Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Critical bending moment of uniform and tapered beams

BERNARDO FALÉ 93331[1], (DETI, UA), HUGO DOMINGOS 98502 [2], (DETI, UA)
[1]Departamento de Eletrónica, Telecomunicações e Informática, Universidade de Aveiro, Aveiro (e-mail: mbfale@ua.pt)
[2]Departamento de Eletrónica, Telecomunicações e Informática, Universidade de Aveiro, Aveiro (e-mail: h.domingos@ua.pt)

This work was done for the Fundamentals of Machine Learning course with the help of class materials

ABSTRACT This study arises because of the lack of a formula to calculate the elastic critical bending moment of tapered steel beams. Unlike uniform beams, there isn't a formula yet due to the huge complexity of the problem, but the necessity of making these calculations on the design phase of the implementation are huge. In this paper we approach the proposed problem by implementing different machine learning problems to predict the critical bending moment of beams.

INDEX TERMS Machine learning, Uniform and tapered beams, civil engineering, artifical intelligence

I. INTRODUCTION

This work consists of studying and comparing Neural Networks and Linear and Polynomial Regressions to predict the critical bending moment on uniform and tapered beams with the usage of 9 features. Furthermore, we take different approaches and implementations to compare accuracy and performance between different machine learning models. In this article we will study the approach of related literature in the field of civil engineering, in the scope of machine learning; We will make a brief review of the state of the art of similar works that are related to the proposed problem and then research new methods, including feature engineering and other machine learning models, as well as different data transformations to achieve accurate results for the specified task. We will also enter in detail in the approach that was taken in the article [3] on IV.

II. RELATED WORK

Within this section, we will provide a brief overview of existing literature and research relevant to the present study. This section aims to establish the foundational context by examining prior works and methodologies employed in the field of civil engineering and machine learning; We will focus on the machine learning approaches and try to extract benefits. By reviewing and analyzing these contributions, this study seeks to build upon the existing body of knowledge and highlight the significance of the current research within the scope of our work.

In [6], the authors provide valuable information on research works related to machine learning-based shear strength prediction models for reinforced concrete beams. This work discusses the potential of artificial neural networks (ANN) and most of the standard machine learning techniques used in evaluating the shear strength of reinforced-concrete deep beams, as well as the development of new design equations using ann and parametric study. Additionally, the advantages and challenges of machine learning models in solving complex engineering problems are presented, along with insights on ensuring the representation of different variations of each feature in the dataset, the use of ensemble and hybrid models, and the optimization of hyperparameters. The authors also conclude from the their literature review that XGboost, Random Forest, and ANN approaches work best. In conclusion, it's a comprehensive study on machine learning in the scope of civil engineering, which gives great insights on the proposed problem. Despite the fact that it is not the same problem, we can extract information on how we could drive our work.

The research article [1] addresses the challenge of accurately predicting the flexural capacity of reinforced concrete beams affected by steel corrosion through machine learning (ML)-based techniques, utilizing six parameters as input features. These parameters include beam width, beam effective depth, concrete compressive strength, reinforcement ratio, reinforcement yield strength, and corrosion level. The research employs four single and ensemble ML models for evaluation: decision tree, support vector machine, adaptive boosting, and gradient boosting. The study optimizes the hyperparameters of each model using grid search and K-fold cross-validation, with root mean squared error used as the performance index.

The benefits of this approach are multifaceted. By leveraging complex and non-linear relationships between input features and the target label, the models capture patterns that traditional analytical and numerical models would overlook, thus improving the accuracy of prediction. The proposed ensemble ML model, particularly the gradient boosting model, demonstrates superior performance with high generalization ability and accurate prediction of the flexural capacity of corroded RC beams. This represents a significant advancement in the field, offering a more reliable and accurate model for such predictions.

In [4], the proposed problem is focused on predicting the ultimate bending moment resistance of high strength steel welded I-section beams subjected to bending, with the aim of investigating and improving the accuracy of cross-sectional design for these beams.

The document outlines the use of machine learning for predicting the behaviour and resistance of beams under bending. Specifically, the study explores the application of Support Vector Regression, Random Forest Regression, Artificial Neural Network, and Boosting algorithm to develop predictive models for the ultimate bending moment resistance of beams. The authors delineate the fact that using ML-based models for this prediction task include the potential for more accurate predictions compared to traditional methods and the potential to overcome the limitations of current design codes that may lead to overbound-ish predictions. The document presents a comprehensive review of the application of ML techniques to address the complexity of the problem and the inherent limitations of traditional modeling methods in accurately predicting the bending resistance of I-section beams. Overall, the proposed techniques offer the potential to improve the accuracy of predicting the ultimate bending moment resistance of high strength steel welded I-section beams, which could have significant implications for the design and application of these structural members in engineering. To help boost the results, this work details the usage of uniform scaling feature transformation, feature selection based on Pearson's correlation approach, and K-fold cross validation hyperparameters optimization. The researchers' findings indicate that when contrasted with the bending moment forecasts derived from design codes and specifications, a linear regression machine learning model demonstrates the potential for increased accuracy. However, it shows relatively more variability in results for sections with smaller dimensions; This can be due to the provided dataset being short, or the algorithm being to simple to generalize. Additionally, alternate machine learning models like SVR, ANN, RFR, and XGBoost, yield superior accuracy, outperforming both the design methods and the linear regression model.

Finally, in [5] it is discussed the use of machine learning approaches to predict the shear strength of reinforced concrete slender beams. Specifically, it mentions the use of various machine learning models such as K-nearest neighbor, decision tree, random forest, gradient boosting decision tree, Adaboost, and XGBoost for training and evaluation purposes. These models are used to address the problem of predicting the shear strength of RC slender beams and assess their performance to identify the model with the best predictive accuracy and generalization ability. Additionally, this work gives an interesting new approach based on the use of interpretability methods, particularly the SHapley Additive exPlanations (SHAP) method, to provide insights into the predictions of the complex machine learning models and to explore the contribution rates of different components to the shear strength of the beams. It was concluded that XGBoost was the best performant and predictive model compared to the other approaches. Although the related work is not about bending beams, we can extract important insights and ideas to apply on our problem; XGBoost was, once again, the framework with the best performant model, which might indicate that testing should be done in the context of this work.

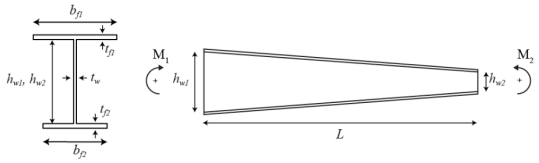

FIGURE 1. Dataset notation

III. DATASET

The original dataset can be found on McrNet repository[2]. The dataset is divided into training set and testing set, each one containing 60549 and 8526, respectively. Each of the samples consists of 9 features, that are measures of the beam in milimeters, and the respective critical bending moment of the beam.

It is also provided the normalized dataset. The method used to normalize the dataset is also refered on the repository [2].

A. DATASET GENERATION

The dataset was generated using a numerical model used to obtain the critical bending moment of steel beams. The numerical model is based on the FEM and it was created using the ANSYS software.

Then it was used a smart sampling technique that consists of two main ideas: a random number generator for each feature with a minimum and maximum limit and a certain number of points in the interval; and constrained geometric proportions defined to keep the samples within the common values used in practice.

IV. NEURAL NETWORK ARCHITECTURE

Several architectures were studied in the article. The architecture consisted of 9 input nodes and 1 output node and one or two hidden layers.

Accuracy and training times of different neural network architectures (using 60549 samples for training).

Architecture	R^2 (train)	R^2 (test)	Training time[a] [min.]
$9 \times 18 \times 1$	0.98765	0.98697	71
$9 \times 32 \times 1$	0.99338	0.99333	77
$9 \times 32 \times 8 \times 1$	0.99878	0.99859	114
$9 \times 32 \times 16 \times 1$	0.99908	0.99897	121
$9 \times 32 \times 32 \times 1$	0.99952	0.99945	134
$9 \times 64 \times 1$	0.99715	0.99694	92
$9 \times 64 \times 8 \times 1$	0.99951	0.99938	132
$9 \times 64 \times 16 \times 1$	0.99974	0.99965	141
$9 \times 64 \times 32 \times 1$	0.99982	0.99966	162
$9 \times 64 \times 64 \times 1$	0.99982	0.99974	207
$9 \times 128 \times 1$	0.9985	0.9983	124
$9 \times 128 \times 16 \times 1$	0.99987	0.99969	190
$9 \times 128 \times 64 \times 1$	0.99997	0.99968	380
$9 \times 256 \times 1$	0.99906	0.99884	240
$9 \times 512 \times 1$	0.99934	0.99906	513

[a] These values are merely indicative.

FIGURE 2. Table of results of the Neural Networks tested on the article

Firstly it was used an architecture with only one hidden layer, starting with 18 nodes; That is twice the number of input nodes. Afterwards the number of hidden nodes was incremented to double the number of hidden nodes of the neural network that was trained before, iteratively, until achieving a maximum of 512 nodes. Then it was introduced another hidden layer, starting at 8 nodes and doubling the number hidden layers of the previously trained neural network, until achieving a maximum of 64 nodes.

passar print para uma tabela

The author of the article [3] considered that the neural network with the architecture 9 x 128 x 16 x 1 was the most usable because of its accuracy and training time when comparing with the other considered approaches.

A. OPTIMIZATION METHOD

In [3] an optimization algorithm was needed to make the cost function converge. Therefore, the author made an approach conseidering an optimization algorithm which is a combination of the backpropagation algorithm and the Levenberg–Marquardt optimization algorithm.

Backpropagation is a standard mechanism to train a neural network, it computes the gradient of the loss function with respect to each weight in the network and propagates the error back through the network layers.

The integration of the Levenberg-Marquardt algorithm is a more advanced method used in conjunction with backpropagation, that allows a faster convergence in the training process of solving non-linear least squares problems, which are common in neural network training.

V. LINEAR AND POLYNOMIAL REGRESSIONS

The main purpose of this work is to compare the neural network approach with the linear and polynomial regressions. We began by making a simple linear regression and polynomial regressions with multiple degrees. However, we thought that the results showed the model was not being capable of learning the correlation between the features and the label, as we can see in Table1 and Figures3, 4, 5, 6, 7.

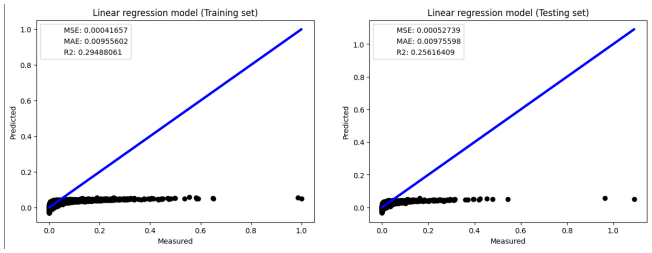

FIGURE 3. Results for a linear regression

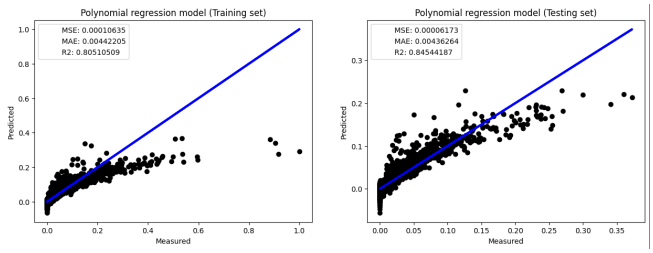

FIGURE 4. Results for a polynomial regression of degree 3

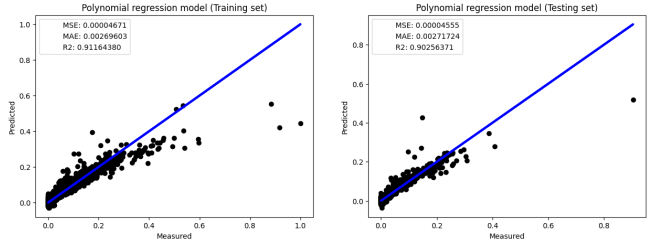

FIGURE 5. Results for a polynomial regression of degree 4

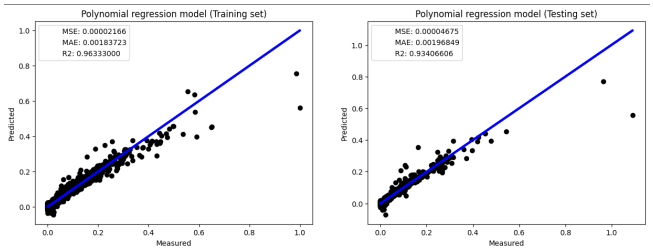

FIGURE 6. Results for a polynomial regression of degree 5

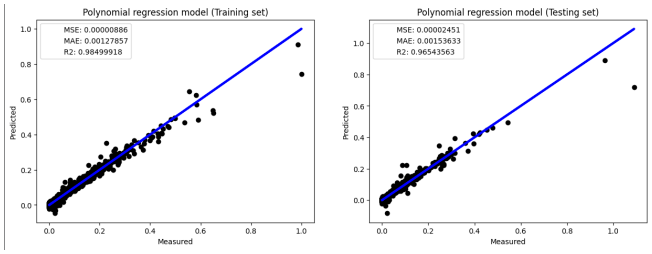

FIGURE 7. Results for a polynomial regression of degree 6

Degree	MSE	MAE	R^2(Training)	R^2 (Testing)
Linear	0.00052739	0.00975598	0.29488061	0.25616409
3	0.00006173	0.00436264	0.80510509	0.84544187
4	0.00004555	0.00271724	0.91164380	0.90256371
5	0.00004675	0.00196849	0.96333000	0.93406606
6	0.00002451	0.00153633	0.98499918	0.96543563

TABLE 1. Performance metrics in Linear and Polynomial Regression Model

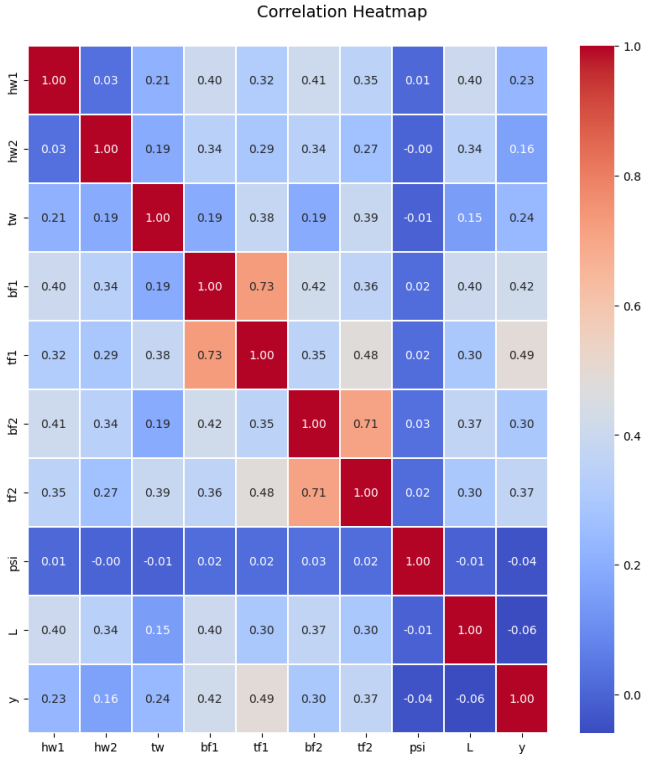

FIGURE 8. Correlation matrix of the original dataset

Thus, we made a correlation matrix 8 to visualise information that corroborated our theory. There were very low levels of correlation between almost every component on the dataset, including the label.

A. FEATURE ENGINEERING

As we saw previously, on the introduction of this section V, the level of correlation between the features themselves and the label were very low. As a result, we decided to try a technique that would apply a transformation to the labels of the data, making it more suitable for the linear and polynomial regressions.

Accordingly, we decided to use the original training and testing sets without normalization and merge them to apply transformations and the corresponding normalization, aiming to approximate the distribution of the labels to a Gaussian distribution.

We can see on Figure 9 that the labels distribution is right skewed. It is evident that the long tail extends to the higher values, which indicates that the data points are concentrated

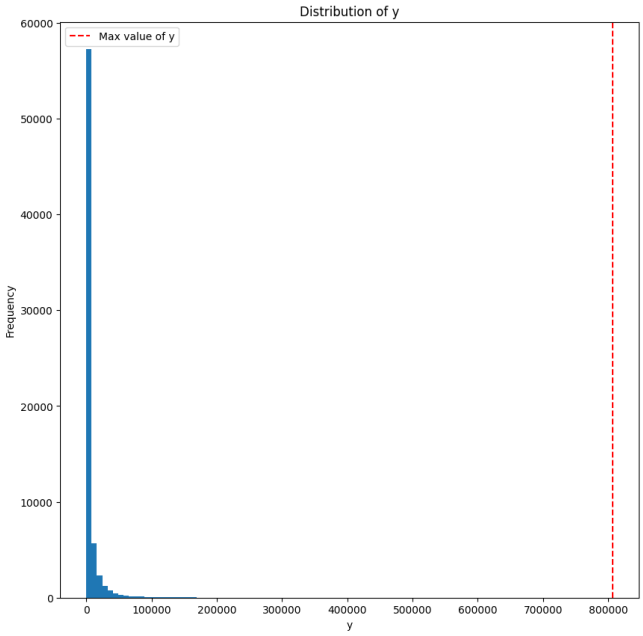

FIGURE 9. Labels distribution of the original dataset

more towards the left margin. This gave key signs that there were some outliers in our dataset, which is a typical problem in regression problems. To solve this problem we applied a logarithmic transformation, more specifically the common logarithm, because log10 is negative for values between 0 and 1. Then it was divided by the max value of the transformation, so that we have the data normalized. This way, we make the distribution more gaussian-ish and compress the scale of the data and reduce the influence of extreme values so that the patterns are more visible.

We also tried the Box-Cox algorithm for this task but we got worst results. Our theory for this failed attempt stands on extreme skewness of the dataset and large number of outliers, which definetly impacts its effectiveness.

As we can observe from Figure 10 the labels distribution as a more lookalike Gaussian distribution.

To transform the features we basically took the same approach, therefore we tried to apply the PowerTranformer; PowerTransformer is a preprocessing technique used to apply transformations to make data conform more closely to a normal distribution. Thus being a technique that goes hand by hand with our normalisation approach.

We also tried a variant of the PowerTransformer, the box-cox transformation, and the StandardScaler. We concluded that our features did not needed to be scaled and centered, which is the purpose that StandardScaler mostly serves.

Following the implementation of the preprocessing techniques, we divided the dataset in order to have 20% of testing data, which was the value that gave us better results when comparing the r2 score between the training data and testing data.

In Figure 11 we can see that the correlation increased, specially between the features and the desired output, making

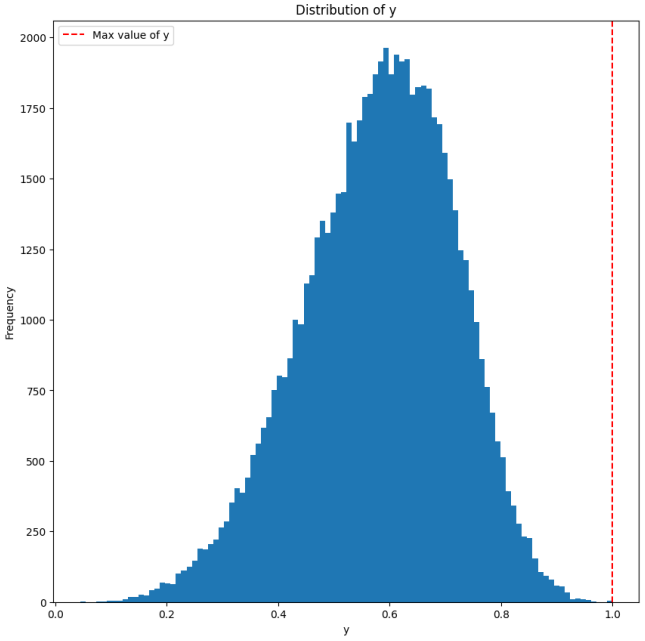

FIGURE 10. Labels distribution of the original dataset after the logaritmic transformation and the normalization

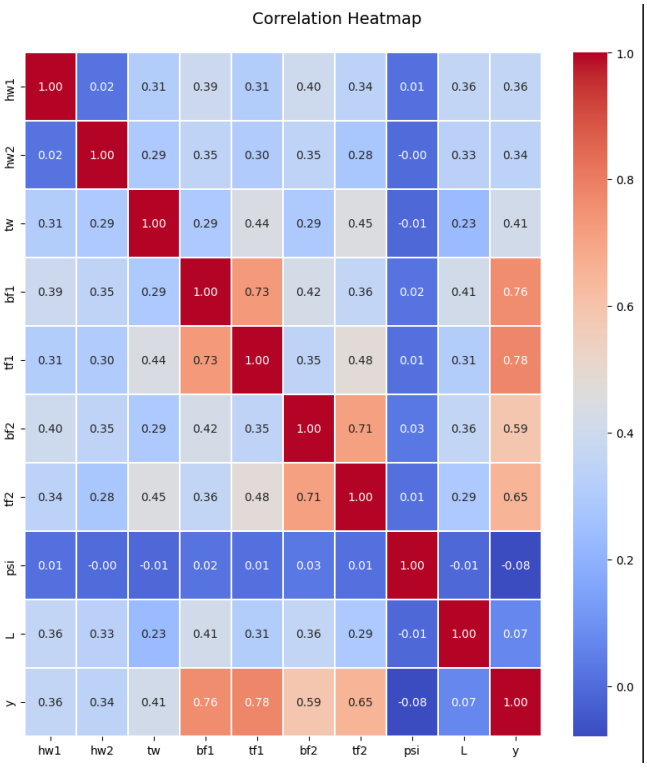

FIGURE 11. Correlation matrix of the dataset after the transformations applied

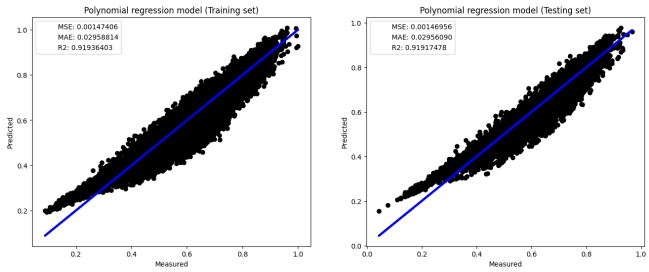

FIGURE 12. Results for a polynomial regression of degree 1

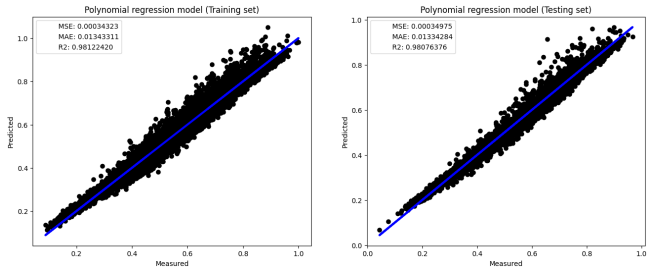

FIGURE 13. Results for a polynomial regression of degree 2

it easier for the linear and polynomial regressions to learn and understand the problem.

B. RESULTS

On this section we can compare the results obtained from the polynomial regression models trained with degrees 1, 2, 3, 4, 5, 6 and 7, on Figures 12, 13, 14, 15, 16, 17 and 18, respectively. We considered the best model the polynomial regression model with 6 degree, since it has slightly worst results when comparing with the 7 degree model on the training data, however the results for the testing data start degrading, which could mean overfitting. The three metrics used to measure the performance of each model were the Mean Squared and the Means Absolute Error for the testing data and the R^2 Score for both the training and testing data as we can see on Table2

Degree	MSE	MAE	R^2(Training)	R^2 (Testing)
1	0.00146956	0.02956090	0.91936403	0.91917478
2	0.00034975	0.01334284	0.98122420	0.98076376
3	0.00008814	0.00655904	0.99517183	0.99515213
4	0.00003728	0.00396505	0.99808835	0.99794964
5	0.00001756	0.00268577	0.99918245	0.99903398
6	0.00000879	0.00191215	0.99965604	0.99951631
7	0.00000973	0.00150236	0.99986107	0.99946511

TABLE 2. Performance metrics in Polynomial Regression Model

VI. OTHER MODELS

Since the neural network model presented on the article [3] could learn successfully even without the feature engineering approach made, we decided to study other models besides the linear and polynomial regressions to understand in a higher

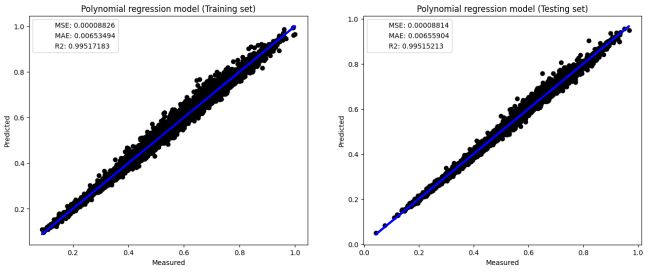

FIGURE 14. Results for a polynomial regression of degree 3

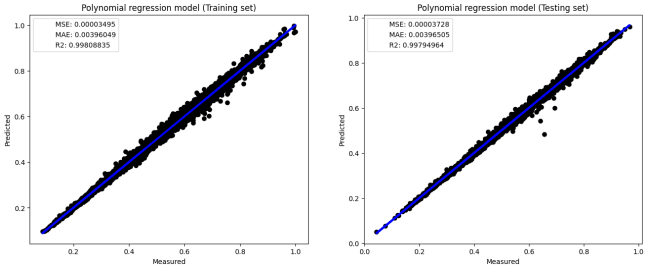

FIGURE 15. Results for a polynomial regression of degree 4

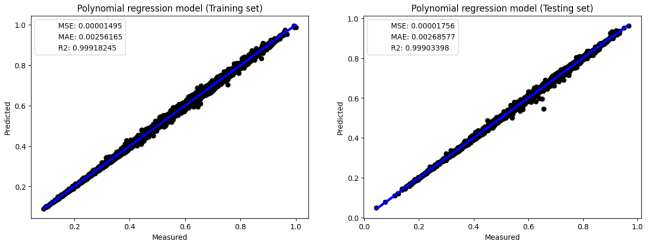

FIGURE 16. Results for a polynomial regression of degree 5

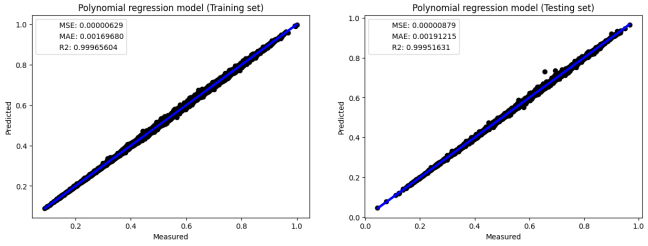

FIGURE 17. Results for a polynomial regression of degree 6

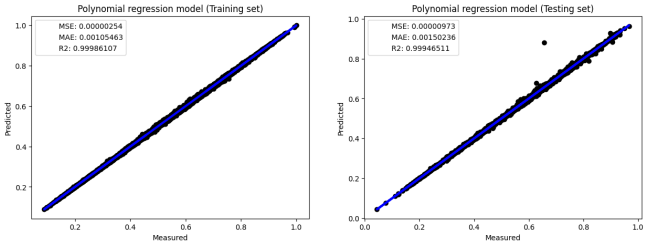

FIGURE 18. Results for a polynomial regression of degree 7

degree if it was possible to achieve good results when the data has low levels of correlation. For the models studied on the following subsections, it was applied the PowerTransformers to the features and the labels were normalized to have all values withing 0 and 1, to better compare with the Neural Network models on the article [3].

A. XGBOOST

XGBoost stands for eXtreme Gradient Boosting, a powerful machine learning algorithm known for its speed and performance in structured/tabular datasets. It's an ensemble learning method that builds multiple decision trees sequentially, where each tree corrects the errors of its predecessor.

Due to the fact that all the reviewed works [1, 4, 5, 6] used at least a model of XGBoost, we thought we might as well give it a try and compare the results of its predictions on the dataset normalised by [3], present in [2], and compare them to our normalised dataset.

We used the mean squared error regularization algorithm. Firstly, we experimented with the standard model learning rate, 0.1. We took the first results and evaluated the model through several performance metrics, such as mean absolute error (MEA), mean squared error (MSR), root mean squared error (RMSE), and R2, for the purposes of quantifying the accuracy of our model. As we can see in 19, the results were a bit scattered close the right margin of the plot; This is in collusion to what we said in V, where the distribution was right skewed. Moreover, the distribution of residuals in 20 was not at the point we wanted it to be; In fact, we wanted the residuals to be as close to 0 as possible. To solve this issue, we made a quick script that allowed us to have the best possible learning between [0, 0.5], through incrementation of steps of 0.1 each; We found the best learning rate when the RMSE was the lowest. All the metrics extracted from the results can be seen in 3. We concluded that the best model had an approximate 0.96 accuracy (R2 score). We can verify these results by analysing 21 and comparing it to the first model we implemented; We can still see the that the results towards the end are still a little bit scattered. This, in a way, corroborates our feature engineering approach, where we make a logarithmic transformation to normalise the labels. However, the predictions are much more uniform, in the sense that it takes a much straighter line, when comparing to 19. We can also see that the residuals are much more concentrated towards 0, which indicates that the model understands the problem in a much better way. Overall, the accuracy increases about 0.2 decimal points. After these tests, we decided to build a model with our own normalised dataset V-A. It's apparent that in 23 the predictions are much closer to the red line on the upper bound of the line, this might say that our model is conservative in its prediction; The residuals are also a magnitude lower in comparison to our other models 24. However, our model presents a common behaviour in regression problems. As we can see in 3, when applying the PowerTransformer transformation, our performance metrics do get much lower, except for the R2 score. This might

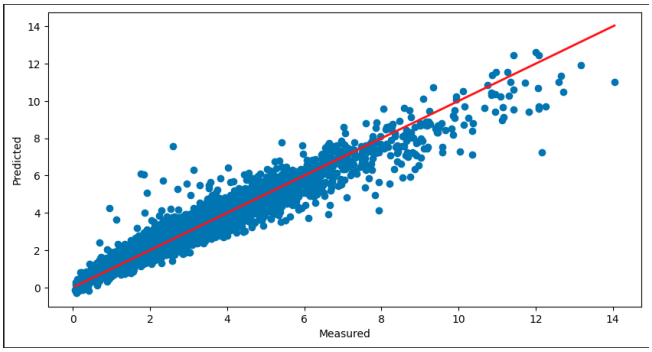

FIGURE 19. Testing dataset - Predicted vs Measured plot when lr = 0.1

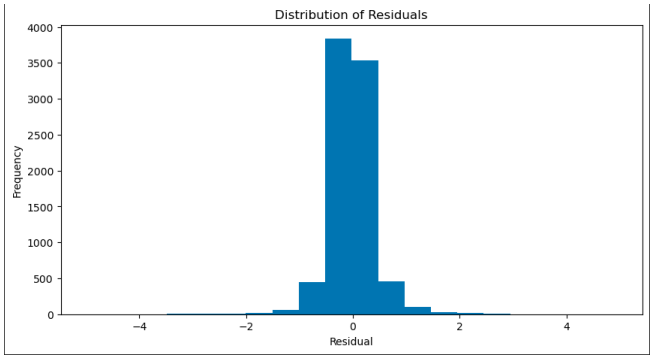

FIGURE 20. Testing dataset - Residual distribution when lr = 0.1

indicate that the model is predicting values closer to the actual ones, but it might not be explaining the variance in the target well enough; It is capturing the general pattern, but not the variance around that same pattern. We were still not satisfied with our results so we applied one last change on our hyperparameters. We increased the number of our training rounds exponentially, from 100 to 100000. We briefly tested both datasets to see if we could get a higher accuracy score. We can verify in 3 we had our best results yet. However, we can see in the cost function 25 that it converged far before the max number of rounds; This means that the optimal number of rounds was much lower, and should be actually around ±8000.

Lastly, we conclude that the XGBoost is a great approach for these kind of problems, like we could see in the related work we reviewed; We found that the results we obtained were really good for such a simple model, which in turn took about 0.3ms and ±10s in our local personal computers, with 100 rounds and 100000 for each training, respectively. It is important to note that in [3], it took much longer to train the artificial neural network implemented in the work. This makes the XGBoost approach an appetizing way of handling this problem with, still, great and considerable results; Which could be even better if, per example, the loss functions were customized.

LR	*		MAE	MSE	RMSE	R2
0.1		Training	±0.214	±0.123	±0.351	±0.954
		Testing	±0.263	±0.171	±0.414	±0.938
0.32		Training	±0.163	±0.065	±0.254	±0.976
		Testing	±0.219	±0.114	±0.337	±0.959
0.32	X	Training	±0.011	±0.0003	±0.017	±0.976
		Testing	±0.018	±0.0007	±0.027	±0.946
0.32	X(100000r)	Training	±0.011	±0.0003	±0.017	±0.976
		Testing	±0.015	±0.0005	±0.023	±0.963
0.32	(100000r)	Training	±0.163	±0.065	±0.254	±0.976
		Testing	±0.160	±0.065	±0.254	±0.976

TABLE 3. Performance metrics in XGboost approach (* With PowerTransformer)

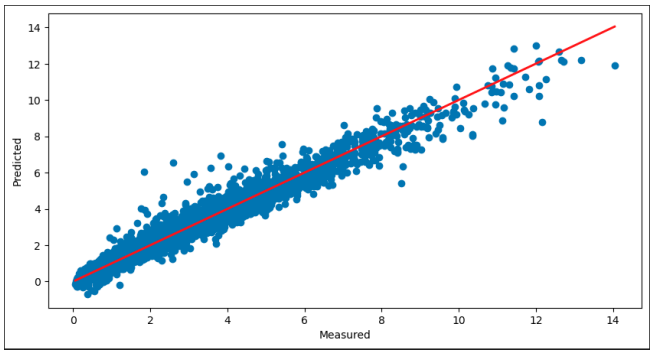

FIGURE 21. Testing dataset - Predicted vs Measured plot when lr = 0.32

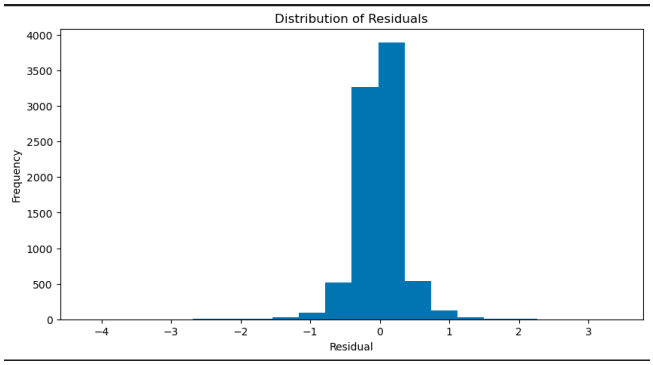

FIGURE 22. Testing dataset - Residual distribution when lr = 0.32

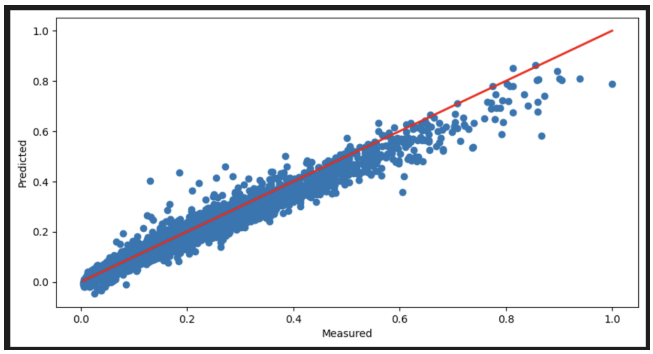

FIGURE 23. Testing normalised Powertransformer dataset - Predicted vs Measured plot when lr = 0.32

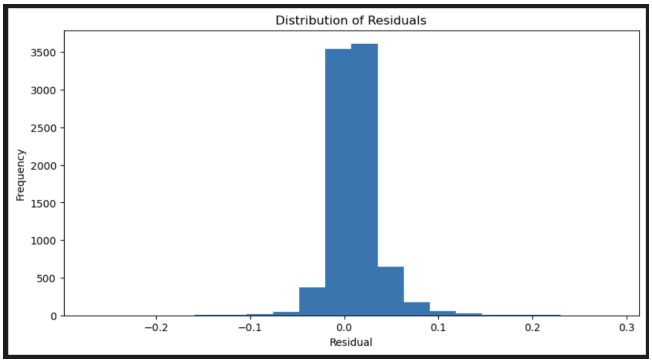

FIGURE 24. Testing normalised Powertransformer dataset - Residual distribution when lr = 0.32

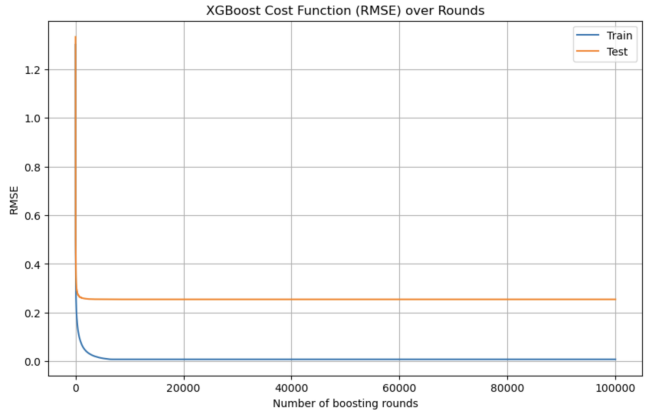

FIGURE 25. XGBoost cost function over 100000 rounds

B. SUPPORT VECTOR REGRESSOR

Support Vector Regressor (SVR) is a type of Support Vector Machine (SVM) used for regression tasks in machine learning, and it's based on the principle of SVM, which is primarily used for classification.

The main goal of SVR is to find a function that approximates the relationship between the input features and the continuous output variable. It tries to fit the best line within a threshold value (margin). The datapoints that fall within this margin are considered support vectors, and SVR tries to minimize the error within this margin, rather than minimizing the error for each data point. SVR can also use the kernel trick to handle nonlinear relationships, which allows SVR to project the data into higher-dimensional spaces where a linear approximation is possible.

The choice of testing the SVR model for this specific problem was because of its mode of operation, that is less sensitive to outliers and noise (as it was explained before, the high values of the dataset could be considered outliers because of the data being right skewed), due to the fact that it allows some deviations on the data. It also allows more flexibility with the kernel function, such as linear, polynomial and radial basis functions that enables the model to fit on more complex relations.

ϵ	MSE	MAE	R^2 (training data)	R^2 (testing data)
0.1	0.00213	0	0.82136	0.82357
0.08	0.0014	0	0.8799	0.8844
0.04	0.00188	0	0.95576	0.95504
0.01	0.00021	0.00838094	0.9843	0.9826

TABLE 4. SVR Results

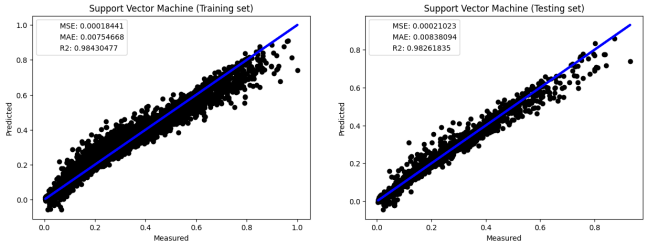

FIGURE 26. Plot of the Predicted vs True labels for the SVR model on training and testing data with $\epsilon = 0.01$

1) Hyper Parameters and Results

The kernel used was the Radial Basis Function (RBF) since it is the more generalized kerneland gama (kernel coefficient) it was set to scale that sets it to $1/(n_features * X.var())$ since it was the hyperparameters that gave better results. Then the ϵ was tested with several values starting on 0.1 to 0.01, since the epsilon specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value.

In table 4 we can see that the ϵ set to 0.01 does the best fit on this model to this specific regression problem.

C. HISTGRADIENTBOOSTINGREGRESSOR

HistGradientBoostingRegressior is a variation of the Gradient Boosting Machine. As the name suggests, it uses histograms to bin continuous feature values, which reduces the number of splitting points to consider and speeds up the training process. This approach makes it much more efficient on large datasets compared to traditional GBMs. It builds an ensemble of decision trees in a sequential manner. Each new tree is built to correct the errors made by the previous trees. It does this by fitting the new tree to the residual errors (the difference between the predicted and actual values) of the existing ensemble. The model iteratively adds trees, each time focusing on the parts of the training data that were not well predicted by the previous trees. We tried this model because it's capable of capturing complex, non-linear relationships in the data, it is suitable for data with high number of features and it is robust to outliers.

1) Hyper Parameters and Results

For the hyper-parameters on the HistGradientBoostingRegressor we had to choose the following:

- max_iterations: This parameter specifies the maximum number of trees in the boosting process. The algorithm will stop after the number of iterations specified unless it converges first.

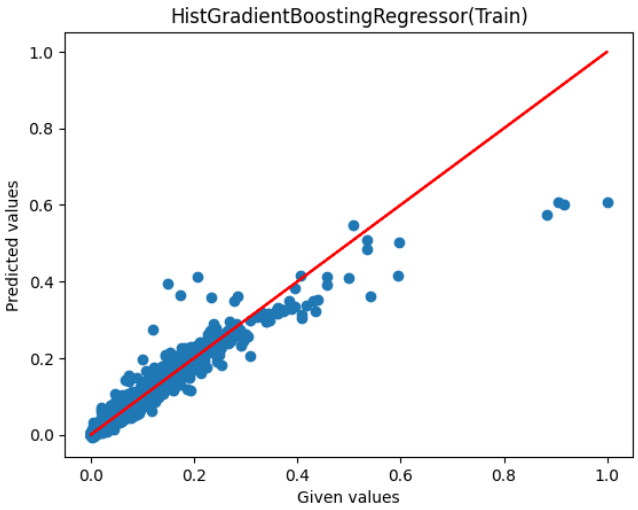

FIGURE 27. Plot of the Predicted vs True labels for HistGradientBoostingRegressor for training dataset

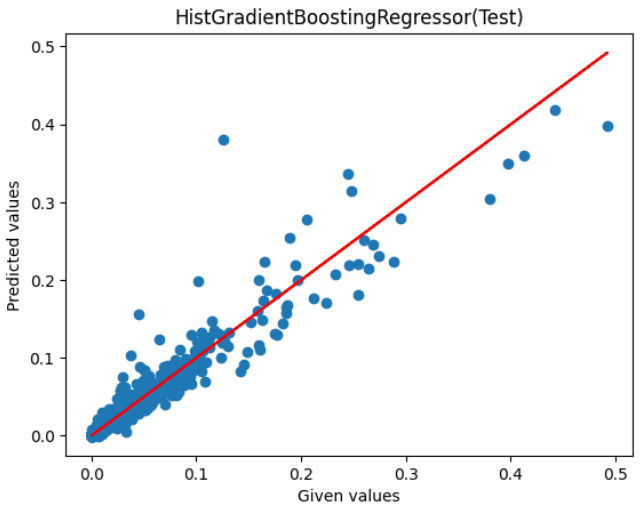

FIGURE 28. Plot of the Predicted vs True labels for HistGradientBoostingRegressor for testing dataset

- learning_rate: It controls the step size at each iteration while moving toward a minimum of a loss function.
- max_depth: This parameter sets the maximum depth of each tree. The depth of the tree is an indication of how many splits it has and consequently how complex the model is.
- min_samples_leaf: This specifies the minimum number of samples required to be at a leaf node.
- max_leaf_nodes: This limits the number of leaf nodes in the tree. Having more leaves makes the model more expressive, as it can learn more fine-grained patterns.

The hyper-parameters choosed were obtained by trial, and the best were 100000 max_iterations, learning_rate 0.01, max_depth to 100, min_samples_leaf 20 and max_leaf_nodes at 40.

The results obtained for MSE and MAE for testing data and R^2 Score for both training and testing data were 0.00002976, 0.00142745, 0.95808652 and 0.93429551, respectively.

VII. CONCLUSION

To wrap up, this study delves into the application of typical regressions techniques in the context of civil engineering problems. Through experimentation and analysis, the findings showcase the effectiveness of these methods in predicting critical bending moment of uniform and tapered beams. We find that using neural networks approaches could be overkill, since the dataset is relatively simple, meaning that it has a low number of features; Despite the fact that it will most likely get better results, the training times and complexity could get unsustainable. Thus, using methods that implement linear and polynomial regressions could handle these types of problems in a more fluid and agile way, since the training times are much lower, and the results are not that different. By taking a systematic approach to parameter selection and feature engineering we conclude that the proposed approach should a better way to handle this problem, which in turn is corroborated with good results. However, further research and refinement are essential to address challenges such as accuracy score. We think that this regression approach can still reach the accuracy the neural networks achieve. Research on different optimization algorithms, feature engineering, and cost functions should be the way and should pave the way for continued advancements in this field of study.

VIII. REFERENCES

References

[1] Abdelrahman Abushanab, Tadesse Gemeda Wakjira, and Wael Alnahhal. "Machine Learning-Based Flexural Capacity Prediction of Corroded RC Beams with an Efficient and User-Friendly Tool". In: *Sustainability* 15.6 (2023). ISSN: 2071-1050. URL: https://www.mdpi.com/2071-1050/15/6/4824.

[2] Carlos Couto. *Critical bending moment of uniform and tapered beams*. URL: https://github.com/ccouto/McrNet/tree/main/Dataset.

[3] Carlos Couto. "Neural network models for the critical bending moment of uniform and tapered beams". In: *Structures* 41 (2022), pp. 1746–1762. ISSN: 2352-0124. DOI: https://doi.org/10.1016/j.istruc.2022.05.096. URL: https://www.sciencedirect.com/science/article/pii/S2352012422004581.

[4] Jun-zhi Liu et al. "Machine learning (ML) based models for predicting the ultimate bending moment resistance of high strength steel welded I-section beam under bending". In: *Thin-Walled Structures* 191 (2023), p. 111051. ISSN: 0263-8231. DOI: https://doi.org/10.1016/j.tws.2023.111051. URL: https://www.sciencedirect.com/science/article/pii/S0263823123005293.

[5] Cailong Ma et al. "Prediction of shear strength of RC slender beams based on interpretable machine learn-

ing". In: *Structures* (Sept. 2023), p. 105171. DOI: 10.1016/j.istruc.2023.105171.

[6] M.S. Sandeep et al. "Shear strength prediction of reinforced concrete beams using machine learning". In: *Structures* 47 (2023), pp. 1196–1211. ISSN: 2352-0124. DOI: https://doi.org/10.1016/j.istruc.2022.11.140. URL: https://www.sciencedirect.com/science/article/pii/S2352012422011791.