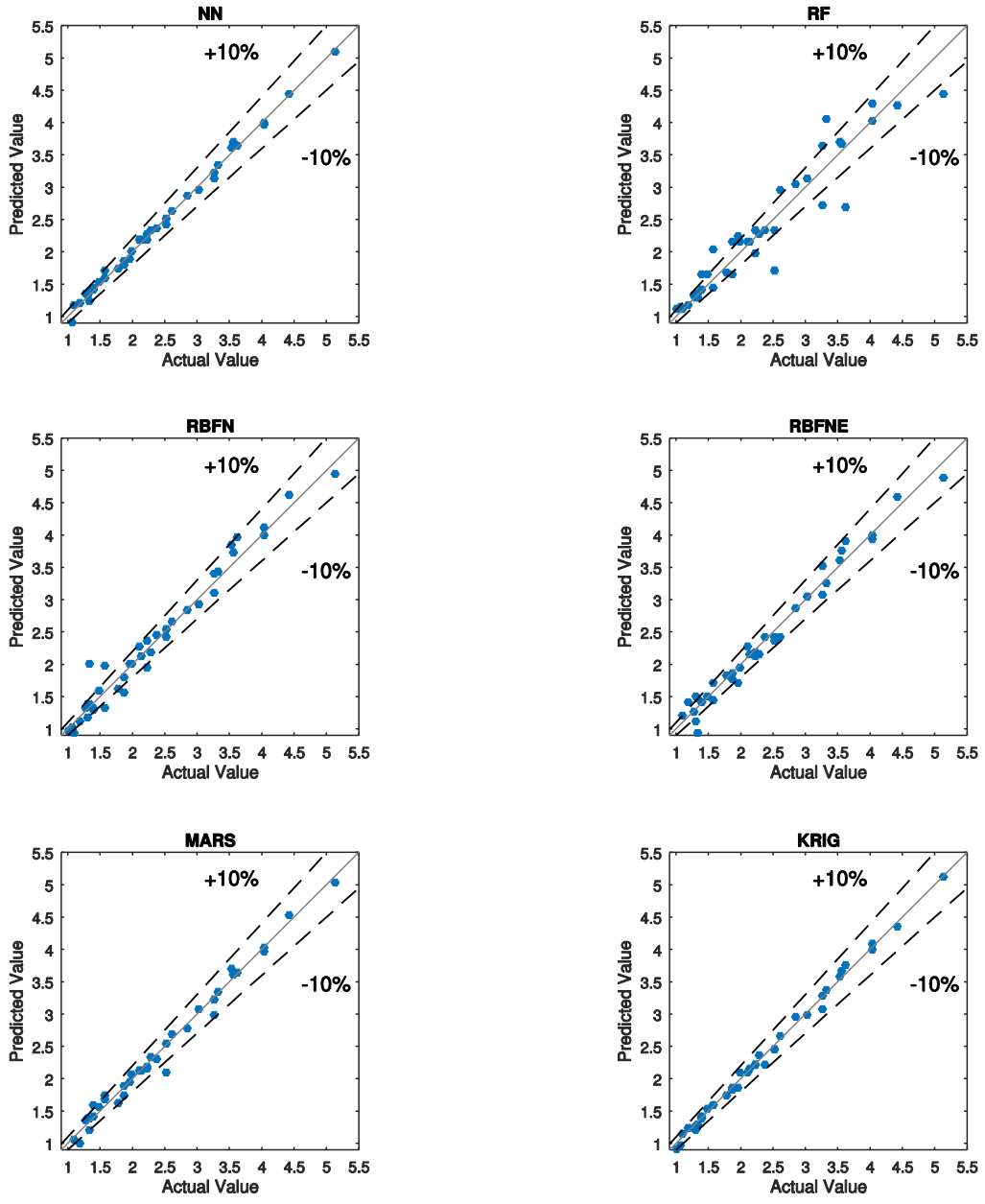


Figure 61: PI Structure – Boston – Structure performance scatter plots

The rank error permutation test results are included in Figure 62 for the Sydney location, the Structure dataset and the best-performing model for that dataset (KRIG). It should be noted that for TMRE, TRE and TFE the number of top performance designs was set to 20, and the permutation test had 1000 iterations.

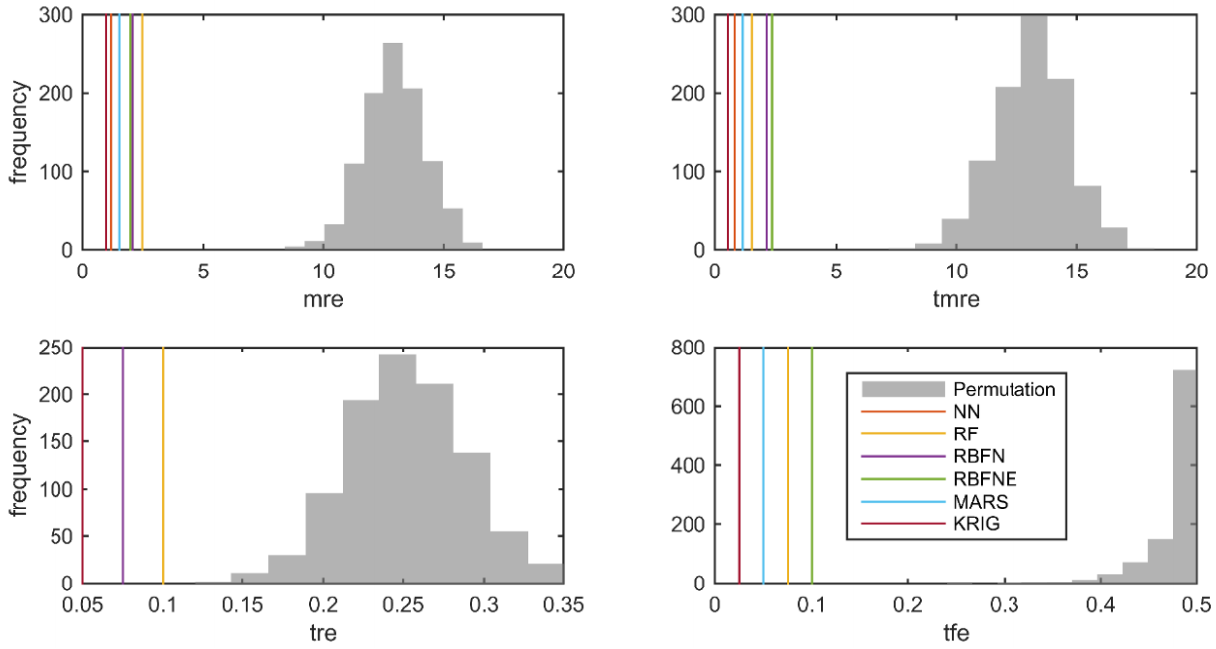


Figure 62: PI Structure-Sydney-Structure- permutation rank test results (Test set)

The p-values for all the rank errors of all the models are practically zero, since the vertical lines of the predicted rank errors do not overlap the histograms. For the KRIG model, which performed best in the respective mean MSE, the TMRE value is 0.5, which means almost perfect rank prediction for the top 20 designs. This means that the rank predictive ability of the model is very good and one can inspect the predicted ranks with some confidence during early-stage design, where the performance rank (in the Structure score in this case) could be equally important with the actual score value.

7.3 Case Study 3 – Airport terminal

For this case study, the locations considered are outlined in Table 26 and the performance scores considered and whose results are shown here were the same as the ones in the PI Structure case study displayed in Table 25. For each location there is also a "Rotated" version which means that the horizontal axis of the structure as shown in Figure 46 is aligned with North-South as opposed to East-West for the locations not marked "Rotated".

	Location	Comments
1	Abu Dhabi	Intense cooling requirements
2	Abu Dhabi Rotated	Intense cooling requirements
3	Boston	Moderate climate. Requires both cooling and heating
4	Boston Rotated	Moderate climate. Requires both cooling and heating
5	Sydney	Primarily cooling, moderate heating requirements
6	Sydney Rotated	Primarily cooling, moderate heating requirements

Table 26: Locations considered for Airport terminal

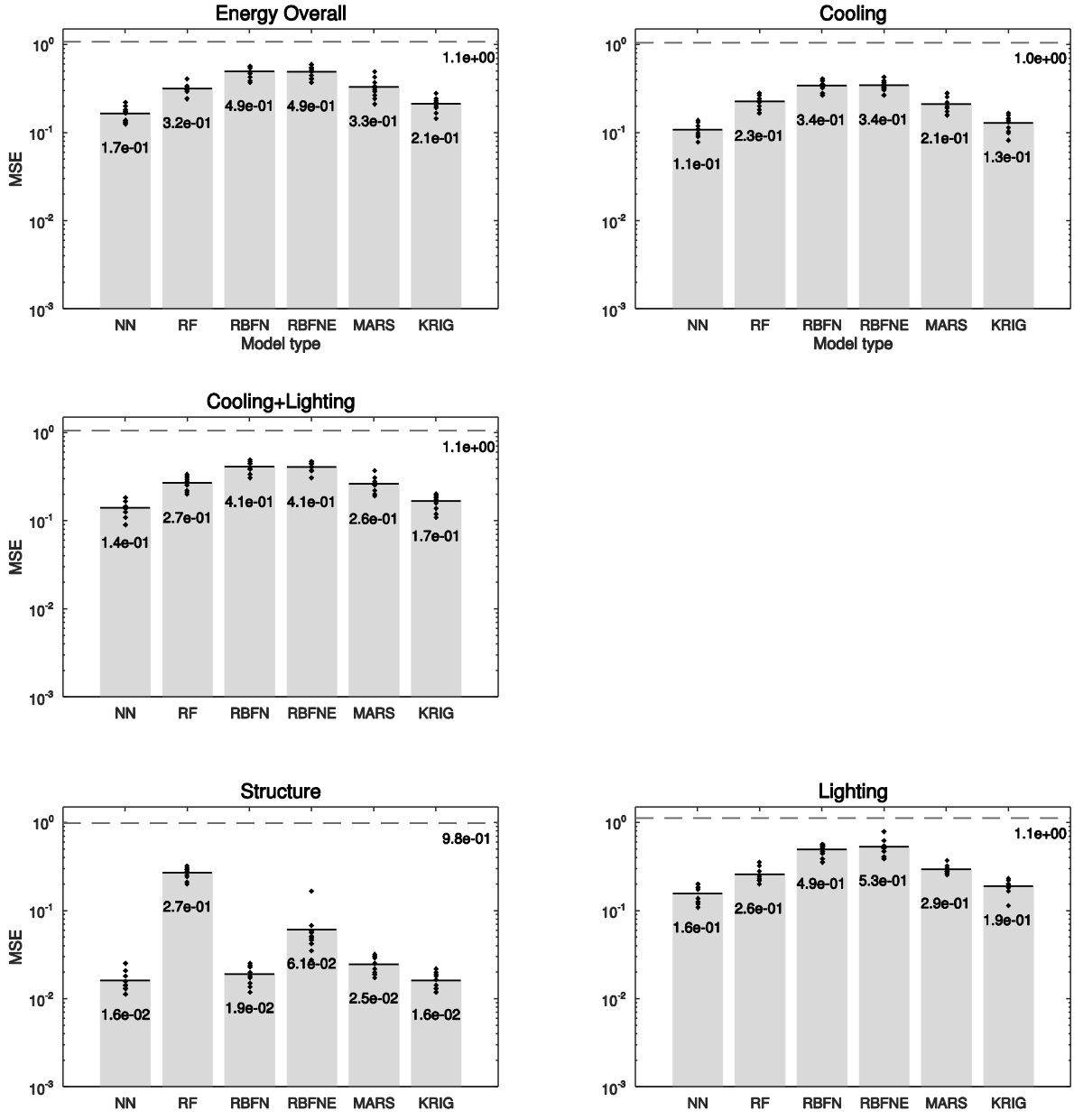


Figure 63: Airport terminal – Abu Dhabi – model comparison

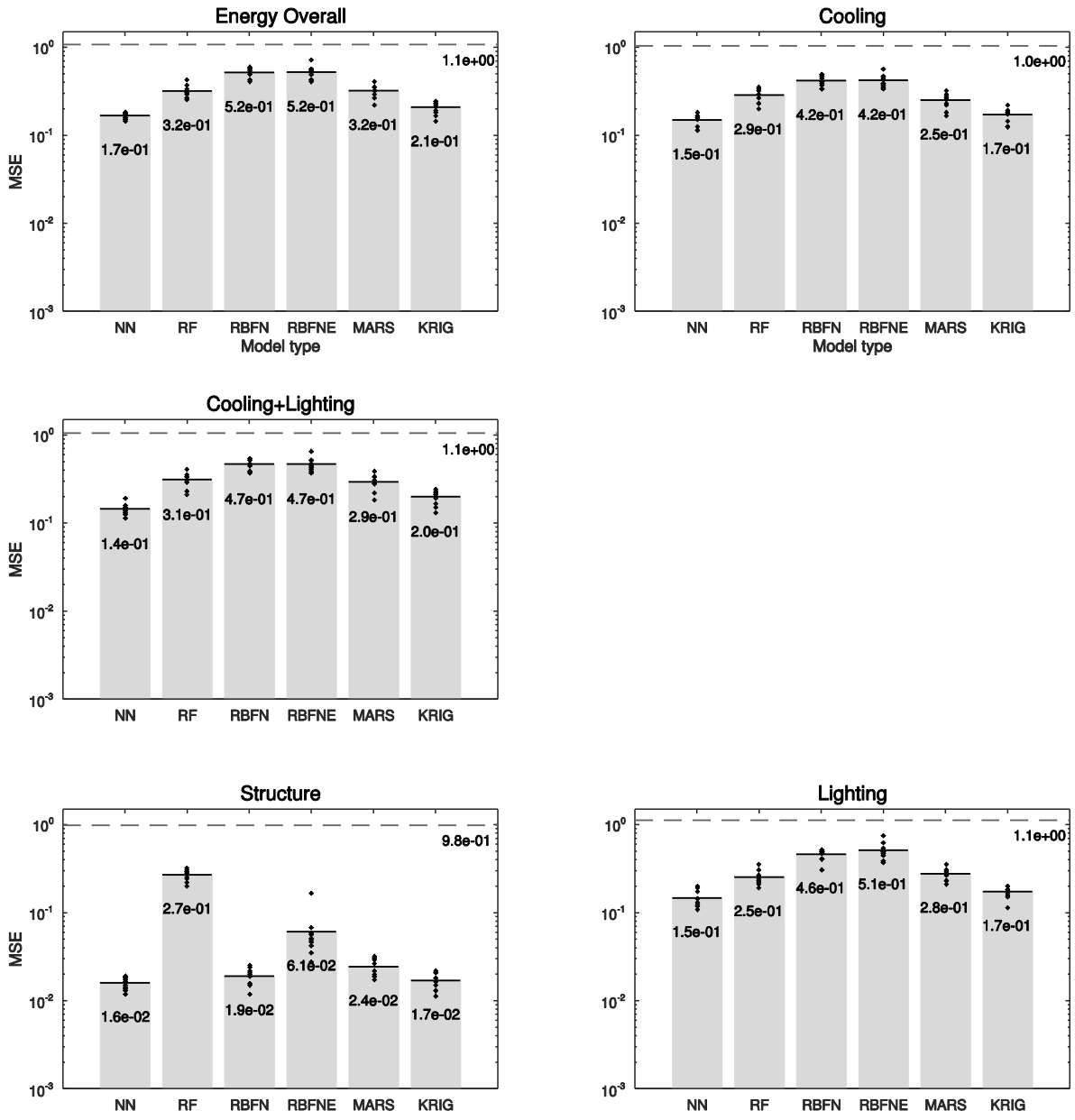


Figure 64: Airport terminal – Abu Dhabi Rotated – model comparison

For the both Abu Dhabi orientations tested, there was no Heating load score. The performance of the models was not good for the energy related scores. The NN model (along with the KRIG), were the ones that performed the best for those scores, achieving almost one order of magnitude better results than the "flat" model error. On the scatter plot of the test set Actual vs. Predicted values for the Energy Overall performance of the NN for the Abu Dhabi set, in Appendix B, shows that all the normalized predicted performances are marginally on the ±10% error range, which means that the accuracy level is satisfactory. For the Structure score it seems (as also in the PI structure case study) that it can be approximated with very low error.

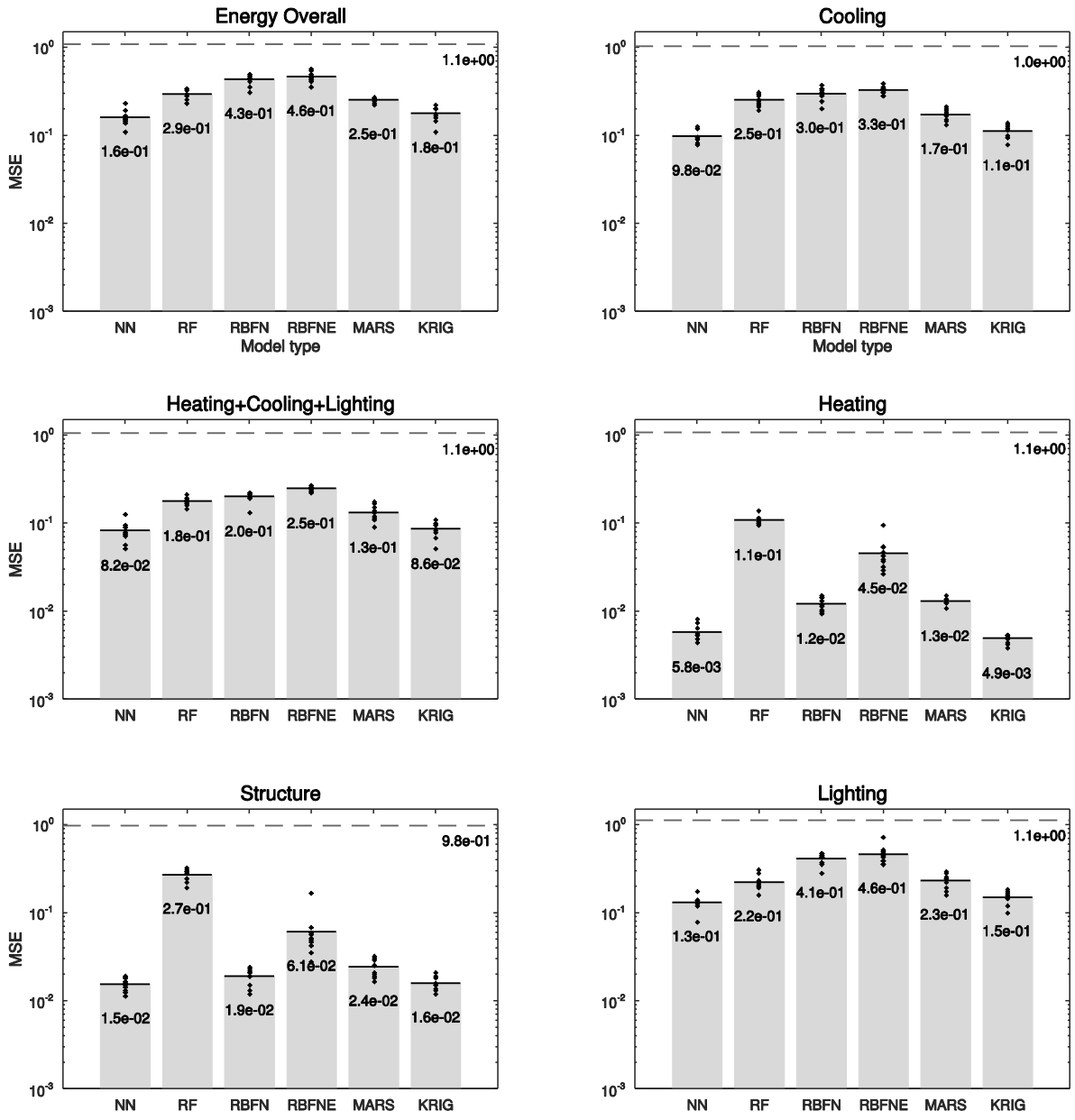


Figure 65: Airport terminal – Boston – model comparison

The Heating load performance score was approximated substantially better than Cooling and Lighting. This was expected since the Heating is governed by the flux equations which are simpler compared to the other scores. Because the Cooling, Heating and Lighting loads have comparable contributions to the total energy requirement load, the H+C+L and Energy Overall have also been poorly approximated.

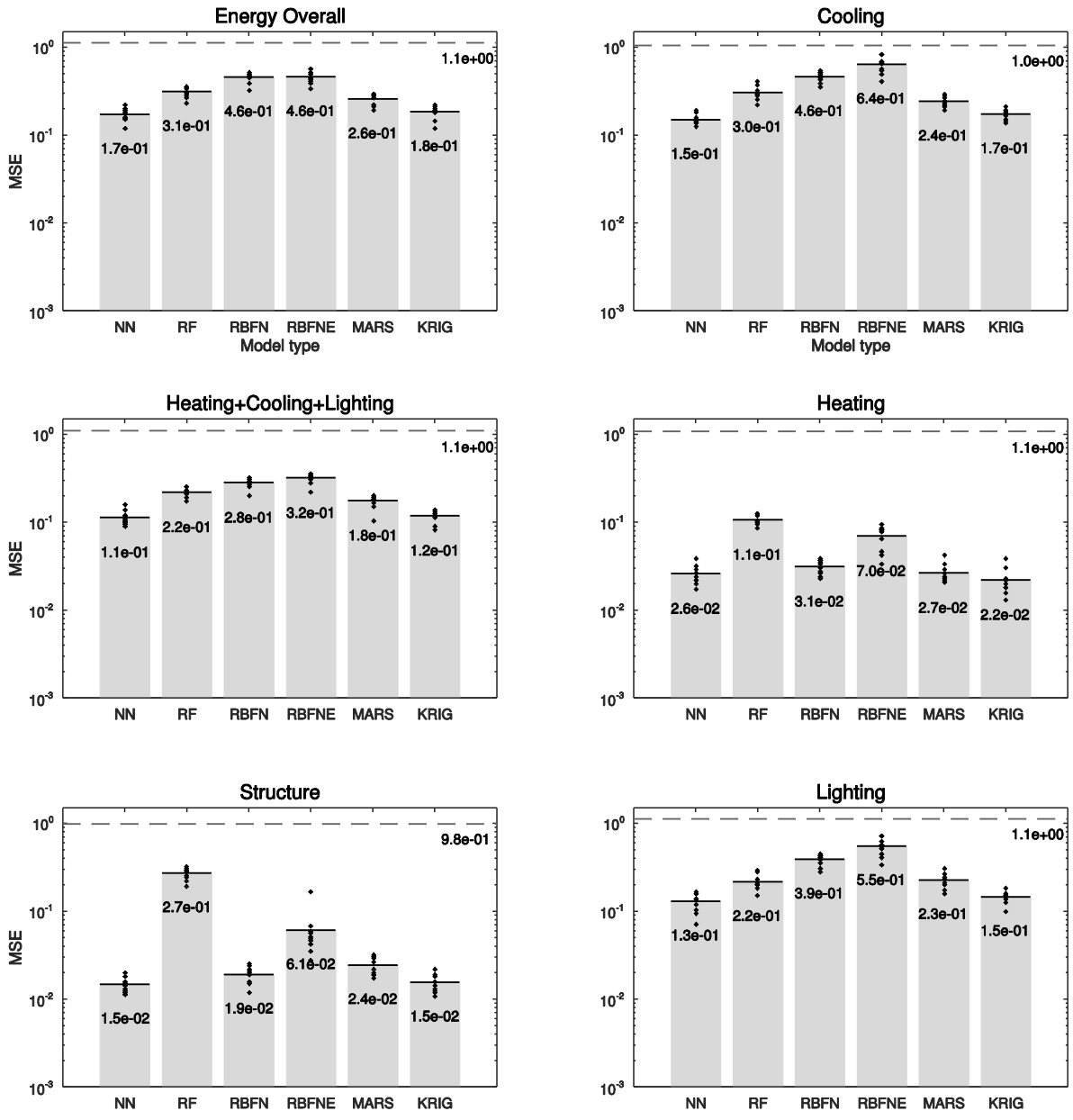


Figure 66: Airport terminal – Boston Rotated – model comparison

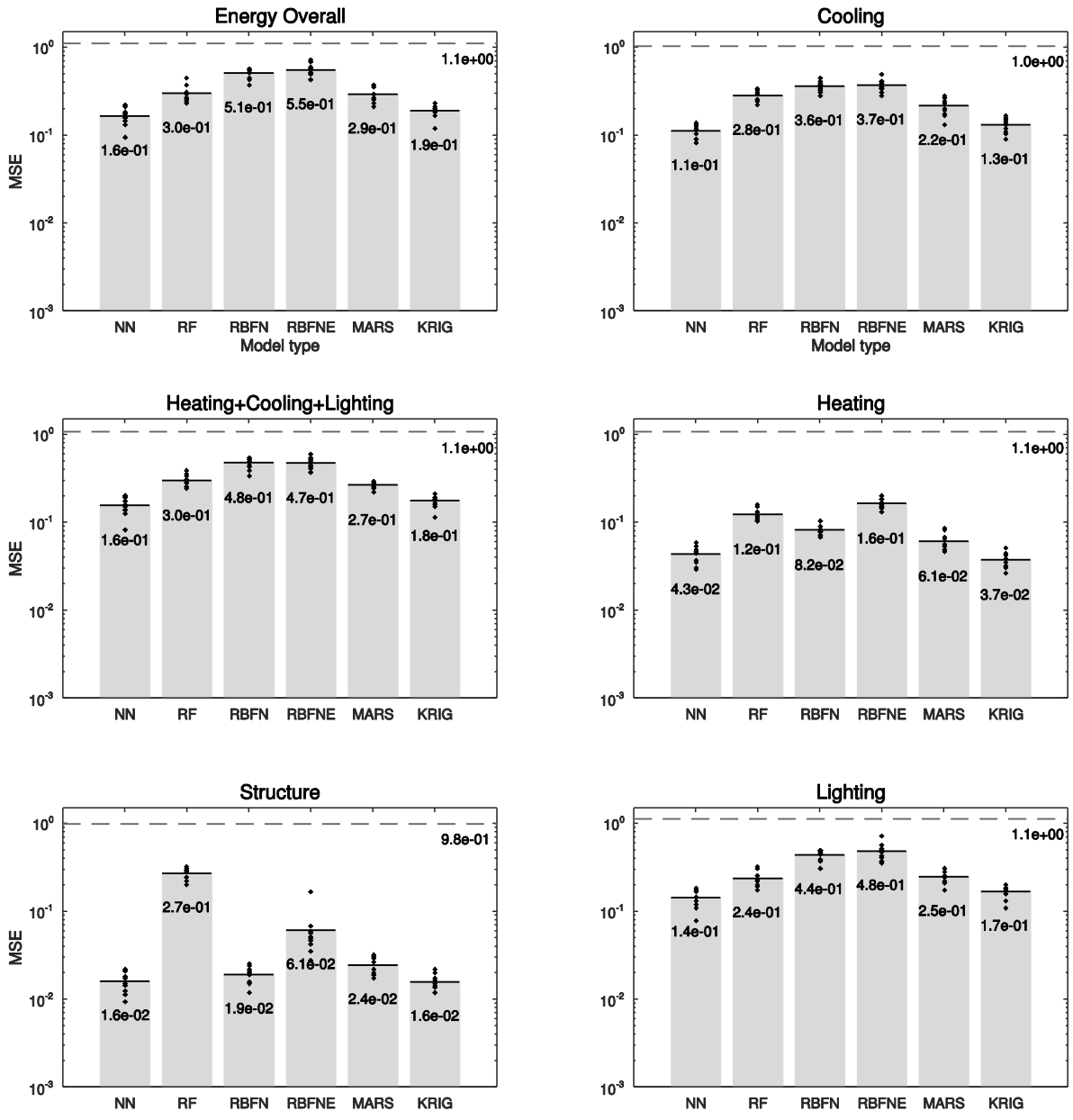


Figure 67: Airport terminal – Sydney – model comparison

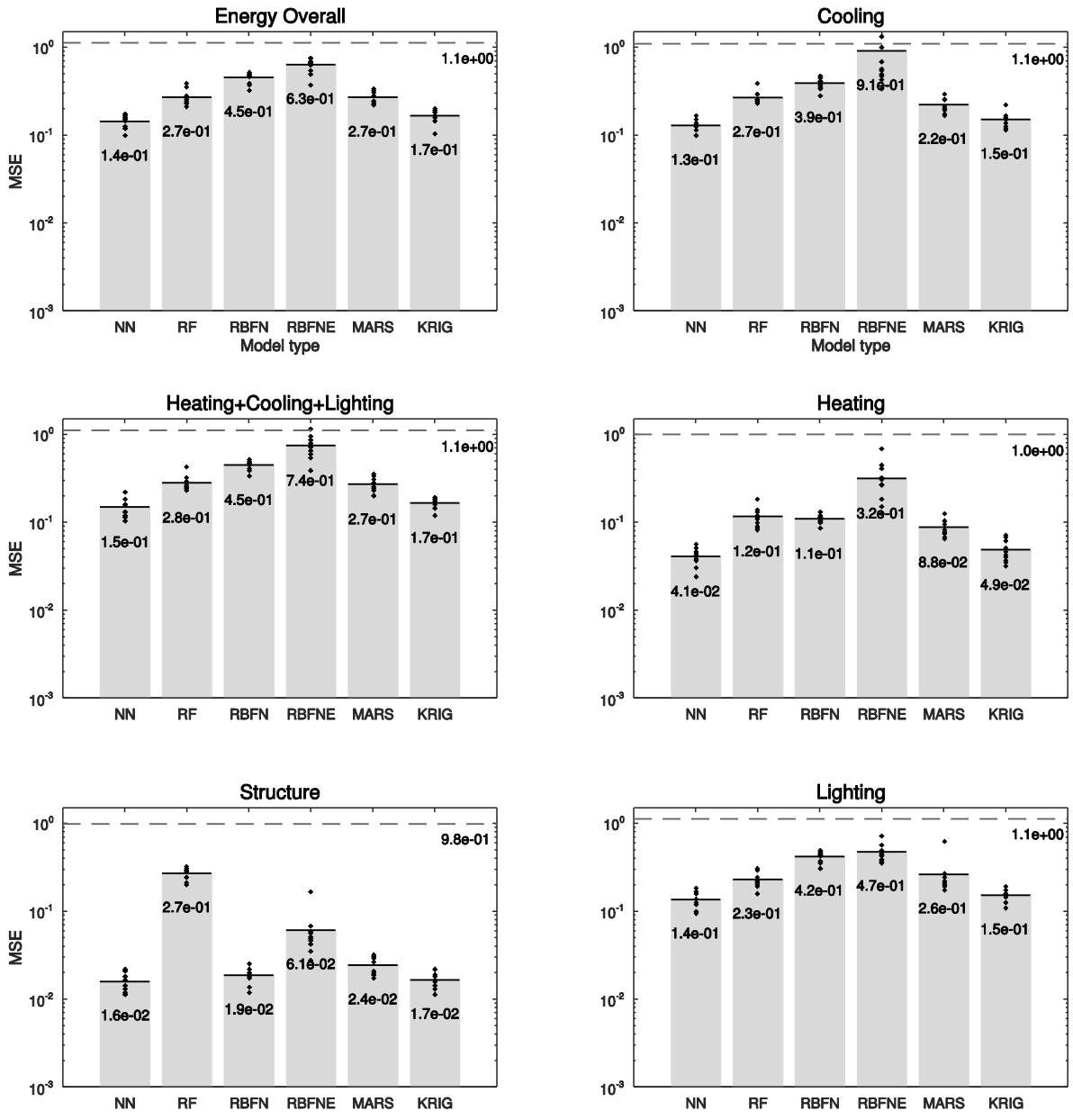


Figure 68: Airport terminal – Sydney Rotated – model comparison

Some representative performance scatter plots are included here; a poorly approximated dataset from most models (Boston-Lighting: Figure 69), and one with a much lower mean MSE (Sydney Rotated-Heating: Figure 70).

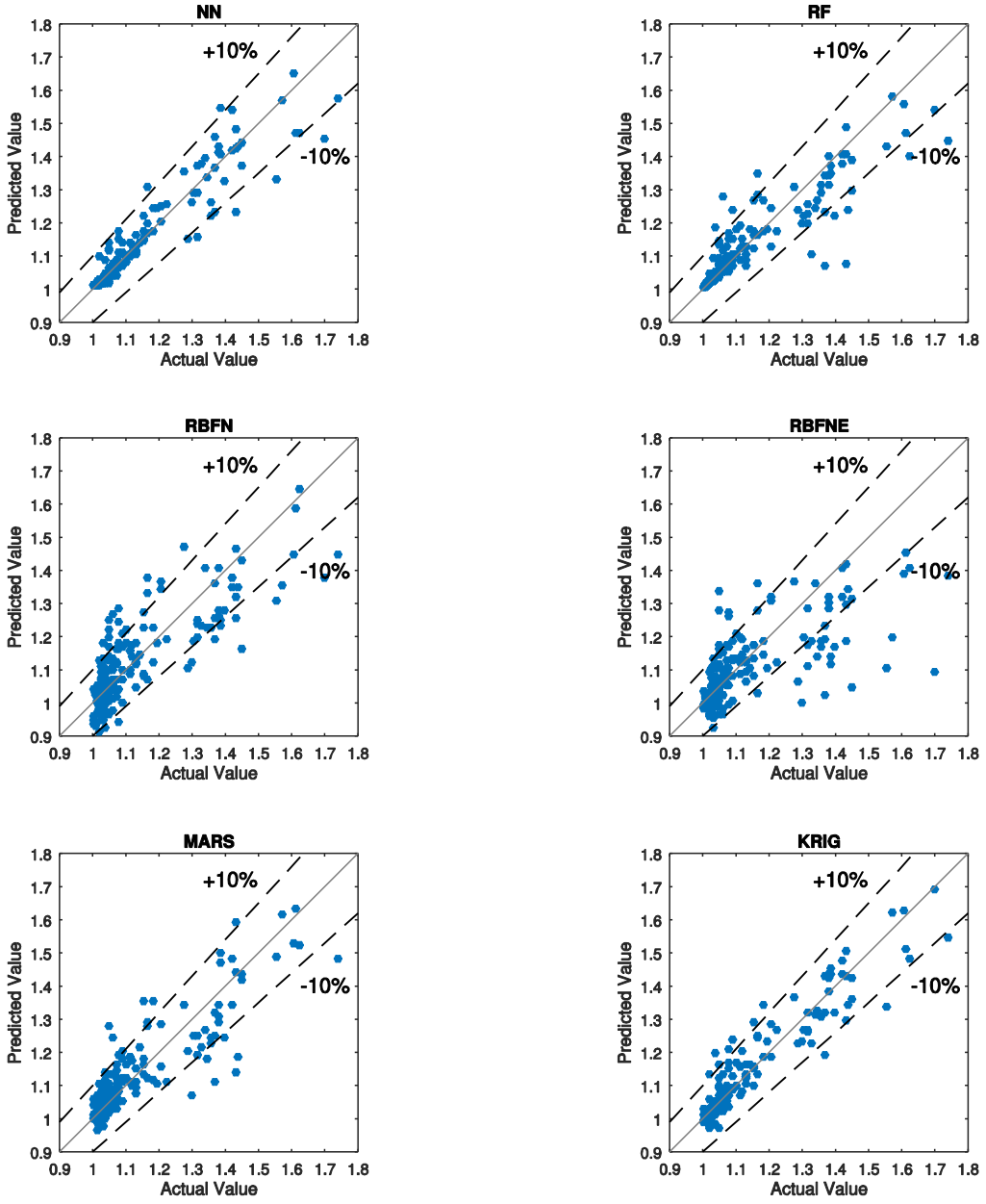


Figure 69: Airport terminal – Boston – Lighting performance scatter plots

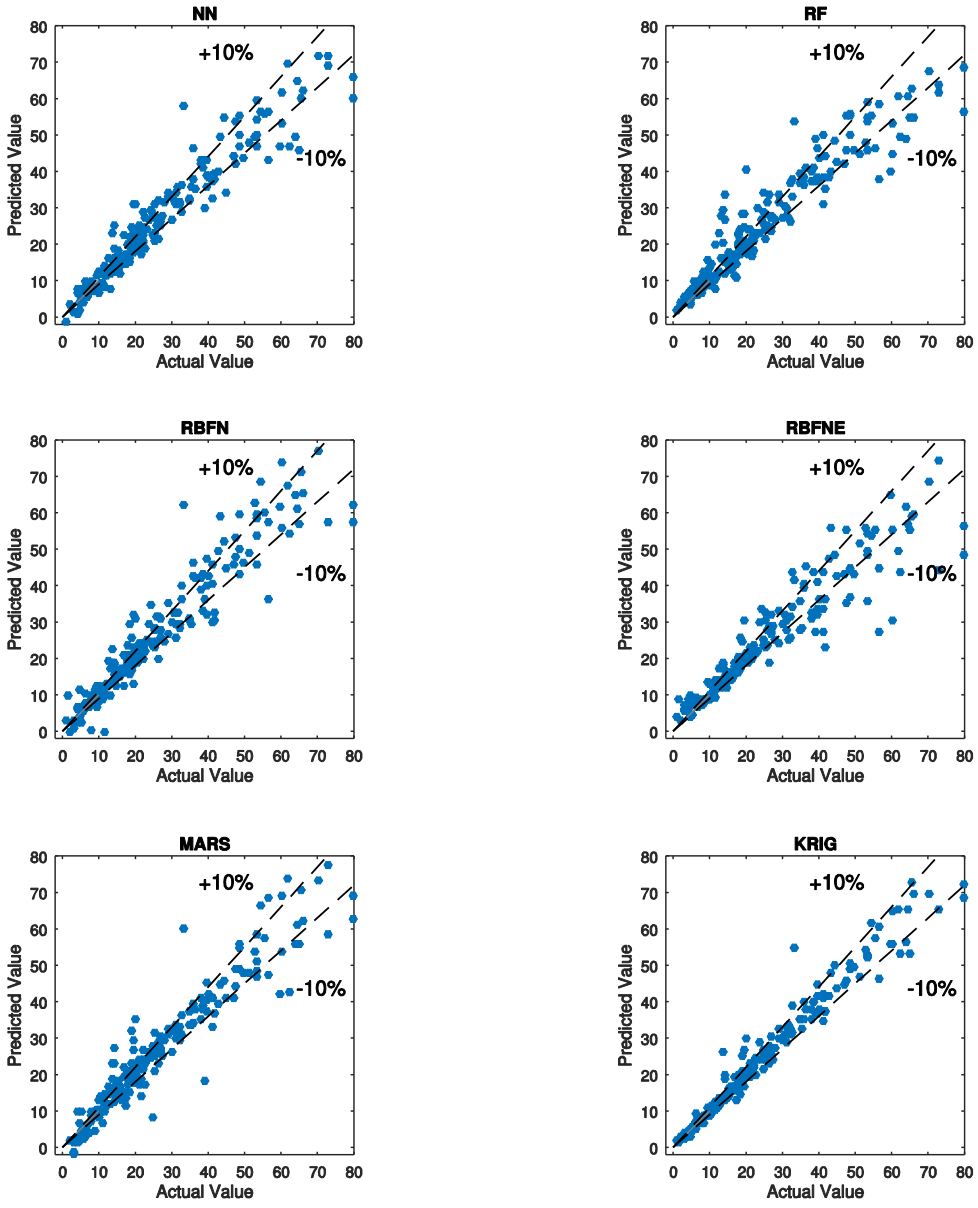


Figure 70: Airport terminal – Sydney Rotated – Heating performance scatter plots

The test set scatter plots of Figure 70, reveal some very interesting characteristics of the results for this performance metric. Firstly, it has a significantly bigger score range than most other performance metrics. With the scores being normalized so that the lowest Heating load receives a score of 1, this means that there exist designs that perform nearly 80 times worst in Heating. However, the Energy Overall scores have a max range of approximately 1.7, which leads to the conclusion that, the Heating load can be extremely increased without affecting the total energy load with the same scale. This was expected because the location was Sydney, a warm climate. Furthermore, this special characteristic of this dataset, revealed in the scatter plots, in a way disproves the low mean validation MSE, since it is observed that the range in many models exceeds the ±10% margins. This showcases once again the great importance and nuisances of normalization schemes as well as of examining the data visually and not relying solely on quantitative metrics.

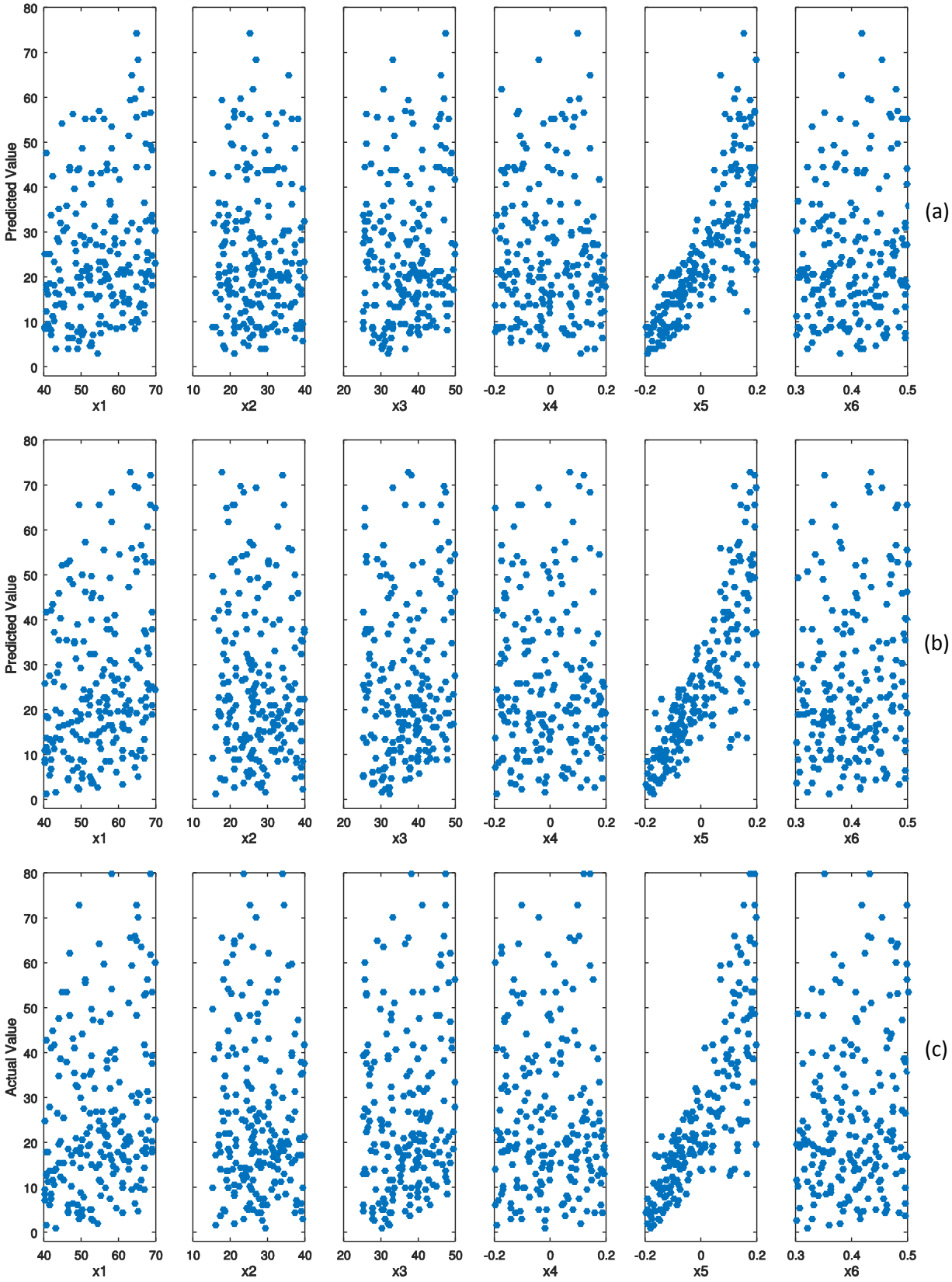


Figure 71: Airport terminal scatter plots (a) RBFE-worst (b) KRIG-best (c) Actual

It would be very interesting to examine a design which performs suboptimal in the Heating score, but taking advantage of the fact that it does not contribute much to the overall consumption and gain substantially in Lighting and/or Cooling. For example by placing large openings in the South direction (because Sydney is in the southern hemisphere), important gains could be made in the Lighting load year round, with a potential (but less important) rise in the Heating load requirement.

Some results concerning the rank errors are also presented. For the Boston location and for the Energy Overall dataset, the best performing model in terms of mean MSE was NN. For this model, a permutation rank test was performed and the results are shown in Figure 72. It should be noted that for TMRE, TRE and TFE the number of top performance designs was set to 20, and the permutation test had 1000 iterations.

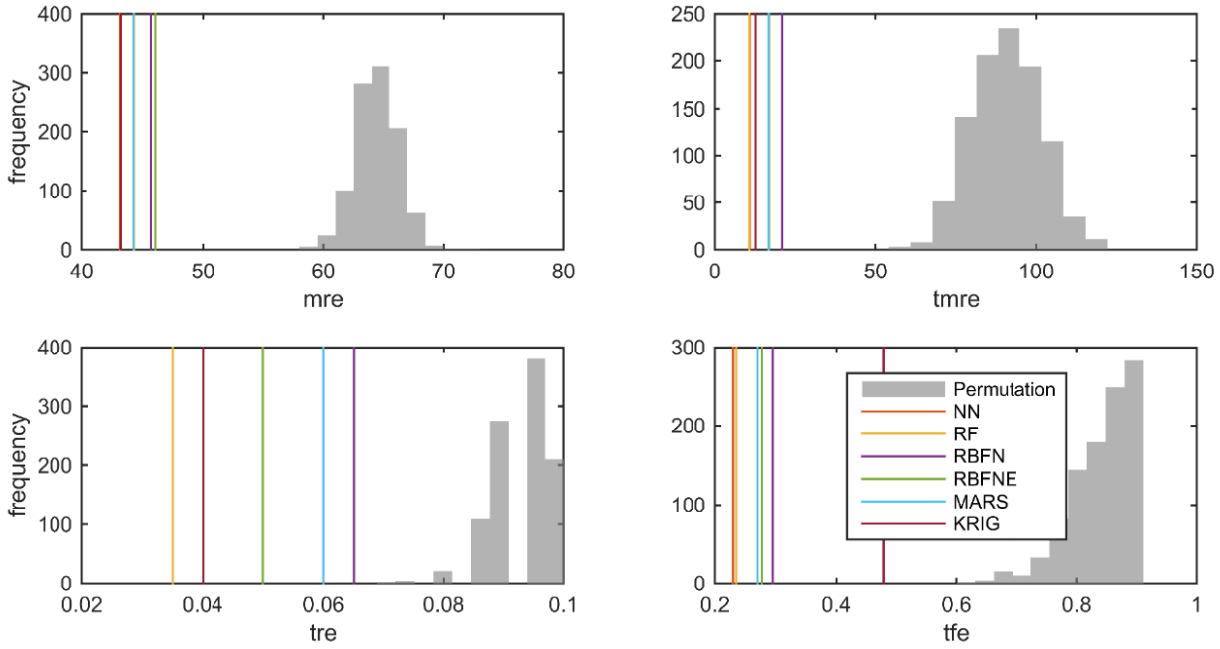


Figure 72: Airport terminal–Boston–Energy Overall- permutation rank test (Test set)

From the permutation test it is observed that the p-values are very close to zero for all the models, since the vertical lines representing the corresponding error of each model do not intersect the histograms. This means that the models have an effect on rank prediction. If one looks at the error values closely for the NN, which performed best in the mean MSE, the TMRE is 10, and from the TRE value it can be calculated that 12 of the top 20 designs were actually predicted inside the top 20. Also, from the TFE it can be calculated that the worst rank prediction of the actual top 20 designs was 66. Those numbers, and especially the TRE value are encouraging so that one can look only at rank predictions of the approximation for a preliminary sorting of the best performing designs.

7.4 Summary table

A summary table with the results of the most important runs from all the case studies examined is included in this section. It includes the Min/Mean/Max validation error for each models as a percentage of the flat model error and the time required for the model construction. Overall NN and KRIG models performed the best, with RF and RBFNE performing the worst. On the other hand, NN and KRIG required substantial time to construct, while RBFNE is constructed almost instantly. When a first rapid approximation is needed then RBFNE could do the job, but a more accurate approach would probably be obtained by NN or KRIG. The takeaway is that all those models and parameters within them are trained and the best one is picked, without any previous intuitive model screening.

				Model Type																							
				NN				RF				RBFN				RBFNE				MARS				KRIG			
				Error			Time	Error			Time	Error			Time	Error			Time	Error			Time	Error			Time
				[% of flat]			[sec]	[% of flat]			[sec]	[% of flat]			[sec]	[% of flat]			[sec]	[% of flat]			[sec]	[% of flat]			[sec]
	Case study	Location	Score	Min	Mean	Max	-	Min	Mean	Max	-	Min	Mean	Max	-	Min	Mean	Max	-	Min	Mean	Max	-	Min	Mean	Max	-
3	Airport terminal	Abu Dhabi	Energy	12	15	21	8.6	23	29	38	21.1	35	46	53	90.0	35	46	55	0.9	20	31	46	35.3	14	20	27	77.0
			Structure	1	2	3	11.7	20	28	32	23.3	1	2	3	23.9	3	6	17	0.9	2	3	3	65.3	1	2	2	81.3
		Abu Dhabi Rotated	Energy	14	16	17	8.6	24	30	41	23.0	38	48	57	80.7	38	49	68	0.9	21	30	39	30.1	13	20	23	67.8
			Structure	1	2	2	10.9	20	28	33	21.9	1	2	3	21.2	3	6	17	0.8	2	2	3	54.8	1	2	2	72.8
		Boston	Energy	10	15	22	11.1	22	27	31	24.4	29	40	46	87.0	33	43	53	1.2	20	23	25	41.9	10	16	20	75.8
			Structure	1	2	2	11.3	20	28	33	25.3	1	2	3	23.1	3	6	17	1.0	2	2	3	59.2	1	2	2	78.6
		Boston Rotated	Energy	11	15	20	9.2	21	28	32	24.9	28	41	46	91.6	30	41	52	0.9	17	23	27	41.9	11	16	20	83.8
			Structure	1	1	2	11.5	20	28	33	24.3	1	2	3	22.6	3	6	17	0.9	2	2	3	62.9	1	2	2	90.3
		Sydney	Energy	1	1	2	11.5	20	28	33	24.3	1	2	3	22.6	3	6	17	0.9	2	2	3	62.9	1	2	2	90.3
			Structure	1	2	2	14.1	20	27	33	24.7	1	2	3	22.9	3	6	17	1.1	2	2	3	59.1	1	2	2	89.9
		Sydney Rotated	Energy	9	13	16	9.6	19	24	35	23.3	29	41	47	74.0	34	57	69	0.8	19	24	30	32.3	9	15	18	65.8
			Structure	1	2	2	10.2	20	27	33	21.6	1	2	3	20.4	3	6	17	0.9	2	2	3	53.2	1	2	2	70.8
2	PI Structure	Boston	Energy	1	3	11	6.7	4	8	10	23.7	1	3	4	5.0	4	9	18	0.8	1	2	4	6.6	1	1	2	2.4
			Structure	0	1	1	4.8	2	7	20	19.5	2	3	7	3.2	2	3	4	0.7	1	2	3	4.1	0	1	1	3.7
		Sydney	Energy	2	6	12	4.0	6	11	15	14.8	2	4	6	2.4	9	16	33	0.5	4	7	18	5.1	2	3	4	2.6
			Structure	0	1	2	6.2	2	7	20	19.0	2	4	6	2.9	2	3	4	0.4	1	1	3	3.1	0	1	1	3.7
1	Bridge	-	Structure	21	29	51	12.9	27	43	63	30.2	18	22	31	80.4	18	25	36	1.2	27	41	64	94.7	13	19	27	195.1

Table 27: Results summary table

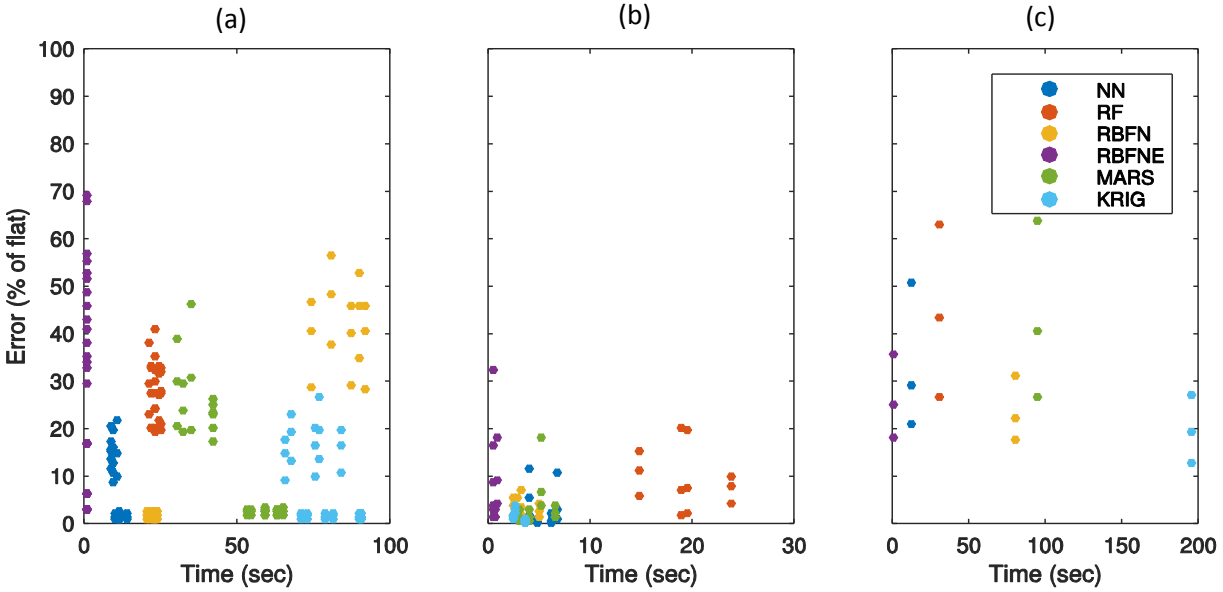


Figure 73: Results summary table visualization (a) Airport terminal (b) PI Structure (c) Bridge

In Figure 73, the results shown in Table 27 are visualized. The results are grouped according to the case study. The model construction time lies in the horizontal axis, with different limits for each case study, because each involved different numbers of samples. The error of the validation set as a percentage of the corresponding "flat" model error is plotted on the vertical axis. The Min, Mean and Max error are all plotted together on the figure, with no distinction to shown the variability of the error. The model construction time was considered the same for each performance metric within each case study, so the Min, Mean and Max errors for the same objective score dataset lie in the same vertical lie, which helps visually distinguish the sets. It should also be noted that the training set sizes for each case study were different; Airport terminal – 600, PI Structure – 120, Grid Truss Bridge – 600.

It can again be observed that the RBFNE models consistently required the least time to build. Also, RBFNE has very good performance for some of the datasets for all the case studies. Especially in the PI Structure and Grid Truss – Bridge case study, RBFNE has performance comparable to that of the other models. For the Airport terminal there are some datasets for which RBFNE did not perform well at all. In general, it can also be observed that KRIG models have comparable performance errors with the NN, but consistently required more time to build, except for PI Structure, where NN and KRIG had comparable build times. This was the case study with the smallest sample set sizes. Therefore it could be concluded that NN is a better choice compared to KRIG for those case studies because of the significant time gains. MARS performs well, but for some datasets it does not (especially for the Airport terminal case study) and required significantly more time than the NN, but not too much less than KRIG. RBFN showed good performance and time alike, except some datasets of the Airport terminal case study for which it showed poor error performance and required substantial training time. In those cases, probably the network could not converge to the training data to the specified accuracy and ended up with high bias, thus the big error. As for RF, it can be seen from the figure that its performance is consistent for all the datasets, unlike any other model. It displays high error variability, but seem to be indifferent of the dataset (within each case study). Also, the time they required for the bigger sample size case studies (Airport terminal and Grid Truss - Bridge) is not forbidding. It should be noted here, that this time could be substantially reduced for the RF construction, because it was observed that for the increasing number of trees from 10 to 300 with a 10-tree step which was examined, the performance did not improve much. The required time to iterate over all those parameters is measured, but since this observation was made, one could reduce the different number of trees examined, subsequently reducing the build time significantly. This effect can be seen in the plots for the RF shown in Appendix A, with similar behavior observed for all the case studies.

Therefore, overall the RBFNE and RF are a choice for a very quick approximation with a higher possibility of poor performance, with NN requiring more time but more possibly giving better performance. The KRIG models show excellent performance error but require substantially more time to construct.

8. Conclusions

The contribution and lessons learned from this thesis are discussed in the final chapter, along with some pointers to potential future research. The surrogate modelling approaches explored and the frameworks developed are put into the greater context.

8.1 Summary of contributions

This thesis explored surrogate modelling techniques with an emphasis on applications to the design of architectural and civil structures. Within this field, this research contributed two new methodologies. The first includes a robust surrogate modelling framework to rapidly build multiple models with different parameters and choose the most suitable for each specific case study application. The machine learning workflow of training/validation/test set partition was used and expanded to include error variability. The developed framework takes a step in simplifying the huge world of different approximation algorithms by allowing easy definition of the parameter exploration by the user. The accuracy vs. construction-speed tradeoff is also acknowledged and examined. The second new methodology, is concerned with assessing the performance of an approximation model by introducing the concept of the "flat" model. The methodology to apply this concept directly to the model's prediction error was explained and tested in multiple case studies. In addition, it was proposed to apply this concept for the quantification of a model's performance on the prediction ranks, using permutation tests. The second methodology is part of a wider theme in this thesis; approximation model error quantification, visualization and comparison. The visualization of the error was found to be of utmost importance and several techniques have been explored and proposed in order to extract the most crucial information from the visualization.

The heart of the thesis is the application of the proposed techniques in case studies from the field of architectural/structural engineering design. A wide range of surrogate models was examined. In terms of model performance, the Neural Network (NN) and Kriging (KRIG) models have been found to perform the best in most of the case study datasets, while the Random Forests (RF) and Radial Basis Function Networks Exact (RBFNE) mostly did not perform well. In terms of the construction time, RBFNE was steadily the best, with KRIG and NN requiring substantial time due to the many different parameter combinations that they involve. It was observed that the approximation for the examined datasets was for the most part lying satisfyingly in the ±10% prediction error range when a specific normalization scheme was applied. This finding is encouraging that the methodologies used could be deployed in accurate large scale design space exploration projects.

8.2 Potential impact

With the industry increasingly gearing towards design solutions that integrate multiple objectives such as structural performance and aesthetics [15], energy efficiency [11], [47] and constructability [48], this

thesis presents frameworks and methodologies that could assist in the rapid and wide exploration of design spaces involving all of those considerations.

The proposed methods could prove powerful in the implementation of this evolving MOO design philosophy in large scale problems and help lead to more functional, better performing and sustainable structures. The results of this thesis are promising in that these methodologies could be viable and realistic to achieve this goal.

8.3 Future work

The next step of this research is to embed the proposed approaches and methodologies into a practical software that will be used by designers from the early-stage of conceptual design. This tool will enable them to more rapidly and efficiently explore the potentials of a certain design concept, its constraints and trade-offs. It would ideally be based on a software that is already used in conceptual design, such as Rhino/Grasshopper, in order to enhance the workflow in a "natural" manner.

Another important direction, would be the creation of a framework based on exploring the parameters of different optimization algorithms, either as a standalone framework or to supplement the developed surrogate modelling framework. Specifically the best obtained approximation model from the surrogate modelling framework could be in succession passed to the optimization framework. Steps in the creation of this framework have already been taken following this research and some preliminary results are shown in Figure 74 and Figure 75 below. The different colors represent different parameters of the optimization algorithms. The figures display the optimal solution that a gradient-based algorithm and a genetic algorithm converged for many runs of different random starting points with the surrogate model as the evaluation function. It is very interesting that the evaluation model and the parameters of the optimization algorithm can have significant effect on the result.

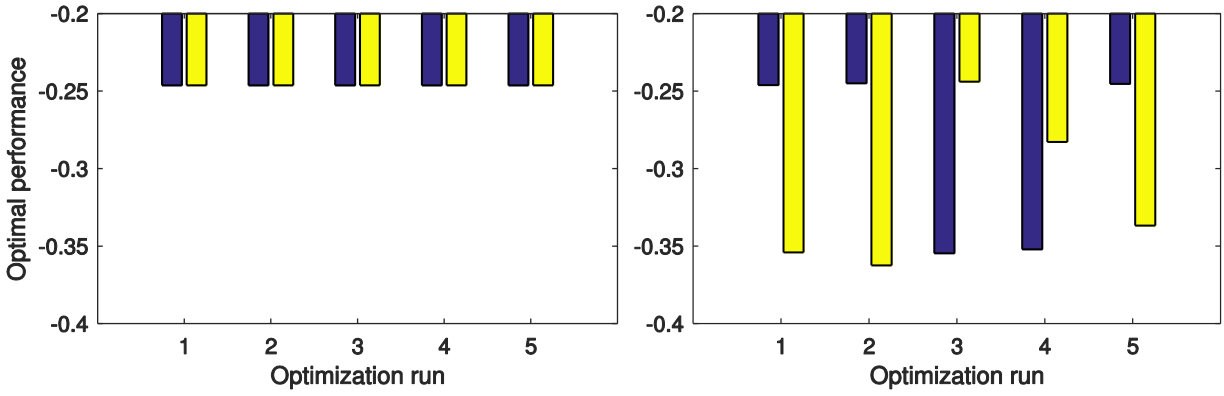


Figure 74: Optimization framework results NN (a) gradient-based algorithm (b) genetic algorithm

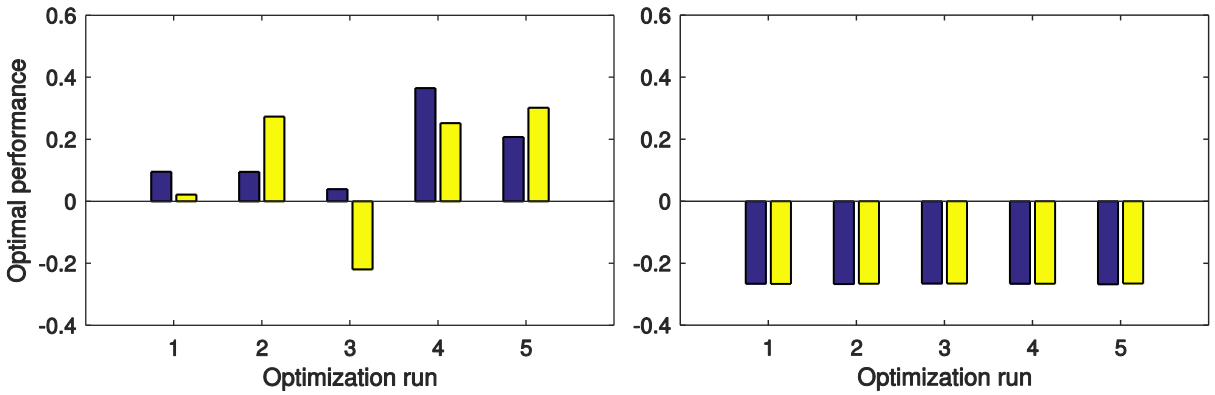


Figure 75: Optimization framework results RF (a) gradient-based algorithm (b) genetic algorithm

8.4 Concluding remarks

The tremendous increase in computational capabilities that occurred over the past decades has enabled solutions to previously unmeetable challenges. However, the increase in capacity brought an increase in problem complexity and demands as well. This thesis was based on this principle; given that capabilities and demand commonly rise with an equal ratio, how designers achieve faster results that are still sufficiently accurate? The methods and approaches of this research allow designers to iterate over solutions faster, thus considering more alternatives and understanding the design parameters in depth, potentially leading to considerable financial and quality-of-life gains.

References

[1] L. L. Stromberg, A. Beghini, W. F. Baker, and G. H. Paulino, "Topology optimization for braced frames: Combining continuum and beam/column elements," *Eng. Struct.*, vol. 37, pp. 106–124, 2012.

[2] P. Amber, "High Line 23 / Neil M. Denari Architects | ArchDaily," 2009. [Online]. Available: http://www.archdaily.com/29582/high-line-23-neil-m-denari-architects. [Accessed: 10-Aug-2015].

[3] W. J. Doherty and A. J. Thadani, "The Economic Value of Rapid Response Time," IBM Corporation, Nov. 1982.

[4] M. Csikszentmihalyi, *Finding Flow: The Psychology of Engagement with Everyday Life*. 1997.

[5] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Kevin Tucker, "Surrogate-based analysis and optimization," *Prog. Aerosp. Sci.*, vol. 41, no. 1, pp. 1–28, 2005.

[6] L. A. J. Schmit and H. Miura, "Approximation Concepts for Efficient Structural Synthesis," 1976.

[7] J. F. M. Barthelemy and R. T. Haftka, "Approximation concepts for optimum structural design - a review," *Struct. Optim.*, vol. 5, no. 3, pp. 129–144, 1993.

[8] P. Hajela and L. Berke, "Neural networks in structural analysis and design: An overview," *Comput. Syst. Eng.*, vol. 3, no. 1–4, pp. 525–538, 1992.

[9] S. Sakata, F. Ashida, and M. Zako, "Structural optimizatiion using Kriging approximation," *Comput. Methods Appl. Mech. Eng.*, vol. 192, no. 7–8, pp. 923–939, 2003.

[10] I. Gidaris, A. A. Taflanidis, and G. P. Mavroeidis, "Kriging metamodeling in seismic risk assessment based on stochastic ground motion," *Earthq. Eng. Struct. Dyn.*, 2015.

[11] L. Magnier and F. Haghighat, "Multiobjective optimization of building design using TRNSYS simulations, genetic algorithm, and Artificial Neural Network," *Build. Environ.*, vol. 45, no. 3, pp. 739–746, 2010.

[12] R. A. Swift and S. M. Batill, "Application of neural networks to preliminary structural design," *Proc. 32nd AIAA/ASME/ASCE/AHS/ASC SDM Meet. Balt. Maryl.*, 1991.

[13] C. T. Mueller, "Computational Exploration of the Structural Design Space," MIT, 2014.

[14] S. Shan and G. G. Wang, "Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions," *Struct. Multidiscip. Optim.*, vol. 41, no. 2, pp. 219–241, 2010.

[15] C. T. Mueller and J. A. Ochsendorf, "Combining Structural Performance and Designer Preferences in Evolutionary Design Space Exploration," *Autom. Constr.*, vol. 52, pp. 70–82, 2015.

[16] X. Shi and W. Yang, "Performance-driven architectural design and optimization technique from a perspective of architects," *Autom. Constr.*, vol. 32, pp. 125–135, 2013.

[17] V. Granadeiro, J. P. Duarte, J. R. Correia, and V. M. S. Leal, "Building envelope shape design in early stages of the design process: Integrating architectural design systems and energy simulation," *Autom. Constr.*, vol. 32, pp. 196–209, 2013.

[18] G. E. P. Box, W. G. Hunter, and J. S. Hunter, *Statistics for Experimenters*. New York, 1978.

[19] J. Sacks, W. J. Welch, J. S. B. Mitchell, and P. W. Henry, "Design and Experiments of Computer Experiments," *Statistical Science*, vol. 4, no. 4. pp. 409–423, 1989.

[20] H. Chung and J. J. Alonso, "Comparison of Approximation Models with Merit Functions for Design Optimization Comparison of Approximation Models with Merit Functions for Design Optimization," *8th AIAA/USAF/NASA/ISSMO Symp. Multidiscip. Anal. Optim.*, 2000.

[21] S. E. Gano, H. Kim, and D. E. B. Ii, "Comparison of Three Surrogate Modeling Techniques: Datascape, Kriging, and Second Order Regression," *11th AIAA/ISSMO Multidiscplinary Anal. Optim. Conf.*, no. September, 2006.

[22] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, pp. 321–355, 1988.

[23] R. L. Hardy, "Multiquadric equations of topography and other irregular surfaces," *Journal of Geophysical Research*. 1971.

[24] J. H. Friedman, "Multivariate Adaptive Regression Splines," *Ann. Stat.*, vol. 19, no. 1, pp. 1–67, 1991.

[25] R. Jin, W. Chen, and T. W. Simpson, "Comparative studies of metamodelling techniques under multiple modelling criteria," *Struct. Multidiscip. Optim.*, vol. 23, no. 1, pp. 1–13, 2001.

[26] D. Gorissen, I. Couckuyt, P. Demeester, T. Dhaene, and K. Crombecq, "A Surrogate Modeling and Adaptive Sampling Toolbox for Computer Based Design," *J. Mach. Learn. Res.*, vol. 11, pp. 2051–2055, 2010.

[27] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed., vol. 1. Springer, 2009.

[28] F. a C. Viana, R. T. Haftka, and V. Steffen, "Multiple surrogates: How cross-validation errors can help us to obtain the best predictor," *Struct. Multidiscip. Optim.*, vol. 39, no. 4, pp. 439–457, 2009.

[29] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[30] "Feedforward neural network - MATLAB feedforwardnet." [Online]. Available: http://www.mathworks.com/help/nnet/ref/feedforwardnet.html. [Accessed: 01-Jul-2015].

[31] L. Breiman, "Random Forests," *Eur. J. Math.*, vol. 45, no. 1, pp. 5–32, 2001.

[32] "Bootstrap aggregation for ensemble of decision trees - MATLAB." [Online]. Available: http://www.mathworks.com/help/stats/treebagger-class.html. [Accessed: 01-Jul-2015].

[33] "Design radial basis network - MATLAB newrb." [Online]. Available: http://www.mathworks.com/help/nnet/ref/newrb.html. [Accessed: 01-Jul-2015].

[34] "Radial Basis Function Network (RBFN) Tutorial | Chris McCormick on WordPress.com." [Online]. Available: https://chrisjmccormick.wordpress.com/2013/08/15/radial-basis-function-network-rbfn-tutorial/. [Accessed: 09-Jul-2015].

[35] "Design exact radial basis network - MATLAB newrbe." [Online]. Available: http://www.mathworks.com/help/nnet/ref/newrbe.html. [Accessed: 01-Jul-2015].

[36] G. Jekabsons, "ARESLab: Adaptive Regression Splines toolbox for Matlab/Octave," 2015. [Online]. Available: http://www.cs.rtu.lv/jekabsons/. [Accessed: 10-Jul-2015].

[37] D. G. Krige, "A statistical approach to some basic mine valuation problems on the Witwatersrand," *J. Chem. Metall. Min. Eng. Soc. South Africa*, vol. 52, no. 6, pp. 119–139, 1951.

[38] S. N. Lophaven, J. Søndergaard, and H. B. Nielsen, "DACE-A MATLAB Kriging Toolbox," Lyngby, Denmark, 2002.

[39] "Rhinoceros 3D." [Online]. Available: https://www.rhino3d.com/. [Accessed: 13-Aug-2015].

[40] "Grasshopper." [Online]. Available: http://www.grasshopper3d.com/. [Accessed: 13-Aug-2015].

[41] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[42] "GhPython | Grasshopper." [Online]. Available: http://www.food4rhino.com/project/ghpython?ufh. [Accessed: 15-Jul-2015].

[43] "Hoopsnake | Grasshopper." [Online]. Available: http://www.food4rhino.com/project/hoopsnake?ufh. [Accessed: 15-Jul-2015].

[44] "Karamba 3d." [Online]. Available: http://www.karamba3d.com/. [Accessed: 24-Jul-2015].

[45] "ARCHSIM." [Online]. Available: http://archsim.com/. [Accessed: 24-Jul-2015].

[46] "EnergyPlus Energy Simulation Software - US Department of Energy." [Online]. Available: http://apps1.eere.energy.gov/buildings/energyplus/. [Accessed: 24-Jul-2015].

[47] N. Brown, S. Tseranidis, and C. T. Mueller, "Multi-objective optimization for diversity and performance in conceptual structural design," in *International Association for Shell and Spatial Structures (IASS)*, 2015.

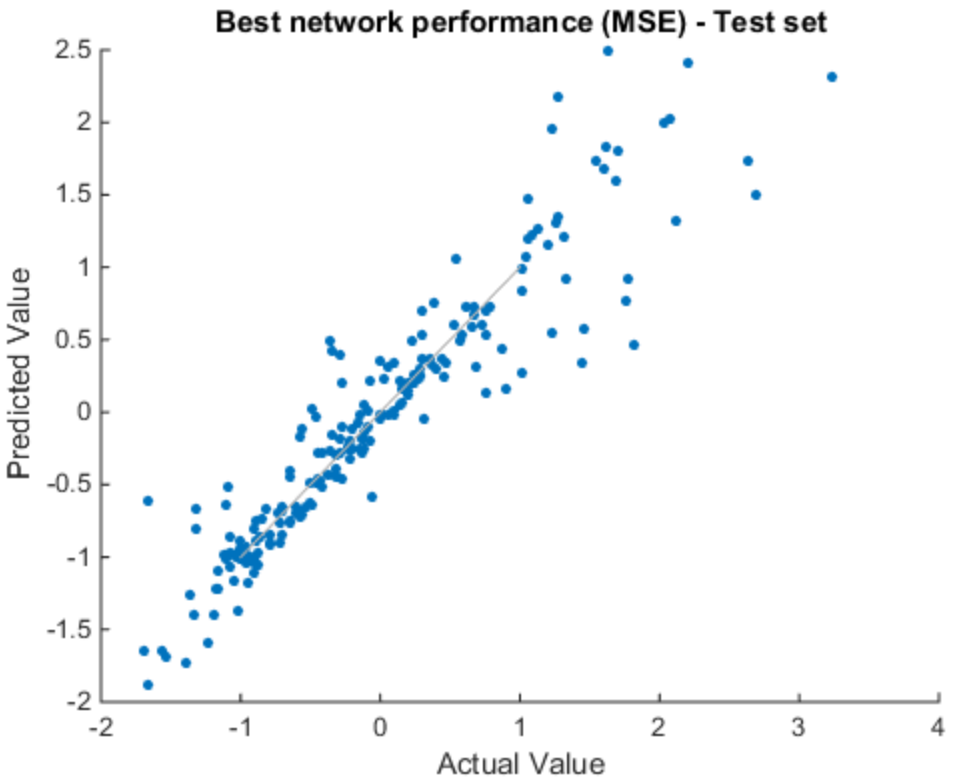
[48] A. Horn, "Integrating Constructability into Conceptual Structural Design and Optimization," Massachusetts Institute of Technology, 2015.

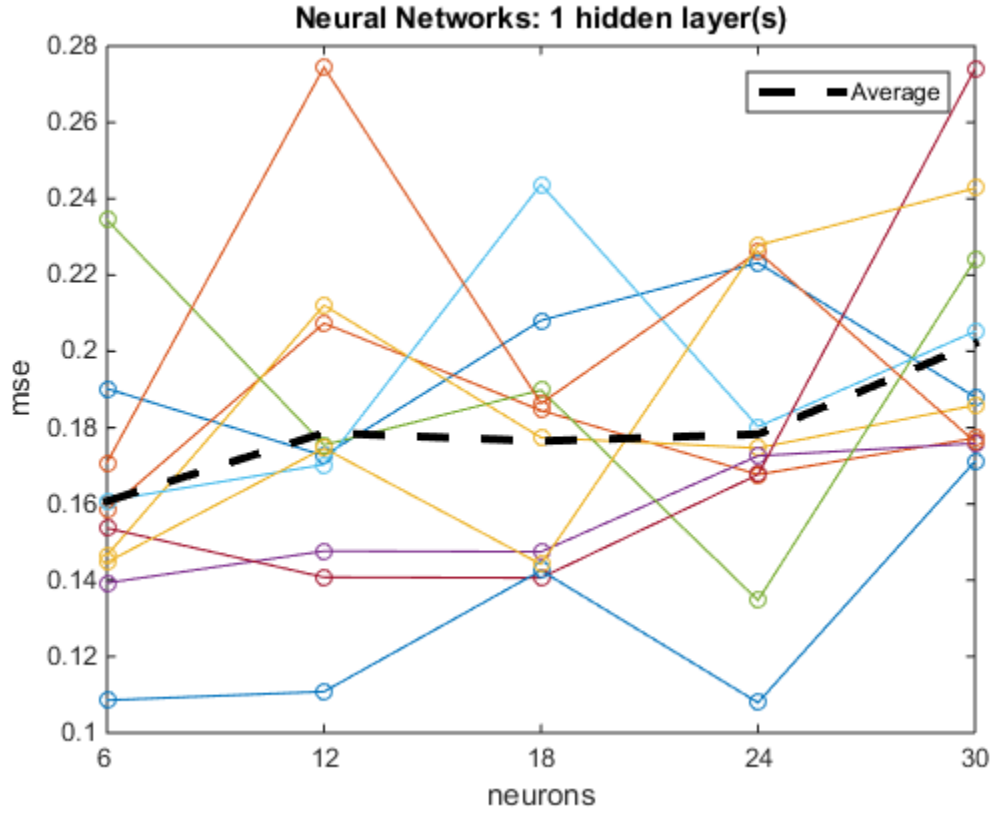
Appendix A: Airport terminal | Energy Overall | Boston runs

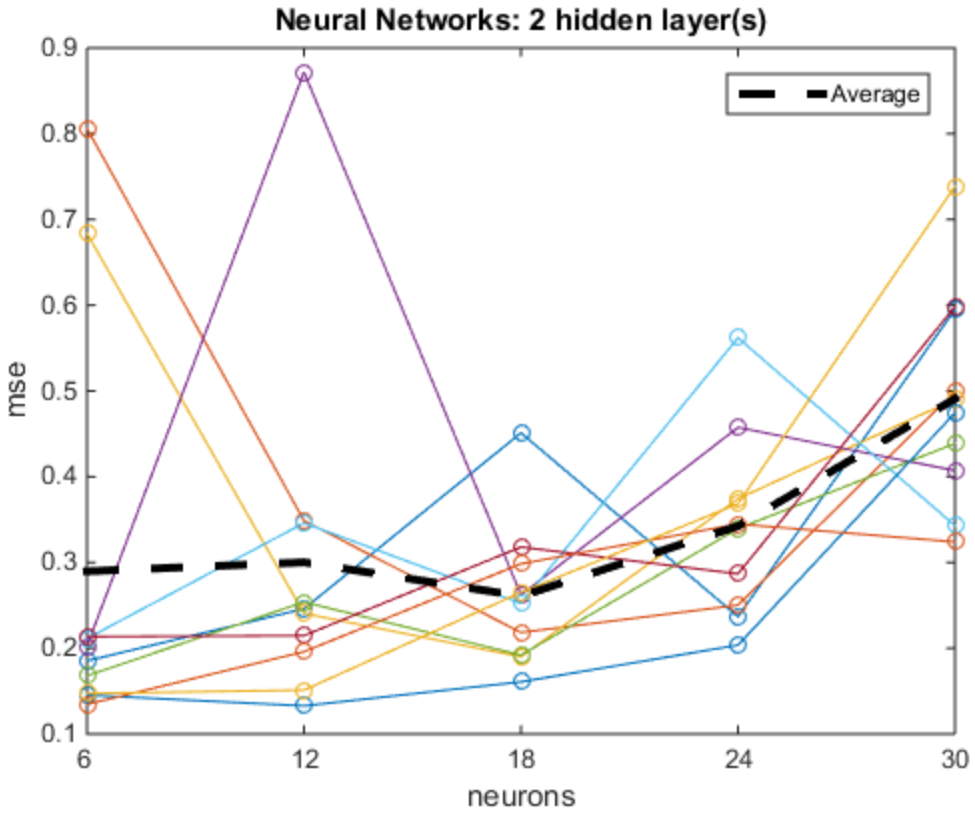
NN – Neural Networks

Parameters

```
clear;clc;close all

dataFile = 'train_val_scaled';          % name of file containing train/validation data
runs = 10;                              % number of training/validation runs
dataFileTest = 'test_set_scaled';       % name of file containing test data

network_type = 'backpropagation'; % type of neural network to train

nneuronsO = 6:6:30;     % number of neurons for every layer to try
nlayersO = 1:2;         % number of hidden layers to try

trainRatio = 0.8;       % training set ratio inside neural network training
valRatio   = 0.2;       % validation set ratio ''   ''     ''      ''
testRatio  = 0;         % test set ratio       ''   ''     ''      ''

trainFcn = 'trainlm';   % training function of neural network

max_fail = 10;          % max number of validation checks to end training

n = 20;                % n top-performing designs

save_ext = 'NNenergy';  % extension of save directory
plot_on = 1;            % flag to make and save final plots

NNtrainModels( dataFile, runs, dataFileTest, network_type, nneuronsO, nlayersO, trainRatio,
valRatio, testRatio, trainFcn, max_fail, n, save_ext, plot_on );
```


Best network performance (MSE) - Test set
Predicted Value
Actual Value
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2
3
4


Neural Networks: 1 hidden layer(s)
Average
mse
neurons
0.28
0.26
0.24
0.22
0.2
0.18
0.16
0.14
0.12
0.1
6
12
18
24
30

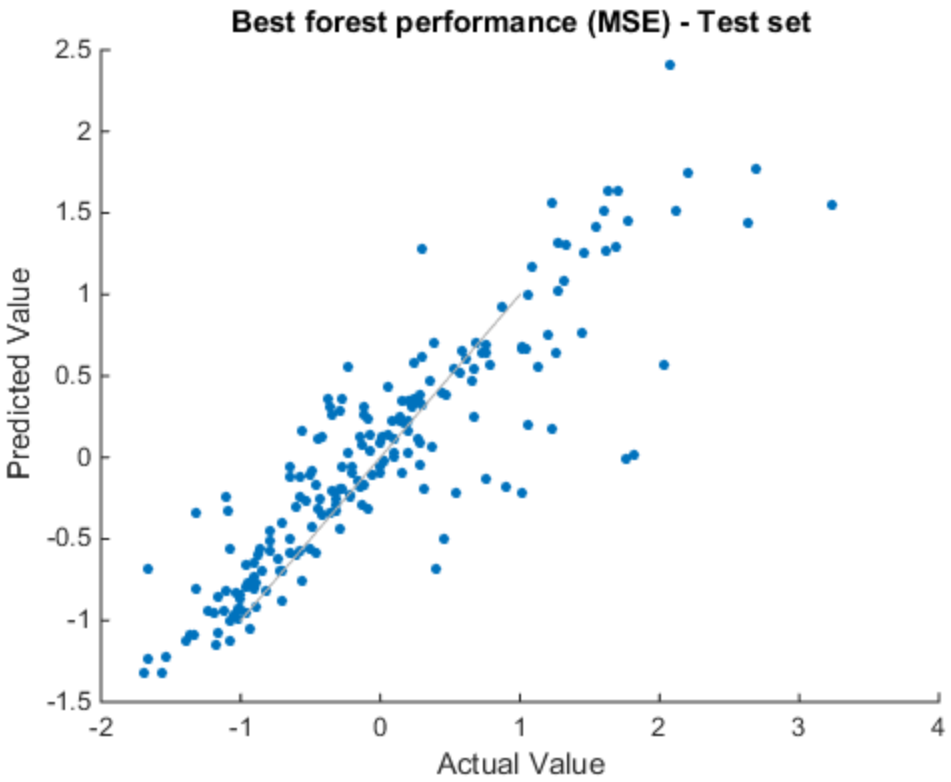


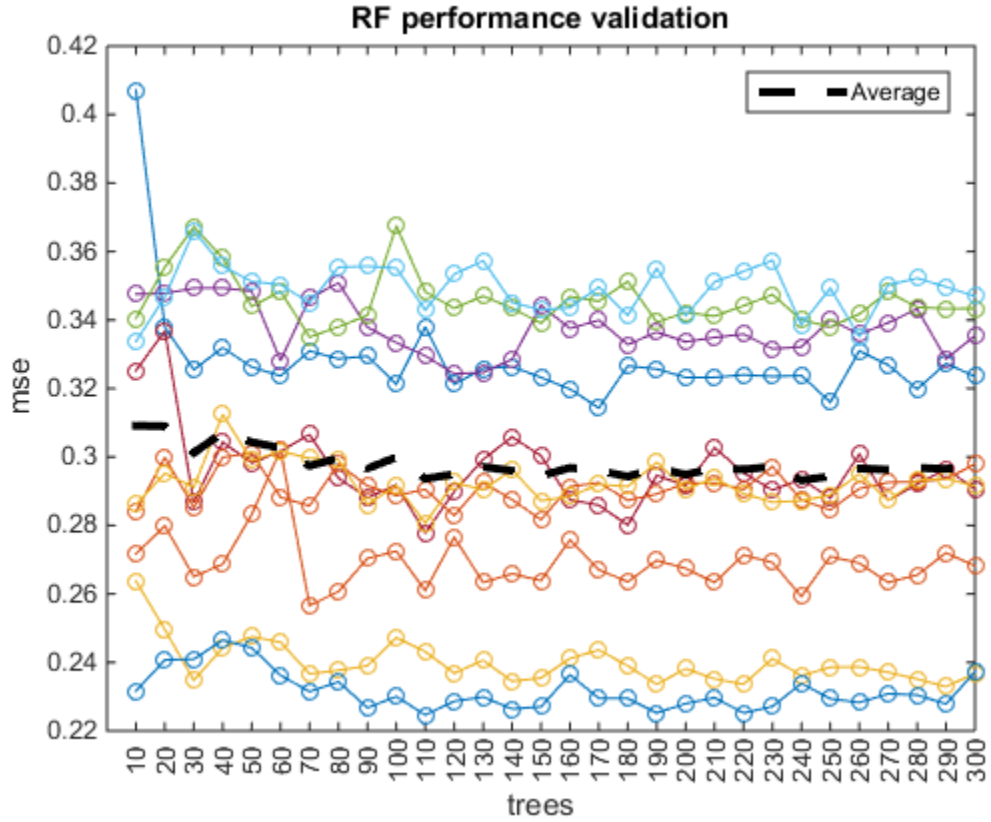
Published with MATLAB® R2014b

RF – Random Forests

Parameters

```
clear;clc;close all

dataFile = 'train_val_scaled';          % name of file containing train/validation data
runs = 10;                              % number of training/validation runs
dataFileTest = 'test_set_scaled';       % name of file containing test data

cart_type = 'random_forest'; % type of CART model to train

ntreesO = 10:10:300;        % number of trees
fboot  = 0.8;                 % fraction of input data to sample with replacement
replacement = 'on';         % sample with replacement: 'on' or 'off'
method = 'regression';      % either 'classification' or 'regression'
nvartosample = 6;   % number of variables to select at each decision split
minleaf = 5;        % minimum number of observations per leaf

n = 20;                     % n top-performing designs

save_ext = 'RFenergy';          % extension of save directory
plot_on = 1;            % flag to make and save final plots

RFtrainModels( dataFile, runs, dataFileTest, cart_type, ntreesO, fboot, replacement, method,
nvartosample, minleaf, n, save_ext, plot_on );
```



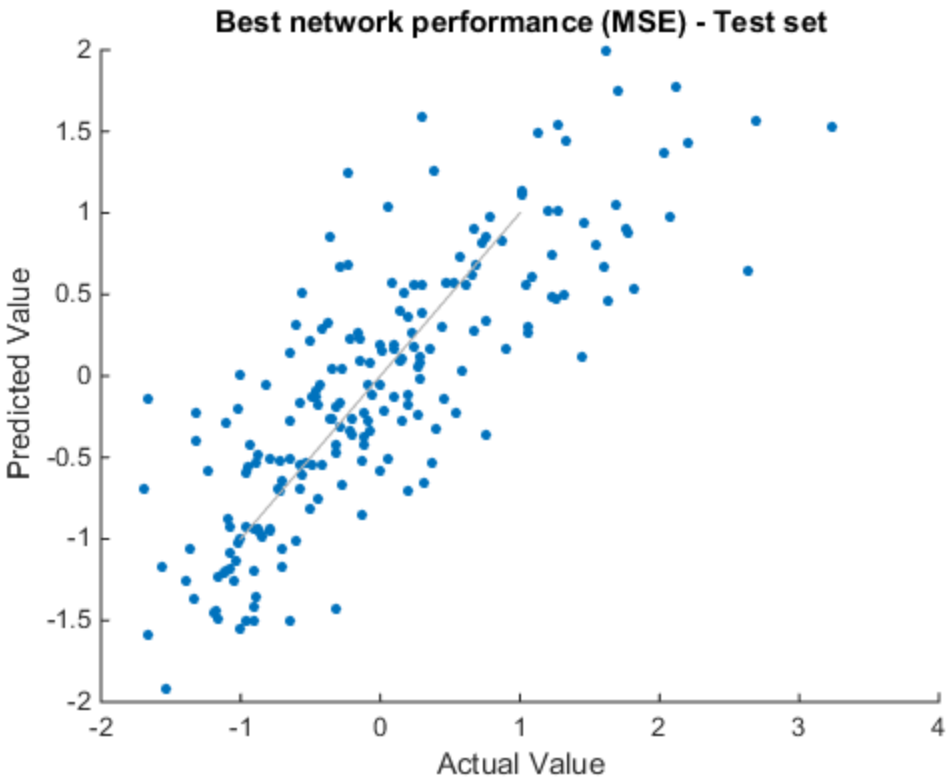


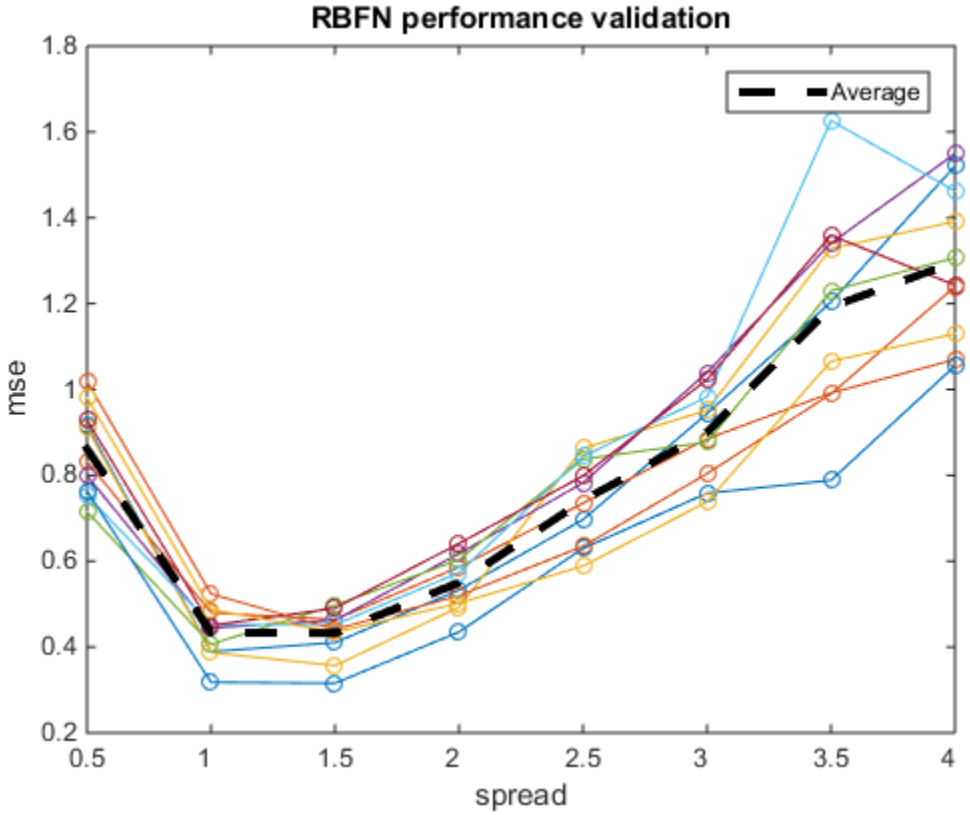
Published with MATLAB® R2014b

RBFN – Radial Basis Function Networks

Parameters

```
clear;clc;close all

dataFile = 'train_val_scaled';          % name of file containing train/validation data
runs = 10;                              % number of training/validation runs
dataFileTest = 'test_set_scaled';       % name of file containing test data

network_type = 'radial_basis'; % type of neural network to train

mse_goal0 = 5.^-(3);       % mean squared error goal (for newrb network)
% mse_goal0 = 0;                % for newrbe network

spread0 = 0.5:0.5:4;             % spread parameter for newrb and newrbe

max_neurons = 600;       % max number of neurons (for newrb) - NaN if default

n = 20;                 % n top-performing designs

save_ext = 'RBFNenergy';          % extension of save directory
plot_on = 1;            % flag to make and save final plots

RBFNtrainModels( dataFile, runs, dataFileTest, network_type, mse_goal0, spread0, max_neurons, n,
save_ext, plot_on );
```



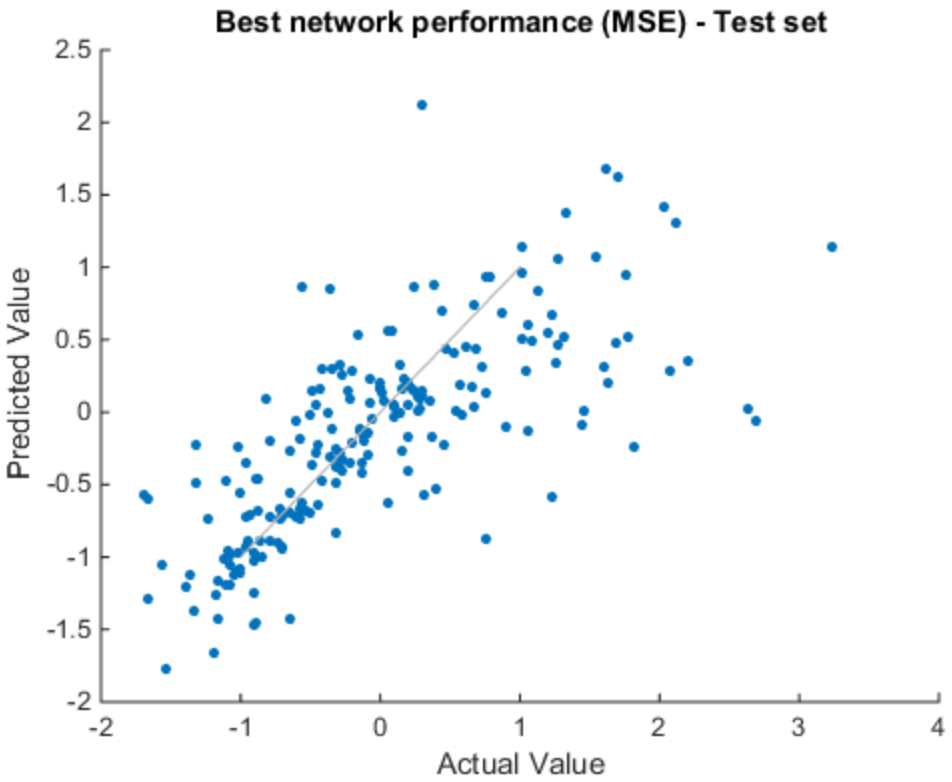


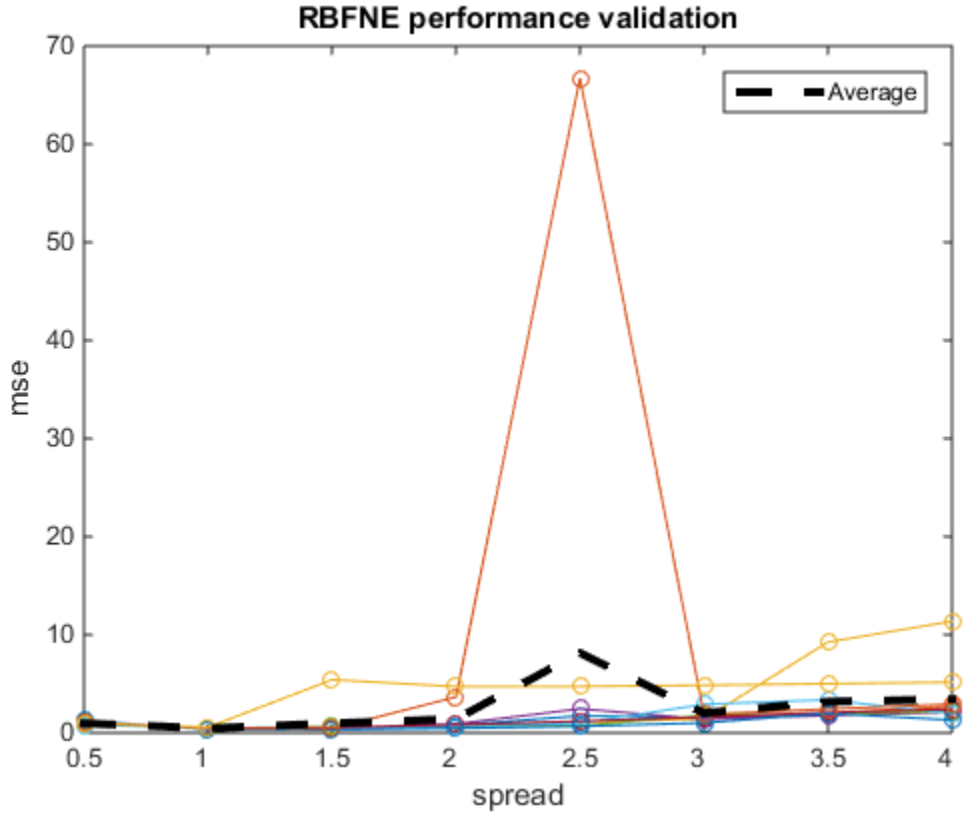
Published with MATLAB® R2014b

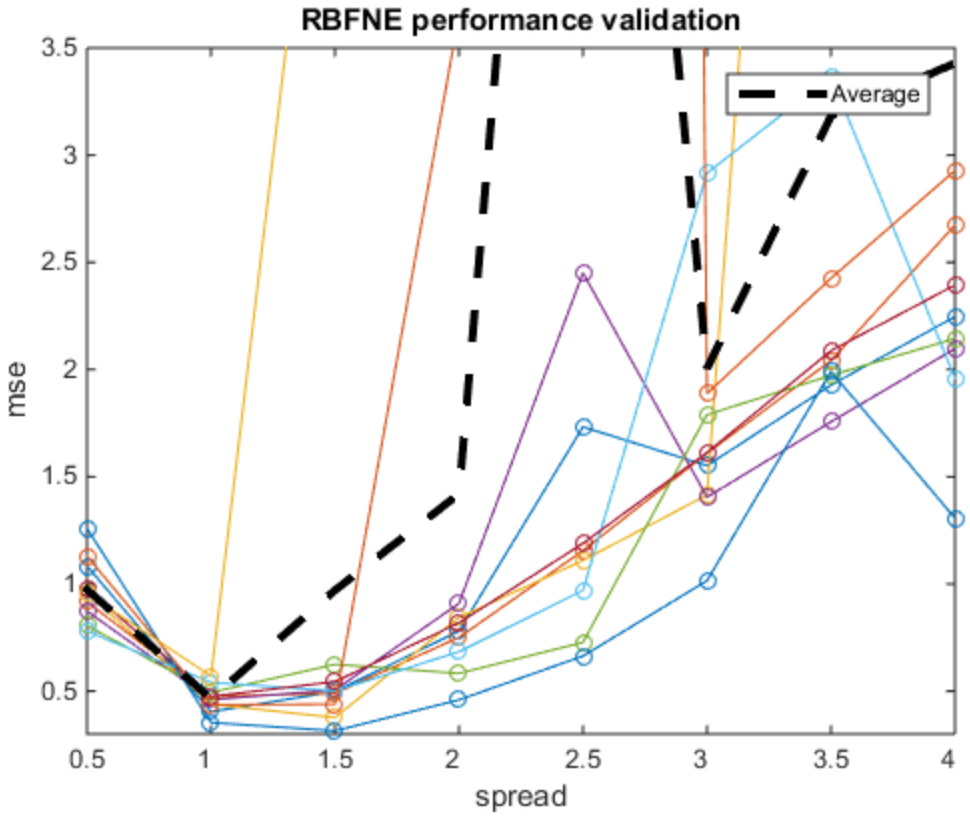
RBFNE – Radial Basis Function Networks Exact

Parameters

```
clear;clc;close all

dataFile = 'train_val_scaled';          % name of file containing train/validation data
runs = 10;                              % number of training/validation runs
dataFileTest = 'test_set_scaled';       % name of file containing test data

network_type = 'radial_basis_exact'; % type of neural network to train

% mse_goal0 = 5.^-(3);       % mean squared error goal (for newrb network)
mse_goal0 = 0;                % for newrbe network

spread0 = 0.5:0.5:4;             % spread parameter for newrb and newrbe

max_neurons = NaN;       % max number of neurons (for newrb) - NaN if default

n = 20;                 % n top-performing designs

save_ext = 'RBFNEenergy';          % extension of save directory
plot_on = 1;            % flag to make and save final plots

RBFNtrainModels( dataFile, runs, dataFileTest, network_type, mse_goal0, spread0, max_neurons, n,
save_ext, plot_on );
```


Best network performance (MSE) - Test set
Predicted Value
Actual Value
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2
3
4


RBFNE performance validation
Average
mse
spread
70
60
50
40
30
20
10
0
0.5
1
1.5
2
2.5
3
3.5
4



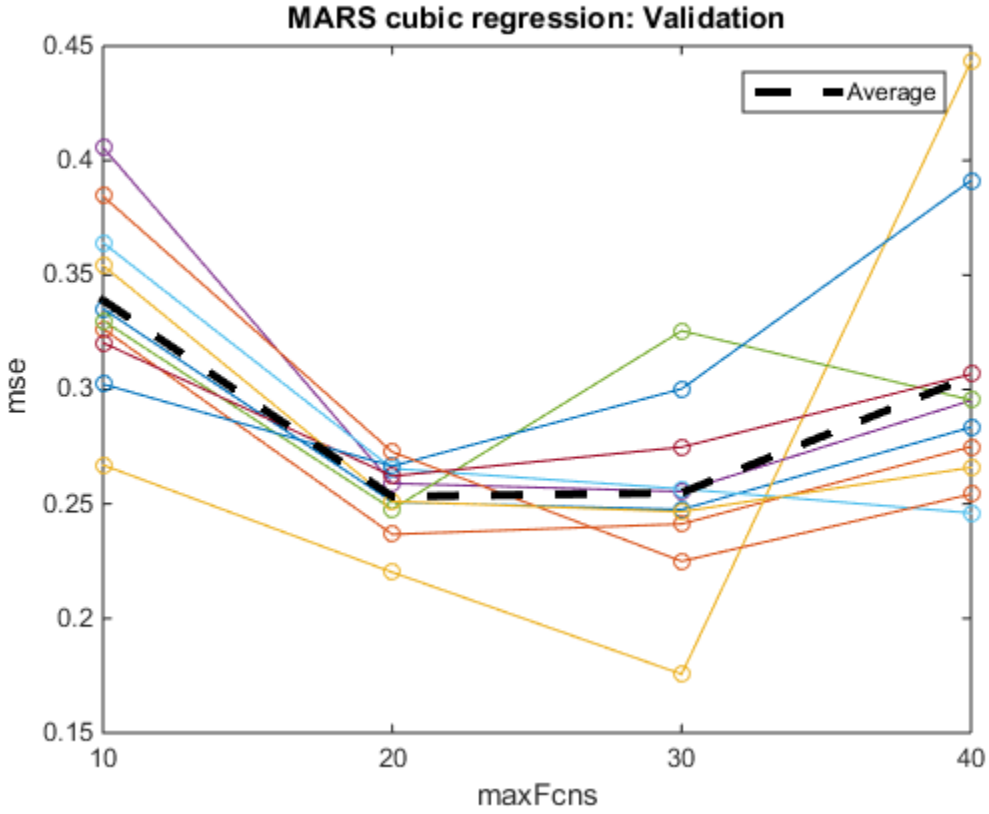
Published with MATLAB® R2014b

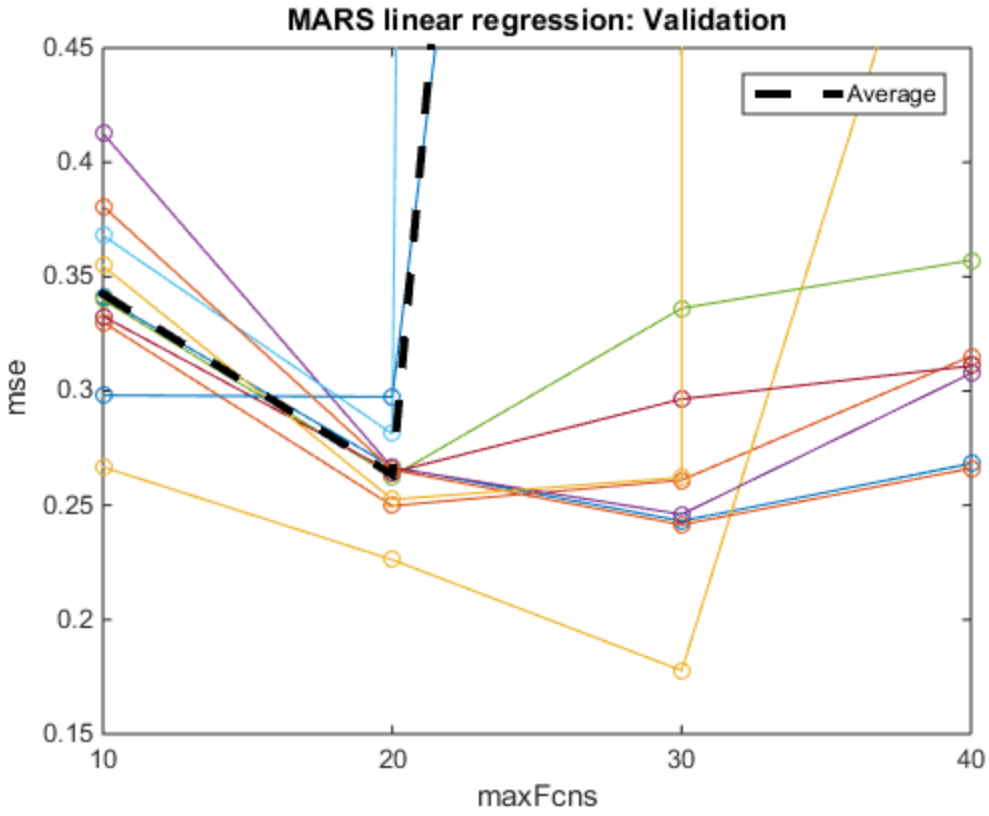
MARS – Multivariate Adaptive Regression Splines

Parameters

```
clear;clc;close all

dataFile = 'train_val_scaled';          % name of file containing train/validation data
runs = 10;                              % number of training/validation runs
dataFileTest = 'test_set_scaled';       % name of file containing test data

model_type = 'piecewise regression'; % type of model to train

maxFcnsO = 10:10:40;       % maximum number of basis functions to use
c = 3;                      % Generalized Cross-Validation (GCV) penalty per knot
cubicO = [true false];      % true: piecewise cubic, false: piecewise linear

selfInteraction = 1;        % default: 1 means no self interactions
maxInteractions = 6;        % recommended to use # of features * selfInteractions

threshold = 1e-3;           % threshold for algorithm termination

n = 20;                     % n top-performing designs

save_ext = 'MARSenergy';          % extension of save directory
plot_on = 1;            % flag to make and save final plots

MARStrainModels( dataFile, runs, dataFileTest, model_type, maxFcnsO, c, cubicO, selfInteraction,
maxInteractions, threshold, n, save_ext, plot_on );
```


Best model performance (MSE) - Test set
Predicted Value
Actual Value
5
4
3
2
1
0
-1
-2
-2
-1
0
1
2
3
4


MARS cubic regression: Validation
Average
mse
maxFcns
0.45
0.4
0.35
0.3
0.25
0.2
0.15
10
20
30
40



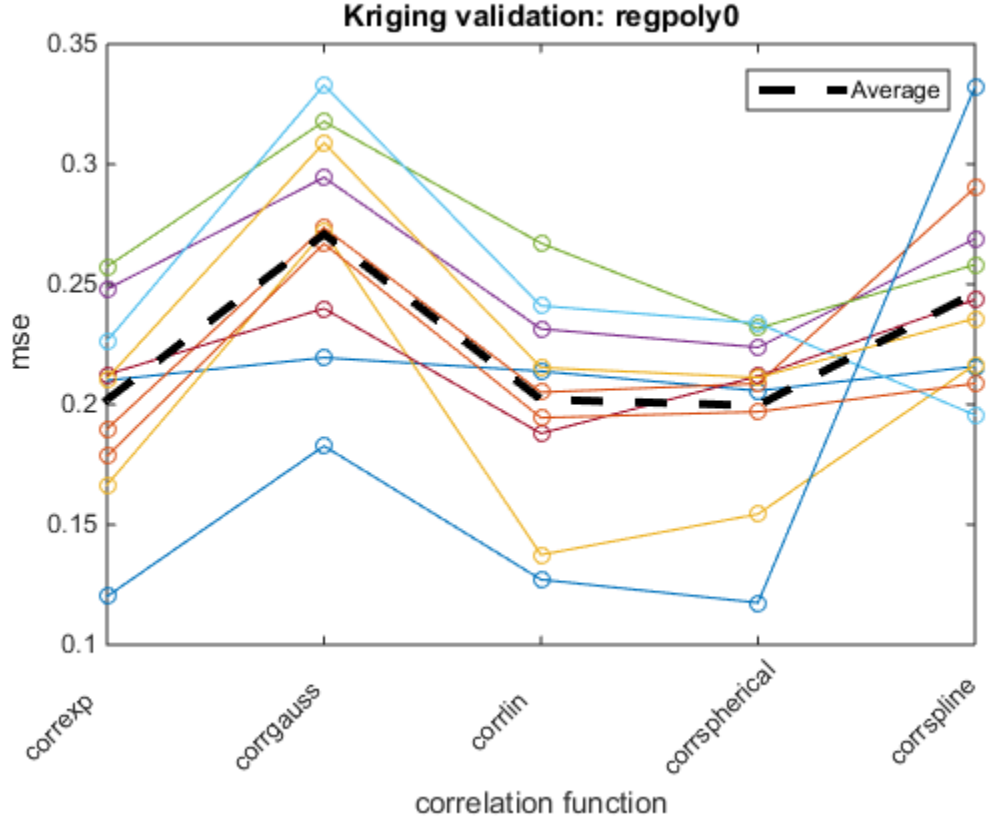
Published with MATLAB® R2014b

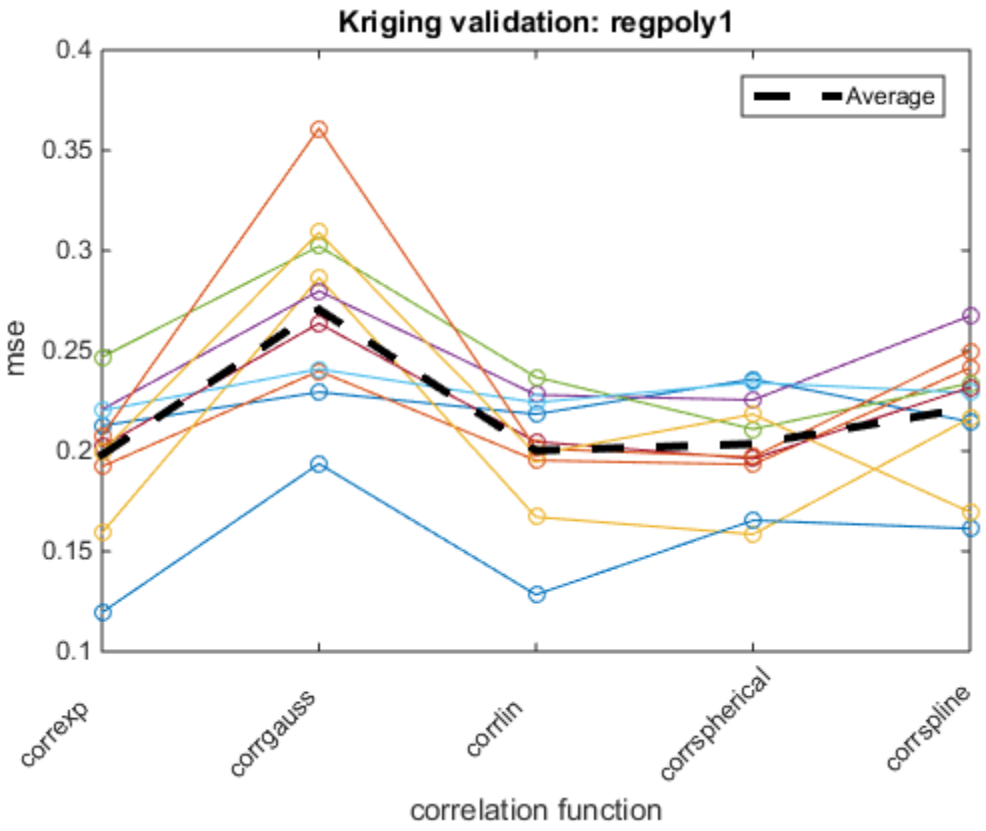
KRIG – Kriging Regression

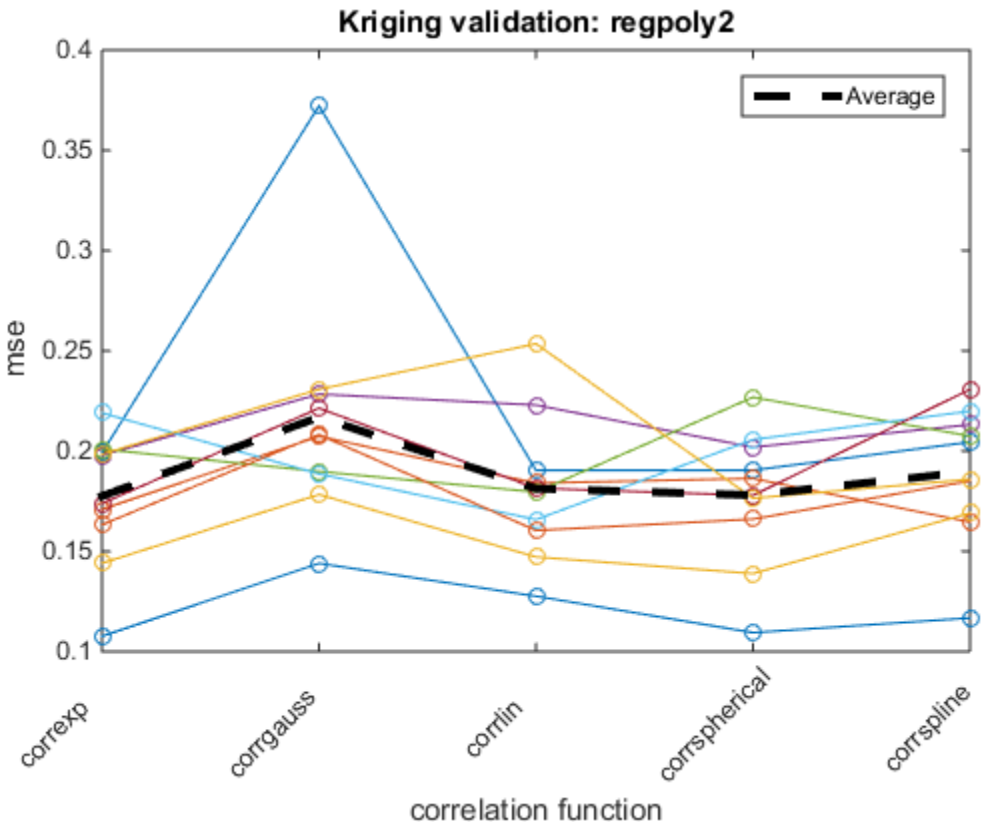
Parameters

```
clear;clc;close all

dataFile = 'train_val_scaled';          % name of file containing train/validation data
runs = 10;                              % number of training/validation runs
dataFileTest = 'test_set_scaled';       % name of file containing test data

model_type = 'kriging_regression';      % type of model to train

lob = 1e-4;                             % lower theta boundary
upb = 2e2;                              % upper theta boundary

regr0 = {@regpoly0, ...  % regression functions to try
         @regpoly1, ...
         @regpoly2};

corr0 = {@correxp, ...   % correlation functions to try
         @corrgauss, ... %@correxpg, ... omitted
         @corrlin, ...
         @corrspherical, ...
         @corrspline};

n = 20;                     % n top-performing designs

save_ext = 'KRIGenergy';          % extension of save directory
plot_on = 1;            % flag to make and save final plots

KRIGtrainModels( dataFile, runs, dataFileTest, model_type, regr0, corr0, lob, upb, n, save_ext,
plot_on );
```


Best model performance (MSE) - Test set
Predicted Value
Actual Value
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2
3
4


Kriging validation: regpoly0
Average
mse
correlation function
0.35
0.3
0.25
0.2
0.15
0.1
correxp
corrgauss
corrlin
corrspherical
corrspline

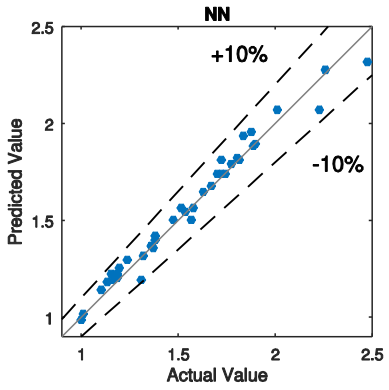


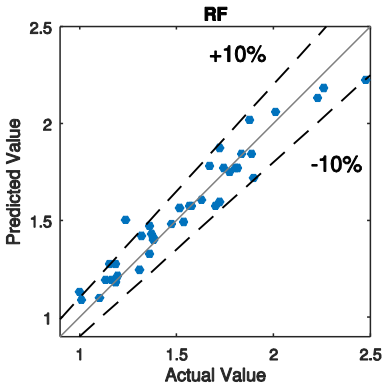


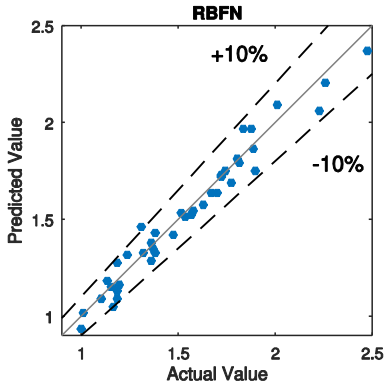
Published with MATLAB® R2014b

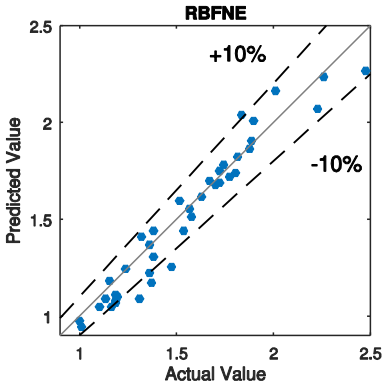
Appendix B: Case study scatter plots

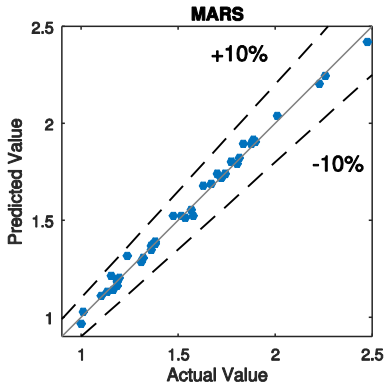
Energy Overall __ Boston __ PI Structure

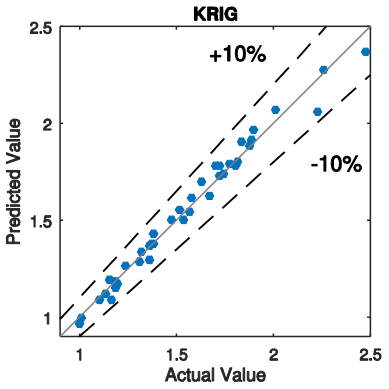




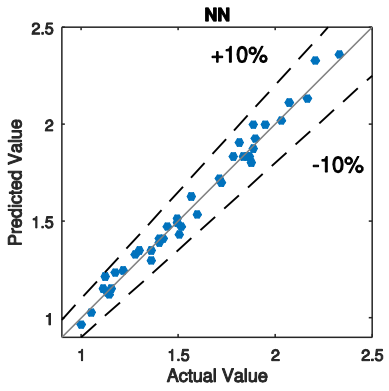


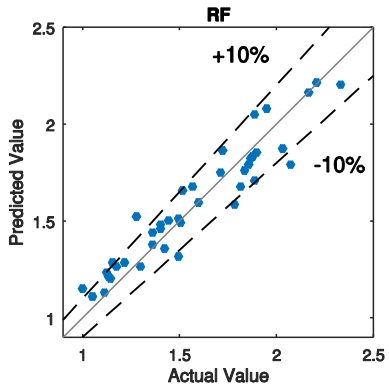


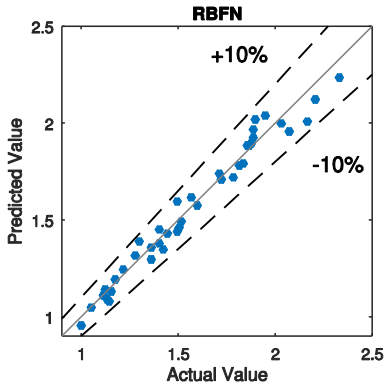


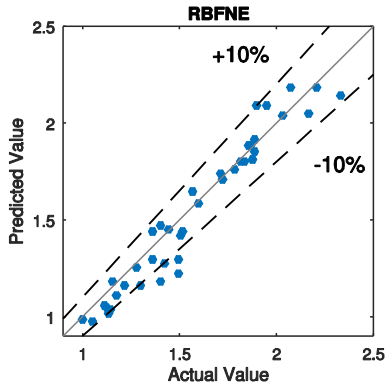


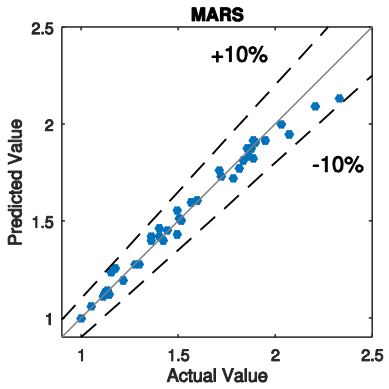
Heating+Cooling+Lighting __ Boston __ PI Structure

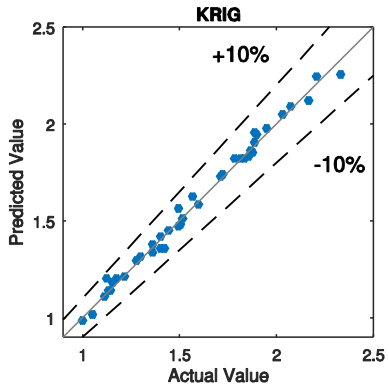




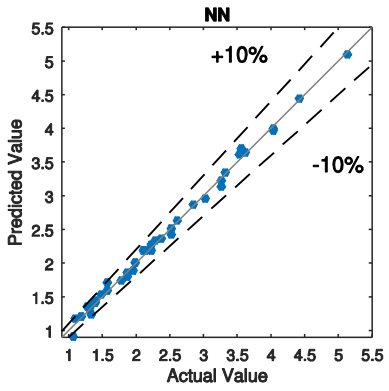


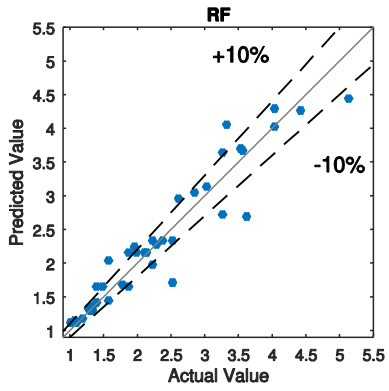


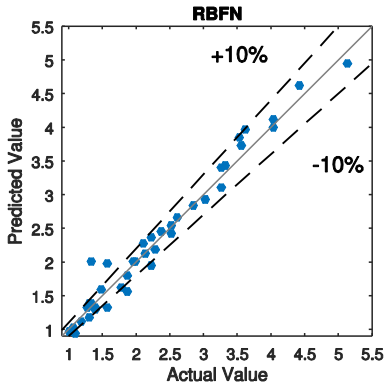


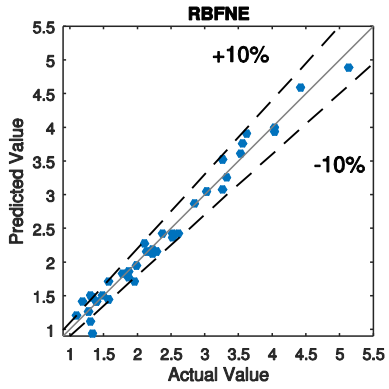


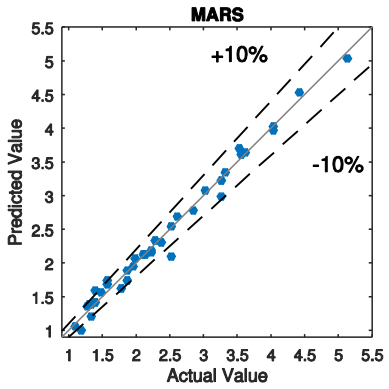
Structure __ Boston __ PI Structure

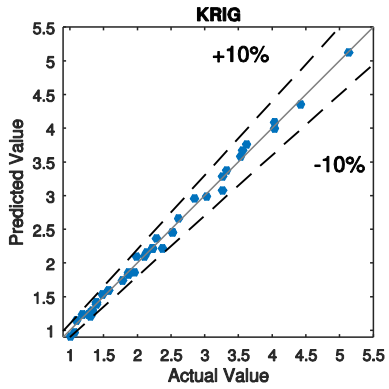




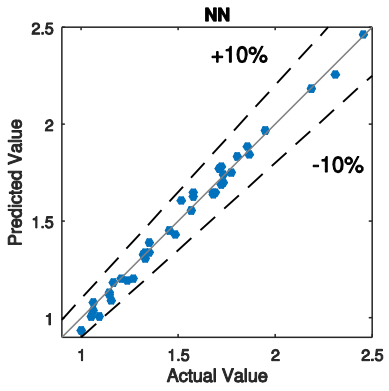


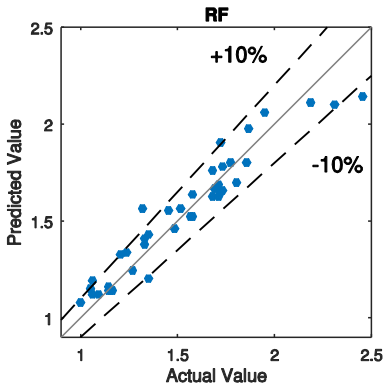


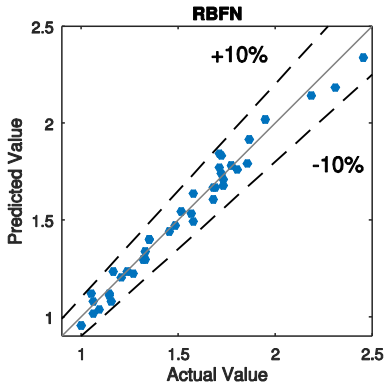


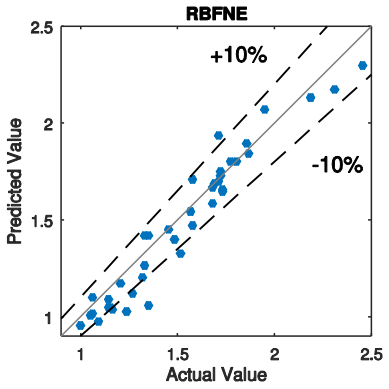


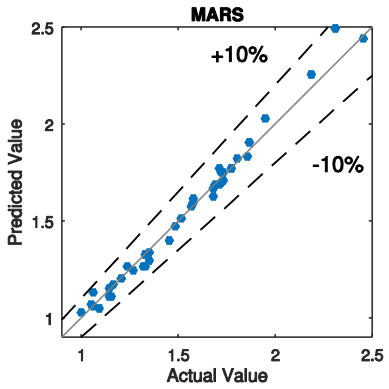
Cooling __ Boston __ PI Structure

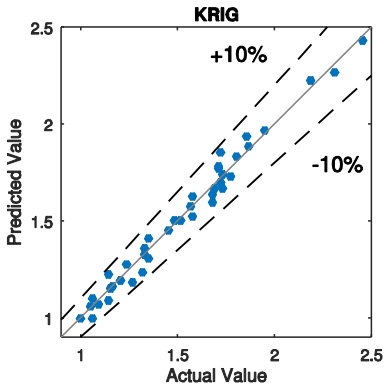




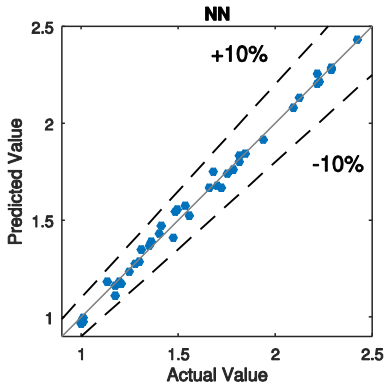


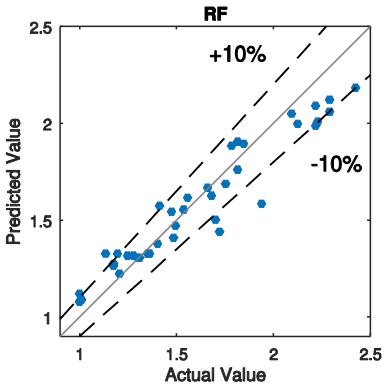


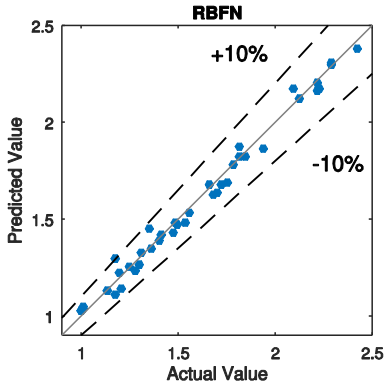


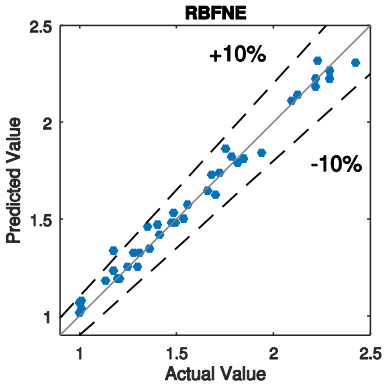


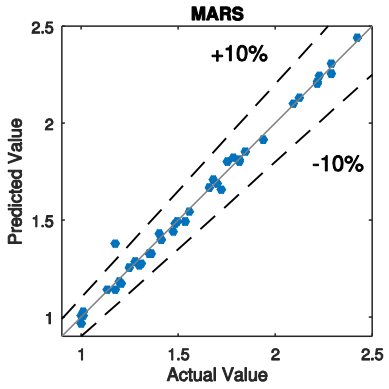
Heating __ Boston __ PI Structure

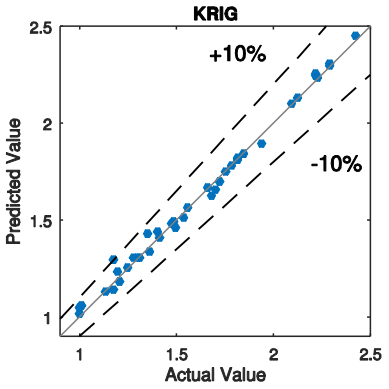




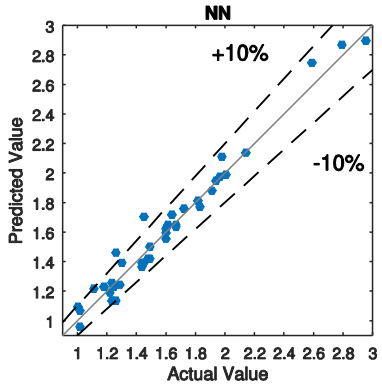


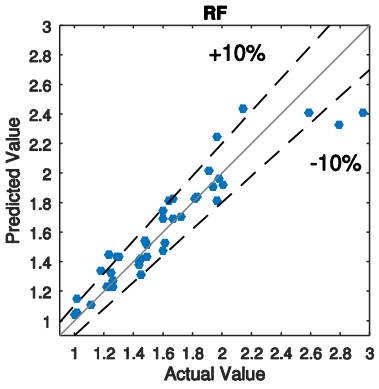


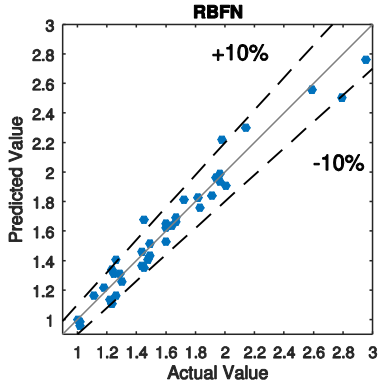


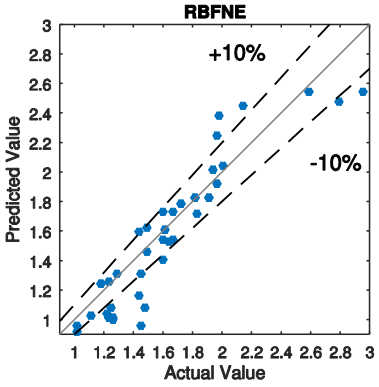


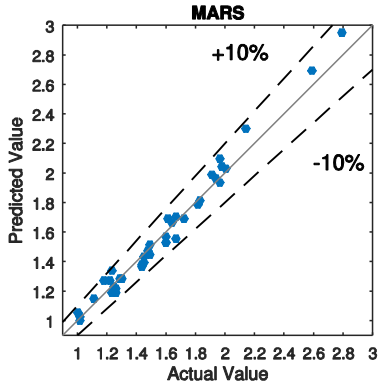
Lighting __ Boston __ PI Structure

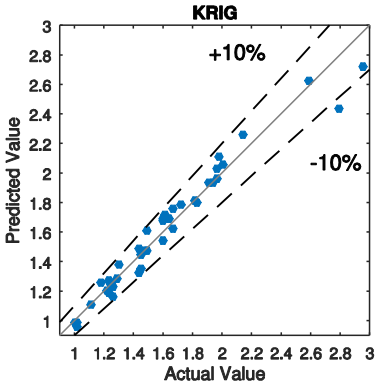




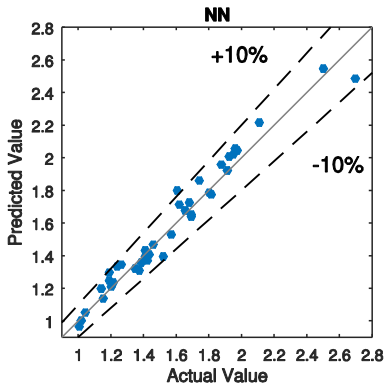


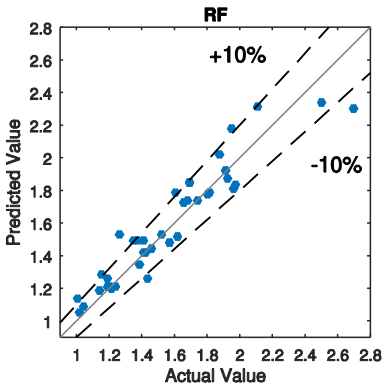


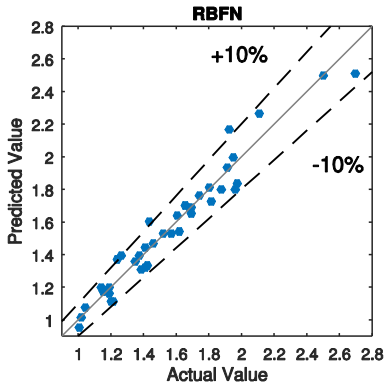


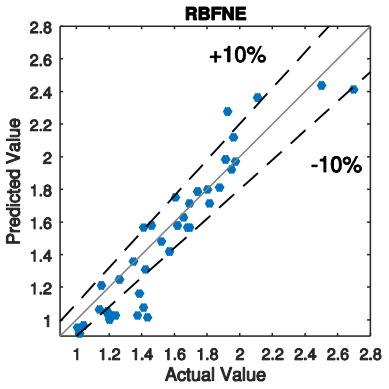


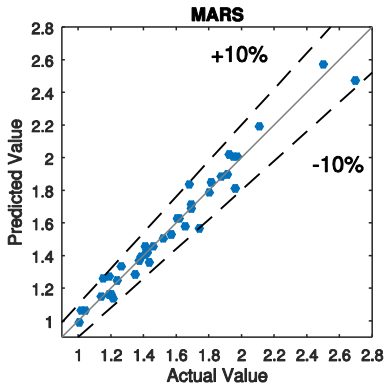
Energy Overall __ Sydney __ PI Structure

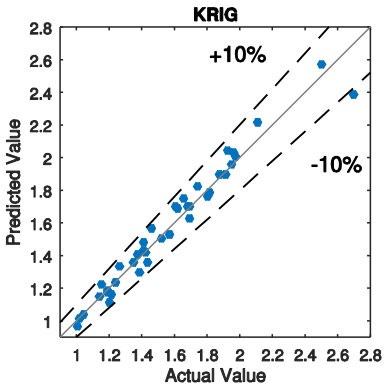




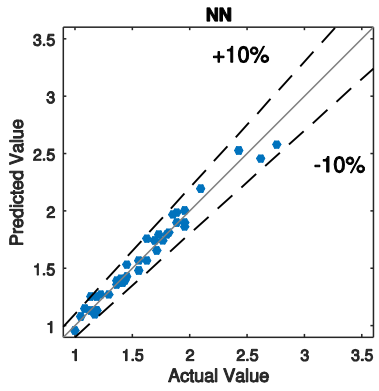


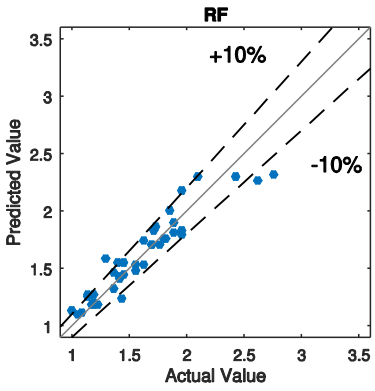


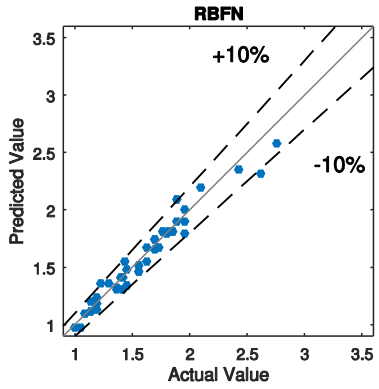


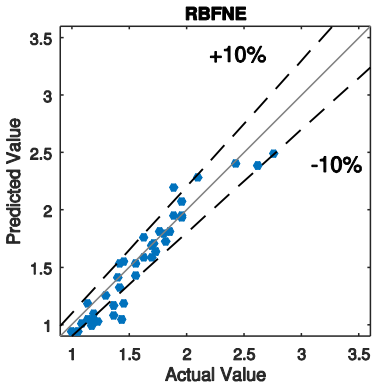


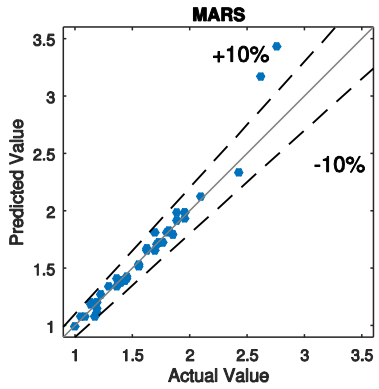
Heating+Cooling+Lighting __ Sydney __ PI Structure

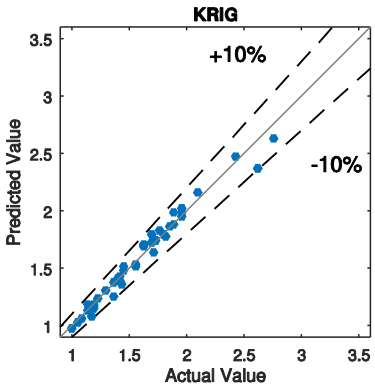




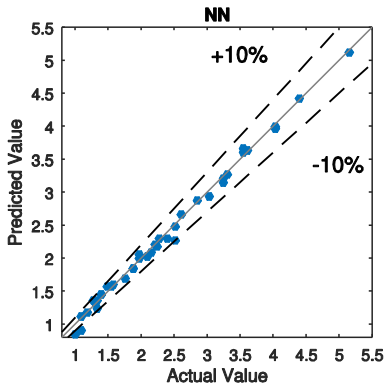


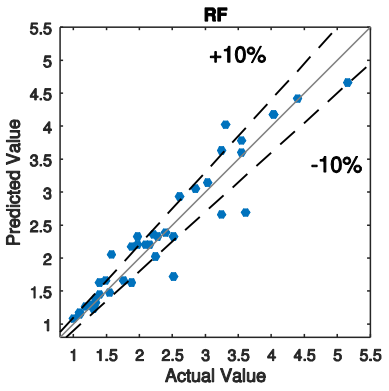


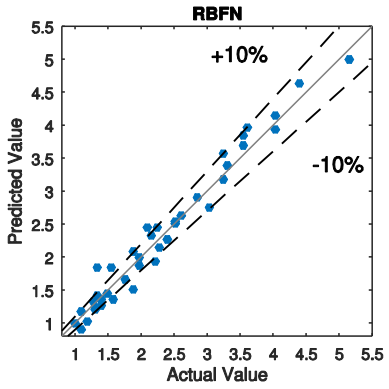


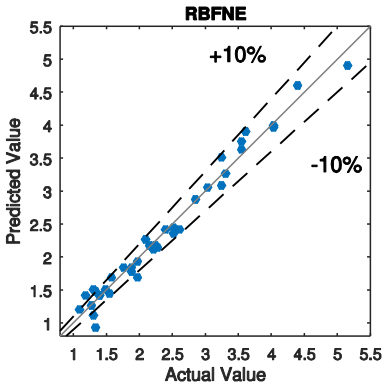


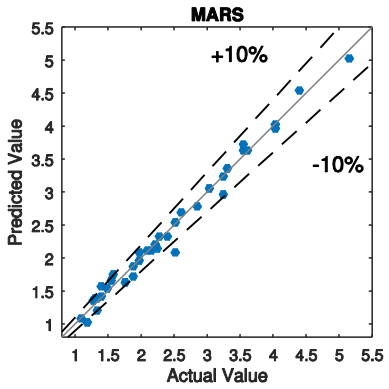
Structure __ Sydney __ PI Structure

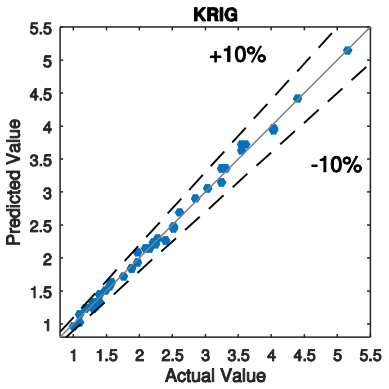




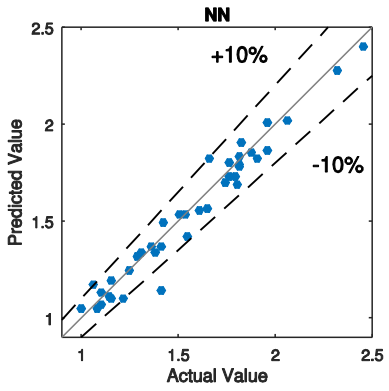


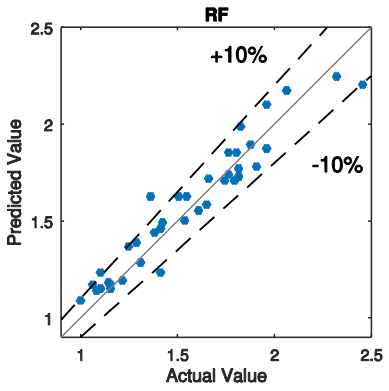


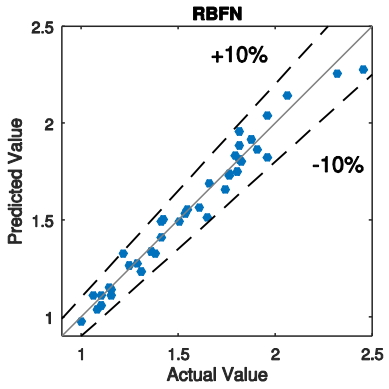


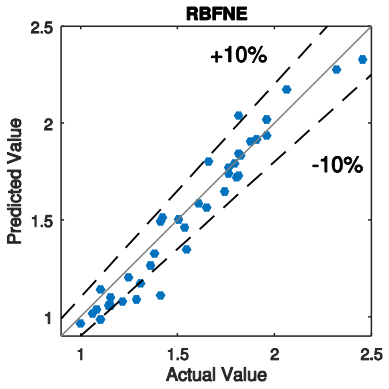


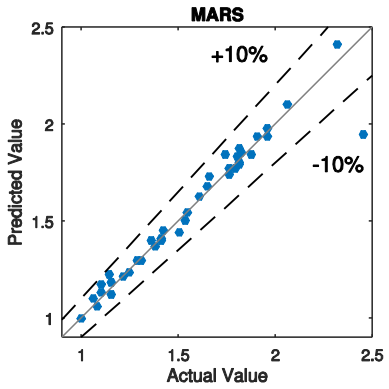
Cooling __ Sydney __ PI Structure

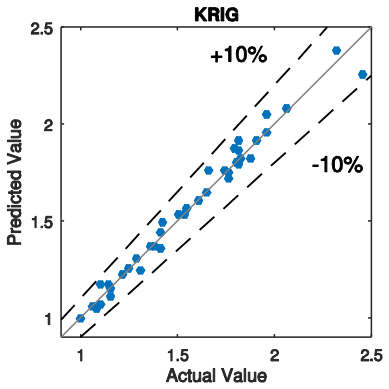




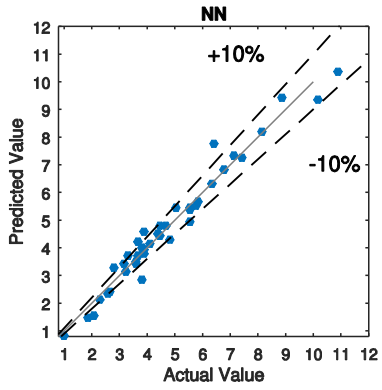


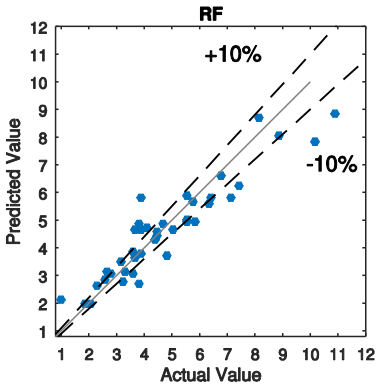


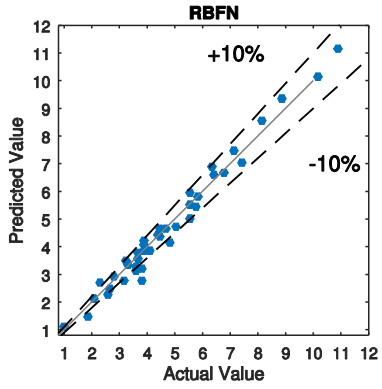


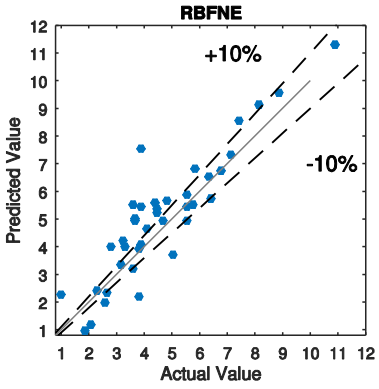


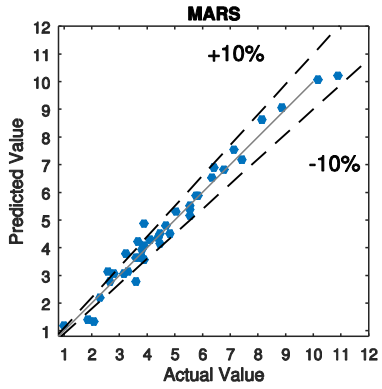
Heating __ Sydney __ PI Structure

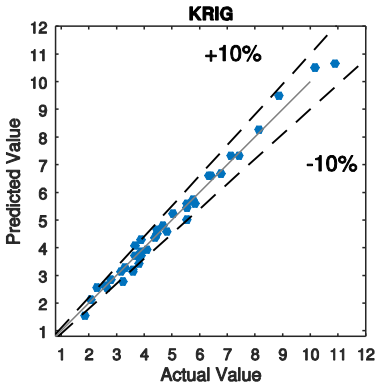




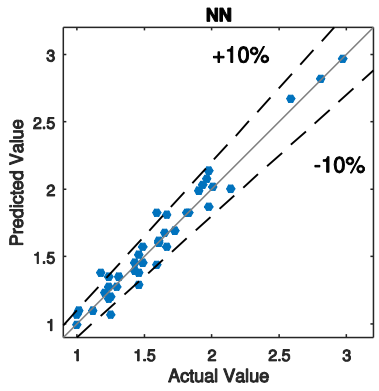


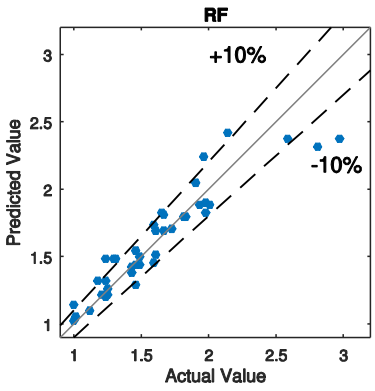


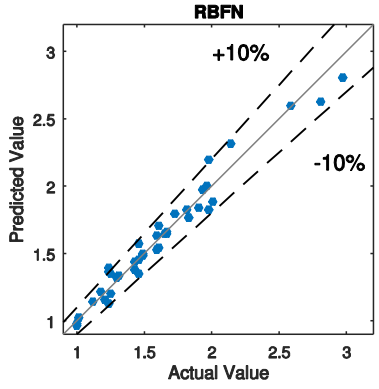


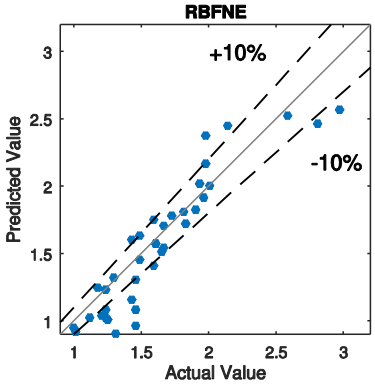


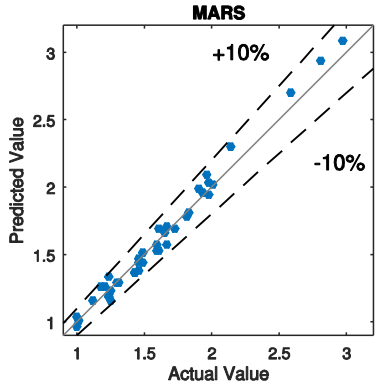
Lighting __ Sydney __ PI Structure

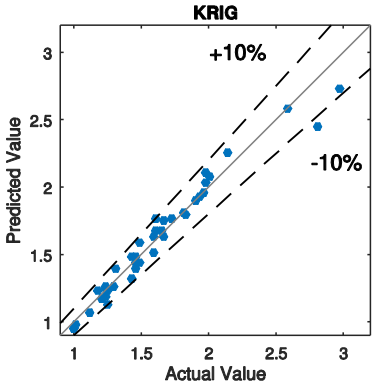




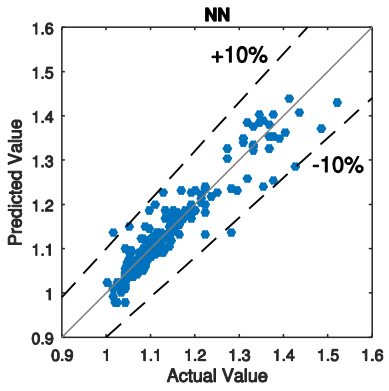


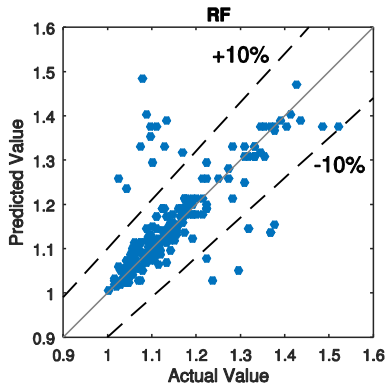


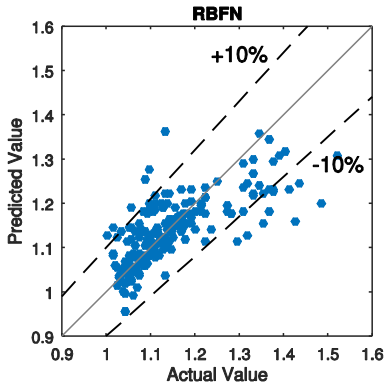


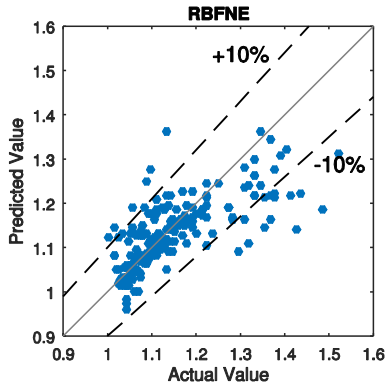


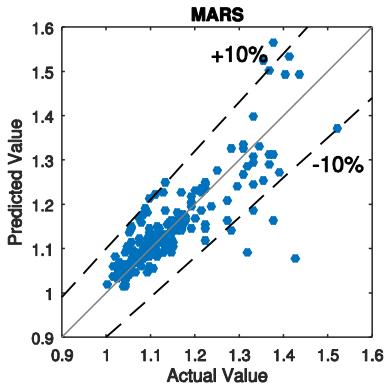
Energy Overall __ Abu Dhabi __ Airport terminal

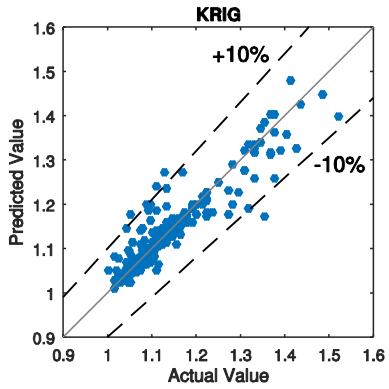




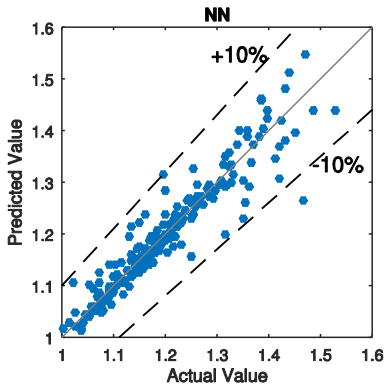


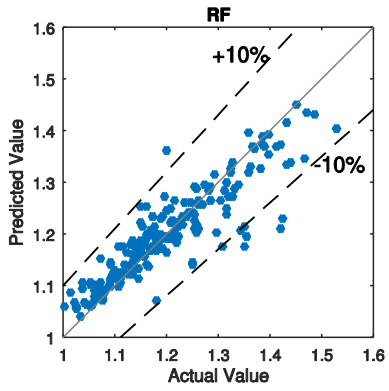


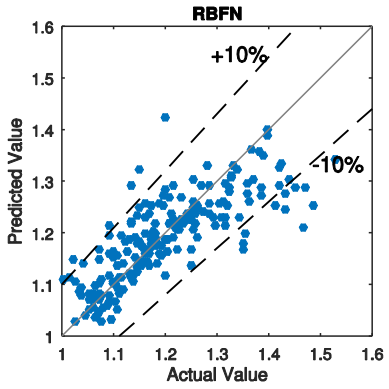


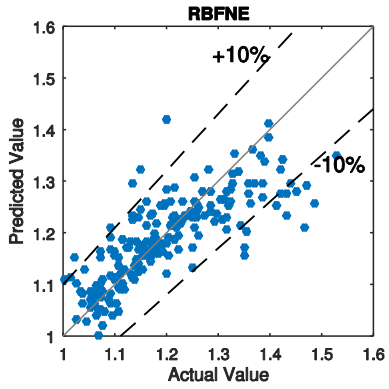


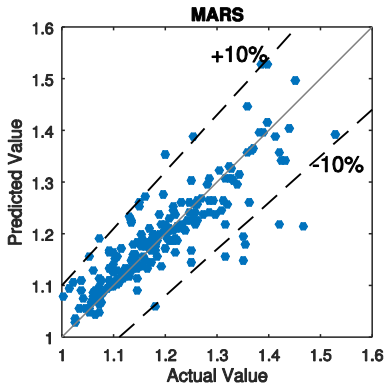
Cooling+Lighting __ Abu Dhabi __ Airport terminal

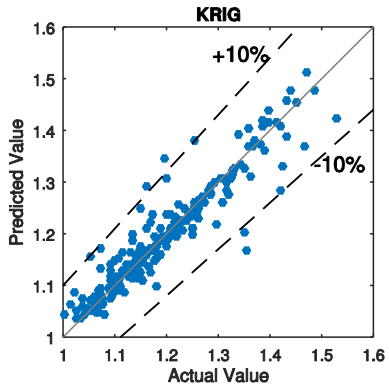




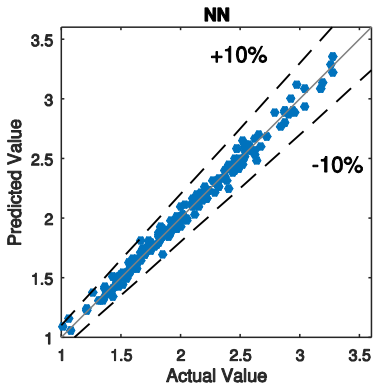


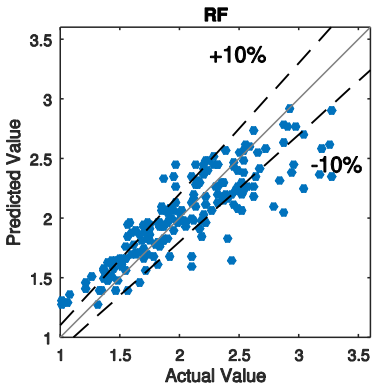


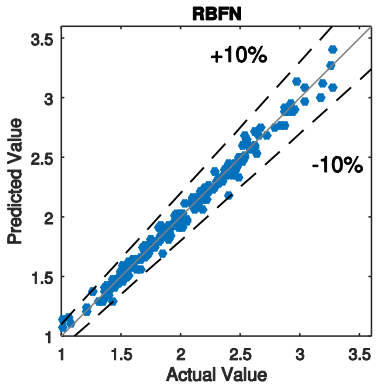


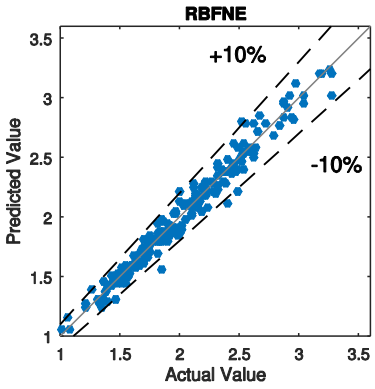


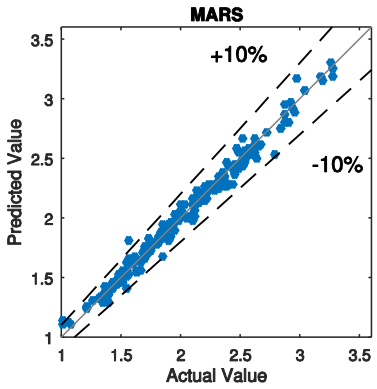
Structure __ Abu Dhabi __ Airport terminal

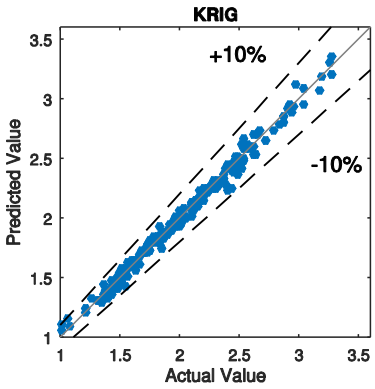




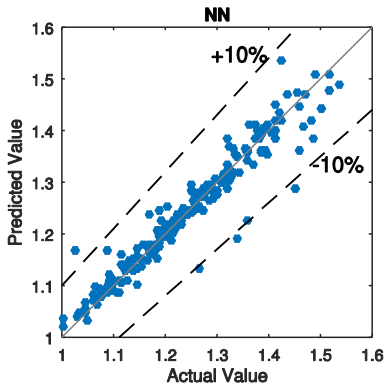


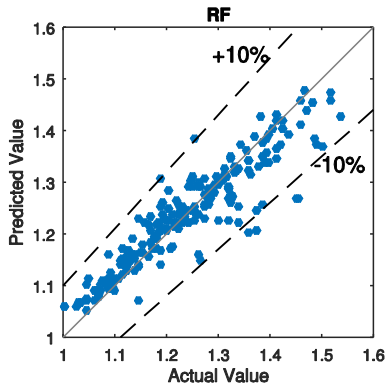


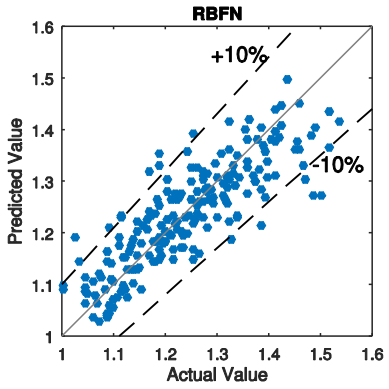


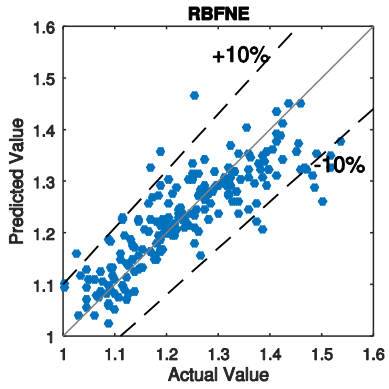


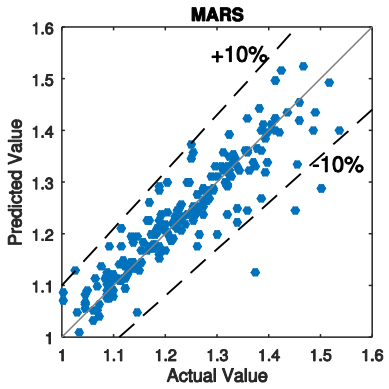
Cooling __ Abu Dhabi __ Airport terminal

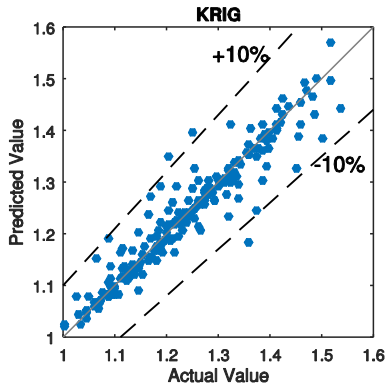




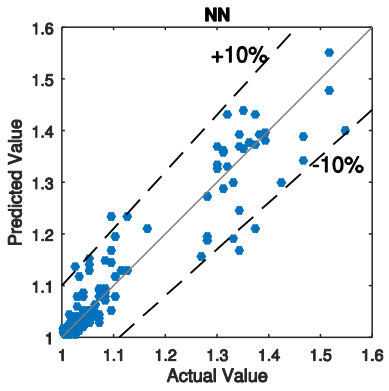


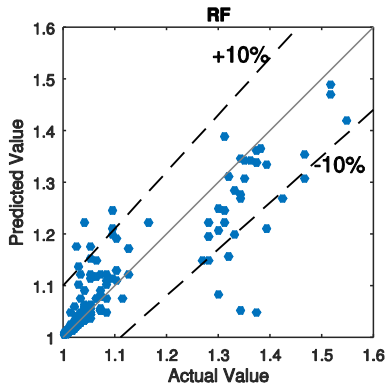


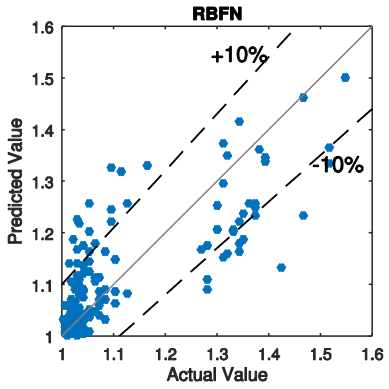


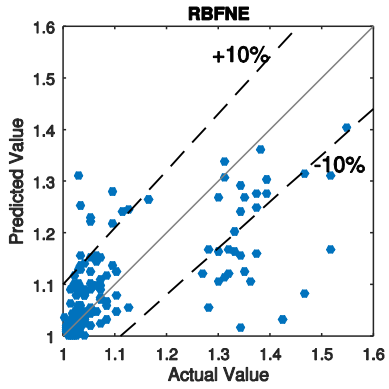


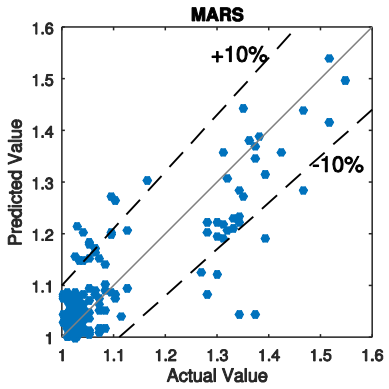
Lighting __ Abu Dhabi __ Airport terminal

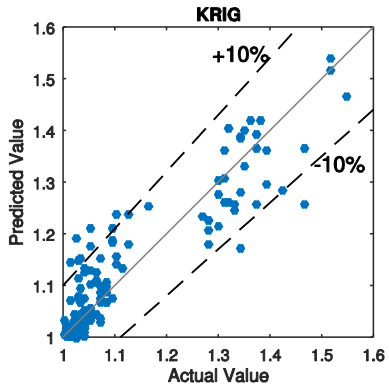




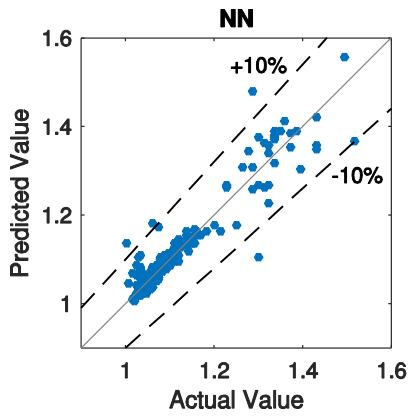


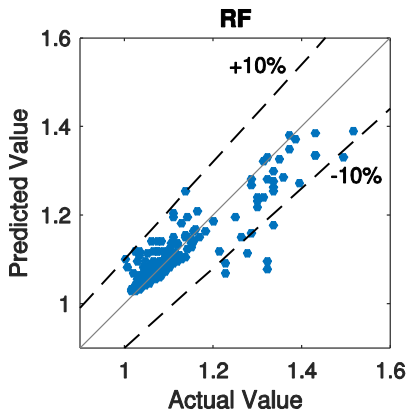


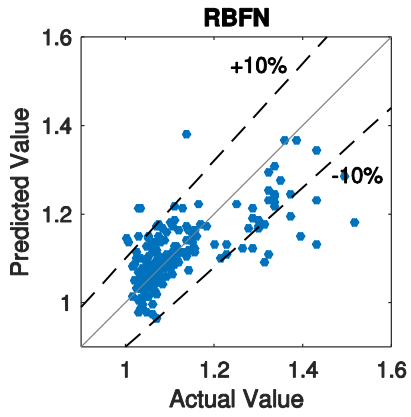


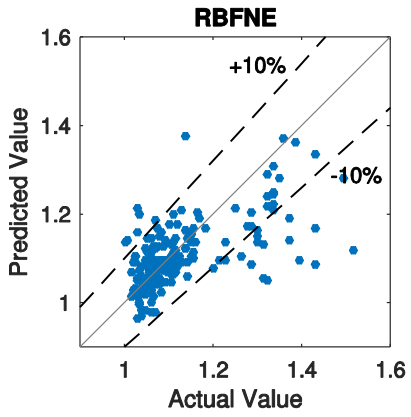


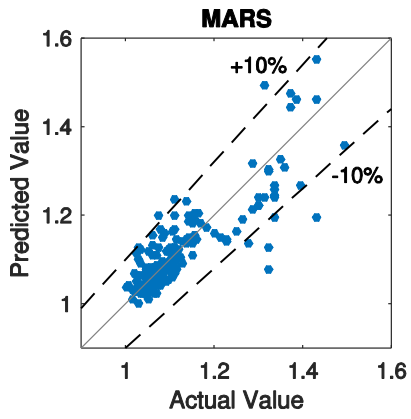
Energy Overall __ Abu Dhabi Rotated __ Airport terminal

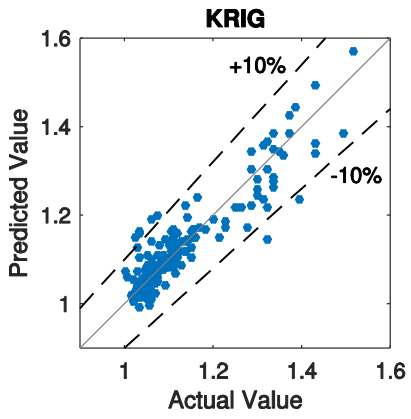




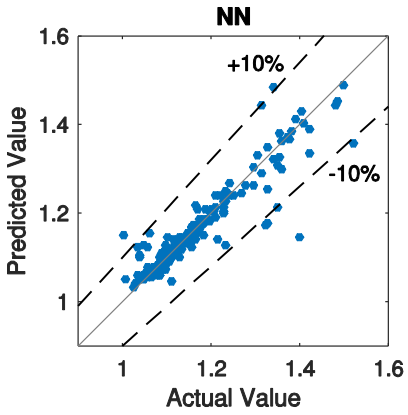


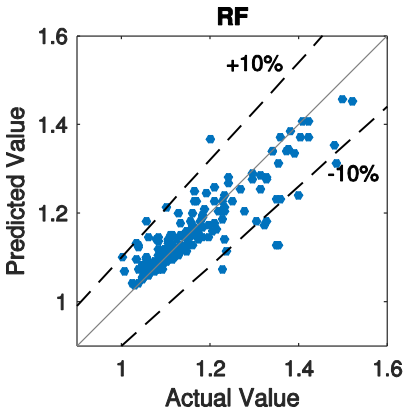


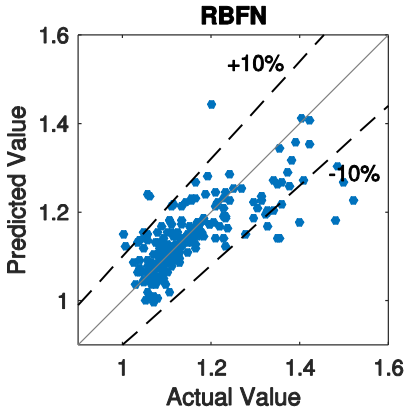


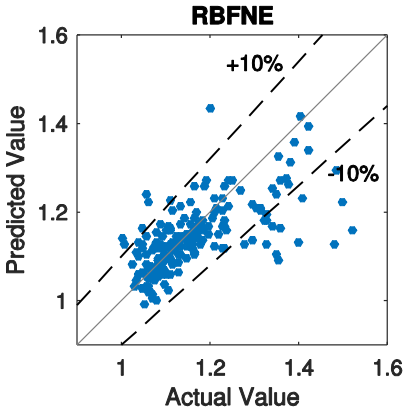


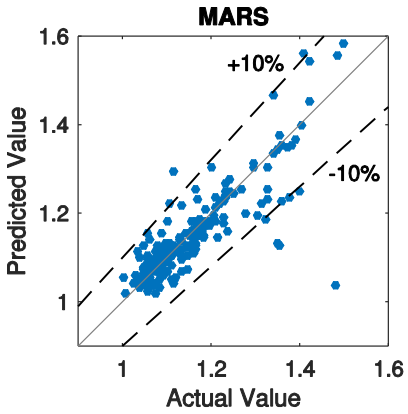
Heating+Cooling+Lighting __ Abu Dhabi Rotated__ Airport terminal

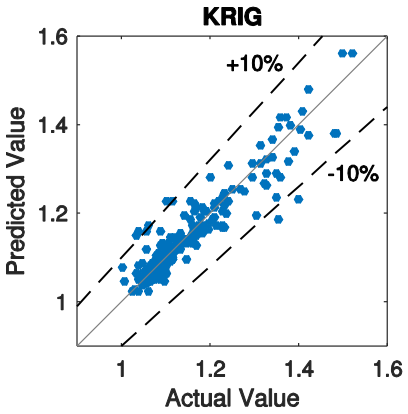




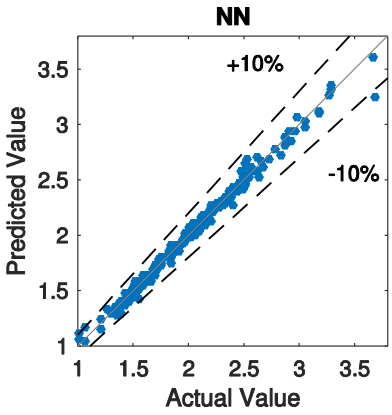


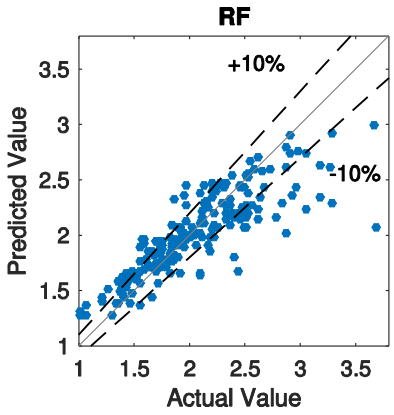


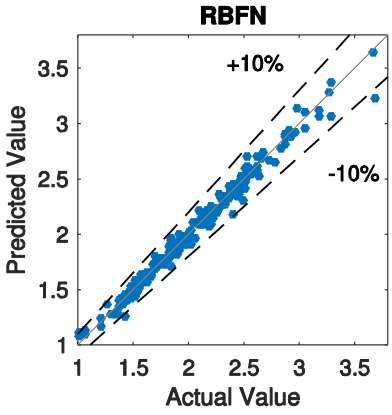


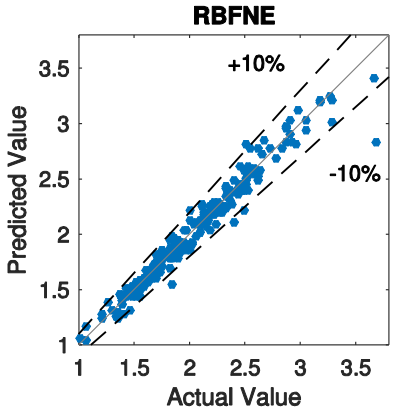


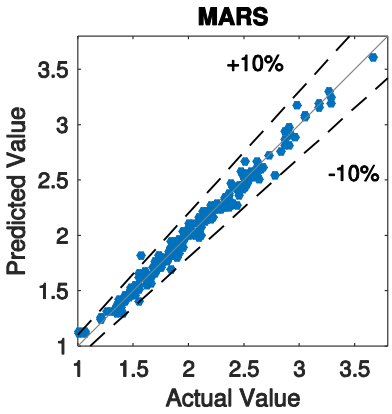
Structure __ Abu Dhabi Rotated __ Airport terminal

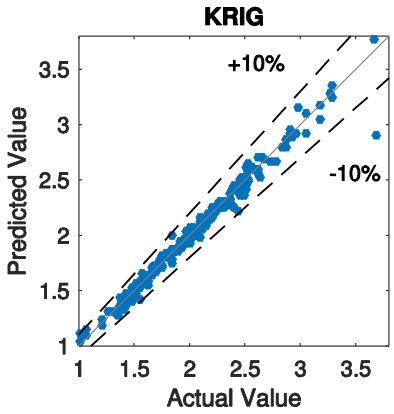




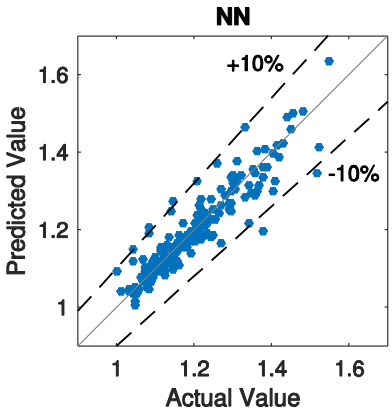


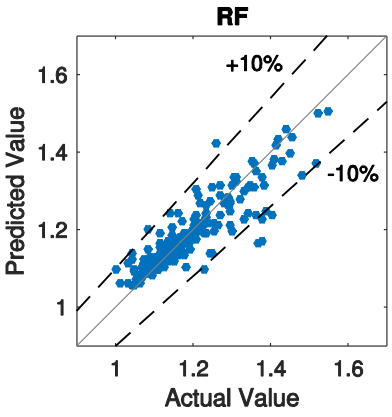


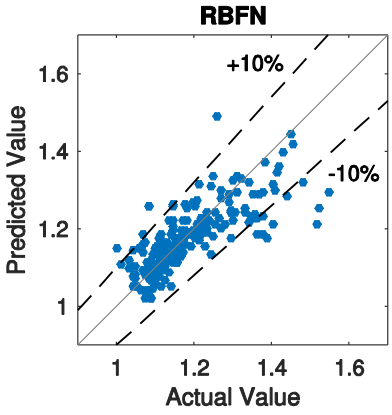


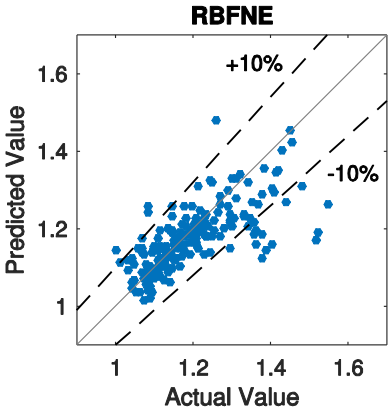


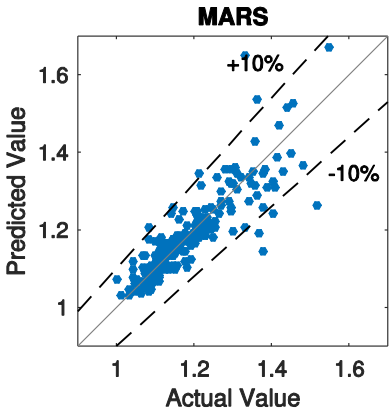
Cooling __ Abu Dhabi Rotated __ Airport terminal

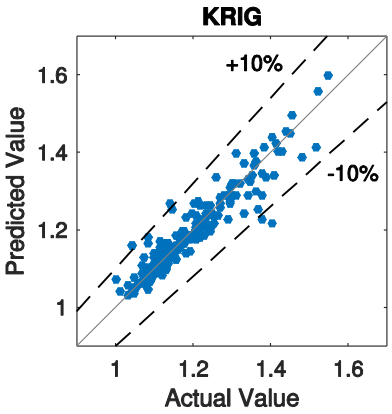




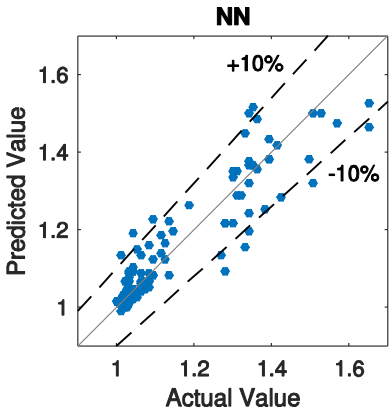


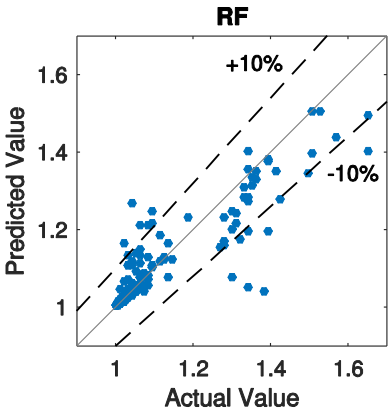


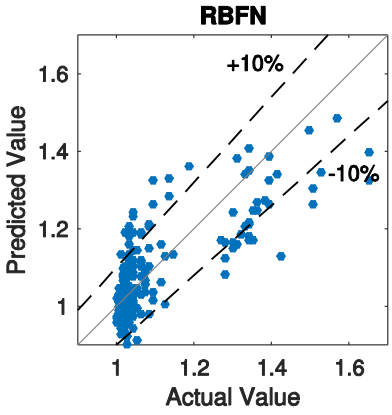


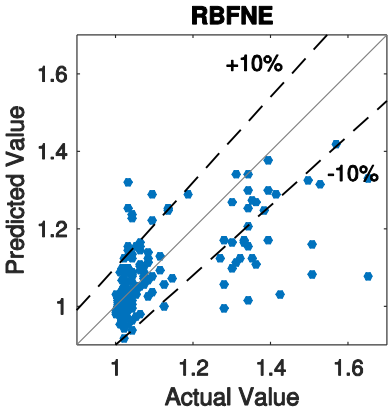


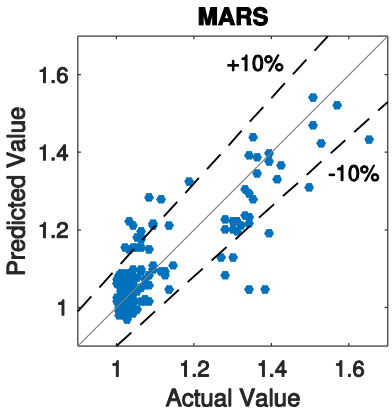
Lighting __ Abu Dhabi Rotated __ Airport terminal

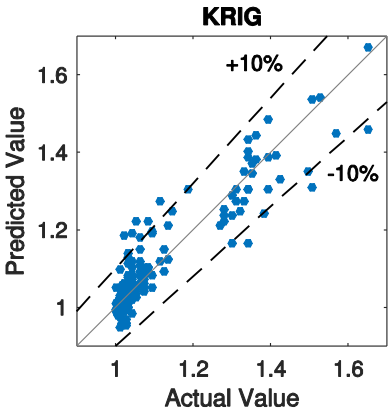




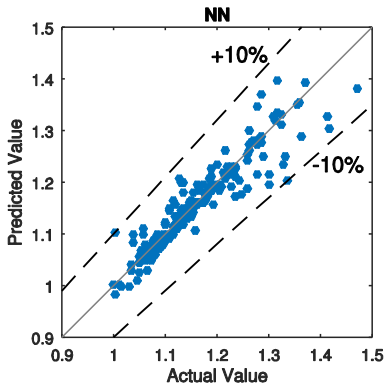


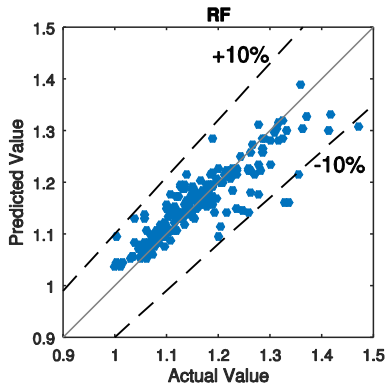


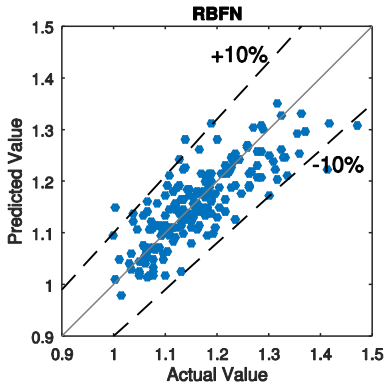


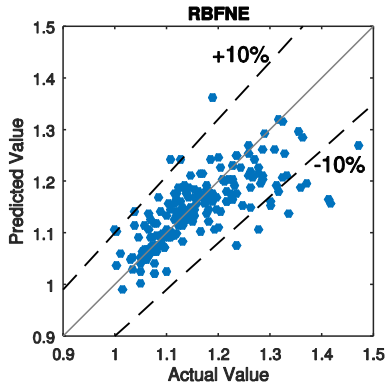


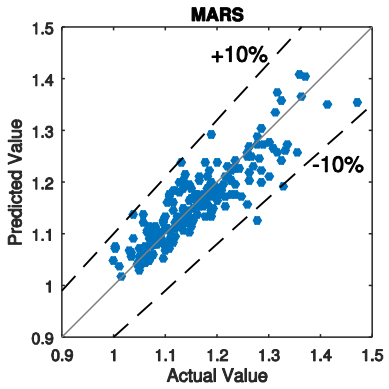
Energy Overall __ Boston __ Airport terminal

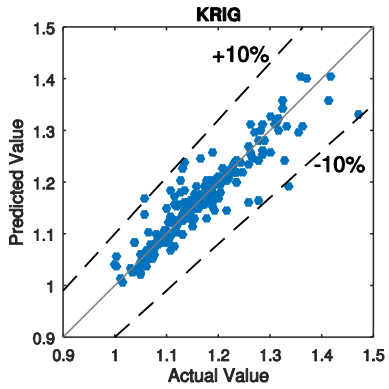




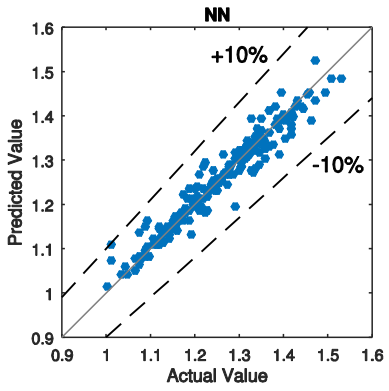


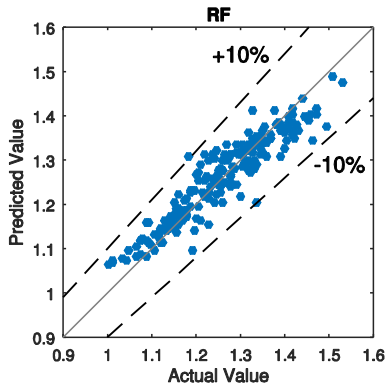


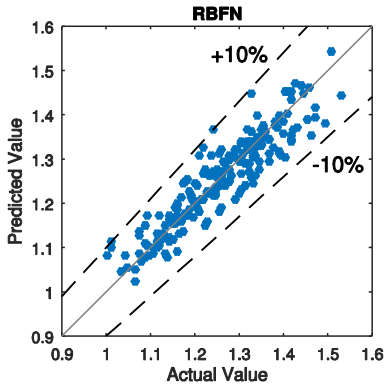


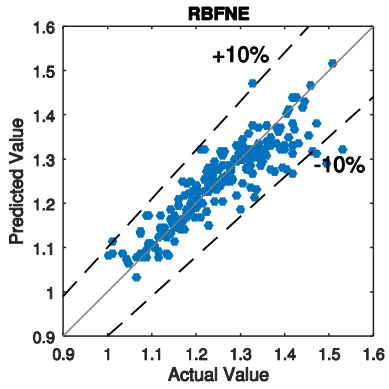


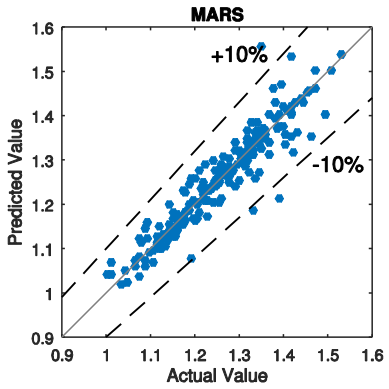
Heating+Cooling+Lighting __ Boston __ Airport terminal

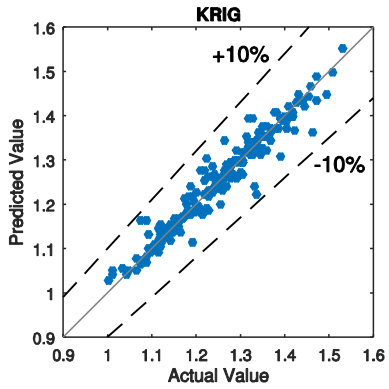




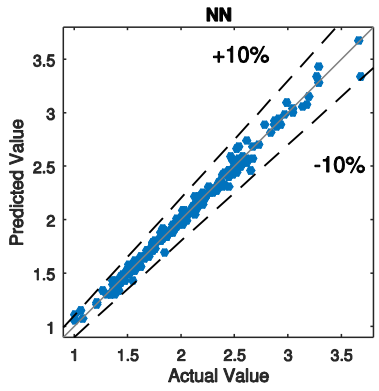


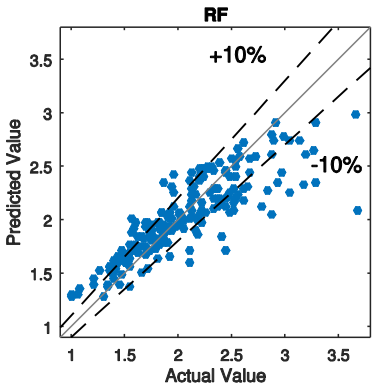


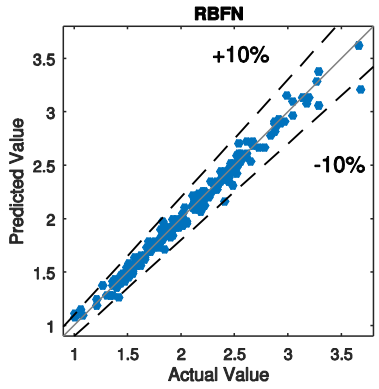


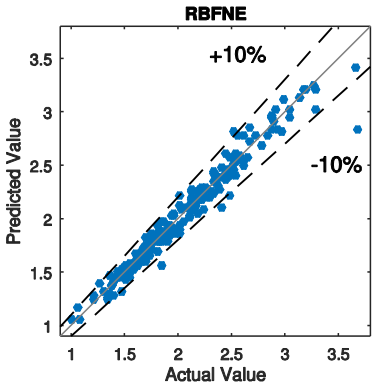


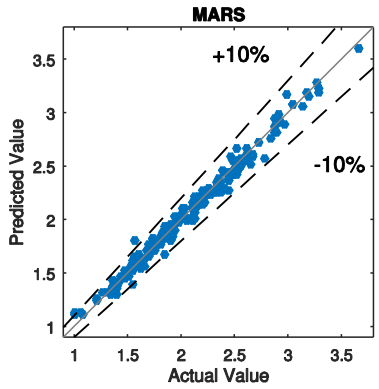
Structure __ Boston __ Airport terminal

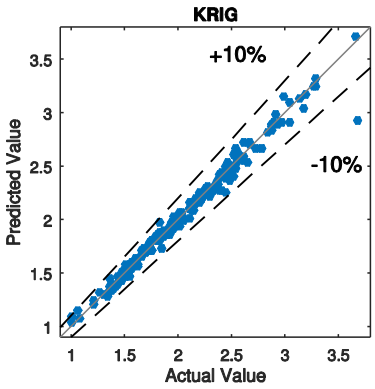




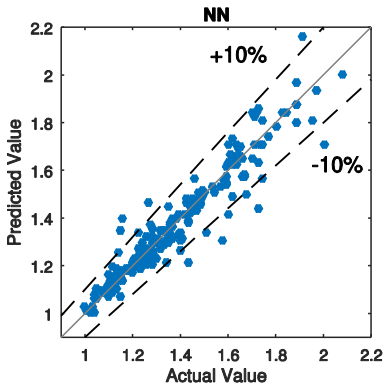


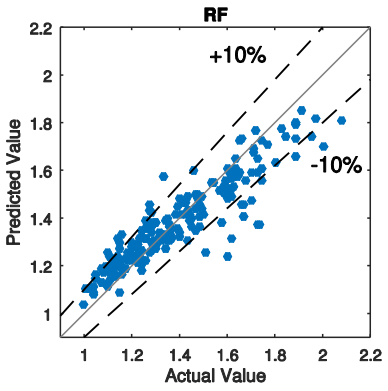


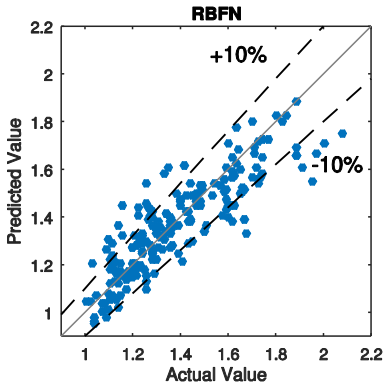


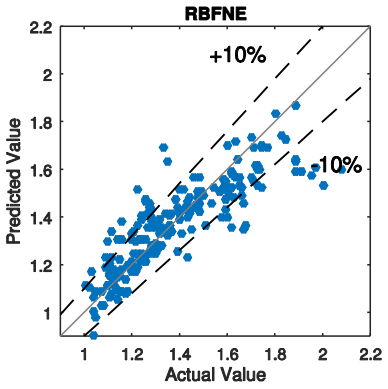


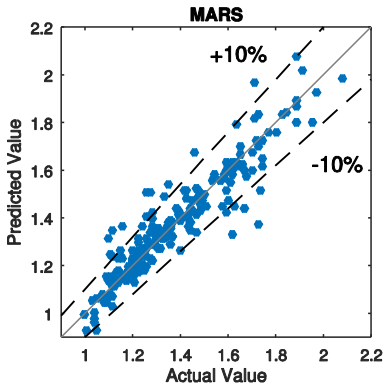
Cooling __ Boston __ Airport terminal

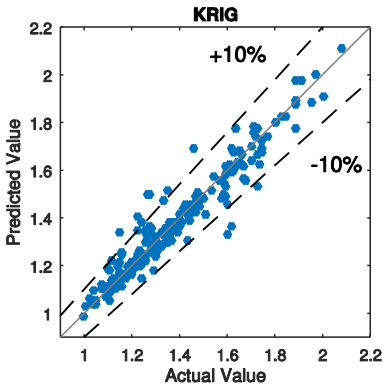




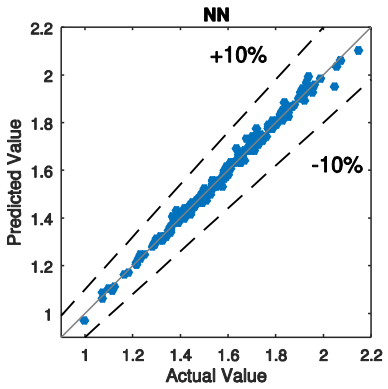


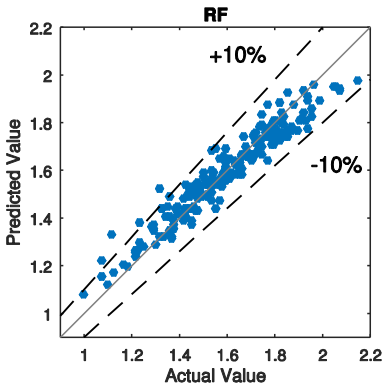


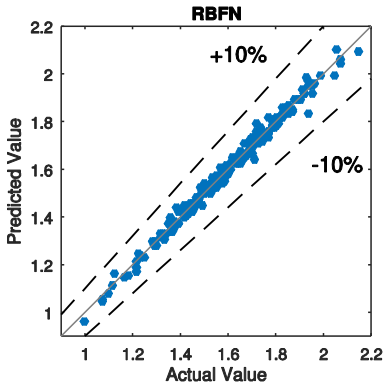


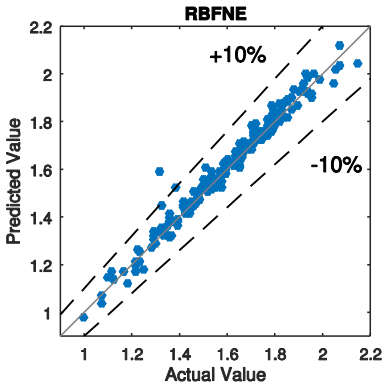


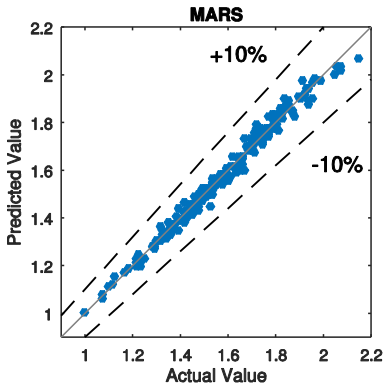
Heating __ Boston __ Airport terminal

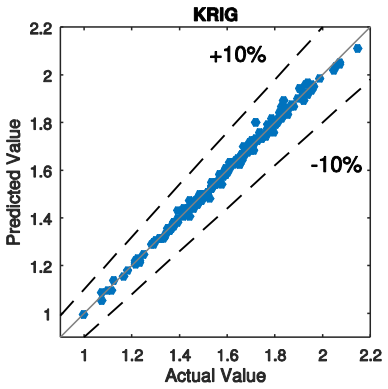




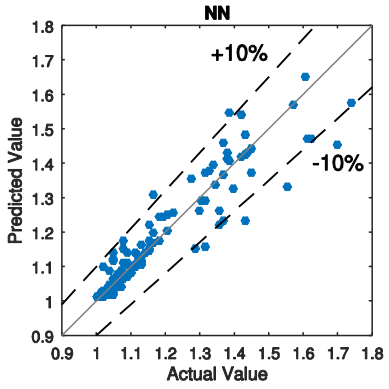


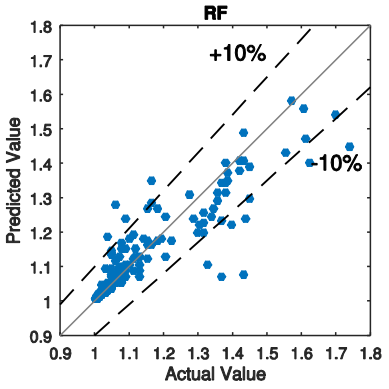


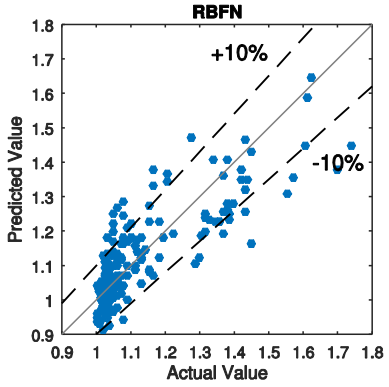


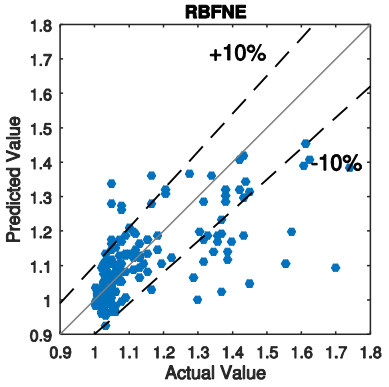


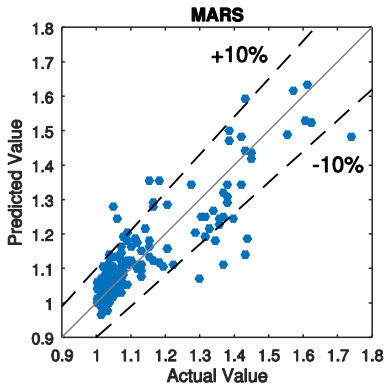
Lighting __ Boston __ Airport terminal

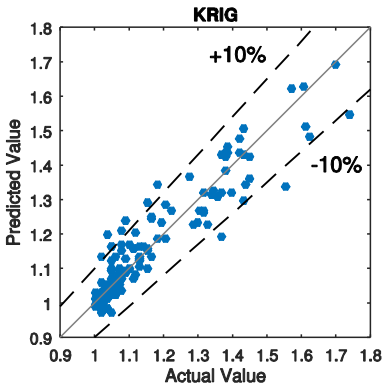




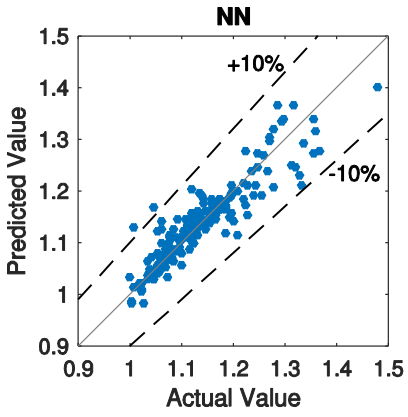


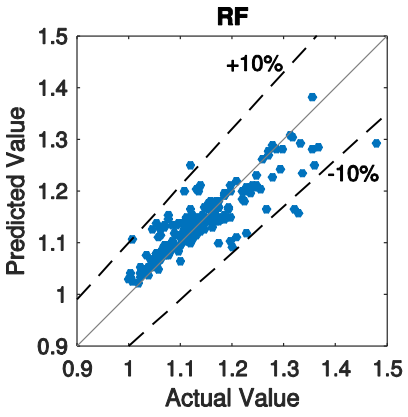


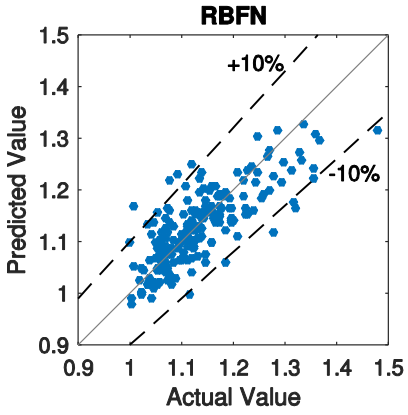


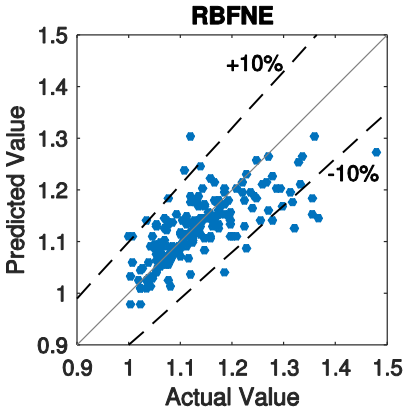


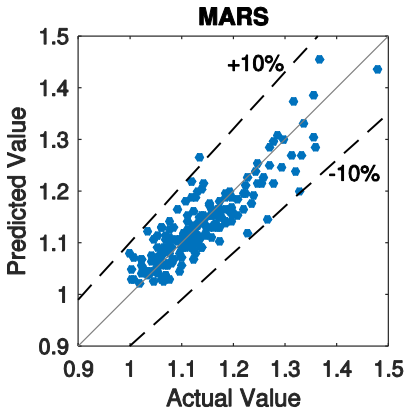
Energy Overall __ Boston Rotated __ Airport terminal

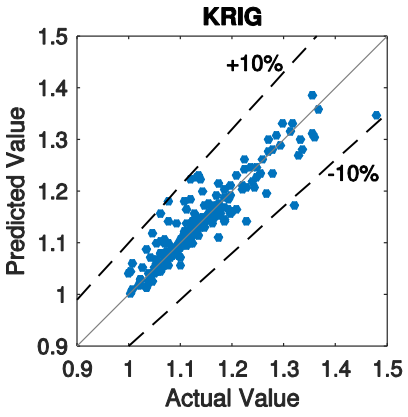




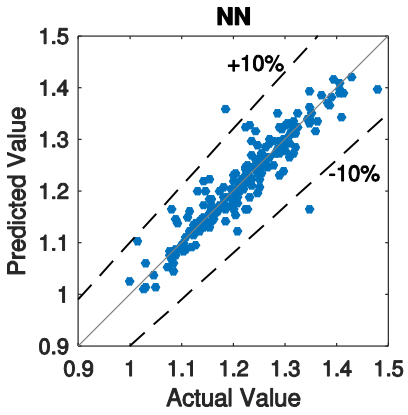


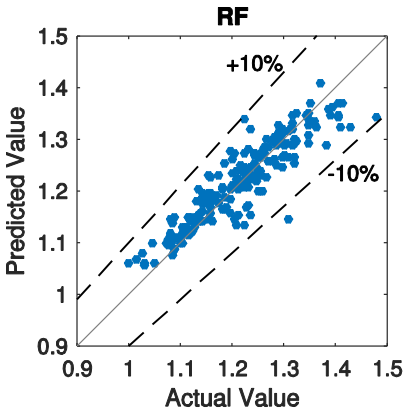


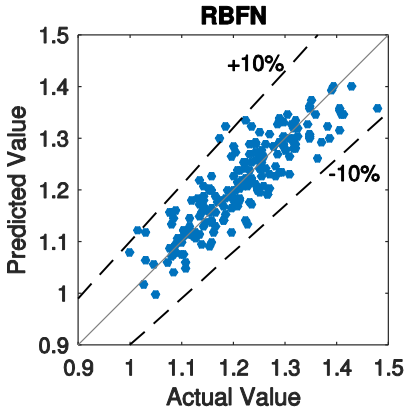


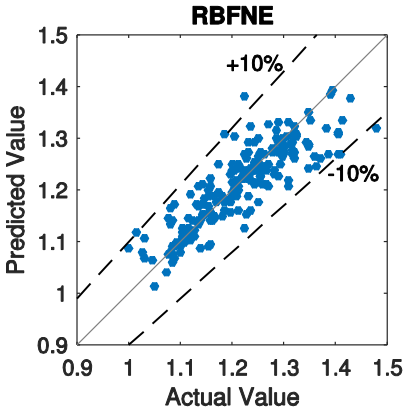


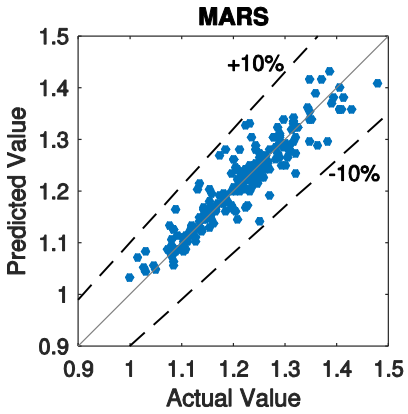
Heating+Cooling+Lighting __ Boston Rotated __ Airport terminal

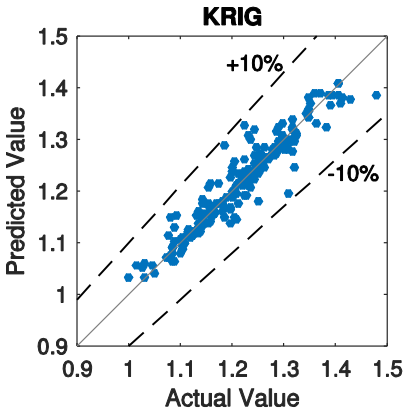




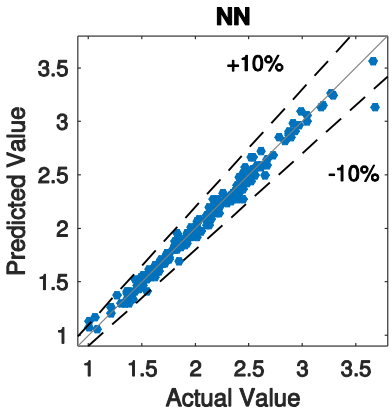


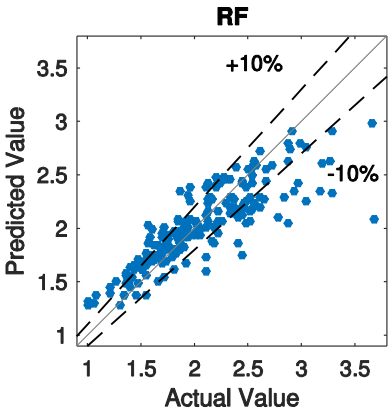


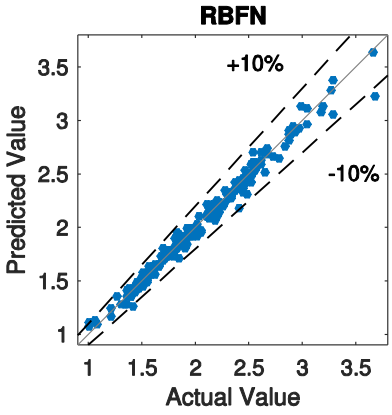


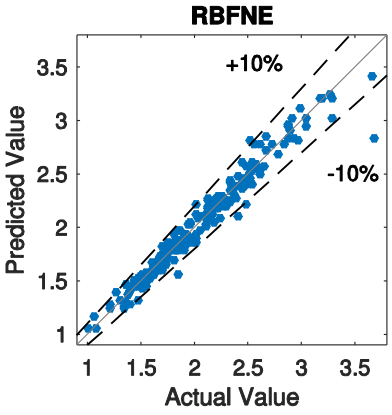


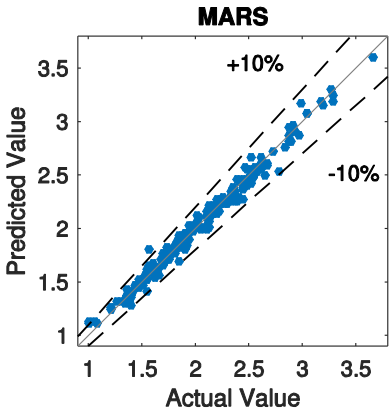
Structure __ Boston Rotated __ Airport terminal

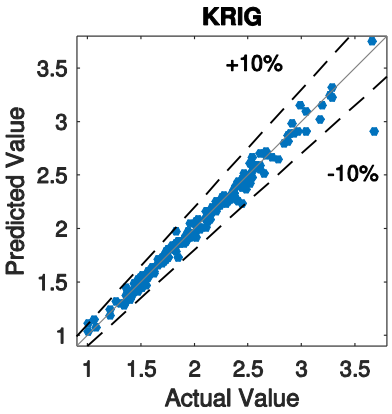




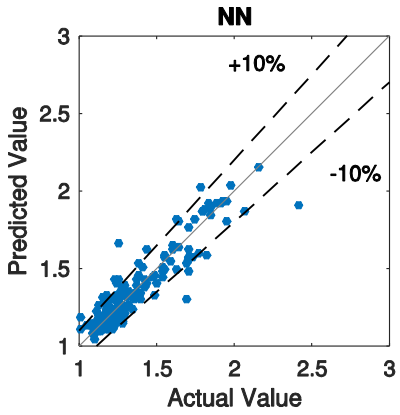


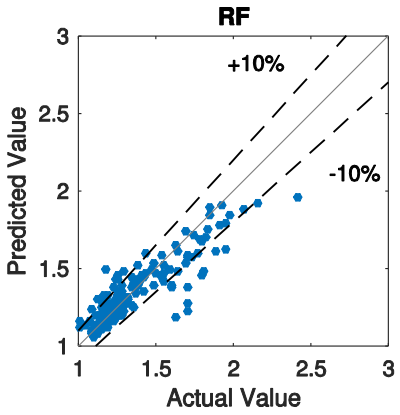


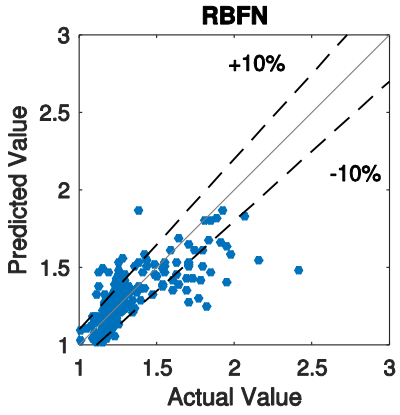


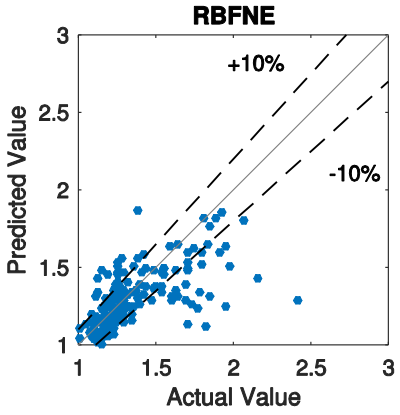


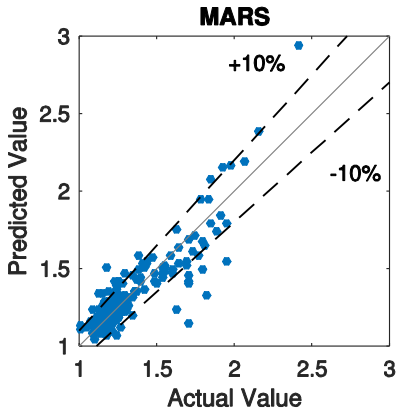
Cooling __ Boston Rotated __ Airport terminal

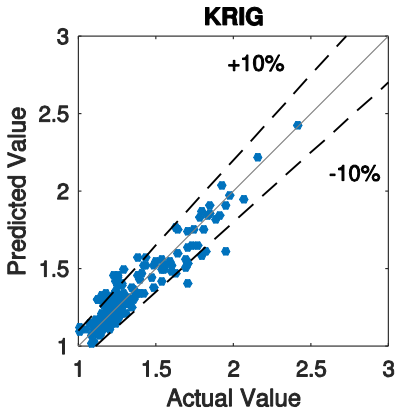




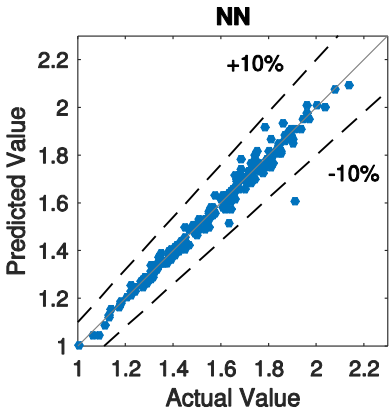


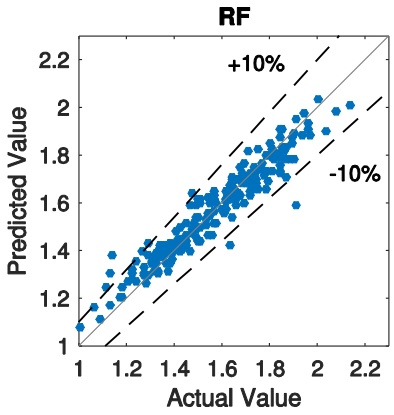


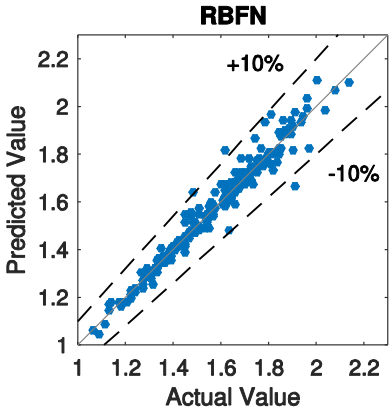


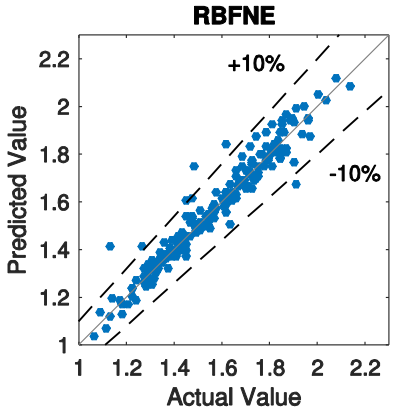


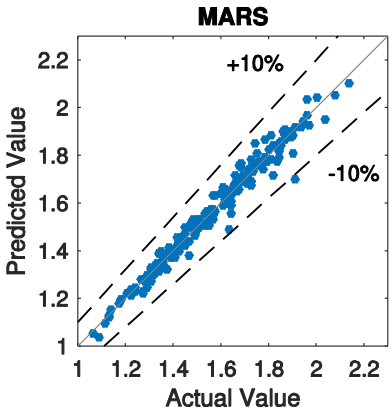
Heating __ Boston Rotated __ Airport terminal

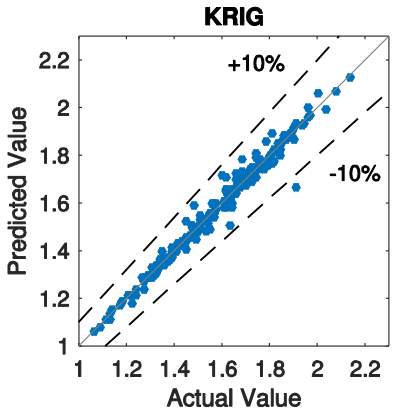




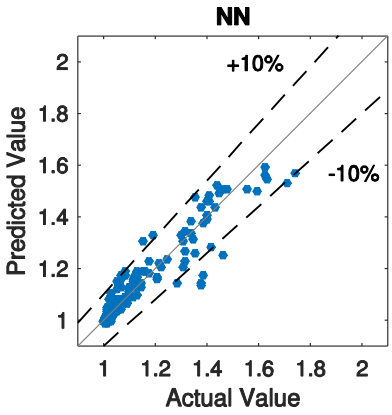


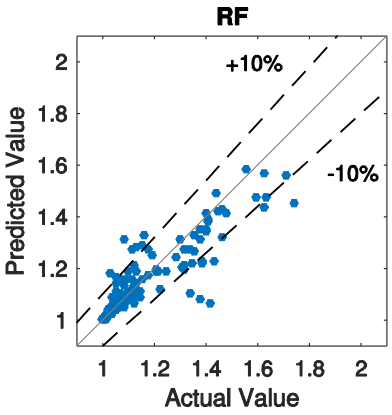


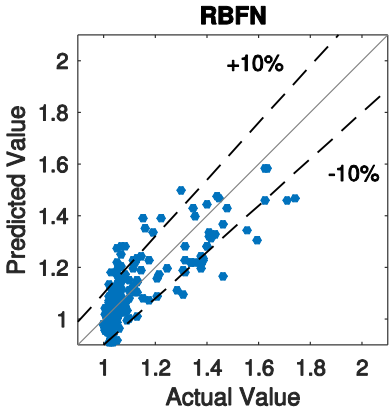


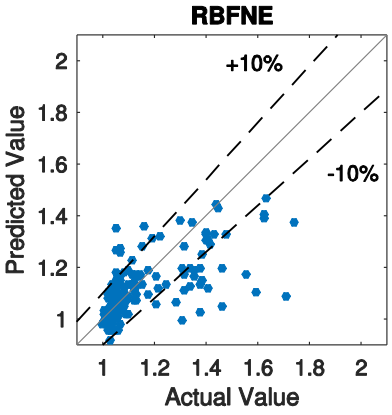


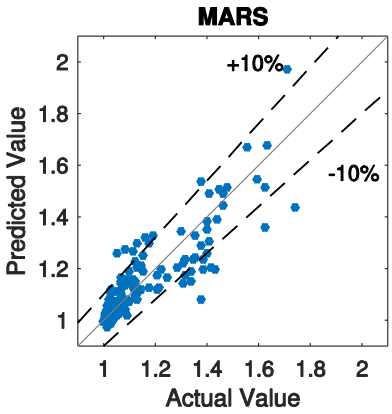
Lighting __ Boston Rotated __ Airport terminal

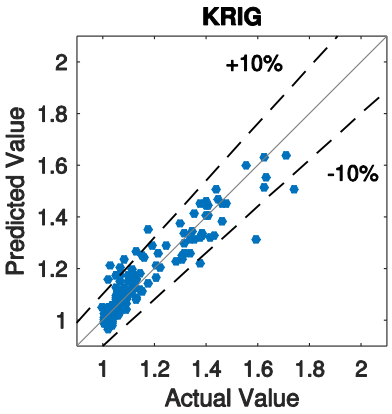




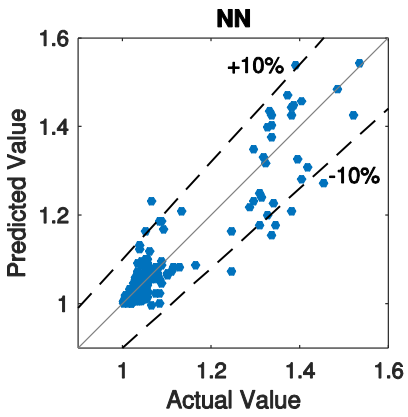


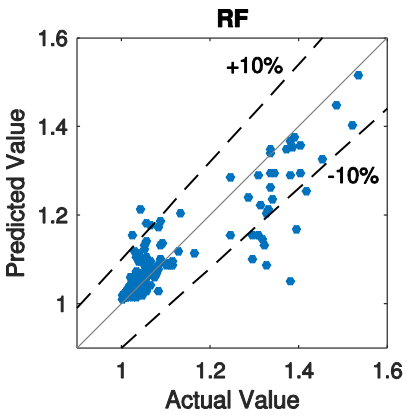


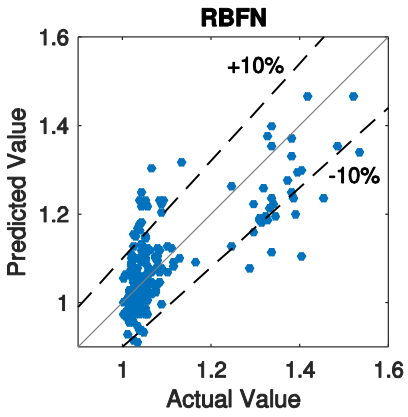


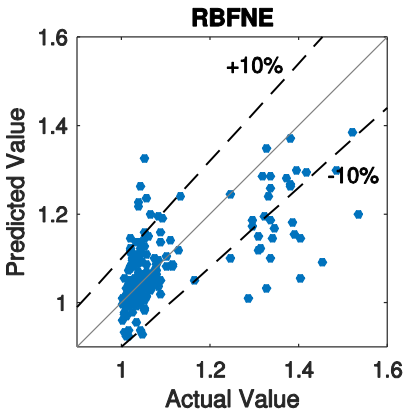


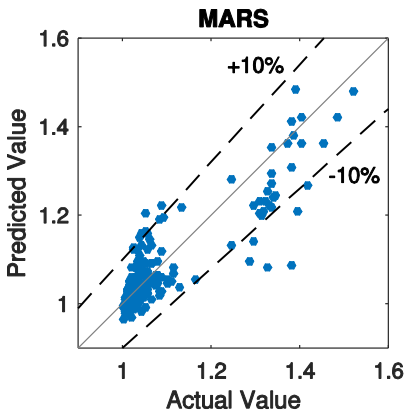
Energy Overall __ Sydney __ Airport terminal

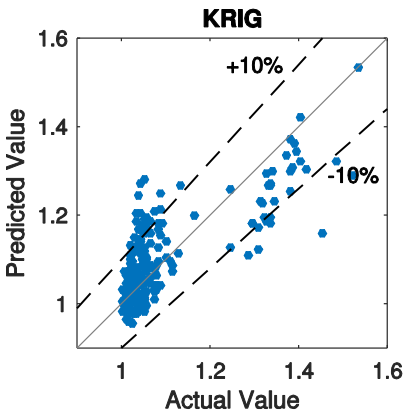




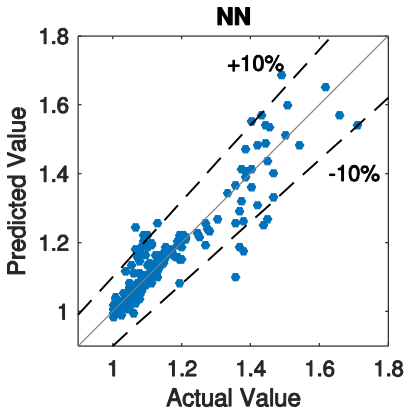


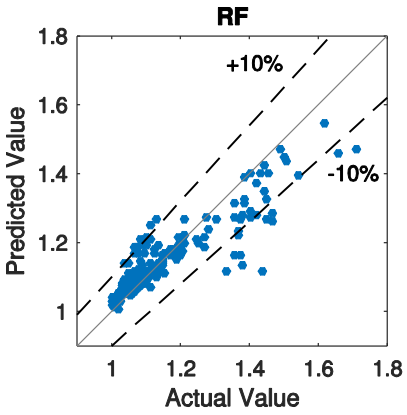


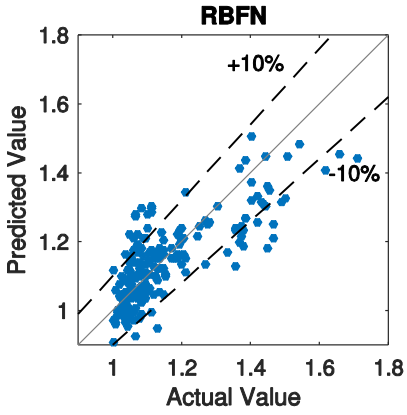


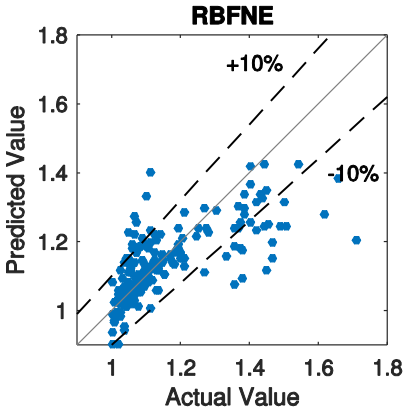


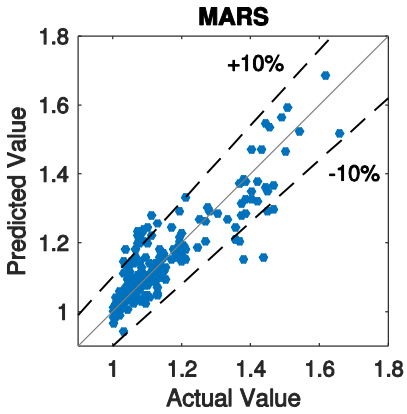
Heating+Cooling+Lighting __ Sydney __ Airport terminal

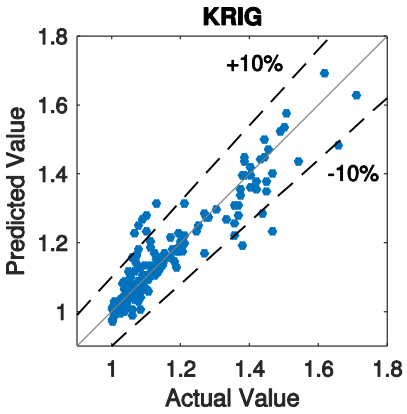




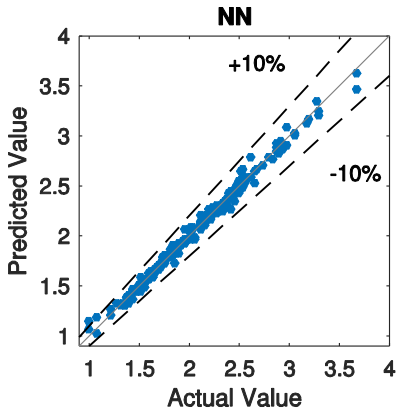


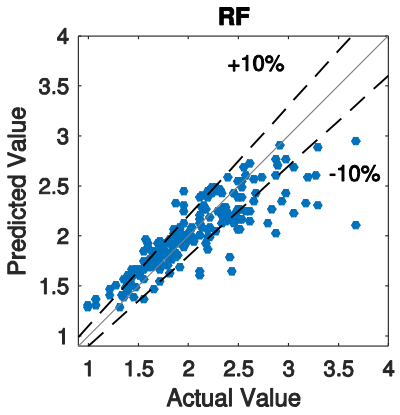


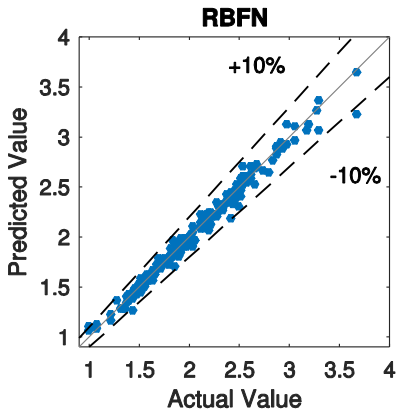


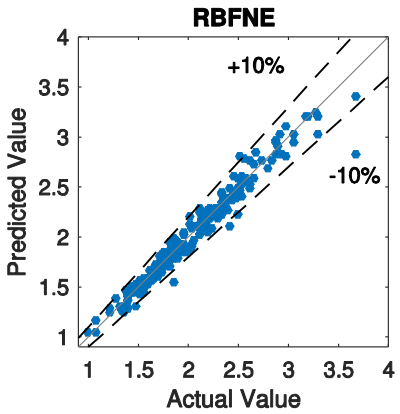


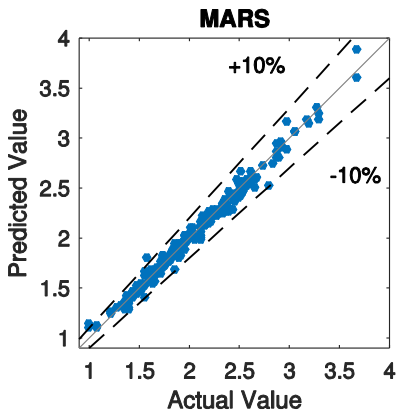
Structure __ Sydney __ Airport terminal

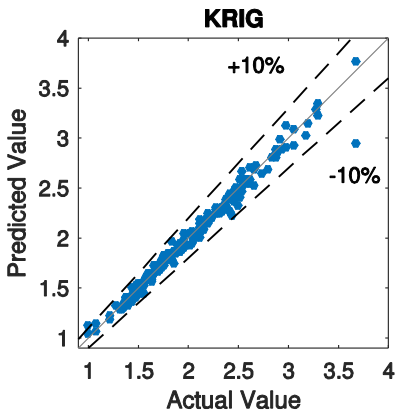




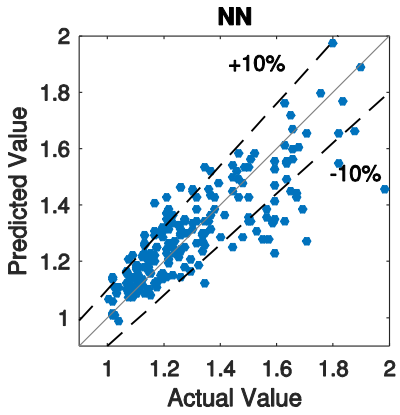


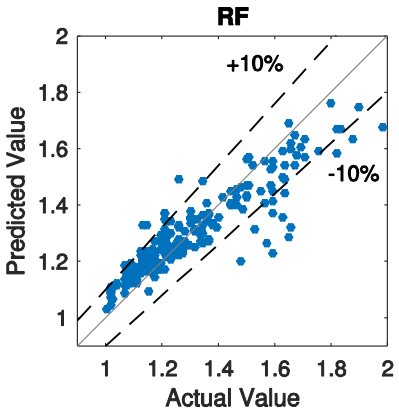


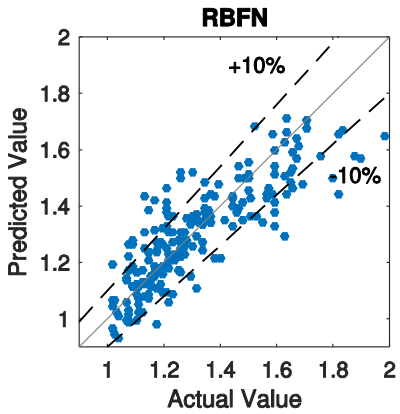


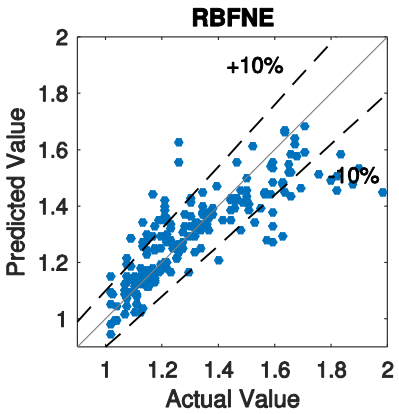


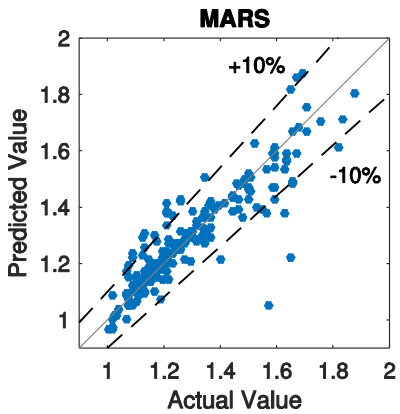
Cooling __ Sydney __ Airport terminal

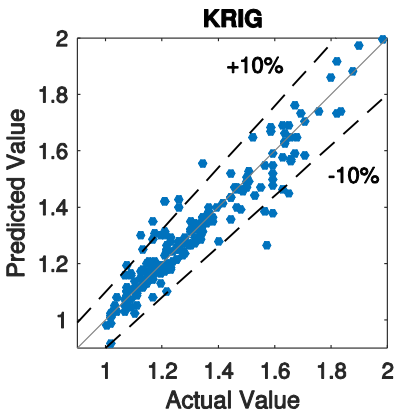




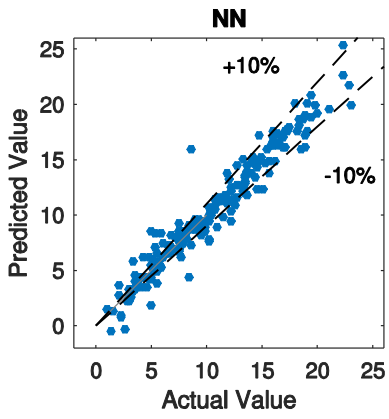


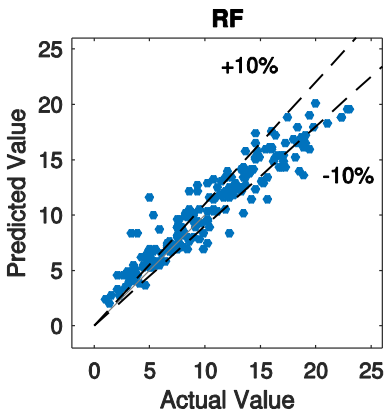


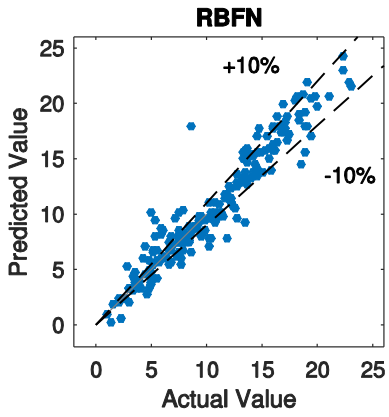


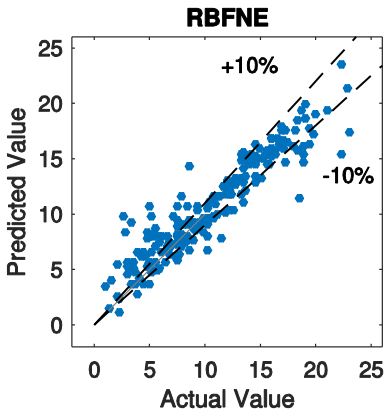


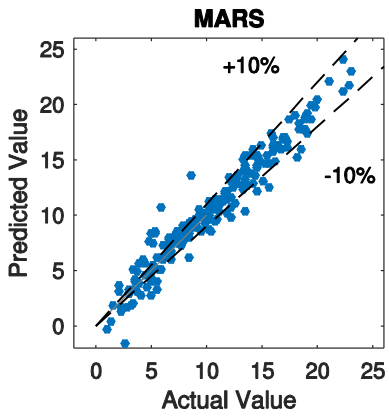
Heating __ Sydney __ Airport terminal

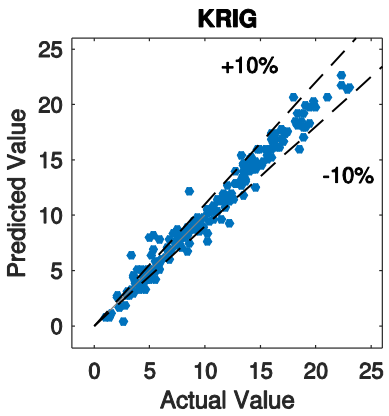




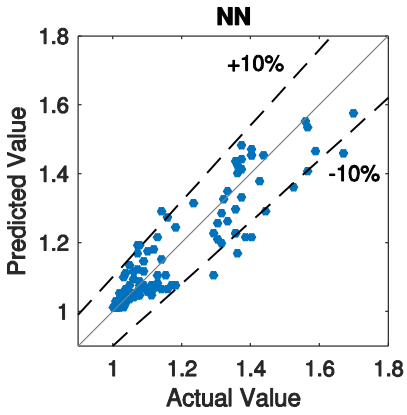


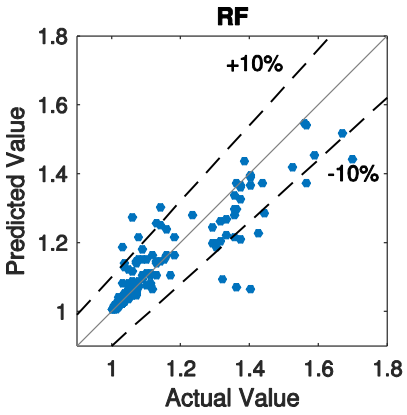


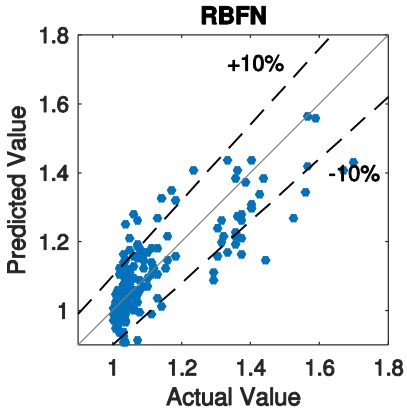


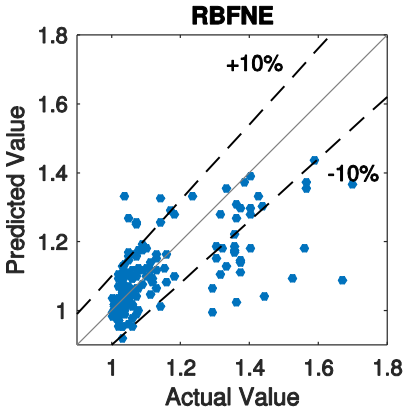


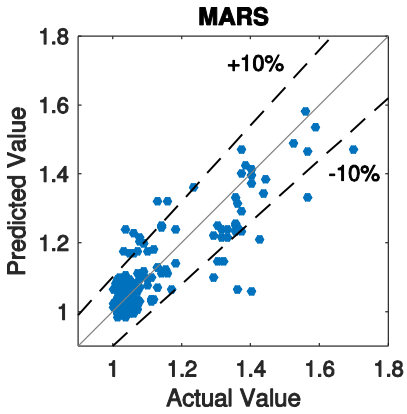
Lighting __ Sydney __ Airport terminal

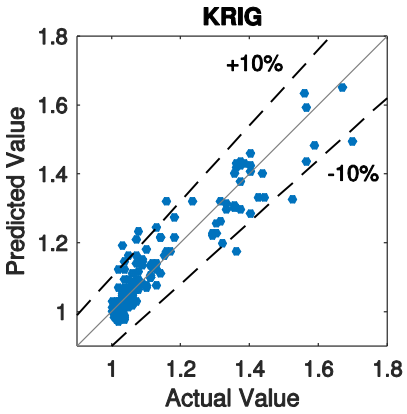




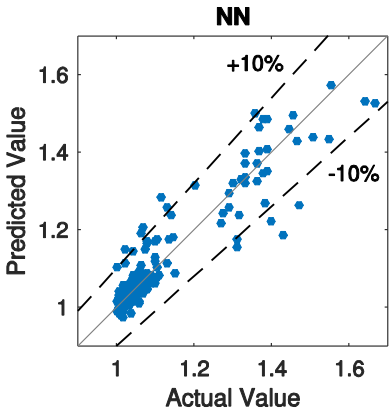


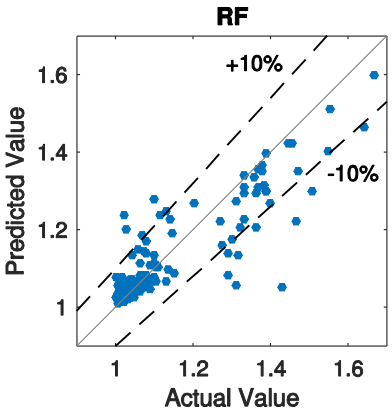


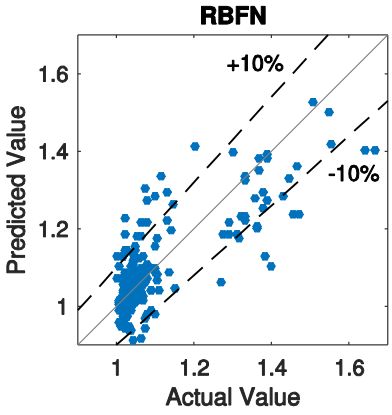


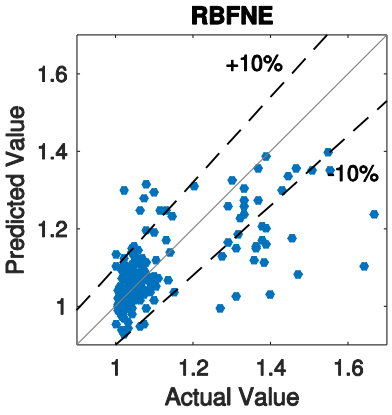


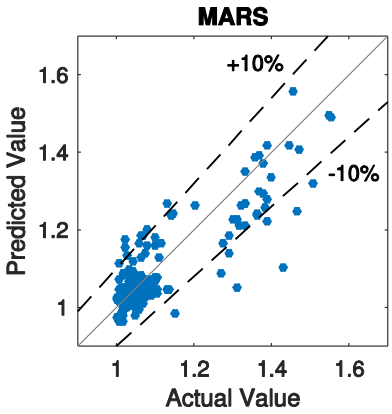
Energy Overall __ Sydney Rotated __ Airport terminal

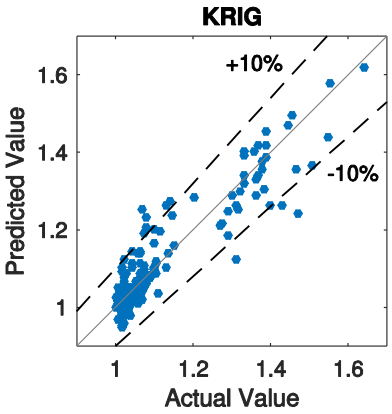




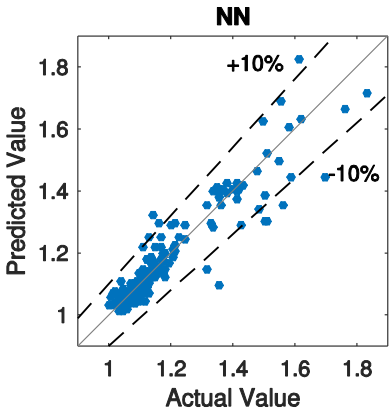


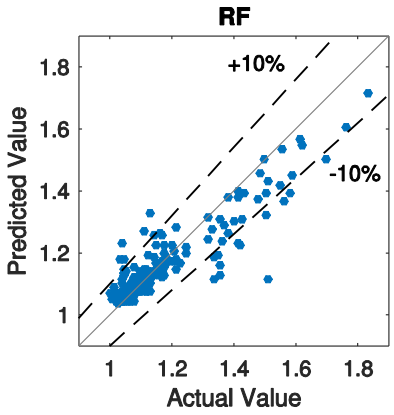


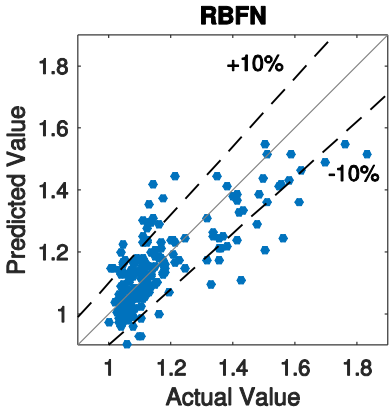


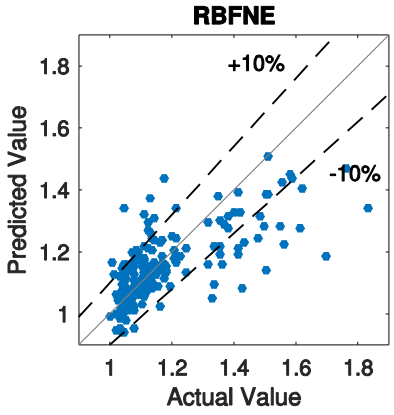


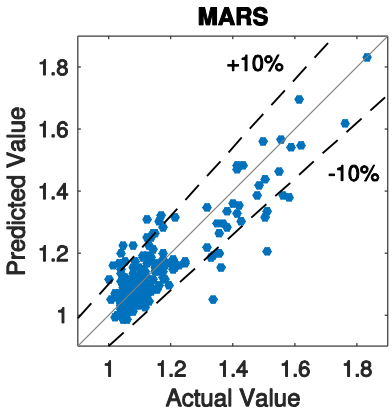
Heating+Cooling+Lighting __ Sydney Rotated __ Airport terminal

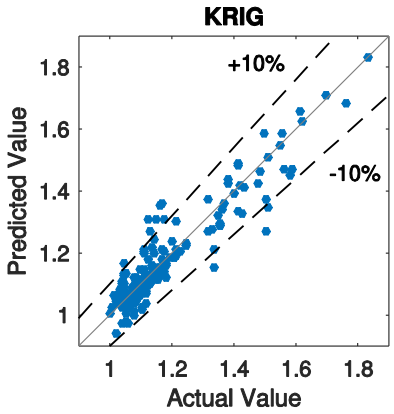




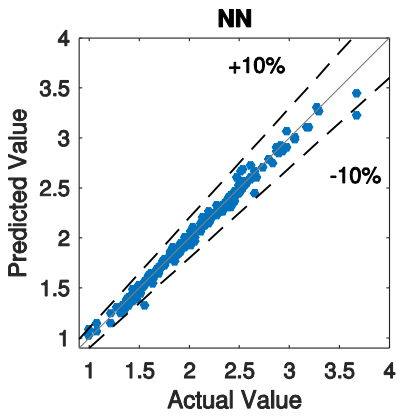


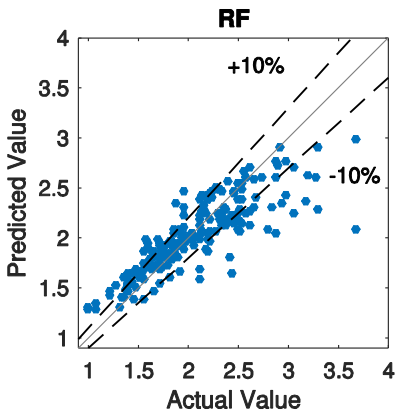


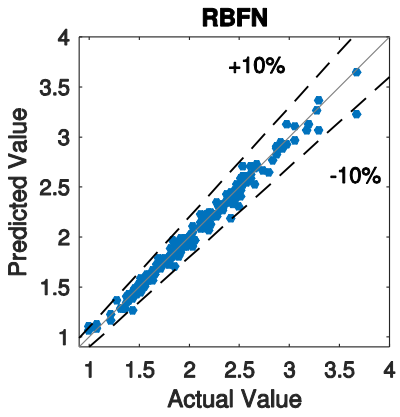


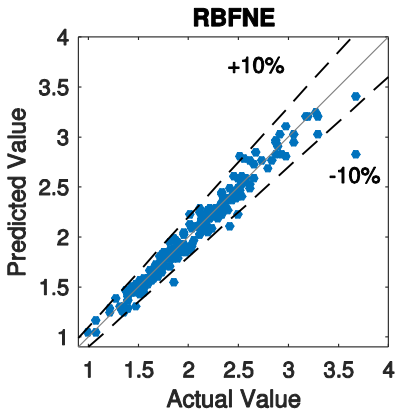


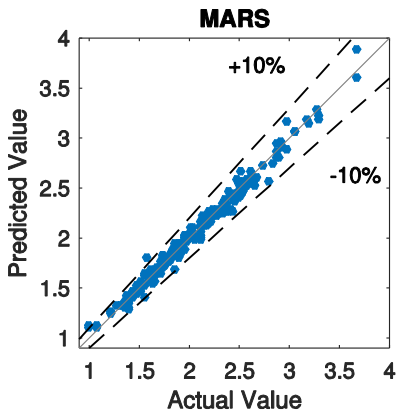
Structure __ Sydney Rotated __ Airport terminal

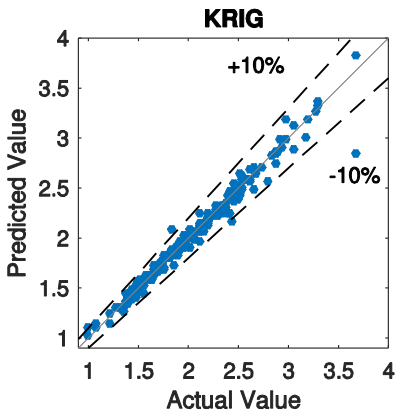




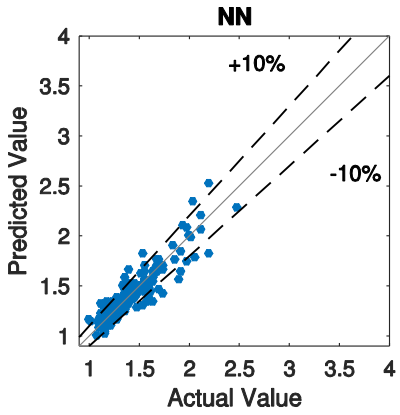


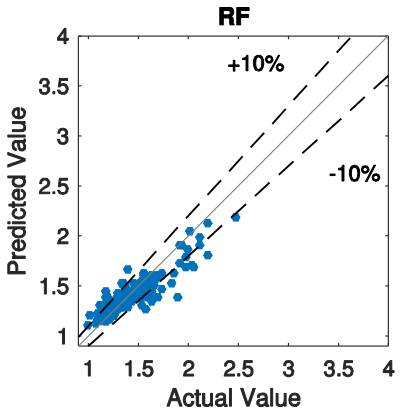


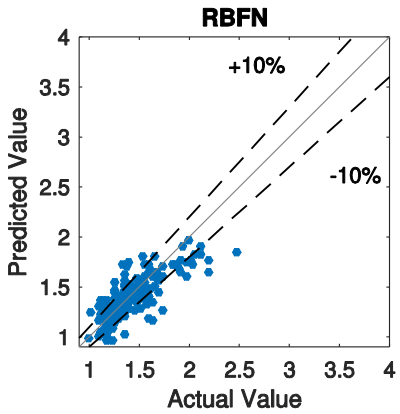


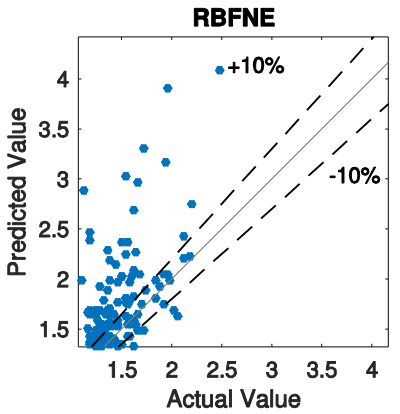


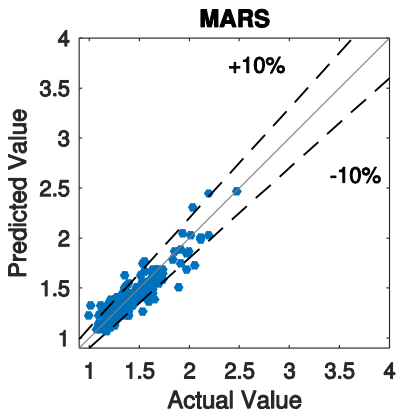
Cooling __ Sydney Rotated __ Airport terminal

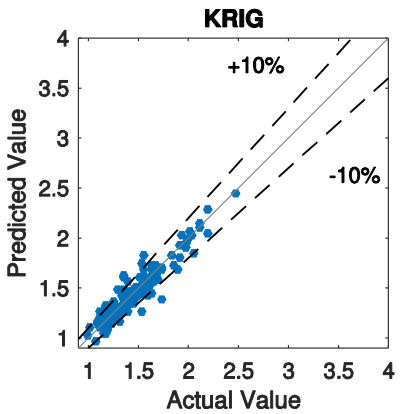




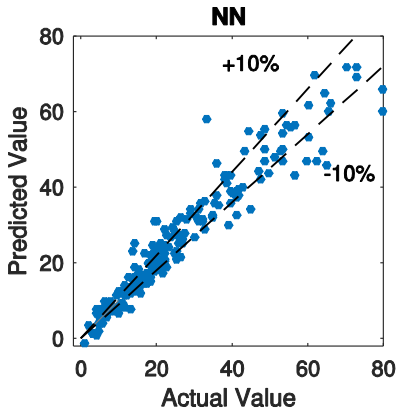


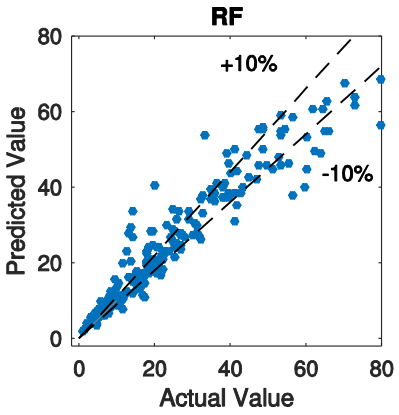


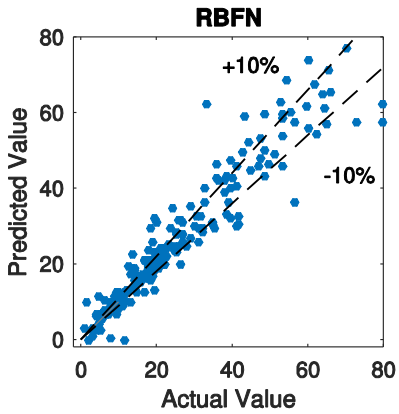


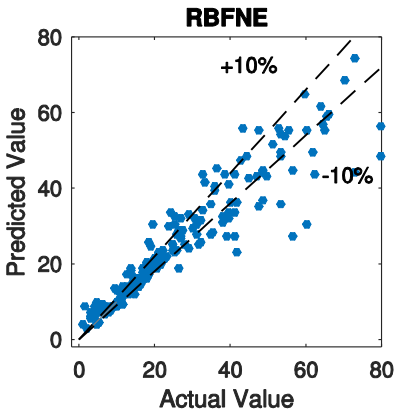


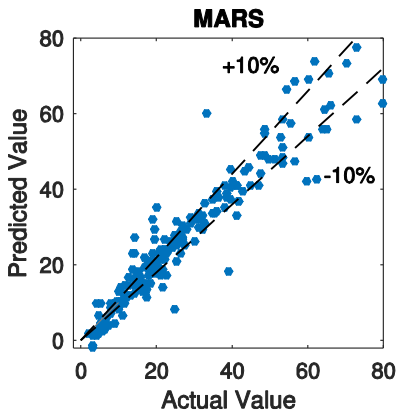
Heating __ Sydney Rotated __ Airport terminal

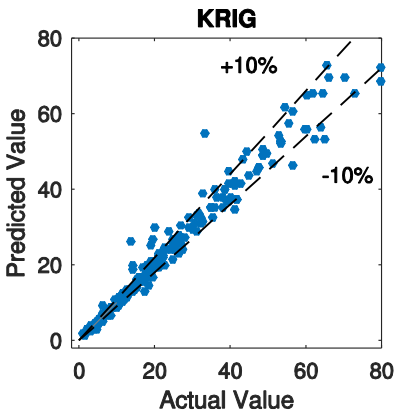












Lighting __ Sydney Rotated __ Airport terminal