energies

MDPI

Article

Power Transformer Fault Detection: A Comparison of Standard Machine Learning and autoML Approaches

Guillermo Santamaria-Bonfil [1], Gustavo Arroyo-Figueroa [2,*], Miguel A. Zuniga-Garcia [3], Carlos Gustavo Azcarraga Ramos [2] and Ali Bassam [4]

[1] Data Portfolio Manager Department, BBVA Mexico, Mexico City 06600, Mexico; guillermo.santamaria@bbva.com
[2] Instituto Nacional de Electricidad y Energias Limpias, Cuernavaca 62490, Mexico; carlos.azcarraga@ineel.mx
[3] PCI Energy Solutions, Norman, OK 73072, USA; miguel.zugar@gmail.com
[4] Facultad de Ingeniería, Universidad Autónoma de Yucatán, Merida 97000, Mexico; baali@correo.uady.mx
* Correspondence: garroyo@ineel.mx; Tel.: +52-7773623820

Abstract: A key component for the performance, availability, and reliability of power grids is the power transformer. Although power transformers are very reliable assets, the early detection of incipient degradation mechanisms is very important to preventing failures that may shorten their residual life. In this work, a comparative analysis of standard machine learning (ML) algorithms (such as single and ensemble classification algorithms) and automatic machine learning (autoML) classifiers is presented for the fault diagnosis of power transformers. The goal of this research is to determine whether fully automated ML approaches are better or worse than traditional ML frameworks that require a human in the loop (such as a data scientist) to identify transformer faults from dissolved gas analysis results. The methodology uses a transformer fault database (TDB) gathered from specialized databases and technical literature. Fault data were processed using the Duval pentagon diagnosis approach and user–expert knowledge. Parameters from both single and ensemble classifiers were optimized through standard machine learning procedures. The results showed that the best-suited algorithm to tackle the problem is a robust, automatic machine learning classifier model, followed by standard algorithms, such as neural networks and stacking ensembles. These results highlight the ability of a robust, automatic machine learning model to handle unbalanced power transformer fault datasets with high accuracy, requiring minimum tuning effort by electrical experts. We also emphasize that identifying the most probable transformer fault condition will reduce the time required to find and solve a fault.

Keywords: transformer fault diagnosis; machine learning; automatic machine learning; power systems

Citation: Santamaria-Bonfil, G.; Arroyo-Figueroa, G.; Zuniga-Garcia, M.A.; Azcarraga Ramos, C.G.; Bassam, A. Power Transformer Fault Detection: A Comparison of Standard Machine Learning and autoML Approaches. *Energies* **2024**, *17*, 77. https://doi.org/10.3390/en17010077

Academic Editor: Ahmed Abu-Siada

Received: 25 November 2023
Revised: 8 December 2023
Accepted: 10 December 2023
Published: 22 December 2023

1. Introduction

Power transformers are key components for transmission and distribution grids. Although transformers are very reliable assets, the early detection of incipient degradation mechanisms is very important to prevent failures that may shorten their life span [1–3]. The life cycle management of the power transformers is composed of several stages, such as transformer specifications, erection, commissioning, operation, maintenance, and end-of-life operations. In particular, for the last two stages, it is of paramount importance to have suitable tools for assessing a power transformer's condition. Economic consequences of a power transformer's catastrophic failure include (i) costs for the lost transmission of electricity and (ii) repairmen or substitution costs for the faulted power transformer, which can vary according to the electrical system power level, substation topology, and technical characteristics of the transformer. For example, consider the case of lost transmission capability due to a single-phase transformer failure rated at 230 kV, 33 MVA, located somewhere in Mexico. The economic impact is composed of (i) the costs of transmission

loss, which rise up to USD 6,177,600 (since the cost for loss of transmission is around 2.6 USD/kWh in Mexico), and (ii) the direct costs, including a 72 h affectation window (firefighting, damaged facilities' repair, soil remediation operations, reserve transformer testing and commissioning, and fitting all the substations' and systems' prior conditions) with a direct cost around of USD 1,280,000. Therefore, grid operators and utilities require tools that allow them to optimize their decision-making processes regarding transformer repair, refurbishment, or replacement under the umbrella of costing, reliability, and safety optimization [4–6].

Condition assessment (CA) is the process of identifying markers and indexes to determine and quantify the degradation level of transformer components [1,7,8]. Power transformer CA strategies include exhaustive electrical and physicochemical testing, online and/or offline diagnosis techniques, analysis of the operation and maintenance parameters, and the use of condition-based strategies supported by suitable standards and expert knowledge. In fact, an expert assessment is the most effective but costly and time-consuming CA strategy. It requires taking transformers offline and the participation of highly qualified experts to carry out the analysis, increasing the process costs. Thus, utilities are looking forward to more cost-effective CA strategies where few or zero expert interventions are required.

One of the main steps of a transformer CA is the identification of faults through a transformer fault diagnosis (TFD) procedure. The TFD focuses on the transformer insulation system, whose integrity is fully related to transformer reliability [9,10]. The insulating system is exposed to electrical, mechanical, and thermal stresses. These phenomena would be considered normal if they were considered during the transformer's design; otherwise, they are considered abnormal. Among the abnormal behaviors are emergency overloading, arc flashes, transient events, and thermal faults, to mention a few [11,12]. The transformer insulation system has two main components: the insulating fluid (commonly mineral oil) and the solid insulation (kraft paper, pressboard, and other materials). Oil plays a very important role in providing highly reliable insulation characteristics and working as an efficient coolant, removing the heat generated at the core and windings during transformer operation [13]. Further, insulating oil analysis can provide important information regarding transformer degradation and behavior at a very low cost, eliminating the need to carry out expensive offline testing.

Transformer insulating oil is a petroleum-derived liquid that can be based on paraffinic, naphthenic, naphthenic–aromatic, or aromatic hydrocarbons. No matter its structure, insulating oil can be decomposed by abnormal stresses (discharges and heating), producing dissolved byproduct gasses correlated to specific faults. Hence, dissolved gas analysis (DGA) is a widely studied diagnostic technique for which many tools are already available. These tools are based on the analysis of each byproduct gas, its concentration, and the interrelationship between them. Among the most classical methods to diagnose oil samples are the Rogers ratio, IEC ratio, Dornenburg ratio, key gas method, Duval triangles [2,12–16], and Duval pentagons [4,17], to mention a few. Most of those methods are based on dissolved gas ratio intervals that classify transformers into different faults. However, these methods are prone to misinterpretations near the fault boundaries [15,16]. Furthermore, classical DGA methods always identify a fault, even when there is not one; thus, expert assessment is still required to accurately determine if there is a fault or not. On the other hand, coarse DGA-based fault classification methods have a high accuracy rate but have poor usability, whereas fine TFD can be used for decision-making, but its accuracy rate is lower [15]. In general, to decide whether to remove, repair, or replace a transformer in the presence of thermal faults, the fault severity must be determined [17]; thus, finer TFD is preferred. An important avenue for TFD methods is machine learning (ML). Databased algorithms have been proposed to improve TFD performance while avoiding the drawbacks mentioned earlier. ML methods provide high flexibility: they are able to handle linear and nonlinear relations, are robust to noise, do not necessarily require taking into account thermodynamic phenomena, and provide high fault diagnosis performance [18]. ML algorithms that

have been used for the TFD endeavor can be divided into supervised and unsupervised approaches. Supervised ML employs different gas-in-oil ratios already diagnosed by experts using chromatographic results, to build a function that relates those gas ratios with transformer faults or normal or faulty status. Unsupervised approaches employ dissolved gas data to cluster the transformers into groups whose gas ratios are similar to each other. Nevertheless, an expert's diagnosis is always required to assess the performance of the models; thus, this study highlights the supervised approach. Most of the ML works applied to the TFD problem cover one or more of the following steps:

1. **ML algorithms**: Several single classifiers have been used such as the artificial neural network (ANN) [1,12,13,15,19]; expert-guided ANN [15,20]; Bayes network [1]; decision tree (DT) [1,13]; extreme learning machine (ELM) [15]; K-nearest neighbors (KNN) [1,12,13,19]; logical analysis of data (LDA) [21]; logistic (LR) and regularized (LASSO) regression [13]; probabilistic neural network (PNN) [12,22]; softmax regression (SR) [15]; and support vector machine (SVM) [13,19]. Ensembles include boosting and bagging [1,23]; eXtreme gradient boosting (XGBoost) [24]; stacked ANN [2]; and even state-of-the-art algorithms such as few-shot learning with belief functions [5].
2. **Data pre-processing methods**: Several data transformations have been used such as data binarization [21]; key [1,2,5,13,15,22,24] and custom gas ratios [2]; logarithmic transformation [1,2,13]; mean subtraction and normalization [2]; standardization [1,2,13]; imputation of missing values using simple approaches [25]; dimensionality reduction such as linear discriminant and principal components analyses [12], and belief functions [5]; feature extraction such as genetic programming [19]; knowledge-based transformations such as expert knowledge rules [15]; and oil–gas thermodynamics [2].
3. **TFD approach**: The TFD problem has been posed as a binary classification (normal or faulty transformer) [1,2,13,24]; as a multi-class classification with coarse [19,21] and fine fault types [2,12,15,22,24]; and even diagnosing fault severity [5,12].
4. **Classes Imbalance**: Several strategies have been used for balancing, such as bootstrap [19]; re-sampling the minority classes [13,15]; class subsetting [13,15]; and assigning weight coefficients to the minority class [24].
5. **Parameters optimization**: Algorithm parameters have been optimized by hand [2] and through trial and error [12]; exact methods such as grid search (GS) [13,20] and mixed integer linear programming [21]; Bayesian methods [24]; and metaheuristics such as the bat algorithm [22] and genetic and mind evolutionary algorithms [15].
6. **Model overfitting assessment**: The problem of overfitting the TFD classifiers has been handled through the usage of classical [1,13,20] and stratified cross-validation (s-CV) [24].
7. **TFD Performance**: Algorithms' performance has been determined using the accuracy percentage [1,2,5,12,13,15,19,21,22], confusion matrix [2,12,19], the area under receiver operating characteristic (ROC) [1,2,13] and precision–recall (PR) [24] curves, and the micro and macro F1-measure [24].

Even while many works have delved into the usage of ML algorithms for the TFD problem, these present one or more shortfalls, such as (i) training and testing their methods using small datasets; (ii) carrying out comparisons using only standard ML supervised algorithms; (iii) considering only coarse fault types by setting aside fault severity (not to mention that none of the reviewed works considered fault severity as defined by the Duval pentagon method); and (iv) a lack of publicly available data. These issues stand in the way of us obtaining a clear idea of which sequence of methods and algorithms provides the best performance for the TFD problem. Also, it makes the reproducibility of the research results difficult and hinders the deployment of ML solutions to solve the TFD problem of real-world utilities.

The construction of high-performance ML pipelines, regardless of their application, requires the involvement of data scientists and domain experts. This allows us to incorporate domain knowledge into the design of specialized ML pipelines (i.e., the sequence of

data pre-processing, domain-driven feature selection and engineering, and optimized ML models for a given problem [26]). However, the construction of specialized ML pipelines using this approach is long-winded, expensive, complex, iterative, and based on trial and error. This analysis (and the related works) reveals the difficulty associated with operational process experts building intelligent models. These power systems experts can be easily overrun by the selection and combination of ever-growing alternatives of pre-processing methods, ML algorithms, and their parameter optimization, for the solution of the TFD problem. Under these circumstances, the probability of obtaining a final ML pipeline that behaves sub-optimally is higher [26,27]. Hence, there is a growing need to provide power systems technicians with ML tools that can be used straightforwardly to solve power systems problems (e.g., TFD). The approaches used for automatically (without human intervention) and simultaneously obtaining a high-performing combination of data pre-processing, learning algorithm(s), and a set of hyperparameters are branded automatic machine learning (autoML) [27,28]. autoML comprises promising approaches that may be used off the shelf for solving the TFD problem of real-world industries.

Therefore, in this work, we present a deep comparative analysis of a large and supervised ML algorithm pool composed of single, ensemble, and autoML classifiers applied to the TFD problem. The purpose of this review is to compare algorithms' performance for the TFD problem under equal experimental settings, by (i) compiling and sharing a transformer fault database (TDB) of the main dissolved gas data of 821 transformers, and their corresponding diagnostics, (ii) using single and ensemble ML algorithms, as well as state-of-the-art autoML frameworks, to solve the TFD problem, and (iii) solving a real-world TFD multi-classification problem using, for the first time (to the best of authors' knowledge), Duval pentagons' fault and severity classes [29]. In doing so, this analysis improves our comprehension of the ML approaches available for the TFD problem, and it gives a view of how much automation can we expect for the TFD problem, particularly when fault severity is taken into consideration.

The structure of this work is organized as follows: The introduction is presented in the first section. The second section presents a detailed definition of materials and methods used for comparative analysis of standard ML and autoML algorithms. The third section outlines the results and discusses the outcomes obtained in the fault diagnosis of power transformers. The conclusion of this work is shown in the fourth section.

2. Materials and Methods

The complete ML applied in the present work for the multi-class TFD problem is presented in Figure 1. For comparison, we termed the part of the pipeline corresponding to single and ensemble classifiers as the standard ML framework, and the part of the pipeline corresponding to autoML as the autoML framework. Furthermore, we specify a shared pipeline for both ML approaches. The overall ML system consists of five major sections:

1. Data recollection and labeling. In this step, we transformed dissolved gas-in-oil and conducted corresponding diagnostics. We double-checked transformers' diagnostics: first using the Duval pentagons method to obtain the fault severity (if not available), and then using the IEEE C57.104-2019 standard and expert validation to identify normal operating transformers.
2. Initial pre-processing. In this step, we pre-processed gas-in-oil information using several methods found in the literature, namely, the replacement of zero measurements, natural logarithm escalation, and derivation of key gas ratios.
3. Data separation into training (i.e., X_{train} and Y_{train}) and testing (i.e., X*test* and Y*test*) datasets. For this splitting, we considered the number of samples in each class, to avoid leaving classes unrepresented in any of the datasets.
4. Training the ML system:
    a. Standard ML framework. In this step, we carried out a second data pre-processing stage, training, and parameter optimization. We optimized the

parameters from single and ensemble classifiers using a grid search (GS) and cross-validation (CV) procedures.

b. AutoML framework. In this step, the code automatically carried out a warm-start procedure, additional data and feature pre-processing methods, classifier optimization, and ensemble construction.

5. Measuring the test error using several multi-class performance measures. In this step, we evaluated the algorithms comprehensively using several multi-class performance measures such as the κ score, balanced accuracy, and the micro and macro F1-measure.

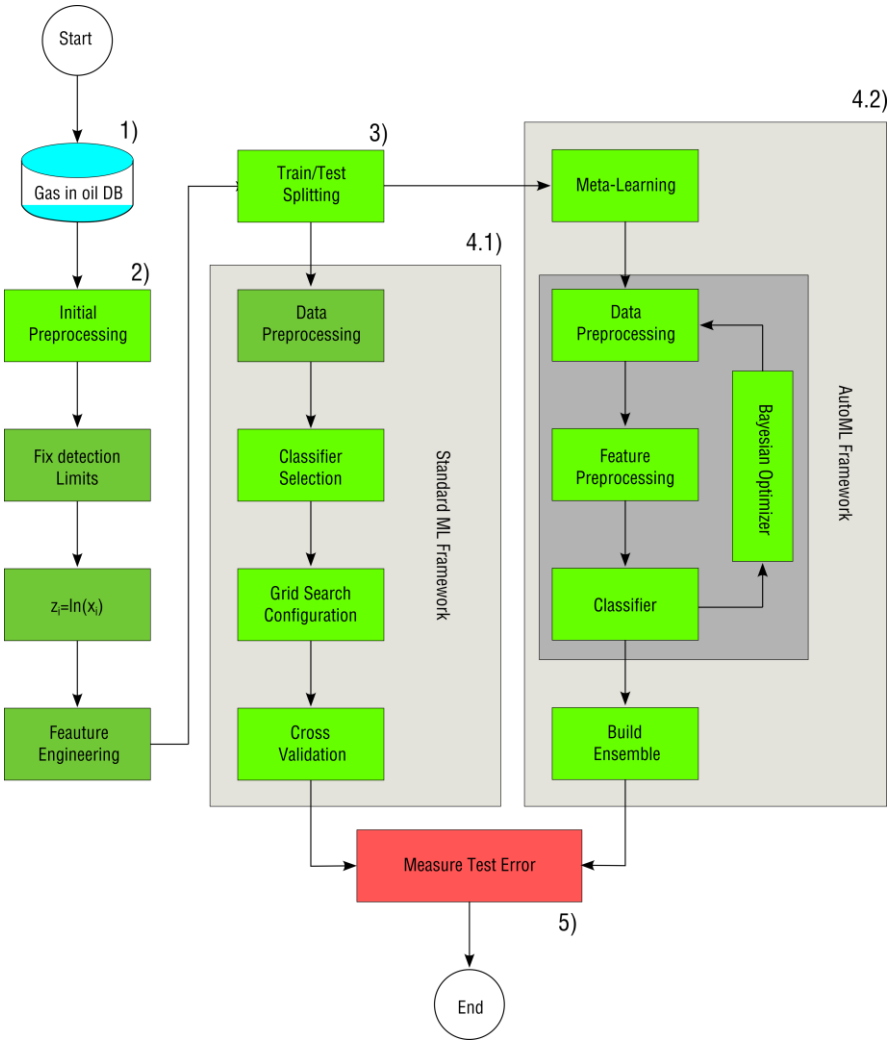


Figure 1. ML methodology developed for the comparison of single, ensemble, and autoML classifiers for the transformer fault classification problem.

2.1. DGA Data

In this work, we constructed a transformer fault database (TDB) comprising 821 samples using different bibliographic sources. These samples were obtained from a specialized database and technical literature: from the International Council on Large Electric Systems (CIGRE), from IEEE [16], technical papers [17,30–34], a CIGRE technical brochure [35], and expert curation. For each transformer, we collected its five thermal hydrocarbon gases and, when reported, their corresponding diagnostics. The collected gases were hydrogen (H_2), methane (CH_4), ethane (C_2H_6), ethylene (C_2H_4), and acetylene (C_2H_2). When available, we recovered the associated diagnostics from the bibliographic sources. Otherwise, we obtained those by means of an analysis method. In this paper, we selected the Duval pentagons as our analysis method [17,29] since it offers not only fault types but also the

severity for thermal faults. It is important to note that, in some cases, this analysis method was also used to confirm the literature-provided diagnostics.

According to [29], the Duval pentagons method first calculates the relative percentage ratios by dividing the concentration of each gas by the total gas content (TGC). Then, the five relative percentage gas ratios are plotted in their corresponding axis in the Duval pentagon, yielding a non-regular five-sided polygon. The centroid of the irregular polygon provides the first part of the diagnostic by indicating the region of the pentagon where it is located. The diagnostic faults available (regions) in the first Duval pentagon are partial discharges (PDs), low and high energy discharges (D1 and D2, respectively), thermal faults involving temperatures less than 300 °C (T1), thermal faults with temperatures ranging from 300 to 700 °C (T2), and thermal faults involving temperatures higher than 700 °C. There is an additional region in the first pentagon called the stray gassing region (S), which reveals another type of gas generation mechanism. Stray gassing is associated with relative low temperatures, oxygen presence, and the chemical instability of oil molecules caused by a previous hydrogen treatment, whose scope is the removal of impurities and undesirable chemical structures in mineral oils. The second part of the Duval pentagons method allows the user to refine the diagnostics by providing advanced thermal diagnostic options: high-temperature thermal faults that occurs in oil only (T3-H), different temperature thermal faults involving paper carbonization (T1-C, T2-C, and T3-C), and overheating (T1-O).

However, all the available classical TFD methods (including the Duval pentagon method) always provide a diagnostic, despite gas concentrations perhaps being too low. To avoid false positives, we used the IEEE C57.104-2019 standard [36] along with expert experience to tag the corresponding transformers with a normal condition diagnostic in these cases. The resulting class distribution for the TDB is shown in Table 1.

Table 1. Transformer fault class distribution.

Fault Type	Freq.	%
Normal condition	270	33
D1	78	10
D2	77	9
PD	23	3
S	83	11
T1-C	4	>1
T1-O	97	16
T2-C	23	3
T3-C	25	3
T3-H	97	12

2.2. Initial Pre-Processing of DGA Data

Before any TFD can be carried out, either with the standard ML or the autoML framework, the TDB must be initially pre-processed. This pre-processing stage consists of three steps: (i) the replacement of zero measurements, (ii) the scaling of measurement values using the natural logarithm (ln) function, and (iii) the derivation of features from dissolved gas ratios. The main reasons for carrying out an initial data pre-processing stage are two-fold. On one hand, data pre-processing methods improve the performance of standard ML frameworks for the TFD problem [1,2,5,13,15,22,24]. On the other hand, autoML frameworks perform better at ML model selection and hyper-parameter optimization (HPO) than the feature engineering (i.e., creation) and data pre-processing methods [26,37]. Furthermore, the selected autoML algorithm used in this work does not consider the pre-processing methods used in the proposed pipeline, nor a feature engineering method that can derive dissolved gas ratios from TDB sample measurements.

The initial pre-processing of DGA data was as follows: First, we considered gas measurements with reported values of zero, as that is below the limit of detection of the chemical procedure analysis. Thus, for the zero measurements, we assumed a small

constant value for mathematical convenience (i.e., 1), or an even smaller constant for C_2H_2 (i.e., 0.1). Second, we scaled gas values using the natural logarithm function. This process is widely suggested for scaling features with positively skewed distributions (i.e., heavy tails), which improves their normality and reduces their variance [38]. Third, we conducted a feature engineering process consisting of the estimation of different ratios from transformed gas values. The relationship between fault types and proportions of dissolved gases in the insulating system has been exploited in traditional DGA methods [9,24,30]. Therefore, several relative ratios based on CH_4, C_2H_6, C_2H_4, C_2H_2, and H_2 were derived. We used these relative ratios as derived features, which are shown in Table 2. In this table, THC (total hydrocarbon content) is the sum of hydrocarbon gas contents, whereas TGC (total gas content) is the total amount of dissolved gas content in the transformer oil.

Table 2. Features derived from dissolved gases.

Feature Name	Definition	Feature Name	Definition
F1	H_2	F13	CH_4/THC
F2	CH_4	F14	C_2H_2/THC
F3	C_2H_2	F15	C_2H_4/THC
F4	C_2H_4	F16	C_2H_6/THC
F5	C_2H_6	F17	CH_4/H_2
F6	$THC = CH_4 + C_2H_2 + C_2H_4 + C_2H_6$	F18	C_2H_6/CH_4
F7	$TGC = H_2 + THC$	F19	C_2H_4/CH_4
F8	H_2/TGC	F20	C_2H_2/CH_4
F9	CH_4/TGC	F21	C_2H_4/C_2H_6
F10	C_2H_2/TGC	F22	C_2H_2/C_2H_6
F11	C_2H_4/TGC	F23	C_2H_2/C_2H_4
F12	C_2H_6/TGC	-	-

2.3. Splitting Data and Training the ML System

Once data were initially prepared, we split them into training and testing datasets. For this split, we considered the proportion of the classes, so each fault type was represented in both datasets, training (X_{train}, Y_{train}) and testing (X_{test}, Y_{test}). The proportions used for splitting the TDB were 70% for training and 30% for testing. Both subsets kept the same class distribution ratios as in the full TDB, to assess classifiers' performance with imbalanced datasets. Afterwards, the ML systems were trained. Before delving into the details of both ML frameworks (standard and autoML), it is worth mentioning that a second stage of data pre-processing was considered to avoid carrying out the same data pre-processing method (i.e., standardization) twice in the autoML approaches.

2.3.1. Standard ML Framework

The standard ML framework follows a classical pipeline: (i) data pre-processing, (ii) selection of the classifier (either single or ensemble), and (iii) optimization of classifier parameters (using a GS-CV procedure). To complete the data pre-processing treatment, we standardized TDB gas measures by subtracting the mean and scaling values by their variance. Next, we selected a classification algorithm, either a single (ANN, DT, Gaussian processes (GPs), naive Bayes (NB), KNN, LR, and SVM) or an ensemble algorithm. The main difference between single and ensemble classifiers is that the first produces a robust model with a good generalization, whereas the second employs several instances of the same classifier. Usually, the classifiers composing the ensemble perform slightly better than a random classifier (e.g., by overfitting), and by using different combining strategies, a good generalization is attained. Among the ensemble strategies are boosting (bagging classifier (BC), histogram (HGB) and extreme (XGBoost) gradient boosting), bagging (random forest (RF)), and stacking (SE). The stacked ensemble is a particular case where two or more strong classifiers are sequentially chained. For this study, an ANN followed by SVM was employed.

Single and ensemble classifiers have been neatly discussed elsewhere; however, for the sake of completeness, they are briefly detailed in Appendices A.1 and A.2, respectively. Meanwhile, in Table 3, the parameters employed by single and ensemble classifiers are presented. The optimal values were estimated using a grid search cross-validation procedure with k = 5 folds.

Table 3. Fault classification models' parameters.

Model	Parameter	Value Range
ANN	Activation function	SVM
	Hidden layer sizes	10–30
	L2 regularization	Linear, Radial
	Learning rate (initial)	0.001–0.4
BC	Base model	SVM
	C	10–30
	Kernel	Linear, Radial
	γ	0.001–0.4
	No. of models	10–100
DT	Max. depth	2–20
	Min. samples per leaf Obj. function	2–5
		Gini, Entropy
	Split strategy	Best
	L2 regularization	1–1.5
GB	Learning rate	0.1–0.15
	Max. iterations	100–200
	Max. depth	2–5
GP	Kernel	RBF, Matern
KNN	K-neighbors	2–10
	Weight	2–10
	p	1–3, 10
LR	Penalty	L2
NB	Variance smoothing	$log10$ (0 to -9)
RF	Class weights	Balanced, Balanced Sub-samples
	No. of trees	200, 500
	Max. depth	2, 3, 10
	Max. features	$\sqrt{\#features}$
	Obj. function	Gini, Entropy
SE	First model	SVM
	C	0.1–8
	Kernel	Linear
	Second model	NB
	Variance smoothing	1×10^{-9}
SVM	Kernel	Linear, Radial
	C	0.1, 0.5, 1
	γ	$\frac{1}{no.features\ \sigma'}$, $\frac{1}{no.features}$
	ε	0.25–0.4

2.3.2. AutoML Framework

AutoML tools are frameworks whose main purpose is to make ML available for people who are not ML experts. Among these tools, we selected the auto-Sklearn algorithm, which is one of the first autoML frameworks and provides robust and expert-competitive results in several ML tasks [26,28,39]. auto-Sklearn is an algorithm based on the Python scikit-learn (Sklearn) library [40]. It is employed for building classification and regression pipelines by searching over single and ensemble ML models. This algorithm explores semi-fixed structured pipelines by setting an initial fixed set of data cleaning steps. Then, a sequential model-based algorithm configuration (SMAC) using Bayesian optimization in combination with a random forest regression allows the selection and tuning of optional pre-processing and mandatory modeling algorithms. In addition, auto-Sklearn provides parallelization

features, meta-learning to initialize the optimization procedure, and ensemble learning through the combination of the best pipelines [26,28,39].

To improve the analysis between standard ML and autoML frameworks, two au-toML versions are considered, namely, vanilla auto-Sklearn and robust auto-Sklearn models. The main differences between them are that (i) the vanilla model only considers a single regression model whereas the robust model employs an ensemble, and (ii) the vanilla model does not employ the meta-learning warm-start stage to initialize the optimization procedure, whereas the robust model does. In this sense, the vanilla model serves as a baseline for the autoML framework. autoML classifiers have been discussed elsewhere; however, for the sake of completeness, they are detailed in Appendix B.

2.4. Classification Performance Metrics

To compare the performance of the standard ML and the autoML frameworks, we employed several multi-classification metrics. As mentioned before, several classification metrics have been employed for the analysis of algorithms' performance for the TFD problem (i.e., the accuracy percentage, confusion matrix, the area under the receiver operating characteristic (AUCROC) and precision–recall (AUCPR) curves, and the micro and macro F1-measure). However, neither the accuracy percentage nor the AUCROC is sensitive to class imbalance. Further, neither the AUCROC nor the AUCPR is suitable for analyzing a multi-classification problem. Therefore, in this work, we employed the confusion matrix (CM), the balanced accuracy (BA), the F1-measure (F1) using micro and macro averages, Cohen's kappa (κ) metric, and Matthews' correlation coefficient (MCC).

On one hand, the CM is a tool to understand the errors of classifiers in binary, multi-class, and even multi-label scenarios. On the other hand, the remaining performance metrics used in this work are obtained from the CM. The selected metrics are useful for assessing the overall performance of a classifier in a multi-class problem. From these, MCC and κ (and, in a lesser sense, F1-macro) are more robust than the remaining for assessing the expected performance of classifiers in the presence of class imbalance.

2.5. Software

We conducted all the experimentation required for TFD ML algorithms' comparison, i.e., pre-processing, training, and testing, using the Python programming language in a Jupyter notebook. We used standard Python packages, such as numpy [41] and pandas [42], for the initial pre-processing stages. For training the classical and most of the ensemble ML algorithms, we employed the sklearn [40] package (in the case of xGB, the xgboost [43] package was used). For the autoML case, we used the autosklearn package [28]. The computer notebook is available in a GitHub repository.

It is worth noting that, while it would be a good idea to use the MCC and κ as a cost function for training the algorithms, due to sklearn package limitations, algorithms' training cost function is restrained to the F1-macro.

3. Results

This section presents the TFD classification results obtained for algorithms of the standard ML and the autoML frameworks. For each classifier, we calculated five (5) performance metrics (as described in the above section). Using those metrics, we carried out a quantitative comparative analysis to determine the best algorithm(s). For a deeper analysis of the performance of the rest of the algorithms, we carried out a multi-objective decision-making (MODM) comparison. Afterward, through the CM, we analyzed class imbalance, false positives, and false negatives of the best-performing algorithm.

3.1. Overall Classifier Performance for the TFD Problem

In Table 4, we present the performance of standard ML and the autoML frameworks' results for the five quality metrics. We highlight the best performing solutions in bold. It can be observed that, in general, the best-performing algorithm is the robust auto-Sklearn model

for the five quality metrics. This model outperformed the rest of the algorithms, particularly for the F1-macro measure, where the closest competitors (ANN and SE models) attained approximately 10% lower F1-macro scores. These results show the ability of the robust auto-Sklearn model to handle an imbalanced TDB, providing the highest classification performance among all the tested algorithms, needing the minimum tuning effort from the humans in the loop (i.e., electrical experts carrying out a TFD). Therefore, the robust auto-Sklearn model seems preferrable as an off-the-shelf solution for the TFD problem.

Table 4. Classifiers' performance attained on the transformer fault detection problem.

Model	BA	F1-Micro	F1-Macro	κ	MCC
vanilla auto-Sklearn	0.812	0.866	0.769	0.837	0.837
robust auto-Sklearn	**0.893**	**0.906**	**0.909**	**0.885**	**0.885**
ANN	0.840	0.882	0.785	0.856	0.857
BC	0.745	0.858	0.742	0.825	0.826
DT	0.744	0.837	0.695	0.802	0.802
GP	0.728	0.900	0.746	0.875	0.876
HGB	0.723	0.886	0.706	0.860	0.862
KNN	0.765	0.858	0.756	0.823	0.823
LR	0.666	0.823	0.673	0.791	0.792
NB	0.700	0.764	0.650	0.714	0.715
RF	0.717	0.870	0.721	0.840	0.841
SE	0.838	0.886	0.809	0.860	0.861
SVM	0.766	0.882	0.767	0.856	0.857
XGB	0.702	0.861	0.683	0.831	0.833

3.2. Analysis of the Frameworks' Performance

The above results show that the robust auto-Sklearn model (autoML algorithm) is the best-performing algorithm in the TFD problem using the TDB. However, it is not clear how worse the performance levels of the remaining algorithms were in comparison. Also, there might be cases where using the robust auto-Sklearn model is not possible due to issues related to model explainability, training computational cost, productizing models, or other business-related issues raised by utility stakeholders. In such scenarios, it would be useful to determine if the vanilla auto-Sklearn model (or another single autoML framework such as auto-WEKA [36]) is better or worse than single/ensemble standard classifiers. When considering the results for the ANN and SE algorithms, we found that these were better for the five metrics in comparison to the vanilla auto-Sklearn model. Similarly, other single and ensemble algorithms (such as SVM and HGB) performed better than the vanilla model for F1-micro, κ, and MCC. To improve the performance comparison, metric results for each algorithm were transformed using the vanilla auto-Sklearn result as a baseline, as follows:

$$N_i(A) = 1 - \frac{M_i(A)}{M_i(auto\ Sklearn\ vainilla)} \quad (1)$$

where $Mi(A)$ corresponds to the i metric result for algorithm A, $Mi(auto\text{-}Sklearn\ vanilla)$ corresponds to the i metric result for the vanilla auto-Sklearn model, and $Ni(A)$ corresponds to the baseline transformed value for the i metric and algorithm A. For instance, for BA and ANN, the baseline transformed value B A(ANN) is obtained, such as $1 - \frac{\mathrm{BA(auto-Sklearnvanilla)}}{\mathrm{BA(ANN)}}$. The transformed values can be interpreted as follows: an $Ni(A) > 0$ value implies that the performance of the A algorithm is better than the vanilla auto- Sklearn algorithm. In contrast, if $Ni(A) < 0$, then the A algorithm's performance is worse than the vanilla auto-Sklearn algorithm.

Once metric values were transformed, we carried out an MODM comparison. MODM deals with problems where two or more performance criteria are used together to make a decision: in our case, we were looking for an algorithm capable of identifying specific electric transformer faults, as accurately as possible, in terms of five performance metrics. In an MODM, model quality is defined by a n-dimensional vector where n corresponds to the number of metrics used. Hence, an algorithm solving an MODM must consider either a

way to simplify a vector of quality metrics into a single scalar, or a way to handle multiple objective functions all at once.

Regarding the methods that solve multiple objective functions, they all use the Pareto approach (PA) [44]. In the PA, instead of handling the burden of collapsing multiple metrics into a single value, you instead look to find a set of solutions (e.g., TFD classification algorithms) that are non-dominated. To define this concept, it is easier first to define the opposite, i.e., dominance. A solution *si* is said to dominate a solution *sj* if *si* is strictly better than *sj* in at least one of the quality metrics *ci*, $i = 1,\ldots, n$, and equal or better in the remaining metrics. Formally, this comprises (i) $\exists ci \mid ci(si) > cj(si)$ and (ii) $\forall ci \mid ci(si) \geq cj(si)$ (where *ci*(*si*) stands for the quality metric value for solution *si*) [44]. On the other hand, two solutions *si* and *sj* are said to be non-dominating with respect to each other if (i) quality metric values for solution *si* are strictly better than *sj* in at least one of the *ci*, $i = 1,\ldots, n$, and (ii) quality metric values for solution *si* are strictly worse than *sj* in at least one of the quality metrics *ci*, $i = 1,\ldots, n$. The set of non-dominated solutions is also known as the Pareto frontier. In Figure 2, the Pareto analysis carried out on the vanilla transformed quality metrics, excluding the robust auto-Sklearn model, is shown.

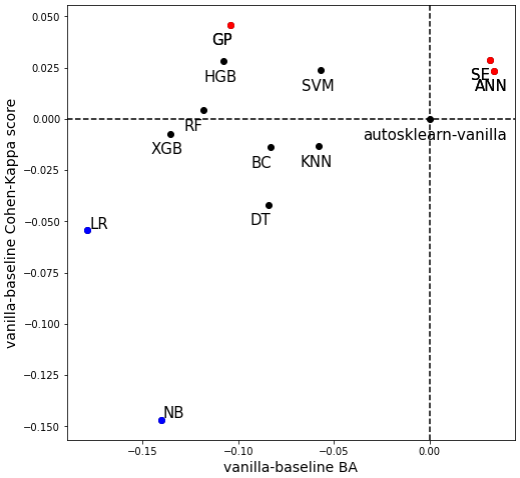


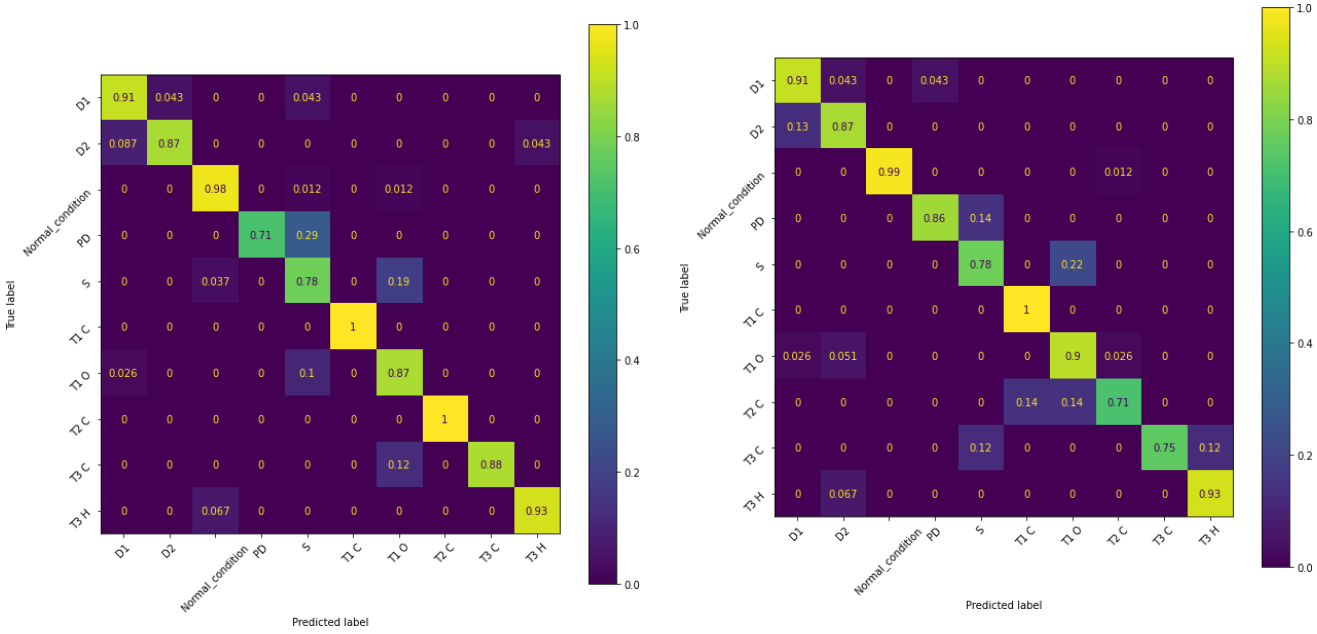
Figure 2. Models' fault classification performance.

Observe that the vanilla auto-Sklearn model is shown at the origin (0,0); algorithms in the Pareto frontier are depicted in red, whereas the worst-performing algorithms are displayed in blue. From this figure, note that the SE, ANN, and GP algorithms performed better than the vanilla auto-Sklearn (for BA, the improvements were 3%, 3%, and −10%, respectively, whereas for κ, the improvements were 3%, 2%, and 4.5%, respectively). Hence, and without considering the robust auto-Sklearn algorithm, either of these can be selected for the TFD problem. On the other hand, HGB and SVM, while they performed better for the κ metric than the vanilla auto-Sklearn (3% and 2%, respectively), could be considered as good as the vanilla auto-Sklearn model in a Pareto front sense (and, in a lesser sense, the RF case). The remaining algorithms should be considered to have performed worse than the vanilla auto-Sklearn model. Specifically, the LR and NB algorithms performed considerably worse than the vanilla model: for the BA metric, 17% and 14% worse, and for the κ metric, 5% and 14%, respectively. In summary, single autoML frameworks provide a good identification of transformer faults with minimal human intervention; still, standard ML approaches such as ANN, SE, or GP classifiers would provide better results for the TFD problem.

3.3. Transformers' Fault Diagnosis in Detail

In accordance with the above results, the overall best-performing algorithm for the TFD problem is the robust auto-Sklearn (AutoML) algorithm. But how was its performance for each transformer fault type? And how did its performance compare against one of

the algorithms belonging to the Pareto frontier such as the SE algorithm? In Figure 3, we present the confusion matrix for both algorithms: in Figure 3a, the robust auto-Sklearn is shown, whereas in Figure 3b, the SE is displayed. It can be observed that, in general, for both algorithms, most fault types were identified with a good ($\geq$80%) to very good ($\geq$90%) accuracy, except for the following: in (a) for PD and S, with an accuracy of 71% and 78%, respectively; in (b) for S, T2-C, and T3-C, with an accuracy of 78%, 71%, and 75%, respectively. To examine the regular performance on these fault types, it is useful to recall that when analyzing the performance of an algorithm using the multi-class CM (see section Appendix C), rows indicate false negatives (FNs) and columns indicate false positives (FPs), respectively. Thus, for the case of the robust auto-Sklearn algorithm, PD faults were misclassified 29% of the time as S fault types; while S faults were misclassified 19% of the time as T1-O faults and 3.7% of the time as a normal condition. For the case of the SE algorithm, S faults were misclassified 22% of the time as a T1-O faults; T2-C faults were misclassified 14% of the time as T1-O and T1-C faults; and T3-C faults were misclassified 12% of the time as T3-H and S faults. From all these errors, the robust auto-Sklearn algorithm incurs in the most expensive ones (i.e., classifying a fault as a normal condition). Further, the misclassification from both algorithms can be attributed to the fault regions described by these for each fault type. Those do not necessarily match the Duval pentagon fault regions, which are geometrically contiguous and do not overlap [17]. In addition, recall that all of these classes, i.e., PD, S, T2-C, and T3-C, are underrepresented in the TDB (see Table 1). In the light of these findings, we can conclude that samples misclassified may lay at the class limits, and/or class boundaries found by the algorithms have a different geometric shape than the one defined by the Duval pentagon. Therefore, increasing the sample size of imbalanced classes (either real or synthetic samples) should be useful for improving the boundaries defined in the feature space for each class by both algorithms. Finally, it is worth noting that both algorithms classified with 100% accuracy the low thermal faults involving paper carbonization (i.e., T1-C), which is the most underrepresented class in the TDB.



(**a**) (**b**)

Figure 3. Confusion matrix for the (**a**) robust auto-Sklearn model and (**b**) for the stacking ensemble algorithms.

4. Conclusions

This paper has presented a comprehensive review and comparative analysis of standard machine learning algorithms (such as single and ensemble classification algorithms) and two automatic machine learning (autoML) classifiers for the fault diagnosis of power transformers. The primary objective of this study was to compare the performance of classical ML classification algorithms, which require human-in-the-loop experts for tuning, with two autoML approaches that demand minimal human operation. To achieve this, data of transformer faults were collected from the literature, as well as from databases from both Mexican and foreign utilities and test laboratories. Subsequently, raw data were curated, and faults were validated and assigned using both the Duval pentagon method and expert knowledge. The methodology used for comparison included: (i) several pre-processing steps for feature engineering and data normalization; (ii) different ML approaches (single ML and ensemble algorithms were trained and tuned using a GS-CV by a data scientist, whereas the autoML models were trained and tuned using Bayesian optimization in combination with a random forest regression with zero human intervention); (iii) several algorithm performance approaches using global metrics, a Pareto front analysis, and a CM to have a detailed look into the types of biases algorithms suffer. A key contribution of this work is that, for the first time (to the best of the authors' knowledge), it has defined fault classes using Duval pentagons and severity classes.

Our results showed that the robust auto-Sklearn achieved the best global performance metrics over standard single and ensemble ML algorithms. On the other hand, the PA showed that the vanilla autoML approach performed worse than some single (ANN, SVM) and ensemble (SE, HGB, GP, and RF) ML algorithms. The CM revealed that, while the robust auto- Sklearn algorithm obtained the highest global performance metric values, it misclassified some faults as a normal condition. This type of error can have a very negative impact on power grid performance (blackouts) with high financial costs. The misclassification can be attributed to the imbalanced TBD. Increasing the sample size of the imbalanced classes (either real or synthetic samples) should be useful for improving the boundaries defined in the feature space for each class. In conclusion, the robust auto-Sklearn model is not only a good off-the-shelf solution for the TFD while handling imbalanced datasets but also achieved the highest global classification performance scores using the minimum tuning effort by a human (i.e., electrical experts carrying out a fault diagnosis). This comparative analysis has extended our comprehension of the ML approaches available for the TFD problem, and it has given a view of how much automation we can expect for a real TFD problem, particularly when fault severity is taken into consideration. In future work, the best models (ensemble SE and robust auto-Sklearn) will be incorporated into a power transformer condition assessment in a maintenance management system. It is expected that failure classification indicating the most probable defect will be used to help engineers reduce the time needed to find and repair incipient faults, which will help to avoid catastrophic failures and fires.

Author Contributions: Conceptualization, G.S.-B., G.A.-F. and C.G.A.R.; software, G.S.-B. and M.A.Z.-G.; validation, G.S.-B. and M.A.Z.-G.; investigation, G.A.-F. and C.G.A.R.; data curation, G.S.-B.; writing—original draft, G.S.-B.; writing—review and editing, G.S.-B., G.A.-F., C.G.A.R., M.A.Z.-G. and A.B.; supervision, G.A.-F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to being compiled from different sources and validated by power systems experts.

Acknowledgments: The authors would like to thank Roberto Liñan from Instituto Nacional de Electricidad y Energías Limpias for his comments and reviews.

Conflicts of Interest: The author G.S.-B. wants to clarify that the presented article contents are all his own opinions and research lines, and not necessarily the opinions of BBVA México. The author M.A.Z.-G. wants to clarify that the presented article contents are all his own opinions and research lines, and not necessarily the opinions of PCI Energy Solutions. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A. Machine Learning Algorithms

Appendix A.1. Single Classifiers

Appendix A.1.1. Artificial Neural Networks

Artificial neural networks (ANNs) are models inspired by the central nervous system, which are made of interconnected neurons [45]. The calculation of a single-layer perceptron is performed using the following equation:

$$y(\hat{X}) = g\left(\sum_{i=1}^{n}(w_i x_i)\right) + bias \tag{A1}$$

where g is the activation function. Most artificial neural networks are composed of three types of layers of neurons: an input layer, one or more hidden layers, and an output layer. These are called multi-layer perceptrons (MLPs). The input layer is responsible for receiving a given input vector and transforming into an output that becomes the input for another layer. Hidden layers transform the output from a previous layer by means of an activation function. The output layer produces the final weighted sum.

$$y(\hat{X}) = g\left(\sum_{i=1}^{n}(w_i x_i) + bias\right) \tag{A2}$$

Appendix A.1.2. Decision Trees

A decision tree is a classification method that creates a recursive partition of the dataset [46]. It consists of nodes that build a tree, which has a node called the "root" that has no incoming connections, and a set of nodes that have one incoming connection. Nodes with outgoing connections are called internal nodes and all other nodes (except the root node) are called leaves or decision nodes. Each internal node splits the dataset into two or more sub-datasets using a certain condition or criteria of the input variables.

The objective is to construct the following classifier:

$$Y(X) = \sum_{i=1}^{n} Y_i x l_i(X) \tag{A3}$$

Appendix A.1.3. Gaussian Processes

The Gaussian process classifier is a classification method that assumes the class densities follow a normal distribution [47]. A Gaussian process is a generalization of a multivariate Gaussian distribution of infinite random variables.

A multivariate Gaussian distribution is defined by the following equation:

$$N(x; \mu, \sum) = \frac{1}{(2\pi)^{d/2}} \sum^{-1/2} exp^{-\frac{1}{2}}(x-\mu) \sum^{-1} (x-\mu)^T \tag{A4}$$

where x is the vector of random variables, μ is the vector of means, and Σ is the covariance matrix of all random variables.

A Gaussian process is a random process in which any point $t \in R^d$ is assigned to a random variable Z_t such that every (Z_{t1}, Z_{t2},…,Z_{td}) is a multivariate Gaussian.

Appendix A.1.4. K-Nearest Neighbors

K-nearest neighbors (KNN) is one of the most basic and essential classification algorithms in machine learning [48]. It belongs to the domain of supervised learning and finds intense application in pattern recognition, data mining, and intrusion detection. The KNN classifier is also an instance-based, non-parametric learning algorithm. Instance-based learning means that our algorithm does not explicitly learn a model. Instead, it chooses to memorize the training instances that are later used as "knowledge" for the prediction phase. Specifically, this means that only when a query is made to our database, that is, when we ask it to predict a label with an input, will the algorithm use the training instances to give an answer.

The KNN algorithm assumes that pairs (X_1, Y_1), $(X_2, Y_2) \ldots (X_n, Y_n)$ take values in the coordinate space R^d, where $X_n \in R^d$ and Y_n is the label of the element X_n. Then, to perform a classification y of a new point $x \in R^d$, the KNN uses some distance measure $\|\cdot\|$ in R^d such that the dataset of pairs $(X_{(1)}, Y_{(1)})$, $(X_{(2)}, Y_{(2)}) \ldots (X_{(n)}, Y_{(n)})$ is reordered given that $\|X_{(1)} - x\| \leq \|X_{(2)} - x\| \ldots \|X_{(n)} - x\|$. The class y is assigned given the maximum probability of the $Y_{(1)} \ldots Y_{(K)}$ estimated, where K is the number of neighbors selected to make the classification.

Appendix A.1.5. Naive Bayes

In simple terms, it assumes that the presence or absence of a particular characteristic is not related to the presence or absence of any other characteristic, given the variable class [49]. For example, a fruit can be considered an apple if it is red, round, and about 7 cm in diameter. A naive Bayes classifier considers that each of these characteristics contributes independently to the probability that this fruit is an apple, regardless of the presence or absence of the other characteristics. This method is based on the Bayes theorem, which is described as follows:

$$P(A|R) = \frac{P(R|A)P(A)}{P(R)} \tag{A5}$$

The naive Bayes method assumes that the probability of all variables is constant, so the Bayes theorem can be rewritten as follows:

$$P(A|R) = P(R).\prod_{i=1}^{n} P(R|A) \tag{A6}$$

Appendix A.1.6. Logistic Regression

Regression analysis determines how a variable y is related to one or more other variables $X = x_1, x_2, \ldots, x_k$. In logistic regression [50], the y_is are considered binary variables and the distribution of y_i given X is assumed to follow a Bernoulli distribution (which is a special case of a binomial distribution), such that

$$log\left(\frac{p(y_i = 1|X)}{1 - p(y_i = 1|X)}\right) = \beta_0 + \sum_{j=1}^{k} \beta_j X_{ji} \tag{A7}$$

Of the several methods to estimate the βs, the method of maximum likelihood is one most commonly used for logistic regression.

Appendix A.1.7. Support Vector Machines

Support vector machines (SVMs) are supervised learning models used for data prediction and classification [51]. This method was first presented in [52] in 1998. Even though SVMs are commonly used for classification problems where training data are linearly separable, nonlinear data can be mapped into a high-dimensional feature space with the help of a kernel function where linear regression can be applied (the hyperplane). The method builds two bounds with radius epsilon (ϵ) parallel to the hyperplane that covers the greatest quantity of data, where ϵ defines a margin of tolerance where no penalty is

given to errors and ϵ is the distance between data not covered by support vectors and bounds [53]. This is achieved by minimizing the following expression,

$$y = max \frac{2}{\|w\|} \tag{A8}$$

subject to

$$w.x + b \geq 1, \forall x = C_1 \tag{A9}$$
$$w.x + b \geq -1, \forall x = C_2$$

Appendix A.2. Ensembles

Ensemble methods are algorithms that improves the accuracy by joining/merging two or more simpler algorithms. Basically, there are two kinds of ensemble algorithms: sequential and parallel. Sequential ensemble algorithms have the characteristic of maintaining dependence across the different models generated. On the other hand, parallel ensemble algorithms look for independence among the different models.

Appendix A.2.1. Random Forest

A random forest is a parallel type of ensemble algorithm [54]. A random forest is a combination of decision trees so that each tree analyzes only a random segment of the data, with the same distribution for each of them. The essential idea is to average many noisy but approximately unbiased models, and thereby reduce variance.

Appendix A.2.2. Bagging Classifier

A bagging classifier is a parallel ensemble algorithm. This algorithm is very similar to the random forest [55]. It takes several samples of the data with the same distribution, to train different models using a variety of methods. The final classifier makes a prediction by combining the predictions of all other models.

Appendix A.2.3. Gradient Boosting

Gradient boosting is a sequential ensemble algorithm [56]. Gradient boosting has two important characteristics: weak models and a differentiable loss function. This algorithm uses weak models as an optimization parameter and it decides to add or remove models depending on the gradient value of the loss function.

Appendix A.2.4. Stacking Ensemble

The stacking ensemble is a sequential algorithm [57]. It uses a meta learning algorithm to learn how to best combine the predictions of two or more basic machine learning algorithms. The predictions of the basic machine learning algorithms are used to create a second training set. This second training set is used to train the meta learning algorithm.

Appendix B. AutoML Algorithms

In accordance with [26], an ML pipeline *h*: X → Y involves the computational, intensive, repetitive, sequential combination of algorithms, which maps any given observation, i.e., x ∈ X, into a discrete (e.g., class) or continuous value, i.e., y ∈ Y.

To define an ML pipeline, i.e., h, let us first define its components. The set of specific algorithms, e.g., data pre-processing, feature selection, and classification, is defined as A = {A^1, A^2,…, A^n}. For each algorithm A^i, a configuration vector of hyperparameters is defined as $\lambda(i) \in \Lambda_{A(i)}$. Algorithms of A are connected with each other in accordance with a structure g. Such a structure is defined as a directed acyclic graph (DAG) where nodes represent algorithms and edges represent the data flow. The structure has implicit constraints (e.g., an imputation algorithm must precede a classification one), and thus, it belongs to a set of valid pipeline structures G; its cardinality (i.e., number of consecutive

sequential algorithms) is given by $|g|$. Therefore, an ML pipeline is formally given by $P(g, \boldsymbol{A}, \boldsymbol{\lambda})$, where $g \in G$ stands for the structure of the valid pipeline, the vector $\boldsymbol{A} \in A^{|g|}$ stands for the algorithms selected for each node, and the vector $\boldsymbol{\lambda} \in \Lambda_A$ stands for the hyperparameters for each of the selected algorithms in the pipeline.

Therefore, given a problem defined by the i.i.*d*. samples $D = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ drawn from the joint probability distribution $P(\mathbb{X}, \mathbb{Y})$, an ML pipeline is created by finding the structure, algorithms, and their corresponding hyperparameters, which minimizes the empirical pipeline performance (EPP) $\hat{R}$, such as

$$\hat{R}\left(P_{g,A,\lambda}, D\right) = \frac{1}{m}\sum_{i}^{m} L(h(X_i), Y_i), \tag{A10}$$

where $h(x_i)$ is the predicted output of the pipeline $\mathcal{P}$, and $\mathcal{L}$ is a loss function. Furthermore, to avoid overfitting $\mathcal{P}$, cross-validation is considered. Hence, D is split into k disjointed folds, that is, $\{D_{valid}{}^{(1)}, \ldots, D_{valid}{}^{(k)}\}$ and $\{D_{train}{}^{(1)}, \ldots, D_{train}{}^{(k)}\}$. By rewriting Equation (A10) to include these, the final objective function is obtained:

$$\left(P_{g,A,\lambda}\right)\epsilon \frac{\arg\ min}{g\epsilon G, A \in A|g|, \lambda \in \Lambda} \frac{1}{k}\sum_{i}^{k} \hat{R}\left(P_{g,A,\lambda,D_{train}^{(t)}}, D_{valid}^{(t)}\right) \tag{A11}$$

To minimize the cost function presented in Equation (A11), three sub-problems must be addressed altogether: the (i) structure search; the (ii) algorithm selection; and (iii) algorithm hyperparameter optimization (HPO). On the one hand, in a recent survey [26], it was stated that most of the autoML frameworks avoid solving the structure search by following a best-practice fixed pipeline structure. This approach removes the burden of determining the graph structure g in Equation (A11). On the other hand, the algorithm's selection and its HPO are simultaneously determined by solving the combined algorithm selection and hyperparameter optimization (CASH) problem [26–28]. Solving the CASH problem is similar to solving Equation (A11). If the pipeline structure g is fixed, i.e., $|g| = 1$, the CASH problem is defined as follows:

$$(A, \lambda) * \frac{arg\ min}{A \in A, \lambda \in A} R\left(P_{g,A,\lambda,D}, D\right) \tag{A12}$$

To simultaneously consider which sequence of algorithms to use and their corresponding hyperparameters, the λr vector is defined. This allows one to map the sequence of algorithms into the Λ configuration space, such that

$$\Lambda = \Lambda_{A(1)} x \ldots \Lambda_{A(n)} x\ \lambda_r.$$

In consequence, the CASH minimization problem is stated as follows:

$$\lambda * \in \frac{\arg min R\left(p_{g,\lambda,D}, D\right).}{\lambda \in \Lambda} \tag{A13}$$

Equation (A13) is a mixed-integer nonlinear optimization problem. Its solution involves finding algorithms' numerical or categorical hyperparameters, which are mandatory or conditional (whose values depend on the selection of other hyperparameters) [26].

To solve all of these numerical cruxes, several autoML frameworks based on standard ML and ensembles [26,39], as well as neural networks' deep learning frameworks [58], have been proposed. Due to the infancy of the autoML area, recent reviews reveal that most of the available autoML tools obtain competitive, but similar, results across several ML tasks [26,37].

Appendix C. Performance Measures

Appendix C.1. Confusion Matrix

A confusion matrix (CM) is employed to assess a classifier's performance by comparing predictions against true class labels. In this way, several performance measurements can be obtained [59–61]. Most CM-derived performance metrics are devised for binary classification problems, and only a few are available for a multi-class problem [61].

The confusion matrices for the binary and the multi-class problems are presented in Figure A1. TP stands for true positive, FP stands for false positive, TN stands for true negative, and FN stands for false negative. Thus, columns represent classifiers' predictions, whereas rows represent the true classes. The CM element at row i and column j (i.e., $C_{i,j}$) provides the frequency of the predicted class j for the actual class i.

In the case of a binary classification problem, the CM is given by a 2×2 matrix. In the case of a multi-class problem, the matrix is given by N different class labels $c_i \mid i = 1, 2, \ldots,$ N; thus, the CM is of size $N \times N$.

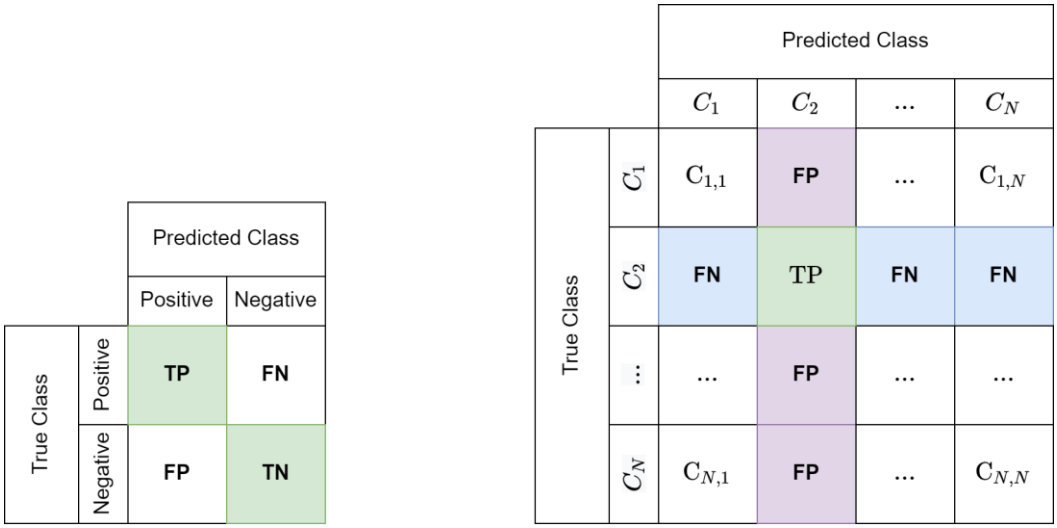


Figure A1. (**a**) CM for a binary classification problem, and (**b**) CM for a multi-classification problem. TP stands for true positive, TN stands for true negative, FP stands for false positive, and FN stands for false negative. Observe that in the multi-class case, only C_2 is being shown.

Appendix C.2. Precision and Recall

Two performance measures are obtained from the confusion matrix of a binary classification problem, namely precision (*Prec*) and recall (*Rec*):

$$Prec = \frac{TP}{TP + FP} \tag{A14}$$

$$Rec = \frac{TP}{TP + FN} \tag{A15}$$

Loosely speaking, precision quantifies a classifier's discrimination ability for separating between the classes, whereas recall quantifies a classifier's ability to identify the samples of a given class in the dataset.

Micro and Macro Averages for Multi-Class

To use these metrics in a multi-classification problem, one must choose macro or micro averaging. In the case of the former, the metric is calculated for each class and then

averaged (i.e., treating all classes equally). In the case of the latter, the contribution of each class is aggregated before computing the average metric, putting more weight on the most populous classes. Formally, if we define $Prec_i$ and Rec_i as the precision and recall obtained for class $i \in N$, respectively, then the macro and micro *Prec* and *Rec* are defined as follows:

$$Prec^{macro} = \frac{1}{N}\sum_{i=1}^{N} Prec_i \tag{A16}$$

$$Rec^{macro} = \frac{1}{N}\sum_{i=1}^{N} Rec_i \tag{A17}$$

$$Prec^{micro} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} TP_i + FP_i} \tag{A18}$$

$$Rec^{micro} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} TP_i + FN_i} \tag{A19}$$

where N is the number of available classes.

Appendix C.3. Balanced Accuracy

BA is one of the most common multi-class performance measures used when class imbalance is present. It is obtained as follows:

$$BA = \frac{1}{N}\sum_{i=1}^{N} Rec_i, \tag{A20}$$

where Rec_i stands for the recall obtained for class i ∈ N. Hence, the *BA* is the same as the Rec^{macro} [60]. This function takes values between $BA \in [0, 1]$, where a perfect classifier attains $BA = 1$, while $BA = 0$ is obtained by a classifier always predicting the wrong class.

Appendix C.4. F1-Measure

Another metric obtained from Equations (A14) and (A15) is the F1-score. This assesses a model's performance using the harmonic mean of the precision and recall measures. As in the case of Prec and Rec, in the multi-class setting, this is obtained via macro or micro averaging. Thus, the F1-score for both averaging approaches is defined as follows:

$$F1^{macro} = 2.\frac{Rec^{macro}.Prec^{macro}}{Rec^{macro}.Prec^{macro}} \tag{A21}$$

$$F1^{micro} = 2.\frac{Rec^{micro}.Prec^{micro}}{Rec^{micro}.Prec^{micro}} \tag{A22}$$

In both cases, the *F*1-function takes values between $F1 \in [0, 1]$. More specifically, in the case of Equation (A21), high values indicate that the algorithm has a good performance in all the classes, whereas low values indicate that all classes are predicted poorly. In contrast, Equation (A22) gives more importance to populous classes, neglecting errors in classes with few samples [60]. Consequently, $F1^{macro}$ is a better metric than $F1^{micro}$ for assessing the performance of classifiers in the presence of class imbalance.

Appendix C.5. Matthews' Correlation Coefficient

A more recent metric, which is more robust to class imbalance, is the MCC. This can be formally defined as follows:

$$MCC = \frac{cxs - \sum_{i=1}^{N} p_i x t_i}{\sqrt{\left(s^2 - \sum_{i=1}^{N} p_i^2\right)\left(s^2 - \sum_{i=1}^{N} t_i^2\right)}} \tag{A23}$$

where $c = \Sigma^{N}{}_{i=1}\, TP_i$ corresponds to the TP for class i, s is the total number of samples, $p_i = TPi + FPi$ is the sum of times the classifier predicted class i (either correctly or incorrectly classified), and $ti = TPi + FNi$ is the total number of samples for which class i appears in the dataset.

MCC values ranged from MCC $\in [-1, 1]$. For higher positive values (MCC ≈ 1), the MCC indicates a strong positive correlation between predictions and the true labels. On the contrary, large negative values (MCC < 0) indicate that the classifier identifies the classes but systematically predicts them wrong (rather as consequence of an implementation problem). If the classifier is randomly guessing, there will be no correlation between predictions and true labels; hence, the MCC is ($\approx$0). Another virtue of the MCC is that it is sensitive to classes imbalance; hence, it is good for measuring a classifier's performance when such a feature is present in the dataset [60].

Appendix C.6. Cohen's Kappa

The last performance measure is κ. This metric was initially devised as a statistical hypothesis test for quantifying the level of agreement between two raters on a nominal scale [60,62]. This metric has been extended for binary and multi-classification problems. In the case of the latter, the κ function is defined as follows:

$$K = \frac{cxs - \sum_{i=1}^{N} p_i x t_i}{s^2 - \sum_{i=1}^{N} p_i x t_i} \tag{A24}$$

where, similar to the MCC metric, $c = \sum_{i=1}^{N} TPi$ corresponds to the TP for class i, s is the total number of samples, $p_i = TP_i + FP_i$ is the sum of times the classifier predicted class i (either correctly or incorrectly classified), and $t_i = TP_i + FN_i$ is the total number of samples for which class i appears in the dataset. Thus, both the MCC and the κ are highly related, providing slightly larger values for κ in comparison to the MCC. Cohen's kappa value ranges from $\kappa \in [-1, 1]$, where $\kappa = 1$ points to a perfect agreement between classifier predictions and true labels, $\kappa = 0$ points to random agreement attained due to the independence between predictions and the actual labels, and $\kappa < 0$ points to the classifier performing worse than a random classifier. Some key advantages of κ are that (i) it measures the dependency between model predictions and the true class distribution; (ii) it is useful for assessing classifiers' performance when class imbalance is present; and (iii) it can be used to compare classifiers' performance on two different datasets (e.g., two TFD datasets with different faults and/or severity details).

References

1. Senoussaoui, M.E.A.; Brahami, M.I.; Fofana, I. Combining and comparing various machine learning algorithms to improve dissolved gas analysis interpretation. *IET Gener. Transm. Distrib.* **2018**, *12*, 3673–3679. [CrossRef]
2. Taha, I.B.; Dessouky, S.S.; Ghoneim, S.S. Transformer fault types and severity class prediction based on neural pattern-recognition techniques. *Electr. Power Syst. Res.* **2020**, *191*, 106899. [CrossRef]
3. Baker, E.; Nese, S.V.; Dursun, E. Hybrid Condition Monitoring System for Power Transformer Fault Diagnosis. *Energies* **2023**, *16*, 1151. [CrossRef]
4. Velasquez, R.M.A.; Lara, J.V.M. Root cause analysis improved with machine learning for failure analysis in power transformers. *Eng. Fail. Anal.* **2020**, *115*, 104684. [CrossRef]
5. Xu, Y.; Li, Y.; Wang, Y.; Zhong, D.; Zhang, G. Improved few-shot learning method for transformer fault diagnosis based on approximation space and belief functions. *Expert Syst. Appl.* **2021**, *167*, 114105. [CrossRef]
6. Duque, J.F.P.; Martinez, M.T.V.; Hurtado, A.P.; Carrasco, E.M.; Sancho, B.L.; Krommydas, K.F.; Plakas, K.A.; Karavas, C.G.; Kurashvili, A.S.; Dikaiakos, C.N.; et al. Inter-Area Oscillation Study of the Greek Power System Using an Automatic Toolbox. In Proceedings of the IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe), Espoo, Finland, 18–21 October 2021; pp. 1–6.
7. Arias, R.; Mejia, J. Health index for transformer condition assessment. *IEEE Lat. Am. Trans.* **2018**, *16*, 2843–2849. [CrossRef]
8. Ghoneim, S.S.M.; Taha, I.B.M. Comparative study of full and reduced feature scenarios for health index computation of power transformers. *IEEE Access* **2020**, *8*, 181326–181339. [CrossRef]
9. Rogers, R. IEEE and IEC Codes to Interpret Incipient Faults in Transformers, Using Gas in Oil Analysis. *IEEE Trans. Electr. Insul.* **1978**, *13*, 349–354. [CrossRef]

10. CIGRE. *Transformer Reliability Surveys*; CIGRE Technical Brochure 642; WW.G. A2.; CIGRE: Paris, France, 2015.
11. Bartley, W. Analysis of transformer failures. In Proceedings of the International Association OF Engineering Insurers 36th Annual Conference, Stockholm, Sweden, 15–17 September 2003; pp. 1–12.
12. Nagpal, T.; Brar, Y.S. Artificial neural network approaches for fault classification: Comparison and performance. *Neural Comput. Appl.* **2014**, *25*, 1863–1870. [CrossRef]
13. Mirowski, P.; Lecun, Y. Statistical machine learning and dissolved gas analysis: A review. *IEEE Trans. Power Deliv.* **2012**, *27*, 1791–1799. [CrossRef]
14. Golarz, J. Understanding Dissolved Gas Analysis (DGA) techniques and interpretations. In Proceedings of the IEEE Power Engineering Society Transmission and Distribution Conference, Dallas, TX, USA, 3–5 May 2016.
15. Wu, Q.; Zhang, H. A novel expertise-guided machine learning model for internal fault state diagnosis of power transformers. *Sustainability* **2019**, *11*, 1562. [CrossRef]
16. Li, E.; Wang, L.; Song, B. Fault diagnosis of power transformers with membership degree. *IEEE Access* **2019**, *7*, 28791–28798. [CrossRef]
17. Cheim, L.; Duval, M.; Haider, S. Combined duval pentagons: A simplified approach. *Energies* **2020**, *13*, 2859. [CrossRef]
18. Wang, J.; Zhang, X.; Zhang, F.; Wan, J.; Kou, L.; Ke, W. Review on evolution of intelligent algorithms for transformer condition assessment. *Front. Energy Res.* **2022**, *10*, 904109. [CrossRef]
19. Shintemirov, A.; Tang, W.; Wu, Q.H. Power transformer fault classification based on dissolved gas analysis by implementing bootstrap and genetic programming. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2009**, *39*, 69–79. [CrossRef]
20. Wu, X.; Wang, P.; Wang, L.; Xu, Y.; Zhao, Z. Transformer combination weighting evaluation model based on bp neural network. In Proceedings of the Genetic and Evolutionary Computing. ICGEC 2021. Lecture Notes in Electrical Engineering, Jilin, China, 21–23 October 2022.
21. Mortada, M.A.; Yacout, S.; Lakis, A. Fault diagnosis in power transformers using multi-class logical analysis of data. *J. Intell. Manuf.* **2014**, *25*, 1429–1439. [CrossRef]
22. Yang, X.; Chen, W.; Li, A.; Yang, C.; Xie, Z.; Dong, H. BA-PNN-based methods for power transformer fault diagnosis. *Av. Eng. Inform.* **2019**, *39*, 178–185. [CrossRef]
23. AI, K.; PV, M.; SA, E.; VZ, M.; AM, B. Romanov AM. Data Mining Applied to Decision Support Systems for Power Transformers' Health Diagnostics. *Mathematics* **2022**, *10*, 2486.
24. Zhang, D.; Li, C.; Shahidehpour, M.; Wu, Q.; Zhou, B.; Zhang, C. A bi-level machine learning method for fault diagnosis of oil-immersed transformers with feature explainability. *Int. J. Electr. Power Energy Syst.* **2022**, *134*, 107356. [CrossRef]
25. Cheim, L. *Machine Learning Tools in Support of Transformer Diagnostics*; CIGRE: Paris, France, 2018; pp. A2–A206.
26. Zöller, M.-A.; Huber, M.F. Benchmark and Survey of Automated Machine Learning Frameworks. *J. Artif. Intell. Res.* **2021**, *70*, 409–472. [CrossRef]
27. Thornton, C.; Hutter, F.; Hoos, H.H.; Leyton-Brown, K. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11–14 August 2013; Part F128815. pp. 847–855.
28. Feurer, M.; Klein, A.; Eggensperger, K.; Springenberg, J.T.; Blum, M.; Hutter, F. Efficient and Robust Automated Machine Learning. In Proceedings of the 28th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 2962–2970.
29. Duval, M.; Lamarre, L. The Duval pentagon-a new complementary tool for the interpretation of dissolved gas analysis in transformers. *IEEE Electr. Insul. Mag.* **2014**, *30*, 9–12.
30. Jakob, F.; Dukarm, J.J. Thermodynamic estimation of transformer fault severity. *IEEE Trans. Power Deliv.* **2015**, *30*, 1941–1948. [CrossRef]
31. Dukarm, J.; Jakob, F. Thermodynamic estimation of transformer fault severity. In Proceedings of the 2016 IEEE/PES Transmission and Distribution Conference and Exposition (T&D), Dallas, TX, USA, 3–5 May 2016. [CrossRef]
32. Londono, S.M.P.; Cadena, J.A.; Cadena, J.M. Aplicacion de redes neuronales probabilısticas en la deteccion de fallas incipientes en transformadores. *Sci. Et Tech.* **2008**, *2*, 48–53.
33. Ranga, C.; Chandel, A.K.; Chandel, R. Condition assessment of power transformers based on multi-attributes using fuzzy logic. *IET Sci. Meas. Technol.* **2017**, *11*, 983–990. [CrossRef]
34. Mharakurwa, E.T.; Goboza, R. Multiparameter-based fuzzy logic health index assessment for oil-immersed power transformers. *Adv. Fuzzy Syst.* **2019**, *2019*, 2647157. [CrossRef]
35. *CIGRE TB 761 Condition Assessment of Power Transformers*; CIGRE: Paris, France, 2019.
36. *IEEE Guide for the Interpretation of Gases Generated in Mineral Oil-Immersed Transformers*; IEEE: New York, NY, USA, 2019.
37. Truong, A.; Walters, A.; Goodsitt, J.; Hines, K.; Bruss, C.B.; Farivar, R. Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools. In Proceedings of the IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), Portland, OR, USA, 4–6 November 2019; pp. 1471–1479.
38. Osborne, J. Notes on the use of data transformations, Practical assessment. *Res. Eval.* **2002**, *8*, 6.
39. Feurer, M.; Eggensperger, K.; Falkner, S.; Lindauer, M.; Hutter, F. Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning. *J. Mach. Learn. Res.* **2020**, *23*, 11936–11996.

40. 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.
41. Oliphant, T.E. *A Guide to NumPy*; Trelgol Publishing: New York, NY, USA, 2006; Volume 1.
42. McKinney, W. pandas: A foundational python library for data analysis and statistics. In Proceedings of the Workshop Python for High Performance and Scientific Computing, Tsukuba, Japan, 1–3 June 2011; pp. 1–9.
43. 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, KDD '16, ACM, New York, NY, USA, 13–17 August 2016; pp. 785–794.
44. Freitas, A.A. A critical review of multi-objective optimization in data mining. *ACM SIGKDD Explor. Newsl.* **2004**, *6*, 77–86. [CrossRef]
45. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning, Springer Series in Statistics*; Springer: New York, NY, USA, 2001.
46. Rokach, L.; Maimon, O.Z. *Data Mining with Decision Trees: Theory and Applications*; World Scientific: Singapore, 2007; Volume 69.
47. Matthews, A.G.D.G.; Rowland, M.; Hron, J.; Turner, R.E. Gaussian process behaviour in wide deep neural networks. *Int. Conf. Learn. Represent* **2018**, *4*, 77–86.
48. Mitchell, T.M.; Mitchell, T.M. *Machine Learning*; McGraw-Hill: New York, NY, USA, 1997.
49. Shobha, G.; Rangaswamy, S.; Chapter; Gudivada, V.N.; Rao, C. *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications, Vol. 38 of Handbook of Statistics*; Elsevier: Amsterdam, The Netherlands, 2018; pp. 197–228.
50. Benjamini, Y.; Leshno, M. Statistical methods for data mining. In *Data Mining and Knowledge Discovery Handbook*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 565–587.
51. Scholkopf, B.; Smola, A.J.; Bach, F. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*; MIT Press: Cambridge, MA, USA, 2002.
52. Vapnik, V.N. *Statistical Learning Theory*; Wiley-Interscience: Hoboken, NJ, USA, 1998.
53. Gunn, S.R. Support vector machines for classification and regression. *Analyst* **1998**, *135*, 230–267.
54. Biau, G.; Scornet, E. A random forest guided tour. *Test* **2016**, *25*, 197–227. [CrossRef]
55. Sutton, C.D. Classification and regression trees, bagging, and boosting. *Handb. Stat.* **2005**, *24*, 303–329.
56. Friedman, J.H. Stochastic gradient boosting; Computational statistics & data analysis. *Comput. Stat. Data Anal.* **2002**, *38*, 367–378.
57. Moon, J.; Jung, S.; Rew, J.; Rho, S.; Hwang, E. Combination of short-term load forecasting models based on a stacking ensemble approach. *Energy Build.* **2020**, *216*, 109921. [CrossRef]
58. He, X.; Zhao, K.; Chu, X. AutoML: A survey of the state-of-the-art, Knowledge-Based Systems 212 (Dl). *arXiv* **2021**, arXiv:1908.00709.
59. Kautz, T.; Eskofier, B.M.; Pasluosta, C.F. Generic performance measure for multiclass-classifiers. *Pattern Recognit.* **2017**, *68*, 111–125. [CrossRef]
60. Grandini, M.; Bagli, E.; Visani, G. Metrics for multi-class classification: An overview. *arXiv* **2020**, arXiv:2008.05756.
61. Markoulidakis, I.; Rallis, I.; Georgoulas, I.; Kopsiaftis, G.; Doulamis, A.; Doulamis, N. Multiclass confusion matrix reduction method and its application on net promoter score classification problem. *Technologies* **2021**, *9*, 81. [CrossRef]
62. Warrens, M.J. Five ways to look at cohen's kappa. *J. Psychol. Psychother.* **2015**, *5*, 1. [CrossRef]