

Article

Robustness of Machine Learning and Deep Learning Models for Power Quality Disturbance Classification: A Cross-Platform Analysis

José Carlos Palomares-Salas * , Sergio Aguado-González  and José María Sierra-Fernández 

Research Group PAIDI-TIC-168: Computational Instrumentation and Industrial Electronics (ICEI), Department of Automation Engineering, Electronics, Architecture and Computers Networks, University of Cádiz, 11202 Algeciras, Spain; sergio.aguadogonzalez@alum.uca.es (S.A.-G.); josemaria.sierra@uca.es (J.M.S.-F.)

* Correspondence: josecarlos.palomares@uca.es

Abstract

Accurate and robust power quality disturbance (PQD) classification is critical for modern electrical grids, particularly in noisy environments. This study presents a comprehensive comparative evaluation of machine learning (ML) and deep learning (DL) models for automatic PQD identification. The models evaluated include Support Vector Machines (SVM), Decision Trees (DT), Random Forest (RF), k-Nearest Neighbors (kNN), Gradient Boosting (GB), and Dense Neural Networks (DNN). For experimentation, a hybrid dataset, comprising both synthetic and real signals, was used to assess model performance. The robustness of the models was evaluated by systematically introducing Gaussian noise across a wide range of Signal-to-Noise Ratios (SNRs). A central objective was to directly benchmark the practical implementation and performance of these models across two widely used platforms: MATLAB R2024a and Python 3.11. Results show that ML models achieve high accuracies, exceeding 95% at an SNR of 10 dB. DL models exhibited remarkable stability, maintaining 97% accuracy for SNRs above 10 dB. However, their performance degraded significantly at lower SNRs, revealing specific confusion patterns. The analysis underscores the importance of multi-domain feature extraction and adaptive preprocessing for achieving resilient PQD classification. This research provides valuable insights and a practical guide for implementing and optimizing robust PQD classification systems in real-world, noisy scenarios.

Keywords: power quality; disturbances; machine learning; deep learning; higher-order statistics



Academic Editor: Alessandro Lo Schiavo

Received: 29 August 2025

Revised: 20 September 2025

Accepted: 25 September 2025

Published: 30 September 2025

Citation: Palomares-Salas, J.C.; Aguado-González, S.; Sierra-Fernández, J.M. Robustness of Machine Learning and Deep Learning Models for Power Quality Disturbance Classification: A Cross-Platform Analysis. *Appl. Sci.* **2025**, *15*, 10602. <https://doi.org/10.3390/app151910602>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The electrical grid is facing new challenges due to the expansion of renewable energies, the proliferation of electric vehicles, and the incorporation of storage systems. As a result, the growing demand for high-quality electrical energy in industrial and domestic applications underscores the importance of accurate monitoring and diagnosis of power grid disturbances, which has generated growing interest in grid improvement studies in recent years.

Power quality refers to the purity and stability of electrical current [1]. A high-quality power supply is essential to ensure the efficiency, safety, and durability of electrical and electronic equipment. Power Quality Disturbances (PQDs), such as voltage sags, harmonics, or transients, can cause equipment failures, increase operating costs, and reduce energy

efficiency. Therefore, the development of automated and efficient classification systems for the detection and categorization of these anomalies is crucial [2–4].

In this context, Machine Learning (ML) and Deep Learning (DL) methodologies have become powerful tools for the analysis and classification of complex signals [5–9]. Their ability to extract complex patterns from large datasets makes them ideal candidates for solving PQDs classification challenges. However, existing studies in this field often present key limitations. Many high-performance models, particularly end-to-end learning architectures, act as “black boxes”, lacking the necessary interpretability for engineering applications where understanding physical phenomena is essential. Moreover, the reliance on purely real-world datasets poses a significant practical challenge, as a wide variety of PQD labeled with different levels of noise is extremely scarce and difficult to obtain.

This study directly addresses these limitations by providing a fundamental cross-platform benchmark of various ML and DL algorithms. Our approach applies and compares a diverse set of models using controlled hybrid datasets, which combine a validated real signal with synthetic perturbations and Gaussian noise to ensure reproducibility and systematic evaluation. The input features for classification include higher-order statistical (HOS) estimators, frequency-domain features, and features derived from the envelope and the derivative of the signal.

Specifically, this work makes the following original contributions:

- It presents a direct comparative analysis of the classification capabilities of various ML models (Support Vector Machines, Decision Trees, Random Forest, k-Nearest Neighbors, Gradient Boosting) and DL (Dense Neural Networks) across a wide range of noise conditions, providing a valuable performance benchmark.
- Examine the impact of different noise levels on model accuracy, specifically focusing on the robustness and noise resilience of each algorithm.
- Analyze and contrast the practical implications of implementing these models on two widely used platforms, MATLAB [10] and Python [11,12], with special attention to their suitability for real PQD classification tasks.

The rest of the paper is structured as follows. Section 2 describes the signal processing methodology, including data generation, the addition of Gaussian noise, and feature extraction. Section 3 provides a brief description of the ML and DL models employed. Sections 4 and 5 present a critical analysis and a discussion of the experimental results, which also suggest future research directions. Finally, Section 6 concludes the article.

2. Materials and Methods

2.1. Data Generation

The main power quality disturbances addressed in this study are defined according to the IEEE 1159-2019 standard [13]. Figure 1 illustrates these disturbances, which were generated as 1-s signals from an ideal function. The specific parameters of the samples used in the classifier are detailed in Table 1.

For our evaluation methodology, we opted for a hybrid dataset because purely real-world datasets containing a wide variety of labeled PQs and covering a broad range of Signal-to-Noise Ratios (SNRs) are extremely scarce and difficult to obtain. This approach, which combines a validated, anomaly-free real-world signal with synthetic disturbances and Gaussian noise, allowed us to create a repeatable and verifiable testbed. This was crucial for our main objective: to evaluate the robustness of various machine learning and deep learning models under precisely defined conditions.

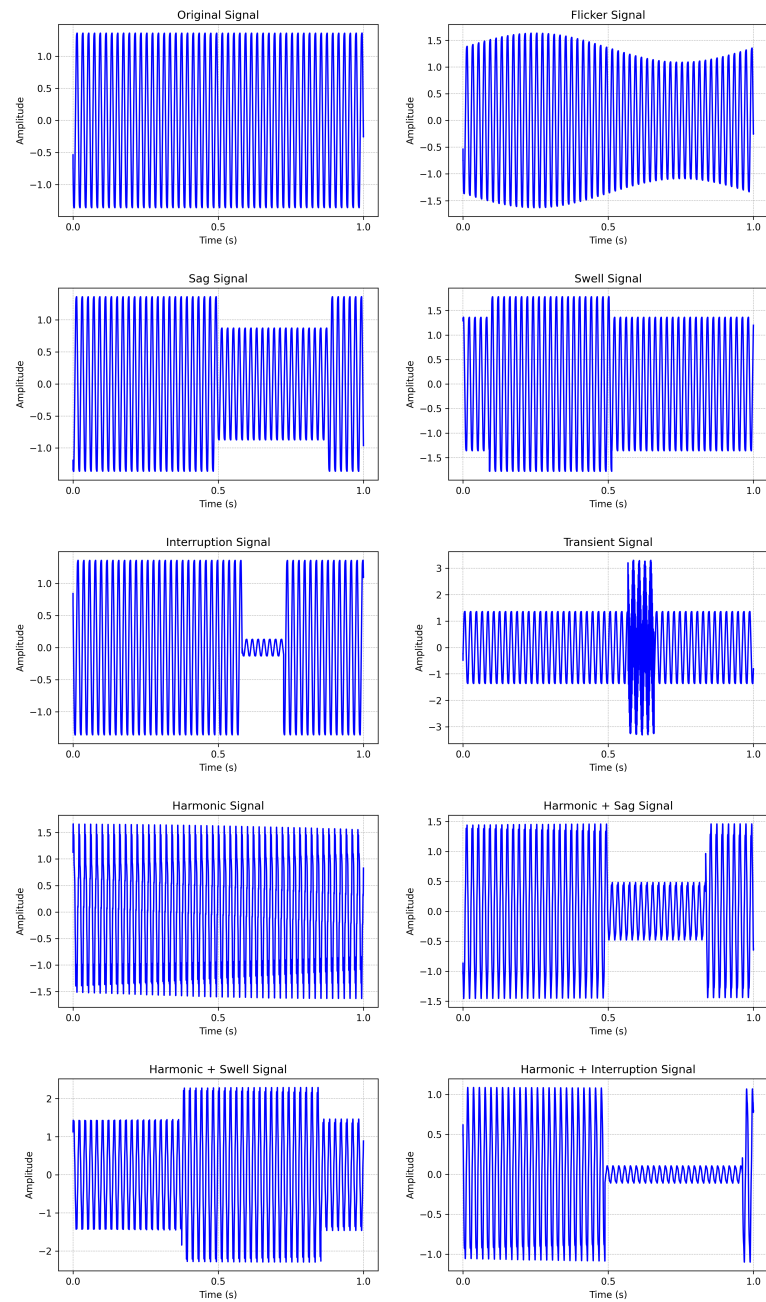


Figure 1. PQDs represented in this article.

Table 1. PQDs used in this article.

PQD	Number of Samples
Healthy signal	10,800
Interruption	
Sag	
Swell	
Flicker	
Transient	
Harmonic	
Harmonic + Sag	
Harmonic + Swell	
Harmonic + Interruption	

To generate the power quality disturbances, we began by acquiring three hours of undisturbed reference signals from the *TIC168* database at the University of Cadiz. After a rigorous validation to confirm the absence of anomalies, we randomly injected disturbances in terms of time and type, strictly following the parameters of the IEEE standards. This procedure ensured the generation of an identical number of disturbed and undisturbed signals, thereby guaranteeing a balanced and robust dataset for comparative analysis.

2.2. Random Noise

To enhance the robustness of the PQD classifier, random Gaussian noise was added to 80% of the samples before feature extraction. The noise was introduced with Signal-to-Noise Ratios (SNRs) ranging from 40 dB to 1 dB, following the sequence below:

- Instantaneous signal power

$$x_{watts}[n] = signal[n]^2 \quad (1)$$

where $signal[n]^2$ is the signals amplitude at instant n .

- Average signal power

$$P_{signal} = \frac{1}{N} \cdot \sum_{n=0}^{N-1} x_{watts}[n] \quad (2)$$

where N are the samples of each signal and P is the average signal power in watts.

- Average signal power in decibels

$$P_{signal,dB} = 10 \cdot \log_{10}(P_{signal} + \epsilon) \quad (3)$$

where ϵ is a small value (1×10^{-9}) to avoid logarithm of zero.

- Average noise power in decibels

$$P_{noise,dB} = P_{signal,dB} - SNR_{obj,dB} \quad (4)$$

where $SNR_{obj,dB}$ is the target signal-to-noise ratio in decibels.

- Average noise power in watts

$$P_{noise} = 10^{\frac{P_{noise,dB}}{10}} \quad (5)$$

- Noise standard deviation

$$\sigma_{noise} = \sqrt{P_{noise}} \quad (6)$$

- Gaussian noise generation

$$N_{volts}[n] \sim N(0, \sigma_{noise}) \quad (7)$$

The noise is drawn from a normal distribution, characterize by a mean of $\mu = 0$ and a standard deviation of σ_{noise} .

- Noisy signal

The noisy signal is obtained by adding the generated noise to the original signal, as follows:

$$y_{noisy}[n] = signal[n] + N_{volts}[n] \quad (8)$$

2.3. Feature Extracted from PQDs

To optimize the classification of PQDs, this experiment incorporated a set of multidomain features. Key information was extracted from the time, frequency, envelope, and derivative domains of the signals. This integration of various features enriched the

model with a multidimensional data representation, significantly improving its ability to accurately identify and classify PQDs.

2.3.1. Time Domain Features

High Order Statistics (HOS)

Statistical features have been widely used over the years to detect trends in the behavior of signal of different natures. Higher-order statistics (HOS) have gained popularity due to their effectiveness when signals exhibit non-Gaussian noise. Higher-order cumulants are mathematical tools that allow the analysis of properties of signals that do not follow a normal (Gaussian) distribution. In signal processing, the following formula is commonly used to calculate the r -th order cumulants of a set of signals $\{x_i\}_{i \in [1,r]}$ [14,15]:

$$\text{Cum}(x_1, \dots, x_r) = \sum (-1)^{p-1} \cdot (p-1)! \cdot E\{\prod_{i \in S_1} x_i\} \cdot E\{\prod_{i \in S_2} x_j\} \dots E\{\prod_{i \in S_p} x_k\} \quad (9)$$

where the summation extends to all possible partitions of the indices, E denotes the expected value, and S_i are the sets that make up the partition. For a stationary signal $x(t)$ of order r , the r -th order cumulant is defined as:

$$C_{r,x}(\tau_1, \tau_2, \dots, \tau_{r-1}) = \text{Cum}[x(t), x(t + \tau_1), \dots, x(t + \tau_{r-1})] \quad (10)$$

For the zero-mean signal $x(t)$, the second, third, and fourth-order cumulants are expressed by:

$$C_{2,x}(\tau) = E\{x(t) \cdot x(t + \tau)\} \quad (11)$$

$$C_{3,x}(\tau_1, \tau_2) = E\{x(t) \cdot x(t + \tau_1) \cdot x(t + \tau_2)\} \quad (12)$$

$$\begin{aligned} C_{4,x}(\tau_1, \tau_2, \tau_3) &= E\{x(t) \cdot x(t + \tau_1) \cdot x(t + \tau_2) \cdot x(t + \tau_3)\} \\ &\quad - C_{2,x}(\tau_1) \cdot C_{2,x}(\tau_2 - \tau_3) \\ &\quad - C_{2,x}(\tau_2) \cdot C_{2,x}(\tau_3 - \tau_1) \\ &\quad - C_{2,x}(\tau_3) \cdot C_{2,x}(\tau_1 - \tau_2). \end{aligned} \quad (13)$$

If $\tau_1 = \tau_2 = \tau_3 = 0$, the equations above simplify to:

$$\gamma_{2,x} = E\{x^2(t)\} = C_{2,x}(0) \quad (14)$$

$$\gamma_{3,x} = E\{x^3(t)\} = C_{3,x}(0,0) \quad (15)$$

$$\gamma_{4,x} = E\{x^4(t)\} - 3(\gamma_{2,x})^2 = C_{4,x}(0,0,0). \quad (16)$$

These last expressions represent variance, skewness, and kurtosis, respectively. Normalized skewness and kurtosis can be defined as $\gamma_{(3,x)} / (\gamma_{(2,x)})^2$ and $\gamma_{(4,x)} / (\gamma_{(2,x)})^2$, respectively. Normalizing these measures is useful, since they remain unchanged regardless of the signal's scale or shift.

Total Harmonic Distortion (THD)

Total harmonic distortion is defined as the ratio between the Root Mean Square (RMS) value of the harmonic components and the RMS value of the fundamental component of the signal. This definition universally applies to both current and voltage signals [16].

$$THD_i = \frac{\sqrt{\sum_{n=2}^{\infty} I_n^2}}{I_1} \quad (17)$$

$$THD_V = \frac{\sqrt{\sum_{n=2}^{\infty} V_n^2}}{V_1} \quad (18)$$

where THD_i and THD_V are the total harmonic distortions for current and voltage, respectively. The terms I_n and V_n denote the current and voltage components of the n -th harmonic, while I_1 and V_1 represent their corresponding fundamental components.

Crest Factor

The crest factor, defined as the ratio between the peak value of a waveform and its RMS value, is a crucial parameter for analyzing low-frequency signals. It is calculated to quantify the influence of low frequencies on a signal. For an ideal sinusoidal waveform, the crest factor is $CF = \sqrt{2}$ [17].

$$V_{RMS} = \sqrt{\frac{\sum_{i=1}^N x^2 \cdot n_i}{N}} \quad (19)$$

$$CF = \frac{V_p}{V_{RMS}} \quad (20)$$

where V_p and V_{RMS} represent the peak and RMS values of the signal, respectively, and N is the number of samples.

2.3.2. Frequency Domain Features

The frequency domain, accessed through the Fourier Transform, offers a critical representation of a signal as the composition of its sinusoidal components. The characteristics derived from this domain are particularly effective for analyzing certain disturbances whose defining features are more pronounced in their frequency content [18].

Frequency Magnitudes

The Fast Fourier Transform (FFT) converts a signal from the time domain into a representation in the frequency domain. The resulting complex coefficients reveal the amplitude and phase of each frequency component present in the original signal. In a 50 Hz power grid, harmonic disturbances are characterized by the presence of energy at integer multiples of the fundamental frequency. Consequently, the detection and quantification of these disturbances can be achieved by identifying and measuring the magnitudes of these harmonic peaks in the frequency spectrum.

Frequency Band Energy

Band energy offers a complementary approach to the analysis of individual frequencies, as it aggregates the total energy within specific frequency segments, which are subsequently normalized to obtain proportional contributions. This metric is especially effective for identifying transients in high-frequency bands and flickers in low-frequency bands, given their characteristic energy concentrations in these regions.

THD in Frequency Domain

Parallel to the analyzes in the time domain, (THD) in the frequency domain provides a comprehensive measure of waveform distortion by summing the magnitudes of its harmonic components. A high THD value indicates a presence of harmonics in the signal, allowing for effective detection and quantitative evaluation of these specific disturbances.

$$THD_V = \frac{\sqrt{\sum_{n=2}^{\infty} X_n^2}}{X_1} \quad (21)$$

2.3.3. Wavelet Features

The wavelet transform is frequently used in PQD classification due to its ability to provide multiresolution analysis, which facilitates the simultaneous extraction of features from the temporal and frequency domains of the signal [19]. In this research, feature extraction was performed using the PyWavelets library [20] and the ‘db4’ wavelet, generating a set of 10 features. The main features obtained from the wavelet coefficients at each level are the mean and standard deviation.

$$s = \frac{\sqrt{\sum_{i=1}^N (x_i - \bar{X})^2}}{N} \quad (22)$$

2.3.4. Envelope Features

The envelope of a signal provides a representation of its amplitude varying over time. A widely used technique to calculate this envelope is the Hilbert transform, which transforms a real signal into its analytic counterpart. The Hilbert transform, $H(x(t))$ [18], of a real function $x(t)$ is defined as:

$$H(x(t)) = \frac{1}{\pi} \cdot \int_{-\infty}^{\infty} \frac{x(\tau)}{t - \tau} d\tau \quad (23)$$

Envelope features are valuable for analyzing variations in signal amplitude.

Mean of the Envelope

The mean of the signal envelope quantifies its average amplitude in a given sample. For slow variations, this metric can effectively indicate the presence of voltage sags or swells.

Standard Deviation of the Envelope

The standard deviation of the envelope quantifies the magnitude of the amplitude variations around its mean. This characteristic is especially useful for measuring flicker, as its repeated characteristic fluctuations directly result in a high standard deviation.

Maximum of the Envelope

By capturing the peak amplitude of the signal, the maximum value of the envelope serves as an effective feature for identifying disturbances such as flickers and voltage swells.

2.3.5. Derivative Features

The derivative of a signal, which serves as a measure of the instantaneous rate of change, is fundamental for identifying rapid and directional changes. Therefore, it is an effective feature for detecting transient events. The derivative of a continuous signal $x(t)$ is defined as:

$$\frac{dx(t)}{dt} = \lim_{\Delta t \rightarrow 0} \frac{x(t - \Delta t) - x(t)}{\Delta t} \quad (24)$$

In the case of electrical signals, the data is collected discretely, in this case with a sampling frequency of 10 KHz.

Maximum Derivative

This characteristic quantifies the fastest and most pronounced change within a signal. An high value serves as a reliable indicator of transients, given its defining characteristic as abrupt changes of large magnitude in the waveform.

Mean of the Absolute Derivative

Although the maximum derivative indicates the most extreme rate of change at a single point, the mean of the absolute derivative provides a more comprehensive measure by calculating the average rate of change over the entire segment of the signal. This makes it specially effective for characterizing events that involve widespread and rapid fluctuations, such as impulses and high-frequency noise.

3. Power Quality Disturbance Classifiers

Figure 2 provides a generalized algorithm detailing the multi-step process involved in the classification of power quality (PQ) events. This study discusses various algorithms and approaches based on machine learning and deep learning (AI), focusing on their application for the classification and detection of disturbances within a power system.

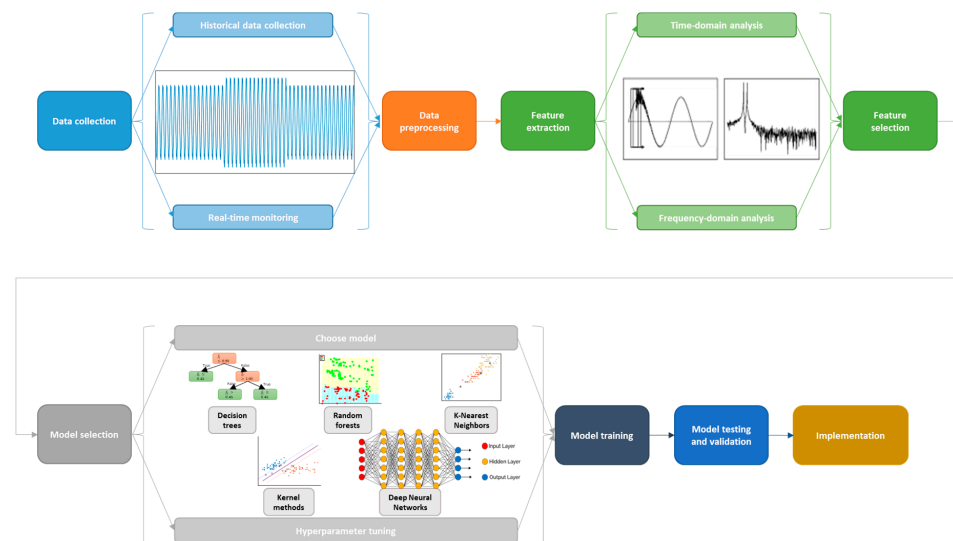


Figure 2. General algorithm for classifying PQ disturbances/events.

3.1. Machine Learning-Based Classifiers

For the classification task, we selected a set of representative and high-performing Machine Learning (ML) models: Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), K-Nearest Neighbours (kNN), and Gradient Boosting (GB). These algorithms are well-suited for feature-based data and serve as a robust benchmark due to their proven effectiveness across a wide range of classification problems.

3.1.1. Support Vector Machine (SVM)

A SVM model was utilized in this study for PQD classification, as it is a model well-suited for handling high-dimensional and nonlinear data. To optimize its performance, the key hyperparameters, C and Gamma, are tuned to find the best balance between a smooth decision boundary and accurate classification of the training data [17].

3.1.2. Decision Tree (DT)

A DT algorithm uses a recursive process to classify data [21]. It works in four main steps: first, it selects the best feature to split the dataset using a criterion like the Gini index. Second, it splits the data into subsets based on that feature. Third, it repeats this process to grow the tree until a stop condition is met. Finally, it “prunes” the tree by cutting branches to prevent the model from overfitting.

3.1.3. Random Forest (RF)

By combining the outputs of multiple independent decision trees, the RF algorithm improves overall predictive performance and reduces the risk of overfitting. The process involves three main steps: first, it creates multiple training datasets through bootstrapping (random sampling with replacement). Next, it trains a separate decision tree for each of these subsets, using a random selection of characteristics to ensure diversity among trees. Finally, the model aggregates the predictions of all individual trees, either averaging them for regression or using a majority vote for classification, to produce the final more robust output [22].

3.1.4. K-Nearest Neighbors (kNN)

Operating on the principle that similar data points are close to each other, the kNN algorithm is a non-parametric classifier [23]. It assigns a label to a new data point based on the majority class of its nearest neighbors in the training set, often determined by the Euclidean distance. This makes kNN a robust choice for datasets with complex, non-linear relationships, particularly when the dataset is not excessively large.

3.1.5. Gradient Booster (GB)

The GB algorithm is a powerful and flexible method to build a predictive model by combining a sequence of simple “weak” learners, typically regression trees. At each step, a new tree is trained to correct the residual errors of the existing ensemble. This is achieved by fitting the new tree to the negative gradient of a chosen differentiable loss function, effectively performing a form of gradient descent in the function space to minimize the overall prediction error [24].

For multi-class classification, a separate regression tree is trained for each class to minimize the loss. Binary classification is a simpler case where only a single regression tree is used per stage.

3.2. Deep Learning-Based Classifiers

For the DL comparison, a Dense Neural Network (DNN) was selected as the foundational architecture due to its ability to learn complex patterns directly from the input features, providing a crucial contrast to the traditional ML models.

The exclusion of other architectures, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) like LSTMs, was based on the scope of this study. Our methodology relies on a multi-domain feature extraction process, transforming the raw time-series signal into a feature vector. The chosen algorithms are well-suited for this feature-based, tabular data format. In contrast, CNNs are primarily designed to handle spatial hierarchies (like images) and local patterns in raw time-series data, while LSTMs are specialized for sequential data where the order is a critical factor.

Dense Neural Networks (DNNs)

Dense Neural Networks feature a fully connected architecture where each neuron in a given layer is connected to every neuron in the preceding layer, allowing for extensive information transfer (Figure 3). This structure is particularly effective for problems that require significant learning capacity, such as natural language processing, image recognition, and signal classification [25].

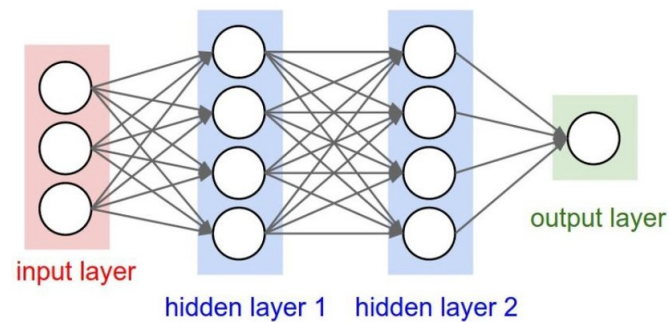


Figure 3. Dense Neural Network Architecture.

4. Results

The experiments were carried out on a robust computational setup consisting of an Apple M2 Pro chip, 12-core CPU with 8 performance cores and 4 efficiency cores, with a total of 19-core GPU and 16-core Neural Engine, and 32 GB of unified memory. This hardware provided the significant resources required for the analysis. Leveraging both MATLAB and Python, this study facilitated a direct comparison of the latest features and optimizations for algorithm development and data analysis across the two platforms.

The classifier's robustness was assessed by evaluating its performance at Signal-to-Noise Ratios (SNRs) ranging from 40 dB to 1 dB. This analysis aims to determine the specific threshold at which the model's accuracy significantly degrades in noisy environments.

The dataset was carefully partitioned to ensure a fair and reliable evaluation.

For all Machine Learning (ML) models, we adopted a standard 80/20 data split. The training set (80%) was used for both model fitting and hyperparameter tuning, which was performed using k-fold cross-validation to prevent data leakage. The testing set (20%) was held out and used exclusively for the final, unbiased evaluation of the model's performance on unseen data.

For the Deep Learning (DL) models, a three-way split was implemented to account for the need for continuous performance monitoring. The dataset was divided into a 70% training set, a 15% validation set, and a 15% independent testing set. The validation set was used to fine-tune the model's hyperparameters and prevent overfitting during the training process, while the testing set provided an unbiased final evaluation of the model's generalizability. Additionally, to ensure a fair evaluation and prevent class bias, all classes within the dataset contained an equal number of samples before the split.

The final performance of all models was then evaluated only on the independent test set, which was not used in any part of the training or validation process. By holding out this final test set, our results provide a reliable and unbiased measure of the models' ability to generalize to unseen data.

Model performance was evaluated using several key metrics to provide a comprehensive assessment. Accuracy measures the overall percentage of correct predictions, but it can be misleading in unbalanced datasets. Recall (or sensitivity) quantifies the model's ability to correctly identify all positive instances, which is crucial in applications where missing a positive case is costly. Finally, the F1-Score is used to provide a single, balanced metric that harmonizes precision and recall, offering a more reliable measure of performance, particularly in imbalanced datasets.

As part of our methodology, each algorithm was rigorously tuned on both MATLAB and Python to achieve its maximum performance. The specific hyperparameters that yielded the best results for each model are detailed in Table 2, providing a clear record for the precise replication of our findings.

Table 2. Summary of the hyperparameters specific to each classifier.

Hyperparameters	SVM	DT	RF	kNN	GB	DNN
C	100	--	--	--	--	--
activation	--	--	--	--	--	tanh
algorithm	--	--	--	auto	--	--
batch_size	--	--	--	--	--	32
bootstrap	--	--	True	--	--	--
break_ties	False	--	--	--	--	--
cache_size	200	--	--	--	--	--
categorical_features	--	--	--	--	from_dtype	--
ccp_alpha	--	0	0	--	--	--
class_weight	None	None	None	--	None	--
coef0	0	--	--	--	--	--
criterion	--	gini	gini	--	--	--
decision_function_shape	ovr	--	--	--	--	--
degree	3	--	--	--	--	--
dropout_rate	--	--	--	--	--	0,2
early_stopping	--	--	--	--	auto	--
epochs	--	--	--	--	--	50
gamma	scale	--	--	--	--	--
hidden_units	--	--	--	--	--	[256, 128, 64, 32]
l2_regularization	--	--	--	--	0.1	0.001
interaction_cst	--	--	--	--	None	--
kernel	rbf	--	--	--	--	--
leaf_size	--	--	--	30	--	--
learning_rate	--	--	--	--	0.1	--
loss	--	--	--	--	log_loss	categorical_crossentropy
max_bins	--	--	--	--	255	--
max_depth	--	None	None	--	None	--
max_features	--	None	sqrt	--	1	--
max_iter	−1	--	--	--	100	--
max_leaf_nodes	--	None	None	--	31	--
max_samples	--	--	None	--	--	--
metric	--	--	--	minkowski	--	--
metric_params	--	--	--	None	--	--
min_impurity_decrease	--	0	0	--	--	--
min_samples_leaf	--	1	1	--	20	--
min_samples_split	--	2	2	--	--	--
min_weight_fraction_leaf	--	0	0	--	--	--
monotonic_cst	--	--	None	--	None	--
n_estimators	--	--	100	--	--	--
n_iter_no_change	--	--	--	--	10	--
n_jobs	--	--	−1	None	--	--
n_neighbors	--	--	--	3	--	--
oob_score	--	--	False	--	--	--
optimizer	--	--	--	--	--	<keras.src.optimizers.adam.Adam object at 0x12c4b4510>
p	--	--	--	2	--	--
patience	--	--	--	--	--	10
probability	False	--	--	--	--	--
random_state	None	--	42	--	123	--
scoring	--	--	--	--	loss	--
shrinking	True	--	--	--	--	--
splitter	--	best	--	--	--	--

Table 2. Cont.

Hyperparameters	SVM	DT	RF	kNN	GB	DNN
<i>tol</i>	0.001	--	--	--	1.0×10^{-7}	--
<i>validation_fraction</i>	--	--	--	--	0.1	--
<i>verbose</i>	False	--	0	--	0	--
<i>warm_start</i>	--	--	False	--	False	--
<i>weights</i>	--	--	--	uniform	--	--

The results presented in Table 3 and Figure 4 demonstrate that all evaluated classifiers have a high level of robustness to noise. This is a significant finding, as noise can easily blur the boundaries between different types of power quality disturbances, making accurate classification challenging. Our results indicate that well-designed Machine Learning models are a viable and effective solution for classification tasks in noisy environments.

Table 3. Summary of classification results using Python.

Classifier	Accuracy	40 dB			20 dB			1 dB		
		Recall	F1-Score	Accuracy	Recall	F1-Score	Accuracy	Recall	F1-Score	Accuracy
SVM	0.9823	0.9823	0.9823	0.9785	0.9785	0.9784	0.6324	0.6324	0.6343	
DT	0.9652	0.9652	0.9652	0.9463	0.9463	0.9463	0.6792	0.6792	0.6770	
RF	0.9829	0.9829	0.9829	0.9727	0.9727	0.9727	0.7875	0.7875	0.7875	
KNN	0.9833	0.9833	0.9833	0.9798	0.9798	0.9798	0.5663	0.5663	0.5659	
GB	0.9872	0.9872	0.9872	0.9796	0.9796	0.9796	0.7884	0.7884	0.7876	
DNN	0.9825	0.9825	0.9825	0.9802	0.9802	0.9802	0.8407	0.8407	0.8438	

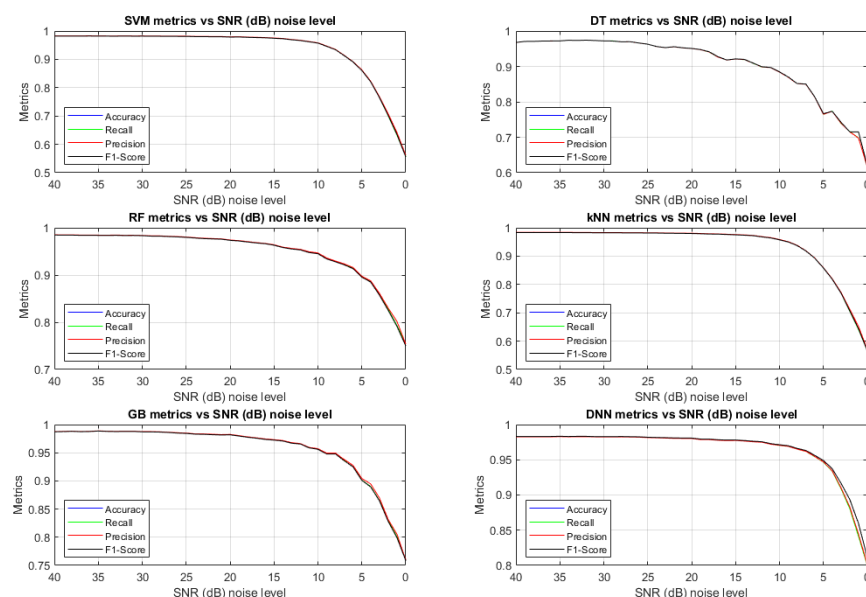


Figure 4. Performance of models implemented in Python under varying SNRs from 40 dB to 1 dB.

As shown in Table 4, the performance of the classifiers evaluated in MATLAB is comparable to those in Python. While MATLAB achieved slightly better results in some cases, this marginal difference can be attributed to the distinct optimization algorithms used in MATLAB's toolboxes compared to scikit-learn in Python. Both experiments were conducted using the same sample set, ensuring a fair comparison.

Table 4. Summary of classification results using MATLAB.

Classifier	40 dB			20 dB			1 dB		
	Accuracy	Recall	F1-Score	Accuracy	Recall	F1-Score	Accuracy	Recall	F1-Score
SVM	0.9838	0.9838	0.9838	0.9805	0.9804	0.9805	0.7926	0.7933	0.7926
DT	0.9750	0.9750	0.9750	0.9718	0.9718	0.9720	0.7426	0.7415	0.7430
RF	0.9849	0.9849	0.9849	0.9801	0.9801	0.9801	0.8206	0.8203	0.8206
KNN	0.9578	0.9577	0.9578	0.9281	0.9280	0.9281	0.3940	0.3967	0.3940
GB	0.9769	0.9768	0.9769	0.9739	0.9738	0.9739	0.7105	0.7125	0.7105
DNN	0.9842	0.9842	0.9842	0.9819	0.9819	0.9819	0.7725	0.7749	0.7725

The Gradient Boosting experiment necessitated different implementations in MATLAB and Python. The Python experiment employed the HistGradientBoostingClassifier [26], while MATLAB's lack of a direct equivalent led to the selection of the AdaBoostM2 algorithm from its documentation. Although these algorithms differ in their core mechanisms, AdaBoostM2 was chosen as the most suitable alternative for a comparative analysis.

In this experiment, signal preprocessing and feature extraction were the most computationally intensive stages. The specific execution times for the subsequent classification and validation phases are detailed in Table 5 for Python and Table 6 for MATLAB.

By far, the most challenging and time-consuming task was the fine-tuning of the SVM hyperparameters. The iterative process required to find the optimal configuration demanded a significant amount of computational time. It is important to note that the total test time reported in the tables includes the time for signal preprocessing, feature extraction, and the final classification execution.

The MATLAB script's reported training time explicitly includes an extensive hyperparameter search using cross-validation. This computationally intensive process, which is integrated into the model training pipeline in MATLAB, is the main source of its longer execution time.

In contrast, our Python implementation used a different approach. Based on preliminary experiments that consistently showed optimal performance with a specific set of hyperparameters ($C = 100.0$, $kernel = rbf$), we chose to hardcode these values. This decision allowed us to exclude the time-consuming hyperparameter search from the final reported training time for the Python script, thereby providing a more direct comparison of the base model training performance itself.

It's important to note that the total test time reported in our tables includes the time for signal preprocessing, feature extraction, and the final classification execution. This methodological choice to hardcode hyperparameters in Python does not impact the validity of the final performance metrics reported for the Python SVM.

Table 5. Execution times for Python-based experiments (seconds).

Classifier	Training	40 dB	20 dB	1 dB
SVM	416.90	198.28	176.24	181.54
DT	470.71	114.74	92.59	96.98
RF	409.97	165.70	136.98	138.61
KNN	480.54	136.63	121.67	122.62
GB	401.67	115.34	93.39	95.91
DNN	1639.65	18.71	18.59	21.45

Table 6. Execution times for MATLAB-based experiments (seconds).

Classifier	Training	40 dB	20 dB	1 dB
SVM	56,395.45	143.22	137.56	153.85
DT	25.67	165.43	142.46	143.81
RF	28.80	149.60	140.04	140.61
KNN	142.47	160.51	161.26	157.89
GB	64.57	160.87	168.20	158.43
DNN	4589.59	148.79	149.56	137.82

5. Discussion

The results indicate that both Python and MATLAB are effective platforms for PQD classification. While MATLAB demonstrates slightly better performance for certain classifiers at higher SNRs, Python's DNN shows superior noise robustness at 1 dB SNR. This suggests that the best choice of platform depends on the specific noise characteristics of the deployment environment and the preferred model architecture.

Across all models, a critical inflection point for performance degradation was observed at approximately 10 dB SNR. This is attributed to the classifier's training range, which spanned from 40 dB down to 10 dB. The classifiers did not perform as well on samples with SNRs below 10 dB because they had not been sufficiently exposed to these conditions during training. The combination of features extracted from multiple domains—including higher-order statistics, wavelet transforms, envelope analysis, and derivatives—was instrumental in enhancing the models' ability to distinguish between different types of disturbances.

The significant distinction between our work and many high-performing studies in the literature lies in our methodology. Many papers, particularly those employing Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), adopt an end-to-end classification approach. These models act as a “black box”, extracting features directly from the raw signal, which can lead to high accuracy but lacks interpretability.

In contrast, our study utilizes a handcrafted feature extraction method, which provides greater insight into the physical characteristics of the disturbances that the model is learning. This approach offers significant practical benefits, including a substantial reduction in the computational resources and time required to obtain the parameters and train the models.

While a direct quantitative comparison is challenging due to these methodological differences, our results are highly competitive. For instance, the Python DNN achieving 84.07% accuracy at 1 dB SNR demonstrates a level of noise robustness that is on par with, and in some cases even surpasses, the performance of more complex end-to-end systems reported in the literature, particularly under high-noise conditions. Our findings thus provide a valuable contribution by showing that a computationally efficient and interpretable feature-based approach can achieve comparable, high-performance results.

Future research will focus on developing a simulated smart grid to conduct more realistic classifier testing. Additional PQDs will be incorporated into the dataset, and further research will be conducted to identify optimal hyperparameters and features to improve performance in highly noisy environments.

6. Conclusions

This study successfully demonstrates that both Machine Learning (ML) and Deep Learning (DL) models can achieve highly reliable Power Quality Disturbance (PQD) classification in noisy environments. The research confirms the viability of these models for real-world applications, with accuracies consistently exceeding 97% for Signal-to-Noise Ratios (SNRs) of 10 dB or higher.

At extreme noise levels, the performance differences become particularly significant. While some ML models demonstrated robustness (e.g., Python's Gradient Boosting and Random Forest maintaining ~78% accuracy at 1 dB SNR), the Python-based Dense Neural Network (DNN) emerged as the most resilient classifier. Achieving 84.07% accuracy at 1 dB SNR, it surpassed other models by 5–7 percentage points and the MATLAB DNN by nearly 7 points.

This numerical superiority has direct and crucial engineering implications. Maintaining a classification accuracy above 80% under severely noisy conditions is critical for industrial grids, as it significantly reduces the risk of misclassification and enhances system reliability. Furthermore, the Python DNN's high efficiency, with an inference time of approximately 18–21 s, makes it exceptionally suitable for near real-time monitoring applications where rapid response is mandatory.

The findings underscore two critical points. First, effective PQD classification in adverse conditions depends on meticulous feature engineering and robust preprocessing strategies. Second, there is no single universally optimal model. The ideal classifier must be carefully selected based on a trade-off between the expected noise level and computational constraints. Our results suggest that while MATLAB implementations remain competitive at higher SNRs, Python combined with a DNN offers a superior balance of noise resilience and computational efficiency, making it the most compelling choice for practical and robust PQD applications.

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

Funding: The authors wish to express their sincere gratitude to the Algeciras Technological Campus Foundation for its crucial support. This research has been made possible thanks to funding (or co-funding) provided by the Ministry of Universities, Research and Innovation of the Government of Andalusia. Additionally, they extend their gratitude to the Government of Andalusia for its continued support of the PAIDI-TIC-168 Research Group (Computational Instrumentation and Industrial Electronics, ICEI).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available on request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Dougherty, J.G.; Stebbins, W.L. Power quality: A utility and industry perspective. In Proceedings of the 1997 IEEE Annual Textile, Fiber and Film Industry Technical Conference, Greenville, SC, USA, 6–8 May 1997; pp. 5–10. [\[CrossRef\]](#)
2. Chawda, G.S.; Shaik, A.G.; Shaik, M.; Padmanaban, S.; Holm-Nielsen, J.B.; Mahela, O.P. Comprehensive Review on Detection and Classification of Power Quality Disturbances in Utility Grid with Renewable Energy Penetration. *IEEE Access* **2020**, *8*, 146807–146830. [\[CrossRef\]](#)
3. Caicedo, J.E.; Agudelo-Martínez, D.; Rivas-Trujillo, E.; Meyer, J. A systematic review of real-time detection and classification of power quality disturbances. *Prot. Control Mod. Power Syst.* **2023**, *8*, 1–37. [\[CrossRef\]](#)
4. Yan, Y.; Chen, K.; Geng, H.; Fan, W.; Zhou, X. A Review on Intelligent Detection and Classification of Power Quality Disturbances: Trends, Methodologies, and Prospects. *Comput. Model. Eng. Sci.* **2023**, *137*, 1345–1379. [\[CrossRef\]](#)

5. Samanta, I.S.; Mohanty, S.; Parida, S.M.; Rout, P.K.; Panda, S.; Bajaj, M.; Blazek, V.; Prokop, L.; Misak, S. Artificial intelligence and machine learning techniques for power quality event classification: A focused review and future insights. *Results Eng.* **2025**, *25*, 103873. [\[CrossRef\]](#)
6. Anjaiah, K.; Divya, J.; Prasad, E.N.V.D.V.; Sharma, R. Signal processing and machine learning techniques in DC microgrids: A review. *Glob. Energy Interconnect.* **2025**, *8*, 598–624. [\[CrossRef\]](#)
7. Oliveira, R.A.d.; Bollen, M.H.J. Deep learning for power quality. *Electr. Power Syst. Res.* **2023**, *214*, 108887. [\[CrossRef\]](#)
8. Samanta, I.S.; Panda, S.; Rout, P.K.; Bajaj, M.; Piecha, M.; Blazek, V.; Prokop, L. A Comprehensive Review of Deep-Learning Applications to Power Quality Analysis. *Energies* **2023**, *16*, 4406. [\[CrossRef\]](#)
9. Albalooshi, F.A.; Qader, M.R. Deep Learning Algorithm for Automatic Classification of Power Quality Disturbances. *Appl. Sci.* **2025**, *15*, 1442. [\[CrossRef\]](#)
10. Mathworks. What Is MATLAB? Available online: <https://es.mathworks.com/discovery/what-is-matlab.html> (accessed on 15 January 2025).
11. Haslwanter, T. *Hands-on Signal Analysis with Python: An Introduction*; Springer Nature: Cham, Switzerland, 2021.
12. W3school, Python Introduction. Available online: https://www.w3schools.com/python/python_intro.asp (accessed on 15 January 2025).
13. *IEEE Std 1159-2019 (Revision of IEEE Std 1159-2009)*; IEEE Recommended Practice for Monitoring Electric Power Quality; IEEE: Piscataway, NJ, USA, 2019; pp. 1–98. [\[CrossRef\]](#)
14. Romero-Ramirez, L.A.; Elvira-Ortiz, D.A.; Jaen-Cuellar, A.Y.; Morinigo-Sotelo, D.; Osornio-Rios, R.A.; Romero-Troncoso, R.d.J. Methodology based on higher-order statistics and genetic algorithms for the classification of power quality disturbances. *IET Gener. Transm. Distrib.* **2020**, *14*, 4580–4592. [\[CrossRef\]](#)
15. Agüera-Pérez, A.; Palomares-Salas, J.C.; González-de-la-Rosa, J.J.; Sierra-Fernández, J.M.; Ayora-Sedeño, D.; Moreno-Muñoz, A. Characterization of electrical sags and swells using higher-order statistical estimators. *Measurement* **2011**, *44*, 1453–1460. [\[CrossRef\]](#)
16. Rampinelli, G.A.; Gasparin, F.P.; Bühler, A.J.; Krenzinger, A.; Romero, F.C. Assessment and mathematical modeling of energy quality parameters of grid connected photovoltaic inverters. *Renew. Sustain. Energy Rev.* **2015**, *52*, 133–141. [\[CrossRef\]](#)
17. Prabakaran, N.; Palanisamy, K. A comprehensive review on reduced switch multilevel inverter topologies, modulation techniques and applications. *Renew. Sustain. Energy Rev.* **2017**, *76*, 1248–1282. [\[CrossRef\]](#)
18. Oppenheim, A.V.; Schaffer, R.W.; Yuen, C.K. Digital Signal Processing. *IEEE Trans. Syst. Man Cybern.* **1978**, *8*, 146. [\[CrossRef\]](#)
19. Khokhar, S.; Zin, A.A.M.; Mokhtar, A.S.; Bhayo, M.A.; Naderipour, A. Automatic classification of single and hybrid power quality disturbances using Wavelet Transform and Modular Probabilistic Neural Network. In Proceedings of the IEEE Conference on Energy Conversion (CENCON), Johor Bahru, Malaysia, 19–20 October 2015; pp. 457–462. [\[CrossRef\]](#)
20. Lee, G.R.; Gommers, R.; Waselewski, F.; Wohlfahrt, K.; O’Leary, A. PyWavelets: A Python package for wavelet analysis. *J. Open Source Softw.* **2019**, *4*, 1237. [\[CrossRef\]](#)
21. Achlerkar, P.D.; Samantaray, S.R.; Manikandan, M.S. Variational Mode Decomposition and Decision Tree Based Detection and Classification of Power Quality Disturbances in Grid-Connected Distributed Generation System. *IEEE Trans. Smart Grid* **2018**, *9*, 3122–3132. [\[CrossRef\]](#)
22. Le, V.; Miller, C.; Tsao, B.H.; Yao, X. Series Arc Fault Identification in DC Distribution Based on Random Forest Predicted Probability. *IEEE J. Emerg. Sel. Top. Power Electron.* **2023**, *11*, 5636–5648. [\[CrossRef\]](#)
23. Akbarpour, A.; Nafar, M.; Simab, M. Multiple power quality disturbances detection and classification with fluctuations of amplitude and decision tree algorithm. *Electr. Eng.* **2022**, *104*, 2333–2343. [\[CrossRef\]](#)
24. Oshiro, T.M.; Perez, P.S.; Baranauskas, J.A. How Many Trees in a Random Forest? In *Machine-learning and Data Mining in Pattern Recognition (MLDM)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7376, pp. 154–168. [\[CrossRef\]](#)
25. Liu, H.; Hussain, F.; Shen, Y.; Morales-Menendez, R.; Abubakar, M.; Yawar, S.J.; Arain, H.J. Signal Processing and Deep Learning Techniques for Power Quality Events Monitoring and Classification. *Electr. Power Compon. Syst.* **2019**, *47*, 1332–1348. [\[CrossRef\]](#)
26. Scikit-Learn. Histogram-based Gradient Boosting Classification Tree. Available online: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingClassifier.html> (accessed on 15 January 2025).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.