XLVI CILAMCE
VITÓRIA - ES - BRAZIL - 2025
XLVI IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGINEERING

Analysis of Steel Profiles under Flexural-Compression stress using Machine Learning Algorithms

David S. Coimbra[1], André S. Müller[1]

[1]*Dept. of Civil Engineering, Federal Institute of Maranhão*
São Luís, 65030-005, Maranhão, Brazil
davidcoimbra@acad.ifma.edu.br, andre.muller@ifma.edu.br

Abstract. The structural design of steel members under flexural-compression demands a rigorous analytical approach due to the simultaneous presence of axial forces and bending moments. This interaction introduces geometric nonlinearities and instability phenomena, resulting in greater complexity than cases of pure tension or compression. Conventional verification methods are often inadequate for rapid and generalized assessments across diverse geometries and loading configurations. In this context, Machine Learning (ML) offers a promising alternative by capturing complex nonlinear relationships between input parameters and structural performance indicators. This study explores the application of supervised ML regression algorithms to predict the utilization factor—a continuous variable defined as the ratio between applied demands and the design resistance of steel members. The dataset was synthetically generated using a Fortran-based numerical model compliant with the NBR 8800:2024 code. Input features encompass cross-sectional geometry, applied loading, member length, and boundary conditions. The regression models evaluated include K-Nearest Neighbors, Decision Tree, Random Forest, and Light Gradient Boosting Machine (LightGBM), implemented through Python's Scikit-Learn framework. The dataset, comprising hundreds of thousands of samples, was divided into 80% for training and 20% for testing. Ensemble-based tree models, particularly Random Forest and LightGBM, demonstrated superior predictive accuracy.

Keywords: Flexural-compression, Machine learning, Utilization factor, Welded I-section.

1 Introduction

The design of steel structural elements presents numerous technical challenges due to the multitude of variables influencing their behavior. This complexity necessitates rigorous analytical approaches to balance structural safety, mechanical performance, and material efficiency. This study focuses on steel members under flexural-compression—a demanding loading condition characterized by structural instabilities and the interaction of axial and bending forces. Given its inherent complexity, this analysis benefits from computational methods capable of handling multiple interdependent variables.

In this context, artificial intelligence (AI) techniques, particularly Machine Learning (ML), are proposed not to replace conventional design methods, but to complement them. ML can optimize the design and verification process through predictive analysis and pattern recognition. Recent studies illustrate its growing relevance in structural engineering: Ribeiro et al. [1] predicted stress distribution in complex structures with openings to improve performance and ensure structural integrity; Moreira and Silva [2] developed models to estimate concrete compressive strength at different ages, enabling faster and more cost-effective evaluations; and Meneses and Sousa Jr. [3] applied ML to forecast key parameters in prestressed composite steel–concrete beams, contributing to optimized designs and enhanced performance.

Collectively, these advances demonstrate that ML extends beyond computational automation, serving as a strategic tool for improving structural behavior prediction, supporting engineering decision-making, and fostering innovation in structural design.

2 Methodology

This section details the methodology developed for predicting the utilization factor in steel profiles under flexural-compression using machine learning models. The process encompasses model development, validation, and implementation. First, fundamental theoretical concepts of ML are introduced. Subsequently, the input and output variables are described, together with the procedure for generating the dataset via numerically implemented Fortran routines. Finally, the implementation of the algorithms in Python is presented, covering the specific libraries employed, the training procedures, and the evaluation criteria adopted for assessing model performance.

2.1 Machine Learning

Machine Learning, a branch of artificial intelligence, focuses on developing algorithms that learn patterns from data without being explicitly programmed for each specific scenario. Mitchell [4] formally defines a machine learning algorithm as one whose performance at a task *T*, measured by a metric *P*, improves with experience *E*.

Supervised learning is an approach where the training dataset comprises both input features and their corresponding target values, known as labels. As noted by Géron [5], this paradigm is applicable to both classification tasks (e.g., spam detection) and regression tasks, which predict continuous numerical values from a set of input features.

This study employs regression algorithms, which model the mathematical relationship between input variables and a continuous target variable. Specifically, the objective is to predict the utilization factor—defined as the ratio between the applied loads and the design resistance of a steel member under flexural-compression.

2.2 Input and Output Variables

Welded I- and H-shaped steel profiles with double symmetry were considered in this study. The geometric dimensions of these profiles – including CS and CVS types – were obtained from technical catalogs of national manufacturers, in compliance with the Brazilian code for welded sections [6]. Boundary conditions were characterized using buckling coefficients (K), with values adopted according to the Brazilian steel structures code recommendations [7].

Member lengths of *3*, *6*, *9*, *12*, and *15 m* were considered in the analysis. Applied loads represented typical internal forces in steel columns: axial compression (N_{Sd}) ranging from *100kN* to *700kN*, in increments of *100kN*, and major-axis bending moment ($M_{x,Sd}$) ranging from *3kN·m* to *90kN·m*, in increments of *3kN·m*. Minor-axis bending moments are not considered in this study, since their inclusion would require a dataset significantly larger than what could be feasibly generated and processed in a standard spreadsheet environment. Therefore, the analysis is restricted to flexural-compression with continuous lateral restraint about the minor inertia axis.

The output variable - utilization factor - was calculated for compression-bending members using either Eq. (1) or Eq. (2) [7].

$$\boldsymbol{UF} = \frac{N_{Sd}}{N_{Rd}} + \frac{8}{9}\left(\frac{M_{x,Sd}}{M_{x,Rd}} + \frac{M_{y,Sd}}{M_{y,Rd}}\right) \text{ if } \frac{N_{Sd}}{N_{Rd}} \geq 0.2\,. \tag{1}$$

$$\boldsymbol{UF} = \frac{N_{Sd}}{2N_{Rd}} + \left(\frac{M_{x,Sd}}{M_{x,Rd}} + \frac{M_{y,Sd}}{M_{y,Rd}}\right) \text{ if } \frac{N_{Sd}}{N_{Rd}} < 0.2\,. \tag{2}$$

2.3 Fortran and Dataset Generation

The dataset was generated using a custom Fortran program that automated: (1) reading structural profile dimensions, and (2) defining combined compression-bending loading scenarios based on code specifications and practical structural cases. The mechanical properties reflected *ASTM A36* steel, with an elastic modulus E = *200GPa* and a shear modulus G = *77GPa*.

For each profile, the code computed geometric properties including cross-sectional area, moments of inertia,

elastic/plastic section modulus, and radius of gyration. Using these properties and boundary conditions, it calculated the utilization factor according to normative criteria across varying member lengths and load intensities. The resulting input variables were structured into tabular files compatible with spreadsheet software.

Table 1 presents representative samples from the final dataset. Each row corresponds to a unique combination of geometric properties, loading conditions, and boundary restraints. The buckling coefficient K was discretized into integer values from 1 to 6, representing standard scenarios. These values correspond to the following boundary condition configurations: (1) Fixed-Fixed, (2) Pinned-Fixed, (3) Guided-Fixed, (4) Pinned-Pinned, (5) Free-Fixed, and (6) Guided-Pinned.

Although the tables retain original profile names (e.g., CS200x56) for readability, these designations were converted into numerical categorical variables during model training, consistent with the treatment of the buckling coefficient K.

Table 1. Representative samples from the dataset generated for training and testing the machine learning models.

Profile (ID)	K	Length (m)	N_{Sd} (kN)	M_{Sd} (kNm)	UF
75	1	3.00	600.00	44.00	0.235
78	2	6.00	700.00	90.00	0.502
18	6	15.00	500.00	51.00	1.085
46	3	12.00	400.00	35.00	0.047
24	4	9.00	300.00	72.00	0.179
…	…	…	…	…	…

2.4 Python and Scikit-Learn

The machine learning algorithms were implemented in Python, leveraging its extensive ecosystem for data science applications. Python's versatility—particularly through libraries such as pandas—combined with its general-purpose capabilities, provides an optimal environment for data-centric development, as emphasized by McKinney [8].

The scikit-learn library was selected for algorithm implementation due to its computational efficiency, robustness, and seamless integration with essential scientific computing tools like NumPy and pandas. This integration facilitated streamlined data processing and statistical analysis throughout the modeling workflow.

2.5 Algorithms and Training

The K-Nearest Neighbors (KNN) algorithm, originally introduced by Fix and Hodges [9] and later extended by Cover and Hart [10], predicts a target value for a new instance based on the k nearest neighbors in the feature space. In this regression context, the prediction corresponds to the average target value of these neighbors. Input features were standardized to ensure fair distance calculations, preventing variables with larger scales from dominating the analysis.

Decision Trees are non-parametric supervised learning methods that build hierarchical models using simple decision rules derived from data features. The implementation used the CART (Classification and Regression Trees) algorithm, which constructs binary trees by selecting the split that maximizes information gain at each node.

To overcome the limitations of individual trees, the Random Forest algorithm, proposed by Breiman [11], combines multiple decision trees trained on random subsets of both data and features. This ensemble approach reduces model variance, enhances robustness, and mitigates overfitting.

LightGBM, developed by Ke et al. [12], is a gradient-boosting framework that employs histogram-based optimization and a leaf-wise tree growth strategy, offering high computational efficiency and scalability for large datasets. It was selected over alternatives like XGBoost due to its superior processing speed without compromising predictive accuracy.

Algorithms such as Support Vector Regression (SVR) and Multi-Layer Perceptron (MLP) were excluded due to their substantial computational requirements, particularly with large datasets. Model selection therefore prioritized computational efficiency alongside strong predictive performance.

Hyperparameter tuning was conducted using Random Search [13] for selected algorithms. For others, stable

default values reported in the literature were adopted, given their consistent performance across similar datasets.

3 Results and Discussion

The optimal hyperparameters identified for each algorithm are summarized in Tab. 2. Subsequently, Tab. 3 presents the results obtained by the models on the test set, considering the metrics R^2, mean squared error (MSE), mean absolute error (MAE), and root mean squared error (RMSE), allowing a direct comparison of the performance of the trained models. Additionally, Figure 1 shows the comparisons between the expected and predicted values for each algorithm on the test samples, providing a visual assessment of the prediction accuracy.

Table 2. Hyperparameters for each supervised regression algorithm.

Algorithm	Hyperparameters		
KNN	n_neighbors = 10	Weights = uniform	metric = Euclidean distance
Decision Tree	max_depth = 14		
LightGBM	Learning_rate = 0.1		n_estimators =200
Random Forest	n_estimators = 100		

Table 3. Model performance on the test set based on R^2, MSE, MAE, and RMSE metrics.

Algorithm	R^2	MSE	MAE	RMSE
KNN	0.90	0.10	0.11	0.32
Decision Tree	0.99	0.00	0.02	0.04
LightGBM	1.00	0.00	0.01	0.02
Random Forest	1.00	0.00	0.00	0.00

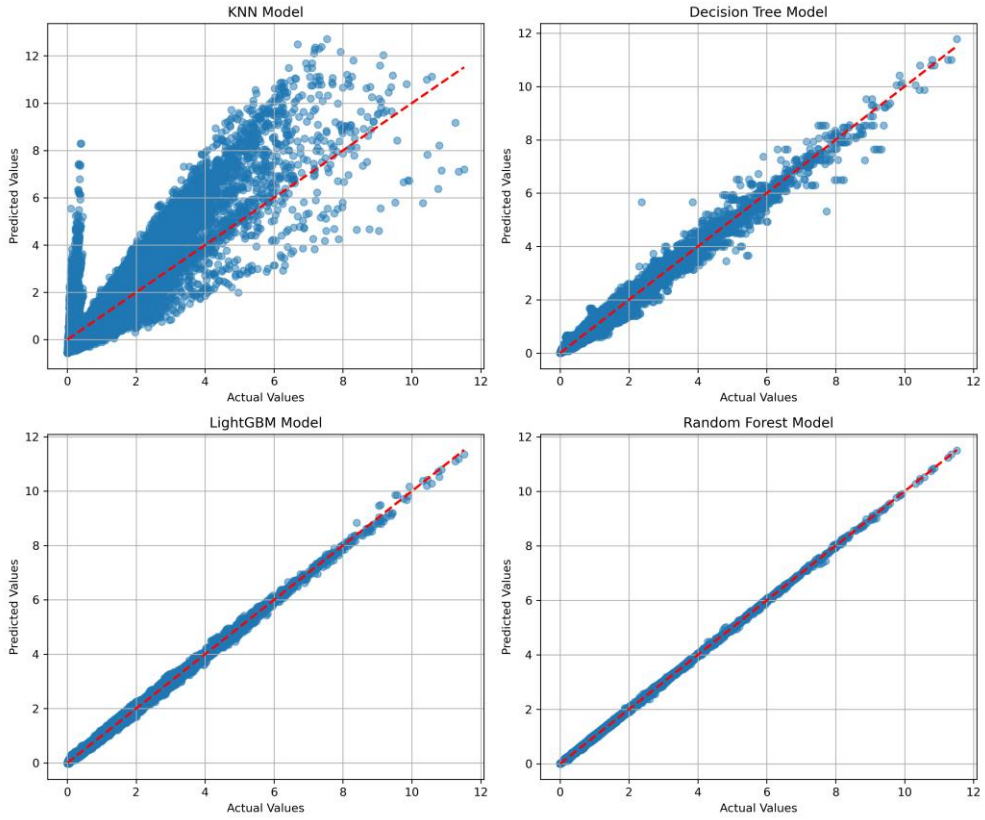


Figure 1. Comparison between Actual and Predicted Values.

To demonstrate the practical performance of the algorithms, consider the column in Fig. 2, composed of a welded I-section. The loading configuration involves the simultaneous action of axial force and bending moments. The boundary conditions consider a movable fixed support at the top and a pinned support at the base for the x-x plane. In the y-y plane, the column is laterally restrained and rotationally fixed, which prevents the development of bending moments about this axis. By means of the algorithms, we will check whether the CS350x86 profile is suitable.

Table 4 presents the comparison between the UF estimated by the machine learning models and the reference value calculated according to NBR 8800 [7], as well as the corresponding relative error. A significant difference is observed between the normative UF and that predicted by the algorithms. Among the evaluated models, the Random Forest achieved the best performance, closely matching the normative value.

The results obtained in this study were satisfactory, indicating that the proposed methodology can provide reliable predictions under the specific conditions analyzed. However, these outcomes are limited to a constrained scenario: bending moments acting exclusively on one principal axis, a defined and relatively narrow range of applied loads, and fixed member lengths. Such constraints stem primarily from the characteristics of the training dataset, limited by the maximum row capacity of Excel (2^{20} rows).

For future developments, it is intended to broaden the scope of the analysis by employing a database with a capacity beyond that of Excel, enabling the inclusion of a substantially larger number of loading combinations. The extended dataset will allow for a wider load range to be considered, as well as the incorporation of bending moments acting simultaneously on both principal axes. These enhancements are expected to improve the robustness and generalizability of the models, making them applicable to a broader spectrum of structural design situations.

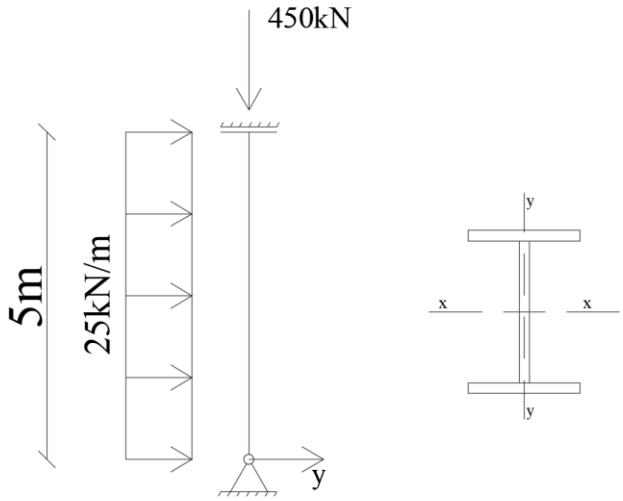


Figure 2. Welded I-section column under flexural–compression used for the evaluation of the algorithms.

Table 4. Comparison between utilization factors estimated by the machine learning models and those calculated according to NBR 8800, with corresponding relative errors.

Algorithm	UF (Estimate by model)	UF (NBR 8800)	ERROR (%)
KNN	0.327	0.246	32.927
Decision Tree	0.253		2.846
Random Forest	0.244		0.813
LightGBM	0.261		6.098

4 Conclusions

This study presented one of the first applications of machine learning models to predict the utilization factor of steel profiles subjected to flexural-compression, in accordance with the criteria established by the Brazilian standard for steel structures. The methodology involved generating a robust dataset using routines developed in Fortran, followed by training regression models using well-established Python libraries.

The results demonstrated that the algorithms effectively captured the structural behavior of elements under combined loading, with tree-based models and ensemble techniques standing out for their superior predictive performance on the test set. Future work will focus on expanding the dataset with additional variables and exploring other algorithms, such as neural networks, to further enhance prediction accuracy.

Acknowledgements. The authors would like to acknowledge IFMA (Federal Institute of Maranhão) for financing this research.

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] J. A. Ribeiro et al., “Advancing structural design with machine learning: Stress field prediction in plates with cutouts,” XLIV Ibero-Latin American Congress on Computational Methods in Engineering, vol. 5, no. 05, 2023.
[2] P. B. A. Moreira and V. M. Silva, “Concrete compressive strength prediction with machine learning,” XLI Ibero-Latin American Congress on Computational Methods in Engineering, vol. 2, no. 02, 2020.
[3] J. B. M. de Sousa Junior and B. M. Meneses, “Contributions to the analysis of prestressed steel and concrete composite beams using machine learning algorithms,” Ibero-Latin American Congress on Computational Methods in Engineering (CILAMCE), 2024.
[4] T. M. Mitchell, Machine Learning. New York: McGraw-Hill, 1997.
[5] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, 2nd ed., O’Reilly, 2019.
[6] Associação Brasileira de Normas Técnicas, “NBR 5884:2013 – Perfil I estrutural de aço soldado por arco elétrico – Requisitos gerais,” Rio de Janeiro, 2013.
[7] Associação Brasileira de Normas Técnicas, “NBR 8800: Projeto de estruturas de aço e de estruturas mistas de aço e concreto de edificações,” 3ª ed., Rio de Janeiro, 2024.
[8] W. McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2nd ed. Sebastopol: O’Reilly Media, 2018.
[9] E. Fix and J. L. Hodges, “Discriminatory analysis. Nonparametric discrimination: Consistency properties,” Report 4, Project 21-49-004, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.
[10] T. M. Cover and P. E. Hart, “Nearest neighbor pattern classification,” IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21–27, 1967, doi: 10.1109/TIT.1967.1053964.
[11] L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
[12] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, “LightGBM: A highly efficient gradient boosting decision tree,” Advances in Neural Information Processing Systems, vol. 30, 2017.
[13] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” Journal of Machine Learning Research, vol. 13, pp. 281–305, 2012.