XLIII CILAMCE
21-25, November, 2022
FOZ DO IGUAÇU - PR - BRAZIL

Artificial Intelligence methods applied to the damage detection in beams

Fernando V. B. Medeiros[1], Luiz C. Wrobel[1]

[1]*Dept. of Civil and Environmental Engineering, Pontifical Catholic University of Rio de Janeiro*
Terminal da PUC - Gávea, Rio de Janeiro, 22541-041, Rio de Janeiro/RJ, Brazil
fernandovbm@aluno.puc-rio.br, luiz.wrobel@puc-rio.br

Abstract. Structural health monitoring (SHM) has become increasingly important in the field of civil engineering. The objective of this paper is on the application of Artificial Intelligence Methods in the SHM field. The formulation uses modal parameters of a structure to detect damage related to the reduction of stiffness of a section. The databases for training and validating the AI methods were generated in a structured and automatic way by an algorithm developed in Python programming language within the finite element software Abaqus. The modal parameters analyzed were the first five natural frequencies of a beam. An exploratory analysis was performed using other characteristics such as: length of the beam, failure severity, material, boundary conditions, cross section and others. It was possible to evaluate the performance of the AI methods on the proposed problem. Finally, a parametric comparison was made between the different Artificial Intelligence methods.

Keywords: Artificial intelligence, Damage detection, Structural dynamics

1 Introduction

In the last few decades there has been an increase in literature production over the SHM to face the challenge in detecting and locating damage on aging structures. One of the main fields of research inside the subject is the development of autonomous, intelligent and real-time monitoring approaches to that matter, da Silva [1], and naturally it has reflected on big efforts towards the use machine learning techniques. Some of the most classic methods of damage detection on structures are vibration-based and consists on analyzing the modal characteristics variation such as natural frequencies, modal damping and mode shapes. This paper aims to compare machine learning methods on vibration data to detect damage.

2 Review

This paper used knowledge on Modal Analysis, the Finite Element Method and Machine Learning methods. In this section there will be a brief review on the concepts used. Vibration-based methods are widely popular techniques for health monitor structures. This paper focus on the extraction of natural frequencies of fixed and propped cantilever beams, Figure 1.

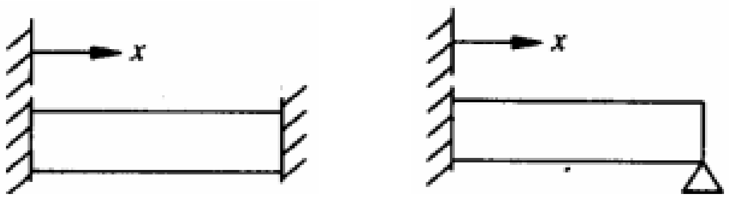

Figure 1. Boundaries conditions used on this paper

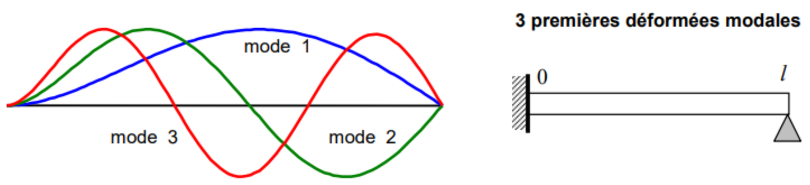

Figure 2. Example of modes shapes (each related to it's respective natural frequency), Pascal [2]

This work made use of a model-driven approach on the construction of a database to apply machine learning methods. Basically, the data was created on the natural frequency extraction of numerical models using the Finite Element Method (FEM). As it was preferable to generate a large database on frequency response, the FEM software Abaqus was chosen due to its possibility to model through a Python code. This made it possible the creation of a database of 2.652 beams and their respective first five natural frequencies.

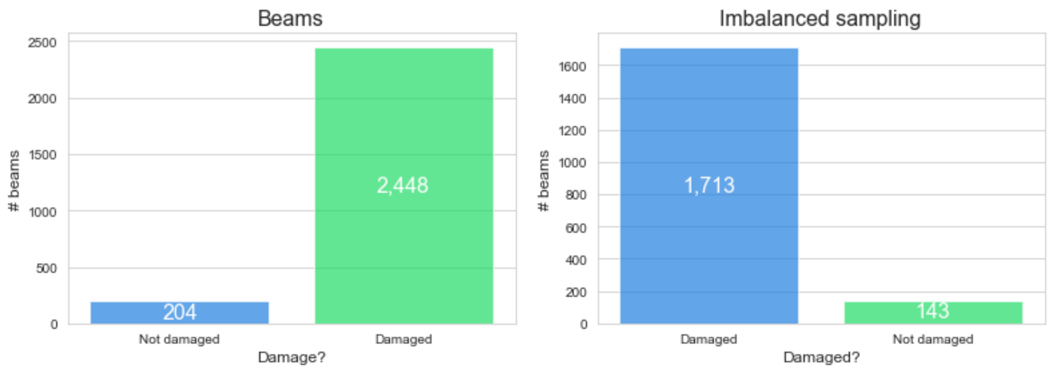

Figure 3. Beams database and imbalanced training sample

2.1 Machine Learning

Machine learning is referred to the development of systems through data instead of explicit programming, looking for patterns or anomalies hiding in the problem complexity. Also, technology development has made possible the popularization of AI techniques due to its increase on process speed and the capacity to absorb a large amount of data.

Normally, machine learning is subdivided into two groups: supervised and non-supervised learning. This paper focus on supervised learning, more specifically on the following three methods: Decision Tree (DT), Random Forest (RF), K-nearest Neighbor (KNN).

Decision tree

Decision trees (DT) offer easy interpretation, stability and high precision. Unlike linear models, they fit well on non-linear relations. The main goal of a DT in classification problems is to predict a target value from the creation of a series of decision rules. Given a desired depth, the algorithm splits data features to find optimal points and rules.

Random forest

Random forests (RF) algorithms are an averaging method of an ensemble of decision trees. It creates a diverse set of DT classifiers by introducing randomness in their construction. Usually, it leads to a reduced variance.

K-nearest neighbor

The K-nearest neighbors (KNN) goal is to label a data point according to the nearest data points. It tries to predict a test data label comparing its distance to the training data.

3 Methodology and results

3.1 Methodology

The first step in this study was to generate data to train and test the Machine Learning models. A script in Python 3 was created to generate the natural frequencies (first five) of beams with the parameters shown in Table 1. The non-damaged beams were those with the initial value of damage severity, e.g. 0.00% of the height damage, and those do not apply for the variation in damage position since it would generate four identical-non-damaged beams. A popular way to pre-dimensioning reinforced concrete beams is to use the height of section equals to 10.0% of its length, for that reason the height of the beams were attached to the length in a relation of 7.5% to 12.5%. Finally, the boundary conditions chosen were the ones usually found on reinforced concrete buildings.

Table 1. Beams database parameters

Parameters	Initial value	End value	Step	Nbr of cases
Length (m)	4.00	8.00	0.25	17
Height (Length percentage)	0.075	0.125	0.025	3
Width (m)	0.25	0.35	0.10	2
Damage severity (Height percentage)	0.00	0.30	0.10	4
Damage position*	0.10	0.40	0.10	4
Boundary conditions	Fixed-fixed	Propped cantilever beam	-	2
Total cases				2.652

Since the data set created was strongly imbalanced, as shown in Figure 3, after splitting the data in training and testing sets, two treatment on the training data were made: under-sampling and oversampling. These types of samples are shown in Figure 4. Under-sampling consists in randomly cut the bigger label to the same size of the smaller one. On the other hand, oversampling creates new data to smaller sized label, traditionally duplicating data. Since duplicating data does not seem to fit on this problem, the oversampling method applied was the Synthetic Minority Over-sampling Technique (SMOTE), Chawla et al. [3], which creates new data with small perturbations. After all it seems to be even more adequate, considering that in experimental modal analysis there is a lot more noise to the modal parameters extracted.

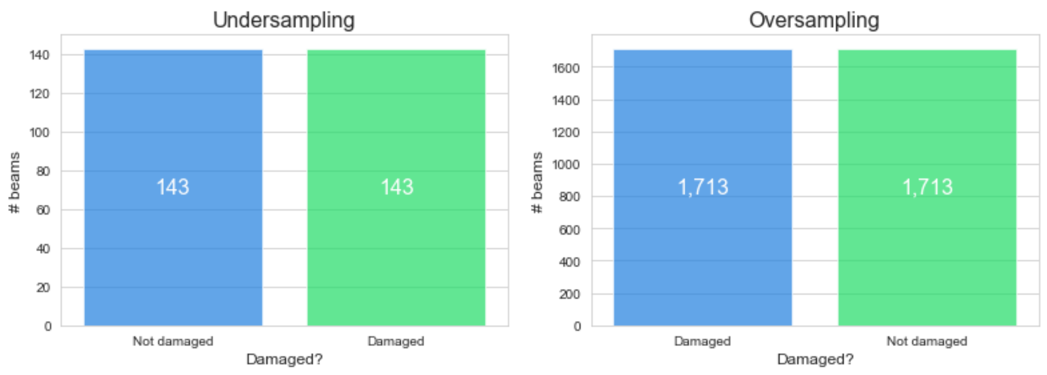

Figure 4. Beams training databases - under-sample x over-sample

In order to fairly compare different methods in side each of the sampling methods, a quick exploratory analysis was made of each method to determine the parameters that perform better given the sampling case.

To the decision tree and random forest algorithms the search for the optimum depth was made for these values: 3 ,4, 5, 6, 7, 8, 9, 10, 11, 12, 13 and 20 (DT 03 and RF 03, DT 04 and RF 04, etc).

To the K-nearest neighbor algorithm the search was made in the weighting method: uniform (KNN 1, 3 and 5) or distance (KNN 2, 4 and 6). And the learning algorithm: ball tree (KNN 1 and 2), KD tree (KNN 3 and 4) and brute (KNN 5 and 6).

Cross validation (CV) was calculated using the k-fold CV approach. The training set is split in a number, k, of smaller sets and then for each set a model is trained using the k-1 other sets and evaluated on this particular one

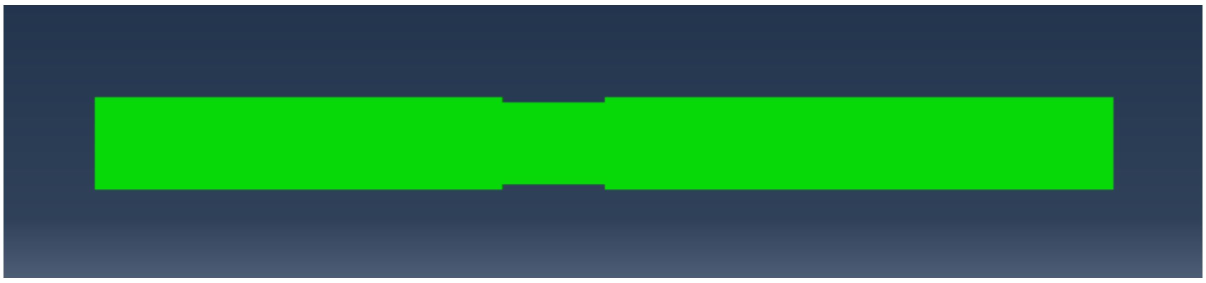

Figure 5. Example of beam model on Abaqus

as a test set. In this study, it was used ten subsets to cross validate the models and the accuracy of those subsets was plotted in ”box-plot” so that the distribution could be analyzed as a decision maker parameter along with the accuracy on the actual test set. All models were developed with scikit-learn library in Python, Pedregosa et al. [4].

Table 2. Decision tree - imbalanced sampling results

Name	CV Accuracy (train)	Accuracy (train)	Accuracy (test)
DT 03	0.921360	0.925108	0.920854
DT 04	0.921895	0.926185	0.920854
DT 05	0.921895	0.928341	0.920854
DT 07	0.921357	0.931573	0.919598
DT 08	0.920820	0.934806	0.919598
DT 09	0.922973	0.934806	0.919598
DT 10	0.924048	0.935345	0.920854
DT 11	0.920282	0.935345	0.920854
DT 12	0.920828	0.936422	0.920854
DT 06	0.922435	0.929418	0.918342
DT 13	0.921357	0.937500	0.920854
DT 20	0.919741	0.943966	0.918342

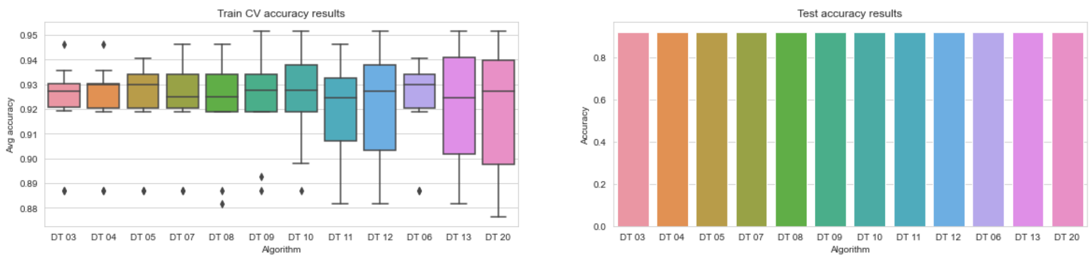

Figure 6. Decision tree results - Cross validation and Test accuracy for imbalanced sample

As we see in Table 2 and Figure 7, even though the accuracy both on the cross validation and the test set is around 92% for this training case (first 5 natural frequencies), it is because of the imbalanced distribution of nondamaged beams, which is around 8%. The trees found great results predicting that all input was damaged. On this scenario that solution is not viable. The dots outside the boundary of box-plot are the outliers prediction accuracy of a cross validation training set.

Table 3. Decision tree - under-sampling results

Name	CV Accuracy (train)	Accuracy (train)	Accuracy (test)
DT 04	0.192488	0.566434	0.841709
DT 03	0.154187	0.566434	0.839196
DT 06	0.305419	0.604895	0.814070
DT 05	0.149138	0.601399	0.812814
DT 07	0.272906	0.625874	0.807789
DT 11	0.412315	0.727273	0.670854
DT 10	0.526847	0.713287	0.650754
DT 13	0.490148	0.769231	0.640704
DT 12	0.468719	0.744755	0.630653
DT 08	0.474631	0.674825	0.581658
DT 09	0.484852	0.685315	0.556533
DT 20	0.485591	0.933566	0.444724

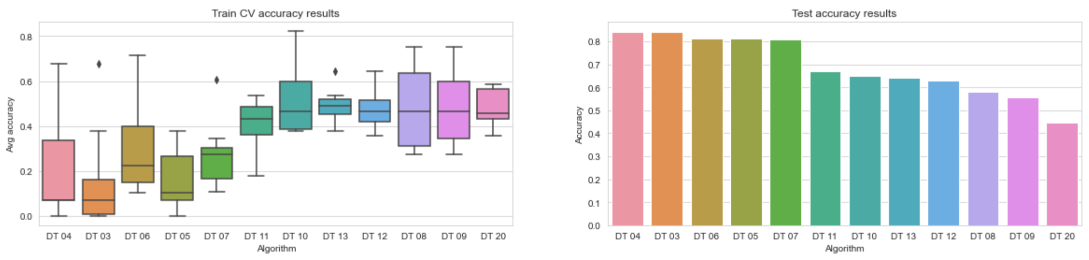

Figure 7. Decision tree results - Cross validation and test accuracy for under-sample

The response for the under-sample scenario, Table 3 and Figure 7, was better than the previous imbalanced one, but the cross validation results did not go over 52% accuracy, as expected for such small training data set. Once again, for those conditions the decision trees did not respond well.

Table 4. Decision tree - oversampling results

Name	CV Accuracy (train)	Accuracy (train)	Accuracy (test)
DT 07	0.561825	0.725628	0.904523
DT 09	0.596560	0.770578	0.891960
DT 20	0.873619	0.971979	0.889447
DT 10	0.619343	0.793637	0.811558
DT 11	0.666626	0.810566	0.801508
DT 12	0.690283	0.833333	0.802764
DT 08	0.548389	0.753357	0.773869
DT 06	0.499298	0.706363	0.750000
DT 13	0.729407	0.851722	0.761307
DT 05	0.534984	0.689142	0.739950
DT 04	0.465381	0.659370	0.572864
DT 03	0.609631	0.649737	0.571608

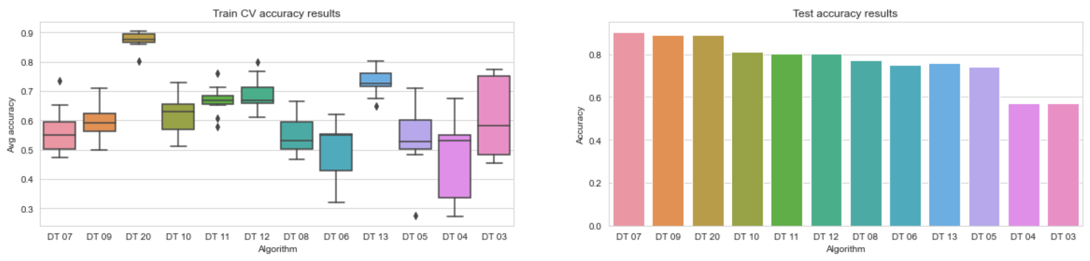

Figure 8. Decision tree results - Cross validation and test accuracy for over-sample

The SMOTE over-sample had the best result for training Decision Trees. On a bigger depth (20) the cross validation accuracy is 87.36% and the test accuracy 88.94%, as seen in Table 4 and Figure 8.

Table 5. Random forest - imbalanced sample results

Name	CV Accuracy (train)	Accuracy (train)	Accuracy (test)
RF 03	0.924060	0.924030	0.923367
RF 04	0.924060	0.924030	0.923367
RF 05	0.924060	0.924030	0.923367
RF 06	0.924060	0.924030	0.923367
RF 07	0.924060	0.924030	0.923367
RF 08	0.924060	0.924030	0.923367
RF 09	0.924060	0.924569	0.923367
RF 10	0.924060	0.926724	0.923367
RF 11	0.924060	0.929957	0.923367
RF 12	0.924060	0.936422	0.923367
RF 13	0.924060	0.950970	0.923367
RF 20	0.925135	0.997306	0.924623

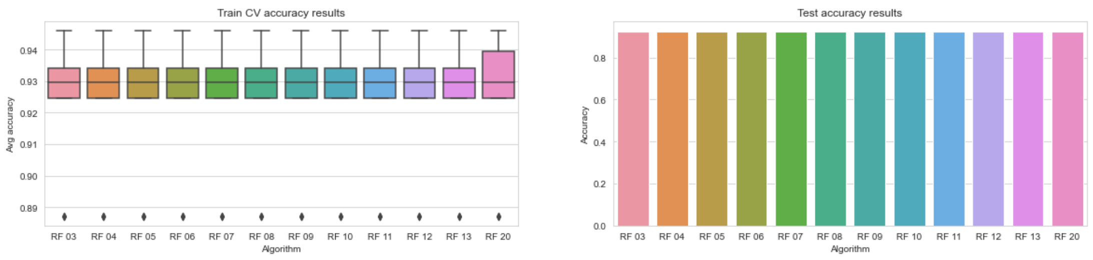

Figure 9. Random forest results - Cross validation and test accuracy for imbalanced sample

The performance of the RF algorithm over the imbalanced training set was similar to the DT, it was biased to the few non-damaged examples, as shown in Table 5 and Figure 9.

Table 6. Random forest - under-sample results

Name	CV Accuracy (train)	Accuracy (train)	Accuracy (test)
RF 03	0.140025	0.730769	0.501256
RF 04	0.252340	0.776224	0.466080
RF 06	0.385099	0.919580	0.453518
RF 07	0.385099	0.919580	0.453518
RF 08	0.371182	0.961538	0.453518
RF 05	0.273276	0.821678	0.443467
RF 09	0.370567	0.986014	0.443467
RF 12	0.398768	1.000000	0.443467
RF 13	0.405788	1.000000	0.443467
RF 10	0.391749	0.986014	0.438442
RF 20	0.419828	1.000000	0.440955
RF 11	0.399015	0.989510	0.433417

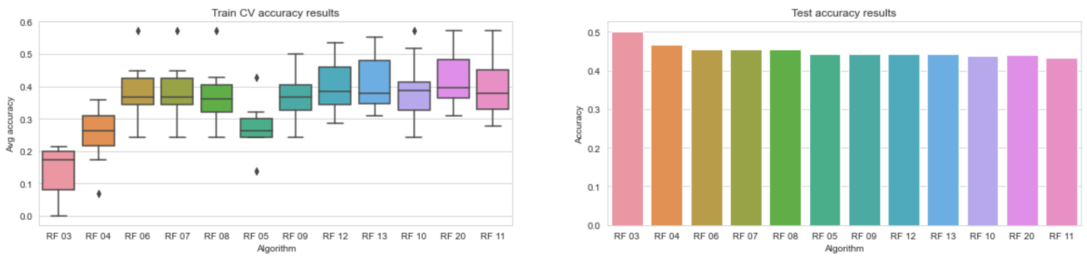

Figure 10. Random forest results - Cross validation and test accuracy for undersample

The results on the undersampled training set was not biased this time but it had low accuracy both on cross validation and test sets just like the DT algorithms, Figure 10. The random forest seems to overfit the training set as shown in Table 6.

Table 7. Random forest - oversample results

Name	CV Accuracy (train)	Accuracy (train)	Accuracy (test)
RF 20	0.921753	0.999708	0.927136
RF 13	0.874759	0.988325	0.900754
RF 12	0.846444	0.978400	0.889447
RF 11	0.815790	0.966725	0.873116
RF 10	0.771123	0.954758	0.868090
RF 09	0.720036	0.918564	0.837940
RF 08	0.659066	0.850846	0.778894
RF 07	0.609735	0.778459	0.709799
RF 06	0.609735	0.778459	0.709799
RF 05	0.575607	0.694396	0.606784
RF 04	0.579442	0.673672	0.570352
RF 03	0.602567	0.646818	0.555276

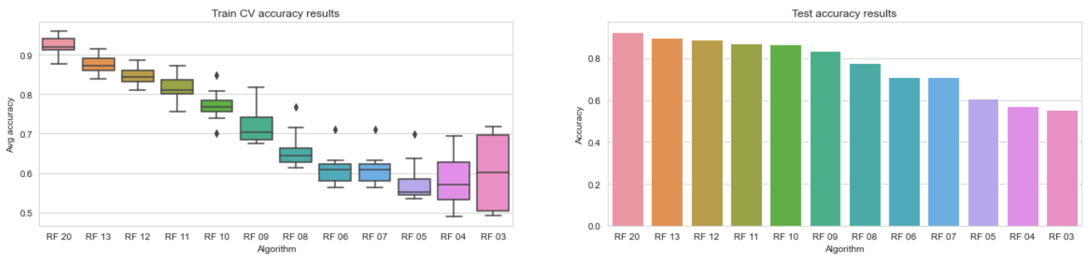

Figure 11. Random forest results - Cross validation and Test accuracy for oversample

Like for the DT algorithm, the RF grew along the depth on the SMOTE oversampling training set, reaching excellent performance on cross validation, 92.17%, and testing set, 92.71%, Table 7 and Figure 11.

Table 8. K-nearest neighbor - imbalanced sample results

Name	CV Accuracy (train)	Accuracy (train)	Accuracy (test)
KNN 02	0.909515	1.000000	0.920854
KNN 04	0.909515	1.000000	0.920854
KNN 06	0.909515	1.000000	0.920854
KNN 01	0.859398	0.990841	0.880653
KNN 03	0.859398	0.990841	0.880653
KNN 05	0.860480	0.990841	0.880653

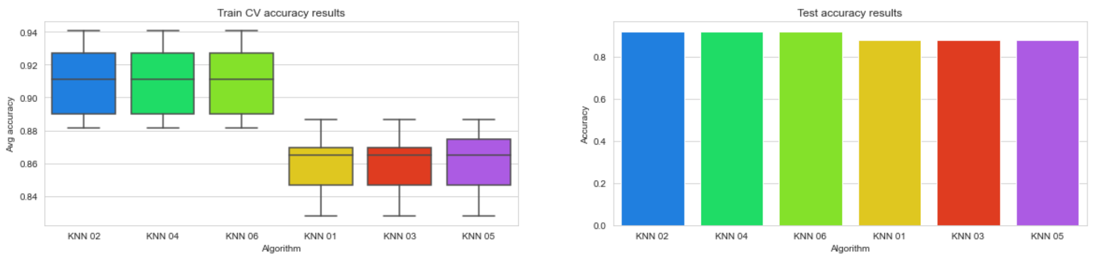

Figure 12. K-nearest Neighbor results - Cross validation and Test accuracy for imbalanced sample

The K-nearest neighbor was set to find two clusters, non-damaged and damaged beams. The results for the imbalanced training was biased just like DT and RF algorithms, Table 8 and Figure 12.

Table 9. K-nearest neighbor - undersample results

Name	CV Accuracy (train)	Accuracy (train)	Accuracy (test)
KNN 02	0.368473	1.000000	0.428392
KNN 04	0.368473	1.000000	0.428392
KNN 06	0.368473	1.000000	0.428392
KNN 01	0.336946	0.762238	0.224874
KNN 03	0.336946	0.762238	0.224874
KNN 05	0.336946	0.762238	0.224874

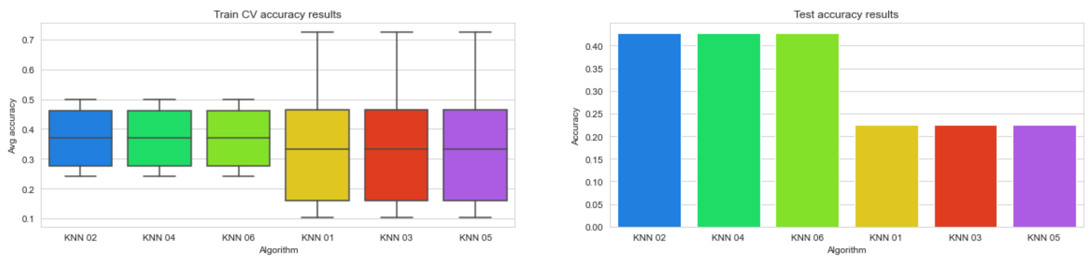

Figure 13. K-nearest Neighbor results - Cross validation and Test accuracy for under-sample

For the under-sample scenario, KNN overfits the training set and heavily underperform on the cross validation and testing phases, Table 9 and Figure 13.

Table 10. K-nearest neighbor - over-sample results

Name	CV Accuracy (train)	Accuracy (train)	Accuracy (test)
KNN 02	0.896987	1.000000	0.893216
KNN 04	0.896987	1.000000	0.893216
KNN 06	0.896987	1.000000	0.893216
KNN 01	0.853844	0.983071	0.826633
KNN 03	0.853844	0.983071	0.826633
KNN 05	0.853844	0.983363	0.826633

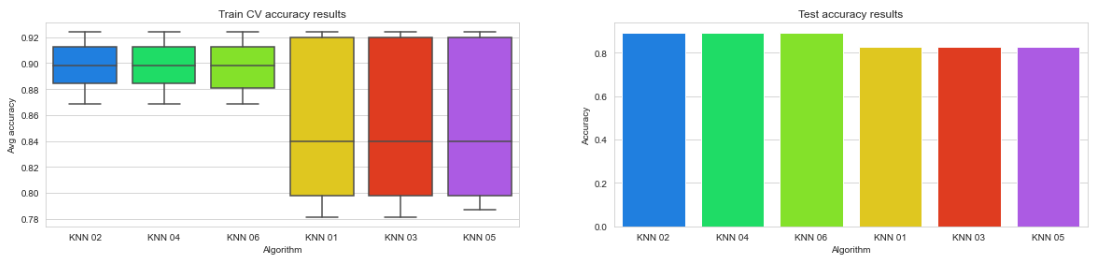

Figure 14. K-nearest Neighbor results - Cross validation and Test accuracy for over-sample

Even though the studied parameters did not make a huge impact, all KNN training cases performed well on the SMOTE over-sample scenario with more than 85% accuracy on cross validation and over 82% on the testing set, Table 10 and Figure 14.

Table 11. Multiple models - imbalanced sample results

Name	CV Accuracy (train)	Accuracy (train)	Accuracy (test)
RF 03	0.924060	0.924030	0.923367
DT 03	0.921360	0.925108	0.920854
KNN 02	0.909515	1.000000	0.920854

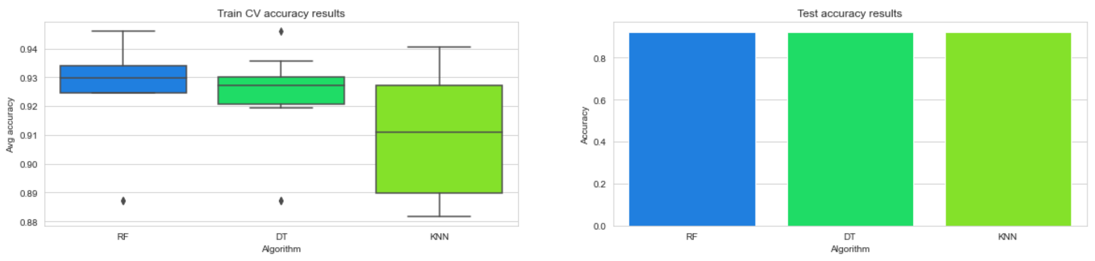

Figure 15. Multiple models results - Cross validation and Test accuracy for imbalanced sample

All three methods on the imbalanced sample were biased and the accuracy obtained should not be considered as good models to predict damage on beams, Table 11 and Figure 15.

Table 12. Multiple models - under-sample results

Name	CV Accuracy (train)	Accuracy (train)	Accuracy (test)
DT 04	0.192488	0.566434	0.841709
RF 03	0.140025	0.730769	0.501256
KNN 02	0.368473	1.000000	0.428392

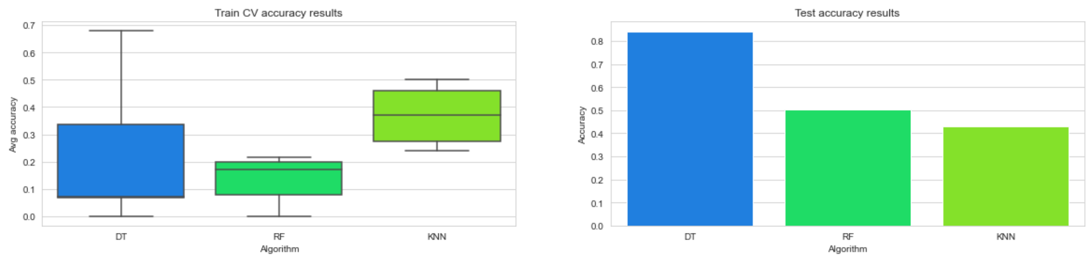

Figure 16. Multiple models results - Cross validation and Test accuracy for under-sample

The under-sample case led to a better performance of the DT algorithm, considering that all three models got low cross validation accuracy. After all, every model under performed for this training set, Table 12 and Figure 16.

Table 13. Multiple models - over-sample results

Name	CV Accuracy (train)	Accuracy (train)	Accuracy (test)
RF 20	0.921753	0.999708	0.927136
DT 20	0.896987	1.000000	0.893216
KNN 02	0.873619	0.889447	0.889447

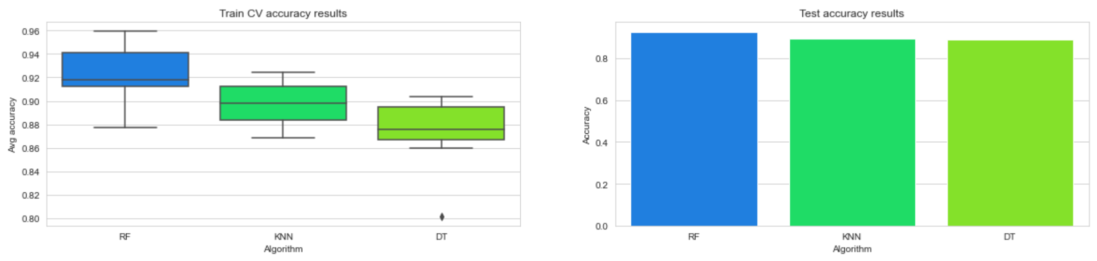

Figure 17. Multiple models results - Cross validation and Test accuracy for over-sample

Finally, all three models obtained high accuracy results on cross validation and testing sets for the SMOTE case. Overall, Random Forests algorithm with depth = 20 got the best results: 92.17% cross validation accuracy and 92.71% test accuracy, Table 13 and Figure 17.

4 Conclusions

In order to assess the problem of damage detection in beams using machine learning techniques, pre-processing data has proved to be indispensable. On this particular case, due to the imbalanced behavior of data, all models suffered from its bias. On the other hand, both under-sample and over-sample training sets do not suffer from it but, due to its small size, the under-sample underperformed. The SMOTE oversampling had the best results for all three models studied: decision trees, random forests and k-nearest neighbors. Finally, although all methods showed satisfactory performance on the over-sample case, but Random Forest had the best results.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] da M. F. M. Silva. *Machine learning and computer vision techniques for damage detection and modal analysis*. PhD thesis, Universidade Federal do Pará, Belém, 2020.

[2] J.-C. Pascal. *Vibrations et Acoustique 2*. Ecole Nationale Superieure d'Ingenieurs du Mans, Universite du Maine, 2008.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, vol. 1, n. 16, pp. 321–357, 2002.

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