mathematics

MDPI

Article

Lead-Time Prediction in Wind Tower Manufacturing: A Machine Learning-Based Approach

Kenny-Jesús Flores-Huamán, Alejandro Escudero-Santana *, María-Luisa Muñoz-Díaz and Pablo Cortés

Departamento de Organización Industrial y Gestión de Empresas II, Escuela Técnica Superior de Ingeniería, Universidad de Sevilla, Cm. de los Descubrimientos, s/n, 41092 Seville, Spain; kflores1@us.es (K.-J.F.-H.); mmunozd@us.es (M.-L.M.-D.); pca@us.es (P.C.)

* Correspondence: alejandroescudero@us.es; Tel.: +34-95-448-60-42

Abstract: This study focuses on estimating the lead times of various processes in wind tower factories. Accurate estimation of these times allows for more efficient sequencing of activities, proper allocation of resources, and setting of realistic delivery dates, thus avoiding delays and bottlenecks in the production flow and improving process quality and efficiency. In addition, accurate estimation of these times contributes to a proper assessment of costs, overcoming the limitations of traditional techniques; this allows for the establishment of tighter quotations. The data used in this study were collected at wind tower manufacturing facilities in Spain and Brazil. Data preprocessing was conducted rigorously, encompassing cleaning, transformation, and feature selection processes. Following preprocessing, machine learning regression analysis was performed to estimate lead times. Nine algorithms were employed: decision trees, random forest, Ridge regression, Lasso regression, Elastic Net, support vector regression, gradient boosting, XGBoost, LightGBM, and multilayer perceptron. Additionally, the performance of two deep learning models, TabNet and NODE, designed specifically for tabular data, was evaluated. The results showed that gradient boosting-based algorithms were the most effective in predicting processing times and optimizing resource allocation. The system is designed to retrain models as new information becomes available.

Keywords: industrial machine learning; deep learning; regression; process time; prediction

MSC: 68T05; 68U35; 68M20; 90B06; 90B90

Citation: Flores-Huamán, K.-J.; Escudero-Santana, A.; Muñoz-Díaz, M.-L.; Cortés, P. Lead-Time Prediction in Wind Tower Manufacturing: A Machine Learning-Based Approach. *Mathematics* **2024**, *12*, 2347. https://doi.org/10.3390/math12152347

Academic Editor: Janez Žerovnik

Received: 14 June 2024
Revised: 23 July 2024
Accepted: 25 July 2024
Published: 27 July 2024

1. Introduction

In recent decades, the field of artificial intelligence has undergone a significant transformation. After experiencing a period of stagnation known as the "AI winter", characterized by a sharp decline in both funding and interest from the business sector due to the limitations of AI at that time, this field has become an essential and indispensable tool in everyday life and across various industries.

Among the different branches of AI, machine learning has emerged as an ideal technology for extracting information from collected data. This technology has the capability to provide predictions and discern complex patterns, leading to the development of intelligent systems. These systems are crucial in various manufacturing and logistic tasks, such as predictive maintenance, quality improvement, supply chain management, task scheduling, and process optimization, among others [1].

The wind energy research industry has primarily focused on the operational phase of wind turbines, spanning from the initiation of electricity generation to the end of their lifespan [2]. However, additional critical stages, such as the construction of wind turbine towers, stand to significantly benefit from the application of machine learning techniques. Strategically integrating machine learning at this stage would enable the prediction of the total flow or completion time, from component arrival to project completion. This

application has been considered strategic in this and other industries since process and lead times are key factors in providing reliable schedules and assessing overall manufacturing process efficiency.

Furthermore, some researchers have begun to focus on estimating the lead time of individual processes, which involves the precise identification of the duration of each phase within the manufacturing process. This is crucial as, by decomposing and thoroughly analyzing each production phase, congestion points and areas for improvement can be identified. Integrating machine learning techniques at this stage allows for the accurate prediction of the execution time for each process and optimizes resource allocation, improving coordination between production stages and avoiding delays. Furthermore, machine learning could be employed to estimate production costs, thereby eliminating the limitations of current techniques such as direct formulation or linear programming. While these traditional methods are commonly used, direct formulation often fails to realistically reflect process times or costs, and linear programming may struggle to adapt in dynamic production environments.

Deep learning has experienced exponential growth in recent years, particularly in processing unstructured data such as text and images. In natural language processing (NLP), recurrent architectures have been fundamental for developing machine translation applications, enhancing the machines' ability to understand and generate text [3]. Similarly, in the field of computer vision, convolutional neural networks have revolutionized image analysis and recognition, being applied in tasks such as image classification [4]. However, recent years have seen significant efforts to advance the development of neural networks for structured data, such as database tables. Models such as TabNet and NODE have emerged as promising alternatives for handling tabular data, offering competitive performance alongside good interpretability [5,6]. Therefore, in this paper, we aim to test two deep neural network algorithms for tabular data, specifically TabNet and NODE, and evaluate their suitability for use in an industrial process.

The main objective of this paper is to research, design, develop, implement, and evaluate machine learning and deep learning algorithms on tabular data to estimate the lead time of each phase in the context of wind turbine tower manufacturing. This improved information will enable two important aspects of the factories to be optimized: (a) efficient process scheduling and resource allocation, and (b) accurate production cost estimation.

During the development of this study, an exhaustive process was undertaken to implement machine learning algorithms in the context of wind turbine tower manufacturing. Special emphasis was placed on the process of data identification to ensure the relevance and accuracy of the data used. Data were collected from 11 January to 22 December 2022 at wind turbine tower construction facilities located in Spain and Brazil, and are described in more detail in Section 3. The dataset resulting from the research process comprises 231,678 observations collected from both factories. For each analyzed operation, specific machine learning algorithms were trained using relevant subsets of the dataset. These data underwent rigorous preprocessing, which included error correction, handling of missing values, and the selection of the most relevant features for each operation in collaboration with the engineering departments of the plants.

The innovations and contributions of this paper are as follows:

1. Design and development of a system employing various machine learning algorithms to predict processing times in the context of the efficient scheduling of different processes and resource allocation. Additionally, this system can be used for accurate production cost estimation.
2. Evaluation of tabular neural network algorithms to determine their usefulness in an industrial context.
3. Analysis and comparison of various evaluation metrics relevant to regression problems within this context. The aim is to enhance understanding and identify the most suitable metric for effectively comparing the performance of the proposed models.

4. Incorporation of a feature into the system that allows for retraining models as they degrade over time, using new records to keep them updated.
5. Validation of the proposed methodology through the use of various datasets obtained from two wind turbine tower construction factories located in Spain and Brazil.

Our study aims not only to address the specific challenges of wind turbine tower manufacturing but also to lay the groundwork for future research and applications in the field of machine learning applied to industry. By developing and evaluating algorithms to optimize resource allocation and cost estimation in this particular context, we hope to improve the efficiency of wind turbine tower production and inspire new research at the intersection of AI and the energy industry. This work could serve as a starting point for the development of broader and more customized solutions in other areas of industrial production, paving the way for smarter, more efficient, and sustainable manufacturing.

The rest of this article is structured as follows. Section 2 provides the context for the construction of wind turbine towers, outlines the relevant processes, and discusses the use of machine learning in wind energy and wind tower manufacturing. Section 3 details the proposed methodology used in this study. Section 4 presents the results derived from analyzing various datasets and the implementation of the system. Section 5 provides a summary and discussion of the analysis results. Finally, Section 6 discusses the conclusions of this work and suggests directions for future research.

2. Machine Learning in Wind Energy and Wind Tower Manufacturing

The construction of a wind turbine is divided into a number of independent manufacturing processes, which, when properly assembled, result in an operational wind turbine. Among the various components of a wind turbine, the tower stands out as one of the most critical elements in the construction of the turbine. It needs to be tall enough to capture strong and stable winds, as well as robust enough to withstand the forces of the wind and the vibrations generated by the rotor. In addition, towers can be difficult to transport and represent a significant proportion of the manufacturing cost of a wind turbine. To provide context for this study, the manufacturing process of a wind tower will be broken down.

The manufacturing process for these structures is complex and demanding, involving the handling of large products and numerous manual operations. The first step in manufacturing a wind turbine tower is to select the appropriate materials, with steel or concrete plates being the primary raw materials due to their strength and durability. These plates are then cut to the tower design specifications, and their edges are prepared for welding through a process known as beveling. The plates are then bent to form rings, referred to in this document as 'cans', and longitudinal welds are made at their ends.

Once several cans have been produced, they are joined together by circular welding to form a tower section. Towers are manufactured in sections for ease of transportation, as it would be impractical to transport a complete tower. In addition, flanges are added to both ends of each section to facilitate assembly. The assembled sections then undergo quality testing and surface treatments, such as painting or corrosion protection, to ensure longevity and prepare internal components and doors. Finally, the tower sections are transported to the installation site where they are mounted and assembled to complete the tower.

Sainz detailed the different stages of the process and pointed out several automation technologies that could increase the production capacity of the industry [7]. In a wind turbine manufacturing facility, each process is divided into a specific number of tasks and operations, which depend on several variables, contingent upon various influencing factors. These operations and variables were subjected to a rigorous analysis in order to identify which of these variables are significant in determining the operation lead time. The following bullet points delve into the diverse processes scrutinized within this study, with the delineation of variable selection deferred to Section 3.

- Plate Cutting: The plate-cutting process involves cutting raw steel plates to produce sheets with the perimeter and length required to form the cylinders. This operation

can be carried out in various ways, depending on the thickness of the steel, using techniques such as plasma cutting or oxy-fuel cutting.

- Plate Beveling: The plate-beveling operation consists of cutting the edges of a steel plate at an angle other than 90 degrees to obtain a stronger and higher quality weld joint. This process is performed on the edges of the previously cut plate, which is then joined by welding to form the cylinder.
- Plate Bending: The plate-bending process involves shaping the steel plates, previously cut and beveled, into a cylindrical form to create the cylinders that constitute the tower section. This is done using bending machines or roller presses, which exert pressure to shape the plate as desired.
- Longitudinal Welding: In the longitudinal welding operation, the ends of the steel plate, previously cut, beveled, and bent, are joined by welding to transform the plate into a ring. This process results in the formation of a cylinder.
- Rebending: In certain circumstances, a cylinder stored for a prolonged period can become deformed due to its own weight. For this reason, before being sent to other areas, it is necessary to restore it to its original ring shape. The rebending process refers to correcting the deformations that may have occurred during storage or due to defects in the bending process.
- Fit-Up and Circular Welding: The fit-up process involves joining different cylindrical cylinders to construct a complete section of the tower through the circular welding process. Additionally, at this stage, flanges are installed at both ends of the segment to facilitate the joining of sections during the assembly phase.

The majority of the scientific literature applying machine learning to the wind energy sector focuses on the operational phase of a project when wind turbines are in full energy production [8]. Three main lines of research stand out in this area.

Firstly, the field of prediction and data analysis has seen significant interest in forecasting the electrical output of wind turbines using various types of data, such as historical records [9] or wind speeds [10–15]. Methods such as decision trees, random forests, gradient boosting, and multilayer perceptrons (MLPs) have been employed to understand the complex relationship between wind conditions and energy production in wind farms [16]. Additionally, MLPs have been used to predict wind speeds on an hourly basis [17].

Deep learning techniques have also been applied in this area. Researchers have used convolutional neural networks (CNNs) to detect objects, thereby identifying locations for offshore wind energy infrastructure [18]. Another study employed CNNs and support vector machines (SVMs) to create a bird identification system designed for offshore wind farms [19,20]. Furthermore, reinforcement learning has been explored to develop a hybrid route-planning method for navigation in areas with offshore wind farms [21].

In the area of optimization and control, research has utilized deep reinforcement learning to develop a wind farm control scheme aimed at optimizing energy generation under various wind conditions [22]. Another study used graph-based neural networks to connect wind turbines based on their geographical location, aiming to optimize turbine layout and improve energy production [23]. Additionally, a multi-objective predictive control strategy for floating wind turbines using recurrent neural networks has been proposed [24].

Lastly, in the area of maintenance and health monitoring, researchers have applied self-organizing maps and multilayer perceptrons to efficiently plan and execute maintenance and repair tasks [25]. Others have focused on the visual inspection of wind turbines using R-CNNs, enhancing early problem detection and optimizing maintenance procedures. Moreover, a fault detection and diagnosis system for the pitch control system of wind turbine blades has been developed using a hybrid approach with a Kalman filter and multilayer perceptron [26].

The literature addressing the operational aspects of wind turbine manufacturing, particularly focusing on fundamental operations and activities, remains rather limited. One study explored the variability in delivery times associated with the bending process in wind

turbine tower manufacturing, utilizing machine learning techniques. The objective was to comprehend the impact of various factors on these times, thereby enhancing plant planning and control [2]. An important aspect of the aforementioned study was the inclusion of operator-related variables. Another research paper proposed a machine learning-based system employing regression models to forecast lead times in sequential manufacturing processes. The efficacy of the system was validated on a wind turbine tower manufacturer, demonstrating improved predictive accuracy, particularly in longitudinal welding operations [27]. The study incorporated the field of knowledge of how variables in upstream processes can influence downstream processes. Existing studies have concentrated on the prediction of one or two specific processes. However, no research has provided a comprehensive prediction of all stages of wind tower production.

3. Proposed Methodology

The methodology behind this study involves developing a system that employs machine learning algorithms to predict processing times in the context of production control and the estimation of production costs. For this purpose, a methodology divided into the following stages has been designed: acquisition and exploration of data, data preprocessing, training and evaluation of algorithms, and generation of time information.

3.1. Acquisition and Exploration of Data

In industrial environments, a significant portion of data originates from both the company's Enterprise Resource Planner (ERP) and the Manufacturing Execution System (MES). Additionally, it is supplemented with insights gleaned from interviews with operators, who provide valuable perspectives on relevant variables and potential shortcomings in data collection.

The collected data are stored on various internal company storage systems, including an internal cloud storage platform. This private cloud infrastructure offers advantages in terms of scalability, accessibility, and data security while maintaining full control over the data within the company's own computing environment. The use of internal cloud storage is especially beneficial in this research due to the large volume of data collected from multiple factories, which facilitates access to data from different locations and collaboration between team members. A wide range of information is thus obtained, covering project-specific technical data such as part dimensions, material consumption, and production times calculated using formulas. In addition, operating hours are recorded in the ERP system, providing a detailed overview of plant activity.

Information regarding recorded delivery times and machine usage is input by plant workers at the end of each operation. Hence, it is essential to consider that data quality may vary due to human error or the presence of missing values. These errors can impact the accuracy of subsequent analyses and must be addressed through data cleaning and validation techniques.

For this research, data were collected throughout the year 2022 from wind tower construction factories located in Spain and Brazil. In the Spanish factory, a total of 20,938 temporal entries were recorded in the system, while the Brazilian counterpart accumulated 210,740 temporal records. This extensive dataset provides a robust foundation for analysis and the identification of areas for improvement in manufacturing processes.

Due to the variety of sources providing data and the necessity to create a dataset for each operation and each factory, it was essential to implement a procedure that enabled the efficient integration of all information within each operation we aim to analyze, in addition to performing an exploratory analysis of the data. This process is illustrated in Figure 1. Table 1 compares the sizes of datasets for each operation across different factories.

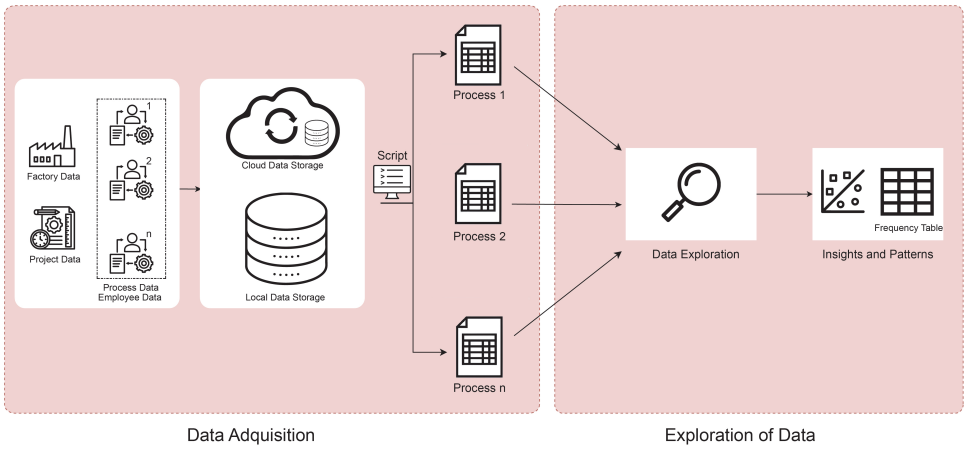

Figure 1. Data acquisition and exploratory data analysis (EDA) workflow.

Table 1. Comparison of dataset sizes for each operation in the respective factories.

Factory/Operation	Cutting	Beveling	Bending	Long. Welding	Rebending	Fit-Up	Circ. Welding
Spain	2163	2135	1985	1987	1432	2150	185
Brazil	17,650	17,704	18,511	16,961	1244	5694	1910

One of the challenges identified in carrying out this task was the manner in which the data are recorded by the ERP system. The recorded operations provide information on the start and end of activities input by the operators. This means that to complete a single operation, multiple entries may have been logged in the ERP. These multiple entries can be due to the process being performed at different times or even by different individuals. Despite these challenges, effective solutions were successfully implemented.

Each dataset contains numerous columns. Below, we present the columns that are common across all datasets:

- Start Date: The date when the process began.
- Start Time: The time when the process began.
- End Date: The date when the process ended.
- End Time: The time when the process ended.
- Area: The area where the process was being carried out.
- Center: The factory where the process was carried out.
- Project: The identification of the project to which it belonged.
- Can: The can of a section to which it belonged. (This was only available if it was an operation where a record is a can; for example, in circular welding, where we work with sections, this did not exist).
- Section: The section of the tower to which it belonged.
- Tower: The tower to which this record belonged.
- Employee Name: The name of the employee(s) assigned to perform a process.
- Workstation: The identifier of the workstation where the process was performed.
- Reported Operation Time: The time recorded for the completion of an operation.
- Downtime: The time during which the process was halted.
- Deviation Reason: The description of the reason why an activity was not performed as planned.
- Plate Properties: The main characteristics of a plate, such as its thickness and length, among others.
- Machine: The machine used in this process.

Additionally, each process had different attributes that were added, such as the types of bevels used in the beveling process.

After collecting the data, an initial exploration was carried out to identify patterns, trends, and possible outliers in each operation. This exploratory analysis relied on a variety of statistical methods and visualization tools to examine the data distribution, as well as to identify possible relationships between variables and detect significant patterns.

It is important to highlight that much of the information about waiting times and machinery usage is entered by workers during operation, which can result in errors. Additionally, it should be considered that the workflow may vary significantly between the two countries.

In summarizing the analyses conducted, it was observed that most of the numerical attributes in each of the processes show an asymmetric distribution, both positive and negative, suggesting a deviation from the normal distribution. Additionally, no significant linear relationship was identified between these attributes and the total execution time.

Anomalies are patterns in data that deviate from normal instances. Detecting these anomalies is of utmost importance, as they can identify issues or failures in data collection or quality. Moreover, this contributes to maintaining data integrity and improving efficiency and performance in general. One widely used technique for anomaly detection is the isolation forest algorithm [28]. This algorithm is based on the premise that anomalous instances are easier to isolate than normal instances

The algorithm works by constructing multiple isolation trees, also known as iTrees. An iTree is a special type of binary decision tree that, instead of classifying or predicting values, aims to isolate all data observations in individual leaves for subsequent analysis of which attributes are anomalous.

The procedure for building these trees involves the selection of an attribute and a cutoff value within the range of that feature. Significantly, the cutoff value is not predetermined but rather randomly selected within the feature's range. This inherent randomness constitutes a crucial aspect of the unsupervised nature of the isolation forest algorithm, thereby enabling its efficacy in effectively isolating anomalies. The dataset is then divided based on that feature and cutoff value. This process is repeated recursively until a stopping condition is met. As shown in Figure 2, anomalous observations tend to be located in the shallower branches of the tree, while normal observations require more divisions, placing them in deeper branches.

To increase the detection capability, an ensemble of isolation trees is constructed to assign anomaly scores to the observations, where the lowest scores indicate a higher probability of being anomalies.

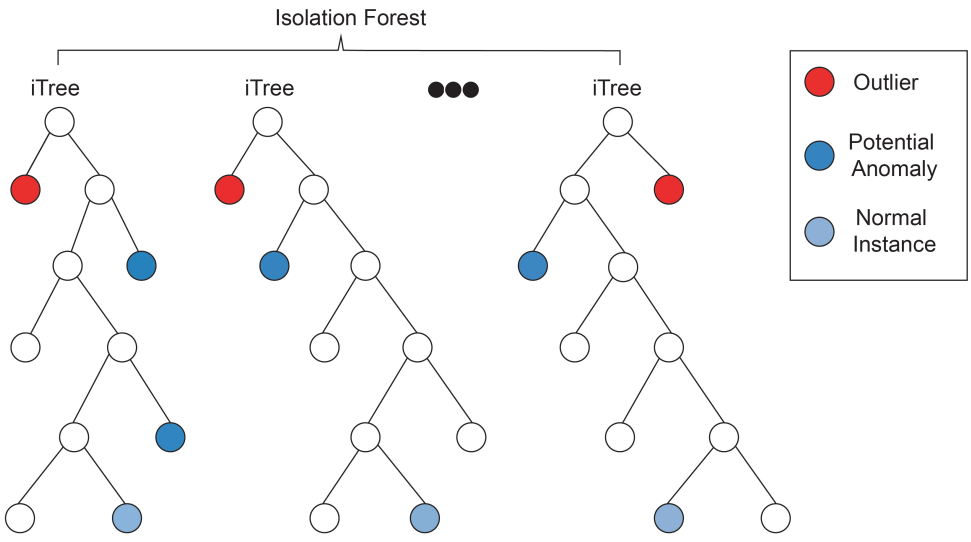

Figure 2. Isolation forest construction.

The isolation forest algorithm was employed to detect potential anomalies in the operation times of the various processes in the factory. It was found that approximately 15–20% of the recorded times for each operation were considered outliers. Addressing these anomalous data points may improve the overall efficiency, productivity, and quality of the products being produced.

3.2. Data Preprocessing

To enhance the quality of information and facilitate the development of predictive models, an exhaustive process of data cleaning and transformation was carried out. This process included error correction in texts, imputation of missing values, and normalization of attributes. Additionally, feature engineering techniques were applied to optimize predictive quality, removing irrelevant attributes and adding new attributes based on knowledge provided by the company's engineering department. In Figure 3, the flow of how a process is cleaned can be observed.

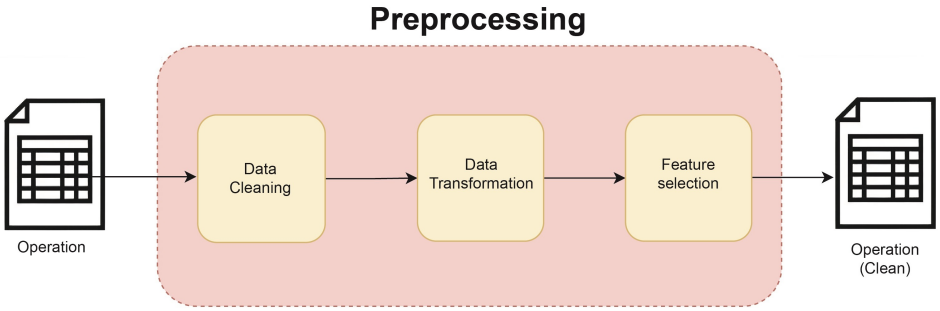

Figure 3. Data preprocessing.

3.2.1. Data Cleaning

The data cleaning process aims to enhance data quality through various techniques, including the removal of duplicate rows and the treatment of missing values.

Although no duplicate rows were identified in the datasets, missing values were detected due to data shortages in the projects or difficulties in reading the information. To address these missing values, the option to delete instances with missing values was dismissed due to the small size of the datasets. Instead, imputation was selected, utilizing the mean for numerical attributes and the mode for categorical variables.

3.2.2. Data Transformation

Although datasets may include categorical variables, most machine learning algorithms and libraries are tailored to handle numerical data. Consequently, it is imperative to transform these columns to ensure precise data interpretation. Additionally, certain models, such as neural networks, can be highly sensitive to data scale, necessitating adjustments to these features for an optimal representation [29].

Data scaling is a fundamental procedure used to standardize variables to a common scale. Its aim is to eliminate disparities in variable magnitudes and ensure optimal performance in machine learning models and algorithms. Various data scaling methods are employed, including standardization and normalization.

Standardization involves adjusting the data such that the mean of the observed values is equal to 0 and its standard deviation is equal to 1. This is achieved through the following formula:

$$X_{\text{std}} = \frac{x - \mu}{\sigma} \tag{1}$$

Normalization adjusts the data distribution to fit within a range between 0 and 1. There are several normalization methods, but one of the most renowned is min-max normalization. The formula for this method is as follows:

$$X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \quad (2)$$

Here, X represents the current value, X_{min} is the minimum value, and X_{max} is the maximum value in the dataset.

It is crucial to note that, besides normalization and standardization, there exist various scaling methods, such as robust scaling. The choice of an appropriate method largely depends on the nature of the dataset and the specific problem being addressed. In this study, normalization was chosen due to its ability to adjust all data within a range of 0 to 1. This technique enabled us to mitigate disparities in the magnitudes of dataset features, thereby facilitating a more equitable comparison among them.

Furthermore, it is essential to consider that algorithms such as support vector machines (SVMs) and neural networks are extremely sensitive to the scale of data. Normalization ensures a uniform scale that is vital for these models, significantly improving their performance and stability.

In addition to data scaling, the encoding of categorical variables is crucial for machine learning applications. Categorical variables are attributes that represent different categories, such as the color of a painting or the type of animal. To utilize these variables in machine learning models, it is necessary to convert them into numerical formats understandable by algorithms. Two common techniques for this conversion are ordinal encoding and one-hot encoding.

Ordinal encoding assigns a numerical value to each category based on a specific order, while one-hot encoding creates a new column for each category, assigning a 1 if the observation belongs to that category and a 0 otherwise.

In our study, we evaluated several encoding techniques and found that one-hot encoding was most suitable for our purposes. This technique has demonstrated robust performance in most cases, despite the resulting increase in dimensionality.

3.2.3. Feature Selection

Feature selection involves identifying the most relevant and representative variables within a dataset to enhance data precision and efficiency. It is a key process in data preprocessing aimed at reducing data dimensionality by eliminating uninformative or noisy features. It is categorized into two main types: unsupervised methods, which identify relevant features based on data structure and distribution, and supervised methods, which utilize label or target value information for feature selection. In our practical case, we employed unsupervised methods to eliminate redundant features and collaborated with the engineering department to enhance predictive quality for specific operations. Table 2 presents the most relevant features considered for each operation under study.

Table 2. Feature selection across various wind tower manufacturing processes.

Feature	Cutting	Beveling	Bending	Long. Welding	Rebending	Fit-Up	Circ. Welding
Plate thickness	✓	✓	✓	✓	✓	✓	✓
Plate length	✓		✓		✓		
Plate width	✓			✓			
Cutting perimeter	✓						
Machine type	✓		✓	✓	✓		
Bevel type		✓					
Plate perimeter		✓					
Taper of ferrule			✓			✓	
Ferrule Diameter						✓	✓
Presence of flange						✓	

In this context, the selection of attributes was determined by cost estimation. However, when developing models for scheduling purposes, it may be necessary to include additional attributes, such as the workers assigned to a specific process or other relevant resources.

3.3. Training and Evaluation of Algorithms

Once the data were prepared for implementation, the next step was to select the optimal model for each of the relevant operations in this study. Therefore, in the following sections, we discuss the training and evaluation processes for each of these relevant operations.

3.3.1. Machine Learning Algorithms Employed

The *No-Free-Lunch* theorem highlights the absence of a universal optimal solution for all problems [30]. In the field of machine learning, this premise implies the need to explore and evaluate a variety of models and approaches to find the most suitable solution within a specific business context. Since each problem presents unique characteristics and particular challenges, it is crucial to conduct a comprehensive evaluation of various models using relevant datasets.

In our study, we employed a variety of machine learning algorithms to address the resolution of our problems. These algorithms can be broadly classified into tree-based methods, regularized linear regression methods, support vector regression methods, models based on gradient boosting, and the multilayer perceptron.

Tree-based methods, such as decision trees (DTs) and random forests (RFs), are popular due to their simplicity and interpretability.

A DT is a graphical representation model used for decision-making in regression and classification problems, structured as a tree with decision and response nodes. The decision nodes pose questions about attributes and divide the data into branches, while the response nodes provide the final decision. The CART algorithm, one of the most employed methods, constructs binary trees based on the reduction of metrics such as the MSE, which measures the average of the squared errors between predictions and actual values. The process continues recursively until stopping criteria are met, producing a decision tree that predicts continuous values with greater accuracy and can be used on new data samples [31].

RF is a machine learning ensemble method that combines multiple DTs to improve accuracy and reduce overfitting in classification and regression problems. Each tree is built using different data samples and randomly selected features, introducing diversity among the trees. The final prediction is obtained by averaging the individual predictions of the trees in the case of a regression problem [32]. This combination of predictions improves the model's generalization capacity and makes it more robust and accurate. Additionally, random forest is known for its ease of use and its ability to provide feature importance measures.

Regularized regression techniques are variants of linear regression that introduce additional penalties in a model's objective function to reduce overfitting, thus improving the model's generalization. LASSO employs a type of penalty that promotes model simplicity by forcing some coefficients to zero, facilitating the automatic selection of relevant variables and reducing dimensionality [33]. In contrast, RIDGE employs a penalty that reduces the coefficients of less important variables without eliminating them [34], which is useful when all variables are desired in the model but with lesser influence. The advantage of RIDGE over LASSO is that it is differentiable at all points, allowing this regularization technique to be used in neural networks. ENET combines both penalties, offering a balance between LASSO's variable selection and RIDGE's handling of correlated variables [35].

SVR is an adaptation of support vector machines (SVMs) that addresses regression problems. Instead of seeking a hyperplane that separates classes, as in the case of SVMs, the objective of this model is to find a function that fits the data as best as possible, considering a tolerable margin of error. This function, known as the decision function, is determined by identifying the support vectors, which are the data instances closest to the error margin limit. SVR is particularly useful when dealing with noisy or nonlinear datasets, as it allows

the selection of different kernel functions, such as linear, polynomial, or Gaussian, to adapt to the data complexity and improve prediction accuracy.

Models based on gradient boosting are machine learning algorithms that employ an iterative approach to improve prediction accuracy. These models start with a simple regression model, such as a decision tree, and then build new models to predict and minimize the residual errors generated by previous models. Each new model focuses on correcting the errors made by previous models, which allows predictions to be gradually improved as more models are added. In the final model, each of the models contributes to the final prediction, and a weight is assigned to each one according to its relative contribution to improve overall accuracy [36].

Following the initial creation of GB, several significant improvements were developed in machine learning algorithms. A notable example is XGBoost, an enhanced and optimized version by Tianqi Chen. This model employs decision trees as weak classifiers and focuses on minimizing residual errors in each iteration [37]. XGBoost has won numerous competitions in Kaggle and has become an essential tool in various industries due to its ability to improve the model's generalization and avoid overfitting problems, especially in small or noisy training datasets.

On the other hand, LightGBM (LGBM), developed by Microsoft Research, is an algorithm based on gradient boosting decision trees (GBDTs) that is designed to reduce memory usage and accelerate the conventional GBDT training process by more than 20 times while maintaining almost equal accuracy. LightGBM employs a histogram-based algorithm to group continuous features into discrete containers, thus facilitating the evaluation of information gain. Additionally, it uses advanced techniques such as Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to accelerate calculations and improve model accuracy [38].

The multilayer perceptron (MLP) is a type of artificial neural network with a feedforward structure, where information flows from the input layer to the output layer through one or more intermediate hidden layers. Each layer is composed of interconnected neurons with associated weights that are adjusted during training to minimize error. MLPs employ nonlinear activation functions in the neurons of the hidden and output layers, allowing them to model complex relationships between input and output variables and learn nonlinear patterns in the data. In the context of a regression problem, the output layer has a single neuron with a linear activation function, and the network is trained to predict a continuous numeric value.

3.3.2. Deep Tabular Learning

In addition to traditional machine learning models, two deep neural network architectures for tabular data are employed. The aim is to evaluate their performance in an industrial environment and compare them with conventional methods.

TabNet is a deep neural network architecture designed to provide a high-performance and explainable model from tabular data. It accepts raw tabular data without the need for any preprocessing and does not require feature engineering, thanks to its sequential attention mechanism that selects relevant features at each decision step [5].

The TabNet architecture is structured into N sequential steps, where inputs are transmitted from one to another. Each step begins with a transformation through an attention block that uses a sparse matrix to perform feature selection, followed by a feature selection mask as a regulatory mechanism. After applying the attention transformer, the feature importances are added to those of other steps to obtain the final feature importances. Finally, a feature transformer is used before restarting the cycle.

NODE is an architecture that combines the interpretability of decision trees with the learning capability of neural networks. It uses layers of decision trees called ODTs (Oblivious Decision Trees), where each tree contributes independent decisions that are combined to form the model's final output. ODTs are less prone to overfitting and more efficient in inference than traditional trees, as all decisions are made in parallel at each depth

level. NODE uses an alpha-entmax function for feature and branch selection, allowing it to capture complex interactions in tabular data through dispersed distributions controlled by a hyperparameter alpha. This architecture can be implemented as a set of independent trees or in a multi-layer structure, similar to DenseNet in CNNs, enabling the capture of complex dependencies between layers [6].

3.3.3. Performance Evaluation Criteria

Metrics play a fundamental role in the field of machine learning, as they enable us to evaluate and quantify the quality and performance of predictive models. These measures provide us with information on how effectively the model adapts to both training data and previously unseen data [39]. However, choosing an inappropriate metric can result in a flawed model evaluation.

An illustrative example would be the use of the **accuracy** metric in a regression problem. This metric measures the number of correct assessments made by a model in relation to the total number of predictions made, making it a commonly used metric in classification problems. However, in the context of this research, which focuses on solving regression problems, the goal is to predict continuous values rather than assign classes or categories. Therefore, calculating accuracy in this situation would be meaningless, as it is not a binary or multiclass classification task.

To evaluate the effectiveness of predictive models, different metrics are used, including the maximum error, mean absolute error (MAE), root mean square error ($RMSE$), mean absolute percentage error ($MAPE$), adjusted coefficient of determination (R^2 adjusted), and training execution time.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{5}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{6}$$

$$R^2_{adj} = 1 - \frac{(1 - R^2) \times (n - 1)}{n - p - 1} \tag{7}$$

where y_i represents the actual values, $\hat{y}_i$ represents the model predictions, n denotes the number of observations, p denotes the number of predictors, $\bar{y}$ represents the mean of the actual values, and R^2 is the coefficient of determination.

However, the $RMSE$ was selected as the primary decision criterion for the following reasons:

1. Clarity: The $RMSE$, through the squaring of individual errors, eliminates the influence of their sign, providing an absolute measure of error performance. By standardizing errors on a consistent scale, it simplifies their interpretation, enhancing clarity in the assessment process.
2. Sensitivity to Large Errors: In contrast to the MAE, which treats all errors equally, the $RMSE$ assigns higher weight to large errors owing to its quadratic nature. Consequently, the $RMSE$ is more adept at capturing the impact of outliers, thus enhancing the robustness of the evaluation process.
3. Comparability: The $RMSE$ enjoys widespread usage in regression problems, facilitating result comparison across studies and contributing to a more cohesive evaluation framework within the analysis domain. Its ubiquity not only ensures compatibility

with the existing literature but also fosters a more comprehensive understanding of model performance in the broader research context.

3.3.4. Training

The training and initial evaluation of the models were conducted without adjusting any hyperparameters, aiming to use them as baseline models for subsequent comparison with those where hyperparameter tuning is performed. In order to impartially assess the initial performance of these models and avoid overfitting, the *hold-out* technique was applied. This technique involves dividing the dataset into two parts: a training set and a test set. The training set, comprising 80% of the dataset, was utilized for model fitting, while the remaining 20% was reserved for performance evaluation.

Once the baseline models were obtained, the process of finding the model that best fits the data in each relevant factory operation was carried out. To achieve this goal, grid search was initially chosen for exhaustive exploration of the hyperparameters. However, it was found that some models, such as XGBoost or LightGBM, had a considerably wide search space in which to perform an exhaustive search. Faced with this scenario, the decision was made to tackle this challenge more efficiently by combining *random search for hyperparameters* with the *K-fold cross-validation* technique across all analyzed models.

Random search is an optimization technique where hyperparameter values are sampled randomly from a specified range or distribution. Unlike grid search, which exhaustively evaluates all possible combinations within a predefined grid, random search evaluates a fixed number of randomly chosen combinations, which often leads to better performance for large or complex search spaces. This method can be more efficient and effective when dealing with models that have many hyperparameters or where the search space is very large [40].

K-fold cross-validation is a technique that allows for a more reliable evaluation of the model's performance by dividing the dataset into multiple subsets or "folds". In each iteration, one of these subsets is used as the test set, while the remaining subsets are used to train the model. This process is repeated until each subset has been used as the test set once. By combining random search with cross-validation, different combinations of hyperparameters are selected to train and evaluate the model on different data splits. This reduces bias and provides a more accurate estimation of performance on unseen data [41].

It is important to note that each model adjusted the number of iterations in the random search of hyperparameters according to its unique characteristics. This decision is based on the inherent complexity of each algorithm and its sensitivity to various configurations. For example, simple models like linear regression require a smaller search space, while neural networks like MLP may need a more extensive exploration due to their larger number of hyperparameters. For details on the hyperparameters used for each algorithm, please refer to Appendix A.

In each operation within the various areas analyzed in the factories, an approach for model selection based on prominent metrics was implemented, initially prioritizing the RMSE criterion. In situations where there is a tie in terms of the RMSE, other metrics are used as criteria to break the tie. Once the model that yields the best results is identified, it is stored for later use.

To visualize this stage, please refer to Figure 4, and to understand how the random search algorithm works for a generic model, it is recommended to consult the pseudocode provided in Algorithm 1.

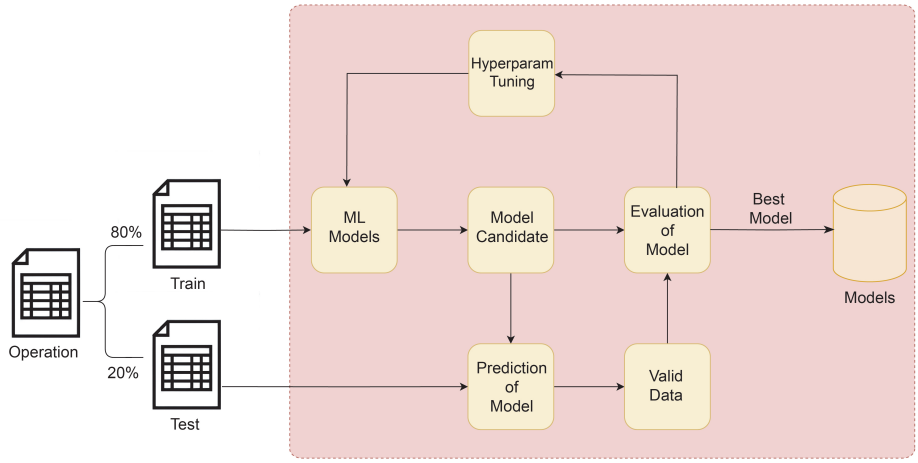

Figure 4. Structure of how ML models are trained.

Algorithm 1 Random search with K-fold cross-validation for regression models. Adapted from [40,41].

Require: Training set $T = \{(x_1, y_1), \ldots, (x_t, y_t)\}$;
The regression model ζ with hyperparameters set $\mathcal{H}$;
Number of iterations N;
Number of folds k for cross-validation.
Ensure: The best set of hyperparameters *best_h* and the best score *best_score*.

```
Initialize best_score ← ∞
Initialize best_h ← {}
for i ← 1 to N do
    Randomly select a set of hyperparameters h_i from H
    Initialize cv_scores ← []
    for each fold f in k-fold cross-validation do
        Split T into training set T_train and validation set T_val for fold f
        Train ζ with hyperparameters h_i on T_train
        fold_score ← Evaluate(ζ, T_val) {Use a regression metric like RMSE}
        Append fold_score to cv_scores
    end for
    score ← Mean(cv_scores)
    if score < best_score then
        best_score ← score
        best_h ← h_i
    end if
end for
return best_h, best_score
```

3.4. Generation of Time Information

Once the models have been evaluated and verified to fulfill their purpose, they can be integrated into a system with the objective of providing significant business value, specifically in calculating the lead times of each operation for scheduling or quotation purposes.

The system operates as follows: when the user submits a request, they must enter a series of inputs, including the type of problem they want to solve (scheduling or quotation), the type of operation to be performed (predictions or retraining), and the relevant dataset, among other inputs. With this information, the system identifies the type of problem to be addressed and determines whether it is necessary to execute a new process or utilize the models that were previously trained and stored during the training and evaluation phase.

It is important to note that although models based on tabular data generally offer good predictions, in systems where training needs to be swift, employing this type of model may not be efficient. This aspect is discussed in more detail in Section 4.

The system performs the corresponding action, whether it is training or prediction, and displays a confirmation message to the user. The structure and functionality of the proposed system are illustrated in Figure 5.

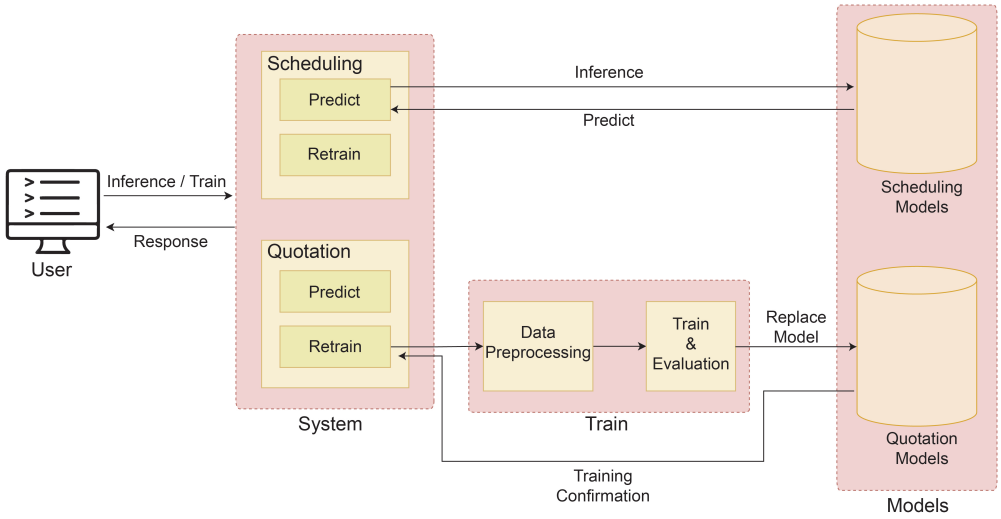

Figure 5. Structure of proposed system.

4. Experimental Results

This section discusses the results obtained using our proposed methodology for estimating lead times. Due to the significant differences between estimating lead times in the two types of problems, we decided to present the results obtained from predicting various operations in the problem of accurate production cost estimation. Each factory was studied independently to ensure the accuracy of our findings.

In addition, the hardware used for the various experiments is described, followed by the presentation of the results in the subsequent tables, which illustrate the outcomes of each process throughout its progression. This is complemented by an overview of the implementation of the developed system, including its prediction and retraining capabilities to address model degradation. The set of hyperparameters that yield optimal performance for the most effective models in each operation are detailed in Appendix C.

4.1. Experimental Environment

In this paper, all the experimental codes are written in Python. The details of the environment on which the experiments depend, including the Python version, libraries, frameworks, hardware specifications, and operating system, are shown in Table 3.

Table 3. Details of experimental setup.

Component	Details
CPU	Intel Core i9-12900K, 16 cores
GPU	NVIDIA GeForce RTX 3080 Ti, 12 GB memory, 10240 CUDA cores
RAM	32 GB
Operative System	Windows 11
Programming Language	Python 3.10
Packages	Numpy [42], Pandas [43], scikit-learn [44], PyTorch Tabular [45], Matplotlib [46], Seaborn [47], PyCaret [48]
IDE	VS Code

4.2. Cutting

The results in Table 4 reveal the performance achieved by each model with the optimal configuration derived through a random hyperparameter search on the sheet-cutting dataset. Notably, in the Spanish factory, except for the MAPE, adjusted R^2, and training time, the NODE model demonstrated superior performance compared to other models. However, random forest (RF) also exhibited similar metrics regarding the RSE, with a significantly better training time. Conversely, in Brazil, the performance of the LightGBM model stood out, surpassing all other models, except for the training time, although it tied with XGBoost (XGB) in the RMSE. Table A6 presents the model that best fits the data from the various factories in the sheet-cutting process.

Table 4. Results obtained for each model with the best configuration through random search on the cutting dataset.

Factory	Model	MAE	RMSE	MAPE (%)	R2 Adjusted	Train Time (s)
Spain	DT	0.4642	0.5854	50.0436	0.0768	1.6846
	RF	0.4551	0.5775	50.04	**0.1016**	5.7529
	RIDGE	0.4787	0.6055	52.8253	0.0122	**0.2167**
	LASSO	0.4778	0.6073	53.1677	0.0064	0.2201
	ENET	0.4777	0.6072	53.1461	0.0068	0.2224
	SVR	0.4771	0.6043	53.2817	0.016	0.9936
	GB	0.4638	0.5879	50.8454	0.0687	3.6476
	XGB	0.4641	0.5881	50.1401	0.068	4.8992
	LGBM	0.4603	0.5827	50.0085	0.0851	7.5646
	MLP	0.4756	0.5968	50.6251	0.0403	7.5191
	TabNet	0.4625	0.5952	**48.0406**	−0.06884	2196.096
	NODE	**0.4526**	**0.5721**	48.1065	0.01174	10,363.06
Brazil	DT	0.1073	0.1331	25.2989	0.172	2.6177
	RF	0.1073	0.1328	25.339	0.1755	16.182
	RIDGE	0.1095	0.1349	26.0212	0.1492	**0.3628**
	LASSO	0.1125	0.1368	26.8496	0.1254	0.3825
	ENET	0.1120	0.1363	26.7176	0.1316	0.441
	SVR	0.1092	0.1346	25.906	0.1532	19.9348
	GB	0.1076	0.1330	25.4048	0.173	14.99
	XGB	0.1073	**0.1325**	25.3456	0.1786	26.977
	LGBM	**0.1072**	**0.1325**	**25.2653**	**0.1789**	51.4082
	MLP	0.1096	0.1347	26.1195	0.1522	16.6133
	TabNet	0.109427	0.136893	25.60443	0.115645	5177.50
	NODE	0.106676	0.132541	25.4344	0.171134	7568.785

4.3. Beveling

Table 5 shows the results of the hyperparameter optimization process conducted via random search. Negative values in the adjusted R-squared column for both factories indicate a poor fit of the model to the data, which could be attributed to data noise or a mismatch between the model and the data structure. In the Spanish factory, the XGB model achieved the best results across most metrics, except for the training time, where the LASSO model was the fastest. On the other hand, in the Brazilian factory, the NODE model stood out for its excellent performance, although its training time was significantly higher compared to other models. The other models achieved slightly less precise results but with much shorter training times. The models that best fit the data, along with their hyperparameters, are shown in Table A7.

Table 5. Results obtained for each model with the best configuration through random search on the *beveling* dataset.

Factory	Model	MAE	RMSE	MAPE (%)	R2 Adjusted	Train Time (s)
Spain	DT	**2.3545**	2.8553	152.8884	0.049	0.2969
	RF	2.3705	2.8501	156.381	0.0525	6.1435
	RIDGE	2.4109	2.868	165.7692	0.0406	0.2645
	LASSO	2.4275	2.8806	168.0643	0.0321	0.271
	ENET	2.4201	2.8751	166.9682	0.0358	**0.2704**
	SVR	2.4081	2.8731	166.0973	0.0371	1.4238
	GB	2.3754	2.8424	159.8668	0.0576	3.4153
	XGB	2.3666	**2.8356**	155.5675	**0.0621**	5.8287
	LGBM	2.3858	2.8606	156.1941	0.0455	9.5314
	MLP	2.4092	2.8655	169.4229	0.0423	29.6221
	TabNet	2.4274	2.9618	136.1569	−0.070	2676.87
	NODE	3.0839	3.7674	**97.5814**	−0.7370	15,866
Brazil	DT	0.0730	0.0947	19.3447	0.1800	2.3477
	RF	0.0729	0.0942	19.3047	0.1878	16.7318
	RIDGE	0.0756	0.0973	19.9906	0.1327	0.4930
	LASSO	0.0798	0.1007	21.1560	0.0724	**0.4050**
	ENET	0.0811	0.1019	21.4878	0.0490	0.4096
	SVR	0.0760	0.0973	20.2003	0.1340	11.2810
	GB	0.0730	0.0944	19.3507	0.1851	14.4489
	XGB	0.0730	0.0943	19.3244	0.1861	28.3986
	LGBM	**0.0727**	**0.0940**	**19.2483**	**0.1908**	28.4344
	MLP	0.0760	0.0957	20.5187	0.1616	22.8199
	TabNet	0.1022	0.1366	26.0893	−0.7776	3083
	NODE	0.0748	0.0963	19.9164	0.1550	6246.319

4.4. Bending

Table 6 summarizes the results of the hyperparameter evaluation. Negative adjusted R-squared values indicate poor model fit due to data noise or mismatch. In the Spanish facility, XGBoost (XGB) performed the best overall, except for the training time. Tabular neural network models also showed competitive performance but required longer training durations. Conversely, at the Brazilian factory, the tabular neural network model NODE demonstrated superior performance in the MAE, RMSE, and MAPE metrics, establishing it as the best model for this task. Table A8 details the model that best fits the data from the factories in the bending process.

Table 6. Results obtained for each model with the best configuration through random search on the bending dataset.

Factory	Model	MAE	RMSE	MAPE (%)	R2 Adjusted	Train Time (s)
Spain	DT	0.2435	0.3114	15.2196	0.0269	0.447
	RF	0.2414	0.3077	15.1508	0.0501	9.4977
	RIDGE	0.2462	0.3155	15.4378	0.0014	0.3818
	LASSO	0.2496	0.3153	15.7818	0.0024	**0.3795**
	ENET	0.2486	0.3145	15.7166	0.0076	0.4239
	SVR	0.2462	0.3152	15.453	0.003	1.3174
	GB	0.2422	0.3076	15.2155	0.0507	7.1917
	XGB	**0.2402**	**0.3073**	**15.003**	**0.0527**	7.8403
	LGBM	0.2445	0.3102	15.3767	0.0345	10.1291
	MLP	0.2448	0.3125	15.4013	0.02	8.0982
	TabNet	0.3409	0.4380	20.2737	−0.71129	2114.821
	NODE	0.2655	0.3364	15.9844	0.030462	26,611.66

Table 6. *Cont.*

Factory	Model	MAE	RMSE	MAPE (%)	R2 Adjusted	Train Time (s)
Brazil	DT	0.217629	0.264736	19.06009	−0.00987	2.686931
	RF	0.21713	0.264244	19.02788	−0.00613	26.32093
	RIDGE	0.216803	0.263774	18.99115	−0.00255	**0.656738**
	LASSO	0.216707	0.263629	18.98983	**−0.00145**	0.65939
	ENET	0.216915	0.263759	19.01269	−0.00244	0.663005
	SVR	0.216688	0.263813	18.9572	−0.00285	64.17738
	GB	0.21708	0.264101	19.02094	−0.00503	23.9229
	XGB	0.217106	0.264052	19.02791	−0.00467	32.71342
	LGBM	0.217182	0.264109	19.03515	−0.0051	50.22291
	MLP	0.218289	0.264646	19.31309	−0.00919	39.64824
	TabNet	0.236308	0.29646	20.11344	−0.26967	4947.336
	NODE	**0.216513**	**0.263311**	**18.78136**	0.001389	22,622.8977

4.5. Longitudinal Welding

In Table 7, the results are presented after performing a random search of the hyperparameters. During dataset testing with these models, issues arose when attempting to train the different neural networks designed for tabular data; hence, they do not appear in the results in this table. In the Seville factory, the LGBM model stood out as it achieved the best metrics, while in the Brazil factory, the XGB model performed the best, demonstrating that in both cases, for this problem, gradient boosting-based models yielded better results. Table A9 presents the best model that fits the data from the different factories in the longitudinal welding process.

Table 7. Results obtained for each model with the best configuration through random search on the longitudinal welding dataset.

Factory	Model	MAE	RMSE	MAPE (%)	R2 Adjusted	Train Time (s)
Spain	DT	0.6155	0.7788	21.1316	0.0783	1.7475
	RF	0.5974	0.7615	20.5429	0.1189	5.8555
	RIDGE	0.6104	0.7701	21.0718	0.0988	**0.2285**
	LASSO	0.6147	0.7697	21.2637	0.0998	0.2319
	ENET	0.6133	0.7699	21.2001	0.0992	0.2367
	SVR	0.6108	0.7732	20.7838	0.0916	1.3211
	GB	0.5960	0.7693	20.5543	0.1006	3.6245
	XGB	0.5942	0.7690	20.5006	0.1014	4.6344
	LGBM	**0.5910**	**0.7608**	**20.3361**	**0.1206**	8.8708
	MLP	0.6096	0.7690	20.9206	0.1015	15.1607
Brazil	DT	0.25589	0.3221	17.30496	0.17474	2.28899
	RF	0.25561	0.32157	17.29898	0.17745	17.53911
	RIDGE	0.25939	0.32513	17.53897	0.15916	**0.33473**
	LASSO	0.27268	0.33667	18.42142	0.0984	0.33708
	ENET	0.28008	0.3445	18.8961	0.05597	0.33469
	SVR	0.25864	0.32493	17.43438	0.16017	48.73003
	GB	0.25587	0.32151	17.31692	0.17775	16.14765
	XGB	**0.25498**	**0.32119**	**17.23131**	**0.17942**	19.91989
	LGBM	0.25576	0.32192	17.29746	0.17564	64.05158
	MLP	0.25968	0.32358	17.9417	0.16714	46.47257

4.6. Rebending

Table 8 shows model evaluations post-hyperparameter tuning via random search. The negative adjusted R2 values suggest poor data fit due to noise or model–data mismatch. In Seville, gradient boosting notably enhanced the MAE, RMSE, and MAPE metrics. Neural networks for tabular data exhibited longer training times but performed competitively. Conversely, in Brazil, NODE excelled in the MAE, RMSE, and MAPE metrics despite

extended training periods. TabNet, however, yielded an MAPE over 100%, indicating substantial prediction discrepancies. Table A10 lists the optimal factory-specific rebending process models.

Table 8. Results of models determined through random search on the rebending dataset.

Factory	Model	MAE	RMSE	MAPE (%)	R2 Adjusted	Train Time (s)
Spain	DT	0.3182	0.3910	41.9814	0.0049	1.7262
	RF	0.3146	0.3865	**41.4139**	0.0281	5.4414
	RIDGE	0.3241	0.3936	44.1764	−0.0081	0.2363
	LASSO	0.3274	0.3957	44.8102	−0.0189	**0.2361**
	ENET	0.3240	0.3932	44.2470	−0.0062	0.2442
	SVR	0.3236	0.3922	44.1203	−0.0011	0.6704
	GB	0.3150	0.3854	41.9179	0.0335	3.2214
	XGB	**0.3139**	0.3857	41.4309	0.0319	3.8591
	LGBM	0.3147	**0.3840**	41.7782	**0.0404**	7.0541
	MLP	0.3212	0.3927	43.0614	−0.0036	4.4895
	TabNet	0.3342	0.4172	46.1915	−0.1439	1917.4800
	NODE	0.3225	0.3851	46.0042	0.0353	8570.6000
Brazil	DT	0.1580	0.1904	42.5823	−0.0314	2.0558
	RF	0.1592	0.1911	42.6567	−0.0384	5.8700
	RIDGE	0.1587	0.1893	43.0207	−0.0191	0.2561
	LASSO	0.1586	0.1887	43.1433	−0.0132	**0.2559**
	ENET	0.1586	0.1891	43.0440	−0.0172	0.2625
	SVR	0.1591	0.1890	43.6599	−0.0160	0.3911
	GB	0.1582	0.1890	42.7117	−0.0162	2.6744
	XGB	0.1587	0.1887	43.2165	−0.0134	2.6293
	LGBM	0.1586	0.1887	43.1433	**−0.0132**	5.3262
	MLP	0.1582	0.1898	42.1096	−0.0250	2.7716
	TabNet	0.4619	0.5189	124.9740	−10.7883	1588.57
	NODE	**0.1484**	**0.1795**	**40.2075**	−0.0427	7738.602

4.7. Fit-Up

After evaluating models on the fit-up dataset, the Spanish factory showed notable performance differences (See Table 9). XGBoost achieved the lowest MAE, while SVR excelled with the best RMSE and adjusted R^2. Despite TabNet's superior MAPE, its overall effectiveness was hindered by prolonged training times. Conversely, in the Brazilian factory, challenges arose with neural networks due to data issues. RF exhibited the lowest MAE, SVR had the lowest MAPE, and XGBoost demonstrated the lowest RMSE, making it the preferred model. The lower adjusted R^2 in Spain suggests potentially higher data quality in Brazil, influencing model performance and predictive accuracy.

Table 9. Results of models determined through random search on the fit-up dataset.

Factory	Model	MAE	RMSE	MAPE (%)	R2 Adjusted	Train Time (s)
Spain	DT	1.2011	1.5541	39.5827	0.1080	2.1730
	RF	1.1936	1.5260	39.4140	0.1400	3.9951
	RIDGE	1.2155	1.5197	40.6624	0.1470	**0.2224**
	LASSO	1.2279	1.5250	41.0332	0.1410	0.2230
	ENET	1.2316	1.5259	41.1330	0.1400	0.2342
	SVR	1.1901	**1.5152**	37.9503	**0.1521**	0.9706
	GB	1.1929	1.5250	39.8664	0.1411	2.5775
	XGB	**1.1803**	1.5183	39.2723	0.1486	3.7558
	LGBM	1.1999	1.5368	39.8310	0.1278	7.9077
	MLP	1.2101	1.5210	40.2099	0.1456	17.1328
	TabNet	1.2550	1.6374	**36.8195**	−0.0185	2698.59
	NODE	1.4181	1.8409	37.2561	−0.3163	78,024.08

Table 9. *Cont.*

Factory	Model	MAE	RMSE	MAPE (%)	R2 Adjusted	Train Time (s)
Brazil	DT	2.3676	3.4031	21.5839	0.8570	0.4868
	RF	**2.3601**	3.4145	21.2198	0.8561	9.4197
	RIDGE	2.4686	3.4660	23.5846	0.8517	**0.4311**
	LASSO	2.4657	3.4634	23.3255	0.8519	0.4376
	ENET	2.4962	3.4678	24.2983	0.8515	0.4406
	SVR	2.4344	3.4686	**21.1076**	0.8515	3.4559
	GB	2.3619	3.4037	21.4262	0.8570	5.1694
	XGB	2.3654	**3.4006**	21.4145	**0.8572**	7.3314
	LGBM	2.3831	3.4333	21.8825	0.8545	23.5088
	MLP	2.4299	3.4292	22.7512	0.8548	50.9705

4.8. Circular Welding

Table 10 presents the results obtained after adjusting the hyperparameters through a random search. Unlike other processes segmented into sections, this case deals with an entire tower section, resulting in significantly higher errors due to longer execution times. It is important to note that these datasets contain a limited number of examples, which complicates the training of tabular neural network models. Therefore, in this study, only traditional metrics have been used.

Table 10. Results obtained for each model with the best configuration through random search on the circular welding dataset.

Factory	Model	MAE	RMSE	MAPE (%)	R2 Adjusted	Train Time (s)
Spain	DT	8.5956	12.5213	**13.0593**	0.6221	1.6795
	RF	9.1742	12.7837	14.2495	0.6061	2.5769
	RIDGE	15.9338	18.5804	24.7620	0.1679	**0.2118**
	LASSO	16.0120	18.6271	24.7898	0.1637	0.2128
	ENET	15.9469	18.6081	24.7673	0.1654	0.2147
	SVR	16.1549	19.2444	25.4084	0.1074	0.2266
	GB	9.1216	12.8523	14.1254	0.6019	0.9746
	XGB	8.9914	12.7037	13.7033	0.6110	1.1620
	LGBM	**8.4832**	**12.4146**	13.2389	**0.6285**	1.0141
	MLP	11.5148	14.8114	17.2949	0.4713	4.6914
Brazil	DT	3.7852	4.8947	10.9932	0.5548	2.1393
	RF	**3.7731**	**4.8448**	**10.9575**	**0.5638**	3.6586
	RIDGE	4.3557	5.5093	12.5142	0.4360	0.2700
	LASSO	4.3671	5.5155	12.5436	0.4347	**0.2636**
	ENET	4.4576	5.5584	12.7941	0.4258	**0.2636**
	SVR	4.1971	5.5583	11.7384	0.4259	0.6297
	GB	3.8114	4.8595	11.0604	0.5612	1.5213
	XGB	3.7851	4.8888	10.9821	0.5558	2.2018
	LGBM	3.8037	4.8549	11.0232	0.5620	4.7461
	MLP	4.1083	5.2086	11.7463	0.4958	21.1404

In the Seville factory, the LGBM model stood out in terms of the MAE, RMSE, and adjusted R^2 metrics. The DT model performed well in the MAPE, while the RIDGE model had a reduced training time. Thus, the LGBM model emerged as the preferred option for the Seville factory. In contrast, in the Brazilian factory, the RF model excelled in most metrics, including the RMSE, making it the preferred option in this case. Table A12 presents the best model that fits the data from the different factories in the circular welding process.

4.9. Implementation of the System

In this subsection, we discuss the implementation of a system designed to perform predictions and retrain models in response to degradation. This discussion is framed within

the context of accurate production cost estimation. Additionally, the procedures for task scheduling and resource allocation are addressed, as they follow a similar methodology.

4.9.1. Prediction

Once the system is initialized, the user accesses the type of operation they wish to solve, opening an interface similar to that shown in Figure 6. The process for performing predictions is as follows: the user selects the factory for which they wish to calculate lead times. Internally, the system identifies the necessary models for making predictions. Subsequently, the user adds rows corresponding to the components, along with the requisite attribute values for prediction. The columns may vary based on the factory and the nature of the problem. Upon clicking the `Calculate` button, the system executes the predictions and outputs a tabular file, as illustrated in Table 11.

Table 11. Results of lead-time estimation (example).

Operation	Can 1	Can 2	Can 3	Can 4	Can 5	Can 6	Can 7	Can 8	Can 9	Section
Cutting	1.60	1.62	1.37	1.37	1.45	1.42	1.60	1.62	1.37	13.42
Beveling	6.19	5.63	5.56	9.04	4.91	5.10	6.19	5.63	5.56	53.81
Bending	1.94	1.82	1.73	1.58	1.65	1.63	1.94	1.82	1.80	15.90
Longitudinal Welding	3.89	3.09	3.09	2.98	2.75	2.57	3.89	3.25	3.09	28.61
Rebending	0.88	0.83	0.85	0.66	0.71	0.78	0.88	0.83	0.79	7.20
Fit-up	3.78	3.82	3.78	2.60	2.90	2.96	3.78	3.82	3.78	31.21
Circ. Welding										103.45

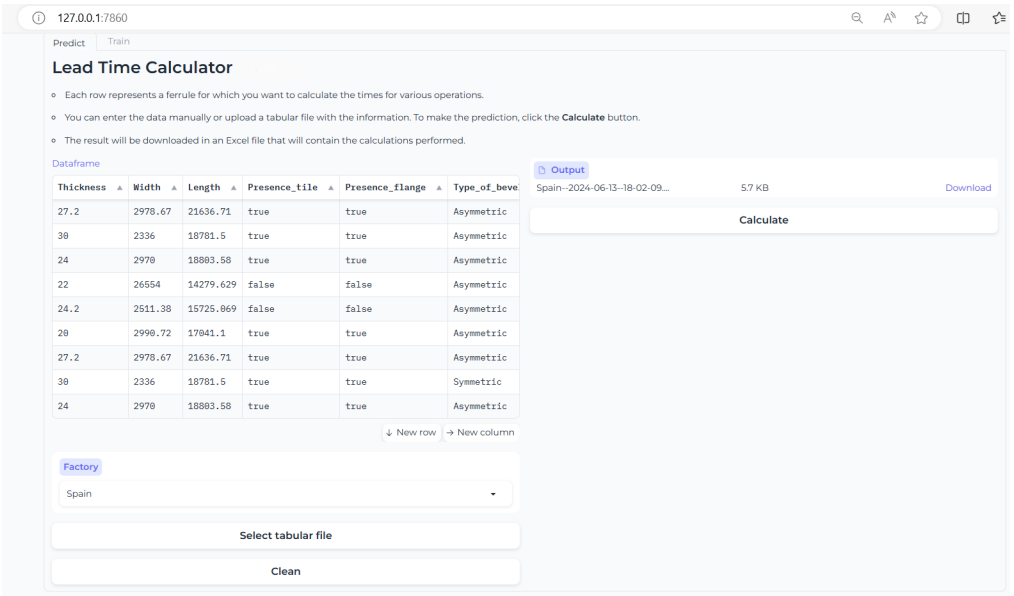

Figure 6. Example of lead-time prediction.

4.9.2. Retraining

Once we understand how to perform predictions, we explore how to train new models adapted as they degrade over time. Figure 7 illustrates the system layout for this functionality. Unlike the previous tab, all components are inputs because the new model is saved internally in the database, replacing the previous model. The process for retraining unfolds as follows: first, you must select the operation and factory for which you wish to modify the model.

In contrast to the previous setup, all these elements are inputs, as the output is stored directly on the server. This setup includes the specific operation and factory for which we aim to train the new model, alongside a read-only dataframe showcasing the essential columns for training a specific operation at the selected factory. Generating a model that predicts all datasets would increase problem complexity. Once these parameters are chosen, the dataframe updates with the necessary columns for model training, and it is time to load the data. Once confirmed, the training process can be initiated by pressing the `Train` button.

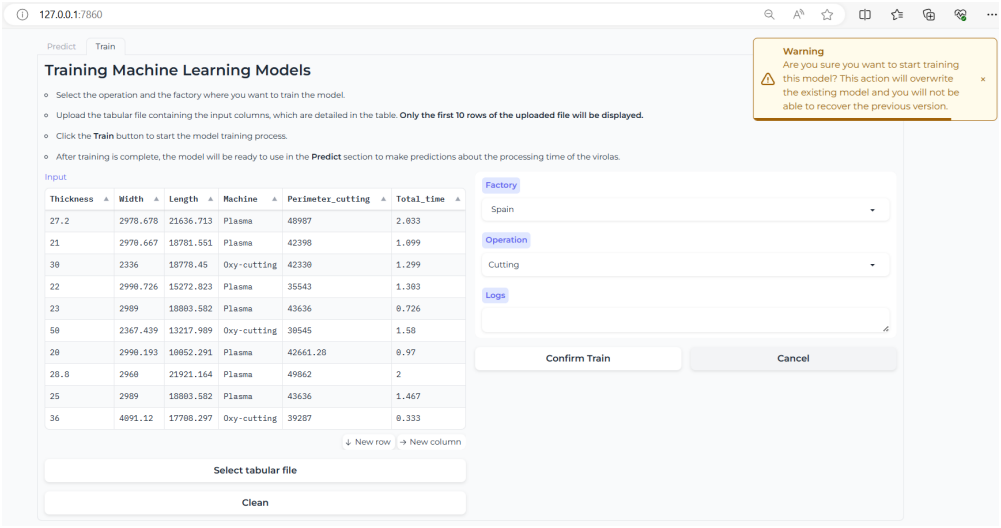

Figure 7. Visualization of the training notification.

Attempting to train a model with an empty dataframe and confirming the operation generates an error, alerting the user to the need to add data for the new model. In the absence of errors, the training process commences, which is reflected in the event logs, where models are trained and adjusted automatically without user intervention. If the process proceeds smoothly, the model is replaced and ready to make predictions.

5. Discussion

The results of this study underscore the performance characteristics of various models, including traditional machine learning models and tabular neural network models, across numerous industrial operations in two distinct factories. The models are evaluated based on several performance metrics, such as mean absolute error (MAE), root mean square error (RMSE), mean absolute percentage error (MAPE), adjusted R^2, and model training time, which are visually represented as radial charts in Appendix B to facilitate comparison.

The results summarized in Table 12 indicate that there is no single model that is optimal for all situations within the set of operations evaluated in each factory. Instead, a limited variety of models show good performance in different contexts. In Spain, models based on gradient boosting, such as XGBoost and LightGBM, stand out for their effectiveness in solving the problem, followed by SVR. However, in Brazil, where some attributes differ, boosting-based models continue to perform well, although NODE achieves better results in some operations. These findings highlight the importance of selecting the most appropriate model for a given dataset based on its characteristics and the specific problem being addressed.

Table 12. Performance assessment of the models that offer the best fit to the data across the diverse relevant operations of the two analyzed factories.

	Spain		**Brazil**	
Operation	**Best Model**	**RMSE (h)**	**Best Model**	**RMSE (h)**
Cutting	NODE	0.5721	LGBM	0.1325
Beveling	SVR	2.8263	LGBM	0.094
Bending	XGB	0.3073	NODE	0.2633
Long. welding	LGBM	0.7608	XGB	0.3211
Rebending	LGBM	0.384	NODE	0.1795
Fit-up	SVR	1.516	XGB	3.40
Circ. welding	LGBM	12.41	RF	4.8448

Upon analyzing the RMSE values, it was observed that several operations, such as bending and recoiling, exhibited low RMSE values, indicating a trend toward accuracy in predictions for these specific datasets. On the other hand, the circular welding operation displayed a significantly higher RMSE value compared to the other operations. This disparity could be related to both the dataset size and the long execution times associated with this operation. Therefore, there is a need to investigate and improve the effectiveness of models in this particular operation.

The results in Section 4 indicate that while neural network models for tabular data can achieve performance comparable to traditional models, they require significantly longer training times. A notable example is the NODE model, which has proven to be the most effective in several tasks but also exhibits the longest training times. Additionally, occasional issues have been reported during its execution. These issues may be attributed to an inefficient memory implementation, which demands a considerable amount of GPU memory to converge [6].

Despite potential advancements, traditional machine learning models are expected to maintain a significantly faster training speed compared to deep learning models for tabular data. Various studies support this claim, highlighting the effectiveness of tree-based models such as XGBoost (XGB) and random forest (RF) on medium-sized tabular datasets (approximately 10,000 samples). These studies have found that XGB consistently outperforms deep learning models like NODE and TabNet in terms of accuracy, training speed, and hyperparameter tuning, requiring substantially less adjustment and computational effort [49,50].

Therefore, for industrial applications, it may be more advantageous to employ traditional machine learning techniques, particularly gradient boosting algorithms such as XGB or LGBM. These models not only offer fast training speeds but also deliver strong performance, making them a preferred choice in many practical scenarios. The combination of their efficiency and effectiveness makes them valuable tools for handling tabular data in industrial environments.

In addition to considering model performance in terms of training speed, it is also crucial to select models that are explainable. Explainability in machine learning refers to the ability to understand and communicate how a model makes specific decisions or generates predictions, which is essential for ensuring transparency and accountability in the use of these models. For instance, in the context of this study, identifying the most relevant features that may influence a particular operation allows us to better understand the model's decision-making process and enhance confidence in its predictions.

Machine learning models are commonly divided into two categories: white-box models and black-box models. Black-box models are those that are difficult to interpret due to their mathematical complexity, such as hyperplane-based models (SVM), models based on biological neural networks, probabilistic and combinatorial logic models, or instance-based models (such as k-nearest neighbors). In contrast, white-box or explainable models, such as decision trees and rule-based systems, are designed to provide clear explanations of how predictions are generated, offering a balance between accuracy and understanding without the need for additional models for interpretation [51].

In this research, the model with the best overall performance, which in this case was LGBM, was chosen. This model provides complete transparency regarding how predictions are made, allowing for a greater understanding and explainability of the obtained results. To demonstrate the explainability of this model, the importance of various features in each model and for each operation was analyzed, excluding the rebending process due to its low frequency in the factory. The metric used for this analysis was gain, which measures the improvement in impurity reduction (such as mean squared error) resulting from including a feature in the model. A higher gain indicates that the feature is more important to the model, as it contributes more significantly to the accuracy of the predictions. Therefore, the results and analyses presented in this study focus on the most significant and frequent processes in the factory.

The obtained results are visualized in Figures 8 and 9. In the Spanish industrial facility, it is evident that the inherent characteristics of the sheet metal are the most relevant, particularly the thickness and length of the sheet. It can be observed that the types of beveling or machine specifications generally have minimal or no importance. However, in the fit-up process, the presence of a flange shows significant relevance. On the other hand, in the Brazilian factory, it is observed that in most processes, the attribute that provides the greatest gain is the thickness of the sheet metal, followed by other related attributes. Similar to the Spanish factory, the types of beveling and other characteristics have very low or no importance.

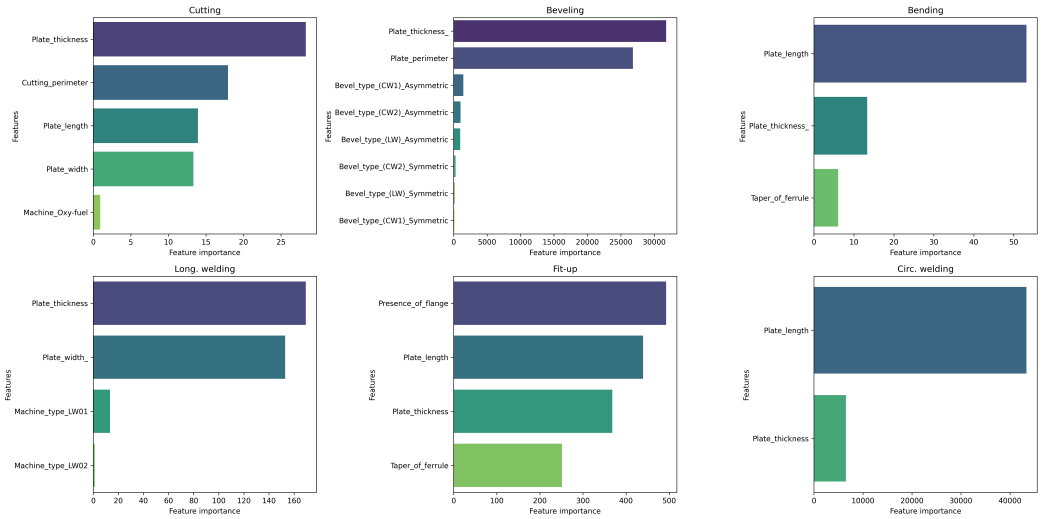

Figure 8. Feature importances across various operations at the factory in Spain.

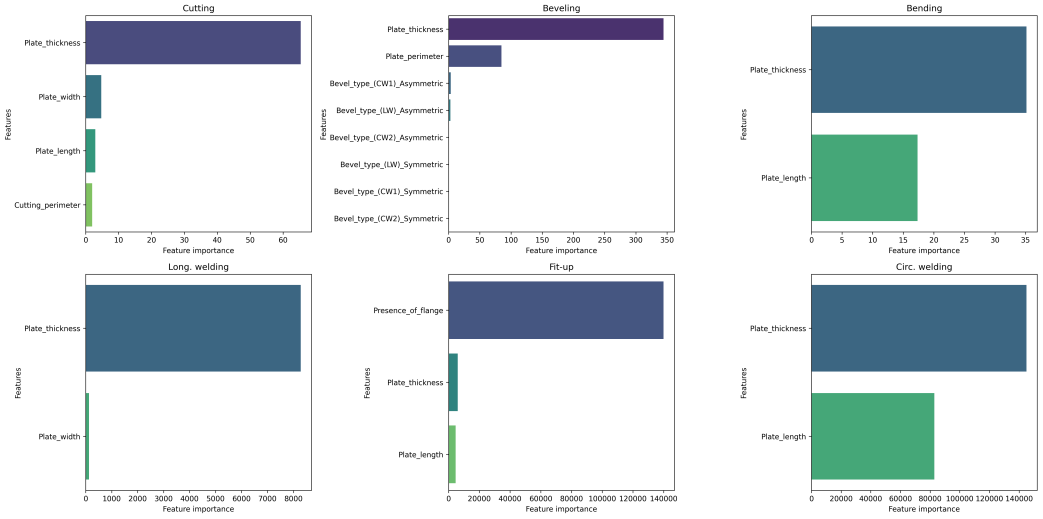

Figure 9. Feature importances across various operations at the factory in Brazil.

6. Conclusions and Future Work

In this study, several machine learning and deep learning algorithms have been examined and evaluated with the purpose of estimating processing times in different operations of factories specialized in the construction of wind turbine towers. The results obtained indicate that the use of machine learning models, capable of analyzing large volumes of data and identifying underlying patterns among relevant variables, significantly improves the accuracy in predicting the total operation time.

Among the various algorithms evaluated, models based on gradient boosting, such as XGBoost and LightGBM, have demonstrated superior performance in addressing the

problems outlined in this study. However, positive results were also obtained with models such as SVR and NODE. Regarding neural network algorithms for tabular data, such as NODE and TabNet, it is important to note that the training times of these models are significantly higher compared to more traditional machine learning techniques. Therefore, in production environments, it may be more beneficial to employ traditional algorithms that offer similar results but in much shorter timeframes.

In summary, this work makes significant contributions in the following aspects: It introduces an innovative method to estimate lead times in the wind tower manufacturing industry, optimizing process scheduling and resource allocation, as well as accurate production cost estimation. This work tests neural network models for tabular data in an industrial context and designs a system capable of applying these models in practice to estimate lead times based on the specific problem and factory. Additionally, the system includes the capability to retrain the models as they degrade over time. This opens the door to substantial improvements in operational decision making within this industrial field.

As future lines of research, it is proposed to extend the analysis carried out in this research to other factory operations. The aim is to obtain a more detailed understanding of the various processes that take place in order to develop specialized predictive models for each of them. This expansion will not only allow the identification of areas for improvement in different processes but will also facilitate the implementation of more effective and tailored solutions for each specific operation.

Additionally, the integration of Explainable Artificial Intelligence (XAI) techniques, such as LIME (Local Interpretable Model-agnostic Explanations) [52] and SHAP (SHapley Additive exPlanations) [53], will be studied to improve the transparency and interpretability of predictive black-box models. XAI will provide detailed information on the contribution of various factors to the predictions, allowing stakeholders to better understand the model's decision-making process. This transparency is essential for building trust and facilitating the adoption of AI-based solutions in industrial environments, leading to more informed and safer decision making in manufacturing processes.

As a prospective line of research, it is suggested to refine the system by incorporating functionalities to automatically exclude models showing poor performance in various operations for future retraining phases. In addition, the gradual integration of new machine learning models could provide a wider and more optimal range of model options for the system.

Author Contributions: Conceptualization, A.E.-S.; methodology, A.E.-S. and K.-J.F.-H.; software, K.-J.F.-H.; validation, A.E.-S., K.-J.F.-H., M.-L.M.-D. and P.C.; formal analysis, A.E.-S., K.-J.F.-H., M.-L.M.-D. and P.C.; investigation, A.E.-S., K.-J.F.-H., M.-L.M.-D. and P.C.; resources, A.E.-S.; data curation, K.-J.F.-H. and M.-L.M.-D.; writing—original draft preparation, K.-J.F.-H.; writing—review and editing, A.E.-S. and K.-J.F.-H.; visualization, K.-J.F.-H.; supervision, A.E.-S. and P.C.; project administration, A.E.-S. and M.-L.M.-D.; funding acquisition, A.E.-S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was co-funded by the Science and Innovation Ministry through the project TRACOMERD (TED2021-130198A-I00) and by the Ministry for Digital Transformation and Public Service, Agencia Red.es through the project INARI (2021/C005/00144319). The programs of these projects are cofounded by the Next-Generation Program and Recovery Plan Transformation and Resilience.

Data Availability Statement: Restrictions apply to the availability of these data. Data was obtained from third party and are available from the authors with the permission of third party.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
DTs	Decision Trees
RF	Random Forest
RIDGE	Ridge Regression
LASSO	Lasso Regression
ENET	Elastic Net
SVR	Support Vector Regression
GB	Gradient Boosting
XGB	eXtreme Gradient Boosting
LGBM	Light Gradient Boosting Machine
MLP	Multilayer Perceptron
NODE	Neural Oblivious Decision Ensembles
MSE	Mean Square Error
MAE	Mean Absolute error
RMSE	Root Mean Square Error
MAPE	Mean Absolute Percentage Error
R^2	Coefficient of determination

Appendix A. Hyperparameters for Machine Learning Models

The tables in this section showcase the hyperparameter configurations used in the machine learning and deep learning models during the training stage. Specifically, Table A1 presents the hyperparameters for decision trees and random forests, while Table A2 covers those for regularized linear regression models. Tables A3–A5 detail the hyperparameters for support vector machines, boosting models, and neural networks, respectively.

Table A1. Parameter tuning ranges and descriptions for decision tree (DT) and random forest (RF) algorithms.

Model	Parameters	Tuning Range	Description
DT	max_depth	[5,30]	Maximum depth of each tree
	min_samples_split	[2,20]	Minimum number of samples required to split a node
	min_samples_leaf	[1,20]	Minimum number of samples required to be at a leaf node
RF	n_estimators	[50,200]	Number of trees in the forest
	max_depth	[5,30]	Maximum depth of each tree
	min_samples_split	[2,20]	Minimum number of samples required to split a node
	min_samples_leaf	[1,20]	Minimum number of samples required at each leaf

Table A2. Parameter tuning ranges and descriptions for regularized linear regression models.

Parameter	Tuning Range	Description
alpha	[0.001,0.99]	Regularization parameter

Table A3. Parameter tuning ranges and descriptions for support vector regression (SVR).

Parameter	Tuning Range	Description
kernel	linear, rbf, poly	Type of kernel
C	[1.5,10]	Regularization parameter
gamma	$[1 \times 10^{-6}, 1 \times 10^{-2}]$	Kernel coefficient
epsilon	[0.01,0.5]	Error tolerance

Table A4. Parameter tuning ranges and descriptions for the gradient boosting (GB), extreme gradient boosting (XGB), and light gradient boosting machine (LGBM) algorithms.

Model	Parameter	Tuning Range	Description
GB	n_estimators	[50,200]	Number of estimators
	learning_rate	[0.01,0.1]	Learning rate
	max_depth	[3,10]	Maximum depth of trees
	min_samples_split	[2,10]	Minimum number of samples required to split a node
	min_samples_leaf	[1,5]	Minimum number of samples required to be at each leaf node
XGB	alpha	[0.001,0.99]	Regularization parameter
	max_depth	[3,11]	Maximum depth of trees
	learning_rate	[0.001,0.1]	Learning rate
	n_estimators	[100,500]	Number of trees in the forest
	gamma	[0,0.2]	Loss reduction threshold to split a node
	subsample	[0.6,1]	Proportion of samples used to train each tree
	colsample_bytree	[0.6,1]	Proportion of features used to train each tree
LGBM	boosting	gbdt, dart, goss	Types of boosting algorithms
	num_leaves	[20,60]	Number of leaves
	max_depth	-1,5,10,15,20	Maximum depth of trees
	learning_rate	[0.001,0.5]	Learning rate
	n_estimators	[100,2000]	Number of estimators
	subsample	[0.2,0.8]	Proportion of samples used to train each tree
	colsample_bytree	[0.4,0.6]	Proportion of features used to train each tree
	alpha	0, 1×10^{-1}, 1, 2, 5, 7, 10, 50, 100	L1 regularization coefficient
	lambda	0, 1×10^{-1}, 1, 2, 5, 7, 10, 50, 100	L2 regularization coefficient

Table A5. Parameter tuning ranges and descriptions for the MLP, TabNet, and NODE algorithms.

Model	Parameter	Tuning Range	Description
MLP	hidden_layer_sizes	(50,), (100,), (50, 50), (100, 50)	Hidden layer sizes
	activation	relu, tanh	Activation function
	solver	ADAM, SDG	Algorithm for weight optimization
	learning_rate	constant, adaptive	Learning rate used by the solver
	max_iter	[100,1000]	Maximum number of iterations
TabNet	n_d	[4,64]	Dimension of prediction layer
	n_a	[4,64]	Dimension of attention layer
	n_steps	[3,10]	Number of successive steps
	gamma	1.0,1.2,1.5,2.0	Discount factor
	mask_type	sparsemax, entmax	
NODE	num_layers	[1,10]	Number of layers
	num_trees	256,512,1024,2048	Number of Oblivious Decision Trees in each layer
	depth	[4,9]	Tree depth
	tree_output_dim	[1,5]	Output tree dimensionality

Appendix B. Comparative Analysis of Performance Metrics across Factory Operations

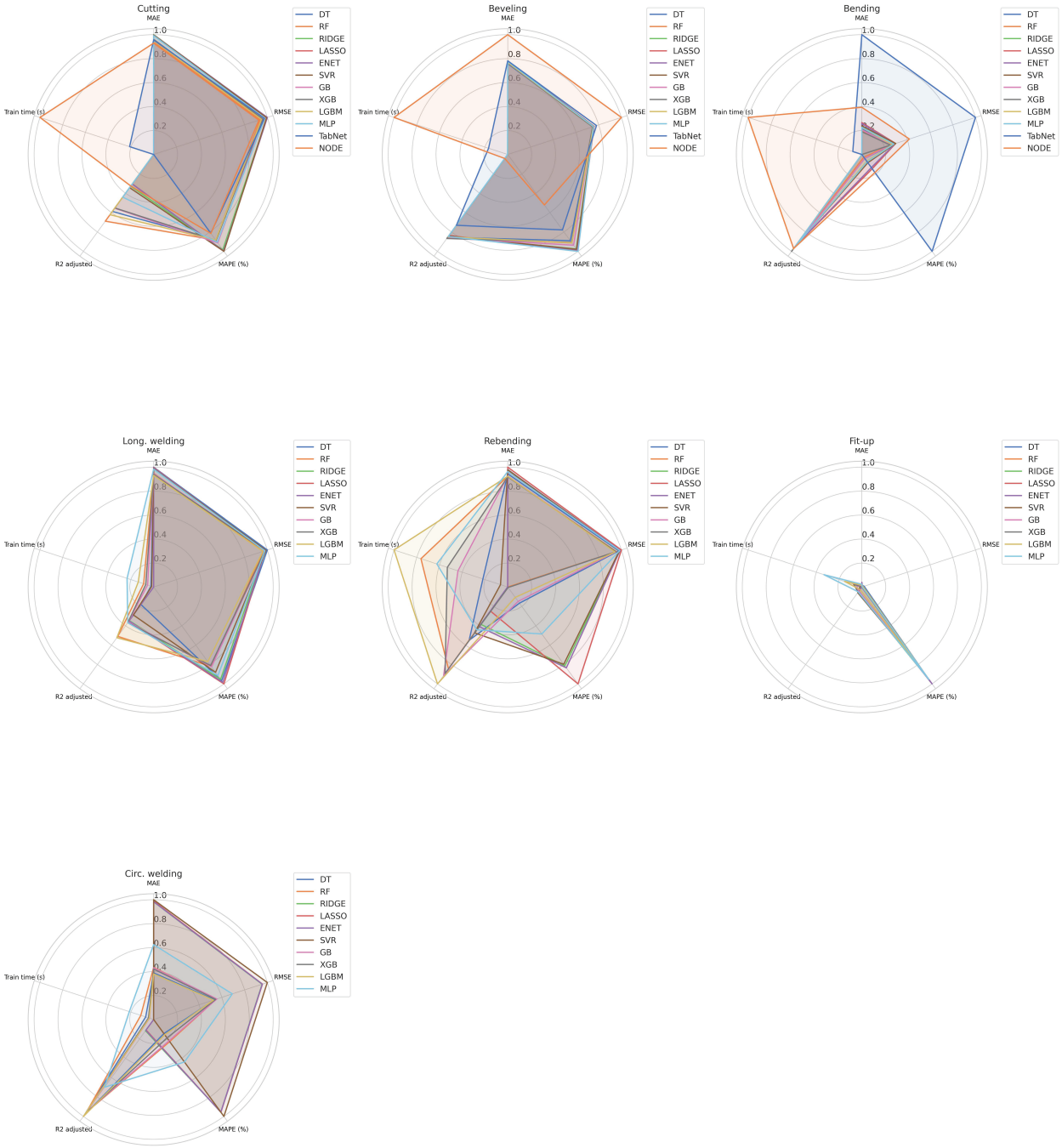

Figure A1. Comparative analysis of performance metrics across factory operations in Spain.

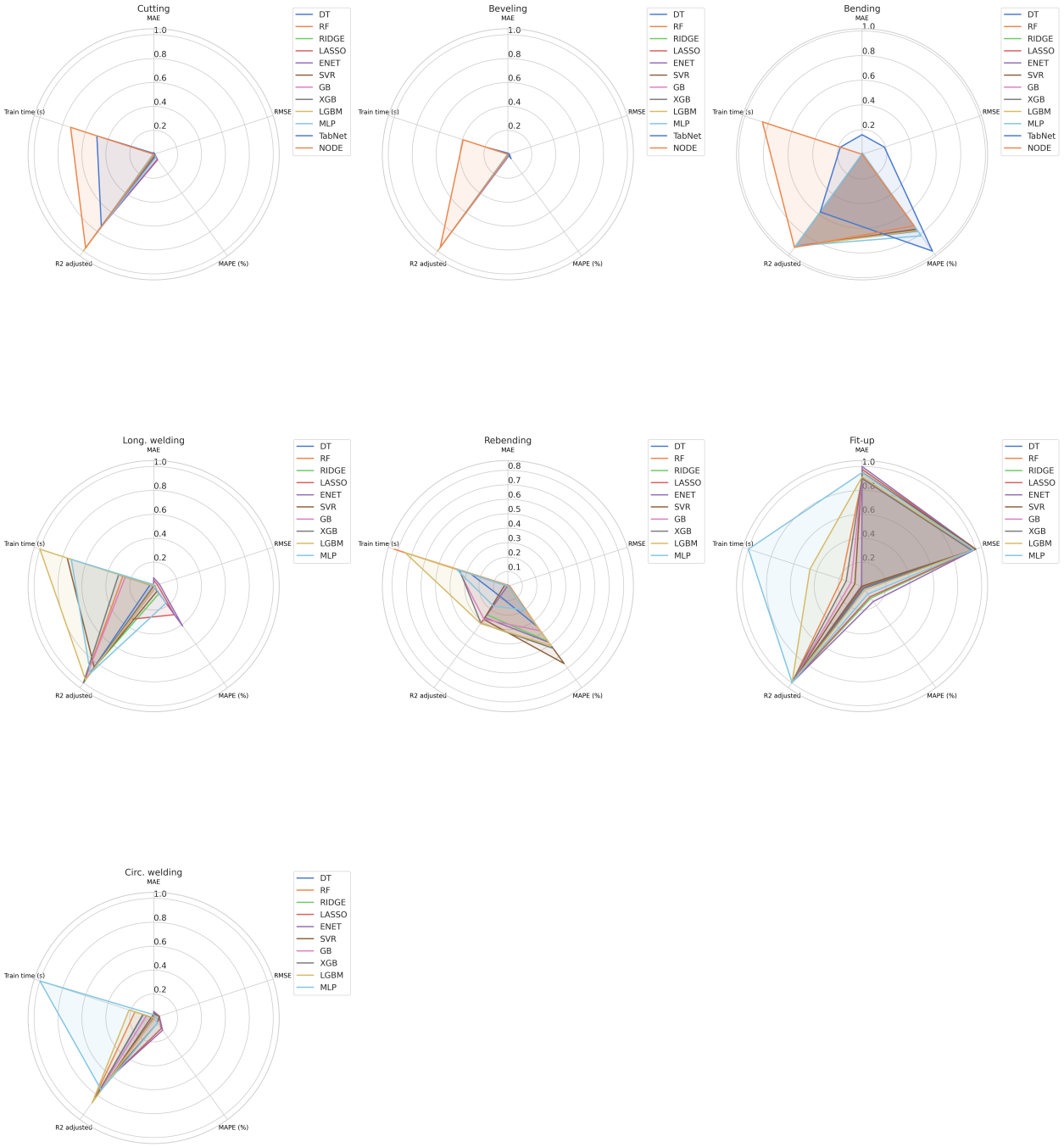

Figure A2. Comparative analysis of performance metrics across factory operations in Brazil.

Appendix C. Summary of Best Model Parameters by Operation

Table A6. Best model parameters for each factory in the Cutting dataset.

Factory	Best Model	Parameter	Parameter Settings
Spain	NODE	num_layer	2
		num_trees	2048
		depth	6
Brazil	LGBM	boosting_type	gbdt
		colsample_bytree	0.991
		learning_rate	0.239
		max_depth	7
		n_estimators	346
		num_leaves	35
		reg_alpha	2
		reg_lambda	50
		subsample	0.382

Table A7. Best model parameters for each factory in the Beveling dataset.

Factory	Best Model	Parameter	Parameter Settings
Spain	XGB	alpha	0.1682
		colsample_bytree	0.6
		gamma	0.1178
		learning_rate	0.0232
		max_depth	3
		n_estimators	198
		subsample	0.8
Brazil	LGBM	boosting_type	gbdt
		colsample_bytree	0.7234
		learning_rate	0.0198
		max_depth	6
		n_estimators	1708
		num_leaves	27
		reg_alpha	1
		reg_lambda	10
		subsample	0.2174

Table A8. Best model parameters for each factory in the Bending dataset.

Factory	Best Model	Parameter	Parameter Settings
Spain	XGBoost	alpha	0.30
		colsample_bytree	0.8
		gamma	0.1936
		learning_rate	0.0227
		max_depth	3
		n_estimators	199
		subsample	0.9
Brazil	NODE	num_layers	6
		num_trees	1024
		depth	6
		additional_tree_output_dim	3

Table A9. Best model parameters for each factory in the Longitudinal Welding dataset.

Factory	Best Model	Parameter	Parameter Settings
Spain	LightGBM	boosting_type	'gbdt'
		colsample_bytree	0.7473
		learning_rate	0.2932
		max_depth	8
		n_estimators	990
		num_leaves	51
		reg_alpha	5
		reg_lambda	0.1
		subsample	0.4015
Brazil	XGBoost	alpha	0.48
		colsample_bytree	1.0
		gamma	0.19
		learning_rate	0.018
		max_depth	3
		n_estimators	360
		subsample	0.7

Table A10. Best model parameters for each factory in the Rebending dataset.

Factory	Best Model	Parameter	Parameter Settings
Spain	LightGBM	boosting_type	gbdt
		learning_rate	0.2819
		max_depth	9
		n_estimators	141
		n_leaves	32
		reg_alpha	5
		reg_lambda	100
		subsample	0.5111
Brazil	NODE	num_layers	8
		num_trees	512
		depth	6
		additional_tree_output_dim	3

Table A11. Best model parameters for each factory in the Fit-up dataset.

Factory	Best Model	Parameter	Parameter Settings
Sevilla	SVR	C	6.74455
		Epsilon	0.496958
		Gamma	0.0091147
		Kernel	rbf
Brazil	XGB	alpha	0.34
		colsample_bytree	1.0
		gamma	0.13
		learning_rate	0.018
		max_depth	3
		n_estimators	333
		subsample	0.6

Table A12. Best model parameters for each factory in the Circular Welding dataset.

Factory	Best Model	Parameter	Parameter Settings
Spain	LGBM	boosting_type	gbdt
		colsample_bytree	0.5185
		learning_rate	0.2899
		max_depth	8
		n_estimators	500
		num_leaves	57
		reg_alpha	0
		reg_lambda	0
		subsample	0.6304
Brazil	RF	max_depth	15
		min_samples_leaf	4
		min_samples_split	5
		n_estimators	88

References

1. Rai, R.; Tiwari, M.K.; Ivanov, D.; Dolgui, A. Machine learning in manufacturing and industry 4.0 applications. *Int. J. Prod. Res.* **2021**, *59*, 4773–4778. [CrossRef]
2. Lorenzo-Espejo, A.; Escudero-Santana, A.; Muñoz-Díaz, M.L.; Robles-Velasco, A. Machine Learning-Based Analysis of a Wind Turbine Manufacturing Operation: A Case Study. *Sustainability* **2022**, *14*, 7779. [CrossRef]
3. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems—Volume 2, Cambridge, MA, USA, 8–13 December 2014; pp. 3104–3112.
4. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
5. Arik, S.Ö.; Pfister, T. TabNet: Attentive Interpretable Tabular Learning. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 6679–6687. [CrossRef]
6. Popov, S.; Morozov, S.; Babenko, A. Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Online, 26–30 April 2020.
7. Sainz, J.A. New Wind Turbine Manufacturing Techniques. *Procedia Eng.* **2015**, *132*, 880–886. [CrossRef]
8. Masoumi, M. Machine Learning Solutions for Offshore Wind Farms: A Review of Applications and Impacts. *J. Mar. Sci. Eng.* **2023**, *11*, 1855. [CrossRef]
9. Treiber, N.A.; Heinermann, J.; Kramer, O. Wind Power Prediction with Machine Learning. In *Computational Sustainability*; Lässig, J., Kersting, K., Morik, K., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 13–29. [CrossRef]
10. Demolli, H.; Dokuz, A.S.; Ecemis, A.; Gokcek, M. Wind power forecasting based on daily wind speed data using machine learning algorithms. *Energy Convers. Manag.* **2019**, *198*, 111823. [CrossRef]
11. An, G.; Jiang, Z.; Chen, L.; Cao, X.; Li, Z.; Zhao, Y.; Sun, H. Ultra short-term wind power forecasting based on sparrow search algorithm optimization deep extreme learning machine. *Sustainability* **2021**, *13*, 10453. [CrossRef]
12. Fotso, H.R.F.; Kazé, C.V.A.; Kenmoé, G.D. A novel hybrid model based on weather variables relationships improving applied for wind speed forecasting. *Int. J. Energy Environ. Eng.* **2022**, *13*, 43–56. [CrossRef]
13. Wang, J.; Li, H.; Wang, Y.; Lu, H. A hesitant fuzzy wind speed forecasting system with novel defuzzification method and multi-objective optimization algorithm. *Expert Syst. Appl.* **2021**, *168*, 114364. [CrossRef]
14. Neshat, M.; Nezhad, M.M.; Abbasnejad, E.; Mirjalili, S.; Tjernberg, L.B.; Garcia, D.A.; Alexander, B.; Wagner, M. A deep learning-based evolutionary model for short-term wind speed forecasting: A case study of the Lillgrund offshore wind farm. *Energy Convers. Manag.* **2021**, *236*, 114002. [CrossRef]
15. Morshed-Bozorgdel, A.; Kadkhodazadeh, M.; Anaraki, M.V.; Farzin, S. A Novel Framework Based on the Stacking Ensemble Machine Learning (SEML) Method: Application in Wind Speed Modeling. *Atmosphere* **2022**, *13*, 758. [CrossRef]
16. hai Nguyen, T.; Toubeau, J.F.; De Jaeger, E.; Vallée, F. Adequacy assessment using data-driven models to account for aerodynamic losses in offshore wind generation. *Electr. Power Syst. Res.* **2022**, *211*, 108599. [CrossRef]
17. Flores, P.; Tapia, A.; Tapia, G. Application of a control algorithm for wind speed prediction and active power generation. *Renew. Energy* **2005**, *30*, 523–536. [CrossRef]
18. Hoeser, T.; Feuerstein, S.; Kuenzer, C. DeepOWT: A global offshore wind turbine data set derived with deep learning from Sentinel-1 data. *Earth Syst. Sci. Data* **2022**, *14*, 4251–4270. [CrossRef]
19. Niemi, J.; Tanttu, J.T. Deep Learning Case Study for Automatic Bird Identification. *Appl. Sci.* **2018**, *8*, 2089. [CrossRef]
20. Niemi, J.; Tanttu, J. Automatic bird identification for offshore wind farms. In *Wind Energy and Wildlife Impacts: Balancing Energy Sustainability with Wildlife Conservation*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 135–151.

21. Zha, T.; Xie, L.; Chang, J. Wind farm water area path planning algorithm based on A* and reinforcement learning. In Proceedings of the 2019 5th International Conference on Transportation Information and Safety (ICTIS), Liverpool, UK, 14–17 July 2019; pp. 1314–1318.
22. Dong, H.; Zhang, J.; Zhao, X. Intelligent wind farm control via deep reinforcement learning and high-fidelity simulations. *Appl. Energy* **2021**, *292*, 116928. [CrossRef]
23. Yu, M.; Zhang, Z.; Li, X.; Yu, J.; Gao, J.; Liu, Z.; You, B.; Zheng, X.; Yu, R. Superposition Graph Neural Network for offshore wind power prediction. *Future Gener. Comput. Syst.* **2020**, *113*, 145–157. [CrossRef]
24. Zhang, Y.; Yang, X.; Liu, S. Data-driven predictive control for floating offshore wind turbines based on deep learning and multi-objective optimization. *Ocean Eng.* **2022**, *266*, 112820. [CrossRef]
25. Hameed, Z.; Wang, K. Development of Optimal Maintenance Strategies for Offshore Wind Turbine by Using Artificial Neural Network. *Wind Eng.* **2012**, *36*, 353–364. [CrossRef]
26. Cho, S.; Choi, M.; Gao, Z.; Moan, T. Fault detection and diagnosis of a blade pitch system in a floating wind turbine based on Kalman filters and artificial neural networks. *Renew. Energy* **2021**, *169*, 1–13. [CrossRef]
27. Lorenzo-Espejo, A.; Escudero-Santana, A.; Muñoz-Díaz, M.L.; Guadix, J. A Machine Learning-Based System for the Prediction of the Lead Times of Sequential Processes. In *Enterprise Interoperability X: Enterprise Interoperability through Connected Digital Twins*; Rodríguez-Rodríguez, R., Ducq, Y., Leon, R.D., Romero, D., Eds.; Springer International Publishing: Cham, Switzerland, 2024; pp. 25–35. [CrossRef]
28. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 413–422. [CrossRef]
29. Müller, A.C.; Guido, S. Chapter 3: Unsupervised Learning and Preprocessing. In *Introduction to Machine Learning with Python: A Guide for Data Scientists*, 1st ed.; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2017; Chapter 3, pp. 131–209.
30. Wolpert, D.; Macready, W. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
31. Breiman, L.; Friedman, J.; Olshen, R.A.; Stone, C.J. *Classification and Regression Trees*, 1st ed.; Chapman and Hall/CRC: New York, NY, USA, 1984; p. 368. [CrossRef]
32. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
33. Tibshirani, R. Regression Shrinkage and Selection via the Lasso. *J. R. Stat. Soc. Ser. B Methodol.* **1996**, *58*, 267–288. [CrossRef]
34. Hoerl, A.E.; Kennard, R.W. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics* **1970**, *12*, 55–67. [CrossRef]
35. Zou, H.; Hastie, T. Regularization and Variable Selection Via the Elastic Net. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2005**, *67*, 301–320. [CrossRef]
36. Friedman, J.H. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
37. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 13–17 August 2016; pp. 785–794. [CrossRef]
38. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.Y. LightGBM: A highly efficient gradient boosting decision tree. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 4–9 December 2017; pp. 3149–3157.
39. Botchkarev, A. Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology. *Interdiscip. J. Inf. Knowl. Manag.* **2019**, *14*, 045–076. [CrossRef]
40. Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
41. Hastie, T.; Tibshirani, R.; Friedman, J. Model Assessment and Selection. In *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*; Springer: New York, NY, USA, 2009; pp. 219–259. [CrossRef]
42. Harris, C.R.; Millman, K.J.; van der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; et al. Array programming with NumPy. *Nature* **2020**, *585*, 357–362. [CrossRef]
43. McKinney, W. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference, Austin, TX, USA, 28 June–3 July 2010; pp. 56–61. [CrossRef]
44. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
45. Joseph, M. PyTorch Tabular: A Framework for Deep Learning with Tabular Data. *arXiv* **2021**, arXiv:2104.13638.
46. Hunter, J.D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [CrossRef]
47. Waskom, M.L. seaborn: Statistical data visualization. *J. Open Source Softw.* **2021**, *6*, 3021. [CrossRef]
48. Ali, M. PyCaret: An Open Source, Low-Code Machine Learning Library in Python. PyCaret Version 1.0.0. 2020. Available online: https://pycaret.org/ (accessed on 24 July 2024).
49. Shwartz-Ziv, R.; Armon, A. Tabular data: Deep learning is not all you need. *Inf. Fusion* **2022**, *81*, 84–90. [CrossRef]
50. Grinsztajn, L.; Oyallon, E.; Varoquaux, G. Why do tree-based models still outperform deep learning on tabular data? *arXiv* **2022**, arXiv:2207.08815.
51. Loyola-González, O. Black-Box vs. White-Box: Understanding Their Advantages and Weaknesses From a Practical Point of View. *IEEE Access* **2019**, *7*, 154096–154113. [CrossRef]

52. Ribeiro, M.T.; Singh, S.; Guestrin, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 13–17 August 2016; pp. 1135–1144. [CrossRef]
53. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 4–9 December 2017; pp. 4768–4777.