COBEM 2023 27TH INTERNATIONAL CONGRESS OF MECHANICAL ENGINEERING

ABCM Brazilian Society of Mechanical Sciences and Engineering
27th ABCM International Congress of Mechanical Engineering
December 04-08, 2023, Florianópolis-SC, Brazil

COB-2023-0216
MACHINE LEARNING APPROACH FOR STRUCTURAL HEALTH MONITORING USING DECISION TREES AND XGBOOST

Heitor Nunes Rosa
São Paulo State University, Enginneering School of Ilha Solteira, Ilha Solteira, São Paulo, Brazil
h.rosa@unesp.br
Elói João Faria Figueiredo
Lusófona University, Lisboa, Portugal
eloi.figueiredo@ulusofona.pt
Samuel da Silva
São Paulo State University, Enginneering School of Ilha Solteira, Ilha Solteira, São Paulo, Brazil
samuel.silva13@unesp.br

Abstract. *Structural Health Monitoring (SHM) is an essential strategy for detecting damage in aerospace, civil, or mechanical engineering infrastructure. This paper proposes a machine learning approach using XGBoost, a Decision Tree based supervised nonparametric algorithm for regression and classification tasks. An important aspect of the algorithm is its capability to extract Information Gain as a feature to accomplish the classification task, which we used to evaluate the sensors' damage sensitivity. Our study was conducted on the Z24 bridge benchmark, considering two types of health conditions, a healthy and a damaged one. To measure the bridge response, eight sensors were attaches in different locations of the bridge. Feature Extraction was carried out considering statistical Time Domain Features and Natural Frequencies. We achieved satisfactory classification results with 0.943 Average Precision Score. A detailed feature importance analysis identifies specific sensor attributes, such as the second Natural Frequency and certain sensor tests, are significantly sensitive to damage detection. These findings opens possibilities of selecting fundamental features to ensure faster and more robust data processing for damage classification.*

Keywords: *Structural Health Monitoring, Decision Tree Learning, XGBoost, Feature Importance, Information Gain.*

1. INTRODUCTION

Structural Health Monitoring (SHM) consists of the process of implementing a damage detection strategy for aerospace, civil or mechanical engineering infrastructure. It involves the observation of a structure or mechanical system over time using periodically spaced dynamic response measurements, the extraction of damage-sensitive features from these measurements, and the statistical analysis of these features to determine the current state of system health. Not only the classification of the health condition of a system is essential, but also the damage localization, damage quantification, and remaining life prediction.

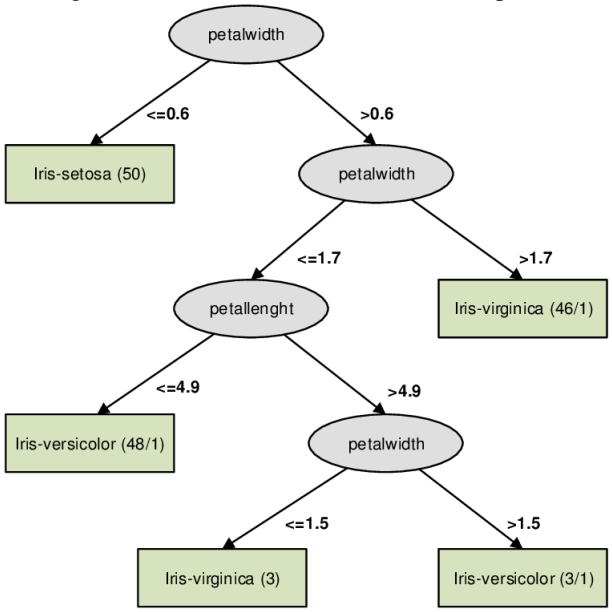
To accomplish its goals, an SHM system must be trained to recognize patterns, a mature discipline in the machine learning study field (Entezami *et al.*, 2019). One of the machine learning algorithms is the Decision Tree. The Decision Tree Algorithm has been used in SHM (Joshuva and Sugumaran, 2016), but it is mainly applied as a base for ensemble methods, averaging, and boosting. The principle of averaging is to create multiple trees independently and average their predictions; meanwhile, boosting builds trees sequentially, aiming to reduce bias error. Mariniello *et al.* (2021) used an averaging approach to detect and localize damage into a tridimensional steel truss frame. Dong *et al.* (2020) used boosting approach to predict concrete electrical resistivity values, Wang and Song (2020) to detect bolt-looseness with a percussion-based method and Leon-Medina *et al.* (2021) to diagnose wind-turbine foundations health.

In the present text, its first discussed what is a Decision Tree and which criteria is used to accomplish a classification task. Then, its explained how the algorithm is trained and its shortcomings, as well as how to avoid them. Next, Feature Importance and its evaluation by a Decision Tree are discussed. After the concepts introduction, the structure which will be used for this study is described, followed by the approaches taken to extract features from the registered sensors outputs. The results are then presented, discussing the algorithm capabilities in classification tasks and to inform feature importances and sensor sensitivity to damage.

2. DECISION TREES

A Decision Tree is a supervised nonparametric algorithm used for regression and classification tasks. The objective is to create a model to predict an outcome (terminal or leaf node) based on subsequential logical tests (internal nodes) resembling a tree, which gives rise to the algorithm's name Breiman *et al.* (2017). For example, The algorithm can classify an iris flower species considering its petal and sepal dimensions, as shown in Fig 1.

Figure 1. Classification of the iris flower species.



Source: (Grabusts *et al.*, 2015)

A Decision Tree grows greedily by selecting the best split in an internal node for all dataset features. The best split is based on the gini impurity criterion for classification tasks and mean squared error for regression tasks.

2.1 Gini Impurity

Gini Impurity is a measure that quantifies the variability of a set. It can be computed by summing the probability of an item with the label being chosen times the probability of this item not being chosen. It reaches its minimum (zero) when all cases in the node fall into a single target category Ziegel (2003). The Equation (1) express this relation, where m is the number of classes, and p_i is the probability of selecting an item, calculated by its frequency in the set.

$$GI = \sum_{i=1}^{m} p_i(1-p_i) = \sum_{i=1}^{m} p_i - \sum_{i=1}^{m} p_i^2 = 1 - \sum_{i=1}^{m} p_i^2 \tag{1}$$

To evaluate the gini impurity of a given split, the algorithm uses a loss function, which is a weighted average of the resulting left and right sets expressed in the eq. 2, where n, n_L, n_R, GI_L, and GI_R are, respectively, the total number of observations, the number of observations in the left and right sets, and the Gini Impurity in the left and right sets.

$$L = \frac{n_L}{n} GI_L + \frac{n_R}{n} GI_R \tag{2}$$

Ideally, a Decision Tree should have leaf nodes with one class ($L = 0$), resulting in a perfect classification.

2.2 Tree Growth

Firstly, the algorithm evaluates all the thresholds of all features and selects the one that reduces the loss function the most, splitting the set into two. Then, the procedure continues to the new internal nodes until the loss function is zero, in other words, until it has terminal nodes with only one class. The algorithm is greedy because it makes a locally optimal split, not ensuring a globally optimal decision tree.

Although a decision tree creates complex structures, if not appropriately trained, it might lead to overfitting, making predictions outside the training data inaccurate. This is due to the objective to reduce the loss function to zero, which makes the tree overly specific and does not generalize data well. To avoid this problem, limits must be put before its training, called hyperparameters.

Hyperparameters can define certain aspects of the tree, such as how many splits can occur, the minimum number of observations to perform a split, the maximum number of leaves or terminal nodes, etc. The *scikit-learn* Pedregosa *et al.* (2011) Decision Tree has plenty implemented, being the most important:

- **max_depth:** The maximum depth of the tree.
- **min_samples_split:** The minimum number of observations required to split an internal node
- **min_samples_leaf:** The minimum number of observations required to be at a leaf node.
- **max_leaf_nodes:** The maximum number of leaf nodes.
- **min_impurity_decrease:** A node will be split if this split induces a decrease of the impurity greater than or equal to this value. In other words, it changes the objective to reduce the loss function from zero to another value.

The hyperparameter tuning will improve the model's results, but only to a certain degree. The Decision Tree must be implemented with an ensemble (Random Forest, XGBoost, Adaboost, etc.) to mitigate its limitations.

3. GRADIENT BOOSTING

Gradient Boosting is a *ensemble* algorithm inspired by the gradient descent optimization method, which directs the search for the minimum of the problem's cost function to the largest possible increment in the attribute vector and multiplies it by a negative learning rate. Similarly, the *Gradient Boosting* algorithm improves its performance based on the residual terms of the previous trees (Friedman, 2001).

3.1 *XGBoost*

XGBoost, or *Extreme Gradient Boosting*, is an improvement of the *Gradient Boosting* algorithm, introducing a regularization term and expanding the gradient to the second order (Hessian), increasing the control of model complexity and its convergence to a optimal result. In addition to these changes, the use of sparse matrices and improved data structures was added for better data processing in training and result prediction, drastically reducing the computational cost (Chen and Guestrin, 2016).

4. FEATURE IMPORTANCE

One of the most important aspects of monitoring a structure is extracting relevant features of the raw data to make assumptions regarding its health. However, it is possible that many extracted features do not add new information when classifying a structure condition. The Decision Tree Algorithm has a particular method to evaluate the importance of a feature, in other words, the Information Gain. Given a trained Decision Tree, the Information Gain of a feature in one node is the difference between the node's gini impurity and the gini impurity weighted sum of the resulting nodes after the split, or child nodes, as presented in Eq. (3). Then, for each feature, The Information Gain is summed in all nodes considering the number of observations in each node, resulting in the Eq. (4). The results are then normalized, so that sum of Feature Importances is 1.

$$IG(f_i, node) = GI(f_i, node) - \sum_{v=1}^{2} \frac{n_v}{n_{node}} GI(f_i, child_v) \tag{3}$$

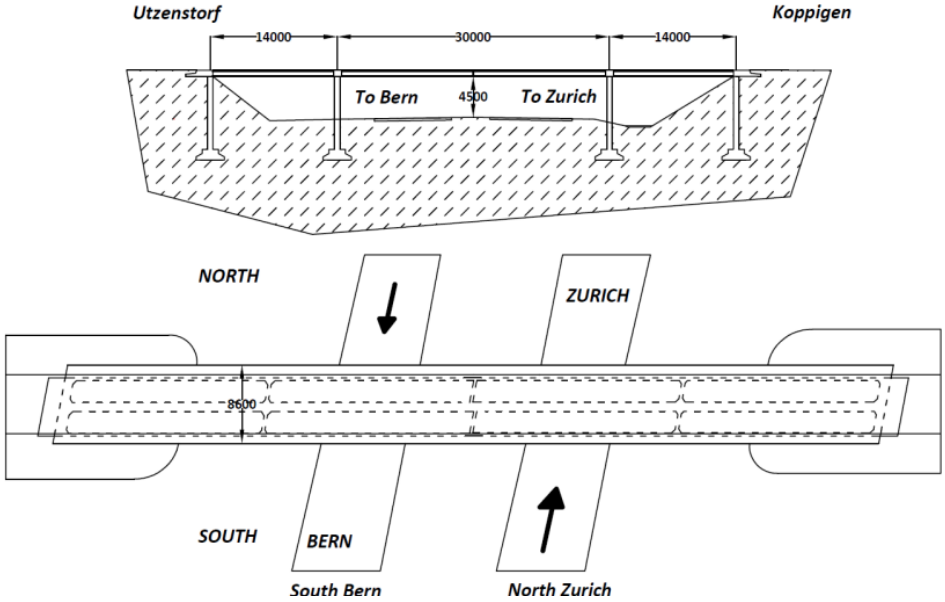
$$IG(f_i) = \sum_{b}^{nodes} \frac{n_b}{n} IG(f_i, b) \tag{4}$$

5. Z-24 BRIDGE

Damage detection, where classification is more than two classes, is considered as a multi- class problem. In this study, two types of damage cases are used, namely, rupture of tendons, and pier settlement of a full-scale bridge, namely, Z24 Bridge. All the damage classes have multiple damage levels. Z24 bridge benchmark data Maeck and De Roeck (2003) is used to evaluate the performance of the proposed method for multiclass damage detection. The bridge was located in the canton Bern near Solothurn, Switzerland. It was a classical post-tensioned concrete two-cell box-girder bridge with a main span of 30 m and two side spans of 14 m, as shown in fig. 2. The bridge was demolished at the end of 1998 because a new railway adjacent to the highway required a new bridge with a larger side span. During the demolition, the bridge data was acquired using 15 accelerometers placed at different spans of the bridge as shown in fig. 3. The bridge was excited by two shakers, one at the mid-span of the bridge and another at a side-span. Because of the size of the bridge, response

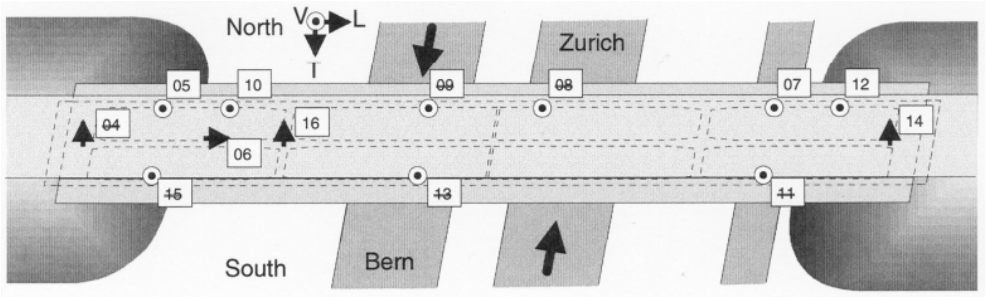
was measured in nine setups of up to 15 sensors each, with three accelerometers and the two force sensors common in all setups. The data was sampled at 100 Hz, and a total of 65536 samples were acquired. This data was made publicly available by researchers at the Katholieke Universiteit Leuven and is available in its website (Reumers, 2022).

Figure 2. Schematic of the Z24 bridge.



Source: (Maeck and De Roeck, 2003)

Figure 3. Sensor placement for data acquisition.



Source: (Maeck and De Roeck, 2003)

6. METHODOLOGY

In order to classify time series, it is necessary to extract features sensitive to damage. For the present work, two approaches were considered: The extraction of Natural Frequencies and the extraction of Statistic Time Domain Features.

To extract natural frequencies, the power spectral density of each sensor was normalized and the result of each normalized density was added for each observation, facilitating the process of obtaining the four natural frquencies. The second approach uses statistical features from a previous exploratory search to find patterns in the dataset. With this approach, it is not required to process the signals exhaustively, being more straightforward to extract features. For this work, it was used Time Domain features present in Tkach *et al.* (2010), which are Variance, Mean Absolute Value, Waveform Length, Skewness, Kurtosis, Amplitude, Mean Crossing and Anderson Test.

The feature extraction was carried out considering a sample size of 65536, obtaining one observation per hour of monitoring. In this way, 5652 observations were obtained, with 4736 healthy observations and 925 damaged ones. A division was made between the training and test sets of 70% and 30% respectively, maintaining the proportion of health conditions in each set. Using the XGBoost Algorithm, the model was trained with the hyperparameters defined according to 1. The model was initially trained considering all features. Feature importance is then extracted from the model.

The Feature Importance is then extracted and analyzed, indicating the features that contribute most to the bridge's health classification. The features were then placed in decreasing order of importance, and the model was then trained and evaluated by adding each of the features to each training session, until the set was composed of all features, it was possible

to evaluate the impact that the addition of features has on the model's performance. Finally, a performance comparison was made considering three conditions, only natural frequencies, only statistical features and all features. The metric used to evaluate the model's performance is the Average Precision Score (AP Score), the area under the Precision-Recall Curve, allowing an independent threshold evaluation, in addition to being more suitable for heavily unbalanced cases than the ROC AUC, as presented by Yuan *et al.* (2015) and Davis and Goadrich (2006).

Table 1. Defined hyperparameters for XGBoost Training

Hyperparameter	**Value**
Max Depth	6
Minimum Sample Leaf	1
Number of Estimators	200
Learning Rate	0.1

7. RESULTS AND DISCUSSION

For the chosen hyperparameters, the achieved AP Score is 0.943. In Fig. 4, features are organized by their importance. Notably, the second Natural Frequency exhibits a significant dominance, followed by the Anderson Test of sensor 7. Subsequent in importance are the mean crossing and Anderson test results of sensor 16, the kurtosis values of sensors 7 and 3, and the first Natural Frequency. These findings point to certain sensor features being more sensitive to damage detection than others, enabling the possibility of selecting the most important features through techniques like Recursive Feature Elimination (Babajanian Bisheh *et al.*, 2019), which can lead to faster and more robust data processing. Furthermore, the notable contribution of Natural Frequencies to the classification highlights the role of spectrum analysis in the frequency domain in achieving superior results.

Figure 4. Features ordered by their Importance.

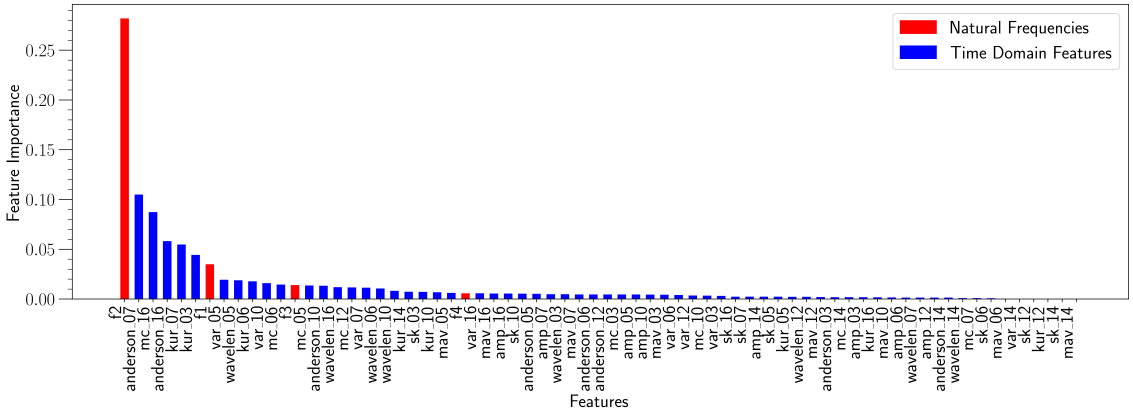


Figure 5 illustrates the variation in the AP Score as features are incrementally incorporated into the model training. Features are added in descending order of importance, starting with the most important. It is apparent that the model achieves its peak performance with the inclusion of thirteen features and does not surpass this performance with the addition of further features. This suggests that leveraging the Feature Importance ranking derived from the XGBoost algorithm can effectively guide feature selection, ultimately enhancing model performance.

Figure 6 depicts Precision-Recall Curves for three different evaluated conditions, demonstrating the trade-off between precision and recall as the classification threshold varies. Corresponding AP Scores can be found in Tab. 2. It is evident that the model utilizing solely natural frequencies provides better coverage compared to the model relying solely on Time Domain Features. Notably, the integration of both feature types results in an improved model.

Table 2. Average Precision scores for different training conditions

Training Conditions	**AP Score**
Only Time Domain Features	0.879
Only Natural Frequencies	0.842
All Features	0.943

Figure 5. Change in performance with addition of features.

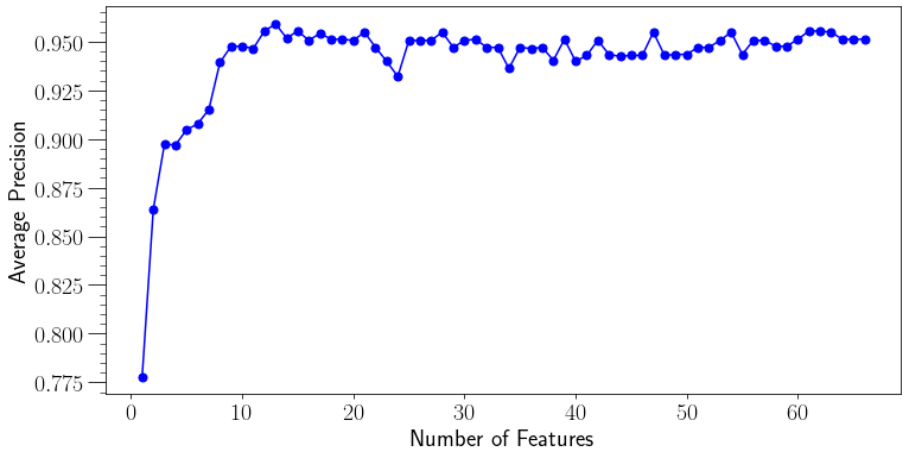
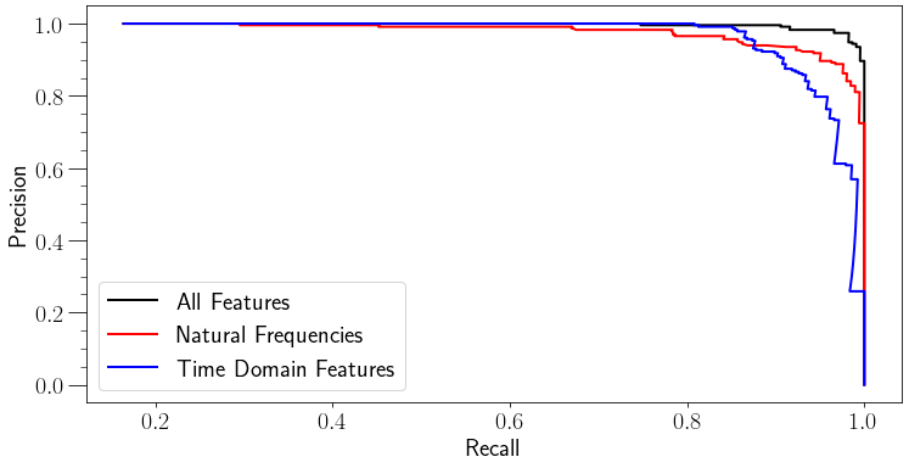


Figure 6. Precision Recall Curves for different training conditions.



8. CONCLUSION

This work addressed the Decision Tree based algorithms applicability in the context of Structural Health Monitoring, more specifically, Bridge Damage Detection. The XGBoost model was able to effectively classify the Bridge damaged Condition, resulting in a Test AP Score of 0.943. The feature importance analysis revealed that certain sensor features, such as the second Natural Frequency and specific sensor Time Domain Features, hold a pronounced sensitivity to damage. These findings offer the potential to optimize data processing by selecting key features. Moreover, the significant contribution of Natural Frequencies underscores the importance of spectrum analysis in the frequency domain for achieving superior classification results. These insights not only enhance our understanding of damage detection but also provide practical strategies for improving the efficiency and robustness of data-driven analysis in this domain.

9. ACKNOWLEDGEMENTS

This work was funded by CAPES under grant number 88887.701388/2022-00.

10. REFERENCES

Babajanian Bisheh, H., Ghodrati Amiri, G., Nekooei, M. and Darvishan, E., 2019. "Damage detection of a cable-stayed bridge using feature extraction and selection methods". *Structure and Infrastructure Engineering*, Vol. 15, No. 9, pp. 1165–1177.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J., 2017. *Classification and regression trees*. Routledge.

Chen, T. and Guestrin, C., 2016. "XGBoost: A scalable tree boosting system". In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '16, pp. 785–794. ISBN 978-1-4503-4232-2. doi:10.1145/2939672.2939785. URL `http://doi.acm.org/10.1145/2939672.2939785`.

Davis, J. and Goadrich, M., 2006. "The relationship between precision-recall and roc curves". In *Proceedings of the 23rd international conference on Machine learning*. pp. 233–240.

Dong, W., Huang, Y., Lehane, B. and Ma, G., 2020. "Xgboost algorithm-based prediction of concrete electrical resistivity for structural health monitoring". *Automation in Construction*, Vol. 114, p. 103155.

Entezami, A., Shariatmadar, H. and Mariani, S., 2019. "Structural health monitoring for condition assessment using efficient supervised learning techniques". *Multidisciplinary Digital Publishing Institute Proceedings*, Vol. 42, No. 1, p. 17.

Friedman, J.H., 2001. "Greedy function approximation: a gradient boosting machine". *Annals of statistics*, pp. 1189–1232.

Grabusts, P., Borisov, A. and Aleksejeva, L., 2015. "Ontology-based classification system development methodology". *Information Technology and Management Science*, Vol. 18. doi:10.1515/itms-2015-0020.

Joshuva, A. and Sugumaran, V., 2016. "Wind turbine blade fault diagnosis using vibration signals through decision tree algorithm". *Indian Journal of Science and Technology*, Vol. 9, No. 48, pp. 1–7.

Leon-Medina, J.X., Anaya, M., Parés, N., Tibaduiza, D.A. and Pozo, F., 2021. "Structural damage classification in a jacket-type wind-turbine foundation using principal component analysis and extreme gradient boosting". *Sensors*, Vol. 21, No. 8, p. 2748.

Maeck, J. and De Roeck, G., 2003. "Description of z24 benchmark". *Mechanical Systems and Signal Processing*, Vol. 17, No. 1, pp. 127–131.

Mariniello, G., Pastore, T., Menna, C., Festa, P. and Asprone, D., 2021. "Structural damage detection and localization using decision tree ensemble and vibration data". *Computer-Aided Civil and Infrastructure Engineering*, Vol. 36, No. 9, pp. 1129–1149.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E., 2011. "Scikit-learn: Machine learning in Python". *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830.

Reumers, P., 2022. "Z24 bridge benchmark". URL `https://bwk.kuleuven.be/bwm/z24`.

Tkach, D., Huang, H. and Kuiken, T.A., 2010. "Study of stability of time-domain features for electromyographic pattern recognition". *Journal of neuroengineering and rehabilitation*, Vol. 7, No. 1, pp. 1–13.

Wang, F. and Song, G., 2020. "Bolt-looseness detection by a new percussion-based method using multifractal analysis and gradient boosting decision tree". *Structural Health Monitoring*, Vol. 19, No. 6, pp. 2023–2032.

Yuan, Y., Su, W. and Zhu, M., 2015. "Threshold-free measures for assessing the performance of medical screening tests". *Frontiers in public health*, Vol. 3, p. 57.

Ziegel, E.R., 2003. "The elements of statistical learning".

11. RESPONSIBILITY NOTICE

The author is solely responsible for the printed material included in this paper.