processes

MDPI

Article

Using Artificial Intelligence to Classify IEDs' Control Scope from SCL Files

Arthur Kniphoff da Cruz [1,*], Ana Clara Hackenhaar Kellermann [2], João Vitor Meinhardt Swarowsky [3], Ingridy Caroliny da Silva [3], Marcia Elena Jochims Kniphoff da Cruz [3] and Lorenz Däubler [2]

[1] Institute of Computer Science, Clausthal University of Technology, 38678 Clausthal-Zellerfeld, NI, Germany
[2] Institute of Electrical Systems and Automation Engineering, Ostfalia University of Applied Sciences, 38302 Wolfenbüttel, NI, Germany; a.hackenhaar@ostfalia.de (A.C.H.K.); l.daeubler@ostfalia.de (L.D.)
[3] Department of Engineering, Architecture, and Computing, University of Santa Cruz do Sul (UNISC), Santa Cruz do Sul 96815-900, RS, Brazil; joaoswarowsky@mx2.unisc.br (J.V.M.S.); ingridy@mx2.unisc.br (I.C.d.S.); mcruz@unisc.br (M.E.J.K.d.C.)
* Correspondence: arthur.kniphoff.da.cruz@tu-clausthal.de

Abstract

IEC 61850 is one of the most accepted standards worldwide for the automation of electrical substations. This standard uses Substation Configuration Language (SCL) for describing the data model and services from electrical substation components, and SCL files are used for the integration of these components throughout the substation. In this context, the integration of bay level Intelligent Electronic Devices (IEDs) into the station level demands a detailed analysis of the IED's control scope in SCL files and advanced know-how in IEC 61850, increasing the complexity in the engineering process. Hence, this work presents a method to automate the analysis of the control scope of IEDs using SCL files, generating their respective control system object. This is achieved via Machine-Learning (ML) concepts, such as supervised learning and classification algorithms. IEDs used for control and protection of feeder and transformer systems were analyzed, and control system objects were generated for them. The results indicate that the developed method makes it possible to classify the control scope of IEDs using the SCL files from the bay level. This method is a unique development for application in the engineering process of digital substations, reducing the complexity of a critical step towards substation automation.

Keywords: IEC 61850; machine learning; substation configuration language; intelligent electronic devices

Academic Editors: Lefeng Cheng, Xiaoshun Zhang and Huaizhi Wang

Received: 23 December 2025
Revised: 4 January 2026
Accepted: 5 January 2026
Published: 7 January 2026

1. Introduction

IEC 61850, developed by the International Electrotechnical Commission (IEC), is currently one of the most widely adopted standards for electrical substation control and communication network architecture. Published by IEC Technical Committee (TC) 57, the standard has undergone several revisions since 2003, including editions 1.0, 2.0, and 2.1 [1]. Edition 3.0 is currently under development [1].

In order to ensure interoperability between electrical substation components, IEC 61850 defines a generic language based on Extensible Markup Language (XML), known as Substation Configuration Language (SCL). Operating with a structured data model that incorporates standardized syntax and semantics, this language facilitates seamless communication between multi-vendor equipment through the Substation Communication Network (SCN) [2,3].

Figure 1 depicts an electrical substation based on IEC 61850, with communication networks, protocols, and devices hierarchically arranged in the different base levels. As stated by IEC 61850, an electrical substation is mainly segmented in process, bay and station level [4,5]. The energy managed by the substation at the sub-process level can be represented as power model.

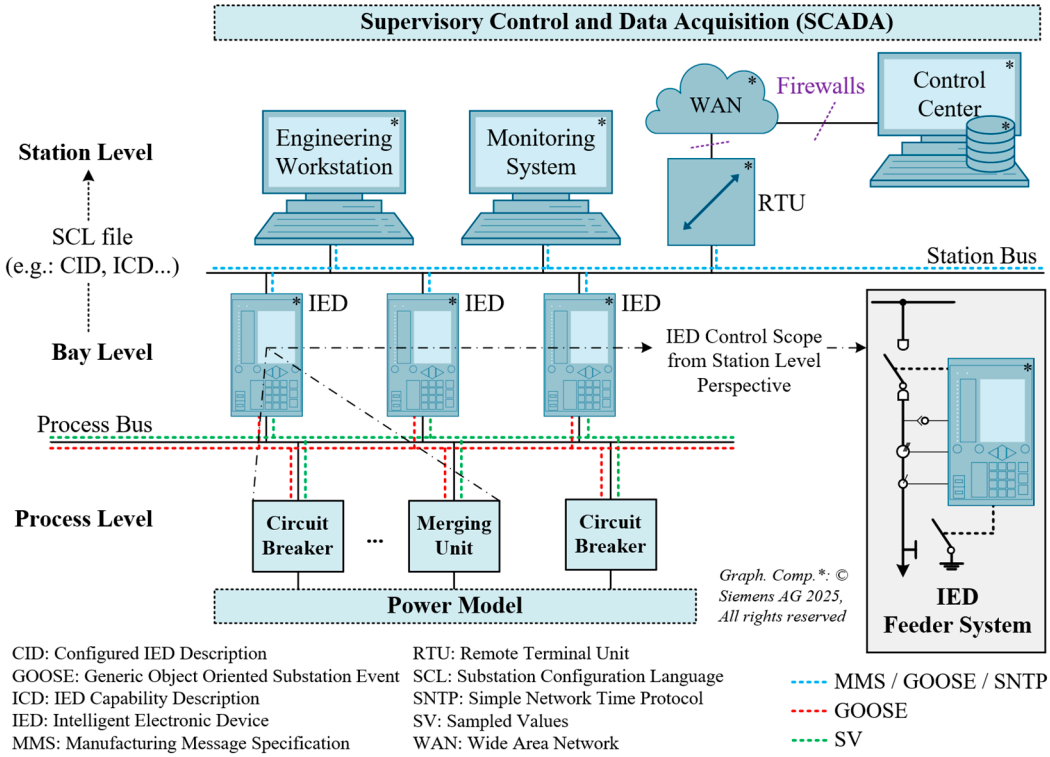


Figure 1. Electrical substation architecture based on IEC 61850 [4,5].

At the process level, components such as Circuit Breakers (CBs), Merging Units (MUs), and different sensors and actuators interface with the power model. The most important level of an electrical substation, the bay level, comprises the IEDs. IEDs, also in practice often known as Protection and Control (P&C) devices, can make autonomous decisions based on process data to orchestrate the energy flow. These devices are also used to transmit direct commands from the station level control system to the process level, as well as to transmit process data and diagnostics from the process level to the control system. Normally, at the station level, a Supervisory Control and Data Acquisition (SCADA) system is used as the control system. Usually, such control systems include Operator Stations (OSs) or Human–Machine Interfaces (HMIs) and archiving systems [4,5].

In many cases, the control systems used in classical Process Automation (PA) and Factory Automation (FA) industries are also used as station level control systems in electrical substations. Often, industrial plants of different sizes contain an electrical substation for the energy management of the plant. In such cases, the same control system—for example, in PA, a Distributed Control System (DCS) with integrated SCADA—is used to control and monitor both the main industrial process and the electrical substation(s). In the classic energy sector, this is frequently the case in power-generating plants (as example: coal-fired, geothermal, gas-steam, and nuclear power plants) where various processes are involved.

A P&C IED represents the control scope of electrical substation components, such as feeders, transformers, line synchronization, or motor protection systems. For example, the classic feeder component, abstractly represented in Figure 1 and commonly used in industrial plant substations, typically comprises various CBs, measurement points, and protection points, among others. In general, the IED control scope is represented as a

modularized object in the control system, using different concepts and methodologies. In this work, it is generically represented as an object or pattern within the control system at the station level. This object consists of different data points (tags) representing the states of the physical system components, a graphical representation, alarms for representing system events, and an archive. For building these objects in the control system, the entire information can be extracted from the IED SCL file—for example from the Configured IED Description (CID) or IED Capability Description (ICD) files.

However, this information extraction process requires a meticulous analysis of the respective IED's SCL file. The station level engineer needs to understand the meaning of each IED data structure, which contains multiple Logical Devices (LDs), Logical Nodes (LNs), Data Objects (DOs) and Data Attributes (DAs), and what they represent. Typically, control system manufacturers provide the object's scope (pattern), and the engineer needs to map the IED's DAs (read from the SCL file) to the object's interface. This process demands extensive IEC 61850 expertise and yet can be challenging even for IEC 61850-experienced engineers, particularly due to its frequent manual configuration requirements. In industrial plant substation projects, particularly where a single control system is used for both energy management and main process control, PA and FA engineers often need to implement the station level of the electrical substation, despite lacking advanced knowledge of IEC 61850. To overcome this barrier, this paper proposes a methodology to assist PA and FA engineers, as well as those from the conventional energy sector, in the process of integrating the bay level into the station level of the electrical substation.

Given the aforementioned challenges, the aim of this article is to present a method that uses Machine Learning (ML) for classifying the bay level IED's control scope based on their SCL files (among others, CID and ICD files). This will enable an automated generation of control system objects, according to the object pattern and in accordance with the respective control scope. In this way, based on a collection of SCL files in the input of the model, an object for the control system is generated in the output. This means that, after assuming the control scope, the model establishes the relationship between the IED DAs and the interface—Input/Output (I/O)—of the object.

For the validation of the prototype developed in this work, files from Siemens AG P&C IEDs for feeder and transformer systems were used. The description of the station level object's I/O was predefined for training the application. As IEC 61850 has a standardized and interoperable data model, SCL files from different manufacturers can be used with this method, and different object definitions can be used in its training. For this purpose, the current state of the prototype presented in this paper needs to be further trained and extended to work with IED's SCL files from different manufacturers. The ML method of supervised learning and classification algorithms Random Forest [6], Logistic Regression, and XGBoost [7] were used.

The trend in modern control systems is towards plant standardization and components modularization, increasing engineering efficiency and enhancing the implementation of plug-and-play systems [8,9]. The proposed solution encompasses both classic and industrial plant electrical substations, facilitating their engineering process through the automated generation of station level objects derived from IED SCL files. The proposed methodology enables the creation of modularized objects, thereby supporting the evolution towards modern control system architectures and helping the process of advancing plug-and-play capabilities in substation engineering and control systems. The implementation demonstrates versatility, being applicable to both greenfield installations and brownfield modernization projects.

This paper is structured as follows: Section 2 presents a background on Artificial Intelligence (AI), a concise review of the IEC 61850 data model, and related research. Section 3

introduces the architecture of the implemented solution, presents the ML methodologies employed, and how the model can be trained, validated and applied. Section 4 presents the obtained results. Finally, Section 5 presents the discussions and conclusions of this research.

2. Background

2.1. Artificial Intelligence

AI is a technology that enables computers and machines to simulate human behavior, evidencing three cognitive skills: learning, reasoning, and self-correction [10]. Its gradual and persistent advance, accompanied by automation, has caused significant and lasting changes across diverse sectors [11]. It covers a series of nested or derivative concepts, such as ML, which implies the construction of models that can learn from data, recognize patterns, and uninterruptedly assess their performance or their ability to decide with little or no human intervention. Among the techniques or algorithms, some include Logistic Regression, Decision Trees, Random Forest and Neural Networks [12,13].

Due to the current potential of these techniques, they were chosen as the basis for the analysis of SCL files, classification of the IED control scope, and generation of objects for the control system within the concept presented in this paper. The following subsections provide a brief introduction to the methods used in this work.

2.1.1. Machine Learning

ML can be subdivided into three different paradigms: supervised, unsupervised, and reinforcement learning. In supervised learning, which is the focus of this study, a program receives data with labels. The algorithm is trained on this input and then tested, in order to assess if it can correctly apply the labels to the new data. Unsupervised learning does not have a training step, so the algorithm needs to look for patterns in the data on its own. It is fed with large amounts of unlabeled data, in which it begins to recognize patterns spontaneously. Similarly, reinforcement learning consists of the interaction between an agent and an environment, in which the agent learns through trial and error to make decisions that maximize a reward function associated with the achievement of a given goal [14].

In addition to the three classical learning approaches discussed above, ML algorithms can be differentiated according to the different learning objectives and practical applications for which they are intended [15]:

- Regression;
- Classification;
- Clustering;
- Association.

Besides these conventional use cases, generative AI has shown significant growth, in which the fundamental principles of classification can be reformulated for the development of generative models based on deep-learning (DL) techniques [15].

The main focus of the research approach presented here is on the classification of existing engineering data based on decision trees and their supervised learning through the analysis of extensive training data.

2.1.2. Decision Trees

Decision trees are among the earliest machine learning techniques and were used early in the natural sciences, e.g., for species identification in biology or in medical diagnosis. They enable the classification of both discrete and continuous data and are trained using supervised learning algorithms.

Figure 2 shows the basic structure of a decision tree.

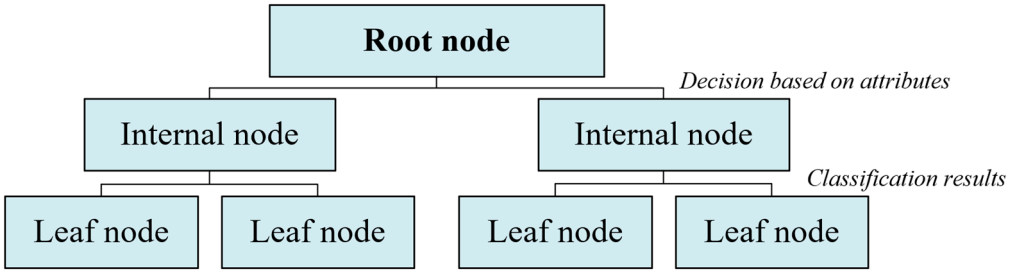


Figure 2. Decision tree, basic architecture [16].

A decision tree is a graph composed of nodes and arcs between nodes, organized according to the typical structure of a tree. From a root node, decisions based on the current attributes of an object to be classified lead, step by step, to the traversal of the internal nodes, until the result of the classification is obtained by reaching the so-called leaf nodes. As a simple example of classification [16], one can consider the current weather conditions (cloud cover, temperature, humidity, and wind) to decide on the practice of outdoor sports (e.g., tennis). Recommendations to practice ("yes") or not to practice ("no") are represented by the leaf nodes. The root node represents the first decision, for example, based on the temperature attribute. This initial decision is followed by internal nodes that represent subsequent decisions, for example, based on humidity or wind. As decision trees constitute a supervised learning methodology, the construction of the decision tree for outdoor sports recommendations is based on example recommendations, known as training data. The goal of the tree-building algorithm is to identify the best root decision from which the tree is built, as well as the organization of subsequent nodes, thus optimizing the performance of the classification, i.e., finding a tree that best fits the sample data.

2.2. IEC 61850

This section presents how IEC 61850 defines its data model, how this model is mapped in SCL, and the different types of SCL files. Furthermore, the last part of this section introduces related works concerning the research topic of this paper.

2.2.1. Data Model

In the field of power systems and electrical substation automation, IEC 61850 is widely adopted, particularly for its capability to enable self-description of data. This standard simplifies design, integration, and maintenance, leading to more efficient and reliable operation of the electrical grid. IEC 61850-7 specifies the Basic Communication Structure, which concerns the exchange of information in substation automation systems. It defines standardized DOs and their DAs, covering aspects such as measurements, control commands, alarms, and configuration parameters [17].

The Common Information Model (CIM) establishes the logical structure of information within the substation, enabling different devices to interpret the semantics of the data they exchange. This ensures that one device can correctly understand the data provided by another. Each DO includes properties, such as interval and unit of measurement, which guarantee consistent interpretation and use of data across all devices in the network. The information model defines five hierarchical levels: server, LD, LN, DO, and DA. Figure 3 shows an example of a simplified IED structure with one LD, two LNs, and some DOs and DAs, organized under different Functional Constraints (FCs) according to their capabilities.

An IED constitutes a physical device that, upon integration into an SCN, can be virtually instantiated as one or more LDs. These LDs may be exclusively associated with a single IED or distributed among multiple IEDs. Within a DO, its corresponding DAs can be systematically classified into FCs according to the semantic relevance and functional

capabilities of the data or service represented by each attribute. A collection of DOs forms an LN, which encapsulates the information necessary to support a specific function or subsystem within the power network. Accordingly, an LD comprises multiple LNs, each encompassing DOs together with their respective DAs. This hierarchical organization grounds the object-oriented nature of the IEC 61850 data model [2,18,19].

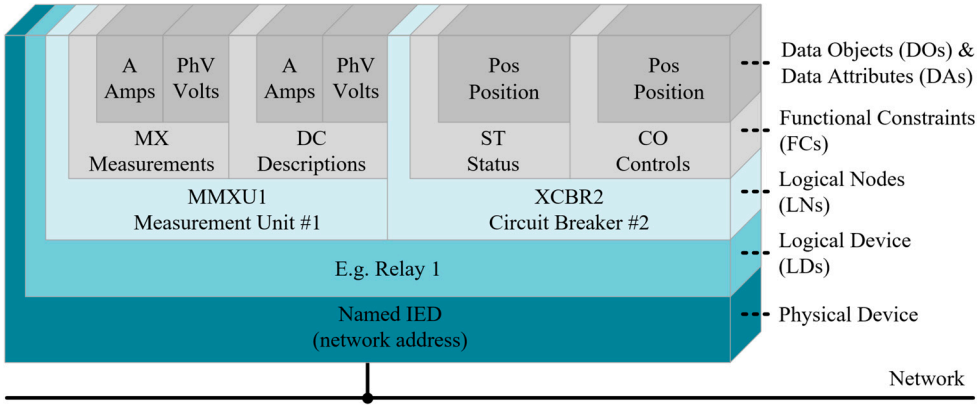


Figure 3. Data model of the standard IEC 61850 [4,5].

2.2.2. Substation Configuration Language

SCL is defined by IEC 61850 as a standard XML-based language used to describe the configuration of substation automation systems. It provides a structured representation of devices, functions, communication connections, and engineering parameters, ensuring interoperability and consistency across multi-vendor equipment.

SCL captures details from the system specification, considering single-line diagrams and the association of LDs, LNs, and their DOs and DAs, with equipment and functions, as well as complete configurations of processes and devices. This enables the definition of pre-configured IEDs with specific functionalities associated with processes or equipment, while also addressing access connections and communication paths available to all components linked to the system [20].

According to Figure 4, the SCL organizes the configuration process into several types of files, each serving a specific purpose within the system engineering workflow [21]:

- **ICD**: defines the functional and communication capabilities of an IED, serving as a basis for integration into the system;
- **SSD** (System Specification Description): describes the specification of the system, including the single-line diagram and the allocation of LNs to the parts and devices of the substation;
- **SCD** (System Configuration Description): consolidates the information of the ICD and SSD files, resulting in the complete configuration of the system, with associations between IEDs and their protection, control and supervision functions;
- **IID** (Instantiated IED Description): contains the detailed configuration of an IED already instantiated in the system, specifying operating parameters and relationships with other devices;
- **CID**: represents the final configuration of the IED, exported to the physical device during the commissioning phase;
- **SED** (System Exchange Description): enables the exchange of data between different IEC 61850 projects, allowing integration between independent engineering systems.

The flowchart seen in Figure 4 contemplates the flow of information between the different SCL files within the engineering environment. The process begins with the database of IEDs (ICD), which powers the System Configurator. This, in turn, uses the SSD to represent the system specification and the SED for project exchanges. After the

integration of the information, the SCD is imported by the IED Configurator for the detailed configuration of the devices. The result of this process is the CID, exported to the IEDs through local or remote transfer, via the substation gateway [21].

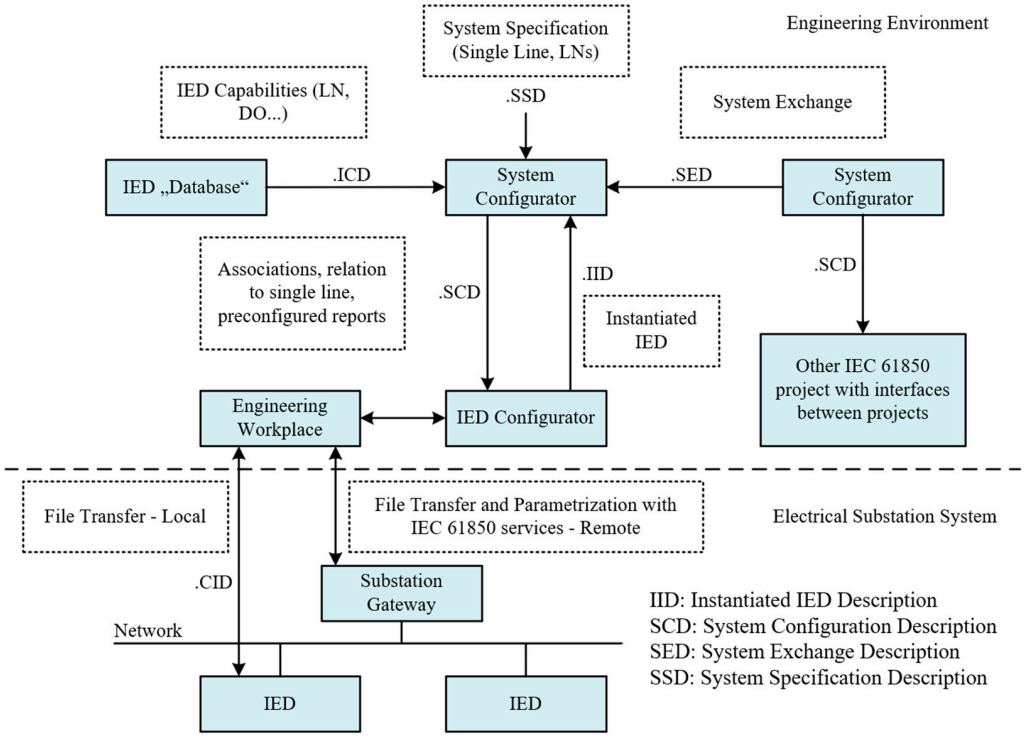


Figure 4. SCL structuring [21].

Thus, the coordinated use of SCL files enables an end-to-end configuration process, from substation architecture modeling to commissioning, ensuring standardization, interoperability, and traceability.

While SCL ensures a standardized representation of configuration, the semantic interpretation of its content for integration remains a manual and resource-intensive process, as highlighted in the specialist literature presented in Section 2.3. Related Work.

2.3. Related Work

IEC 61850-based automation is underpinned by the description of highly structured data through SCL. Sizu et al. [22] demonstrate that converting SCL into valid MMS entities requires parsing mechanisms due to the hierarchical and multi-relational nature of the data model across SSD, SCD, ICD, and CID files, as well as the relationships between LNs, DOs, and DAs. Similarly, Della Giustina et al. [23] emphasize that distributed logics of Generic Object Oriented Substation Event (GOOSE)-based “Fault Location, Isolation, and Service Restoration” (FLISR) schemes rely on the accurate functional description of IEDs in the SCL to ensure deterministic coordination in tasks such as fault location and system reconfiguration. GOOSE is a protocol defined in IEC 61850 for real time communication.

The advancement of AI in electrical power systems provides significant supplemental support. Stock et al. [24] synthesize ML applications, such as, Support Vector Machine (SVM), Artificial Neural Network (ANN), CNN, and fuzzy methods, for prediction, optimization, stability, and event analysis in distribution networks. In turn, Alam et al. [25] discuss AI architectures for monitoring, control, and operation of advanced systems, emphasizing the need for consistent static engineering data to enable proper algorithm modeling. Ahmadi et al. [26] expanded this perspective by systematizing AI methods applied to stability and reliability, including classifiers, deep networks—LSTM, CNN, Deep Neu-

ral Network (DNN)—and hybrid models, and also integrated ML techniques for fault detection, dynamic control, and real-time operation.

The methodological foundations for adopting supervised ML are laid by Saravanan et al. [27], who reviewed the main admissible techniques for classifying complex data, including Decision Trees, Random Forests, Neural Networks, and various classifier approaches, highlighting criteria such as generalization, interpretability, and computational performance. N Rincy et al. [28] extend this perspective by detailing models of supervised, unsupervised, and reinforcement learning, explaining their training mechanisms and suitable application scenarios, particularly in supervised classification when labels are known. These studies underpin the use of supervised machine learning to extract functional meaning from structured and complex data, such as that contained in SCL.

In summary, three crucial pillars emerge: (i) the functional inspection of SCL remains manual and prone to inconsistencies; (ii) AI, ML, and DL are already widely applied to highly complex operational problems in electrical power systems; (iii) supervised modeling is suitable for extracting functional semantics from structured data. Thus, the proposed method—applying supervised ML to infer the functional scope of IEDs from SCL—is positioned as a solution aligned with existing gaps and persistent demands, delivering greater accuracy and scalability in distinguishing IEDs and strengthening the reliability of IEC 61850 architectures.

3. Materials and Methods

3.1. Model Architecture

A structure composed of three modules was outlined as the model architecture, each with clearly defined functions. Figure 5 illustrates the model architecture in an abstract manner. The division of responsibilities between the three classes parser, classifier, and generator, aims to ensure the stability, maintainability and modularity of the system, in line with good programming practices. This approach facilitates error detection and correction, as well as the reuse of each module in other projects without the need for substantial modifications.

The first module "Parser" corresponds to a class that parses the SCL files. The second module "Classifier" constitutes the core of the model, implementing the decision tree-based classification algorithms Random Forest, Logistic Regression and XGBoost. In addition, it includes methods for preprocessing data and generating graphical representations. Finally, the third module "Generator" is dedicated to the generation of the output files, containing the relation of the object's I/Os and the IED's DAs.

3.1.1. Software Environment

Python 3.11.9 was chosen as the programming language due to its widespread adoption in the field of AI. Among the tools utilized, the scikit-learn (sklearn) 1.7.2 [29] and XGBoost 3.0.0 [7] libraries were selected to enable the application of the classification algorithms Random Forest, Logistic Regression and XGBoost to the model.

Additionally, the pandas [30] library was used to create and manipulate DataFrames, while Phyton's built in xml.etree.ElementTree [31] package was used to read and extract data from the SCL files. Moreover, the "YAML Ain't Markup Language" (YAML) library was imported to help with the process of mapping training features and external objects. Also, the set of libraries formed by NumPy 2.3.0 [32], Matplotlib 3.10.3 [33], and Seaborn 0.13.2 [34], combined with the sklearn.metrics [29] and sklearn.calibration [29] packages, was used to generate the graphical representations of the model's performance. Finally, other modules such as logging, typing, and itertools were used to complement the code and support the development of high-quality Python program. Additionally, Python Enhancement Proposal 8

(PEP 8) [35] was adopted as a reference for coding style. In addition to the rules defined by the standard, common software development techniques such as modularization, readability, and distribution of responsibilities, among others, were used.

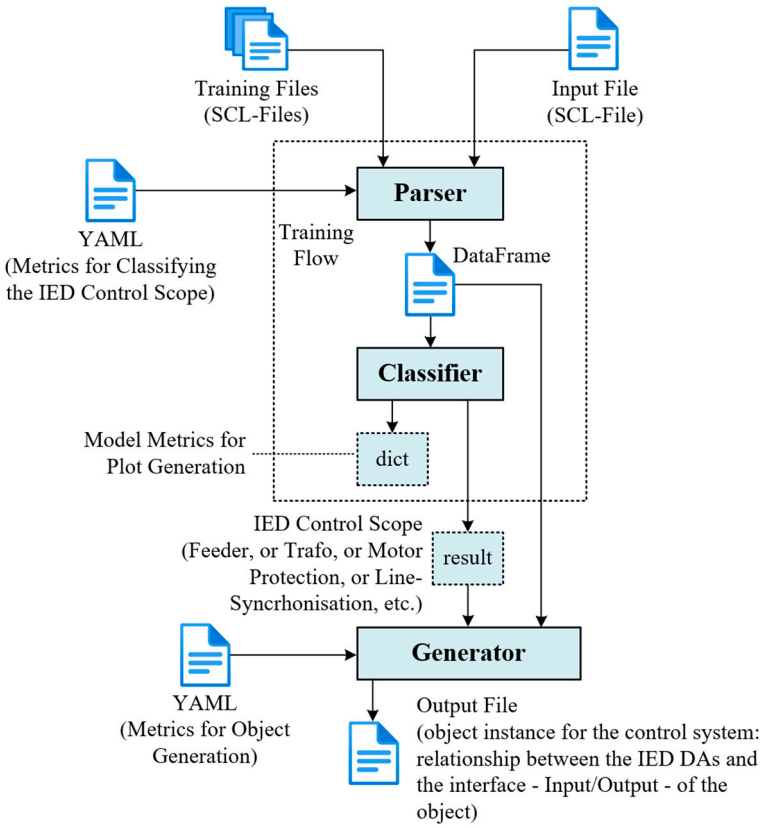


Figure 5. Model architecture.

3.1.2. Parser Module

The Parser class provides the input data both for training to construct optimal decision trees (training files and YAML metrics) and for classifying unclassified SCL files (input files). Thereby it has two purposes in the classification method developed in this work: selecting features (metrics contained in a YAML file for classifying the IED control scope) for the model training and collecting the information required, from the SCL files, for generating DA IEC 61850 addresses in the output. The parser can extract all IED data at the DA level from the various SCL file types defined in IEC 61850-6 [21].

Regarding the training procedure, the parser needs to know the type of control scope (target class) that the IED in the respective file represents. In the developed prototype, it is required to group files into specific directories that represent the control scope of an IED. For example, a training SCL file for a feeder object must be saved in a prototype-access directory named "FEEDER". This requirement arises because the code assigns the name of the directory containing the target file as the value of its target column, which is essential for training and evaluating the model. Although SCL files can contain more than one IED, regarding the prototype's training and input files, each file may only contain a single IED.

Finally, upon completion of the parsing procedure, a DataFrame will have been constructed containing all LNs—standard LNs defined by IEC 61850-7-4 [36] and non-standard, proprietary LNs—associated with the selected IED, along with the IED's name, type, manufacturer, IP address, control scope (target), and source SCL file. Depending on the adopted parsing flow, the DataFrame may also include all LDs, DOs, DAs, and FCs associated with the IED. This DataFrame will serve as the basis for processing in the subsequent classes.

3.1.3. Classifier Module

The Classifier Module, one of the modules shown in Figure 5, is subdivided into different methods. Figure 6 illustrates the complete sequence of Classifier Module steps used to prepare the DataFrame for the pre-processing and training procedures. The first method invoked in the Classifier Module is prepare_train_dataframe(). A variant of this method, called prepare_prediction_dataframe(), is applied to handle DataFrames coming from files intended for classification only. In this variant, only the first three and the fifth preparation steps are performed. The other operations are unnecessary for prediction since they only adapt the DataFrame to training requirements, such as keeping the targets available for testing and splitting the samples between train and test groups.

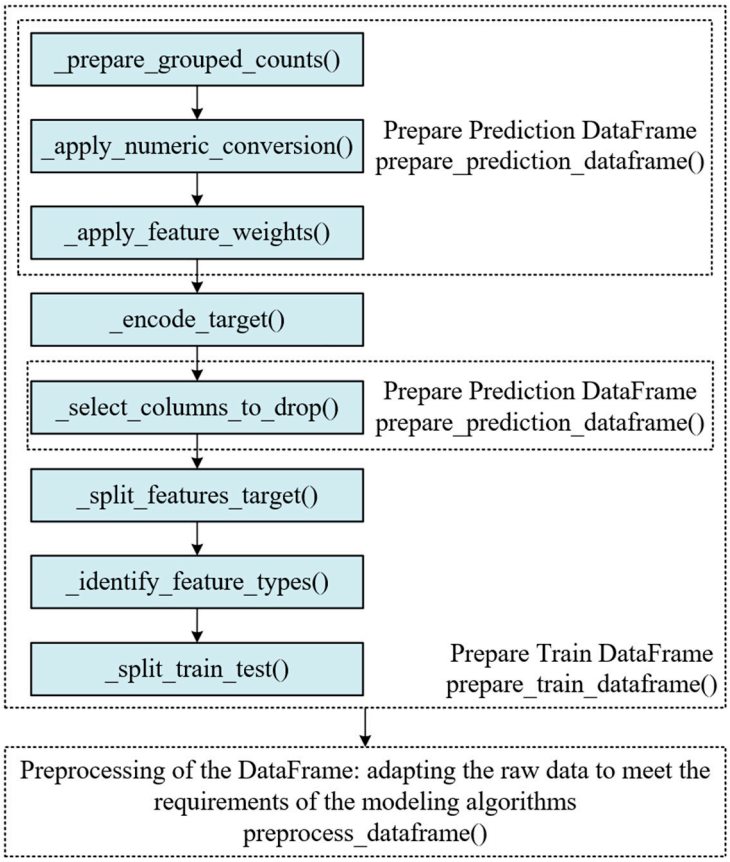


Figure 6. DataFrame preparation workflow.

As represented in Figure 6, the first step is performed by the function _prepare_grouped_counts(), which enriches the DataFrame by incorporating aggregated frequency counts of each LN category across combinations of IED scope type, IED manufacturer, and source file. These grouped counts are then merged back into the DataFrame, ensuring that each record corresponds to a single file. The procedure subsequently converts all applicable fields to a numeric form, coercing non-convertible values to zero via the _apply_numeric_conversion() function, and applies predefined custom feature weights, using the _apply_features_weights() function. Additionally, the target column is encoded by _encode_target(), using LabelEncoder [29] to ensure compatibility with learning algorithms that require numerical targets, such as XGBoost. Due to the application of LabelEncoder, the targets received integer values, in this prototype, control scope of feeder system (FEEDER) being "0" and of transformer (TRAFO) being "1". Finally, a predefined set of columns deemed unnecessary for the model in the next procedures is removed by

_select_columns_to_drop(), and the DataFrame is divided by _split_features_target() into two subsets: "Features (X)" and "Targets (Y)".

The features correspond to the input attributes originating from the source file, while the targets refer to the labels or classes of the problem corresponding to the IEDs to be classified. In the context of this study, IEC 61850 LNs and proprietary LNs, under the alias lnClass, and the IED type and manufacturer were selected as features. Examples of standardized LNs used include MMXU, PTOC, and LPHD, and examples of proprietary LNs include SSYM and ROPA. On the other hand, in this case the targets correspond to FEEDER and TRAFO. Thus, _identify_feature_types() identifies and segregates the resulting numerical and categorical feature types. This segregation of columns based on their type is necessary for preprocessing, as numeric columns are treated differently from categorical or other types of columns. Finally, a stratified train–test split is performed by _split_train_test(), allocating 80% of the samples for training and 20% for testing.

The second method, preprocess_dataframe(), is designed to perform the preprocessing step of the DataFrame. This phase is crucial in the context of machine learning, as it adapts the raw data to the requirements of the modeling algorithms. In the adopted library, the sklearn.preprocessing package [29] offers tools to transform original variables into representations compatible with estimators. Three main tools were used: ColumnTransformer, StandardScaler and OneHotEncoder [29].

ColumnTransformer enables the application of different preprocessing steps to distinct columns within the same dataset. By specifying which transformer corresponds to which set of features, ColumnTransformer parallelly processes all transformations and merges the outputs into a single feature matrix. Columns that are not explicitly assigned may be dropped, kept unmodified, or transformed separately [29]. StandardScaler standardizes numerical attributes by removing the mean and adjusting the variance to unity. This normalization prevents variables on different scales from influencing learning unevenly. In addition, StandardScaler calculates statistics on the training set and applies the same transformation to the test data, ensuring consistency in the pipeline and preventing data leakage [29]. In turn, OneHotEncoder converts each categorical variable into a binary vector (one-hot encoding), avoiding incorrect interpretations of order between categories. [29].

The combined use of these tools ensures that numerical and categorical variables are appropriately handled. Consequently, properly preprocessed data becomes suitable input for the pipelines of the methods from each classification algorithm. Regarding the chosen algorithms, three classifiers were selected: Random Forest, Logistic Regression, and XGBoost. These algorithms were chosen because of their strong capability to process structured data, their suitability for classification, and their ability to handle categorical features. The first two are available in the scikit-learn library, while the third comes from the xgboost library. Implementation and testing were carried out on all three models to determine which one performs best in the analyzed context.

Random Forest is an ensemble algorithm based on the combination of multiple randomized decision trees. It works according to the perturb-and-combine principle, in which different trees are built from subsets of the dataset and attributes, both randomly selected. This double introduction of randomness, in the samples (via bootstrap sampling) and in the variables considered in each division, promotes diversity among the base classifiers, reducing the correlation between them [29]. Each tree is trained on a sample of the same size as the original set, obtained with replacement. In the scikit-learn library used in this work, individual predictions are aggregated by averaging the predicted probabilities for each class, rather than simple majority voting [29]. The combination of these strategies (injection of randomness and aggregation of predictions) aims to reduce model variance,

improve predictive accuracy, and mitigate the risk of overfitting [29]. The configured parameters for the Random Forest pipeline are shown in Figure 7.

```
model = Pipeline(
    steps=[
        ("pre", self.preprocess),
        ("clf", RandomForestClassifier(
            n_estimators=300,
            random_state=42,
            n_jobs=-1,
        ))
    ]
)
```

Figure 7. Random Forest parameters; pipeline call in Python.

First, the n_estimators parameter defines the number of decision trees in the forest. The value of 300 is considered a balance to achieve a good accuracy rate without compromising computational performance. Second, random_state ensures the reproducibility of results by controlling randomization processes. Finally, the n_jobs setting allows the use of all Central Processing Unit cores, effectively reducing execution time through parallelization [29].

XGBoost stands for Extreme Gradient Boosting. Gradient boosting is a machine-learning technique in which predictive models are sequentially built. Each new model is trained to reduce the residual errors of its predecessors. This process constitutes a boosting strategy, where a set of low-accuracy models (called "weak learners") are combined to form a single high-accuracy model (called a "strong learner"). The XGBoost solution offers an optimized and scalable implementation of gradient-boosted decision trees. In this framework, decision trees function as weak learners whose sequential aggregation improves predictive accuracy. The resulting model is a group composed of multiple Classification and Regression Trees (CARTs). CARTs always use binary splits, have stricter rules and structure than other decision trees, and are developed especially for classification and regression problems [7]. The parameters configured for the XGBoost pipeline are shown in Figure 8.

```
model = Pipeline(
    steps=[
        ("pre", self.preprocess),
        ("clf", XGBClassifier(
            n_estimators=500,
            learning_rate=0.05,
            max_depth=6,
            subsample=0.9,
            colsample_bytree=0.9,
            reg_lambda=1.0,
            random_state=42,
            n_jobs=-1,
            eval_metric="mlogloss",
            verbosity=0,
        ))
    ]
)
```

Figure 8. XGBoost parameters; pipeline call in Python.

Regarding the selected parameters, the verbosity parameter controls the amount of information printed during training, with a value of zero indicating that no output should be produced. The subsample parameter specifies the fraction of training instances randomly selected in each boosting iteration, and values equal to or greater than 0.5 are recommended to help reduce overfitting. Complementing this parameter, colsample_bytree defines the fraction of columns sampled when constructing each tree. Additionally, max_depth determines the maximum depth of each tree, while learning_rate regulates the contribution of each tree to prevent overfitting. The reg_lambda parameter (L2 regularization) penalizes large weights, and higher values lead to more conservative models. The eval_metric parameter specifies the evaluation metric used; in this case, the metric is logistic loss [7]. Finally, the parameters n_estimators, random_state, and n_jobs have the same meaning as explained for the aforementioned method.

Logistic regression, contrary to what its name suggests, is a linear classification model. Its internal routine aims to estimate the probability of a sample belonging to a given class based on a linear combination of input attributes. This probability is modeled by applying a logistic (sigmoid) function, which converts the result to a value between zero and one. Thus, based on a defined threshold, the algorithm makes its final decision regarding the chosen classification. This algorithm can be applied to both binary and multiclass classification problems. In addition, regularization techniques such as L1, L2, or Elastic-Net are used to avoid overfitting [29]. Unlike the previous algorithms, this one underwent a specific preprocessing step. The same dataset was used, but in this case, the numeric columns of the DataFrame were scaled to improve the model's performance. Although scaling provides significant benefits for linear regression-based algorithms, its impact on other classifiers is generally insignificant. The parameters configured for the Logistic Regression pipeline on this occasion are shown in Figure 9.

```
model = Pipeline(
    steps=[
        ("pre", self.preprocessLogReg),
        ("clf", LogisticRegression(
            solver="saga",
            max_iter=2000,
            n_jobs=-1,
            multi_class="auto",
            random_state=42,
        ))
    ]
)
```

Figure 9. Logistic Regression parameters; pipeline call in Python.

The first parameter configured was the solver with the value "saga", chosen for its compatibility with different forms of regularization and its efficiency with scaled data. The max_iter parameter was set to 2000 to ensure sufficient iterations for the algorithm to converge [29]. In addition, the multi_class parameter enables automatic selection of the most appropriate strategy between one-vs-rest and multinomial, according to the characteristics of the problem. Finally, both random_state and n_jobs parameters are configured in the same way as the aforementioned pipelines.

Despite the differences between the algorithms, they all share a common feature in their implementation: the return at the end of the function. In all cases, the result consists of a call to the Cross-Validation (CV) method _cross_validate(), to which the pipeline, built with the respective classifier and its configuration, is assigned as a parameter. This method applies the cross-validation procedure to the model assigned as a parameter.

CV is a technique used to evaluate how well a ML model is expected to perform on unseen data. Rather than dividing the dataset once into training, validation, and test subsets, cross-validation repeatedly trains the model on most of the data while reserving different portions for evaluation. In k-fold CV, the dataset is partitioned into k subsets, and the model is trained k times, each time leaving out one subset for testing. The final performance metric is computed as the average across all folds [29]. The CV technique is depicted in Figure 10. The primary factor influencing the adoption of the cross-validation technique was the small number of samples used on the first validation.

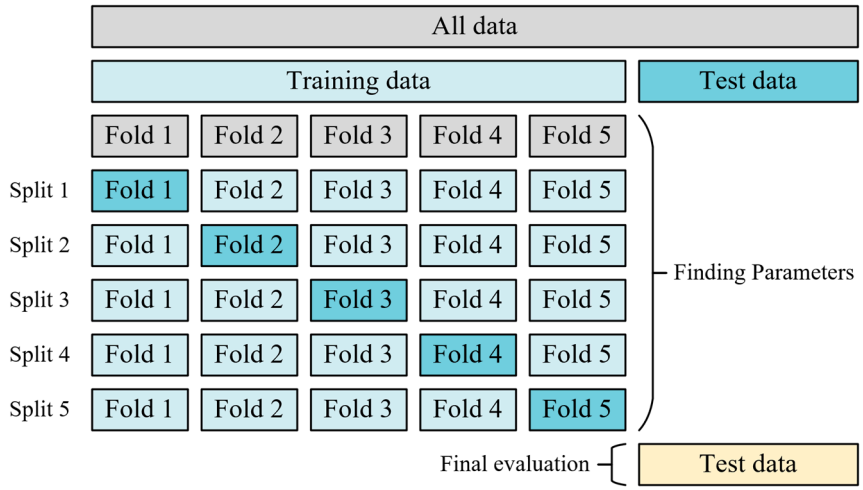


Figure 10. Cross Validation [29].

3.1.4. Generator Module

The goal of this module is to generate the object for the control system. The generator receives as input the DataFrame containing all the LDs, LNs, DOs, DAs, and FCs from the listed IED, as well as the classifier's output indicating the control scope of the IED, and a YAML configuration file that includes the skeleton of an object and metrics to enable the generator to establish the relationship between the IED's DAs and the I/O of the object skeleton. In this way, the generator is able to associate the I/Os of the object skeleton from the YAML file of a specific control scope, with their respective IEC 61850 DA addresses "LD/LNFCDO$DA" from the DataFrame. To achieve this, the build_output_block() function was implemented. Within the function, addresses are constructed for the instantiated elements in the processed file. Using the configuration files, the function also loads standardized object definitions according to the respective IED scope, defined by the received class value.

A loop-based logic was implemented to filter addresses based on the elements of interest, for example, DOs, and iterate through all address entries to identify the correct match for each object. The YAML configuration files facilitated this process by explicitly specifying the expected DO and DA for each object. In the SCL files used for this initial validation, two sections were of primary interest: one containing the LNs declarations, in which all possible DOs and DAs were defined, and another containing only the instantiated LNs, along with their respective instantiated DOs and DAs. Thus, in this layout, not all declared DAs were instantiated, but all instantiated DAs corresponded to a prior declaration. Therefore, it is important to note that the parsing procedure operated on instantiated elements rather than their declarations.

3.2. Training

The method developed in this research can be applied, for example, using the following practical options offered by Siemens AG control systems, which are widely employed across

various industries such as PA, FA, and the conventional energy sector: Power Control Library for SIMATIC PCS 7 (DCS with integrated SCADA) [37], Station Gateway-Advanced Power Control Library V4.04 for SIMATIC TIA Portal (unified engineering framework for Programmable Logic Controller-based automation) [38], and Advanced Power Control Library for WinCC Open Architecture (SCADA system) [39]. These three options enable the integration of vendor-independent IEDs into Siemens AG control systems using IEC 61850. Furthermore, they allow the representation of an IED's control scope in an object format, such as feeder system, transformer, and line synchronization. In this work, the skeletons of a feeder object (FEEDER) and a transformer object (TRAFO) were employed during the training and validation process of the developed prototype. The skeletons of these two objects are presented in Appendices A and B (columns "Object name" and "Meaning").

The model was trained using a total of 80 samples, evenly divided between two IED types: 40 FEEDER files, model Siemens AG 7SJ82 (Nuremberg, Germany) and 40 TRAFO files, model Siemens AG 7UT86 (Nuremberg, Germany). This balanced class distribution eliminated the need for additional preprocessing or algorithmic adjustments to mitigate class imbalance. Of the 80 samples, 64 (80%) were used for training and 16 (20%) were reserved for testing and performance evaluation. All samples originated from the same manufacturer, shared an identical standard SCL file structure, but not identical data structure.

As IEC 61850 is globally recognized for its object-oriented, standardized data model represented by SCL, it enables interoperability among devices from different vendors. Consequently, it also allows vendors to define vendor-specific LNs, which can be accessed by IEDs from other vendors. Therefore, since the model developed in this paper is designed in accordance with SCL and its features, it can be used by any vendor adhering to the standard, considering that the vendor-specific LNs are added to the model parameters before training. Thus, the feature-weighting strategy was derived from patterns observed in the standard and in manufacturer's files. LNs unique to a particular IED model were assigned a weight of 1.5, whereas LNs common to both models were assigned a weight of 1.0. The IED type (7SJ82 or 7UT86) received a weight of 2.0, while the manufacturer attribute was assigned a reduced weight of 0.5. From a machine learning perspective, both the limited sample size and the lack of structural diversity among the samples constrain the ability to reliably assess the model's generalization performance. These limitations should be considered when interpreting the evaluation results.

4. Results

4.1. Model Metrics and Evaluation

This section provides an overview of the metrics obtained from the three tested module algorithms, followed by an analysis of their performance. To support an accurate comparison of the algorithms, the following evaluation graphs, presented in the respective figures in Appendix C were generated: confusion matrices-without normalization (Figure A1), confidence histograms (Figure A2), calibration curves (Figure A3), precision–recall curves (Figure A4), and Receiver Operating Characteristic (ROC) curves (Figure A5). In all graphs presented in the annex, the target classes are encoded numerically: FEEDER is represented by zero "0", and TRAFO is represented by one "1".

Table 1 provides an overview of the test results achieved with the different training algorithms and their evaluation signed with "+" (good evaluation result), "0" (minor) and "−" (bad).

The confusion matrix is used to evaluate the accuracy of a classification model. The three algorithms analyzed showed 100% accuracy in the 16 test cases (Figure A1a–c). Although this result is highly satisfactory, the absence of incorrect classifications should be interpreted with prudency. This issue is discussedz in more depth in Section 5.

Table 1. Test results.

Metrics	Random Forest	XGBoost	Logistic Regression
Confusion Matrix	+	+	+
Confidence Histogram	+	–	0
Calibration Curve	+	0	0
Precision–Recall Curve	+	+	+
ROC Curve	+	+	+

A confidence histogram illustrates the distribution of predicted probabilities. Since all models correctly classified every case, high confidence scores were expected. Random Forest exhibited the highest confidence, with 15 predictions between 0.9 and 1.0 and only one prediction below 0.95 (Figure A2a). In contrast, XGBoost showed the lowest confidence, with all 16 predictions below 0.97 (Figure A2b). Logistic Regression demonstrated the widest variability in confidence, with 13 predictions between 0.98 and 1.0, one prediction between 0.96 and 0.98, and one prediction below 0.89 (Figure A2c).

A calibration curve indicates how closely predicted probabilities correspond to the true likelihood of an event [29]. Random Forest exhibited the best calibration, with its curve remaining closest to the reference line (Figure A3a). Logistic Regression and XGBoost both obtained a Brier score of 0.0011, also an excellent result. However, their calibration curves differed: Logistic Regression displayed high overconfidence (above the reference line) for predicted probabilities above 0.1 (Figure A3c), whereas XGBoost shifted from underconfidence (below the reference line) at probabilities below 0.4 to overconfidence at probabilities above 0.6 (Figure A3b).

This graph complements the information provided by the confidence histogram. Considering both metrics, it can be concluded that the high confidence of the Random Forest model is well calibrated, whereas the confidence levels above 0.8 observed in Logistic Regression and XGBoost reflect overconfidence and do not accurately represent the true likelihood of correct predictions.

Precision–Recall Curves illustrate the balance between precision and recall [29]. The results are similar for all three algorithms. The curves (Figure A4a–c) indicate both high precision and high recall, showing that all models return accurate results. By comparing these curves with the previous graphs, it can be confirmed that the models are indeed accurate, although small differences in performance exist among them.

ROC curves illustrate the variation of the true positive rate as a function of the false positive rate and are widely used to evaluate the performance of binary classifiers [29]. In the results obtained in this study, the metrics converge to a scenario considered "ideal", in which the rate of true positives reaches the unit value when the rate of false positives is equal to zero [29]. Therefore, the area under the curve for all evaluated algorithms is equal to 1.00 (Figure A5a–c). However, this scenario should be interpreted with care since it is not considered representative of practical conditions.

The comparative analysis summarized in Table 1 shows that the Random Forest algorithm presents moderately superior performance in the construction of the decision tree, which qualifies it as the most suitable option for future training missions.

4.2. Generated Output

Finally, a spreadsheet of logical node addresses was generated following the input parameters of the generator class—YAML file with an ideal configuration, expressing the relation between the IED DAs and object I/Os for the 7SJ82 (FEEDER) and 7UT86 (TRAFO) models. The addresses include status objects standardized with the FC ST, and measurement objects standardized with the FC MX. In addition, IEC 61850 standard also

dictates that control objects should have FC CO, but this has not yet been implemented. All objects included in the YAML file and therefore in the spreadsheet were mapped based on validation using multiple sources [37–39].

Finally, two files previously unseen by the model in the training stage were selected for the validation of the address block generation: one corresponding to a 7SJ82 (FEEDER) device and the other to a 7UT86 (TRAFO) device. Using the mappings and the developed code, it was possible to generate an address block for the 7SJ82 FEEDER model containing 160 registered objects and 86 automatically generated addresses, as well as an address block for the 7UT86 TRAFO model containing 69 registered objects and 54 automatically generated addresses. Both resulting spreadsheets are included in the Appendices A and B. The missing addresses correspond to addresses that are not applicable to the specific IED model or data-structure, for example, in Table A1, the object "Pos9_Val" does not exist in the configuration of the IED (previously unseen file for the FEEDER 7SJ82). This means that the station-level FEEDER object contains a "Pos9_Val" object that could be used, but the tested IED does not contain an DO/DA for it.

5. Discussion and Conclusions

This paper introduced a method to automate the analysis of the control scope of IEDs using SCL files, and to generate their respective control system object with its I/Os and DA addresses. This method mainly contributes to the engineering process of integrating the bay level into the station level of the electrical substation. The model was able, under predefined boundaries, to successfully define the control scope and generate the relation between IED's DAs and control system object I/O of FEEDER and TRAFO IEDs, for implementation in the substation station level control system. In addition, a comparison between the different ML models used for the prediction of the control scope of the IEDs was performed. Based on the analysis of the graphs presented in the results, it can be concluded that the three algorithms evaluated in this study achieved performance metrics consistent with the expected outcomes. However, a detailed comparison of the models indicates that the Random Forest algorithm performed slightly better than the others (XGBoost and Logistic Regression), although all achieved 100% accuracy, Random Forest demonstrated the best calibration, making it the most reliable, not just the most accurate. This conclusion is primarily supported by the model's calibration curve, which demonstrates that Random Forest not only generated predictions with a high degree of confidence but also accurately represented the actual distribution of the samples.

Although this prototype successfully fulfilled the aim of this article, it is important to emphasize that this development is characterized by a limited number of samples, low variability among them, and the presence of strong discriminative features, such as the type of IED. These conditions may have influenced the observed performance results. Rather than undermining the findings, these limitations highlight promising directions for future research, such as expanding the dataset diversity and exploring features that are independent of specific models. Consequently, future work will prioritize expanding both the quantity and diversity of samples. Increasing sample availability will enable the development of models that are more adaptable to a wider range of scenarios and will yield performance metrics that reflect real-world conditions with higher accuracy. Moreover, a broader dataset will enhance understanding of the internal structure of SCL files and their constituent elements, supporting the development of a more versatile and universally applicable parsing logic. Finally, depending on the expanded availability of samples, future research may also explore the feasibility of transitioning from a machine learning-based model to a deep learning-based approach.

Regarding the generated address block, its construction remains limited to the 7SJ82 and 7UT86 IED models. Since the standard also allows vendor-specific objects, as a result, the current implementation is still unable to adapt the block's characteristics to devices produced by other manufacturers or to models outside the scope of this research, although, because the model proposed in this work was built following the SCL specification and its associated characteristics, it is applicable to any vendor that complies with the standard, provided that the vendor-specific LNs are incorporated into the model's parameters prior to training. Addressing this limitation constitutes an additional objective for future work. Consequently, with a larger set of SCL files from different vendors, it will be possible to deepen the understanding of the elements they contain, their internal relationships, and the different applications of declared objects and their instances. This expanded knowledge will support the development of more robust and generalizable code.

This work represents an initial step toward a machine-learning-based software solution capable of generating control system objects for IEDs by deriving them from the device's control scope as described in their SCL files, including the mapping between their I/Os and the corresponding DA addresses. The future work outlined in this section will focus on scaling and further enhancing this solution, given the relevance of a streamlined substation engineering process that supports PA and FA engineers in integrating substations into industrial plants and processes, an increasingly common requirement in a world with ever-growing energy demands.

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

Funding: This research received no external funding.

Data Availability Statement: All original contributions presented in this study are included within the article. Any additional inquiries should be addressed to the corresponding author.

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

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
ANN	Artificial Neural Network
CARTs	Classification and Regression Trees
CB	Circuit Breaker
CID	Configured IED Description
CIM	Common Information Model
CNN	Convolutional Neutral Network
CV	Cross-Validation
DA	Data Attributes
DCS	Distributed Control System
DL	Deep Learning
DNN	Deep Neural Network
DO	Data Objects
FA	Factory Automation
FC	Functional Constraints

FLISR	Fault Location, Isolation, and Service Restoration
GAN	Generative Adversarial Network
GOOSE	Generic Object Oriented Substation Event
GRU	Gated Recurrent Unit
HMI	Human–Machine Interface
HPC	High-Performance Computing
I/O	Input/Output
ICD	IED Capability Description
ID	Identifier
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
IID	Instantiated IED Description
LD	Logical Devices
LN	Logical Nodes
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multilayer Perceptron
MMS	Manufacturing Message Specification
MU	Merging Unit
OS	Operator Station
P&C	Protection and Control
PA	Process Automation
PEP 8	Python Enhancement Proposal 8
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
SCD	System Configuration Description
SCL	Substation Configuration Language
SCN	Substation Communication Network
SED	System Exchange Description
sklearn	scikit-learn
SNTP	Simple Network Time Protocol
SSD	System Specification Description
SV	Sampled Values
SVM	Support Vector Machine
TC	Technical Committee
WAN	Wide Area Network
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

Appendix A. Generated FEEDER Object

Table A1. Pattern object of a FEEDER and the mapped IEC 61850 Data Attribute addresses of the IED 7SJ82.

Object Name [37–39]	Meaning [37–39]	Automatically Mapped and Generated Addresses
Pos0_Val	Position of Q0	Dc1/CSWI.ST.Pos.stVal
Pos1_Val	Position of Q1	Dc2/CSWI.ST.Pos.stVal
Pos2_Val	Position of Q2	Dc3/CSWI.ST.Pos.stVal
Pos8_Val	Position of Q8	CB1/CSWI.ST.Pos.stVal
Pos9_Val	Position of Q9	
Pos51_Val	Position of Q51	
Pos52_Val	Position of Q52	
Pos53_Val	Position of Q53	
Q0_EnaCls_Val	IEC Interlock Enable Closing Switch Q0	Dc1/CILO.ST.EnaCls.stVal
Q1_EnaCls_Val	IEC Interlock Enable Closing Switch Q1	Dc2/CILO.ST.EnaCls.stVal
Q2_EnaCls_Val	IEC Interlock Enable Closing Switch Q2	Dc3/CILO.ST.EnaCls.stVal

Table A1. *Cont.*

Object Name [37–39]	Meaning [37–39]	Automatically Mapped and Generated Addresses
Q8_EnaCls_Val	IEC Interlock Enable Closing Switch Q8	CB1/CILO.ST.EnaCls.stVal
Q9_EnaCls_Val	IEC Interlock Enable Closing Switch Q9	
Q51_EnaCls_Val	IEC Interlock Enable Closing Switch Q51	
Q52_EnaCls_Val	IEC Interlock Enable Closing Switch Q52	
Q53_EnaCls_Val	IEC Interlock Enable Closing Switch Q53	
Q0_EnaOpn_Val	IEC Interlock Enable Opening Switch Q0	Dc1/CILO.ST.EnaOpn.stVal
Q1_EnaOpn_Val	IEC Interlock Enable Opening Switch Q1	Dc2/CILO.ST.EnaOpn.stVal
Q2_EnaOpn_Val	IEC Interlock Enable Opening Switch Q2	Dc3/CILO.ST.EnaOpn.stVal
Q8_EnaOpn_Val	IEC Interlock Enable Opening Switch Q8	CB1/CILO.ST.EnaOpn.stVal
Q9_EnaOpn_Val	IEC Interlock Enable Opening Switch Q9	
Q51_EnaOpn_Val	IEC Interlock Enable Opening Switch Q51	
Q52_EnaOpn_Val	IEC Interlock Enable Opening Switch Q52	
Q53_EnaOpn_Val	IEC Interlock Enable Opening Switch Q53	
OpTmh_stVal	Operation Time	Application/LPHD.ST.OpTmh.units.stVal
OpTmh_Q	Operation Time Quality Code	Application/LPHD.ST.OpTmh.q
Loc_stVal	True: Local, False: Remote	Dc1/CSWI.ST.Loc.stVal
Loc_Q	True: Local, False: Remote Quality Code	Dc1/XSWI.ST.Loc.q
Q0_OpCnt_Val	Q0 Switch Count	Dc1/XSWI.ST.OpCnt.stVal
Q0_OpCnt_q	Q0 Switch Count quality code	Dc1/XSWI.ST.OpCnt.q
Q1_OpCnt_Val	Q1 Switch Count	Dc2/XSWI.ST.OpCnt.stVal
Q1_OpCnt_q	Q1 Switch Count quality code	Dc2/XSWI.ST.OpCnt.q
Q2_OpCnt_Val	Q2 Switch Count	Dc3/XSWI.ST.OpCnt.stVal
Q2_OpCnt_q	Q2 Switch Count quality code	Dc3/XSWI.ST.OpCnt.q
Q8_OpCnt_Val	Q8 Switch Count	CB1/XSWI.ST.OpCnt.stVal
Q8_OpCnt_q	Q8 Switch Count quality code	CB1/XSWI.ST.OpCnt.q
Q9_OpCnt_Val	Q9 Switch Count	
Q9_OpCnt_q	Q9 Switch Count quality code	
Q51_OpCnt_Val	Q51 Switch Count	
Q51_OpCnt_q	Q51 Switch Count quality code	
Q52_OpCnt_Val	Q52 Switch Count	
Q52_OpCnt_q	Q52 Switch Count quality code	
Q53_OpCnt_Val	Q53 Switch Count	
Q53_OpCnt_q	Q53 Switch Count quality code	
Beh_stVal	Device behavior: 1 = on; 2 = Locked; 3 = Test; 4 = Test/Locked; 5 = Off	Application/LLN0.ST.Beh.stVal
Beh_Q	Device Behavior Quality Code	Application/CALH.ST.Beh.q
Health_stVal	Device Health	Application/LLN0.ST.Health.stVal
Health_Q	Device Health Quality Code	Application/CALH.ST.Health.q
A_phsA_cVal_mag_f	Current Phase A	VI3p1_OperationalValues/MMXU.MX.A.phsA.f
A_phsA_q	Current Phase A Quality Code	VI3p1_OperationalValues/MMXU.MX.A.phsA.q
A_phsB_cVal_mag_f	Current Phase B	VI3p1_OperationalValues/MMXU.MX.A.instCVal.f
A_phsB_q	Current Phase B Quality Code	VI3p1_OperationalValues/MMXU.MX.A.phsA.q
A_phsC_cVal_mag_f″	Current Phase C	VI3p1_OperationalValues/MMXU.MX.A.mag.f
A_phsC_q	Current Phase C Quality Code	VI3p1_OperationalValues/MMXU.MX.A.phsA.q
A_neut_mag_f	Current neut	VI3p1_OperationalValues/MMXU.MX.A.cVal.f
A_neut_q	Current neut Quality Code	VI3p1_OperationalValues/MMXU.MX.A.mag.q
Hz_mag_f	Frequency	VI3p1_OperationalValues/MMXU.MX.Hz.instMag.f
Hz_q	Frequency Quality Code	VI3p1_OperationalValues/MMXU.MX.Hz.mag.q
PPV_phsAB_cVal_mag_f	Phase Voltage A-B	VI3p1_OperationalValues/MMXU.MX.PPV.phsAB.f
PPV_phsAB_q	Phase Voltage A-B Quality Code	VI3p1_OperationalValues/MMXU.MX.PPV.phsAB.q
PPV_phsBC_cVal_mag_f	Phase Voltage B-C	VI3p1_OperationalValues/MMXU.MX.PPV.instCVal.f
PPV_phsBC_q	Phase Voltage B-C Quality Code	VI3p1_OperationalValues/MMXU.MX.PPV.phsAB.q
PPV_phsCA_cVal_mag_f	Phase Voltage C-A	VI3p1_OperationalValues/MMXU.MX.PPV.mag.f
PPV_phsCA_q	Phase Voltage C-A Quality Code	VI3p1_OperationalValues/MMXU.MX.PPV.phsAB.q
PhV_phsA_cVal_mag_f	Voltage phase A	VI3p1_OperationalValues/MMXU.MX.PhV.phsA.f
PhV_phsA_q	Voltage phase A quality code	VI3p1_OperationalValues/MMXU.MX.PhV.phsA.q
PhV_phsB_cVal_mag_f	Voltage phase B	VI3p1_OperationalValues/MMXU.MX.PhV.instCVal.f
PhV_phsB_q	Voltage phase B quality code	VI3p1_OperationalValues/MMXU.MX.PhV.phsA.q
PhV_phsC_cVal_mag_f	Voltage phase C	VI3p1_OperationalValues/MMXU.MX.PhV.mag.f
PhV_phsC_q	Voltage phase C quality code	VI3p1_OperationalValues/MMXU.MX.PhV.phsA.q
TotPF_mag_f	Total Power Factor	VI3p1_OperationalValues/MMXU.MX.TotPF.instMag.f
TotPF_q	Total Power Factor Quality Code	VI3p1_OperationalValues/MMXU.MX.TotPF.mag.q
TotP_mag_f	Total Active Power	VI3p1_OperationalValues/MMXU.MX.TotW.instMag.f
TotP_q	Total Active Power Quality Code	VI3p1_OperationalValues/MMXU.MX.TotW.mag.q
TotQ_mag_f	Total Reactive Power	VI3p1_OperationalValues/MMXU.MX.TotVAr.instMag.f
TotQ_q	Total Reactive Power Quality Code	VI3p1_OperationalValues/MMXU.MX.TotVAr.mag.q
TotS_mag_f	Total Apparent Power	VI3p1_OperationalValues/MMXU.MX.TotVA.instMag.f
TotS_q	Total Apparent Power Quality Code	VI3p1_OperationalValues/MMXU.MX.TotVA.mag.q
SupWh_actVal	Accumulated active energy towards busbar	
SupWh_q	Accumulated active energy towards busbar Quality Code	
SupVArh_actVal	Accumulated reactive energy towards busbar	
SupVArh_q	Accumulated reactive energy towards busbar Quality Code	
DmdWh_actVal	Accumulated active energy from busbar	
DmdWh_q	Accumulated active energy from busbar Quality Code	
Pos0_ctl	Command object for Q0	Dc2/CSWI..Pos.
Pos1_ctl	Command object for Q1	Dc2/CSWI..Pos.
Pos2_ctl	Command object for Q2	Dc2/XSWI..Pos.pulseConfig.
Pos8_ctl	Command object for Q8	Dc2/XSWI..Pos.pulseConfig.
Pos9_ctl	Command object for Q9	Dc2/XSWI..Pos.pulseConfig.
Pos51_ctl	Command object for Q51	Dc2/XSWI..Pos.pulseConfig.
Pos52_ctl	Command object for Q52	Dc3/CSWI..Pos.
Pos53_ctl	Command object for Q53	Dc3/CSWI..Pos.
PTOC1_Op	51 Overcurrent Trip	VI3p1/PTRC.ST.Op.general
PTOC1_Str	51 Overcurrent picked up	VI3p1/PTRC.ST.Str.general

Table A1. *Cont.*

Object Name [37–39]	Meaning [37–39]	Automatically Mapped and Generated Addresses
PTOC2_Op	51N Overcurrent Trip	VI3p1/GAPC.ST.Op.general
PTOC2_Str	51N Overcurrent picked up	VI3p1/PHAR.ST.Str.general
PTOC3_Op	67-TOC Overcurrent Trip	VI3p1_5051OC3phase1/PTRC.ST.Op.general
PTOC3_Str	67-TOC Overcurrent picked up	VI3p1/GAPC.ST.Str.general
PTOC4_Op	67N-TOC Overcurrent Trip	VI3p1_5051OC3phase1/PTOC.ST.Op.general
PTOC4_Str	67N-TOC Overcurrent picked up	VI3p1_5051OC3phase1/PTRC.ST.Str.general
PTOC5_Op	46-TOC Overcurrent Trip	VI3p1_5051OC3phase1/PTOC.ST.Op.general
PTOC5_Str	46-TOC Overcurrent picked up	VI3p1_5051OC3phase1/PTOC.ST.Str.general
PTOC6_Op	50-1 Overcurrent I > Trip	VI3p1_5051OC3phase1/PTOC.ST.Op.general
PTOC6_Str	50-1 Overcurrent I > picked up	VI3p1_5051OC3phase1/PTOC.ST.Str.general
PTOC7_Op	50-2 Overcurrent I >> Trip	VI3p1_5051NOCgndB1/PTRC.ST.Op.general
PTOC7_Str	50-2 Overcurrent I >> picked up	VI3p1_5051OC3phase1/PTOC.ST.Str.general
PTOC8_Op	50N-1 Overcurrent IE > Trip	VI3p1_5051NOCgndB1/PTOC.ST.Op.general
PTOC8_Str	50N-1 Overcurrent IE > picked up	VI3p1_5051NOCgndB1/PTRC.ST.Str.general
PTOC9_Op	50N-2 Overcurrent IE >> Trip	VI3p1_5051NOCgndB1/PTOC.ST.Op.general
PTOC9_Str	50N-2 Overcurrent IE >> picked up	VI3p1_5051NOCgndB1/PTOC.ST.Str.general
PTOC10_Op	67-1 Directional Overcurrent I > Trip	VI3p1_5051NOCgndB1/PTOC.ST.Op.general
PTOC10_Str	67-1 Directional Overcurrent I > picked up	VI3p1_5051NOCgndB1/PTOC.ST.Str.general
PTOC11_Op	67-2 Directional Overcurrent I >> Trip	VI3p1_SwitchOntoFault/PTRC.ST.Op.general
PTOC11_Str	67-2 Directional Overcurrent I >> picked up	VI3p1_5051NOCgndB1/PTOC.ST.Str.general
PTOC12_Op	67-2 Directional Overcurrent I >> picked up	VI3p1_SwitchOntoFault/RSOF.ST.Op.general
PTOC12_Str	67N-1 Directional Overcurrent IE > picked up	VI3p1_SwitchOntoFault/PTRC.ST.Str.general
PTOC13_Op	67N-2 Directional Overcurrent IE >> Trip	CB1/PTRC.ST.Op.general
PTOC13_Str	67N-2 Directional Overcurrent IE >> picked up	VI3p1_SwitchOntoFault/RSOF.ST.Str.general
PTRC1_Tr	Trip signal for CB	CB1/PTRC.ST.Tr.general
PTRC1_Str	Trigger signal for CB	CB1/PTRC.ST.Str.general
PTRC2_Op	Trip signal for CB	
PTRC2_Str	Trigger signal for CB	
PTRC3_Op	Trip signal for CB	
PTRC3_Str	Trigger signal for CB	
RBRF1_OpEx	Breaker Failure—External Trip	
RBRF1_OpIn	Breaker Failure—Bay Internal Trip	
RBRF1_Str	Breaker Failure detected	
PDIF1_Op	Differential protection IDIFF > Trip	
PDIF1_Str	Differential protection IDIFF > picked up	
PDIF2_Op	Differential protection IDIFF >> Trip	
PDIF2_Str	Differential protection IDIFF >> picked up	
PDIS1_Op	Impedance protection Z1 Trip	
PDIS1_Str	Impedance protection Z1 picked up	
PDIS2_Op	Impedance protection Z2 Trip	
PDIS2_Str	Impedance protection Z2 picked up	
PDIS3_Op	Impedance protection Z1B Trip	
PDIS3_Str	Impedance protection Z1B picked up	
PDIS4_Op	Impedance protection general Trip	
PDIS4_Str	Impedance protection general picked up	
PDUP1_Op	Underexcitation protection Characterisitic 1 trip	
PDUP2_Op	Underexcitation protection Charactristic 2 trip	
PDUP3_Op	Underexcitation protection Characteristic 3 trip	
PTOV1_Op	Overvoltage U> trip	
PTOV1_Str	Overvoltage U > picked up	
PTOV2_Op	Overvoltage U >> trip	
PTOV2_Str	Over voltage U >> picked up	
PTUF1_Op	Frequency 1 under range trip	
PTUF1_Str	Frequency 1 under range picked up	
PTUF2_Op	Frequency 2 under range trip	
PTUF2_Str	Frequency 2 under range picked up	
PTUV1_Op	Under voltage protection U < trip	
PTUV1_Str	Under voltage protection U < picked up	
PTUV2_Op	Under voltage protection U << trip	
PTUV2_Str	Under voltage protection U << picked up	
PVOC2_Op	Voltage controlled overcurrent protection	
PDOP1_Op	Directional over power protection	
GAPC1_Op	External trip	
GAPC1_Str	External trip picked up	

Appendix B. Generated TRAFO Object

Table A2. Pattern object of a TRAFO and the mapped IEC 61850 Data Attribute addresses of the IED 7UT86.

Object Name [37–39]	Meaning [37–39]	Automatically Mapped and Generated Addresses
Loc	Operation Mode (1: Local, 0: Remote)	CB1/XCBR.ST.Loc.stVal
Error	alarm active	
Warning	warning active	Application/CALH.ST.GrWrn.stVal
PTOC1_Op	Overcurrent I > Trip	PTS1/PTRC.ST.Op.general
PTOC1_Str	Overcurrent I > picked up	PTS1/PTRC.ST.Str.general
PTOC2_Op	Overcurrent I >> Trip	PTS1/PTTR.ST.Op.general

Table A2. *Cont.*

Object Name [37–39]	Meaning [37–39]	Automatically Mapped and Generated Addresses
PTOC2_Str	Overcurrent I >> picked up	PTS1/PTTR.ST.Str.general
PTOC3_Op	Overcurrent Ip Trip	PTS1/PDIF.ST.Op.general
PTOC3_Str	Overcurrent Ip picked up	PTS1/PDIF.ST.Str.general
PTOC7_Op	Overcurrent 3I0 > Trip	PTS2/PTRC.ST.Op.general
PTOC7_Str	Overcurrent 3I0 > picked up	PTS1/PHAR.ST.Str.general
PTOC8_Op	Overcurrent 3I0 >> Trip	PTS2_5051OC3phA1/PTRC.ST.Op.general
PTOC8_Str	Overcurrent 3I0 >> picked up	PTS2/PTRC.ST.Str.general
PTOC9_Op	Overcurrent 3I0p Trip	PTS2_5051OC3phA1/PTOC.ST.Op.general
PTOC9_Str	Overcurrent 3I0p picked up	PTS2/PHAR.ST.Str.general
PTOC12_Op	Unbalanced Load I > Trip	PTS2_5051OC3phA1/PTOC.ST.Op.general
PTOC12_Str	Unbalanced Load I > picked up	PTS2_5051OC3phA1/PTRC.ST.Str.general
PTOC13_Op	Unbalanced Load I >> Trip	PTS2_5051OC3phA1/PTOC.ST.Op.general
PTOC13_Str	Unbalanced Load I >> picked up	PTS2_5051OC3phA1/PTOC.ST.Str.general
PTOC15_Op	Unbalanced Load I2thTrip	PTD1/PTRC.ST.Op.general
PTOC15_Str	Unbalanced Load I2th picked up	PTS2_5051OC3phA1/PTOC.ST.Str.general
PTTR1_Op	Overload Theta Trip	PTD1_87TrafoDiffProt1/PDIF.ST.Op.general
PTTR1_Str	Overload Theta picked up	PTS2_5051OC3phA1/PTOC.ST.Str.general
PDIF1_Op_R24	Differential protection IDIFF > Trip	PTD1_87TrafoDiffProt1/PDIF.ST.Op.general
PDIF1_Str_R25	Differential protection IDIFF > picked up	PTD1/PTRC.ST.Str.general
PDIF2_Op_R26	Differential protection IDIFF >> Trip	PTD1_87TrafoDiffProt1/PTRC.ST.Op.general
PDIF2_Str_R27	Differential protection IDIFF >> picked up	PTD1_87TrafoDiffProt1/PDIF.ST.Str.general
PDIF3_Op_R28	Differential protection I REF Trip	PTE1/PTRC.ST.Op.general
PDIF3_Str_R29	Differential protection I REF picked up	PTD1_87TrafoDiffProt1/PDIF.ST.Str.general
GAPC1_Op_R30	External trip	PTE1_50N51NOC1phA1/PTRC.ST.Op.general
GAPC1_Str_R31	External trigger	PTD1_87TrafoDiffProt1/PTRC.ST.Str.general
RBRF1_OpEx_R32	Breaker Failure—External Trip	CB1/RBRF.ST.OpEx.general
RBRF1_OpIn_R33	Breaker Failure—Bay Internal Trip	CB1/RBRF.ST.OpIn.general
RBRF1_Str_R34	Breaker Failure detected	PTE1/PTRC.ST.Str.general
PTRC1_Op_R35	Trip signal for CB	PTE1_50N51NOC1phA1/PTOC.ST.Op.general
PTRC1_Str_R36	Trigger signal for CB	PTE1_50N51NOC1phA1/PTRC.ST.Str.general
Beh_R1	Device Behaviour	Application/LLN0.ST.Beh.stVal
Health_R2	Device Health	Application/LLN0.ST.Health.stVal
OpCnt_R1	Switch Count	CB1/XCBR.ST.OpCnt.stVal
OpTmh_R2	Operation Time	Application/LPHD.ST.OpTmh.units.stVal
IP_A_P1	Primary Phase Current A	PTS1_OperationalValues/MMXU.MX.A.phsA.f
IP_B_P2	Primary Phase Current B	PTS1_OperationalValues/MMXU.MX.A.phsA.f
IP_C_P3	Primary Phase Current C	PTS1_OperationalValues/MMXU.MX.A.units.f
IS_A_P4	Secondary Phase Current A	PTS1_OperationalValues/MMXU.MX.A.phsB.f
IS_B_P5	Secondary Phase Current B	PTS1_OperationalValues/MMXU.MX.A.phsB.f
IS_C_P6	Secondary Phase Current C	PTS1_OperationalValues/MMXU.MX.A.units.f
IS2_A_P7	Secondary 2 Phase Current A	
IS2_B_P8	Secondary 2 Phase Current B	
IS2_C_P9	Secondary 2 Phase Current C	
f	Frequency	PTS1_OperationalValues/MMXU.MX.Hz.units.f
U_AB_P11	Phase Voltage A-B	PTS1_OperationalValues/MMXU.MX.PPV.phsAB.f
U_BC_P12	Phase Voltage B-C	PTS1_OperationalValues/MMXU.MX.PPV.phsAB.f
U_CA_P13	Phase Voltage C-A	PTS1_OperationalValues/MMXU.MX.PPV.units.f
U_n	Voltage neut	
TotPF_P15	Total Power Factor	PTS1_OperationalValues/MMXU.MX.TotPF.f
TotP_P16	Total Active Power	
TotQ_P17	Total Reactive Power	PTS1_OperationalValues/MMXU.MX.TotVAr.units.f
TotS	Total Apparent Power	PTS1_OperationalValues/MMXU.MX.TotVA.units.f
Tmp1_P19	Temperature 1	
Tmp2_P20	Temperature 2	
Tmp3_P21	Temperature 3	
TmpMax_P22	Temperatrue Max	
L_RES_Str_P23	Load Reserve Warning	PTE1_50N51NOC1phA1/PTOC.ST.Str.
L_RES_Op_P24	Load Reserve Alarm	PTE1_50N51NOC1phA1/PTOC.ST.Op.
ALT_RATE_P25	Altering Rate	
SupWh_P1	Accumulated active energy towards busbar	
SupVArh_P2	Accumulated reactive energy towards busbar	
DmdWh_P3	Accumulated active energy from busbar	
DmdVArh_P4	Accumulated reactive energy from busbar	

Appendix C. Test Results

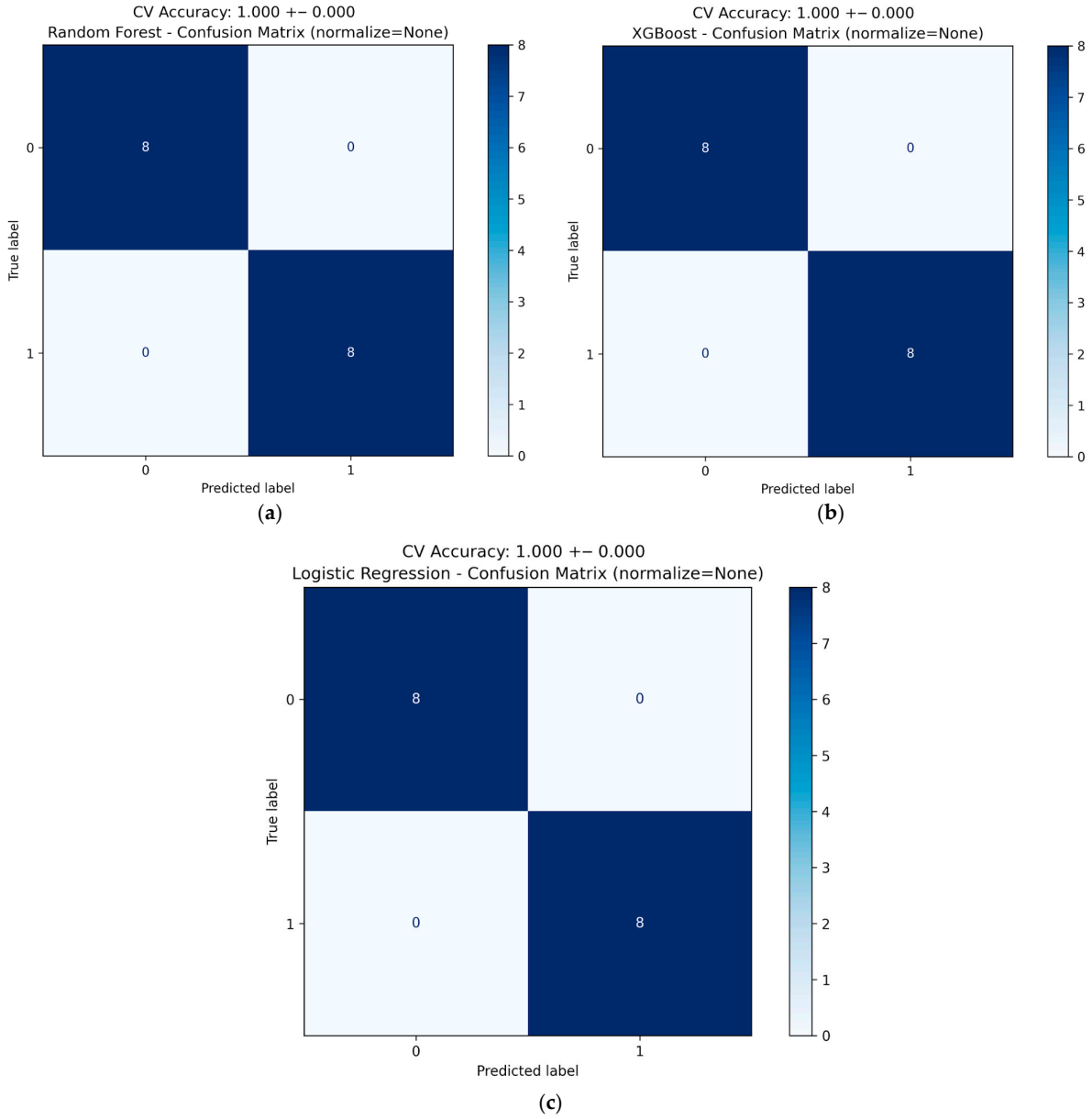


Figure A1. Confusion Matrix generated using: (**a**) Random Forest; (**b**) XGBoost; (**c**) Logistic Regression. The symbol "+ −" in this picture represents "±".

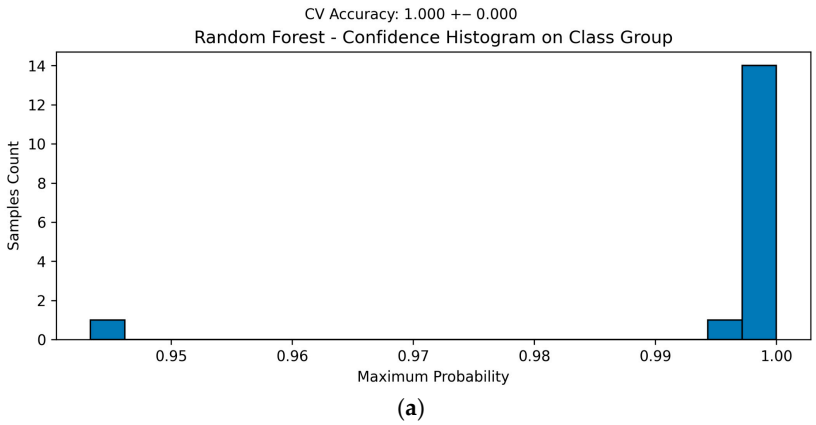
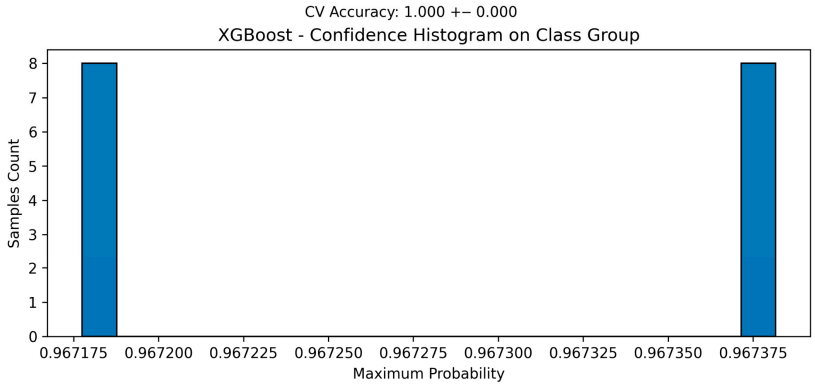
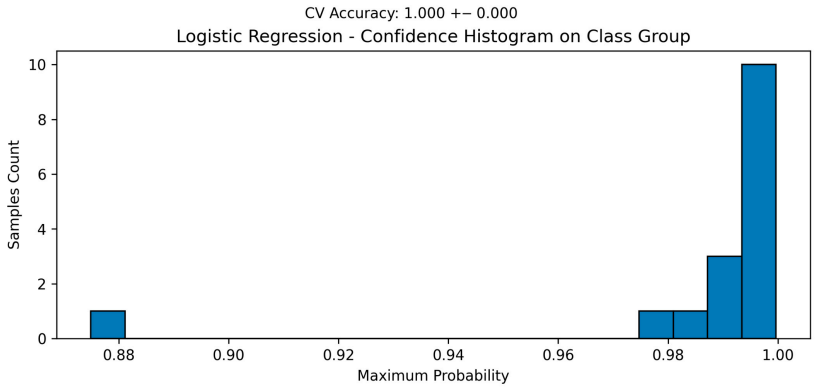


Figure A2. *Cont.*

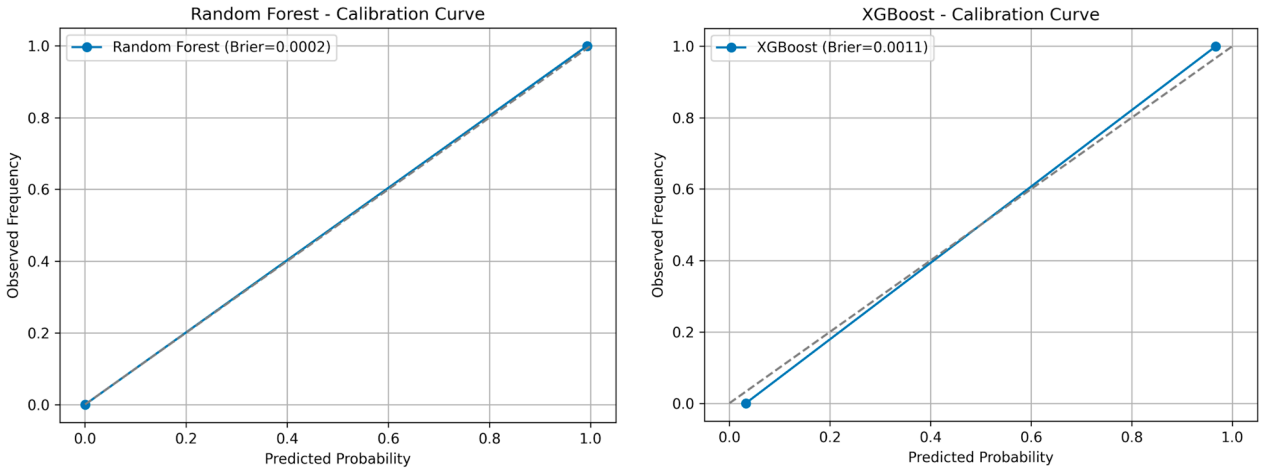


(**b**)



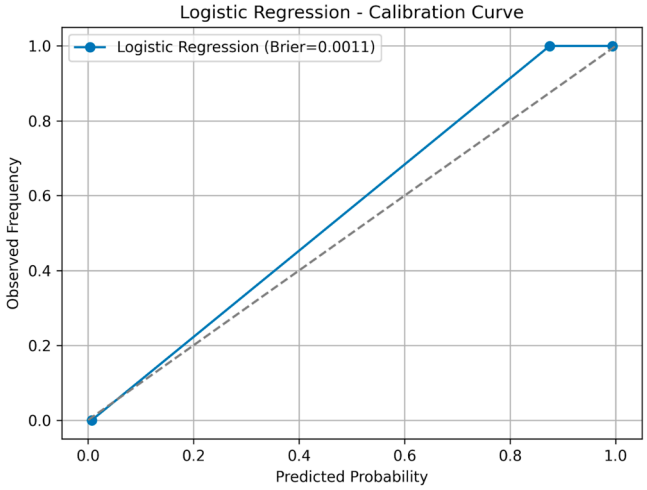
(**c**)

Figure A2. Confidence Histogram generated using: (**a**) Random Forest; (**b**) XGBoost; (**c**) Logistic Regression. The symbol "+ −" in this picture represents "±".



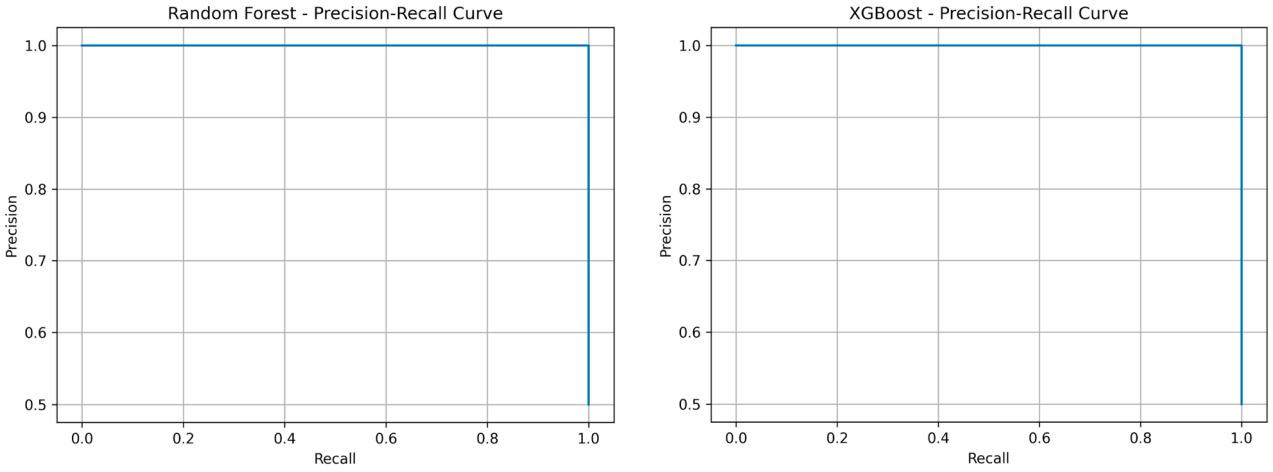
(**a**) (**b**)

Figure A3. *Cont.*



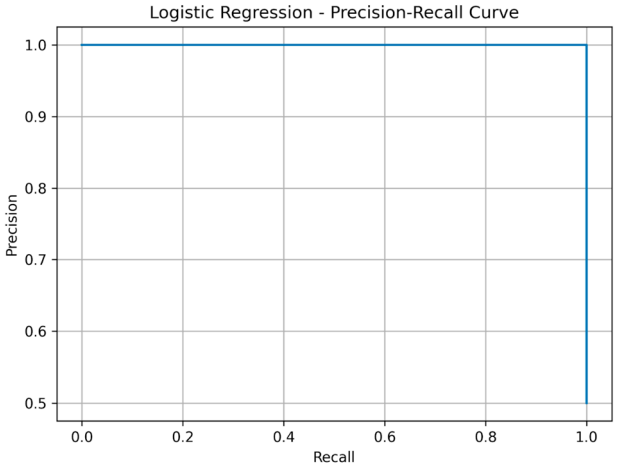
(**c**)

Figure A3. Calibration Curve generated using: (**a**) Random Forest; (**b**) XGBoost; (**c**) Logistic Regression. In this figure, the dashed gray line represents perfect calibration (the ideal case).



(**a**)

(**b**)



(**c**)

Figure A4. Precision–Recall Curve generated using: (**a**) Random Forest; (**b**) XGBoost; (**c**) Logistic Regression.

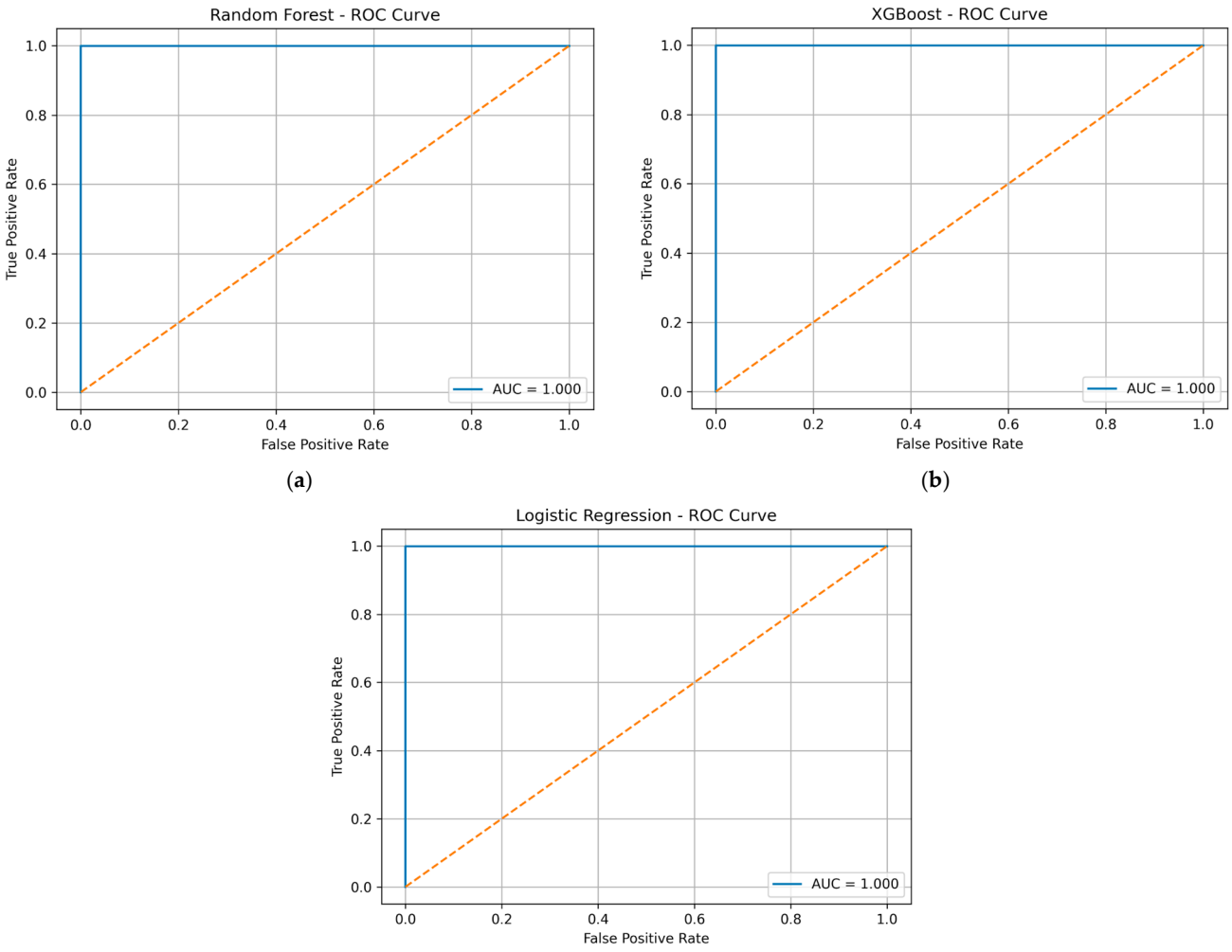


Figure A5. ROC Curve generated using: (**a**) Random Forest; (**b**) XGBoost; (**c**) Logistic Regression. In this figure, the dashed orange diagonal line represents the reference for random performance.

References

1. Ali, N.H.; Borhanuddin, M.A.; Othman, L.M.; Abdel-Latif, K.M. Performance of communication networks for Integrity protection systems based on travelling wave with IEC 61850. *Electr. Power Energy Syst.* **2018**, *95*, 664–675. [CrossRef]
2. Lozano, J.C.; Koneru, K.; Ortiz, N.; Cardenas, A.A. Digital Substations and IEC 61850: A Primer. *IEEE Commun. Mag.* **2023**, *61*, 28–34. [CrossRef]
3. Habib, H.F.; Fawzy, N.; Brahma, S. Performance Testing and Assessment of Protection Scheme Using Real-Time Hardware-in-the-Loop and IEC 61850 Standard. *IEEE Trans. Ind. Appl.* **2021**, *57*, 4569–4578. [CrossRef]
4. Elbez, G.; Keller, H.B.; Hagenmeyer, V. A Cost-efficient Software Testbed for Cyber-Physical Security in IEC 61850-based Substations. In Proceedings of the 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), Aalborg, Denmark, 29–31 October 2018; pp. 1–6. [CrossRef]
5. Da Cruz, A.K.; Lechner, D.; Siemers, C.; Kellermann, A.C.H.; Däubler, L. Novel approaches for the integration of the electrical substation IEC 61850 bay level in modern Industry 4.0 SCADA and Edge Systems based on OPC UA, TSN and MTP Standards. *Cad. Pedagógico* **2025**, *22*, 15221. [CrossRef]
6. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
7. Chen, T.; Guestrin, C. XGBoost: A scalable Tree Boosting System. In Proceedings of the KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13 August 2016; pp. 785–794. [CrossRef]
8. Reiche, L.; Fay, A. Concept for extending the Module Type Package with energy management functionalities. In Proceedings of the 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), Stuttgart, Germany, 6–9 September 2022; pp. 1–8. Available online: https://ieeexplore.ieee.org/document/9921612 (accessed on 15 November 2025).

9. Process INDUSTRIE 4.0: The Age of Modular Production on the Doorstep to Market Launch. 2019. Available online: https://share.google/fzMcdmcAXUZir0k0s (accessed on 21 November 2025).
10. Khan, A. Simulating Intelligence. In *Artificial Intelligence: A Guide for Everyone*; Springer Nature: Cham, Switzerland, 2024; pp. 105–114. [CrossRef]
11. Jaboob, A.; Durrah, O.; Chakir, A. Artificial Intelligence: An Overview. In *Engineering Applications of Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2024; pp. 3–22. [CrossRef]
12. Stryker, C.; Kavlakoglu, E. What is Artificial Intelligence (AI)? 2024. Available online: https://www.ibm.com/think/topics/artificial-intelligence (accessed on 18 November 2025).
13. Khan, A. AI Subfields. In *Artificial Intelligence: A Guide for Everyone*; Springer Nature: Cham, Switzerland, 2024; pp. 179–195. [CrossRef]
14. Sheikn, H.; Prins, C.; Schrijvers, E. Artificial Intelligence: Definition and Background. In *Mission AI*; Springer International Publishing: Cham, Switzerland, 2023; pp. 15–41. [CrossRef]
15. Witten, I.H.; Frank, E.; Hall, M.A.; Pal, C.J. *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed.; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2016; ISBN 978-0-12-804291-5.
16. Kavlakoglu, E. What Is a Decision Tree? 2024. Available online: https://www.ibm.com/de-de/think/topics/decision-trees (accessed on 15 December 2025).
17. Brunner, C. IEC 61850 for power system communication. In Proceedings of the IEEE/PES Transmission and Distribution Conference and Exposition, Chicago, IL, USA, 21–24 April 2008; pp. 1–6. Available online: https://ieeexplore.ieee.org/document/4517287 (accessed on 15 November 2025).
18. Aftab, M.A.; Hussain, S.S.; Ali, I.; Ustun, T.S. IEC 61850 based substation automation system: A survey. *Int. J. Electr. Power Energy Syst.* **2020**, *120*, 106008. [CrossRef]
19. Cavalieri, S. Semantic Interoperability between IEC 61850 and oneM2M for IoT-Enabled Smart Grids. *Sensors* **2021**, *21*, 2571. [CrossRef] [PubMed]
20. Huang, Z.; Gao, L.; Yang, Y. IEC 61850 Standards and Configuration Technology. In *IEC 61850-Based Smart Substations*; Academic Press: Oxford, UK, 2019; pp. 25–62. [CrossRef]
21. *IEC 61850-6*; Communication Networks and Systems for Power Utility Automation—Part 6: Configuration Description Language for Communication in Power Utility Automation Systems Related to IEDs. IEC: Geneva, Switzerland, 2018; ISBN 978-2-8322-5802-6.
22. Hou, S.; Liu, W.; Kong, F. IEC 61580 Gateway SCL Configuration Document Research. In Proceedings of the 2013 Third International Conference on Intelligent System Design and Engineering Applications, Hong King, China, 16–18 January 2013; pp. 824–831. Available online: https://ieeexplore.ieee.org/abstract/document/6455739 (accessed on 15 November 2025).
23. Della Giustina, D.; Sotomayor, A.A.; Dedè, A.; Ramos, F. A Model-Based Design of Distributed Automation Systems for the Smart Grid: Implementation and Validation. *Energies* **2020**, *13*, 3560. [CrossRef]
24. Stock, S.; Babazadeh, D.; Becker, C. Applications of Artificial Intelligence in Distribution Power System Operation. *IEEE Access* **2021**, *9*, 150098–150119. Available online: https://ieeexplore.ieee.org/document/9599712 (accessed on 15 November 2025). [CrossRef]
25. Alam, M.M.; Hossain, M.J.; Habib, M.A.; Arafat, M.Y.; Hannan, M.A. Artificial intelligence integrated grid systems: Technologies, potential frameworks, challenges, and research directions. *Renew. Sustain. Energy Rev.* **2025**, *211*, 115251. [CrossRef]
26. Ahmadi, M.; Aly, H.; Gu, J. A comprehensive review of AI-driven approaches for smart grid stability and reliability. In *Renewable and Sustainable Energy Reviews*; Elsevier: Amsterdam, The Netherlands, 2026; p. 116424. [CrossRef]
27. Saranavan, R.; Sujatha, P. A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification. In Proceedings of the 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 14–15 June 2018; IEEE: New York, NY, USA, 2018; pp. 945–949. Available online: https://ieeexplore.ieee.org/document/8663155 (accessed on 15 November 2025).
28. N Rincy, T.; Gupta, R. A Survey on Machine Learning Approaches and Its Techniques. In Proceedings of the 2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), Bhopal, India, 22–23 February 2020; pp. 1–6. Available online: https://ieeexplore.ieee.org/document/9087123/authors (accessed on 15 November 2025).
29. 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. Available online: https://dl.acm.org/doi/10.5555/1953048.2078195 (accessed on 15 November 2025).
30. McKinney, W. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference (SciPy), Austin, TX, USA, 28 June–3 July 2010; pp. 56–61. [CrossRef]
31. xml.etree.ElementTree—The ElementTree XML API. Available online: https://docs.python.org/3/library/xml.etree.elementtree.html (accessed on 26 November 2025).

32. Harris, C.R.; Millman, K.J.; van der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; et al. Array programming with NumPy. *Nature* **2020**, *585*, 357–362. [CrossRef] [PubMed]
33. Hunter, J.D. Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [CrossRef]
34. Waskom, M.L. Seaborn: Statistical Data Visualization. *J. Open Source Softw.* **2021**, *6*, 3021. [CrossRef]
35. PEP 8—Style Guide for Python Code. Available online: https://peps.python.org/pep-0008 (accessed on 26 November 2025).
36. *IEC 61850-7-4*; Communication Networks and Systems for Power Utility Automation—Part 7-4: Basic Communication Structure—Compatible Logical Node Classes and Data Object Classes. IEC: Geneva, Switzerland, 2020; ISBN 978-2-8322-7890-1.
37. Process Control System PCS 7 PowerControl Library Objects. Available online: https://cache.industry.siemens.com/dl/files/788/109799788/att_1079692/v1/PCS7_PowerControlObjects_en.pdf (accessed on 26 November 2025).
38. IEC 61850 Station Gateway for TIA Portal. Available online: https://support.industry.siemens.com/cs/ww/de/view/109820678 (accessed on 26 November 2025).
39. Advanced Power Control Library for SIMATIC WinCC OA. Available online: https://support.industry.siemens.com/cs/ww/en/view/109995244 (accessed on 26 November 2025).