Structural Health Monitoring:
A machine learning approach

Manuel Lobo Fernandes de Castro Mota

Dissertação de Mestrado

Orientador na FEUP: Prof. Vera Miguéis

Mestrado Integrado em Engenharia Mecânica

2021-06-28

“Scientists study the world as it is, engineers create the world that never has been.”
- Theodore von Karman

Aos Meus. E a ti, Tio Nuno.

Abstract

In order to meet the needs of today's society and the constant economic development, the number of civil infrastructures, such as bridges, tunnels, dams, and buildings, used by people in their daily lives, is growing considerably. Therefore, for efficient management of the built patrimony, Structural Health Monitoring (SHM) tools have gained momentum due to the automation of the monitoring and diagnosis process through the collection of data from structures' sensors.

In this context, this work aims to build a SHM system that, thanks to artificial intelligence, 'learns' progressively the regular behaviour of a real structure, to the point of allowing making predictions. For this purpose, two different machine learning (ML) approaches were followed: a multi-output neural network (supervised algorithm) and an autoencoder (unsupervised algorithm). Both were supported by data collected by some of the temperature and force sensors of the stay-cables of the Corgo Bridge – the case study explored in this work – between January 2015 and January 2018.

The multi-output neural network model was built to predict the future behaviour of the structure, i.e., to predict simultaneously the values of the force in the instrumented stay-cables (dependent variables), and to detect deviations in the presence of damage. The temperatures and two temporal indices, used to handle the trend of the time series, were used as explanatory variables. In contrast, as the output of the autoencoder is the reproduction of its inputs, the autoencoder model was only trained with the recorded stay-cables' forces. In the case of damage in the structure, the error associated to the inputs' reconstruction is higher than the error associated to an undamaged state. The detection of anomalies is achieved by detecting this deviation in the reconstruction error.

The results obtained from the multi-output neural network and autoencoder modelling enabled to realize the potential of these models in monitoring the behaviour of a structure, more specifically in anomaly detection.

Summing up, it was concluded that the neural network algorithm and the autoencoder algorithm may be used to support analytical methods to prevent situations caused by damage.

Resumo

Hoje em dia, em linha com o constante desenvolvimento económico, estruturas, pontes, túneis, barragens e edifícios constroem-se cada vez mais rápido e cada vez em maior número em resposta às necessidades da sociedade atual. Portanto, para uma gestão eficiente do património construído, as ferramentas de *Structural Health Monitoring* (SHM) têm ganho cada vez mais relevância devido à sua capacidade de automatizar o processo de monitorização e diagnóstico através da recolha de dados dos sensores das estruturas.

Neste contexto, este trabalho visa construir um sistema SHM que, graças à inteligência artificial, "aprende" progressivamente o comportamento regular de uma estrutura real, permitindo fazer previsões do seu comportamento. Para este efeito, foram seguidas duas abordagens de *machine learning* (ML): uma recorrendo a uma *multi-output neural network* (algoritmo supervisionado) e outra utilizando um modelo de *autoencoder* (algoritmo não supervisionado). Ambos os modelos foram basearam-se nos dados recolhidos por alguns dos sensores de temperatura e força dos tirantes da Ponte da Corgo - o caso de estudo explorado neste trabalho - entre janeiro de 2015 e janeiro de 2018.

O modelo da *multi-output neural network* foi construído com o objetivo de prever de forma simultânea o futuro comportamento da estrutura, isto é, o valor da força nos tirantes instrumentados (variáveis dependentes), e detetar desvios aquando da presença de dano. Foram utilizadas como variáveis explicativas as temperaturas e dois índices temporais de forma a ser possível lidar com a tendência das series temporais. Já o modelo criado com *autoencoder*, foi treinado apenas com os valores das forças registadas nos tirantes uma vez que o output do *autoencoder* consiste na reprodução dos seus inputs. Na presença de dano na estrutura, o erro associado a essa reconstrução é superior ao registado num estado não danificado. A deteção de anomalias é conseguida através da deteção desse desvio no erro de reconstrução.

Os resultados obtidos a partir da *multi-output neural network* e do *autoencoder* permitiram perceber o potencial destes modelos na monitorização do comportamento de uma estrutura, mais especificamente na deteção de anomalias podendo-se então concluir que a *multi-output neuralnetwork* e o *autoencoder* são algoritmos que podem ser utilizados para apoiar métodos analíticos para prevenção de situações causadas por danos.

Acknowledgments

This work was carried out under the project POCI-01-0145-FEDER-031355 - PTDC/ECI-EGC/31355/2017 "S4Bridges - A smart approach for the maintenance of existing bridges", co-financed by the Foundation for Science and Technology IP (PIDDAC) and the European Structural and Investment Funds (FEEI), through COMPETE2020 - Programa Operacional Competitividade e Internacionalização (POCI).

First and foremost, I would like to express my gratitude to my supervisor Vera Migueís for the opportunity to integrate this project to develop my master's thesis. I reiterate my gratitude for all the guidance, kindness, motivation, comprehension, and availability since the first day.

I extend my appreciation to all the S4Bridges team for the kind welcoming and for all their contribution to the project's development.

I would also like to thank all the teachers and technicians I had the opportunity to interact with during these five years of MIEM. Indeed, the rigor and quality of their teaching will make me a better engineer.

I also would like to thank my friends for their friendship, motivation, and unconditional support along the way.

To my family, my parents, my siblings, nephews, and niece, for supporting me in all kinds of ways, the biggest thank you.

The final words go to Beatriz for being with me since the beginning of this adventure at FEUP, for the love, the patience, and everything she shared with me.

Contents

List of Abbreviations

- ANN - Artificial Neural Networks
- KDD – Knowledge Discovery in Databases
- ML – Machine Learning
- MSE – Mean Squared Error
- SHM – Structural Health Monitoring

List of Figures

List of Tables

1 Introduction

1.1 Project context and motivation

Nowadays, in order to meet the needs of today's society and the constant economic development, the civil infrastructures, such as bridges, tunnels, dams, and buildings, used by people in their daily lives, are growing faster and more extensive.

Besides operational functionality, there is inherent to this growth the need and the importance of guaranteeing public safety. Therefore, it is essential to minimise any type of risk, avoid potential accidents, and save victims that may arise from eventual tragedies. As these infrastructures are subject to irregular and adverse (operational and environmental) external conditions, sometimes extreme ones, as in the case of being affected by natural catastrophes (Rodrigues et al., 2021), rigorous and constant revision and maintenance become indispensable for faster detection of damage or degradation, avoiding severe and irreversible consequences. Only in this way, it is possible to understand the structure's condition and state to guarantee the safety, operability, and prosperity of the infrastructures.

In this context, monitoring the integrity of structures has valuable potential for efficient management of the built patrimony, particularly because of the extensive park of infrastructures. Moreover, this contributes to avoid the time-consuming, dangerous and expensive traditional visual inspection needed to evaluate the structural condition (Zhang and Burton, 2019). Furthermore, visual inspection is not capable of tracking condition changes in real-time (Rodrigues et al., 2021). Hence, systems known as Structural Health Monitoring (SHM) have emerged and gained increasing attention (Xu and Xia, 2012; Liu and El-Gohary, 2021; Spencer et al., 2019). A SHM system is a tool that allows to predict the future behaviour of the structure, optimize inspections, minimize the risk of damage or structural collapse, and extend the service life of the structure (Leon 2021).

These intelligent systems have gained momentum due to the automation of the monitoring and diagnosis process through the collection of data from sensors capturing displacements, vibrations, strengths, and temperatures. They also take advantage of the data to infer the structure's condition, and trigger alarms when structural damage is detected.

Although a lot of data have already been collected, few stakeholders are already using SHM to support decision-making in the maintenance of a structure (Tomé, 2019).

1.2 Project objectives

This work is carried out within the scope of the activity *Damage detection methods* integrated into the *S4Bridges* project that consists of a smart approach for the maintenance of existing bridges. This activity aims at the development, implementation, and validation of damage detection and localisation algorithms. *S4Bridges* project started in 2018 and has the collaboration of the Instituto Superior de Engenharia, Faculdade de Engenharia da Universidade do Porto and Construct - Instituto de I&D em Estruturas e Construções.

The project explores analytical models and develops an innovative structural monitoring and predictive analysis system from a machine learning (ML) perspective. Thus, the goal is to build a SHM system that, thanks to artificial intelligence, 'learns' progressively the regular behaviour of a structure, to the point of allowing making predictions.

More specifically, the goal is to model a real structure's natural and regular behaviour, specifically using machine learning regression algorithms such as multi-output neural networks and autoencoders. In addition, anomaly detection tools will be used to detect significant deviations between the predicted normal behaviour and the monitored behaviour. It is expected that in the case a damage occurs, the prediction provided by the model does not coincide with the observed values.

The case study is the Corgo bridge, a part of the Transmontana Highway - A4 located in the North of Portugal. Data is collected from the sensors distributed across the various parts of the bridge. Through this data, models will be created, and it will be possible to trigger alarms when damage occurs and avoid, as far as possible, false alarms.

All the tasks needed to achieve the mentioned objective are done in *R Studio* software.

1.3 Methodology

To achieve the aforementioned goal, firstly, it was necessary to analyse and understand all the data available for the construction of the models. Thus, in an early stage, the treatment and processing were essential to prepare data into a ready-to-use state to feed the models. After preparing the data set, the construction of the model began. This modelling involved an in-depth study of machine learning tools. The chosen algorithms were a neural network and an autoencoder. Finally, performance metrics as R^2, MSE, RMSE, MAPE were used to evaluate both built models.

1.4 Structure

This thesis is divided into five further chapters. In the next one, presented as "Background Knowledge", fundamental concepts related to data analytics and machine learning tools are introduced, focusing on the techniques applied in the project's development. Chapter 3, named "Literature Review", presents a state of the art of the various possible alternatives in Structure Health Monitoring and damage detection, focusing on machine learning approaches.

After the theoretical introduction, the "Data and Methodology" chapter presents the case study of this project, the built dataset, and the developed methodology. Then, in the "Results" chapter, the obtained results for each approach are introduced, evaluated, and compared.

Finally, the "Conclusions and Future work" chapter summarizes all the steps of the project, analysing the results obtained with both approaches. Also, some suggestions for future work are provided.

Additional data, as charts, complementary information, as well as, R scripts that were crucial for the development of this project, can be found in “Appendixes”.

2 Background Knowledge

This section presents the main concepts necessary and inherent to the development of the project. Firstly, the concept and procedures of Knowledge Discovery are explained in general terms, as well as its goals. The basic and essential details of machine learning tools are then covered, focusing on Artificial Neural Networks and Autoencoders. The last subsection covers the Structural Health monitoring theme.

2.1 Knowledge Discovery

With the advances in technology, data collection and storage have enabled organizations to accumulate large amounts of data. Nevertheless, extracting helpful information has proven extremely challenging. In this way, new methods to analyse data have emerged since traditional data analysis techniques, which are only possible to apply to relatively small data sets, have often encountered practical difficulties in meeting the challenges posed by new data sets.

First of all, the overall process of converting raw data into useful information or knowledge in the form of rules, patterns, classifications, or clusters is iterative and interactive, and it is called Knowledge discovery in databases (KDD) (Cios et al., 2007). It consists of multiple steps that are executed in sequence, as shown in Diagram 2-1.

Diagram 2-1 – Steps of Knowledge discovery in databases

According to Tan et al. (2014), the input data, i.e., the raw data (database, images, video, semi-structured data), can be stored in various formats such as spreadsheets and flat files. These inputs are transformed into an appropriate format for analysis in the most time-consuming and laborious step of the KDD process, known as the data Processing phase, whose goal is to make the data more suitable for the next task. The tasks involved in this process include:

- **Data cleaning:** to remove noise and duplicate observations, and to handle missing data;
- **Feature selection**: to select features and records that are relevant to the data mining task at hand, reducing the number of resources required to describe a large dataset;
- **Data reduction:** to reduce, if possible, the number of variables under study;
- **Data subsetting**: to extract an intact smaller-sized set of data from the original dataset.

As it is illustrated in the diagram below, Data Mining is an integral part of the KDD process and precisely comes after the data preprocessing phase. Their techniques have deployed to bring more efficient and scalable tools that could handle diverse data types. Thus, data mining tools

such as machine learning algorithms aim to search for valuable and novel patterns that might otherwise remain unknown in data repositories and to predict the outcome of a future observation. After that, the postprocessing step ensures that only useful and validated results are incorporated into the decision support systems. In other words, it is the step of integrating data mining results into these systems. During this step, statistical measures or hypothesis testing methods can be applied to eliminate spurious data mining results (Tan et al., 2014). Finally, the 'real' information is gotten.

All the steps of the KDD process mentioned constitute a powerful tool to assist the data mining process, either by preparing the data, which significantly facilitates data mining, either by interpreting and representing intuitively the knowledge scoured (Carvalho, 2020).

In summary, all the mentioned phases of the KDD process are crucial to attain the most valuable and relevant knowledge from a database.

2.2 Machine Learning concepts

In the age of Big Data, the need for machine learning methods is rapidly growing, as increasing amounts of data are available due to significant information technologies progress. Therefore, machine learning is a technology that blends traditional data analysis methods with sophisticated algorithms for processing large volumes of data. It can be applied to support a wide range of application fields such as Engineering, Medicine, Science, and Business.

2.2.1 Overview of Machine Learning and Learning Methods

As Goodfellow et al. (2016) state, machine learning algorithms enable us to tackle tasks that are too difficult to solve with fixed programs written and designed by human beings. Using machine learning algorithms, the data is analysed by reading the data, extracting features from it, and identifying rules using that data using a machine such as a computer. In general, these analysis methods aim to elucidate the relationship between instances, as data points, in the given data or to predict specific properties of unknown data related to the given data (Baladram et al., 2020).

In 1997, Mitchell provided the following definition for a better explanation of what is a machine learning algorithm: "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*." A great variety of experiences *E*, tasks *T*, and performance measures *P* are possible.

Despite some exceptions, as the reinforcement learning deployed by Sutton and Barto (2014), most machine learning algorithms simply experience a unique dataset. As their names suggest, these algorithms are able to learn from data and can be broadly categorized as supervised or unsupervised, according to the kind of experience they are allowed to have during the learning methods (Goodfellow et al., 2016).

Accordingly to Baladram et al. (2020), **supervised learning:** attempts to determine a relationship or a function based on labelled training and uses the function to map new unlabelled data. In these methods, such as multilayer perceptron, logistic regression, decision tree, support vector machine (SVM), a model is developed from a training dataset where the input and output values are previously known. To have a good performance, i.e., creating a good model, supervised learning methods require an acceptable number of labelled records. It is important to emphasize that these methods uncover hidden patterns in unlabelled data.

Generally, a supervised learning method produces a predictor, which accepts input data and outputs a prediction result against the input data. In this process, there are included two main phases: learning and prediction phases. In the learning phase, the system reads the input and target vectors and improves the rules. This procedure is repeated until the quality of the rules becomes sufficiently good to achieve a good prediction performance for as long as possible. Finally, in the prediction phase, the properties or features of new data are predicted by the system (Baladram et al., 2020).

Generally, supervised learning techniques used to deal with two different types of tasks depending on the problems to be solved:

Classification: The goal of this kind of task is to specify which of k categories some input belongs to. Although there are some variants of the classification task, the learning algorithm is usually asked to produce a function $f: \mathbb{R}^n \rightarrow \{1, \dots, k\}$. The model to solve such tasks assigns an input described by vector x to a category identified by numeric code y when $y = f(x)$. There are other variants of the classification task, for instance, where the output of function f is a probability distribution over classes. Object recognition is another of the several application examples for which this algorithm is used.

If some of the inputs are missing, rather than providing a single classification function, the learning algorithm must learn a set of functions where each function corresponds to classifying x with a different subset of its inputs missing. It is called *classification with missing inputs*, a more challenging machine learning task. To define such a set of functions is necessary to learn a probability distribution over all the relevant variables and, after that, the classification task should be solved by marginalizing the missing variables. Considering n input variables, it is possible to attain all 2^n different classification functions needed for each possible set of missing inputs, but the system needs to learn only a single function describing the joint probability distribution.

Regression: It is similar to classification, excepting the output format. In this kind of task, the computer program is asked to predict a numerical value given some input. To make a prediction, the learning algorithm is asked to output a function $f: \mathbb{R}^n \rightarrow \mathbb{R}$. Regression analysis convolves around simple algorithms, which are often used in finance, investing. Basically, it seeks to build a mathematical equation that defines y as a function of the x variables.

This equation can be used to predict the outcome y based on new values of the predictor variables x. In Figure 2-1 is illustrated a simple classification problem through which it is possible to better understand the mentioned procedure. In Table 2.1 are listed four given instances, each consisting of a two-dimensional input vector and one-dimensional target vector.

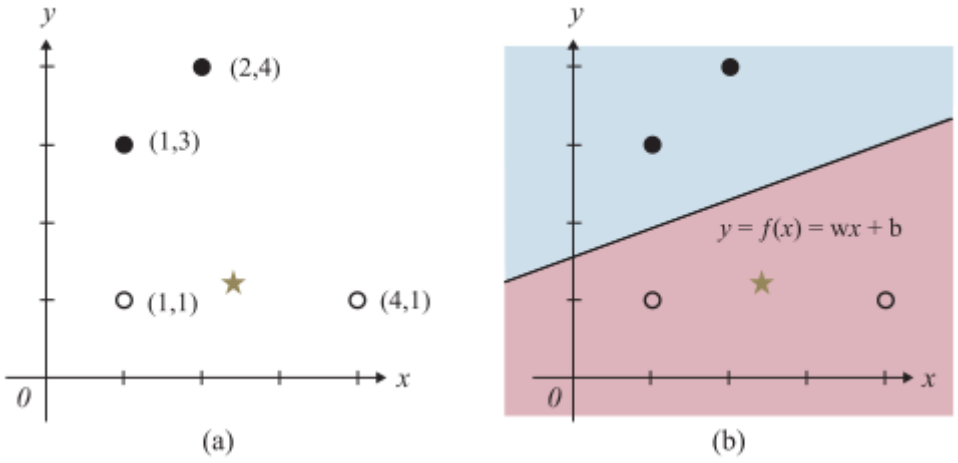


Figure 2-1 - Example of a supervised learning method. (a) Classes indicating open and filled circles are 0 and 1, respectively. The star represents the new data to be predicted. (b) Classification boundary after the learning process. Source: (Baladram, 2020)

s

Table 2.1 - Input vectors and corresponding target vectors. Source: (Baladram, 2020)

Input Vectors	Target Vectors
(1,1)	(0)
(1,3)	(1)
(2,4)	(1)
(4,1)	(0)

The input vectors are plotted in the xy-coordinate plane and are separated by their target vectors. The goal is to check whether a new input vector, which is represented by a star, has a value of 0 or 1. In this case, a machine learning method attempts to determine a classification curve. Thus, to solve the presented problem, it should be defined a classification curve, represented in Figure 2-1 (b), $y = f(x) = wx + b$. The w and b are called the parameters of the equation, and their optimal values might be obtained using several machine learning techniques. The main difference between these techniques is based on the methods used to determine the classification curves (Baladram et al., 2020).

According to Alpaydin (2014), traditionally, there are two types of problems to be solved by supervised learning methods that can be categorized into: classification and regression problems. While with the regression problems a numerical value is predicted, with classification problems, a class (or a flag) of the input data is predicted. Both are presented in more detail in the next section alongside some other machine learning tasks.

On the other hand**, unsupervised learning** consists of identifying patterns on relationships data points themselves. There are no output variables to be predicted, i.e., only inputs are previously known. These algorithms used to be utilized when learning the entire probability distribution that generated a dataset, whether explicitly, as in density estimation, or implicitly, for synthesis

or denoising tasks (Goodfellow et al., 2016). Hierarchical clustering, k-means clustering, autoencoders, principal component analysis are other unsupervised learning methods.

In essence, unsupervised learning involves observing several examples of a random vector x and attempting to implicitly or explicitly learn the probability distribution $p(x)$, or interesting properties of that distribution. However, supervised learning methods involve observing several examples of a random vector x and an associated value or vector, then learning to predict y from x, usually by estimating $p(y \mid x)$.

In summary, while the term supervised learning came from the view of the target y being provided by an instructor or teacher who shows what to do to the machine learning system, an unsupervised learning algorithm must learn to make sense of the data without that guide.

2.2.2 Machine learning performance measure

The evaluation of a machine learning algorithm is made with a quantitative measure of its performance, specific to the task being carried out by the system. Regarding classification, it is measured through the model's accuracy or error rate. While accuracy is the proportion of examples for which the model produces the correct output, the error rate is the proportion for which the model produces incorrect output.

The evaluation metrics for regression models are quite different from the metrics used to evaluate classification models because the regression predicts in a continuous range instead of a discrete number of classes (Jordan, 2017). All the following regression metrics are mathematical formulas that compare the model predictions with actual values, outputting a metric representing the model's suitability for predicting the dependant variable. In these formulas, y_{true} represents the actual value and y_{pred} represents the predicted observation i.

Mean squared error (MSE): represents the average of squared differences between the predicted output $y_{pred\,i}$ and the actual output $y_{true\,i}$. It is used because it is agnostic to whether the prediction is too high or too low.

$$MSE = \frac{1}{n_{samples}} \sum (y_{true\,i} - y_{pred\,i})^2 \tag{2.1}$$

Root Mean Square Error (RMSE) is used more commonly than MSE because sometimes MSE values can be too big to compare easily. Also, MSE is calculated by the square of error, and thus square root brings it back to the same level of prediction error and makes it easier for interpretation (Songhao, 2020).

$$RMSE = \sqrt{MSE} \tag{2.2}$$

R^2 coefficient represents the proportion of variance in the outcome that the model is capable of predicting based on its features.

$$R^2 = 1 - \frac{\sum (y_{true\,i} - y_{pred\,i})^2}{\sum (y_{true\,i} - \bar{y})^2} \tag{2.3}$$

Mean Absolute Percentage Error (MAPE) represents the error as a percentage of the real value.

$$MAPE = \frac{1}{n}\sum i = \ln \left| \frac{y_{true\ i} - y_{pred\ i}}{y_{true\ i}} \right| \quad (2.4)$$

When a machine learning model is built, it is fundamental to understand how good its performance is when the data has not been seen before. Thus, to measure that performance, a test set - a collection of data separated from the data used for training the machine learning model – is used.

Although the choice of performance measure might seem straightforward, it is not easy to choose the one that corresponds well to the desired behaviour of the system (Goodfellow et al., 2016).

2.2.3 Overfitting and Underfitting

The most challenging issue concerning the use of machine learning algorithms is to get a good performance not only on the data on which the model was trained but on new data. The errors committed by a predicter can be grouped into training errors, which refer to the misprediction of training records and generalization errors (Carvalho, 2020), which refer to the error of prediction in unseen data.

The factors determining how well a machine learning algorithm will perform are:

- Its ability to make the training error small;
- Its ability to make the gap between training and test error small.

According to Daneshtalab and Modarressi (2020), two factors correspond to the two central challenges in machine learning: underfitting and overfitting. When a predictor has a low training error but loses performance for non-training data getting high generalization error, we are facing the phenomenon of model overfitting. Tan et al. (2014) pointed out that some causes for this event might be noise in training data and lack of representative samples. On the other hand, when the model is not able to obtain a sufficiently low error value on the training set we are facing model underfitting.

2.3 Artificial Neural Networks

Artificial neural networks (ANN) constitute a group of machine learning algorithms.

According to Daneshtalab and Modarressi (2020), the study of ANNs was inspired the biological neural systems in order to provide solutions for real-world problems in the same way as the human brain operates, i.e., as the natural neural networks works. Machine learning method mimics the behaviours of the brain and nervous system.

A natural neural network structure, i.e., a neuron structure, includes axons, dendrites, and synapses. (Gurney, 1997) as shown in Figure 2-2.

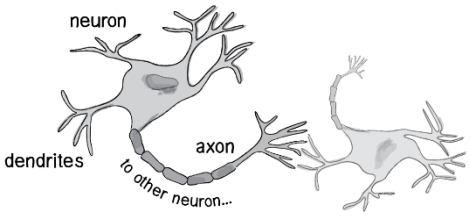


Figure 2-2 - Structure of a neuron. Source: (Shiffman, 2012)

The communication between neurons includes inputting data (signals), activating neural cells and synapses (receivers of the signals), and outputting signals. In other words, Dendrites receive input signals and, based on those inputs, fire an output signal via an axon (Shiffman, 2012).

2.3.1 Perceptron

A perceptron is a simple structure network without any hidden layers. It has only one layer, and that layer has only one neuron. This neural network can only converge to an optimal value if the input and target data can be linearly separated. On the other hand, it has several defects, such as the impossibility of learning nonlinear relationships in the given data, which is a big problem because most real-world data cannot be linearly separated. In addition, perceptron is not robust against noise in the data and requires a considerable time for convergence.

2.3.2 Multilayer Perceptron

In contrast with the simpler model mentioned above, the multilayer perceptron (MLP), represented in Figure 2-3, can learn nonlinear relationships in the given data (Baladram et al., 2020). Thus, when the class patterns are not linearly separable it is required adding one or two hidden layers to the neural network.

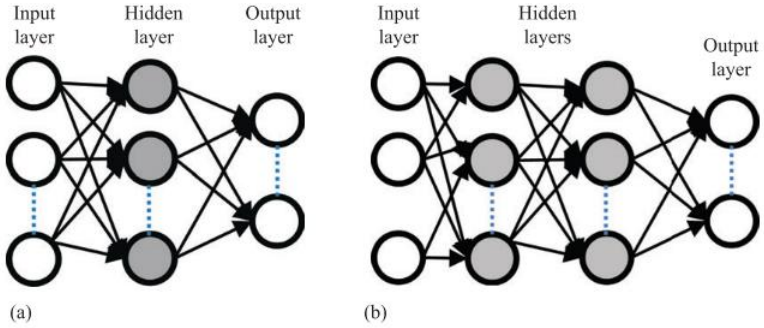


Figure 2-3 - MLP neural networks. (a) one hidden layer. (b) two hidden layers.
Source: (Daneshtalab, 2020)

2.3.3 Simple, multiple and extended regressions

To understand the principle of a neural network is important to know in what simple, multiple and extended regression consists because a neural network has almost the same structure as the extended regression method.

Regarding the simple regression, it can be observed as a type of machine learning method. Considering a one-dimensional input vector x and one-dimensional target vector y, whose values must be fit on an optimal line to the data. Further, letting y be the output prediction value of the novel input vector.

With a simple regression analysis, according to the given data the following equation is obtained:

$$y = f(x) = wx + b \tag{2.5}$$

If the number of variables is greater than two, the regression method is called a multiple regression method. This method is basically an extension of the simple regression method. As an example, considering $x = (x_1, x_2)^T$ as the input vector and y as a target vector. The goal is to determine an optimal regression plane to predict the value y of a novel input vector. In this case, the equation of the multiple regression method is expressed as follows:

$$\begin{aligned} y &= f(x_1, x_2) \\ &= w_1 x_1 + w_2 x_2 + b \\ &= (w_1, w_2)(x_1, x_2)^T \\ &= \boldsymbol{w}^T \boldsymbol{x} + b \\ &= f(x) = \end{aligned}$$

Where:

$\boldsymbol{w} = (w_1, w_2)^T$ – weight parameters

$\boldsymbol{b}$ - bias term

In the multiple regression analysis, it is attempted to determine optimal values of w and b by repeatedly reading the input data and learning the features of the given data. The difference between simple regression and multiple regression is only the number of explanatory variables and their corresponding weight parameters.

Generalizing, the multiple regression method into any arbitrary number of dimensions the regression equation can be expressed as follows:

$$y = f(x) = \boldsymbol{w}^T(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b \tag{2.6}$$

Where:

$\boldsymbol{W}$– m x n matrix

$\boldsymbol{b}$ – m x 1 matrix (m-dimensioned vector)

$\boldsymbol{w}, b$ - m x 1 and 1 x 1 parameter metrics, respectively.

Regardless of the number of parameters, the dimensions of the input and output vectors do not change as that of the multiple regression. The difference between multiple regression and this extended regression is only in the number of parameters.

2.3.4 Principle of neural network

The unique difference between a neural network and the extended regression method is that the neural network introduces an "activation function''ϕ to the extended regression. Thus, the

equation of the regression equation in the neural network is expressed by the following equation:

$$y = f(x) = w^T \phi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b \tag{2.7}$$

Baladram et al. (2020) mentioned that simple and multiple extended regression methods cannot separate nonlinear data. Therefore, activation functions, one of the most important components of neural networks, are needed for neural networks to mimic nonlinear functions to respond to real-world data. There are several known activation functions, such as the sigmoid function

$$\phi(u) = \frac{1}{1 + e^{-u}} \tag{2.8}$$

And the ReLu function

$$\phi(u) = \max(0, u) \tag{2.9}$$

The activation function works as an activator of neural cells, and this function decides whether the previous signals should be communicated to the subsequent neurons or not.

Figure 2-4 represents a conceptual image of the MLP, which illustrates the process of a feedforward neural network. As can be seen, the represented MLP has three layers: the input, middle, and output layers. The neurons are represented as circles, and the connecting lines between the neurons are weight parameters W that must be optimized.

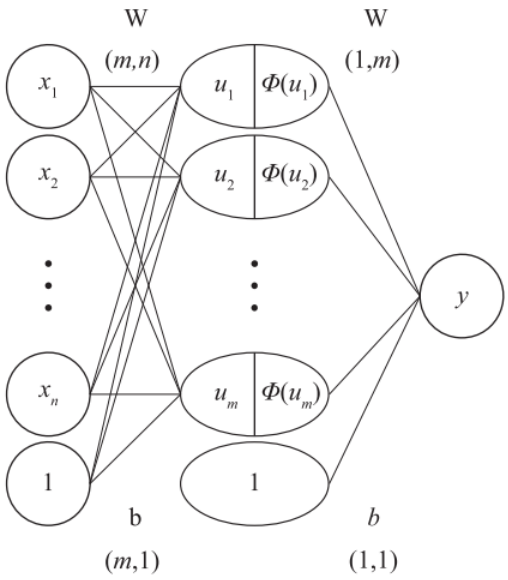


Figure 2-4 - Multilayer perceptron (MLP). Source: (Baladram, 2020)

Analysing Figure 2-4, it is clear that the process consists of: firstly, the elements of input vectors, x_i, are transformed to some element u_j. After that, $u_j$is activated by an activation

function ϕ and, in the end, the activated elements $\phi(u_j)$ are transformed to the output scalar value y.

2.3.5 Objective function

In the learning process, when training data is fed into the neural network model, the network reads the input data repeatedly and attempts to determine the optimal value of parameters in a trial-and-error manner.

To optimize the networks' parameters, namely the weights, an objective function, also known as loss function E, is required. The mean square error (MSE), which is primarily used for regression problems, is used as a loss function, and its function is expressed as follows:

$$E(\theta) = \frac{1}{2}\sum_{i=1}^{N} \|t_i - y_i\|^2 \tag{2.10}$$

Where:

$\boldsymbol{\theta}$ - the parameters to be optimized.

$\boldsymbol{N}$ - the number of instances.

$\boldsymbol{t_i}$ - the *i-th* output target vector.

$\boldsymbol{y_i}$ - the *i-th* output vector from the predictor.

The neural networks have multiple parameters, whose gradient, $\nabla E(\theta)$, must be calculated. For that, it used to be utilized the backpropagation method. Then, for parameter optimization, gradient descent methods can be applied. The gradient descent methods consist of a relatively simple algorithm, and it has a straightforward implementation. Also, it has global convergence and a good complexity of computational space. The optimal values are obtained by iterating and updating randomly chosen initial values, i.e., initial parameters to the optimal values, gradually. The backpropagation and gradient descent methods are explained in more detail in Baladram et al. (2020).

In conclusion, neural networks have taken a significant role in the recent boom in artificial intelligence as one of the most popular machine learning methods. There are various types of neural networks, including Multilayer perceptron, Convolutional neural networks applied primarily in image processing studies, and Recurrent neural networks, one of the most promising architectures for achieving artificial general intelligence.

2.4 Autoencoders

An important part of the historical landscape of neural networks is the development of the autoencoder (Ballard, 1987). This neural network was developed in the late 1980s and since then has been applied to anomaly detection, data denoising (ex: images, audio), information retrieval, and image inpainting.

It is a neural network that is part of the family representation learning methods capable of automatically learning features from unlabeled data. In other words, autoencoders are an unsupervised learning technique used for the task of representation learning (Jordan, 2018). This means that an autoencoder neural network is trained to learn features, i.e., representations of the input data (Boehmke and Greenwell, 2021).

As shown in Figure 2-5, in the autoencoder model, the network includes two sequenced parts: encoding and decoding. These two parts are neural networks called encoder and decoder, respectively. While the decoding part performs the dimension reduction, the decoding part is added to reconstruct the former inserted patterns and ensure that no information from input patterns is lost due to encoding (Daneshtalab and Modarressi, 2020).

The encoder and decoder architecture can be customized. It is possible to have a shallow design involving just input and output layers or a deeper structure involving a higher number of layers and neurons per layer and a larger code size. The architecture of the decoder used to be the mirror image of the architecture of the encoder (Hugo et al., 2020).

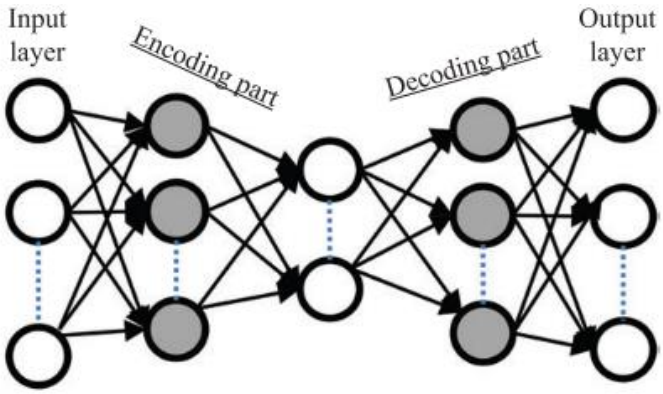


Figure 2-5 - Diagram of an autoencoder network. Source: (Daneshtalab, 2020)

2.4.1 Method description

As Hugo et al., (2020) state, an autoencoder is a particular case of feedforward neural networks and, as such, can be trained using the same techniques, including backpropagation and the gradient descent method.

The goal of this network is to generate an output that is identical to the input. It can be achieved using a specific loss function and minimizing it during the training of the model. This loss function, known as reconstruction loss, is usually the mean squared error between the output $x'$and the original input x, and it is expressed as follows:

$$L(x, x') = \|x - x'\|^2 \tag{2.11}$$

Despite attempting to generate a copy of the input may sound useless, in representation learning, the interest is not in the output of the decoder, but it is in the generation of a copy of the input data by training the autoencoder. The result is a latent code with useful properties, which, ideally, has less correlated/ redundant features and/or lower dimensionality than the input data. These are crucial characteristics to avoid the waste of processing time and the overfitting of machine learning models.

According to Jordan (2018), it is important to emphasize that this compression and subsequent reconstruction would be a complicated task if the input features were each independent of one another. However, it is possible to handle the case when some sort of structure exists in the data (i.e., correlations among different dimensions of the input patterns). Such structure can be learned and consequently leveraged thanks to the network's bottleneck, which constrains the amount of information that can traverse the full network, forcing a learned compression of the input data. Figure 2-6 shows an example of a bottleneck in an autoencoder structure.

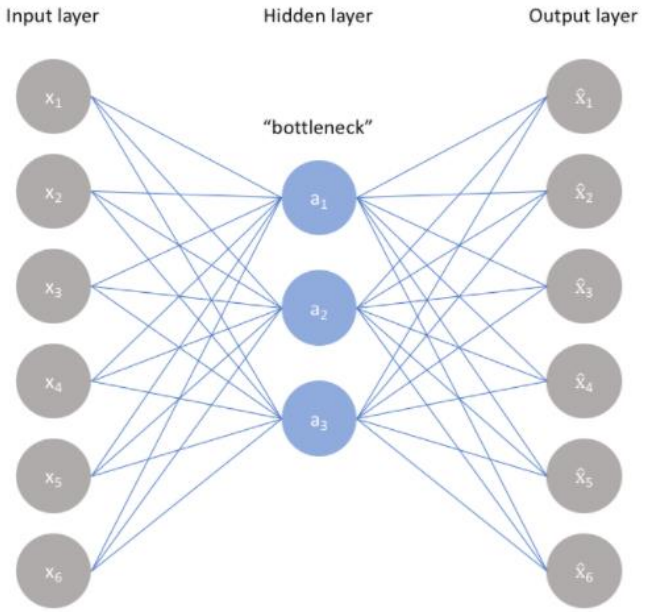


Figure 2-6 - Autoencoder – bottleneck. Source: (Jordan, 2018)

2.5 Structural Health Monitoring

Both aforementioned techniques, the autoencoders and the Artificial Neural networks, can be applied to the structural health monitoring field.

Bao et al. (2019) defines SHM as a multi-discipline field that involves automatic sensing, evaluation and warns about the structural condition in real-time. This is done by means of a large number of instruments and sensors followed by a diagnosis of structural health based on a large amount of data generated and collected every day. Carvalho (2020) states that SHM can also be seen as the application of new data analysis methods as machine learning algorithms to control the structure's health condition.

A SHM system should ensure constant monitoring and subsequent analysis of the structure's behaviour, detecting in advance abnormal changes that may indicate damage and degradation (Bao et al., 2019). These systems also enable real-time tracking of structural integrity to

guarantee the expected level of performance, safety, operability, and reduced maintenance and inspection costs for such structures (Carvalho, 2020). The recent development in sensors and communication technologies have contributed to a high rate of data collection. This development has made the sensors capable of monitoring environmental parameters, such as temperature, vibration, strain, flows, and displacements.

To build a SHM system, two main approaches used to be applied: physical and data-driven models. A physical-based model relies on the application of analytical methods directly relating features to physical parameters and implies a previous deep understanding of the structure. On the other hand, a data-driven model deals with statistical features, comparing them to evaluate different structure states. With this kind of model, it becomes possible to consider environmental effects such as those referred above.

Diagram 2-2 shows the steps of damage detection process of an SHM system.

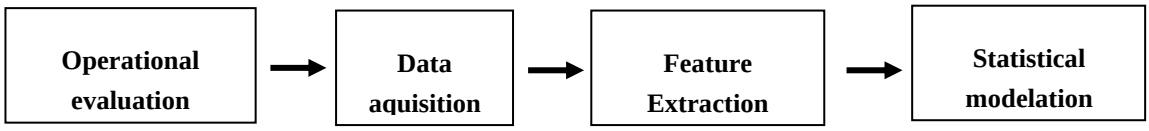


Diagram 2-2. The steps of a SHM system.

Firstly, it is essential to understand the structure's environmental and operational conditions to describe the purpose of the SHM. Then, data acquisition describes the steps followed for data collection and data processing. The third step represented in the above diagram consists in determining the significant signals for damage detection. Lastly, the model which reflects the health condition of such structure is created.

Nowadays, complex engineering structures in civil, mechanical engineering, and aerospace (Rodrigues et al., 2021) have been equipped with SHM systems seeking to understand the behaviour and an automatic real-time warning (Bao et al., 2019).

3 Literature Review

This chapter presents the related work and highlights the techniques used for structural behaviour monitoring and anomaly detection in structures.

Until the present time, several studies have been developed in the field of monitoring the structures' behaviour. As the data have recently become essential in society as their availability and effectiveness create value, to sense and understand the behaviour of the structures, data also becomes a core part of the SHM field.

First, in these studies, the used data might be recorded from different sources. Concerning its sources, the following approaches can be applied.

- **Virtual data approach**: Data is sourced from a virtual structure for a damaged and undamaged state. As the name indicates, no real-life risk is taken, and the damage insertion is straightforward. On the other hand, some real-world variables may not be covered, making these models less trustworthy.
- **Virtual damage approach**: Data is recorded from a real structure in an undamaged state, and virtual data generation for the damage state.
- **Physical data approach**: Data is generated from a real-life structure for an undamaged and damaged state.

The first mentioned approach has been applied to SHM systems since 1998: (Dunia and Qin, 1998; Kim et al., 2003; Yan et al., 2005; Slišković et al., 2012; Kromanis and Kripakaran, 2013).

Regarding the virtual damage approach, Wipf et al., 2007; Tomé et al., 2019; Tomé et al., 2020 are examples of studies that use a real bridge to record data in an undamaged state This approach enables a close view of the reality as the built models are capable of incorporating the real behaviour of the bridge in question. For example, in 2019, Nguyen et al. used transmissibility functions calculated from the simulation of the vibration responses at some points of a real bridge under a moving truck as input data to train artificial neural networks (ANN). After a successful training, the networks could quantify and locate the damage.

Concerning physical damage approaches, although it is not frequent to be possible to gather data of a damaged bridge in a controlled way, there are some studies in the literature that use data from the damaged and undamaged states recorded from a real-life infrastructure (e.g. Cross et al., 2011; Reynders et al., 2014; Farreras-Alcover et al., 2015; Da Silva, 2017). This approach also includes the cases in which small scale model is used to represent the undamaged and damaged states. Examples of the application of physical damage approaches include Kesavan et al., 2008; Tibaduiza et al., 2011; Wipf et al., 2007; Rosales and Liyanapathirana, 2017; Rodrigues, 2020. In these studies, several models such as linear regression and principal

component analysis are used for SHM purposes. For example, PCA is useful when applied as a pattern recognition because it permits to compare the state of a current structure with a baseline in order to determine if exist some changes and, besides whether these changes can be considers as a damage or not (Tibaduiza et al., 2011). It can be also used for data cleansing and compression. Regarding Regression analysis, it is a tool used to relate the observed environmental and/or operational factors with the observed structural responses and/or features. (Tomé 2019)

Besides being classified on the basis of its data source, any approach can also be generally classified into two categories: model-based approach and data-driven approaches. The data-driven approaches are usually robust regardless of the complexity of the physical model used. These algorithms imply constructing a surrogate model from collected data that substitutes the high-fidelity model that is addressed to detect damage efficiently. The techniques used to support this approach can be broadly categorized as supervised or unsupervised. While the supervised, which is the most used in the literature regarding the available SHM systems, is applied when there is a dataset that represents the structure's health condition as a baseline, the unsupervised learning is used when such baseline is not known but it able to capture inherent patterns in data that might be essential to detect anomalies (Rodrigues, et al., 2021). As mentioned in chapter 2.2.1, there are two subgroups within supervised learning: Classification and regression methods. The most applied algorithms for classification purposes are Support Vector Machines, Artificial Neural Networks, and Random Forests SHM35. Regarding regression, the most used methods that have been developed are Linear Regression, Autoregressive models (Park and Inman, 2007), Artificial neural networks, Random Forests. These studies tend to apply control charts for damage detection.

Regarding data-driven approaches, they seek to build a reliable model of the system, usually a finite element model, including natural frequency and curvature. Sensor data is usually integrated into models or simulations for calibration or to identify physical characteristics and evaluate the conditions of a structure. When the goal is to evaluate a structure's condition, the damage detection performed by the model is done by perceiving difference comparing responses from the model and the actual structure. Chen et al. (2020) investigated these models, which are well-established according to Gui et al. (2017). On the other hand, Sun et al. state some limitations of model-based techniques as the prior knowledge needed on the real model for the estimated parameters to lie within acceptable confidence interval and the parameters estimated by the mathematical model may not be unique result in lack of physical interpretability. Besides this, Gopalakrishnan (2009) affirms that due to the high number of elements involved in generating a finite element model, achieving a good performance requires a massive computational effort. Azimi et al. (2020) also studied the traditional physics-based model and concluded that the minimum noise in the data is needed for such models, which is quite difficult considering the nature of structures and their environmental and operational conditions.

For any approach, continuous or discrete variables can be used to build the model depending on the nature of the algorithms used. While in case a small-scale structure is used to replicate the real structure, the most-reported continuous variables are strain and vibration, when following a physical damage approach with a real structure, the most used variables are temperature and vibration. Variables such as impedance (Park and Inman 2007), displacement (Posenato et al., 2008), and flow (Slišković et al., 2012) are also popular in the literature. The discrete variables, on the other hand, are less widely used, but, as an example, were used by

Alipour et al.' research (2017) that studied a bridge´s load capacity rating using variables such as climate zone, number of lanes of the structure, water's presence.

SHM studies can follow a local or a global approach (Rodrigues, 2020). As its name implies, the local approach focuses on specific parts of the structures, such as the cable bridges. First, the location is determined, and after that, the damage severity is quantified in specific parts of the structures, such as the cable bridges. On a global approach, data is collected and analysed to build a model that simulates the regular, i.e., undamaged condition of the structure.

As one of the main objectives of structure monitoring is the detection of damage and the SHM systems are operated in harsh and noisy environments, data with outliers, trends, saturation, and missing data are often observed. This can be a barrier for the automatic warning of SHM systems because it becomes difficult to distinguish which abnormal data are caused by damage of the instrumented structures or caused by out-of-order SHM systems (Bao et al., 2019). To overcome this issue, Yuen and Mu (2012), H. Liu, Shah, and Jiang (2004) investigated some outlier detection approaches. On the other hand, concerning abnormal data detection, the researches still insufficient (Peng et al., 2017; Chang et al., 2017).

The large variation in the features extracted from the immense SHM data can also become a barrier as it can cause conventional anomaly detection techniques to perform poorly. To overcome this, even with expertise, the parameter tuning associated with multiple data preprocessing techniques is still a challenge and makes the procedure inefficient and expensive. This way, to deal with the massive data collected by multiple sensors from an actual SHM system, smarter approaches are essential (Bao et al., 2019).

As anomaly detection focuses on a single type of anomaly, it often misses detecting other anomalies. This way, the deep learning (DL) – based approaches and machine learning (ML) have emerged as they might be promising alternatives to diagnose kinds of abnormal data through learning from big data containing abnormal data. In this context, imputation algorithms for addressing missing data or anomaly detection have also been investigated (Z. Chen et al., 2019; 2018). Also, Machine learning techniques, known as pattern-recognition methods, have been studied to provide advanced mathematical algorithms and frameworks that can help discover and model the performance and behaviour of a structure through deep mining of monitoring data. Thus, machine learning took an opportunity to establish novel machine learning paradigm for structural health diagnosis and prognosis theory in the SHM field (Bao and Li, 2020).

Some studies based on these algorithms have been done for anomaly detection (Chandola et al., 2009; Bao et al., 2019). The multilayer feedforward artificial neural networks (ANNs), also called multilayer perceptrons (MLPs), have been the most well-known and widely used ML algorithm for automated damage detection (Kudva et al., 1992; Based, 1992; Mehrjoo et al. 2008; Y. Y. Liu et al. 2011). Also, auto-associative networks, which are another type of Artificial Neural Network (ANN) architectures that have been used (Yasarer and Najjar, 2014). On the auto-associative networks, the knowledge to be extracted from a database is the identity function. In other words, this particular network is trained to reproduce its inputs and output(s). Since the network is optimized on inputs, as well as outputs, obtaining highly accurate results can be challenging. Yasarer and Najjar, (2014) investigated and concluded that the auto-associative network does not perform well on most of the databases in terms of error reduction but is capable of discovering the relationship between inputs and output. Even though the results from the auto-associative network are not comparable with those obtained via other approaches, they are still considerably promising.

Lastly, in the scope of the damage detection studies in the SHM field, autoencoders' implementation, one of the case studies of this project, has a minor representation of the overall studies. As it is not usually possible to collect data from damaged states of a civil structure using supervised algorithms, Rastin et al. (2021) propose a new unsupervised deep learning-based method for structural damage detection based on convolutional autoencoders (CAEs). The goal of the proposed method was to identify and quantify structural damage using a CAE network that employs raw vibration signals from the structure and is trained by the signals solely acquired from the healthy state of the structure. In that study, a CAE was chosen to take advantage of the high feature extraction capability of convolution layers and at the same time use the advantages of an autoencoder as an unsupervised algorithm that does not need data from damaged states in the training phase.

4 Data and Methodology

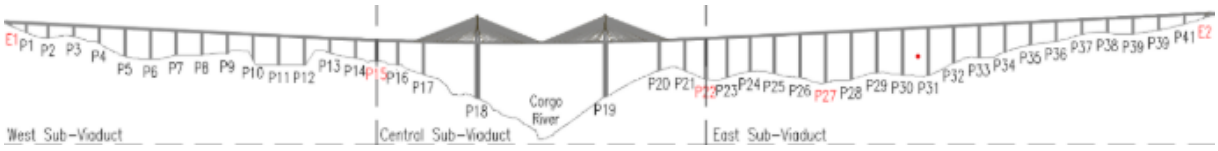
This chapter presents the carried-out procedures in the development of the project. Initially, the object of study of this project - the Corgo Bridge - is presented. Next, the data handling and processing that were done to the available data to prepare the dataset are presented. The procedures made to model the multi-output neural network and the autoencoder for monitoring the Corgo Bridge are also described in this chapter.

4.1 Corgo Bridge

In this subsection, the Corgo Bridge is presented, precisely the case study infrastructure explored in this study. It is a road artwork, part of the Transmontana Highway - A4, in Portugal, which does not present any structural damage. The methodology proposed in this study and introduced in this chapter is validated from the continuous monitoring system of this prestressed concrete cable-stayed bridge, more specifically in the structural monitoring system of the stay cables.

The Corgo Bridge, illustrated in Figure 4-1, opened to traffic in September 2013, is a prestressed concrete box-girder bridge with a length of 2790m, divided into three continuous sub-viaducts: the West Sub-Viaduct (WSV), the Central Sub-Viaduct (CSV), and the East Sub-Viaduct (ESV). In this work, the focus was on The Central Sub-Viaduct, a cable-stayed bridge with a 300m long span and two continuous end spans with 48m and 60m on each side. Its suspension system consists of a single central plane with four symmetric semi-fans of 22 stay-cables each.

(a)



(b)

Figure 4-1 - Corgo bridge. (a) general view. (b) side elevation view. Source: (Tomé, 2019)

A structural monitoring system is installed on the Corgo Bridge. It is made up of fibre optic and electric sensors, and it focuses on CSV due to its high structural complexity.

4.2 Data

In the present work, from the data of all the sensors that compose the system, only the experimental data collected between January 2015 and March 2018 by the temperature and the stay-cable forces sensors were considered. The temperature charts made with the original data are presented in Appendix B.

According to Tomé (2019), this consideration was made considering that the uniform temperature governs the variation of the force in all instrumented stay cables. Tomé (2019) also noted that the leading causes of the force variations in the stay cables near the mid-span of the central span are the uniform temperature component of the stay cable and the girder.

Regarding stay-cable sensors, which collect the data and estimate in real-time the installed force in the stay cables, they take readings every 30 minutes and are individually named as follows: 'F_TxxCxx'. In contrast, the temperature sensors collect the temperatures every 15 minutes, i.e., four readings per hour ('ST_T_Pxx_xx').

In this study, the data from ten stay cable forces sensors and nine temperature sensors were used. Table 4-1. lists the name of the sensors whose data was considered.

Table 4.1 - Sensors

Stay-cables force sensors	Temperature sensors
F_T18C02	ST_TT19_C13
F_T18C06	ST_P_P27_1
F_T18C13	ST_P_P27_2
F_T18C20	ST_P_P27_3
F_T18L06	ST_P_P27_4
F_T19C02	ST_P_P26_1S
F_T19C06	ST_P_P26_2S
F_T19C13	ST_P_P26_1I
F_T19C20	ST_P_P26_1I
F_T19L06	-

The location in the bridge of all aforementioned sensors can be found in the Appendix A.

4.2.1 Data Averaging

The data collected from the SHM system of the Corgo bridge, as every continuous monitoring system, have missing data and abnormal values, also known as outliers. It happens due to external factors or due to some problems in the measuring devices. However, all the data used on this project was already preprocessed by Tomé (2019). For instance, regarding outlier removal, the Interquartile Range Analysis (IQR) was applied.

After gathering all data, the data averaging process was implemented. This implementation was made by taking the data concerning the referred period and aggregating this data by hour, i.e., for each hour, the average of all the observations collected during that hour was computed.

According to Tomé (2019), this procedure works as a low pass filter removing unusual changes that are difficult to model. For example, such variations can occur because of environmental and operational variations (EOVs) such as wind, solar radiation, and traffic. Besides this, the data averaging may increase the correlation between the dependent variables and the predictor variables.

It is important to highlight that after the daily data aggregation, all the days which had more than seven hours of missing data were not taken into account. Figure 4-2 (a) and Figure 4-2 (b) illustrate, as examples, the resulting time series of the data averaging for the F_T19C13 sensor and ST_P_P27_1 sensor, respectively.

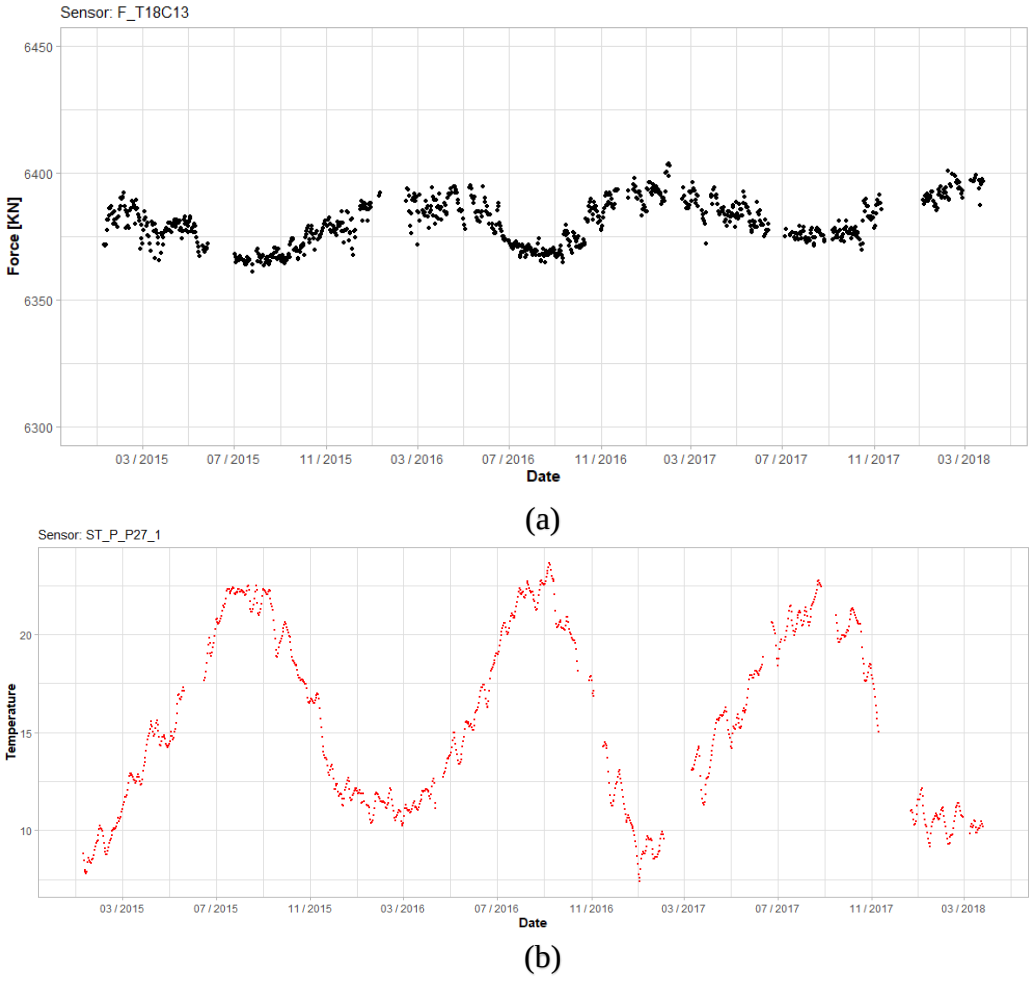


Figure 4-2 - (a) Average daily force in F_T19C13. (b) Average daily temperature in ST_P_P27_1.

As can be seen in the charts, the removal of days with more than 7 hours of missing data was reflected in some long periods of days where the forces recorded in the stay-cables and/or the temperatures recorded by the temperature sensors mentioned were not considered.

A dataset with 1173 observations corresponding to 1173 days between 09/01/2015 and 26/3/2018 was obtained at the end of the described process. Such dataset contains in total 19 features: nine temperatures and ten forces.

4.3 Multi-output Neural Network

A multi-neural network was one of the machine algorithms chosen to build a model for monitoring the structural condition of Corgo Bridge. This machine learning algorithm, as its name implies, works as a typical neural network. The only difference is that the multi-output approach allows obtaining predictions for each output simultaneously (Claveria et al., 2015). The Multi-output neural network also allows the underlying nonlinear correlation between output variables to be extracted. In this way, the accuracy of the output may be enhanced.

As mentioned in section 4.2, the variation of the force in all instrumented stay cables is governed by the uniform temperature. Therefore, the temperatures were considered inputs of the multioutput neural network. On the other hand, the forces were the outputs. In essence, the goal is to build a model where the temperatures, as independent variables, may reflect the real behaviour of the structure, i.e., the stay-cable forces.

However, as it is a time-series modelling and forecasting problem, the time series trend had to be taken into account (Qi and Zhang 2008). As a result of this, two more inputs were considered:

- ***Time index*:** an incremental variable associated with every single week, which seeks to represent the bridge's age.

  $$Time\ index \in [1,2,3,..,N]\,, \qquad N = number\ of\ weeks\ of\ the\ considered\ data$$

- ***Month*:** a variable that represents the month of the year to which observation corresponds. This variable represents the seasonality that might exists in collected data.

  $$Months \in [1,2,3,4,5,6,7,8,9,10,11,12]$$

As the training of a neural network should only include observations that do not have any missing attribute, i.e., without missing data, after adding the new variables (*Time index* and *Month*) to the dataset, all observations/days that had at least one missing attribute were removed from it. This way, the dataset was reduced to 837 observations with the following 21 features:

Inputs (11):

- *Time index*
- *Month*
- *ST_TT19C13*
- *ST_P_P27_1*
- *ST_P_P27_2*
- *ST_P_P27_3*
- *ST_P_P27_4*
- *ST_P_P26_1I*
- *ST_P_P26_2I*
- *ST_P_P26_1S*
- *ST_P_P26_2S*

Outputs (10):

- *F_T18C02*
- *F_T18C06*
- *F_T18C13*
- *F_T18C20*
- *F_T18L06*
- *F_T19C02*
- *F_T19C06*
- *F_T19C13*

Three main parameters were defined to build the multi-output neural network:

- The number of hidden layers
- The number of neurons
- The learning rate – the hypermeter that determines the value of the updated parameter computed with values in the gradient vector (Baladram et al., 2020).

4.3.1 Cross Validation in time series

To evaluate the performance of the model a cross-validation procedure was used. It is a statistical method that helps determine the model's best parameters preventing, for instance, model overfitting.

Although in the machine learning field the regular cross validation approach is made by randomly picking samples out of the available data and split it into train and test set, this process cannot be applied in time series data, because it makes no sense to use values from the future to forecast values in the past (Shrivastava, 2020).

Thus, a *60 observation forward-chaining* (a rolling-basis cross validation) was used, and it was based on a method called forward-chaining and rolling-origin-recalibration evaluation. As shown in the diagram illustrated in Figure 4-3., every new 60 observations (i.e., 60 days of recorded data) a new rolling window with an incremented training horizon is created. For that rolling window, these new 60 observations are considered the test set and all the previous data is assigned to the training set.

In summary, it consists of sliding window parameterization where every 60 observations, such parameters are updated, contributing to a better performance of the model.

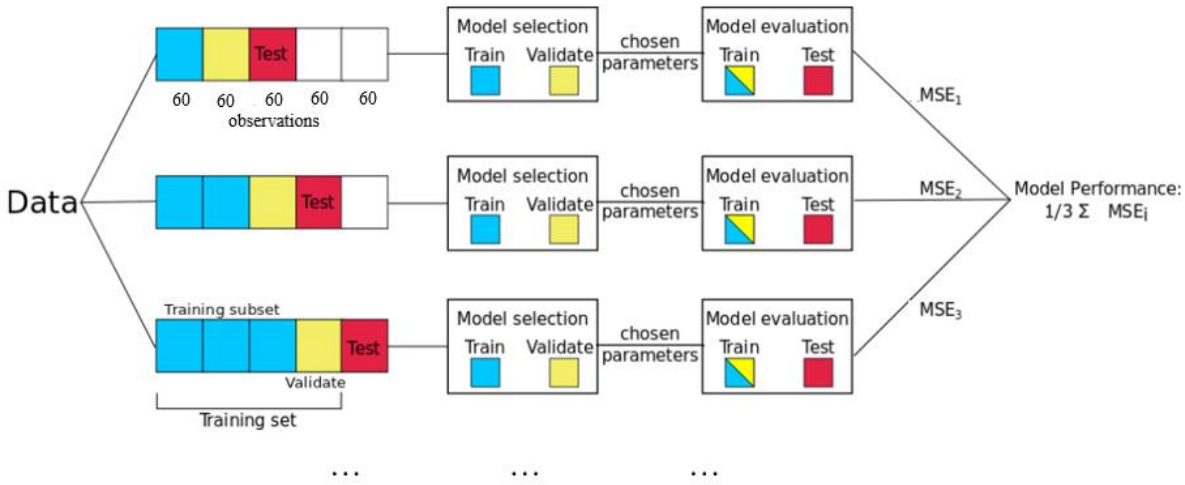
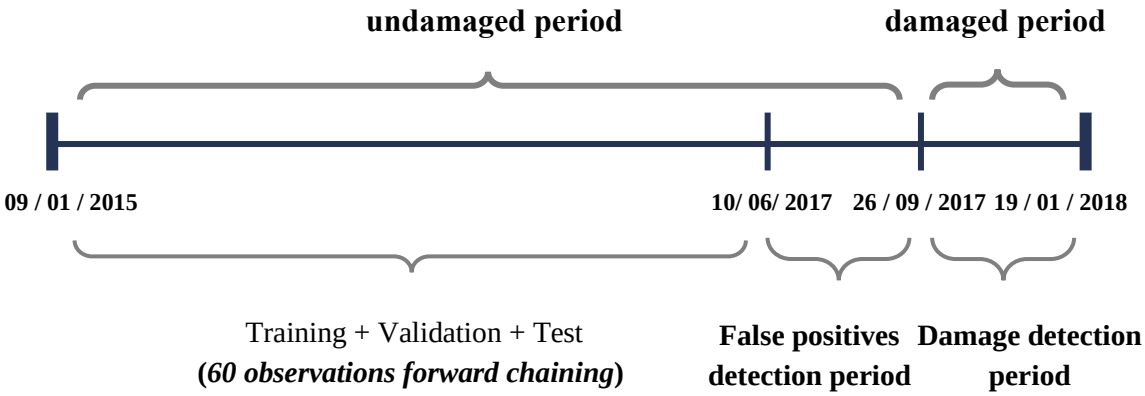


Figure 4-3 - *60 observations Forward-Chaining*. Source: (Shrivastava, 2020)

In order to be in line with this method of cross validation the number of observations considered in the dataset was reduced to the highest multiple of 60 lower than 827. Therefore, the latest 47 days of data (observation) were excluded forming a dataset composed of 780 observations from 09/01/2015 to 19/01/2018. This dataset was divided chronologically as the following diagram shows:



This sliding window parameterisation provides an updated model every time 60 new observations are recorded. Then, an evaluation of the model performance is made. First, it was verified if the model did not cause false alarms that may arise from an undamaged state so, a period of 60 days/observations, called the false positives detection period, was set for that purpose. The same was naturally done for the last 60 observations of the data - a period in which the data was corrupted (explained in chapter 4.3.2) - in order to represent possible damage. The goal in this final set, called the damage detection period, of data is to evaluate the model´s capability to detect damage.

Each rolling window of the mentioned method is divided into three portions: training, validation, and testing. The training portion is used to train the model. As it is the real behaviour of the structures to be predicted, it is clear that the model is trained only using data from the undamaged period. The validation set is then used to select the best model among all the models built with the training portion with every single combination of the following parameters and tested values:

- The number of neurons: 10, 12, 17
- The learning rate: 0.001, 0.0031, 0.01, 0.031, 0.1, 0.316, 1
- The number of hidden layers was fixed in one.

The performance measure used to select the best model was the mean squared error (MSE). This refers to the combination of parameters that best reflects the behaviour of the stay-cables forces with the validation data of the rolling window concerned. After this, the selected model is tested with unseen data, the testing set.

As mentioned, both the false positive period and the damage period are composed of 60 observations each. Analogously to what was previously described, the detection of false positives and the presence of damage was checked in the last window of the rolling windows set. While the 60 observations corresponding to the period of false positives detection constitute the validation set of this window, the latest 60, referring to the detection of damage, form the testing set.

In this testing set, anomaly detection is expected by comparing the observed values with the values predicted by the model created by the neural network. In the presence of damage, such difference should be substantially higher. Without any significant change in the explanatory variables, it is expected to see an increasing error in the predictions in this period, as temperature variables can no longer explain the dependent variables correctly. On the other hand, in the false positive detection period, it is expected that the predicted values are as close as possible

to the observed values, having a low prediction error and a low number of false alarms signalled by high prediction errors.

In the end, it is calculated the mean squared error MSE with the observed and predicted values for each period (false positives and damage detection periods). The two MSE obtained are then compared, being expected a significant difference between these two values.

4.3.2 Data normalisation

As non-normalized data in a machine learning algorithm might cause domination of the variables that use a larger scale affecting the model performance, it was necessary to adapt the normalisation process to the mentioned methodology: *60 observations forward chaining*, introduced earlier. The normalisation chosen was Min-max scaling, an approach that scales the data to the fixed interval: [0,1], where 1 corresponds to the maximum value of a given feature x, and 0 to the minimum value recorded in that same feature, for the data set to be normalised.

So, to normalise the data referring to observation i, the formula used by this approach was the following:

$$x = \frac{x_i - min(x)}{\max(x) - \min(x)} \tag{4.1}$$

Briefly, for each rolling window of the process, the training data is initially normalised according to the formula presented above. In turn, the validation data set is normalised through the formula presented above, but using for each feature x, the $max(x)$ and $min(x)$ of the respective training data. Concerning the rolling window test phase, at first, the training and validation data mentioned above are gathered and normalised using the formula presented above to re-training the model. Then, to evaluate its performance, the testing dataset is normalised through the formula, using, for each feature, the maximums and minimums resultant from the normalization of the junction of the training data with the validation data.

4.3.3 Damage Period

The values observed for the damage period, previously mentioned, were obtained by resorting to a numerical damage simulation (made via the finite element model) performed for the Corgo bridge to analyse its structural behaviour when subjected to the following damage scenario: a reduction of 10% of the area of the stay-cable T19C19. Thus, the experimental time series collected by the sensors of the bridge corresponding to the 10 dependent variables were corrupted with the shifts presented in Table 4-2.

Table 4.2 - Force shifts caused by a reduction of 10% of the section area of the stay cable T19C19.

Sensors	$\Delta\ [KN]$	Sensors	$\Delta\ [KN]$
F_T18C02	-0.4836	F_T19C02	2.5912
F_T18C13	2.2253	F_T19C06	10.032
F_T18C20	12.964	F_T19C13	34.028
F_T18L06	44.708	F_T19C20	44.708
F_T18L06	2.599	F_T19L06	-3.212

As can be seen through the table analysis, while the stay-cables T19C13 and T19C20 show large variations in their strength, the T18C02 shows a minor variation. This can be seen in the chart illustrated in Figure 4-4., which shows an example of the corrupted time series of the T19C20 and T19C13 stay-cables. The orange points represent the same data but without damage, i.e., no corrupted.

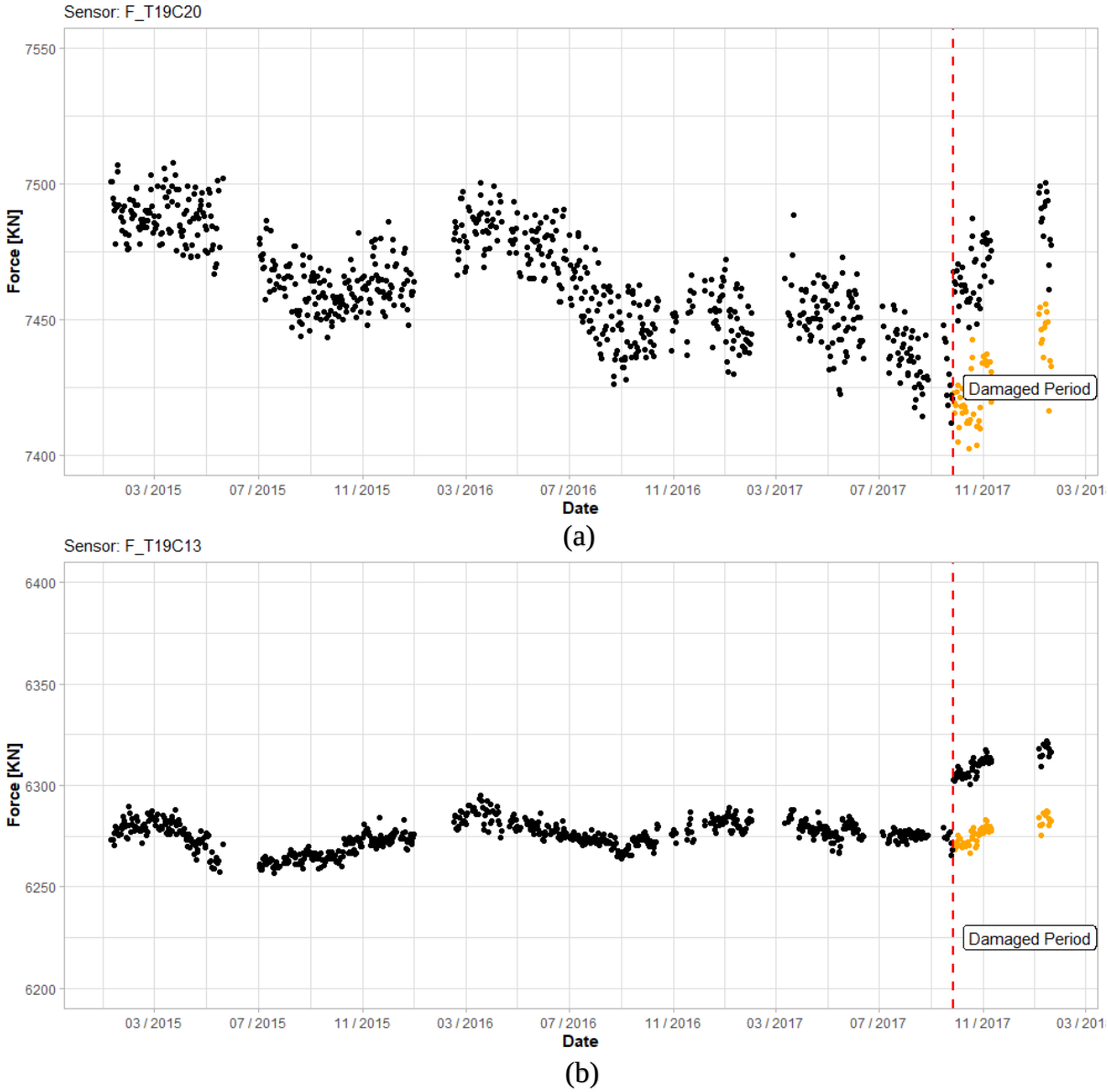


Figure 4-4 - Daily averaged experimental time series with a simulated damage scenario corresponding to 10% of area reduction in the stay cable T19C20 (a) and T19C13 (b) introduced in 26/09/2017 (vertical dashed line).

The daily averaged corrupted time series of the remain stay-cables force sensors are shown in Appendix C.

4.4 Autoencoders - Methodology

In addition to using multi-output neural networks for monitoring the condition and detect damage on the Corgo bridge, an approach using autoencoders was also adopted. These are, fundamentally, feedforward models, so flexibility and other benefits provided by deep learning models may be achieved using them. As explained in chapter 2, the main difference comparing a common neural network with the unsupervised context of the autoencoders is that, regarding

the autoencoders, the number of neurons in the output is the same as the number of neurons in the input. The autoencoder's goal is to reproduce its inputs with the least possible reconstruction loss – measured by the reconstruction loss function. The recreation of these inputs is done by the hidden layers that try to reproduce such inputs after training. Since this is an unsupervised machine learning algorithm, the built model is only 'fed' with inputs.

In view of the objective of detecting anomalies in the Corgo bridge, to build the autoencoder model, the stay-cables forces, i.e., the outputs considered in the construction of the multi-output neural network, were considered the inputs.

Inputs (10):

- *F_T18C02*
- *F_T18C06*
- *F_T18C13*
- *F_T18C20*
- *F_T18L06*
- *F_T19C02*
- *F_T19C06*
- *F_T19C13*
- *F_T19C20*
- *F_T19L06*

To be built, and as illustrated by the example in Figure 4-5, the autoencoder can be described in two parts:

- An autoencoder function $Z = f(X)$ that converts the ten considered inputs to Z codings.
- A decoder function $X' = g(z)$ that produces the reconstructed inputs X' .

Where, X represents the values recorded from the force sensors: *F_T18C02, F_T18C06, F_T18C13, F_T18C20, F_T18L06, F_T19C02, F_T19C06, F_T19C13, F_T19C20, F_T19L06.*

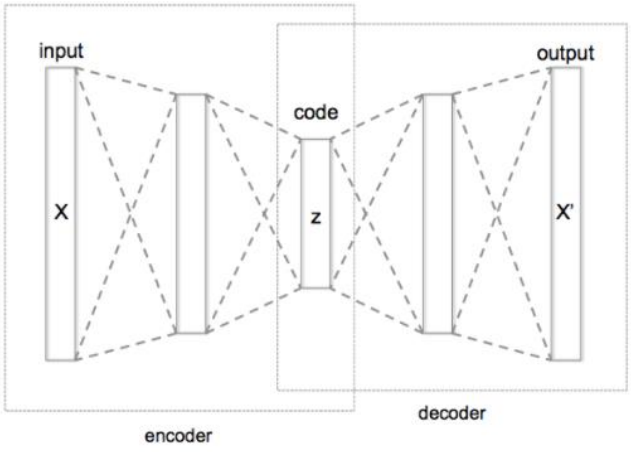


Figure 4-5 - An example of the schematic structure of an autoencoder. Source: (Boehmke, 2020)

The compression of the hidden layers forces the capture of the most dominant features of the input data. Such dimension reduction aims to create a set of codings that represents, as good as possible, the inputs X.

4.4.1 Cross Validation

The same method of cross validation with time series was used for modelling the autoencoder: *60 observations forward chaining*. This way, it was used the same time series dataset and the

same temporal division presented in chapter 4.3.1. The unique difference was the exclusion of the non-necessary features (*Time index*, *Month*, and 9 *Temperatures*).

When building an autoencoder, the most important parameter is the architecture of the hidden layers presented in chapter 2.4. This structure usually is symmetric, and it is defined by the number of hidden layers and the number of neurons of each layer (ex: 4,2,4 - 3 hidden layers each with, respectively, four neurons, two neurons, and four neurons). Thus, the hidden layers structures used for the parameter tuning in any training performed were the following:

Hidden layers architecture (18):

- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 4,2,4
- 4,3,4
- 5,2,5
- 5,3,5
- 6,3,6
- 6,4,6
- 7,3,7
- 7,4,7
- 8,3,8
- 8,4,8
- 8,5,8

Although autoencoders are usually trained with a single hidden layer, they can have multiple hidden layers. It was aiming for a more robust model that, for each training phase of the various rolling windows, the multiple structures presented above were used for parameterizing.

4.4.2 Anomaly Detection

As the loss function of an autoencoder measures the reconstruction error $L(x, x') = \|x - x'\|^2$, it turns possible to identify observations that have larger error rates. Such observations have feature attributes that differ significantly from the 'normal' features.

Through this analysis, it is expected to be possible to detect anomalies (i.e., change of behaviour in the forces recorded by the stay cables sensors) in the Corgo bridge. In its simple form, and since the autoencoder is only trained with undamaged sets of data, it is expected that when fed with data sets that differ from the normal data, the ability to reproduce those inputs is lower, consequently, the reconstruction error higher.

Regarding the false-positive and damage detection analysis, after training, the autoencoder was re-trained on a subset of the inputs that represent high-quality inputs. Therefore, in that re-training, there were only included inputs that achieved a reconstruction error within the *95–th* percentile, excluding the rest of the data that may compromise its performance. Therefore, after re-training, the autoencoder is expected to be more sensitive to detect anomalous inputs, i.e., anomalies.

In summary, it is expected that for the damage detection period where the inputs correspond to time series corrupted by damage scenarios, the reconstruction error, i.e., the MSE is expected to be higher than in the false positive detection period.

5 Results

5.1 Multi-output Neural Network

As mentioned in chapter 4, a sliding window approach was taken - every 60 days, the parameters were updated, simulating data acquisition over time and a constant update of the model.

Table 5.1 presents the attained best parameters and the obtained metrics R^2, RMSE, MSE, and MAPE for the various tests set corresponding to each rolling window used to build the multi-output neural network model.

Table 5.1- Metrics for multi-output neural network rolling windows.

Rolling window	Number of hidden neurons	Learning Rate	RMSE	MAPE	R^2	MSE
1	10	0,1	0,8046	1,3843	0,2939	0,6473
2	10	0,0032	0,1794	0,3951	0,5328	0,0322
3	10	0,3162	0,4861	0,6808	0,2083	0,2363
4	10	0,0316	0,2111	0,2752	0,6195	0,0446
5	10	0,1	0,2239	0,5834	0,7661	0,0501
6	10	0,1	0,1369	0,4319	0,8347	0,0187
7	17	0,0032	0,6279	0,9684	0,4159	0,3943
8	10	1	0,1467	0,2410	0,8824	0,0215
9	10	1	0,1053	0,1919	0,8142	0,0111
10	10	0,01	0,1243	0,3195	0,8315	0,0154

Observing the metrics obtained for each rolling window, it is clear that there is a significant increase in the robustness of the model over time, that is, from window to window. On the other hand, these results reveal that in windows 1 and 7 the results are not as good as expected. In window 1, the first window it was observed a low R^2 and a high MSE. This poor performance might be explained by the tiny amount of data included in the training set, since it is the first window of the model.

In window 7, the high MSE may indicate that the use of 17 neurons in the hidden layer of the neural network may not be the best option. In fact, it was the only time it was selected as the best parameter value, and its performance stood out negatively.

Despite this, for other windows, the prediction of the stay-cables forces, as can be seen in the example of the T18C20 stay-cable in Figure 5-1, is quite positive. In this case, concerning the fourth rolling window, the prediction error is continuously low. Further examples of the charts obtained for other sensors can be found in Appendix D.

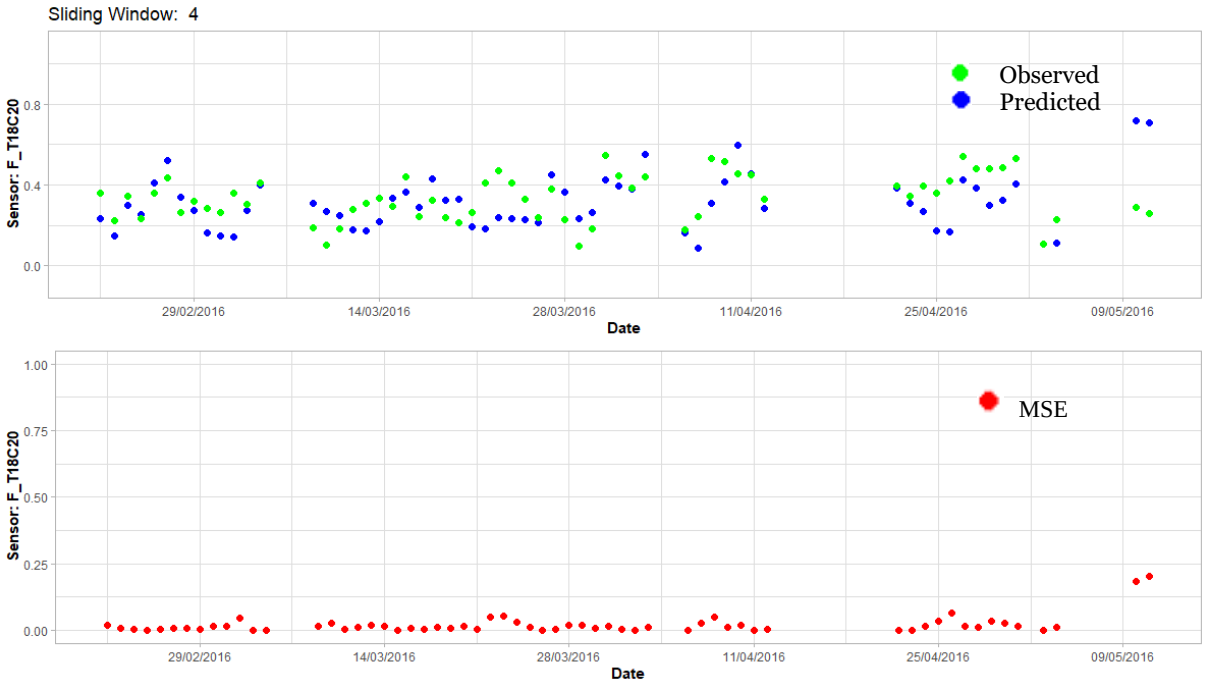


Figure 5-1 - Prediction values *vs* observed values of stay-cable T18C20 concerning the test set of the fourth rolling window.

It may be stated that excluding the MSE of windows 1 and 7 in the expectation of being a kind model's performance 'outlier' and taking the average of all the other MSE obtained for the remaining windows, the model built is robust in the sense that in its performance is reflected in an average MSE value of 0.0537.

This way, it is expected that the MSE relative to the predictions to be made for the false positive detection period are similar, and, ideally, the lowest possible. On the other hand, it is expected a much higher error for the damage detection period.

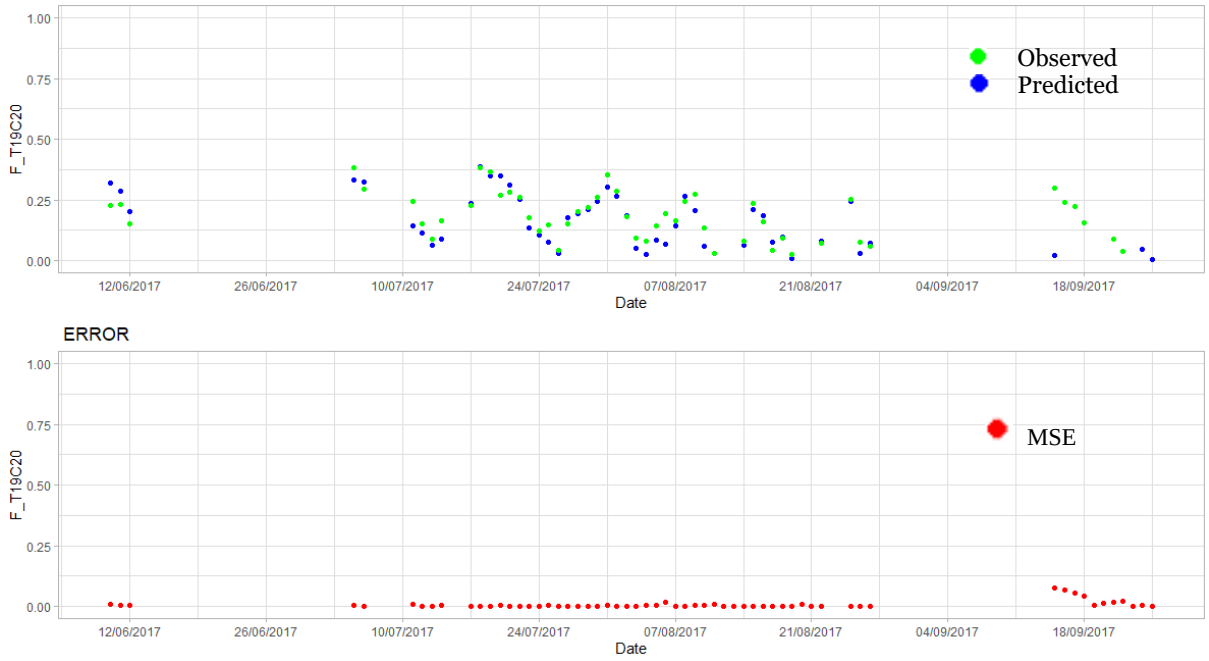
The prediction made for the false positives detection period, as explained in chapter 4.3, was made by training the model with data up to 10/06/2017. Thus, after choosing the best parameters, it was possible to obtain the following metrics presented in Table 5.2 for the false positive detection period.

Table 5.2 - Parameters and the overall metrics obtained from the false positives detection period.

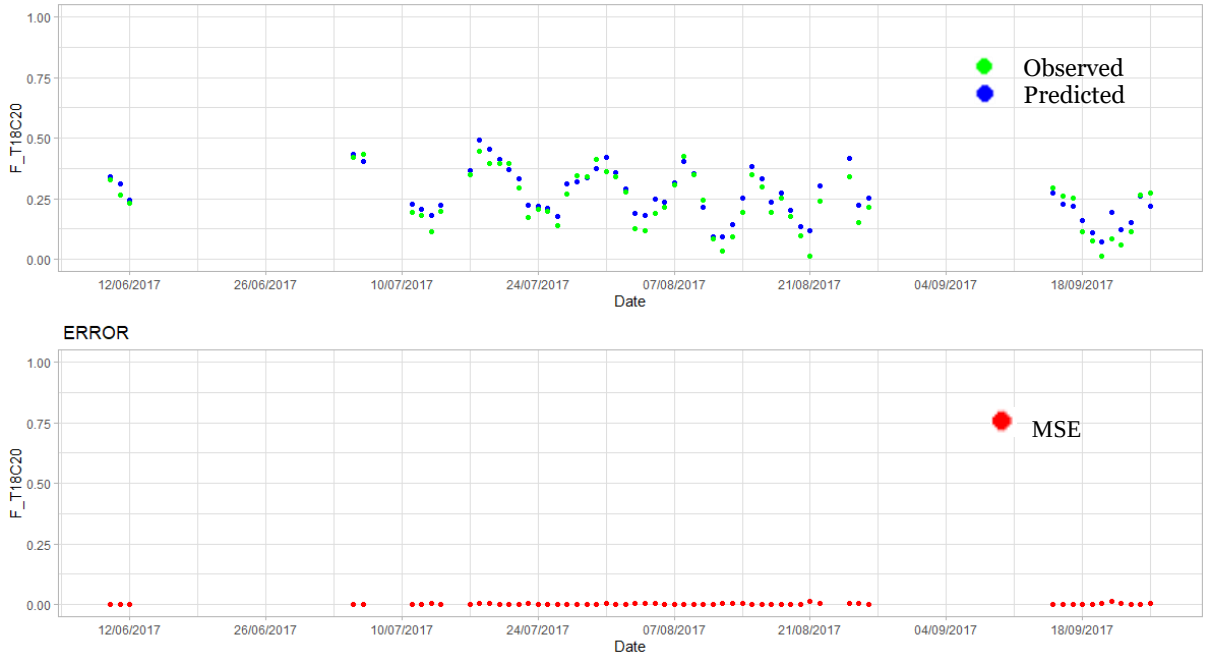
Number of hidden neurons (best)	Learning Rate (best)	RMSE	MAPE	R^2	MSE
10	0,001	0,119	0,309	0,802	0,0142

It is possible to observe that the model for the period in question presents an MSE of 0.014 lower than the model's average mentioned above. It might mean that the performance of the model is getting better over time.

This value, together with the charts presented in Figure 5-2 associated with the T19C20 and T18C20 sensors, shows the good performance of the model since the values of the predicted forces in the stay-cables are very close to the actual values.



(a)



(b)

Figure 5-2 - Prediction values *vs* observed values of stay-cables: T19C20 (a) and T18C20 (b) for false positives detection period.

After analysing the 60 observations, that are part of this period illustrated in the charts, it can be concluded that false positives are practically non-existent.

5.1.1 Damage detection period

The model's accuracy in detecting anomalies was checked between 26/09/2017 and 19/01/2018, corresponding to the 60 observations corrupted by the shifts that resulted from the damage scenario discussed in chapter 4.3.2.

Thus, the multi-output neural network was trained with the first 720 observations of the dataset, including the 60 observations corresponding to the false positive detection period. The same parameters that performed the prediction in testing false positives were used to perform such training.

In order to be possible to compare the MSE during the false positive detection period and during the anomaly detection period, charts like the ones represented in Figure 5-3 were created for all the sensors.

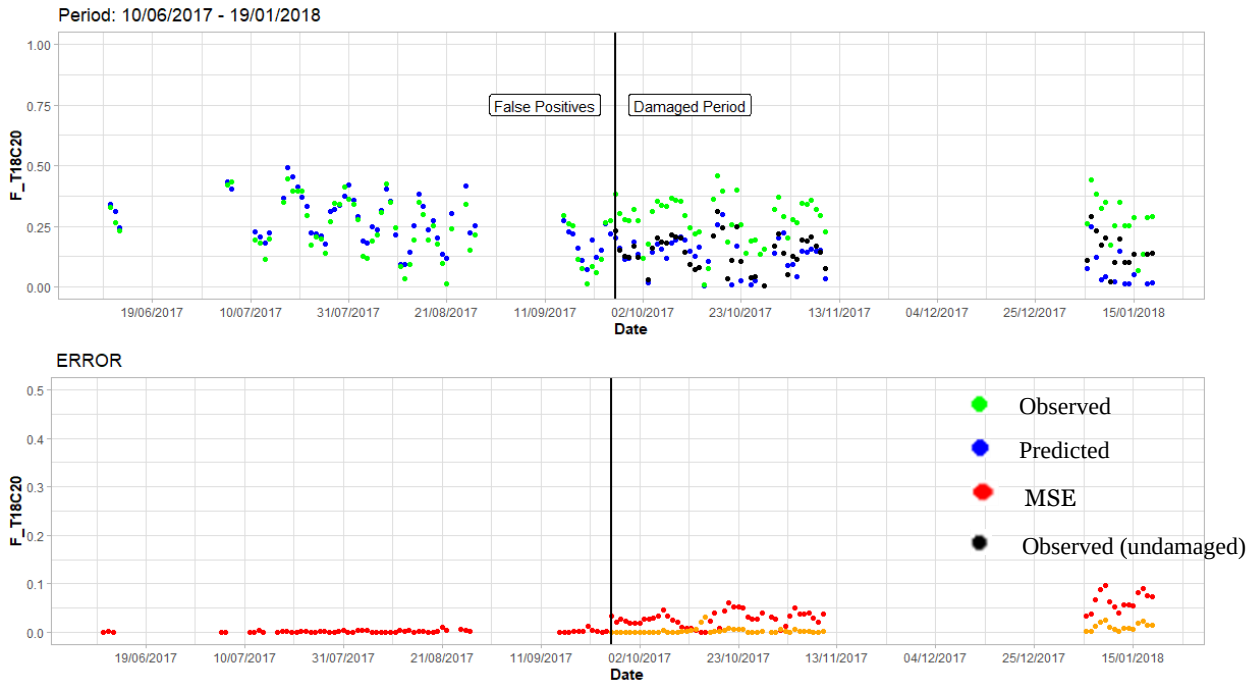


Figure 5-3 - Predicted values *vs* observed values of stay-cable T18C20. Period: False positives detection period + Damage detection period.

As it is possible to observe in Figure 5-3, a slight change in the force of the T18C20 stay-cable between the two periods is noticeable. First, however, it was essential to verify if this slight increase did not correspond to a prediction problem, i.e., to a higher MSE caused by incorrect predictions.

Thus, and since it was expected that the model would predict uncorrupted force values, an analysis of the predicted values was performed by comparing them with the experimental (undamaged) values of the observations of this same period. Through this analysis, as proven by the prediction error (measured between the predicted values and the observed ones without induced damage) represented in orange in the chart, it was concluded that the presence of damage effectively justifies the change in the behaviour of the MSE.

Considering that the applied damage scenario is reflected in different variations in the values of the forces from the various sensors, as represented in Figure 5-4., it has become fundamental to understand to what extent small, medium, and large variations are detectable with the use of this model.

Having already presented the force's behaviour in the T18C20 stay-cable, affected by a variation of 12.964 KN, the charts corresponding to the RSM of ties T18C02 and T19C20 are presented in Figure 5-4 and Figure 5-5, respectively, for the same time horizon.

The charts of the remaining seven sensors are shown in Appendix C.

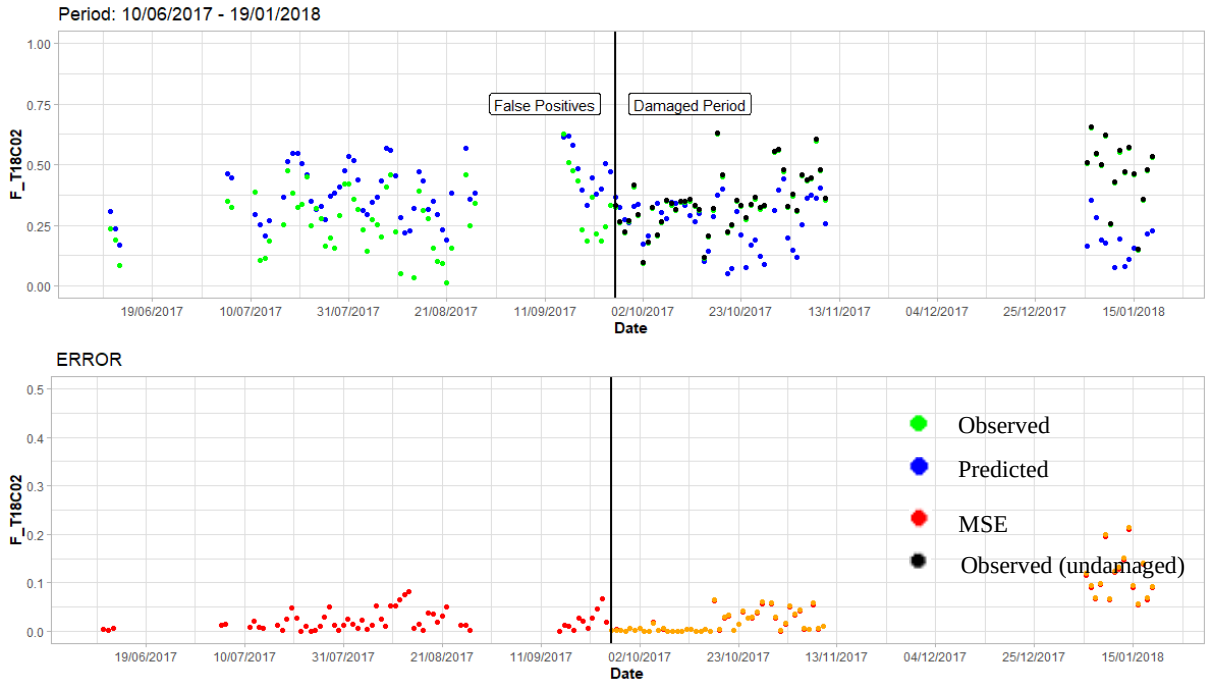


Figure 5-4 - Predicted values *vs* observed values of stay-cable T18C02. Period: False positives detection period + Damage detection period.

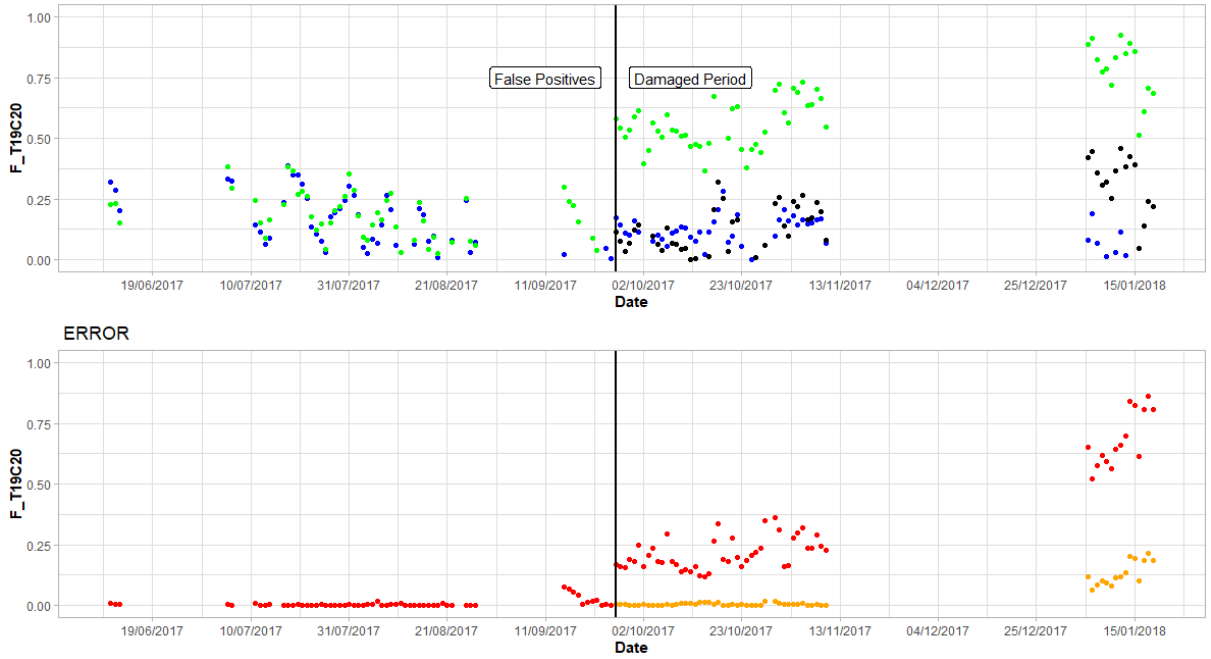


Figure 5-5 - Predicted values *vs* observed values of stay-cable T19C20. Period: False positives detection period + Damage detection period.

It can be seen that the increase in the MSE prediction error, as expected, is more and more evident as the variation induced by the stay-cable damage is higher. On the other hand, the stay-cables less affected by the damage, as is the case of T18C02 whose variation was -0.4863KN, no change in the MSE values is verified in the transition from the false positive detection period to the damage detection period.

The sum up, the overall metrics obtained for the damage detection period are shown in Table 5.3.

Table 5.3 - The overall metrics obtained from the damage detection period.

Number of hidden neurons	Learning Rate	RMSE	MAPE	R^2	MSE
10	0,001	0,407	0,639	0,475	0,166

5.1.2 Analysis - Model Performance

The MSE associated with each of the 120 observations that compose the false positive detection period (60), and the damage detection period (60) were analysed to compare the performance of the multi-output neural network with the performance of the autoencoder. For that, the mean squared error of the predictions of the ten stay-cables forces was calculated for each observation. In this way, the damage detection was evaluated with data from all the sensors and not based only on specific sensors.

The results of the presented procedure are illustrated in the histograms represented in Figure 5-6 and Figure 5-7.

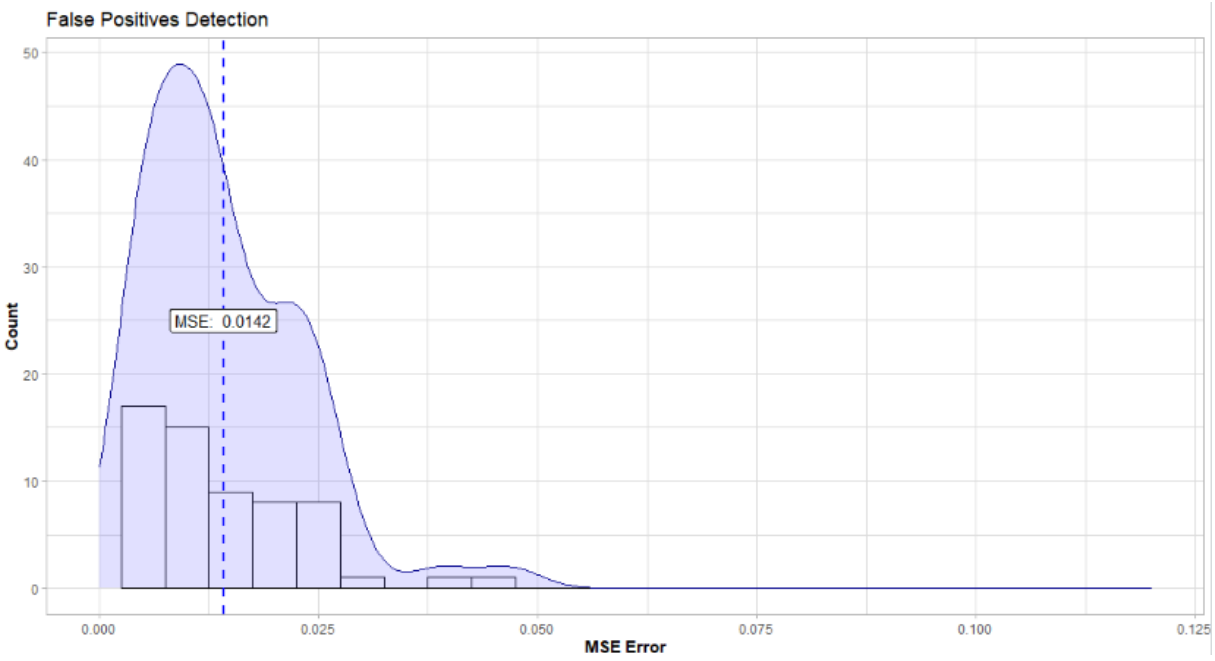


Figure 5-6 – MSE of the 60 observations predicted for the false positives detection period.

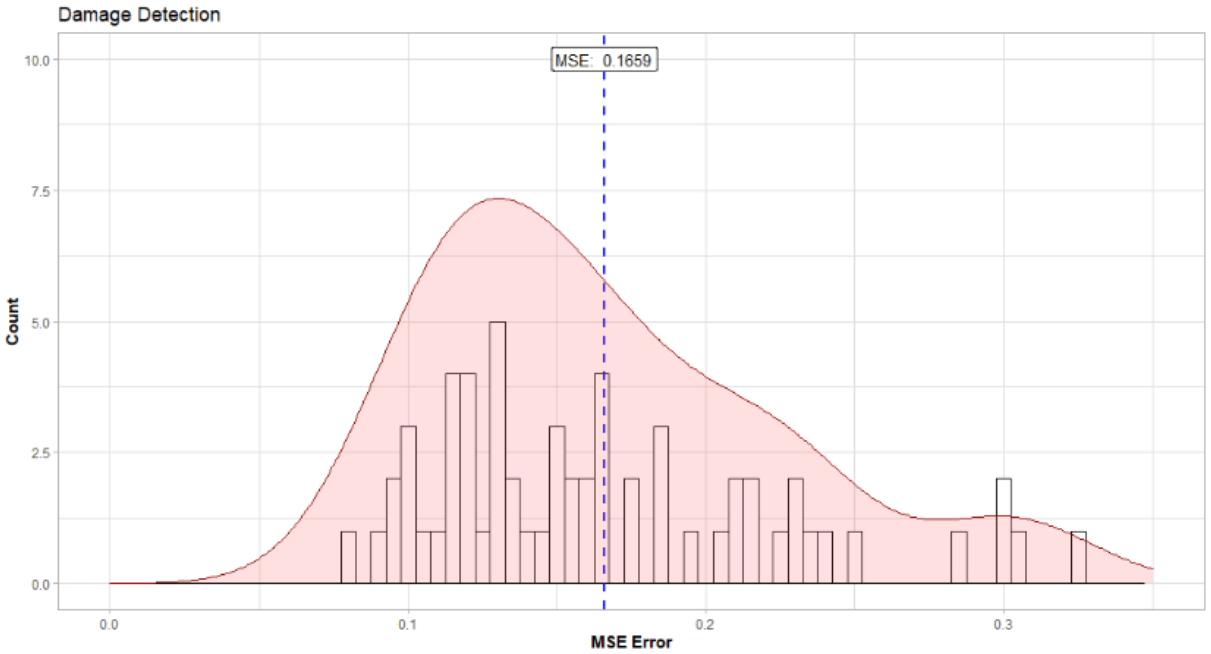


Figure 5-7 - MSE of the 60 observations predicted for the damage detection period.

Comparing the results for each period can be concluded that the prediction error of the multi-output neural network is substantially higher in the case of damage. Furthermore, a detailed observation of the histograms also allows us to note that any mean squared error associated with the observations that belong to the damage detection period is higher than the highest value obtained from the MSE in the false positive detection period.

5.2 Autoencoders

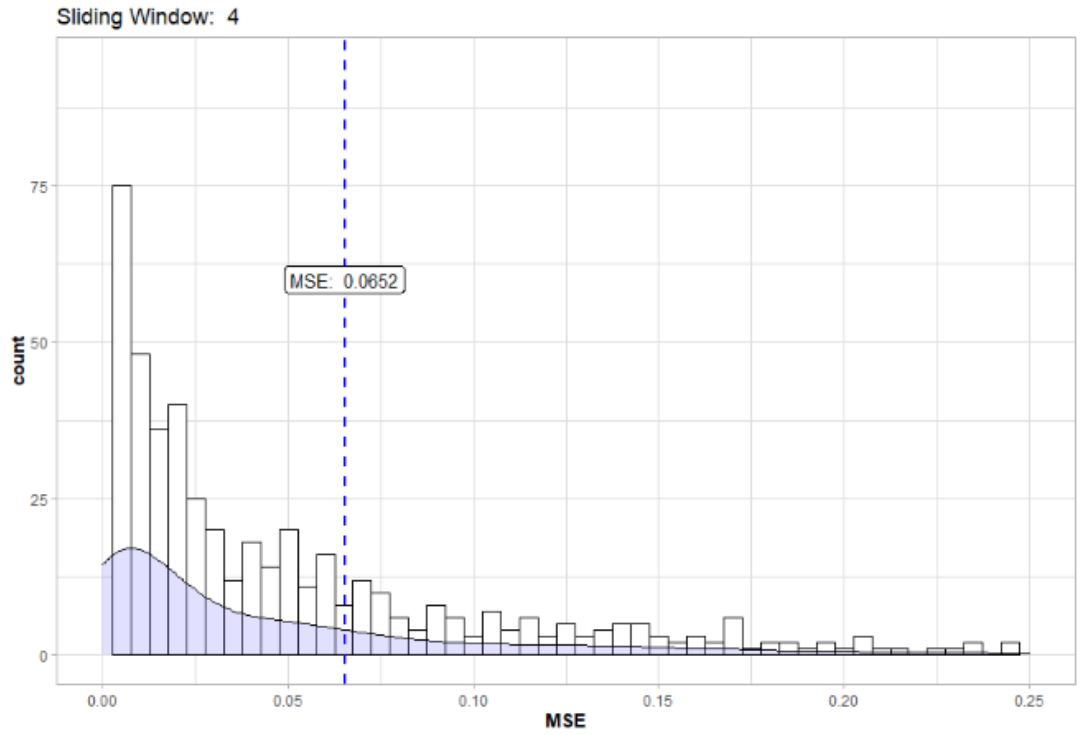
Concerning the implementation of the autoencoders, a sliding window approach was also considered. This way, the parameters were also updated every 60 days/observations, simulating data acquisition over time and a constant update of the model with new data. Table 5.4 shows the achieved metrics for each rolling window' testing set.

Table 5.4 - Metrics for multi-output neural network rolling windows.

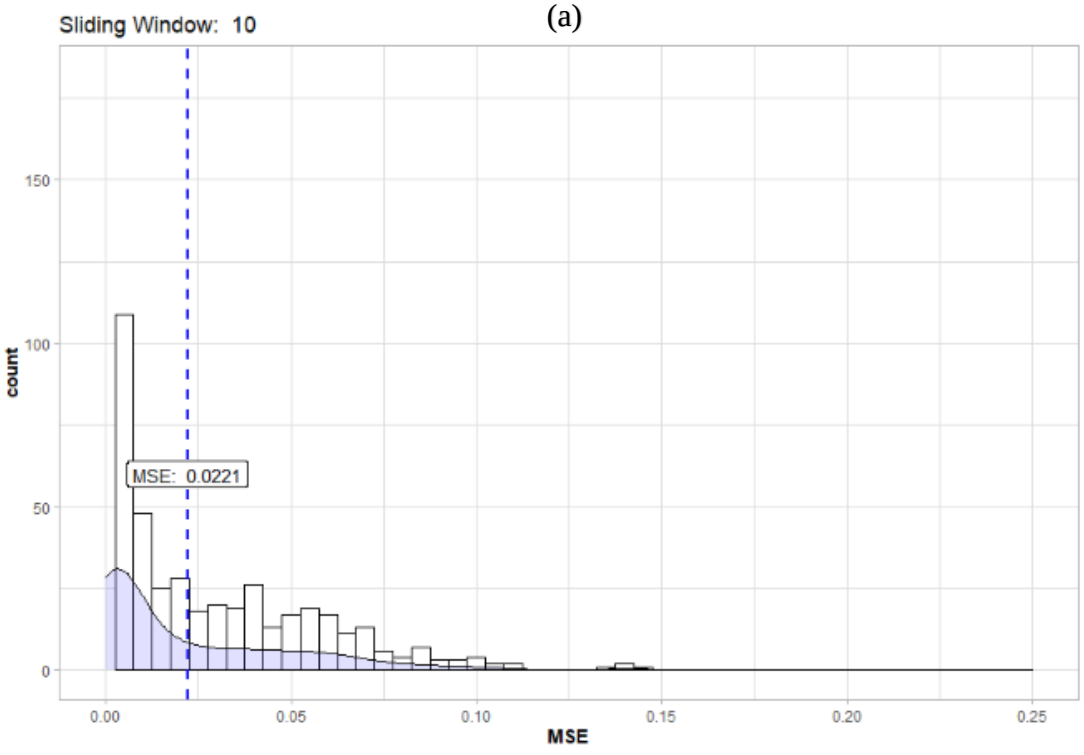
Rolling window	Hidden layers' structure	RMSE	MAPE	R^2	MSE
1	[4,3,4]	0,4878	1,0323	-0,1885	0,2380
2	[7,3,7]	0,2289	0,5223	0,3642	0,0524
3	[7,3,7]	0,3065	0,4479	0,4028	0,0940
4	[7,3,7]	0,2554	0,3354	0,2821	0,0652
5	[8]	0,2643	0,4014	0,2770	0,0699
6	[7,4,7]	0,2142	0,8422	0,7429	0,0459
7	[5]	0,2335	0,6567	0,8180	0,0545
8	[8,5,8]	0,1831	0,2946	0,8110	0,0335
9	[8,5,8]	0,1660	0,3177	0,7147	0,0276
10	[6]	0,1485	0,4731	0,8673	0,0221

As we can observe, over time, this unsupervised machine learning algorithm, is able to replicate its inputs (all the stay cables forces) with an increasingly lower MSE. However, as occurred in the first window of the multi-output neural network implementation, the performance of the first rolling window is not as good as the other rolling windows. It may be due to the yet reduced volume of data that constitute the training dataset.

Histograms in Figure 5-8 represent two examples of MSE – reconstruction error – for the fourth (a) and the tenth (b) sliding window.



(a)



(b)

Figure 5-8 – Examples of the reconstruction error distribution over the implementation of the sliding window approach. Fourth rolling window (a). Tenth rolling window (b).

The deviation of the dashed blue line to the left, from the fourth window to the tenth, as shown in Figure 5-8, proves the model's performance improvement in reconstructing the inputs as more data is integrated into it.

As in multi-output neural networks, it is necessary to train the model with the dataset data recorded until 09/06/2017. After the validation, i.e., the definition of the main parameter (the hidden layers structure), is obtained a model that is used later to evaluate anomaly detection and possible false positives This model reconstructed its inputs with an average reconstruction error shown in Table 5.5 and the chart represented in Figure 5-9.

Table 5.5 - Parameters and reconstruction error of the model

Hidden layers' structure	MSE – Average of Reconstruction error
[8]	0.0506

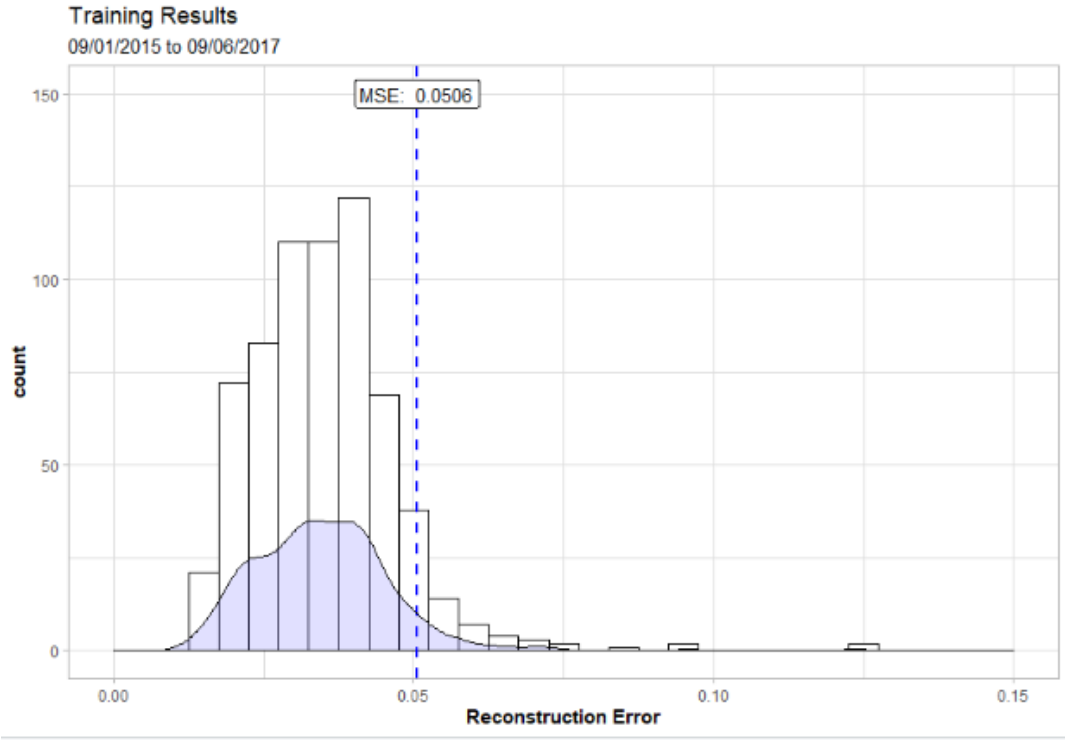


Figure 5-9 - Reconstruction error distribution before the data subsetting

As we can see from the histogram analysis, very few reconstruction errors (MSE) of the inputs are much higher than the general average. Thus, as explained in chapter 4.4.2, in order to make the model more robust and a little more sensitive to anomaly detection (damage), after the autoencoder training and the selection of the best parameters, the autoencoder was re-trained with all the inputs that achieved a reconstruction error within the *95-th* percentile and excluding the remain data.

Summing up, the autoencoder was re-trained on a subset of the inputs that had been determined as a good representation of high-quality inputs.

Thus, after the best parameters' selection and the re-training, it was possible to obtain the following average for the reconstruction error presented in

Table 5.6.

Table 5.6 - Parameters and reconstruction error of the model after subsetting (percentile *95-th*)

Hidden layers' structure	New MSE (after percentile)
[8]	0.0427

It was then verified that excluding from the autoencoder training data (observations) considered not so well representative of what is the normal behaviour of the inputs, the mean value of the reconstruction error decreases approximately 0.01.

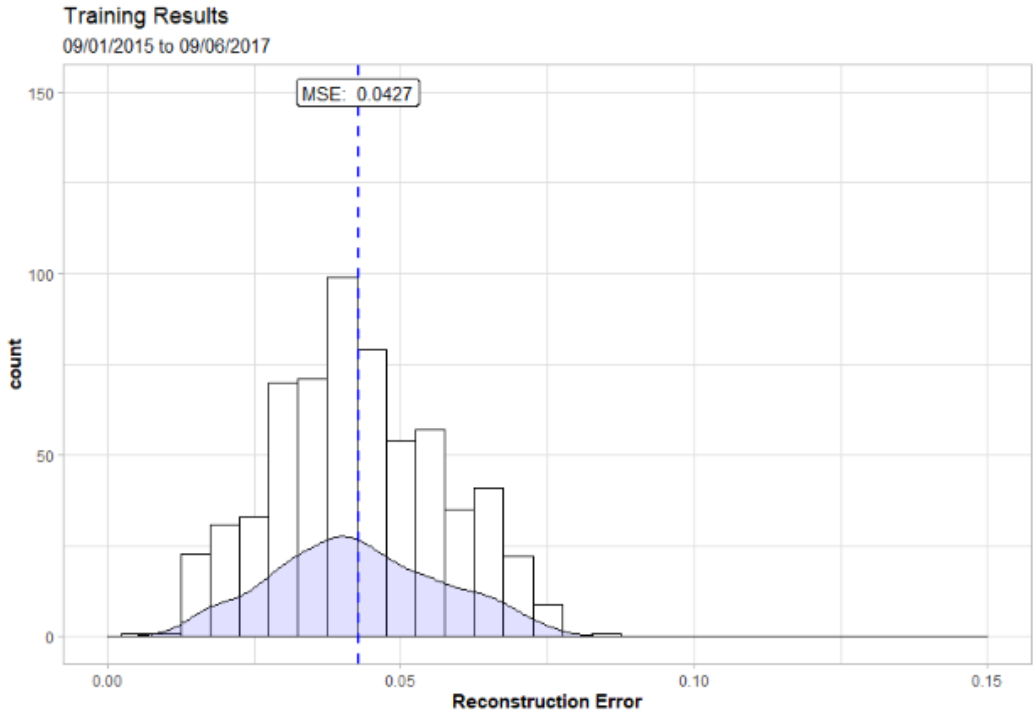


Figure 5-10 - Reconstruction error distribution after data subsetting

5.2.1 False Positives Detection

After building the model, its sensitivity to false positives was evaluated. Once the autoencoder was trained with data corresponding to the undamaged period, it can be stated that the model is having a good performance if the reconstruction error associated with the 60 observations that constitute the period of detection of false positives does not deviate from the normal and remains close to 0.0427.

The errors associated with this period are represented in the histogram in Figure 5-11.

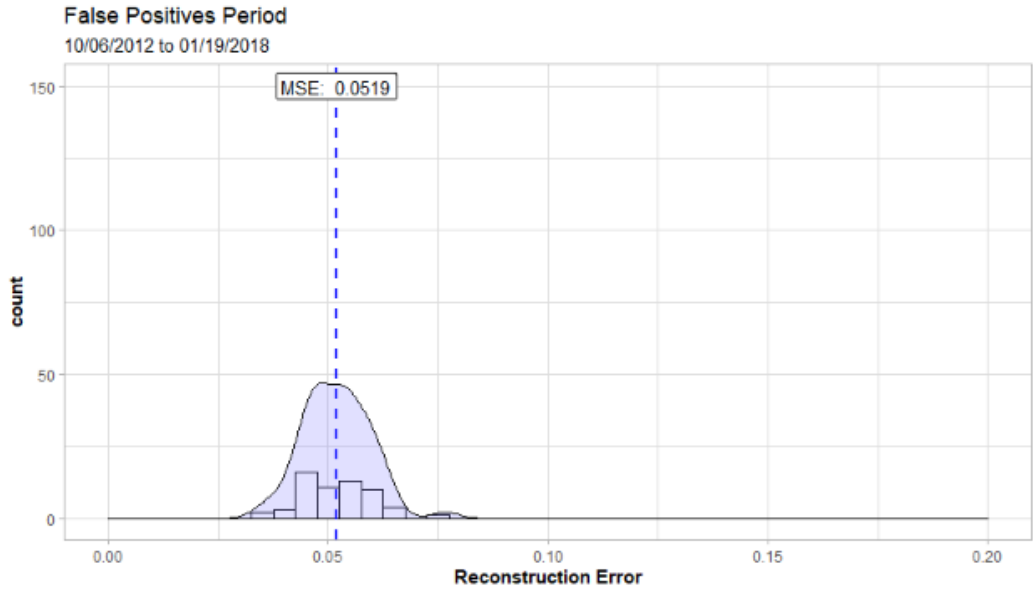


Figure 5-11 - Reconstruction error of the false positives detection period.

The mean value of the reconstruction error obtained for this period, 0.0519, was low and very close to the model's RSM, 0.0427. This proximity between these values was already expected since the inputs - the forces in the stay-cables of all the 60 observations - and the data fed the model in its training phase correspond to regular values that the model already is able to replicate.

5.2.2 Anomaly Detection

For the damage detection period, it could be anticipated that a higher reconstruction error was expected, since the autoencoder inputs in this period were never previously seen by the model because they were corrupted values associated with structural damage, in this case in one of the stay-cables - as explained in Chapter 4.3.2.

The histogram in Figure 5-12 shows that, in the presence of damage, the errors associated with the reconstruction of the inputs become average twice as large as in the false positive detection period. This highlights the ability of autoencoders to detect anomalies when the input data 'deviates' from normal.

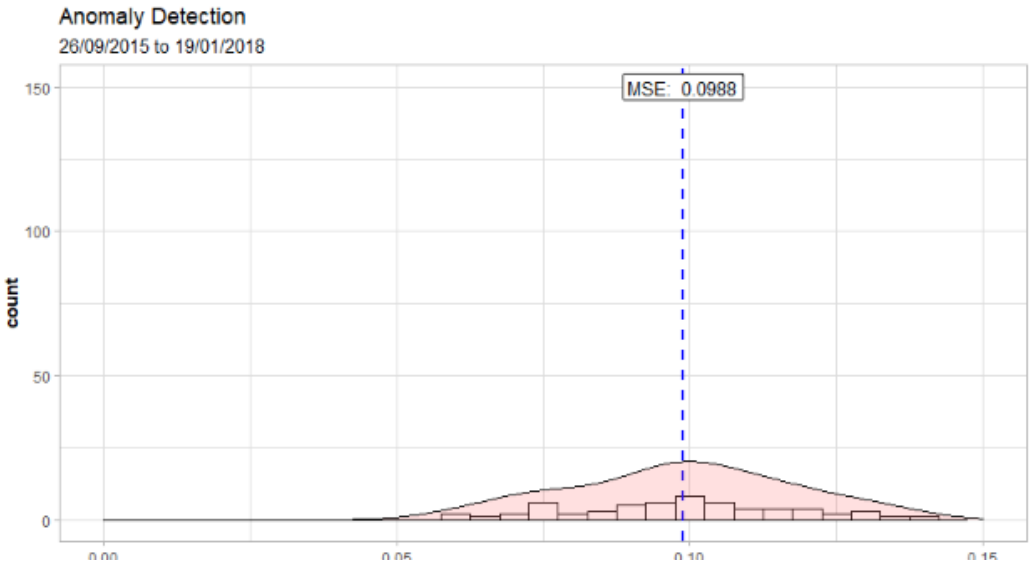


Figure 5-12 - Reconstruction error of the damage detection period.

5.3 Global analysis of the models – Anomaly detection

After building the two models and analysing the results, it is possible to state that both approaches in the presence of damage behave differently, i.e., produce different outputs from what is expected under a regular condition.

However, some differences between the two methods need to be highlighted:

- The multi-output neural network is a supervised algorithm while the autoencoder is unsupervised;
- The multi-output neural-network takes into account the sensor temperatures and the stay-cable forces while the autoencoder analysis is only based on the stay-cable forces;
- The processing time of the multi-output neural network, due to its complex structure, is long. For example, the last rolling window took about 8 hours to present results. In contrast, the autoencoder took just over 5 minutes to process the ten rolling windows used to simulate updating the parameters over time.

6 Conclusions

Keeping in mind the current state-of-the-art of Structural health monitoring, this project aimed to study and create an innovative approach to structural health monitoring. For that, a multi-output neural network model and an autoencoder were created. Both machine learning algorithms used were based on the data collected by some of the temperature and force sensors of the stay-cables of the Corgo Bridge between the period January 2015 and January 2018.

Although two different models were created, their validation followed the same method: 60 observations forward chaining. The use of this method made both models more robust to the extent that for both the neural network and the autoencoder their parameters are updated whenever the bridge sensor system records 60 new observations. Since each observation corresponds to the daily averages of the data collected by the sensors, it can be stated that such update occurs every 60 days. However, these days may not be followed as it happened in this work since they may contain missing data.

This parametric simulation effect was simulated, and it was verified that, in fact, over time (i.e., every time 60 new observations were added to the dataset) the parameters were changing, and the prediction error between updates was decreasing. Thus, it can be said that with this method of parametric updating, it is possible to apply this methodology to any instrumented infrastructure since practically the beginning of its activity. Since the model is based on collected data and, at first, these are scarce, it is expected that its performance is not the best and, consequently, its reliability is lower. However, and as verified in this study, from the first update onwards, the model starts to reproduce the natural behaviour of the infrastructure progressively better. Therefore, as more and more data is included in the model training over time, the reliability of the SHM system of which the model in question is a part, tends to increase. Thus, it can be concluded that this cross validation method allows for good results and immediate applications in structures without the need to wait for a large amount of data to be collected in order to create a robust model.

Regarding the results obtained from the multi-output neural network and autoencoder modelling, the potential of these models in monitoring the behaviour of a structure, more specifically in anomaly detection, was proven. The significant difference between the MSE obtained between the damage detection period and the false positive detection period of each of the models highlights the sensitivity to damage detection and the resistance against possible false alarms. Therefore, it can be concluded that the neural network algorithm and the autoencoder algorithm may be used to support analytical methods to prevent situations caused by damage.

Based on the results obtained in this project, a comparison of their performance can be made by constructing a confidence interval for the error. This way, it would be possible to understand after which day each of the models begins to detect values outside that range, i.e., anomalies.

As future work, based on the work developed, it would be interesting to create a model that was not based only on a single damage scenario, as was done in this project, but on a significant number of simulated damage scenarios in order to build a model capable of detecting several types of damage. It would also be worthwhile to study the influence of the size of the rolling windows on the model construction. Also, a possible variation of the size of the rolling windows into the same implementation should be studied in order to try to avoid, for instance, the bad performance observed in the first window of this study.

Bibliography

Alipour, M., D. K. Harris, L. E. Barnes, O. E. Ozbulut, and J. Carroll. 2017. "Load-Capacity Rating of Bridge Populations through Machine Learning: Application of Decision Trees and Random Forests." *Journal of Bridge Engineering* 22 (10): 04017076. https://doi.org/10.1061/(asce)be.1943-5592.0001103.

Alpaydin, E. 2014. *Introduction to Machine Learning*. MIT Press.

Azimi, M. Eslamlou, A., D., Pekcan, and G. 2020. "Data-Driven Structural Health Monitoring 476 and Damage Detection through Deep Learning: State-of-the-Art Review." *Sensors* 20 (10).

Baladram, M. S., A. Koike, and K. D. Yamada. 2020. "Introduction to Supervised Machine Learning for Data Science." *Interdisciplinary Information Sciences* 26 (1): 87–121. https://doi.org/10.4036/iis.2020.a.03.

Ballard, D. H. 1987. "From: AAAI - 87 Proceedings. Copyright © 1987 , AAAI (Www.Aaai.Org). All Rights Reserved. Dana H. Ballard Department of Computer Science University of Rochester, Rochester, New York 14627," 279–84.

Bao, Y., Z. Chen, S. Wei, Y. Xu, Z. Tang, and H. Li. 2019. "The State of the Art of Data Science and Engineering in Structural Health Monitoring." *Engineering* 5 (2): 234–42. https://doi.org/10.1016/j.eng.2018.11.027.

Bao, Y., and H. Li. 2020. "Machine Learning Paradigm for Structural Health Monitoring." *Structural Health Monitoring*. https://doi.org/10.1177/1475921720972416.

Bao, Y., Z. Tang, and H. Li. 2019. "Computer Vision and Deep Learning – Based Data Anomaly Detection Method for Structural Health Monitoring." https://doi.org/10.1177/1475921718757405.

Based, N. 1992. "A Neural Network Based Damage Analysis of Smart Composite Beams Y . Teboub and P . Hajela Troy , New York Fourth AIAA / USAF / NASA / OAI Symposium on ~ Ultidisblinar ~ Analysis and Optimization For Permkskn t Analysis of Smart Composite Beams."

Boehmke, B., and B. Greenwell. 2021. *Hands-On Machine Learning with R*. https://bradleyboehmke.github.io/HOML/.

Carvalho, J. F. 2020. "Developing Data-Driven Models to Assess Structures ' Health Condition." Faculdade de Engenharia da Universidade do Porto.

Chandola, V., A. Banerjee, and V. Kumar. 2009. "Anomaly Detection : A Survey" 41 (3): 1–58. https://doi.org/10.1145/1541880.1541882.

Chang, C.-M., J.-Y. Chou, P. Tan, and L. Wang. 2017. "A Sensor Fault Detection Strategy for Structural Health Monitoring Systems." *Smart Structures and Systems* 20 (1): 43–52.

Chen, J., S. Yuan, and H. Wang. 2020. "On-Line Updating Gaussian Process Measurement

Model for Crack Prognosis Using the Particle Filter." *Mechanical Systems and Signal Processing* 140: 106646. https://doi.org/10.1016/j.ymssp.2020.106646.

Chen, Z., Y. Bao, H. Li, and B. F. Spencer. 2018. "A Novel Distribution Regression Approach for Data Loss Compensation in Structural Health Monitoring." *Structural Health Monitoring* 17 (6): 1473–90. https://doi.org/10.1177/1475921717745719.

———. 2019. "LQD-RKHS-Based Distribution-to-Distribution Regression Methodology for Restoring the Probability Distributions of Missing SHM Data." *Mechanical Systems and Signal Processing* 121: 655–74. https://doi.org/10.1016/j.ymssp.2018.11.052.

Cios, K. J., R. W. Swiniarski, W. Pedrycz, and L. A. Kurgan. 2007. *The Knowledge Discovery Process*. Boston: Springer US.

Claveria, O., E. Monte, and S. Torra. 2015. "Multiple-Input Multiple-Output vs. Single-Input Single-Output Neural Network Forecasting."

Cross, E. J., K. Worden, and Q. Chen. 2011. "Cointegration: A Novel Approach for the Removal of Environmental Trends in Structural Health Monitoring Data." *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 467 (2133): 2712–32. https://doi.org/10.1098/rspa.2011.0023.

Daneshtalab, M., and M. Modarressi. 2020. *Hardware Architectures for Deep Learning*. Institution of Engineering and Technology.

Dunia, R., and S. J. Qin. 1998. "Subspace Approach to Multidimensional Fault Identification and Reconstruction." *AIChE Journal* 44 (8): 1813–31. https://doi.org/10.1002/aic.690440812.

Farreras-Alcover, I., M. K. Chryssanthopoulos, and J. E. Andersen. 2015. "Regression Models for Structural Health Monitoring of Welded Bridge Joints Based on Temperature, Traffic and Strain Measurements." *Structural Health Monitoring* 14 (6): 648–62. https://doi.org/10.1177/1475921715609801.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Gopalakrishnan, S. 2009. "Modeling Aspects in Finite Elements." In *Encyclopedia of Structural Health Monitoring*. American Cancer Society.

Gui, G., L. Pan, H., Z., Li, Y., and Z. Yuan. 2017. "Data-Driven Support Vector Machine with Optimization Techniques for Structural Health Monitoring and Damage Detection." *Journal of Civil Engineering* 21 (2): 523–34.

Gurney, K. 1997. *An Introduction to Neural Networks*. 1st ed. CRC Press.

Hugo, W., L. Pinaya, S. Vieira, R. Garcia-dias, A. Mechelli, and S. Andre. 2020. "Chapter 11 - Autoencoders." In *Machine Learning*, 193–208. Elsevier Inc. https://doi.org/10.1016/B978-0-12-815739-8.00011-0.

Jordan, J. 2017. "Evaluating a machine learning model." 2017. https://www.jeremyjordan.me/evaluating-a-machine-learning-model/.

———. 2018. "Introduction to autoencoders." 2018. https://www.jeremyjordan.me/autoencoders/.

Kesavan, A., S. John, and I. Herszberg. 2008. "Strain-Based Structural Health Monitoring of Complex Composite Structures." *Structural Health Monitoring* 7 (3): 203–13. https://doi.org/10.1177/1475921708090559.

Kim, J. T., Y. S. Ryu, H. M. Cho, and N. Stubbs. 2003. "Damage Identification in Beam-Type Structures: Frequency-Based Method vs Mode-Shape-Based Method." *Engineering*

Structures 25 (1): 57–67. https://doi.org/10.1016/S0141-0296(02)00118-9.

Kromanis, R., and P. Kripakaran. 2013. "Support Vector Regression for Anomaly Detection from Measurement Histories." *Advanced Engineering Informatics* 27 (4): 486–95. https://doi.org/10.1016/j.aei.2013.03.002.

Kudva, J. N., N. Munir, and P. W. Tan. 1992. "Damage Detection in Smart Structures Using Neural Networks and Finite-Element Analyses." *Smart Materials and Structures* 1 (2): 108–12. https://doi.org/10.1088/0964-1726/1/2/002.

Leon. 2021. "Digital Twin for Intelligent Monitoring," 2021. https://westatix.com/blog/digital-twin-for-intelligent-monitoring/.

Liu, H., S. Shah, and W. Jiang. 2004. "On-Line Outlier Detection and Data Cleaning." *Computers and Chemical Engineering* 28 (9): 1635–47. https://doi.org/10.1016/j.compchemeng.2004.01.009.

Liu, K., and N. El-Gohary. 2021. "SemanticNeuralNetwork Ensemble ForAutomatedDependency 550 Relation Extraction FromBridge InspectionReports." *Journal OfComputing InCivil Engineering* 4: 35.

Liu, Y. Y., Y. F. Ju, C. D. Duan, and X. F. Zhao. 2011. "Structure Damage Diagnosis Using Neural Network and Feature Fusion." *Engineering Applications of Artificial Intelligence* 24 (1): 87–92. https://doi.org/10.1016/j.engappai.2010.08.011.

Mehrjoo, M., N. Khaji, H. Moharrami, and A. Bahreininejad. 2008. "Damage Detection of Truss Bridge Joints Using Artificial Neural Networks." *Expert Systems with Applications* 35 (3): 1122–31. https://doi.org/10.1016/j.eswa.2007.08.008.

Mitchell, T. M. 1997. *Machine Learning*.

Nguyen, D. H., T. T. Bui, G. De Roeck, and M. Abdel Wahab. 2019. "Damage Detection in Ca-Non Bridge Using Transmissibility and Artificial Neural Networks." *Structural Engineering and Mechanics* 71 (2): 175–83. https://doi.org/10.12989/sem.2019.71.2.175.

Park, G., and D. J. Inman. 2007. "Structural Health Monitoring Using Piezoelectric Impedance Measurements." *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 365 (1851): 373–92. https://doi.org/10.1098/rsta.2006.1934.

Peng, C., Y. Fu, and B. Spencer. 2017. "Sensor Fault Detection, Identification, and Recovery Techniques for Wireless Sensor Networks: A Full-Scale Study." In *PhD Project: Sudden Event Monitoring Using Wireless Smart Sensors*. The 13th International Workshop on Advanced Smart Materials and Smart Structures Technology, Tokyo, Japan.

Posenato, D., F. Lanata, D. Inaudi, and I. F. C. Smith. 2008. "Model-Free Data Interpretation for Continuous Monitoring of Complex Structures." *Advanced Engineering Informatics* 22 (1): 135–44. https://doi.org/10.1016/j.aei.2007.02.002.

Qi, M., and G. P. Zhang. 2008. "Trend Time – Series Modeling and Forecasting With Neural Networks" 19 (5): 808–16.

Rastin, Z., G. G. Amiri, and E. Darvishan. 2021. "Deep Convolutional Autoencoder" 2021.

Reynders, E., G. Wursten, and G. de Roeck. 2014. "Output-Only Structural Health Monitoring in Changing Environmental Conditions by Means of Nonlinear System Identification." *Structural Health Monitoring* 13 (1): 82–93. https://doi.org/10.1177/1475921713502836.

Rodrigues, M. 2020. "Structural Health Monitoring A Data-Driven Damage Detection Approach A Data-Driven Damage Detection Approach By."

Rodrigues, M., M. V. L., C. Felix, and C. Rodrigues. 2021. "Machine Learning and Co-

Integration for Structural Health Monitoring under Environmental and Operational Variabilities," 1–35.

Rodrigues, M., V. L. Miguéis, C. Felix, and C. Rodrigues. 2021. "MACHINE LEARNING AND COINTEGRATION FOR STRUCTURAL HEALTH MONITORING OF A MODEL UNDER ENVIRONMENTAL EFFECTS."

Rosales, M. J., and R. Liyanapathirana. 2017. "Data Driven Innovations in Structural Health Monitoring." *Journal of Physics: Conference Series* 842 (1). https://doi.org/10.1088/1742-6596/842/1/012012.

Shiffman, D. 2012. *The Nature of Code*. Edited by Shannon Fry.

Shrivastava, S. 2020. "Cross Validation in Time Series." 2020. https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4.

Silva, M. F. M. Da. 2017. "Machine Learning Algorithms for Damage Detection in Structures under Changing Normal Conditions."

Slišković, D., R. Grbić, and Ž. Hocenski. 2012. "MULTIVARIATE STATISTICAL PROCESS MONITORING."

Songhao, W. 2020. "3 Best metrics to evaluate Regression Model?" 2020. https://towardsdatascience.com/what-are-the-best-metrics-to-evaluate-your-regression-model-418ca481755b.

Spencer, B. F., V. Hoskere, and Y. Narazaki. 2019. "Advances in Computer Vision-Based Civil 581 Infras." In *Engineering*, 5th ed., 199–222.

Sutton, R. S., and A. G. Barto. 2014. *Reinforcement Learning : An Introduction*. 2nd ed. London, England.

Tan, P.-N., M. Steinbach, and V. Kumar. 2014. *Introduction to Data Mining*. Limited, Pearson Education.

Tibaduiza, D. A., L. E. Mujica, and J. Rodellar. 2011. "Structural Health Monitoring Based on Principal Component Analysis: Damage Detection, Localization and Classification." *Advances in Dynamics, Control, Monitoring and Applications, Universitat Politècnica de Catalunya, Departament de Matemàtica Aplicada* 3 (1): 8–17.

Tomé, E. 2019. "Smart Structural Health Monitoring to Management and Conservation of Bridges."

Tomé, E., M. Pimentel, and J. Figueiras. 2019. "Damage Detection under Environmental and Operational Effects Using Cointegration Analysis–Application to Experimental Data from a Cable-Stayed Bridge." *Structural Health Monitoring 2019*.

———. 2020. "Damage Detection under Environmental and Operational Effects Using Cointegration Analysis–Application to Experimental Data from a Cable-Stayed Bridge." *Mechanical Systems and Signal Processing* 135: 106386.

Wipf, T. J., B. Phares, J. D. Doornink, L. Greimann, and D. L. Wood. 2007. "Evaluation of Steel Bridges , Volumes I & II."

Xu, Y. L., and Y. Xia. 2012. *Structural Health Monitoring of Long-Span Suspension Bridges*. 1st ed.

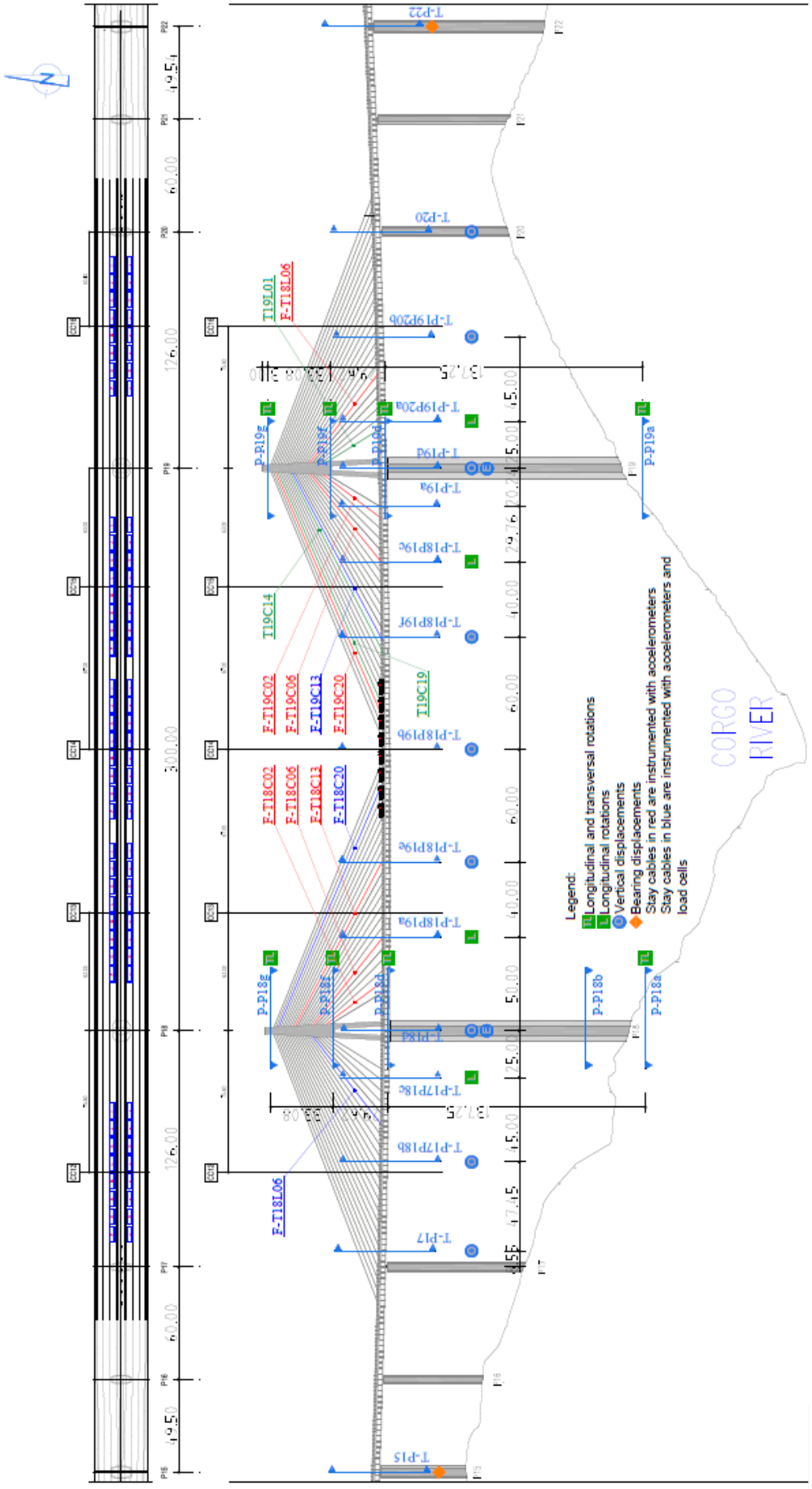
Yan, A. M., G. Kerschen, P. De Boe, and J. C. Golinval. 2005. "Structural Damage Diagnosis under Varying Environmental Conditions - Part I: A Linear Analysis." *Mechanical Systems and Signal Processing* 19 (4): 847–64. https://doi.org/10.1016/j.ymssp.2004.12.002.

Yasarer, H., and Y. Najjar. 2014. “Assessing the Auto Associative Network Approach for Prediction in Civil Engineering Databases.” *Procedia Computer Science* 36 (C): 618–22. https://doi.org/10.1016/j.procs.2014.09.064.

Yuen, K. V., and H. Q. Mu. 2012. “A Novel Probabilistic Method for Robust Parametric Identification and Outlier Detection.” *Probabilistic Engineering Mechanics* 30: 48–59. https://doi.org/10.1016/j.probengmech.2012.06.002.

Zhang, Y., and H. V. Burton. 2019. “Pattern Recognition Approach to Assess the Residual Structural 605 Capacity of Damaged Tall Buildings.” *Structural Safety*, 12–22.

Appendix A: Side elevation of the Central Sub-Viaduct, location of the instrumented sections of the CSV



Appendix B: Daily temperatures recorded

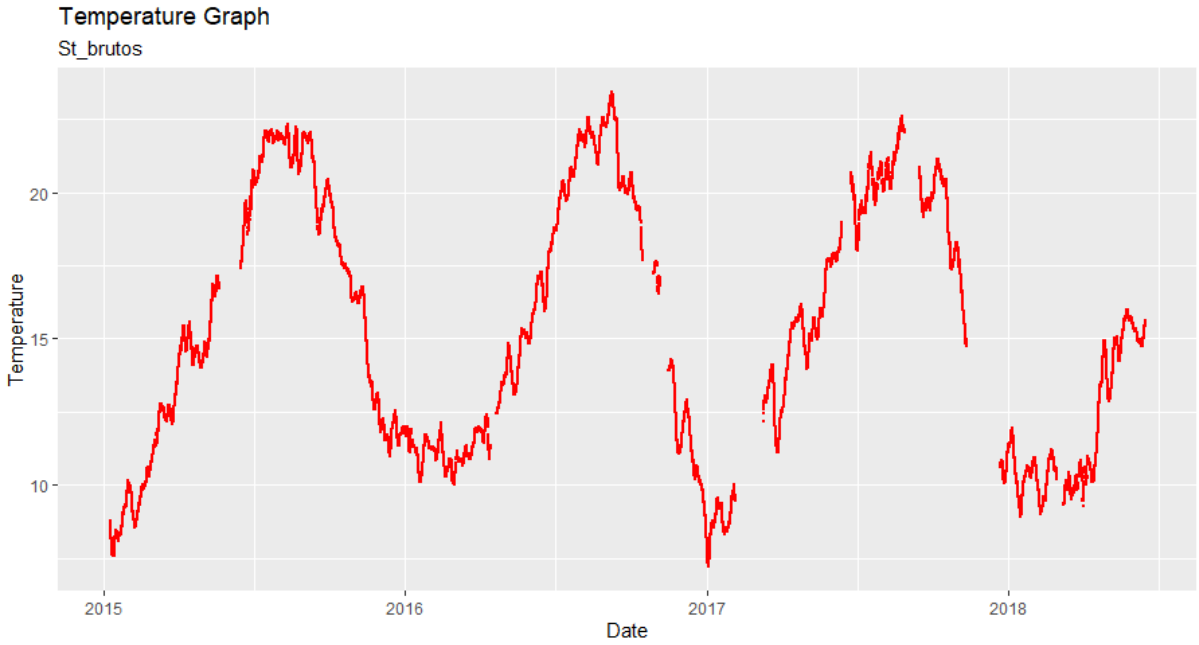


Figure 6-2 - Daily average temperature recorded by - ST_T_P27_2 sensor.

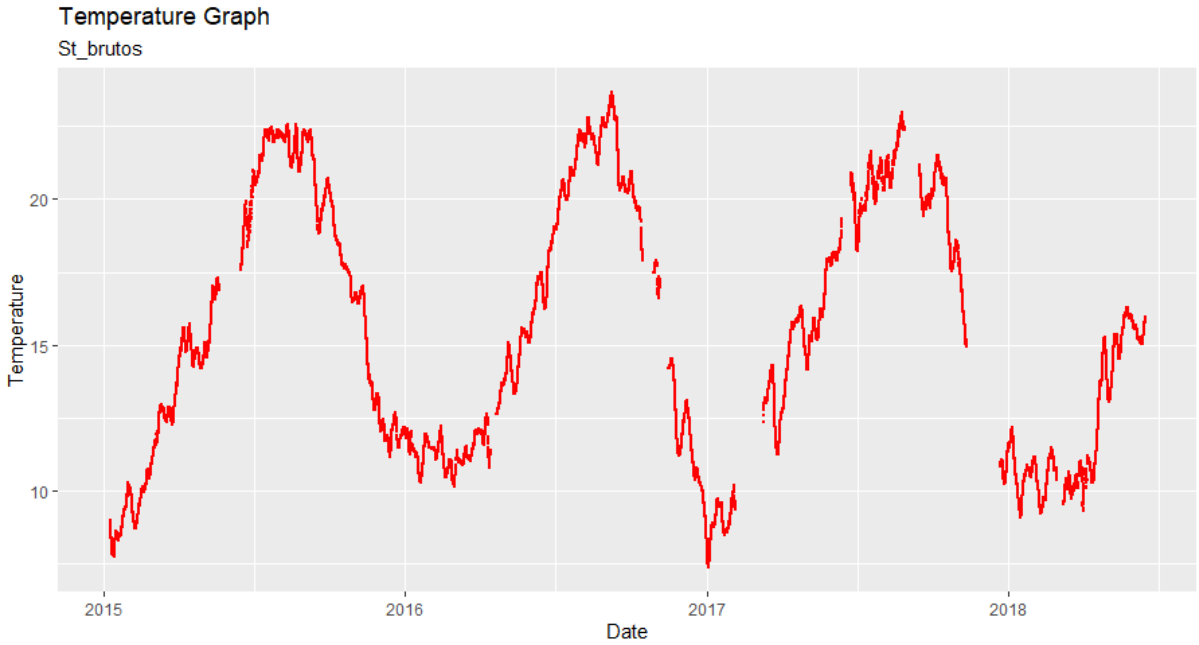


Figure 6-1 - Daily average temperature recorded by - ST_T_P27_1 sensor.

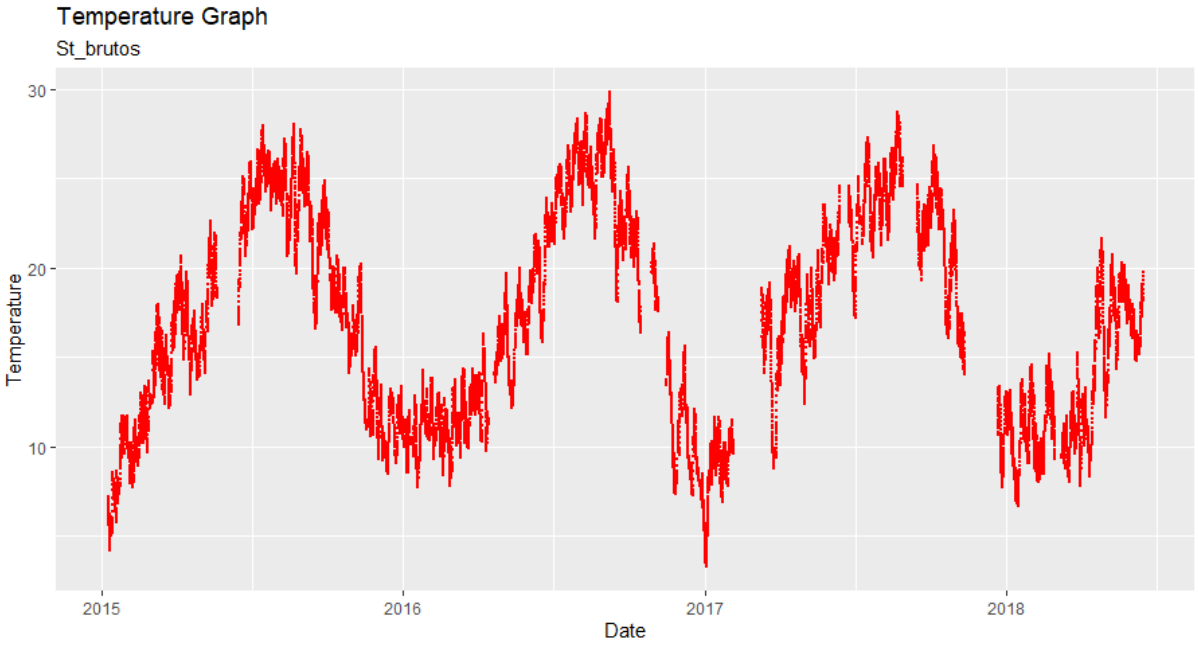


Figure 6-4 - Daily average temperature recorded by - ST_T_P27_4 sensor.

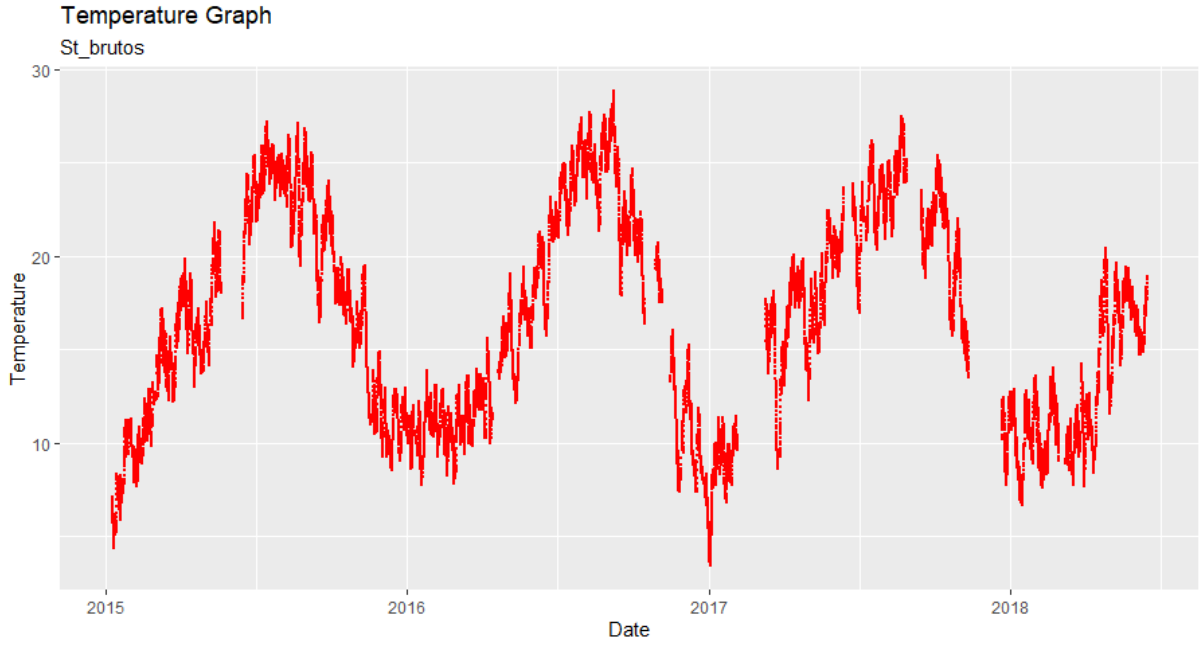


Figure 6-3 - Daily average temperature recorded by - ST_T_P27_3 sensor.

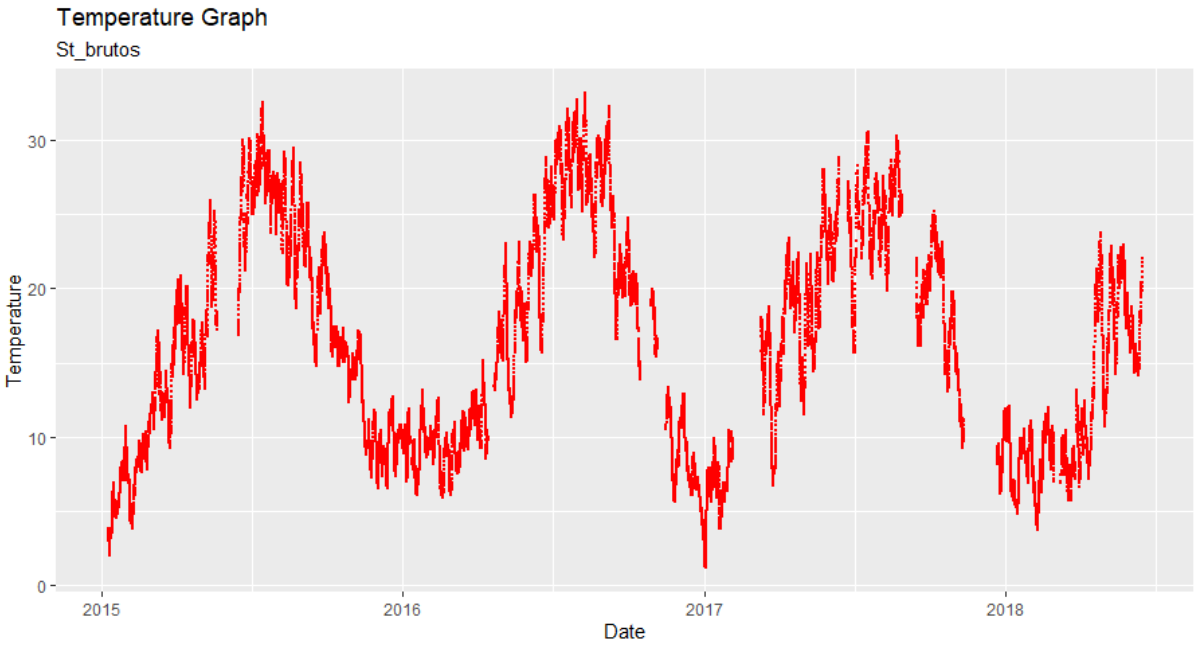


Figure 6-6 - Daily average temperature recorded by - ST_T_P26_1I sensor.

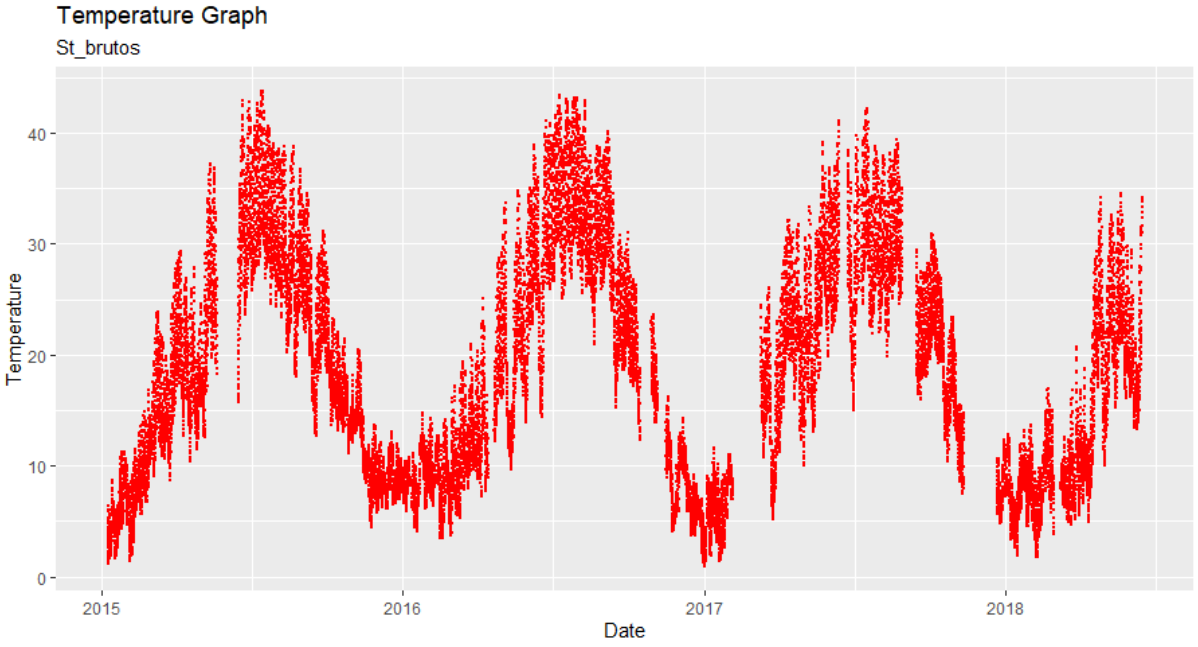


Figure 6-5 - Daily average temperature recorded by - ST_T_P26_1S sensor.

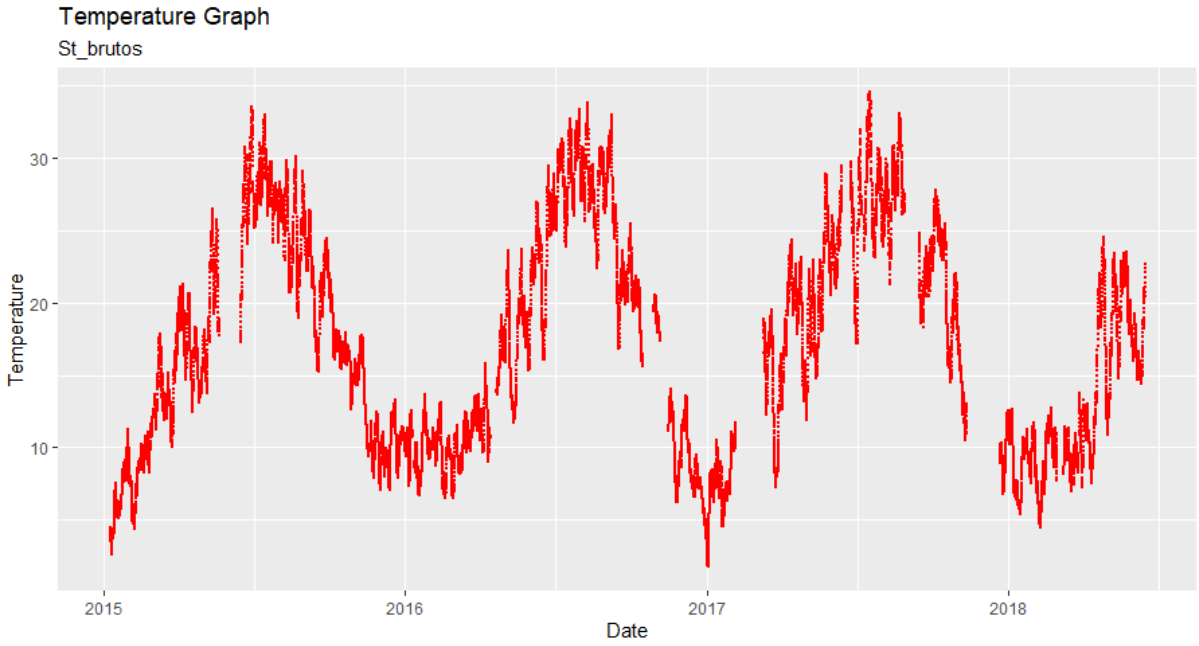


Figure 6-7 - Daily average temperature recorded by - ST_T_P26_2I sensor.

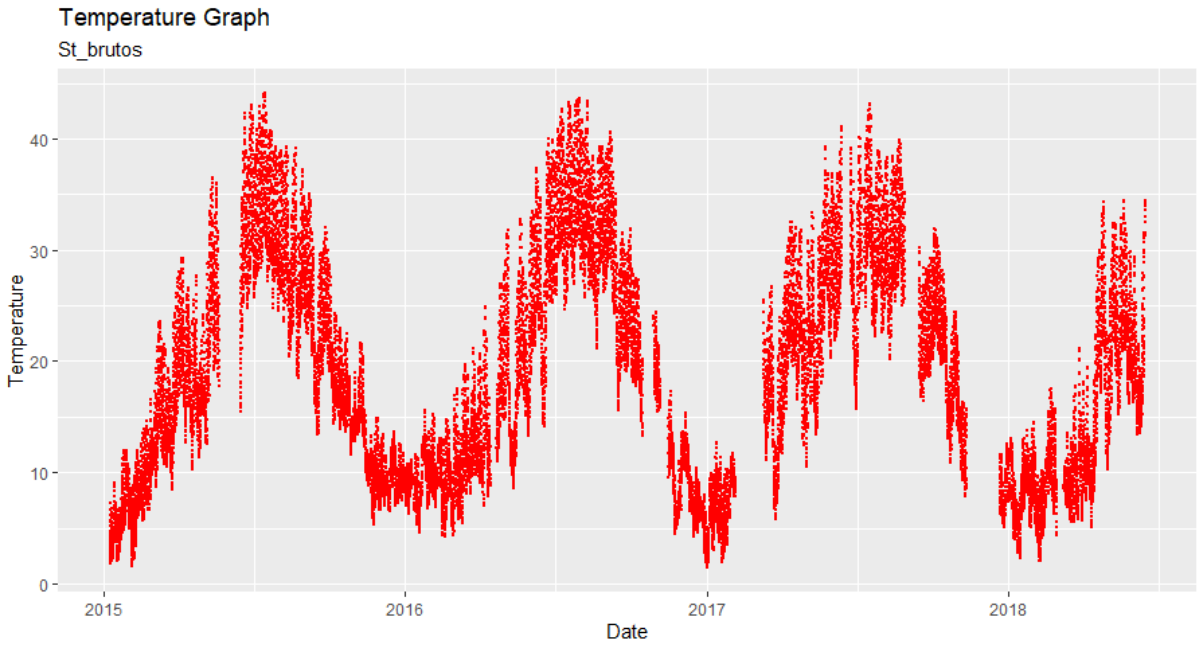


Figure 6-8 - Daily average temperature recorded by - ST_T_P26_2S sensor.

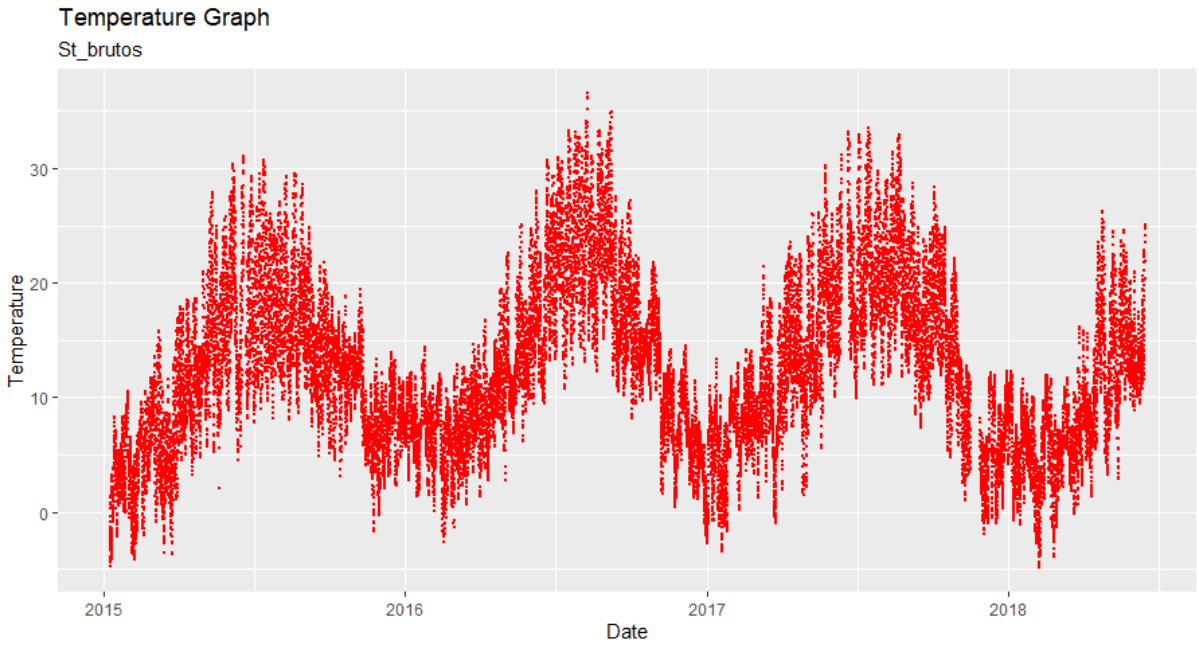


Figure 6-9 - Daily average temperature recorded by - ST_TT19_C13 sensor.

Appendix C: Daily averaged experimental time series with a simulated damage scenario

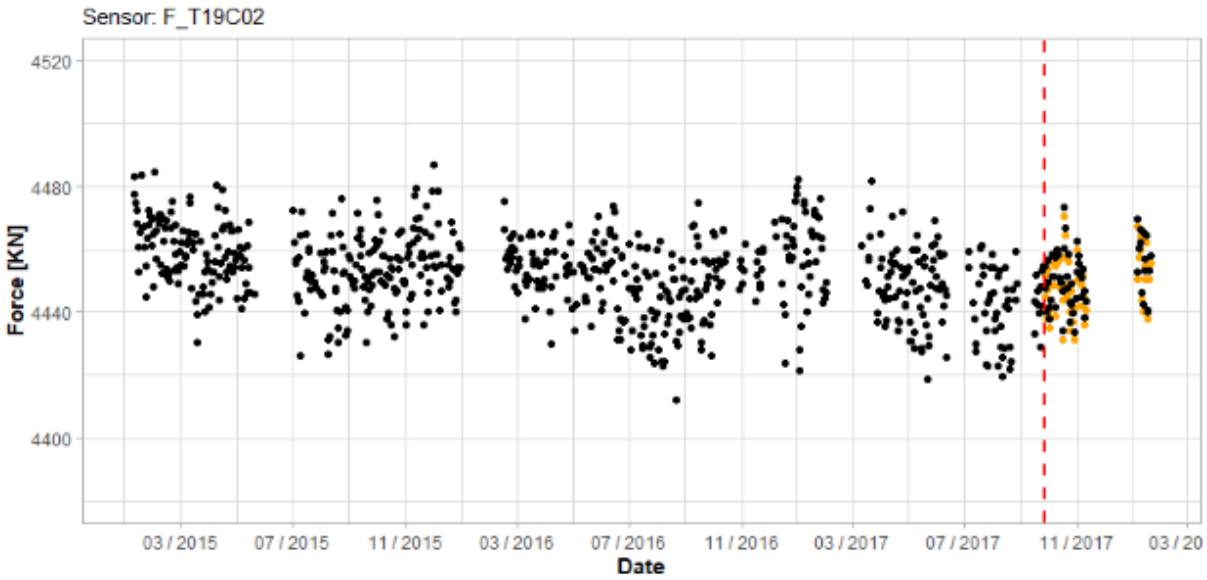


Figure 6-10 - Daily averaged experimental time series with a simulated damage scenario corresponding to 10% of area reduction in the stay cable T19C02 introduced in 26/09/2017 (vertical dashed line).

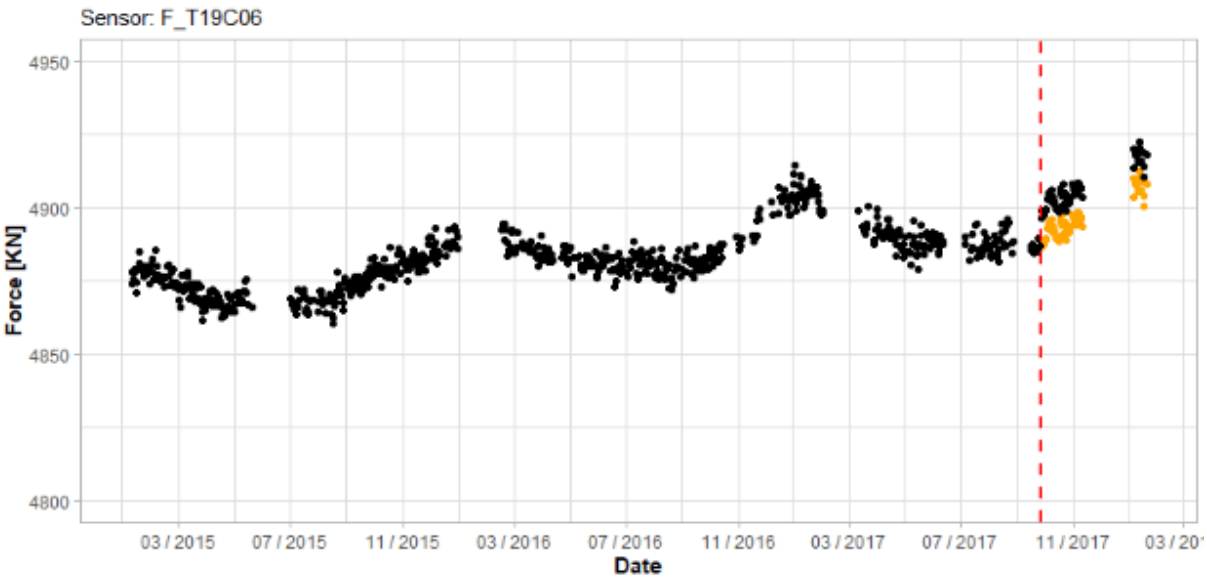


Figure 6-11 - Daily averaged experimental time series with a simulated damage scenario corresponding to 10% of area reduction in the stay cable T19C06 introduced in 26/09/2017 (vertical dashed line).

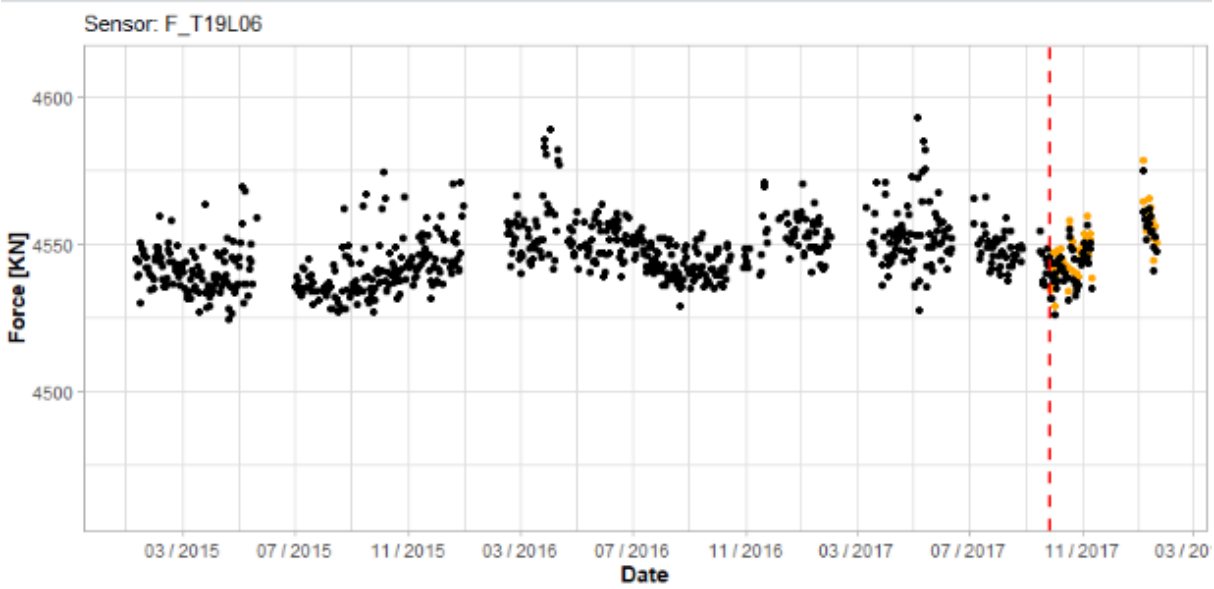


Figure 6-13 - Daily averaged experimental time series with a simulated damage scenario corresponding to 10% of area reduction in the stay cable T19L06 introduced in 26/09/2017 (vertical dashed line).

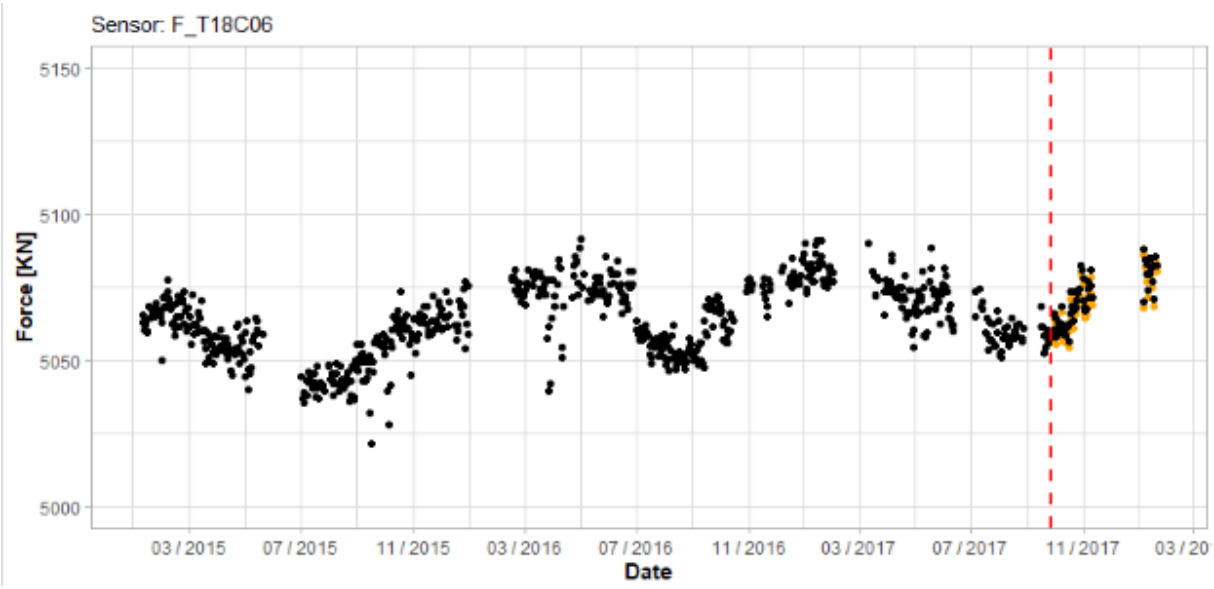


Figure 6-12 - Daily averaged experimental time series with a simulated damage scenario corresponding to 10% of area reduction in the stay cable T18C06 introduced in 26/09/2017 (vertical dashed line).

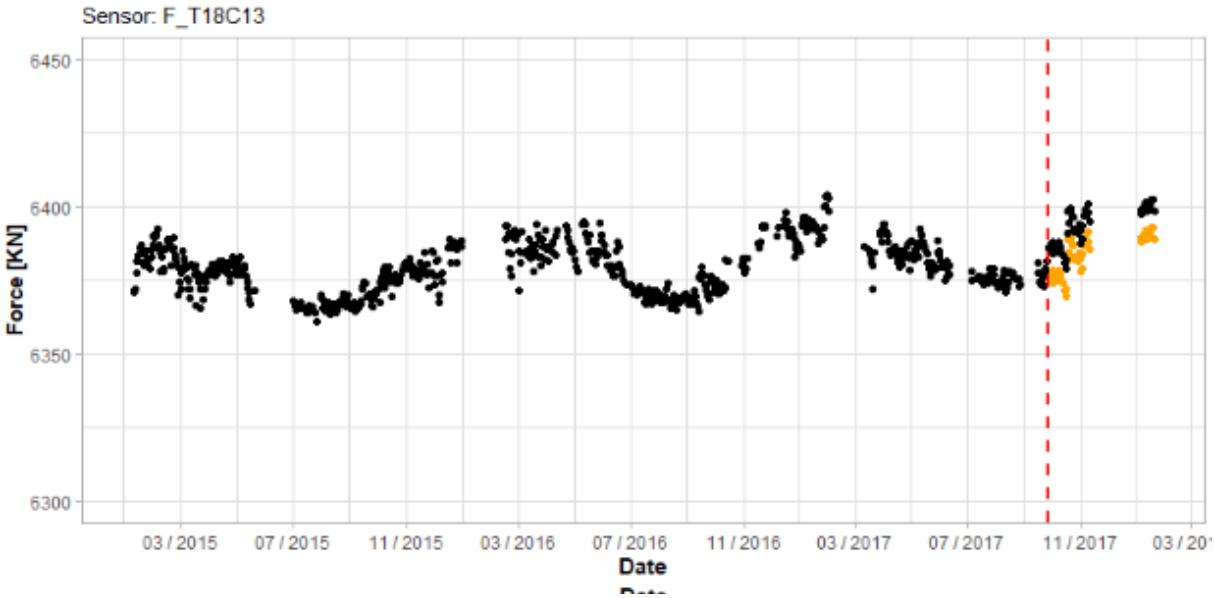


Figure 6-16 - Daily averaged experimental time series with a simulated damage scenario corresponding to 10% of area reduction in the stay cable T18L06 introduced in 26/09/2017 (vertical dashed line).

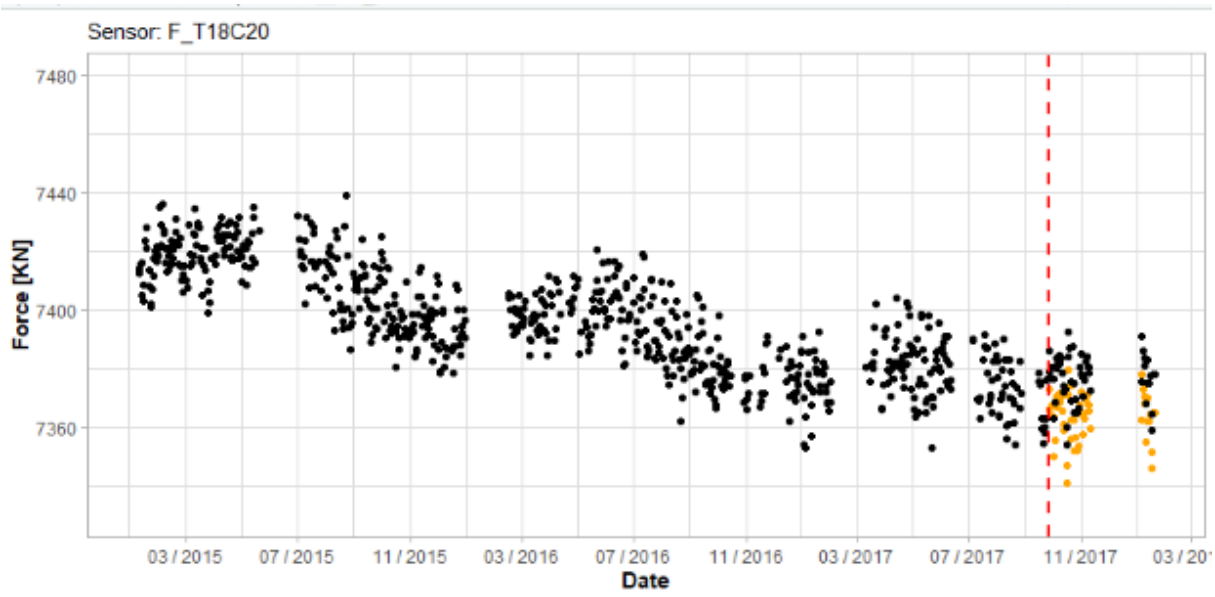


Figure 6-15 - Daily averaged experimental time series with a simulated damage scenario corresponding to 10% of area reduction in the stay cable T18C20 introduced in 26/09/2017 (vertical dashed line).

Appendix D: MSE during the false positive detection period and during the anomaly detection period – Multi-output Neural Network

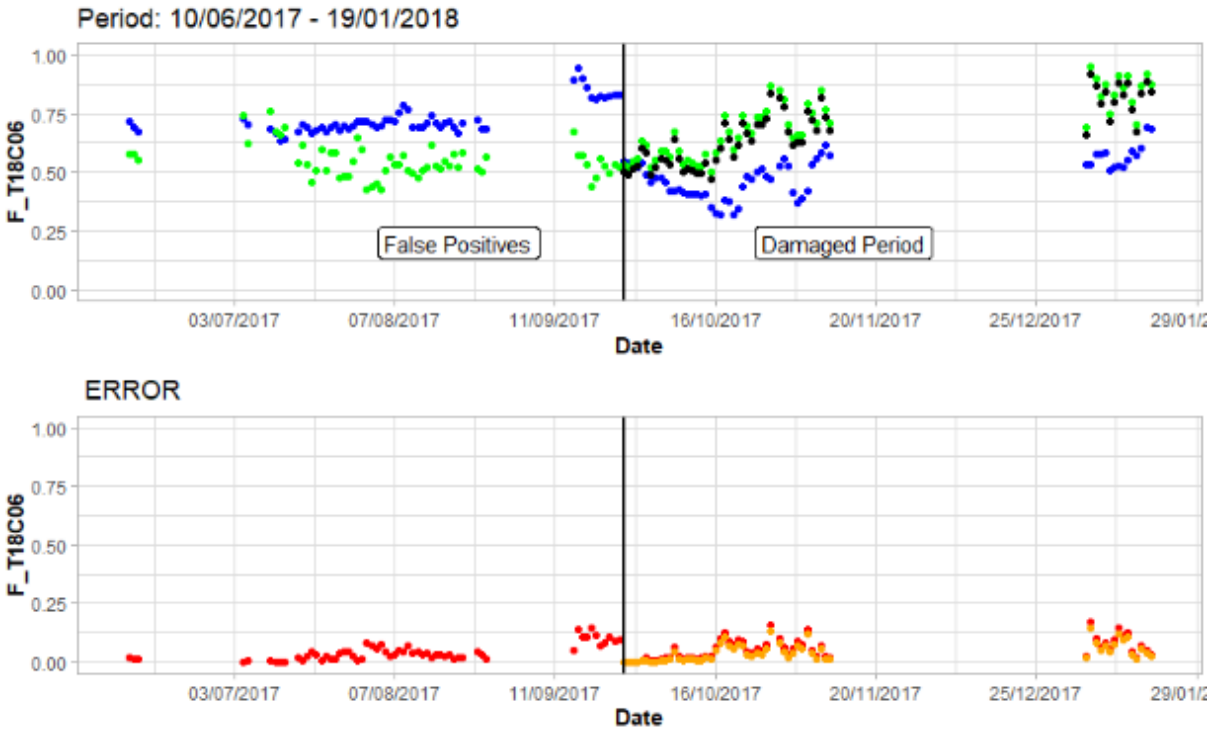


Figure 6-18 - Predicted values *vs* observed values of stay-cable T18C06. Period: False positives detection period + Damage detection period.

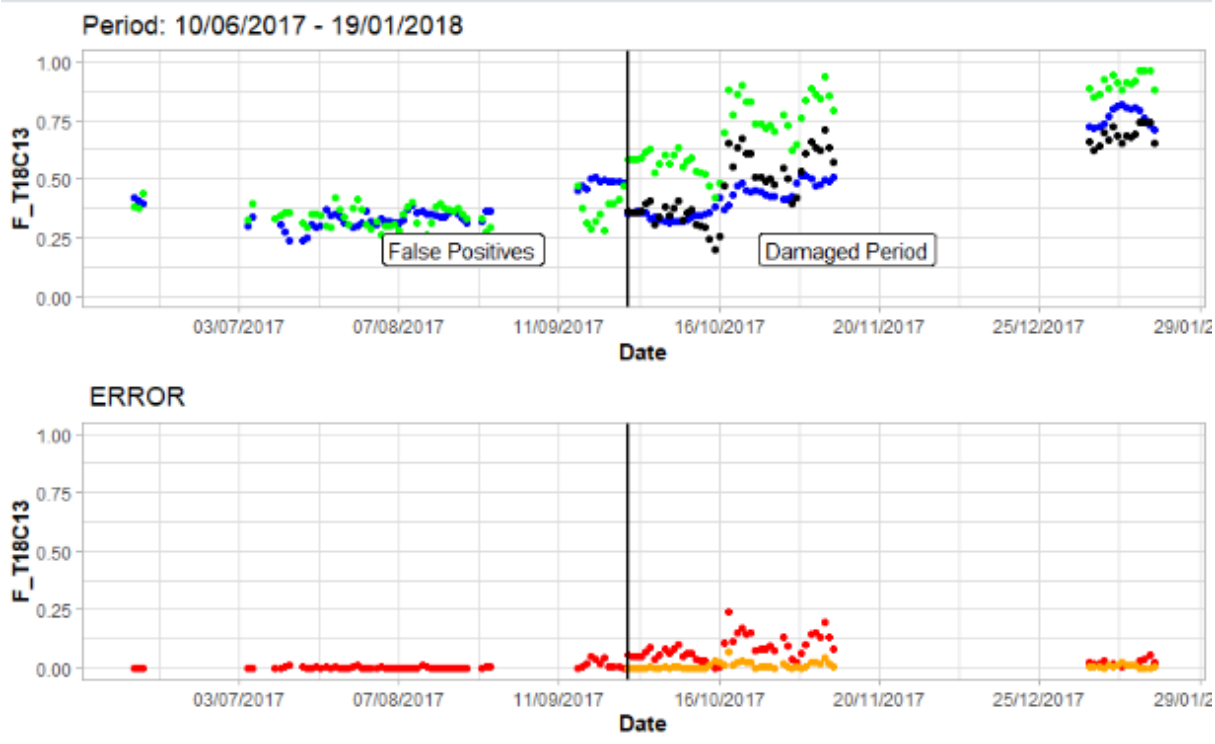


Figure 6-17 - Predicted values vs observed values of stay-cable T18C13. Period: False positives detection period + Damage detection period.

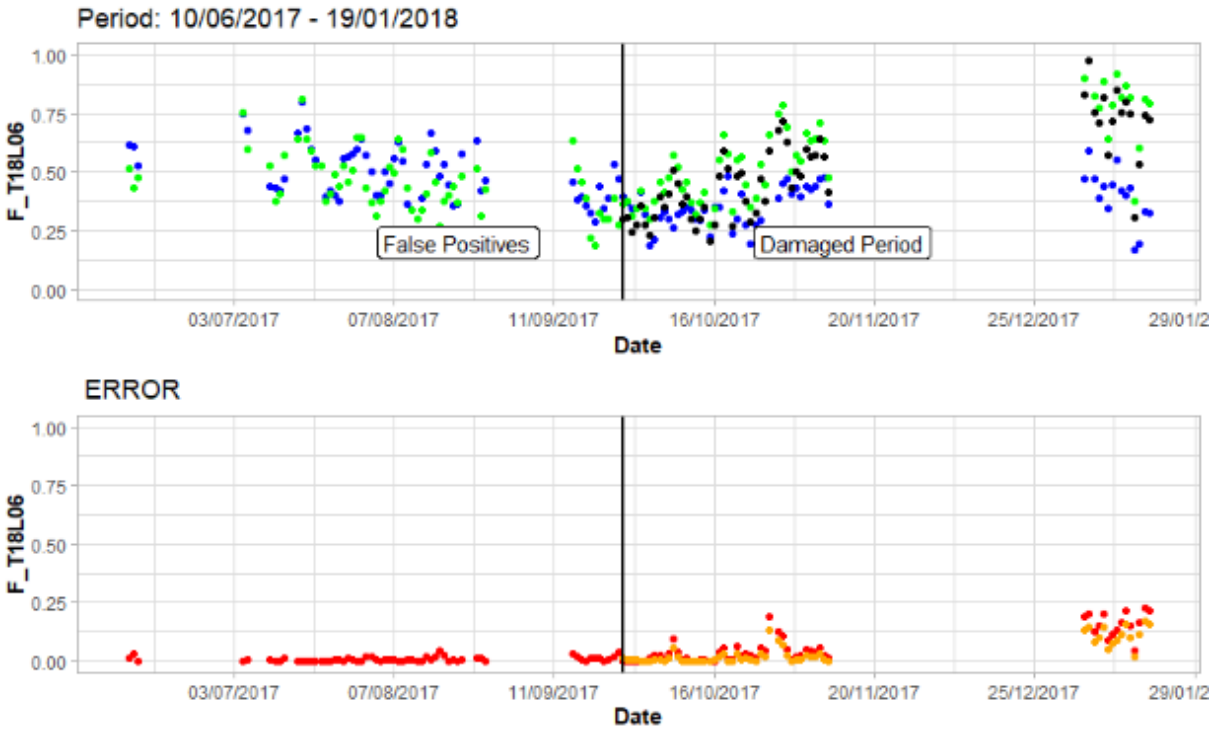


Figure 6-19 - Predicted values vs observed values of stay-cable T18L06. Period: False positives detection period + Damage detection period.

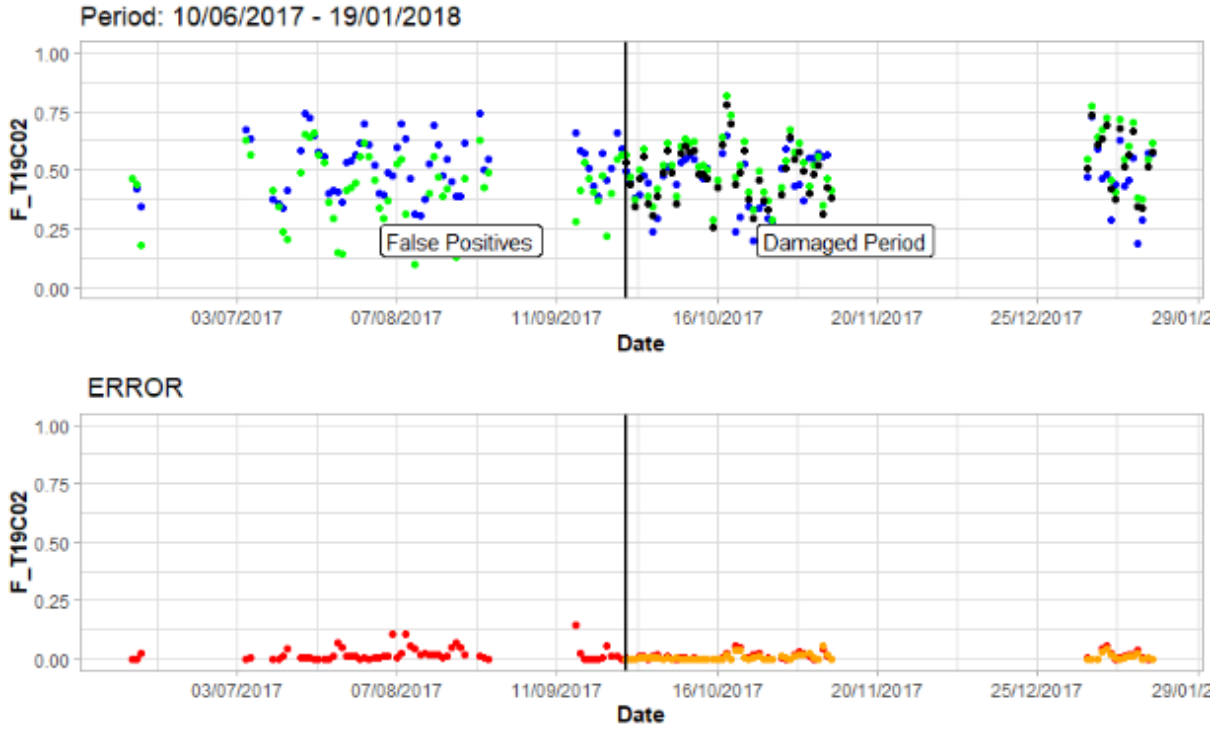


Figure 6-20 - Predicted values vs observed values of stay-cable T19C02. Period: False positives detection period + Damage detection period.

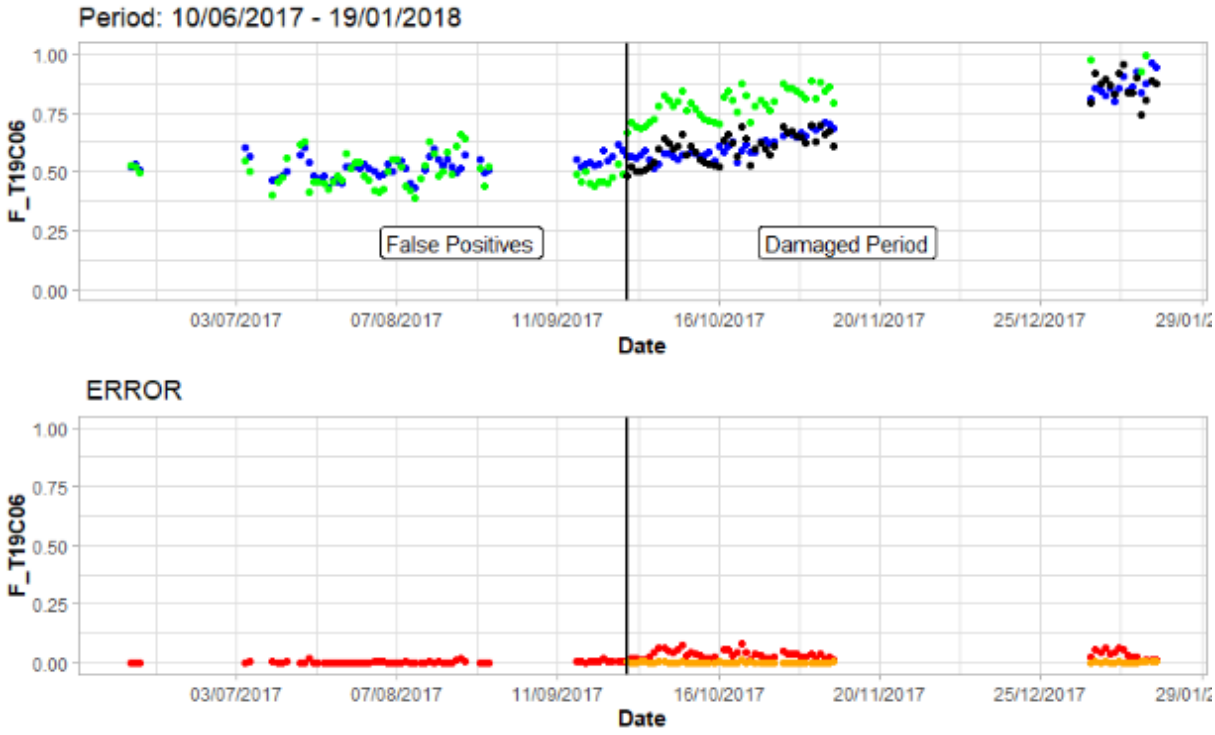


Figure 6-21 - Predicted values vs observed values of stay-cable T19C06. Period: False positives detection period + Damage detection period.

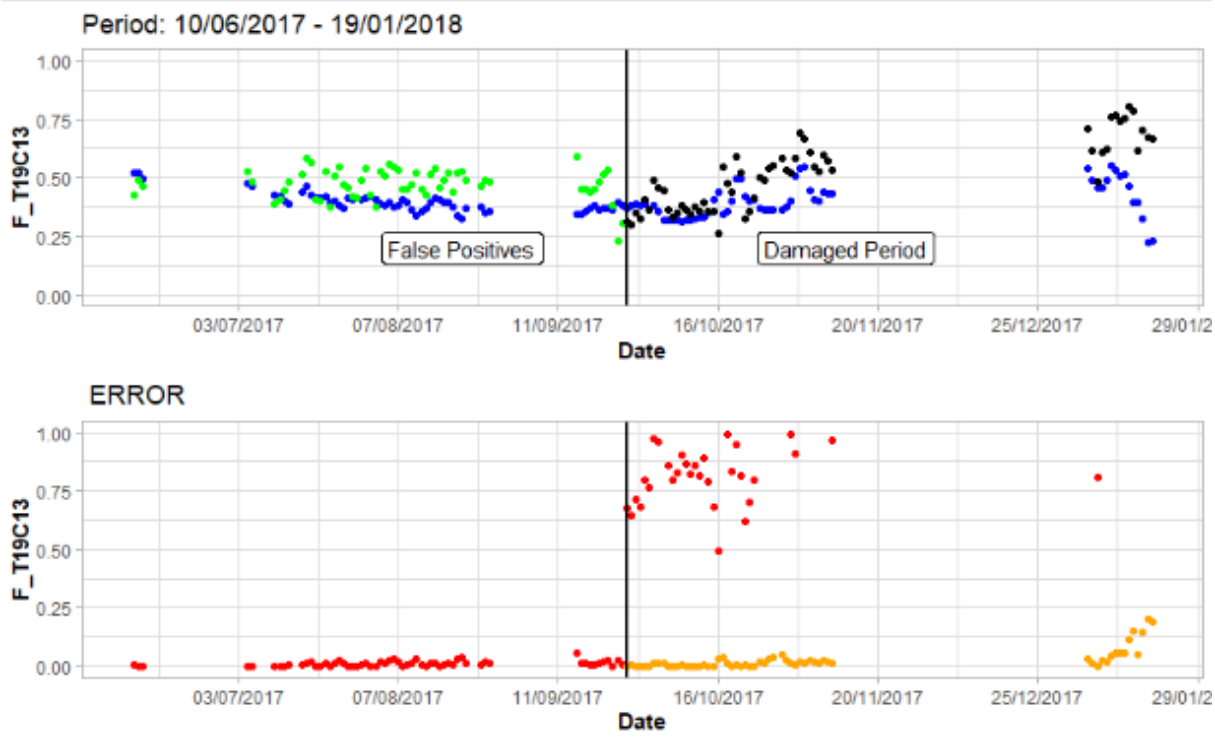


Figure 6-22- Predicted values vs observed values of stay-cable T19C13. Period: False positives detection period + Damage detection period.

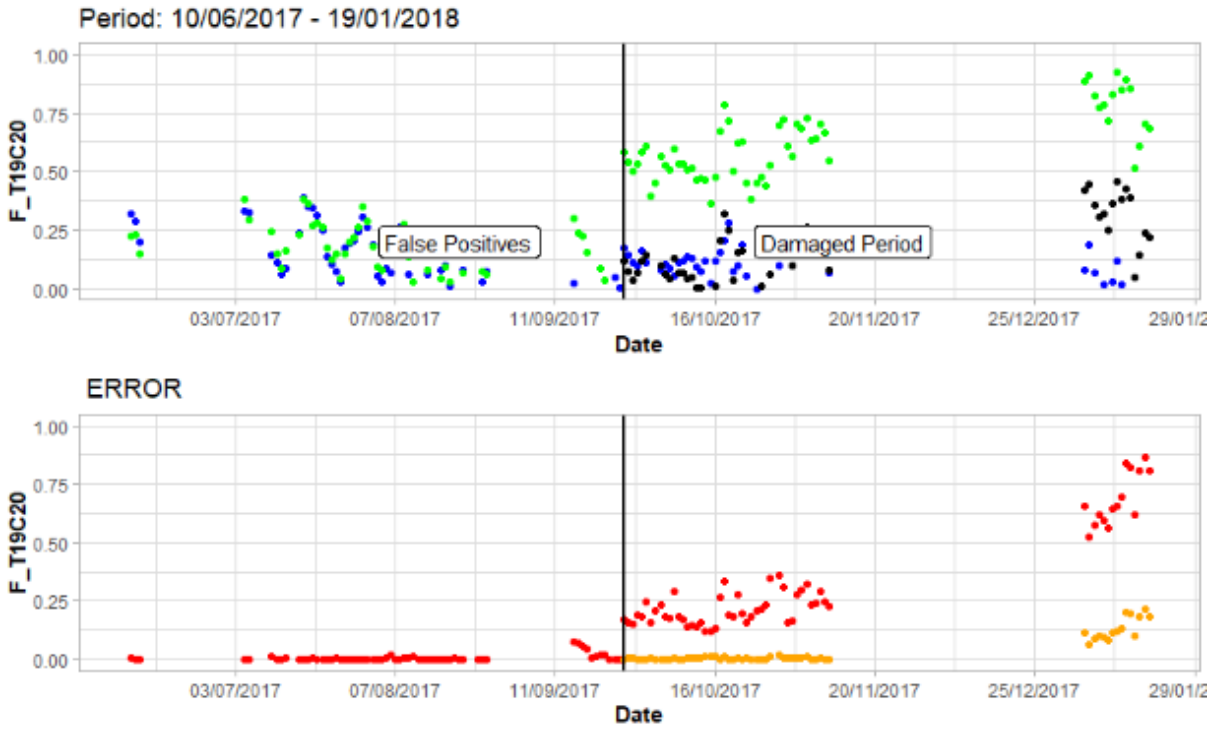


Figure 6-24 - Predicted values vs observed values of stay-cable T19C20. Period: False positives detection period + Damage detection period.

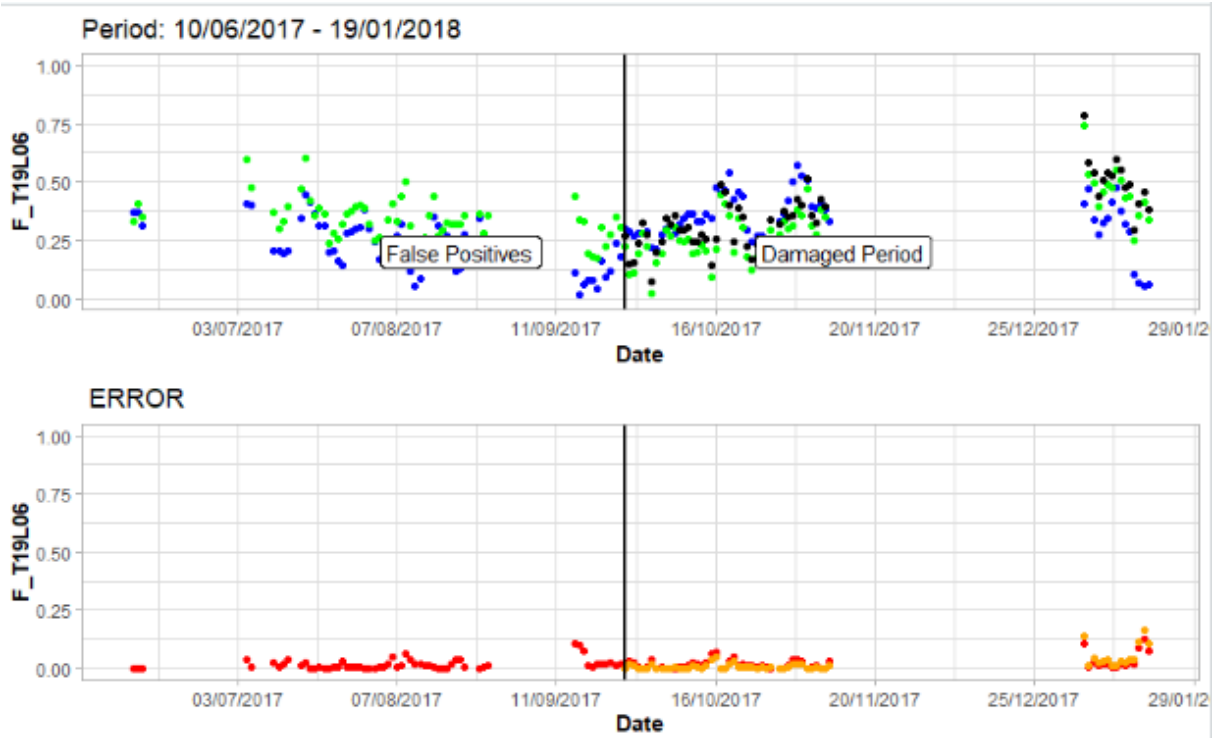


Figure 6-23 - Predicted values vs observed values of stay-cable T19L06. Period: False positives detection period + Damage detection period.