


Predictive Fraud Detection: An Intelligent Method for Internet of Smart Grid Things Systems


Lucas Bastos   [Federal University of Pará | lucas.bastos@itec.ufpa.br]


Bruno Martins  [Federal University of Pará | bruno.martins@itec.ufpa.br]

Hugo Santos  [Federal University of Pará | hugosantos@ufpa.br]

Iago Medeiros  [Federal University of Pará | iago.medeiros@itec.ufpa.br]

Paulo Eugênio  [Federal University of Rio Grande do Norte | paulo.filho.071@ufrn.edu.br]

Leonardo Marques  [Federal University of Rio Grande do Norte | leonardo.augusto.103@ufrn.edu.br]


Denis Rosário  [Federal University of Pará | denis@ufpa.br]

Eduardo Nogueira  [Federal University of Rio Grande do Norte | eduardo.nogueira@ufrn.br]

Eduardo Cerqueira  [Federal University of Pará | cerqueira@ufpa.br]

Márcio Kreutz  [Federal University of Rio Grande do Norte | marcio.kreutz@ufrn.br]

Augusto Neto  [Federal University of Rio Grande do Norte | augusto@dimap.ufrn.br]

 UFPA - R. Augusto Corrêa, 01 - Guamá, Belém - PA, 66075-110.

Received: 26 December 2022 • Accepted: 02 July 2023 • Published: 06 November 2023

Abstract Today, grid resilience as a feature has become non-negotiable, significantly when power interruptions can impact the economy. The widespread popularity of Intelligent Electronic Devices (IED) operating as smart meters enables an immense amount of fine-grained electricity consumption data to be collected. However, risks can still exist in the Smart Grid (SG), as valuable data are exchanged among SG systems; theft or alteration of this data could violate consumer privacy. The Internet of Things for Smart Grid (IoSGT) is a promising ecosystem of different technologies that coordinate with each other to pave the way for new SG applications and services. As a use case of IoSGT for future SG applications and services, fraud detection, Non-technical losses (NTL), emerges as an important application for Smart Grid (SG) scenarios. A substantial amount of electrical energy is lost throughout the distribution system, and these losses are divided into two types: technical and non-technical. Non-technical losses (NTL) are any electrical energy consumed and not invoiced. They may occur due to illegal connections, issues with energy meters such as delay in the installation or reading errors, contaminated, defective, or non-adapted measuring equipment, very low valid consumption estimates, faulty connections, and disregarded customers. Non-technical losses are the primary cause of revenue loss in the SG. According to a recent study, electrical utilities lose \$89.3 Billion per year due to non-technical losses. This article proposes ensemble predictor-based time series classifiers for NTL detection. The proposed predictor ministers the user's energy consumption as a data input for classification, from splitting the data to executing the classifier. It encompasses the temporal aspects of energy consumption data during preprocessing, training, testing, and validation stages. The suggested predictor is Time Series (TS) oriented, from data splitting to the classifier's performance. Overall, our best results have been recorded in the fraud detection-based time series classifiers (TSC) model scoring an improvement in the empirical performance metrics by 10% or more over the other developed models.

Keywords: Smart Grid, Non-technical losses, Ensemble, TSC, Edge Computing

1 Introduction

The main benefits expected from the Smart Grid (SG) are increasing grid resilience and improving environmental performance Gunturi and Sarkar [2021]. Today, grid resilience as a feature has become non-negotiable, significantly when power interruptions can potentially impact the economy. SG promises to provide flexibility and reliability by enabling additional dispersed power supply, facilitating the integration of new resources into the grid, and enabling corrective capabilities when failures occur Wang *et al.* [2019]. However, risks can still exist in the SG since any interruptions in power generation could disturb SG stability and potentially have considerable socio-economic impacts.

The widespread popularity of Intelligent Electronic De-

vice (IED) as smart meters enables an immense amount of fine-grained electricity consumption data to be collected, which are transmitted across network connections in the form of telemetry data Modesto *et al.* [2022]. Billing is no longer the only function of IEDs since high-resolution data from IEDs provide rich information on the electricity consumption behaviors and lifestyles of the consumers Wang *et al.* [2019]. In addition, as valuable data are exchanged among SG systems, theft or alteration of this data could violate consumer privacy. Because of these weaknesses, the SG has become the primary target of attackers, which attracted the attention of government, industry, and academia Gunturi and Sarkar [2021].

To tackle the bottleneck of the Internet of Things (IoT), various new technologies are proposed and investigated in

academic and industrial areas Deng *et al.* [2022]; Meena *et al.* [2023]; Lobato *et al.* [2023]. In this context, secure versions of the IoT and extensive data systems are challenging and have become a research trend in the last decade. Internet of Things devices are known to possess limited storage, energy resources, and computational abilities Belhadi *et al.* [2022]. However, the Internet of Smart Grid Things (IoSGT) Santos *et al.* [2022] appear as a promising ecosystem of diverse technologies orchestrated to pave the way for new SG applications and services. Specifically, the cloud-native approach of the IoSGT advances across servers that are settled at both core (*cloud computing*) and edge (*edge computing*) facilities, establishing an *IoT-to-edge-to-cloud* continuum. Hence, the IoSGT accelerates SG automation and control by enabling functions to be executed at edge data centers, *i.e.*, close to IED, rather than in cloud computing where data must be sent to faraway centralized data centers (likely in a different country). Our findings from the IoSGT validation outcomes suggested that SG functions running at edge servers are proven to have improved performance than in central clouds (among other benefits) due to the high proximity between IEDs and computation, data storage, and SG control application capabilities.

As a use case of IoSGT for future SG applications and services, fraud detection, *i.e.*, Non-technical losses (NTL), appears as an important application for SG scenarios de Souza Savian *et al.* [2021]. Expressly, NTLs represent the electricity consumed but not billed to the user, which various illegal methods can cause, such as using a different billing system for the day, very low valid consumption estimates, and others Chuwa and Wang [2021]. Intruders may attack such a communication environment. This results in leaks and interruptions in data transfer. Data that is both stored and in transit must be protected Ramana *et al.* [2022]. NTL is a common phenomenon and a significant problem for electricity providers Zheng *et al.* [2017], where NTL is one of the main sources of economic loss for power distribution enterprises Ramos *et al.* [2018]. For instance, electrical utilities lose \$89.3 Billion per year due to NTLs. Developing countries like India and Brazil are each losing 42% and 8% of the total electricity produced annually due to energy theft Chuwa and Wang [2021]. Energy losses in developed countries range from 0.5% to 3% of annual revenues. Though the amount might seem small, financial losses in the United States alone are as high as up to 6 billion Chuwa and Wang [2021]. However, there is no recipe for developing effective NTL recognition techniques de Souza Savian *et al.* [2021].

Although adding categories to a classification problem adds complexity to the classification process, it is crucial to identify the types of fraud committed by end-users. For example, accurately detecting any fraud will allow companies to better identify the root causes of fraud and calculate its financial impact, develop new security measures, and improve the accuracy and efficiency of inspection teams. However, according to Chuwa and Wang [2021], existing NTL detection methods cannot effectively identify all types of fraud simultaneously. In addition to being challenging to classify fraud, distinguishing and detecting different types of fraud is more complex using unique Machine Learning (ML) techniques. In this context, ensemble learning is an ML tech-

nique where multiple predictors are trained to solve the same problem and combined to get better results Araujo *et al.* [2020]. For instance, ML models often perform not so well by themselves either because they have a high bias or variance, and the idea of ensemble methods combining several of them to create an aggregated learner (or ensemble model) that achieves better performances.

Ensemble predictors, *i.e.*, aggregated ML models, have become increasingly popular after showing excellent results in various research fields by combining the outputs of multiple ML algorithms to achieve better performance than a single classifier Araujo *et al.* [2020]. The integration is more heterogeneous by considering the different time series classifiers types to classify these different parameters and attributes found in the user class. In this case, ensemble predictors that consider the time-dependent nature of energy consumption data tend to improve the accuracy and efficiency of detecting fraud, which remains an open question Messinis and Hatziargyriou [2018].

This article proposes a time series classifiers (TSC)-based ensemble predictor for NTL detection as a use case for the IoSGT system, called HybridForest. In this context, the IoSGT system allows fraud detections in edge data centers close to users, rather than cloud computing, where data must be sent to distant centralized data centers (probably in a different country). Our findings from the IoSGT validation results suggested that SG functions running on edge servers demonstrably perform better than on core clouds (among other benefits) due to the high proximity between users and computing, data storage, and control application resources SG. hybridForest considers an ensemble predictor with different TSC classifiers, namely, time series Forest (TSF), Catch22, Weasel, k-NN for time series, and Arsenal. After training, the generated predictor classifies new data into thirteen classes (either honest or one of twelve different types of fraud), which facilitates the development of methods to improve the detection of specific types of fraud.

We evaluate the HybridForest (HybridForest) using smart energy data from the Irish Smart Energy Trial Archive [2012], which consists of approximately 4700 users with 535 days of sample data. Following the approach of existing work, we add twelve fraud types already defined in the literature to create a synthetic dataset of honest and fraudulent customers and randomly select the amount of fraud data generated among users. The results underscore the performance of the HybridForest over other state-of-the-art NTL predictors. For example, the HybridForest predictor classifies different types of high-performing fraudulent users, regardless of their category.

This article extends the previous work described in Santos *et al.* [2022], and its main research contributions include

1. Introduce an intelligent method for fraud detection as a use case implemented on the IoSGT system.
2. The HybridForest method classifies the data into thirteen classes (either honest or one of twelve different types of fraud).

This article is organized as follows. Section 2 analyzes the existing works about NTL detection. Section 3 describes

fraud detection based on the IoSGT system. Section 4 introduces an experiment analysis. Finally, Section 5 concludes the paper and presents future work suggestions.

2 Related Works

Previous studies have introduced techniques to classify NTL; therefore, the literature offers many ways to characterize and classify NTLs through models, algorithms, and techniques. Some of them follow the approach of using ML.

Passos Júnior *et al.* [2016] evaluated the optimum-path forest (OPF) clustering for NTL. This work considers two private datasets of a Brazilian electrical power company: one composed of commercial profiles and another managed to find industrial consumers. Another main contribution of such work is to model the problem of NTL identification as an anomaly detection task. The classifier is trained with regular consumers only.

Barja-Martinez *et al.* [2021] proposed a holistic analysis of ML applications in distribution power systems after identifying and classifying the different data-driven techniques for power systems and the data sources involved in the data acquisition. These applications include operation and monitoring, predictive maintenance, non-technical loss detection, forecasting, flexibility management, and planning of distribution grids.

Gunturi and Sarkar [2021] proposed an energy fraud detector based on ensemble classifiers, which uses real-time data from IEDs to analyze trends in user consumption behavior. The proposal uses Synthetic Minority Over-sampling Technique (SMOTE) re-sampling techniques to balance the minority class data and generate synthetic fraud data using the same work fraud case. After re-sampling, the data is normalized and trained. The authors used six ensemble algorithms to classify users, including ADaptive Boosting (ADB), CATegorical boosting (CAT), extreme gradient Boosting (XGB), Light Gradient Boosting (LGB), Random Forest (RF), and Extra Trees (ET). The proposal generated promising results, mainly when comparing ensemble predictors. Still, it did not consider the time series of the data when separating the training and testing, and the approach didn't use any TSC to compare with their classifiers.

Jokar *et al.* [2016] proposed a consumption pattern-based energy theft detector. This detector uses a Support Vector Machine (SVM) classifier to analyze each user's samples and classify them as either honest or fraudulent. However, the results were different from other ML algorithms that performed better. Punmiya and Choe [2019] considered three state-of-the-art gradient boosting algorithms, namely XGBoost, catBoost, and LightBoost, for NTL detection. The detector shows significantly better results compared to related methods. However, the authors did not use techniques to select hyperparameters for the classifier, nor did they use cross-validation techniques to split the data.

Belhadi *et al.* [2021b] developed a new framework based on privacy reinforcement learning to accurately identify anomalous patterns in a distributed and heterogeneous energy environment. The local outlier factor is first performed to derive the local simple abnormal patterns in each site of the

distributed energy platform. Reinforcement privacy learning is then established using blockchain technology to merge the local anomalous patterns into global complex abnormal patterns. Besides, different optimization strategies are suggested to improve the whole outlier detection process. To demonstrate the applicability of the proposed framework, intensive experiments have been carried out on the well-known CASAS (Center of Advanced Studies in Adaptive Systems) platform.

Zanetti *et al.* [2019] presented an approach to create a tunable profile-based by showing that using only a small set of recent measures to define a consumption pattern is possible. Fraud inspection is triggered when a discrepancy between the energy supplied by the grid and that registered by the meters is detected. Consumption reports registered shortly before and after the discrepancy detection are compared to detect fraud. Consumption patterns are created from consumption data using unsupervised learning clustering algorithms and a semiautomated feature extraction method. They called these patterns short-lived (SL) because they are expected to represent a consumer's behaviour for a short period, only enough to detect an ongoing fraud.

Yip *et al.* [2018] presented two anomaly detection schemes that adopt linear programming (LP) to overcome some problems associated with existing NTLs detection schemes. They design two anomaly detection schemes for detecting energy theft attacks against AMI and locating metering defects in smart grid environment regardless of whether they occur all the time or at varying rates during intermittent periods in a day; NTLs detection accuracy and reduce false positives by taking the impact of technical losses (TLs) and measurement noise on the detection framework into consideration. Two metrics, referred to as loss factor and the error term, are introduced for capturing the percentage of TLs and amount of measurement noise, respectively, in the service area, and; Investigate and generate a diverse set of NTLs attack functions such that they closely resemble real-world AMI energy frauds/metering defects scenarios.

Messinis *et al.* [2019] proposed a hybrid system for detecting NTLs consisting of three modules based on different principles. It ensures that a large variety of frauds can be detected. The first module (entitled SVM module) utilizes breakout detection and other features to detect frauds in yearly consumption time series by training an SVM. The second (SENS) module calculates voltage self-sensitivities from meter measurements and compares them to their theoretical values (extracted from network topology). Finally, the third module (entitled NTL-MIN module) solves an optimization problem where the objective is to minimize losses. The decision variables are consumers' active energy consumptions, while voltage magnitude measurements are introduced as constraints.

Our previous work presented in Santos *et al.* [2022] showed that the Internet of Smart Grid Things (IoSGT) appears as a promising ecosystem of diverse technologies orchestrated to pave the way for new SG applications and services. Specifically, the cloud-native approach of the IoSGT advances across servers that are settled at both core (*cloud computing*) and edge (*edge computing*) facilities, establishing an *IoT-to-edge-to-cloud* continuum. Hence, the IoSGT accelerates SG automation and control by enabling functions

Table 1. Summary of Related Work

Works	Split for TS	Techniques	Ensemble Predictor	Data Type	TSC	# of Classes
Passos Júnior <i>et al.</i> [2016]	No	OPF, k-Means, GMM, AP, Birch, SVM	No	Electricity Consumption	No	2
Barja-Martinez <i>et al.</i> [2021]	No	Correlation, DT, LR, LogR, MLP, DNN	No	Customer behavior, Operational, Weather, Social Media, GIS, Electricity Consumption	No	2
Gunturi and Sarkar [2021]	No	ADB, CAT, XGB, LGB, RF, ET	Yes	Electricity Consumption	No	2
Jokar <i>et al.</i> [2016]	No	SVM	No	Electricity Consumption	No	2
Belhadi <i>et al.</i> [2021b]	No	CASAS	No	Electricity Consumption	No	2
Messinis <i>et al.</i> [2019]	No	SVM	No	Voltage	No	2
Zanetti <i>et al.</i> [2019]	Yes	FDS	No	Electricity Consumption	No	2
Yip <i>et al.</i> [2018]	Yes	Linear Programming (LP)	No	Electricity Consumption	No	2
Punmiya and Choe [2019]	No	XGBoost, CatBoost, LightBoost	No	Electricity Consumption	No	2
HybridForest	Yes	Signature, CatBoost, XGBoost, LightBGM, ROCKET, Weasel, TSF, SVM, Catch22, k-NN	Yes	Electricity Consumption	Yes	13

to be executed at edge data centers close to sgs rather than in cloud computing, where data must be sent to faraway centralized data centers (likely in a different country). Their findings from the IoSGT validation outcomes suggested that SG functions, such as, NTL detection, running at edge servers are proven to have improved performance than in central clouds (among other benefits) due to the high proximity between SG service and computation, data storage, and SG control application capabilities.

Table 1 summarizes the main characteristics of previous works aimed at NTL detection in terms of the split for time series, ML algorithms employed, ensemble predictor considered, data used for NTL prediction, TSC, and several output classes for NTL prediction. First, the ensemble enables different TSC types for classifying other classes to classify the parameters and properties in the user class. Second, a single classifier has many application dependencies, as they are limited to making binary classifications (either fraudulent or non-fraudulent) and cannot accurately classify different classes. NTL detection remains an open issue based on our analysis of state-of-the-art approaches. However, enabling NTL detection to be executed at edge data centers close SGs from IoSGT SG facilities be verified quickly. In this context, developing methods that compare different ML algorithms and maintain the temporal dependence of the data is paramount in providing results that better reflect real-life scenarios. Additionally, we deployed our algorithm into an Edge Cloud Data Center (DC) to emulate a more realistic data-gathering environment.

3 An Intelligent Method for IoSGT System

IoSGT ecosystem aspires to deliver means for directing and surveying SG service with high dexterity and efficiency by harnessing the IoT-to-Edge-to-Cloud continuum. Especially, IoSGT integrates these physical machines into a frauds domain for a more flexible, monitorable, and adaptable environment to accommodate new SG services and applications without causing significant changes to the adjacent landscape. In this section, we propose SG service to predictively detect fraud by applying ML at the Edge Cloud DCs.

3.1 IoSGT Architecture

We developed a ML framework at the network edge and the Cloud DC to process SG data. We deployed an Internet of Smart Grids for IoT (IoSGT) system divided into three main layers, namely Extreme Edge, Edge Cloud DC, and Central Cloud DC, as shown in Figure 1. The Extreme Edge layer consists of SGs to gather energy-consumption information at a given period. The SG measures voltage, current, active and reactive power, and others and fetches into an embed low capacity storage. The sets of SGs organized in groups have common similarities in terms of installed location, version, technical specifications, and consumption profile (*i.e.*, Residential, commercial, and industrial). Each SG transmits the collected data to the Edge Cloud DCs using a well-known Internet of Things (IoT) communication protocol, such as Message Queuing Telemetry Transport (MQTT). The Edge Cloud DC acts as a gateway and manager for SG groups,

inter-mediating data transmission to ML services atop the IoSGT platform. Edge Cloud DC could be deployed at IoT gateways, and network access points, routers or dedicated edge nodes. On the other hand, the extreme edge could be deployed at On the other hand, the extreme edge could be deployed at the IED

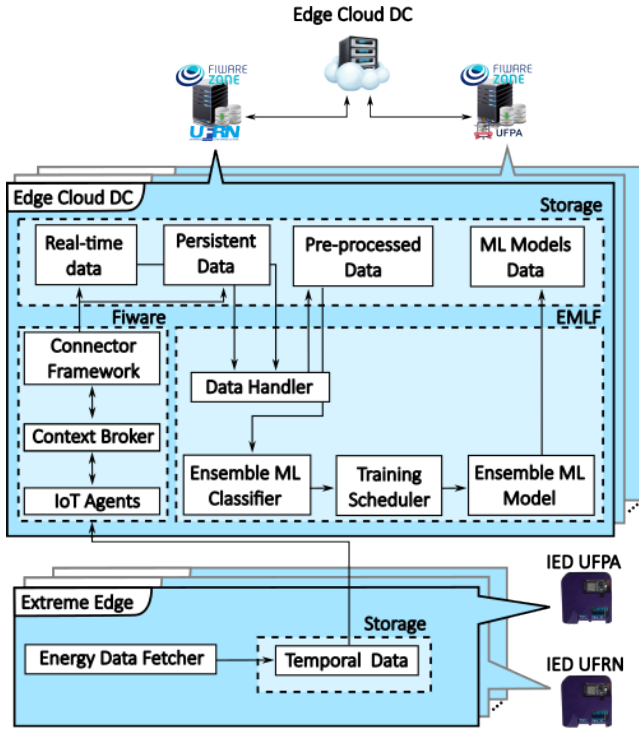


Figure 1. IoSGT Architecture

The IoSGT considers the FIWARE ecosystem using Docker containers to enable greater scalability to merge or fork Edge Cloud DCs according to chosen policies. The Extreme Edge interacts with the Edge Cloud DC through the IoT Agents suitable for computing constrained IoT devices. IoT Agents group devices to allow them to send their data to and be managed using their native and lightweight communication protocols for IoT. These protocols enable authentication and authorization of the channel between devices and the Context Broker. The SGs interact using lightweight based on Ultralight 2.0 protocol. The Context Broker acts as an interface of the Ultralight protocol, enabling context information querying. For instance, we use the Orion Context Broker as a Publish/Subscribe interface between Ultralight and the Next Generation Service Interfaces (NGSI).

In effect, this brings a standard interface to all IoT interactions at the context information management level. Each group of IoT devices are able to use their own proprietary protocols and disparate transport mechanisms under the hood whilst the associated IoT Agent offers a facade pattern to handle this complexity.

The Connector Framework creates the history of contextual data into persistent databases. Specifically, we use Cygnus to automate and manage the flowing data toward Storage in third-party databases. For instance, we store real-time data in a MongoDB database to keep the last readings. Moreover, we persist long-term data into a relational database using MySQL to organize groups of SGs. We use

the stored data to extract useful information attaching our proposed service Edge Machine Learning Framework (EMLF).

In this context, we consider a scenario with periodic energy consumption readings produced by SGs, stored at the persistent data module of IoSGT architecture. In this sense, we define EC as a vector with real daily consumption readings over a 24-h period that comprises n samples, such that $EC = ec_1, \dots, Ec_n$. In this sense, we create a dataset where each row represents a daily reading, and each column represents the value of one reading over a time interval for a given user.

The EMLF service consists of ML classifier. The persistent data collected from SGs needs formatting to feed the service. Therefore, we developed a Data Handler to perform pre-processing, such as cleaning errors, and null values, detecting outliers, transforming data types, and re-sample data into different granularity. We store pre-processed into our Storage and feed the Ensemble Classifier. Periodically, the classifier updates its models using newly collected and pre-processed data with the Training Scheduler. The training generates enhanced Ensemble ML Models for each group of SGs and saves them into a persistent database for further usage to detect, for example, frauds on energy consumption.

Finally, within the IoT-to-Edge-to-Cloud continuum, the Central Cloud DC layer complements the IoSGT with high processing and storing capabilities, enabling a global view of the SG system. For instance, the Central Cloud DC collaborates with the Edge Cloud DCs services with powerful analytics, ML models merging for more accurate models and decision-making schemes, personalized SG domain predictions from the global model, and others. Our prototyping employs an edge-based approach; therefore, the Central Cloud DC instance is not in the scope of our current work.

3.2 Predictive Fraud Detection as a service at EMLF

Figure 2 presents the HybridForest overview for predictive fraud detection as a service for the EMLF module at the IoSGT architecture. Since users may have different energy consumption patterns, we introduce a fraud detection method for users individually. For example, a commercial user will most likely consume a lot more energy when compared to a residential user. In this case, it does not make sense to work with frauds that try to detect the user's variation from his consumption pattern. The methodology adopts the following steps: i) Data acquisition, ii) Data treatment, and iii) model evaluation. In this sense, we consider the energy consumption value stored in the persistent data module of IoSGT architecture. Afterward, it is required to perform pre-processing and ensemble prediction to detect different types of fraud. Besides, we have a testbed with a set of SGs, cloud, and fog instances, as introduced by Santos *et al.* [2022]; it might not be enough to validate a fraud detection method with a large dataset.

To cope with this issue, we consider the Irish smart energy dataset Archive [2012], a widely used dataset for SG scenarios. The Irish dataset contains the consumption data of 4710 domestic and commercial clients and 535 days of readings collected in Ireland between 2009 and 2010, enabling

validation of the fraud detection method in a large-scale scenario. This dataset considers IEDs registering consumption readings every half hour, where the first reading corresponds to the interval between 0h0min0s and 0h 29min 59s, and the second reading corresponds to the interval between 0h 30min 0s and 0h 59min 59s and so forth. Thus, every day is composed of 48 sequential readings, *i.e.*, sample n of vector EC equals 48.

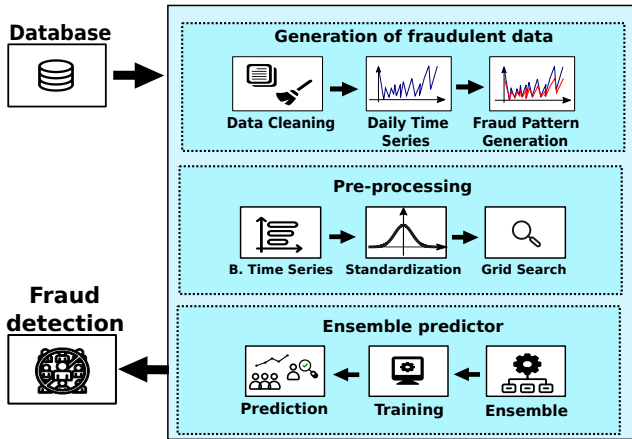


Figure 2. Adopted methodology for Fraud Detection as a service for EMLF

3.2.1 Generation of Fraud Data Phase

The *generation of fraud data phase* starts with the load of the vector with real daily consumption readings. Afterward, we need to clean the data to find errors or null values in the dataset, detect outliers, and transform them into acceptable types using scale-changer methods Gunturi and Sarkar [2021]. For instance, the number of samples must be the same for each daily vector EC since some classifiers can not handle different length time series. Hence, we drop daily vectors without the expected samples, as those could represent failures in the SG, meaning non-reliable information. This step represents a reduction of only 1% in the original number of daily vectors and does not represent a significant loss of information.

The Irish dataset organization consists of four columns: the id column uniquely identifies each user in the dataset; the day column indicates the day each measurement was made; the measurement column identifies each of the 48 samples for a certain day; the consumption column contains the measured energy consumption for that day. In this sense, we need to rearrange the readings into daily vectors, where each row represents a daily reading, and each column represents the value of one reading over a time interval. The reorganization into daily vectors is crucial, given that fraud cases are based on daily readings and need this organization to generate fraud data. Specifically, each row contains a time series with samples ranging from 0 to 47, the class label (*i.e.*, either zero for regular consumption or a number between 1 and 12 for fraudulent patterns), the day this data, and the corresponding user.

It is reasonable to assume that this dataset contains only honest consumption data since it was generated in uninterrupted monitoring of knowing customers who previously ac-

cepted the terms of the agreement and who had to answer a questionnaire before and after the measuring period. Similar to other work using the Irish dataset, we also add different types of fraud to generate a synthetic dataset with honest and fraudulent customers due to the lack of real attack samples in the dataset Chuwa and Wang [2021]. Specifically, we add twelve fraud types already defined in the literature for the whole time series, from the first to the last reading, as other works Punmiya and Choe [2019]; de Souza *et al.* [2020]; Gunturi and Sarkar [2021] have also done. In general, frauds report less energy than the actual energy consumed or redistribute the energy consumption at different times to take advantage of the varying billing system Gunturi and Sarkar [2021]. It is important to mention that the amount of samples for each fraud in the dataset is smaller because those classes would be less frequent than the normal class on a real problem. For instance, the abnormal classes have 25% fewer samples than the regular class.

Attack 1: Introduced by Jokar *et al.* [2016], this attack consists of multiplying all the readings ec_t with the same real pseudo-random number α in a predetermined interval between min and max values as Eq. (1) shown.

$$ec_t = ec_t * \alpha, \quad (\text{where } \alpha = \text{random}(min, max)) \quad (1)$$

It is considered the most commonly observed attack on the SG scenario Chuwa and Wang [2021]. Specifically, α values closer to min mean higher attack severity. It is a fraud attack in which the user artificially reduces daily consumption continuously (*e.g.*, α value of 0.3 means that SG reports only 30% of energy consumption) by the same proportion between daily measurements, as shown in Figure 3. This fraud is a complex attack to detect, especially when it started before the observation window Chuwa and Wang [2021].

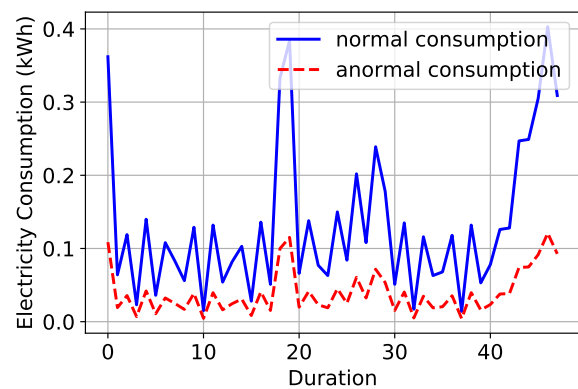


Figure 3. Energy consumption considering attack 1

Attack 2: Introduced by Jokar *et al.* [2016], this attack, also known as an on-off attack, multiplies each meter reading ec_t with a different integer random number $\gamma_t \in [0, 1]$, as Eq. (2) shown.

$$f_2(ec_t) = ec_t * \gamma_t, \quad (\text{where } \gamma_t = \text{randint}(0, 1)) \quad (2)$$

Besides being indicative of fraudulent behavior, this pattern can also indicate the malfunction of a grid infrastructure

or SG. For this pattern, the readings are either interrupted or canceled for periods of the day, *i.e.*, it replaces the consumption samples for zero each day in a random duration, as shown in Figure 4. This attack is easily detected, especially after a long period of zero reporting Chuwa and Wang [2021].

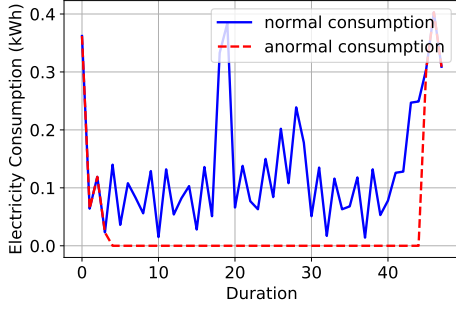


Figure 4. Energy consumption considering attack 2

Attack 3: Defined by Jokar *et al.* [2016], we generate the abnormal pattern by multiplying each meter reading ec_t by a different real pseudo-random number β_t between min and max values in a predetermined interval, as equation 3 shown.

$$ec_t = ec_t * \beta_t, \text{ (where } \beta_t = \text{random}(min, max)) \quad (3)$$

Similarly to attack 1, the fraud level increases as β_t value decrease. Although this attack has a similar equation compared to fraud 1, the β_t value is different for each sample reported, as can be seen in Figure 5. This difference means the pattern shows different attack intensities for each sample.

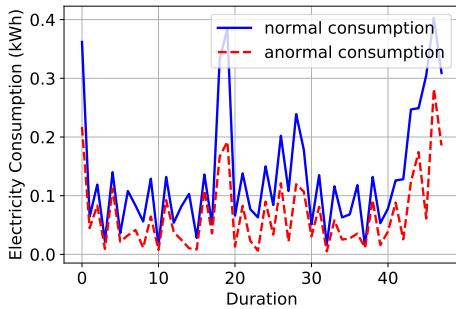


Figure 5. Energy consumption considering attack 3

Attack 4: Passos Júnior *et al.* [2016]; Yip *et al.* [2017] introduced this attack, which simulates an energy theft that takes place over a certain period, as shown in Eq. (4).

$$ec_t = \gamma_t * ec_t, \gamma_t = \begin{cases} \alpha, & t_s < t < t_x \\ 1, & \text{else} \end{cases} \quad (4)$$

This attack is similar to attack 1, but attack 4 occurs in a specific time interval instead of occurring throughout the entire time series. The duration, intensity, and beginning of this attack are selected randomly and according to the attackers' needs. However, the attackers would most likely reduce energy consumption during peak hours, days, and seasons, since the energy fee is usually higher, as shown in Figure 6.

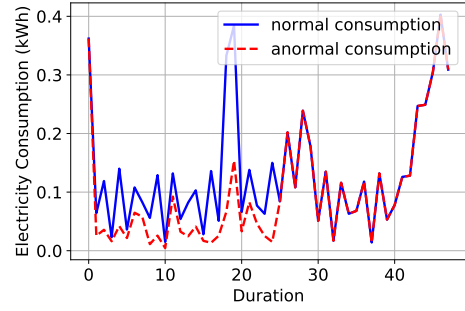


Figure 6. Energy consumption considering attack 4

Attack 5: Introduced by Jokar *et al.* [2016], we generate this attack using the mean values of the readings multiplied by a real pseudo-random number θ_t in a predetermined interval between min and max values, as shown Eq. (5).

$$ec_t = \text{mean}(ec) \times \theta_t, \text{ (where } \theta_t = \text{random}(min, max)) \quad (5)$$

In this attack, the user produces a new pattern utterly different from the regular energy consumption, as shown in Figure 7. It represents the average of the readings made over the day with a continuous reporting of "fraudulent" values.

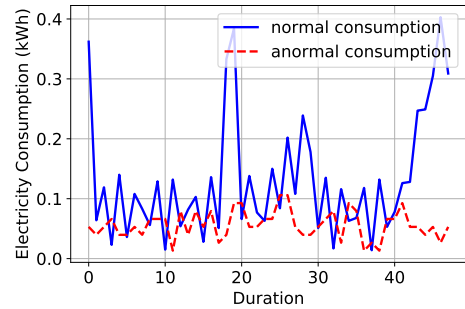


Figure 7. Energy consumption considering attack 5

Attack 6: Introduced by Zanetti *et al.* [2019], this attack pattern consists of selecting a random cut-off point and replacing all readings above that point with the cut-off value, as shown in Eq. (6).

$$ec_t = \begin{cases} ec_t, & ec_t \leq \alpha \\ \alpha, & ec_t > \alpha \end{cases} \quad (6)$$

The attacker does not exceed the maximum pre-defined limit, as shown in Figure 8. In this sense, the attacker must carefully choose the cut-off point for this type of attack since low values result in constant measurements that can be easily detected, while high cut-off points might not benefit the attacker.

Attack 7: Also introduced by Zanetti *et al.* [2019], this attack picks a random cut-off point and subtracts from the actual samples, returning zero if the result is less than zero and the subtracted value otherwise, as shown in Eq. (7).

$$ec_t = \max(ec_t - \alpha, 0) \quad (7)$$

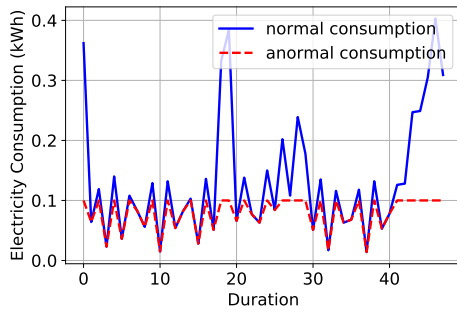


Figure 8. Energy consumption considering attack 6

If the result obtained is less than zero, zero is reported. This attack may be due to injecting false data or bypassing the meter by connecting the load directly to the distribution transformer. Thus, the amount consumed by the load is not recorded by the meter, as shown in Figure 9.

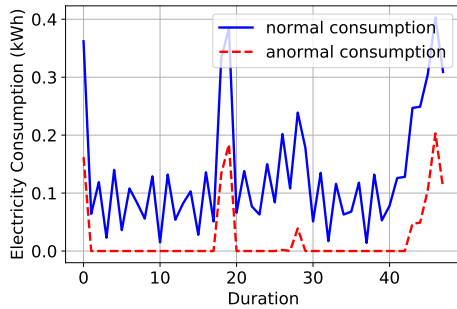


Figure 9. Energy consumption considering attack 7

Attack 8: This attack is also known as an intelligent attack proposed by Messinis *et al.* [2019], where energy gradually decreases until it reaches maximum intensity, staying at that point for the rest of the attack. The gradual decrease depends on the change in the intensity of the attack. Equation 8 describes the mathematical formulation of this attack, where the attack intensity ($0 < i_t < 1$), s indicates the rate of change in attack intensity, and t_{max} is the time with maximum intensity.

$$ec_t = (1 - i_t) ec_t, i_t = \begin{cases} i_{max}, t \geq t_{max} \\ s(t - t_s), t_s < t < t_{max} \\ 0, t < t_s \end{cases} \quad (8)$$

Figure 10 illustrates this attack. s indicates the rate of change in attack intensity, while t_{max} is when the maximum intensity takes place.

Attack 9: Defined by Jokar *et al.* [2016], this attack pattern replaces each measurement in a day with their mean value, as shown in Eq. (9).

$$ec_t = \text{mean}(ec) \quad (9)$$

This fraud represents the exact average of readings over the day, as shown in Figure 11. This attack can be easily detected since the pattern does not reflect a regular consumption that changes randomly Chuwa and Wang [2021].

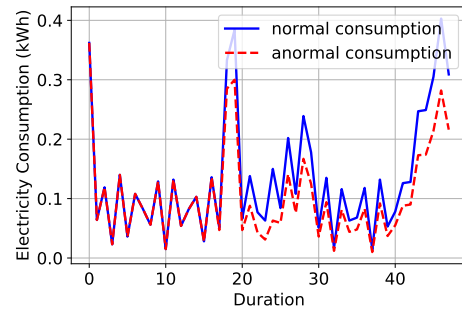


Figure 10. Energy consumption considering attack 8

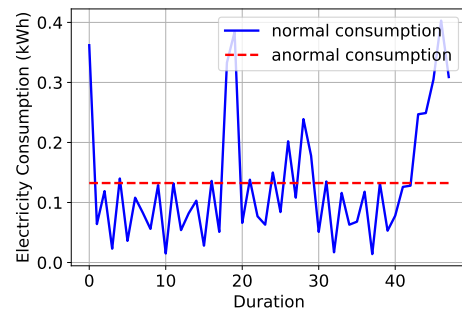


Figure 11. Energy consumption considering attack 9

Attack 10: Introduced by Jokar *et al.* [2016], this attack pattern consists of changing the chronological order of the readings, as defined in Eq. (10).

$$ec_t = ec_{T-t} \quad (\text{where } T \text{ is the sample size per day}) \quad (10)$$

Although the method does not steal energy, the attacker might reduce the overall bill by shifting high consumption periods to peak from off-peak, as shown in Figure 12. This attack patterns of target systems that bill clients differently according to the time of the day.

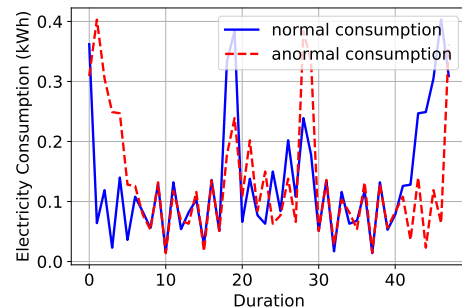


Figure 12. Energy consumption considering attack 10

Attack 11: As described by Yip *et al.* [2018], this attack only reduces energy consumption by a specified period, as defined by Eq. (11). Where t_s is the starting time of the highest consumption n time period, N is the total number of samples, $t_x = t_s + n$, and $\epsilon = \sum_{j=1}^n ec_{t+j-1}$. In this sense, the reduced amount of energy is distributed for the remaining time of that day.

$$ec_t = \begin{cases} e_t - \lambda ec_t, & t_s < t < t_x \\ ec_t + \epsilon/N - n, & \text{else} \end{cases} \quad (11)$$

The overall consumption for this attack stays the same. However, the billing could be lower if the company uses time-varying pricing systems, as shown in Figure 13. In this sense, this type of attack is only valid if there is no fixed pricing system.

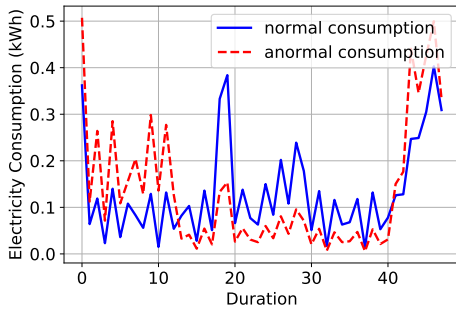


Figure 13. Energy consumption considering attack 11

Attack 12: Also described by Yip *et al.* [2018], this attack happens when an attacker switches their consumption pattern with another user z , as described in Eq. (12).

$$ec_t = ec_{tz} \quad (12)$$

In this case, the legitimate user will end up paying for the attacker's electricity. Additionally, this attack may happen when users under-report their consumption and over-report the same proportion to their neighbors Krishna *et al.* [2016]. This type of attack can not be easily detected. Figure 14 shows the attack.

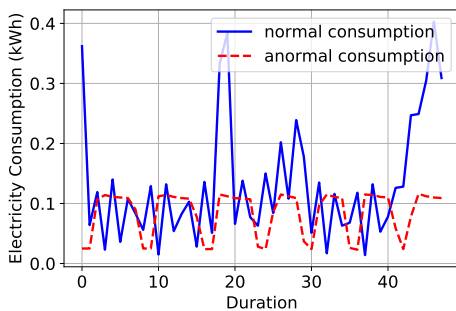


Figure 14. Energy consumption considering attack 12

Most attacks caused by injecting false data need attackers to have full or partial network information to modify the meter readings. Besides, attackers need to know the trend of real consumption before generating attack vectors so that attacks can occur successfully with little chance of being detected by the electricity provider. In some attacks, attackers should be familiar with the pricing system used to maximize their benefit. Attackers often need help to gain full access to the smart grid network. Although there is a low probability of such attacks, they can not be ignored.

3.3 Pre-processing

Energy consumption data is conventionally sampled in a unidimensional time series whose readings follow a chronological order. One of the essential steps of the pre-processing phase consists of adjusting this data. The Algorithm employs a time series cross-validation technique to generate comparatively more reliable and solid results. First, the data is split into two subsets: the training and test sets. Blocking time series divides the data into n Folds, each with specific training and test sets. Each fold contains the same number of samples, and each sample is sequentially divided. The classifiers are then trained with each training subset, and the parameters shown to reduce classification error for each validation set are selected. Lastly, the predictor is configured with the best-performing parameters and trained with the complete training set. The final performance is the mean value of the individual performances for every fold.

The Blocking time series split adds margins in two directions. One margin prevents the classifiers from memorizing future trends, and the other contains the classifiers from remembering patterns in-between interactions. Despite the increased complexity of cross-validation, it is indispensable because it makes the predictors more error-resistant. This application needs to consider time series attributes when splitting data, guarantee that the predictors learn from past data (training stage), and make predictions about what future data should look like (test stage), making it better at working with real-world scenarios.

After being split, the training sets are normalized in the standardization step. Normalization brings all features to the same scale, between 0 and 1. Normalized entries prevent the classifiers from leaning towards features with more outstanding orders of magnitude, which will improve predictor generalization.

Afterward, we apply a grid search method to find the best hyperparameters using the validation set. As the performance of a classifier is highly dependent on the tuned hyperparameters, the grid search method will test different combinations of the hyperparameters we specified and then find the best combination among them. In our case, this is done exhaustively, and all the possible combinations are tested in the method. In that way, we can better estimate the real performance of the evaluated classifiers without making assumptions based on a particular result.

The grid search method relies on a cross-validation (CV) strategy to provide good results. The idea behind CV consists of splitting a dataset into two parts, one for training and another for testing, repeating the process is repeated for different iterations. However, as we are working with time series, data partitioning must be made with care since we could erroneously train the models using data from the future and making predictions about the past, which would provide unrealistic results. Therefore, we use a CV strategy to handle time series known as blocking time series split.

3.4 Ensemble Predictor

In HybridForest method, multiple TSC algorithms (*i.e.*, signature, Catch22, Weasel, TSF, K-NN for time series, and Ar-

senal) are trained to detect frauds in a given dataset. Then ensemble learning builds a more robust predictor by combining multiple TSC algorithms to provide better results than single classifier predictors Géron [2019]. In ensemble learning theory, we can obtain more accurate and/or robust models as soon as we correctly combine weak models. Specifically, We call weak learners (or base models) models that can be used for designing more complex models by combining several of them. These basic models often perform poorly by themselves because they have a high bias, such as low degree of freedom models, or because they have too much variance (e.g., high degree of freedom models). Hence, an ensemble predictor reduces the bias and/or variance of basic classifiers by combining several to create an aggregated learner (or ensemble predictor) that achieves better performances Araujo *et al.* [2020].

We consider a voting class classifier for the ensemble predictor, combining different ML classifiers conceptually. For a classification problem, if the base models return the class labels, then one way of combining is by considering the class returned by each weak classifier (base model) as a vote, and the class with the highest number of votes is returned by the ensemble model (hard voting). Afterward, the ensemble predictor is created and trained with the training set; the predictor must be evaluated. We rely on a CV strategy to analyze the performance of HybridForest method to provide good results. In this sense, we split the data into two sets for each CV iteration: testing and training. Specifically, classifiers are always trained with data from the past and evaluated on new occurrences. In terms of Big-0 analysis, the complexity of the proposed HybridForest ensemble is: $O(C)$ for training and inference, where C is the number of the ML classifiers used. The ensemble's complexity for searching, however, is exponential $O(2^C)$ since the voting strategy belongs to the NP-hard complexity class, as seen in Fersini *et al.* [2014]. In the following, we describe the considered TSC algorithms to combine in the proposed ensemble.

3.4.1 Signature

The signature transform is a mathematical technique used in TSC as a feature extraction tool, where the discrete time series is transformed in continuous paths through interpolation techniques, and an infinite set of features, known as signatures, can be computed from the new sequence Chevyrev and Kormilitzin [2016]. These signatures are combined with a traditional ML classifier to produce an output. This method employs augmentations, windows, transformation, and rescaling, all grouped in a single mathematical framework.

3.4.2 Catch22

22 Canonical TS Characteristics (catch22) is a dynamic and commonly used technique for time series data. Catch22 captures time series' diverse and interpretable characteristics according to their properties, including linear and nonlinear autocorrelation, continuous differences, value distributions, outliers, and volatility scaling properties Lubba *et al.* [2019]. This technique utilizes a reduction in dimensionality from

4791 to 22, correlates with a roughly 1000-fold decrease in computation time, and scales almost linearly with time series length, despite an average 7% reduction in classification accuracy.

3.4.3 Weasel

Word extraction for time-series classification (WEASEL) is a TSC based on the bag-of-patterns, which is scalable and accurate. Weasel has considered the differences between classes in the feature discretization process rather than relying on fixed, data-independent intervals; this results in a highly discriminatory feature set. Weasel does not treat each fixed-length window as an independent feature but uses windows of different lengths and considers the windows' order. Weasel applies aggressive statistical feature selection instead of simply using all features for classification; this results in smaller function space and dramatically reduces runtime without sacrificing accuracy Schäfer and Leser [2017].

3.4.4 Time series forest (TSF)

Tree-ensemble classifier: time series forest (TSF) employs a new measure called the Entrance (entropy and distance) gain to identify high-quality splits, such as using entropy gain and two One-nearest-neighbor with dynamic time warping algorithms Xi *et al.* [2006]. Using a random feature sampling strategy, TSF has computational complexity linear in the time series length. Firstly, TSF uses a new splitting criterion named Entrance gain that combines the entropy gain and a distance measure to identify high-quality splits. Experimental studies on 45 benchmark data sets show that the Entrance gain improves the accuracy of TSF. Secondly, TSF randomly samples features and thus makes the computational complexity linear in the time series length. In addition, each tree in TSF is grown independently, and, therefore, modern parallel computing techniques can be leveraged to speed up TSF Deng *et al.* [2013].

3.4.5 k-NN for Time Series

Due to the importance of interpretation and insight, k-Nearest Neighbour (k-NN) methods have a prominent position in data analysis. Because k-NN methods are translucent, they deliver interpretable models. It also applies to time series analysis, where a side-by-side comparison of time series can demonstrate similarities and differences between methodologies. Nevertheless, when using k-NN methods for time series analysis, there are extra challenges in developing metrics that can truly capture the similarity between time series. Two-time series can still be similar if one is stretched or shifted relative to the other. It could also be that the similarity depends on short or even tiny signatures in the time series Mahato *et al.* [2018].

3.4.6 Arsenal

It is an ensemble of Rocket transformers using a Ridge classifier with built-in cross-validation. Rocket (Random Convolutional Kernel Transform) transform time series using many random convolution kernels, *i.e.*, H. Kernels with

random lengths, weights, warping, dilation, and padding. The transformed features are used to train a linear classifier. The Rocket and logistic regression combination form a single-layer convolutional neural network with random kernel weights, and the transformed features form the input to the trained softmax layer. However, it uses a ridge regression classifier on all but the most extensive datasets. It has the advantage of fast cross-validation on the regularization hyperparameters (and no other hyperparameters). Nonetheless, logistic regression trained with stochastic gradient descent is more scalable. Extensive datasets use logistic regression when the number of training samples is significantly larger than the number of features Dempster *et al.* [2020].

4 Evaluation

This section describes the methodology and performance metrics used to evaluate the predictors for NTL detection. We compared the performance results obtained with the algorithm with other TSC and non-TSC ML algorithms. We used the following performance metrics: Precision, F1-Score, Accuracy, FPR, Recall, and Area Under the Curve (AUC) to evaluate the effectiveness of the Ensemble algorithm. Moreover, this is an open-source project available on GitHub¹.

4.1 Evaluation Methodology and Metrics

We evaluate the algorithms proposed in the python language. We used the sktime Löning *et al.* [2019], a python package popular for tasks involving time series such as forecasting, classification, regression, clustering, outlier detection, anomaly detection, and segmentation. The package reunites many state-of-art algorithms that are recurrent in the time series literature. We also consider other package modules for data analysis, such as pandas and NumPy for data manipulation, matplotlib for plotting, and sci-kit-learn for evaluation.

We followed the methodology for fraud detection as a service for EMLF introduced in Section 3.2 and depicted in Figure 2. In this sense, we selected the parameters of the ensemble learning grid search based on tools for hyperparameter tuning. As mentioned earlier, ML boils down to comparing different models and trying to find the best one that works. Furthermore, they are chosen individually for each algorithm used in the ensemble. First, we try to maximize the efficiency of these algorithms individually. We then compare the impact of each algorithm on the ensemble by choosing the best set of weights for the algorithm. Table 2 shows the best set of parameters tested for each ML technique and also for the ensemble, where the values from left to right describe weights for Catch22, Weasel, TS forest (TSF), k-NN for TS, and Arsenal, respectively.

The models were assessed using the blocking time series split cross-validation (CV) strategy with five folds for each user, for every user in the dataset. Cross-validation is important because the performance variance is reduced as the model is evaluated several times per user, providing more reliable results when compared to a single ensemble run Maleki *et al.* [2020]. For each metric, we extract variation of the CV

Table 2. Table of Parameters

Classifier	Predictor
SVM	C = 100 kernel = rbf gamma = 0.1
Signature	estimators = 100 estimator = Random Forest window depth = 3 depth = 4
CatBoost	estimators = 150
LightGBM	estimators = 150
XGBoost	estimators = 150
Catch22	estimators = 200 estimator = Random Forest kernels = 1000
Arsenal	transform = rocket estimators = 25 estimator = Ridge Classifier
k-NN	neighbors = 1 distance = DTW
Weasel	binning strategy = information gain window increment = 2 bigrams = true

strategy's iterations that represent the standard deviation of each one. The standard deviation is a measure of how dispersed the data is in relation to the mean. Low standard deviation means data are clustered around the mean, and high standard deviation indicates data are more spread out. Hence, the statistics obtained provide a better insight into the actual performance than a single run of the ensemble.

We considered the following metrics to evaluate the performance of the analyzed fraud detection method: Precision, F1-Score, Accuracy, False Positive Rate (FPR), Recall, and AUC. For that, we used True Negative (TN) and False Negative (FN) too. Precision tries to answer the question: what proportion of attributes are correctly identified? Another way to express accuracy is the overall ratio of True Positives (TP) to predicted positives. Precision considers the number of features correctly assigned to a given class versus the number of correct and incorrect assignments False Positive (FP). Precision measures the correctness of the classifier and the correlation of the positive classification, which is computed based on Eq. (13). Higher Precision means more true positives and fewer false positives.

$$Precision = \frac{TP}{TP + FP} \quad (13)$$

On the other hand, the F1-Score is used to assess the data's positive predictive value and sensitivity to find some balance when using the harmonic mean. The F1 score is the most appropriate metric for imbalanced datasets representing different class distributions among the five performance metrics. The F1-Score is the weighted average of the Precision, where the first value is the ratio of the number of correctly predicted positive observations to the total number of positive observations, as shown in Eq. (14).

$$F1 - Score = 2 * \frac{Recall * Precision}{Recall + Precision} \quad (14)$$

¹<https://github.com/Euronym/predictive-fraud-detection-smart-grids>

Accuracy is a metric used to evaluate classification models. The accuracy increases with values of the parameter and stabilizes when reaching the optimal point. The best parameter values obtained in this step are used in the remaining of the experiments Belhadi *et al.* [2021a]. Informally, accuracy is the score our model predicted correctly. Accuracy has the following definition: accuracy means correct predictions, total predictions. The false-positive rate is a metric that can assess the accuracy of ML. A model must know "basic reality" or the real state of affairs to measure its true accuracy. The model's accuracy can then be directly assessed by comparing the model's output with the ground truth.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (15)$$

FPR evaluates false positives, *i.e.*, misclassified samples as fraudulent. FPR is proportional to the additional cost incurred by the utility because a team was mistakenly dispatched to a frequent visitor inspection. Of course, this is also very annoying for misidentified clients.

$$FPR = \frac{FP}{FP + TN} \quad (16)$$

Recall calculates how many of the Actual Positives our model capture by labeling it as Positive (True Positive), computed according to Eq. (17). Applying the same understanding, we know that Recall shall be the model metric we use to select our best model when there is a high cost associated with a False Negative. For instance, in fraud detection, If a fraudulent transaction (Actual Positive) is predicted as non-fraudulent (Predicted Negative), the consequence can be terrible.

$$Recall = \frac{TP}{TP + FN} \quad (17)$$

Finally, the AUC spans true positive and false positive rates and varies between 0 and 1, as in Eq. (18). AUC is an excellent metric for evaluating imbalanced databases. The higher the AUC, the more correctly the predictor can predict the outcome.

$$AUC = \int_{x=0}^1 Recall(FPR^{-1}(x))dx \quad (18)$$

4.2 Analysis for each classifier to detect fraud

Figure 15 shows the precision performance of all the classifiers implemented for NTL detection. By analyzing the results, we can conclude that the precision results closely follow the recall results that will show next, which corroborates with the possibility of adopting TSC classifiers for NTL detection. It is important to highlight that HybridForest also shows better Precision than the classifiers analyzed. Specifically, HybridForest provides average precision results of 80.46 %, which is in the average 8% and 5% higher compared to CatBoost (*i.e.*, the best-performing conventional classifier) and Signature (*i.e.*, the best-performing TSC classifier), respectively. On the other hand, in the best results of standard deviation, HybridForest provides precision is 4%

and 3% higher compared to CatBoost and Signature, respectively. Finally, in the worst results, we are higher 2% with the average standard deviation compared to Signature and 4% to CatBoost. Based on the analysis of precision results, it is possible to conclude that HybridForest achieve a higher probability of randomly selecting a relevant sample, *i.e.*, the number of hits returned that was TP or TN.

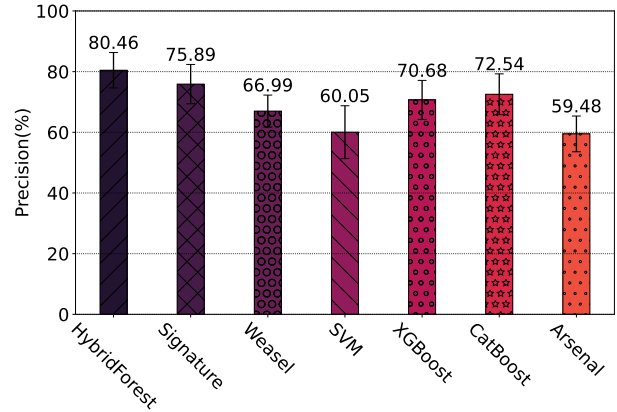


Figure 15. Precision for different TSC algorithms for fraud detection

Figure 16 presents the F1-Score for the analyzed NTL detection predictor. Accordingly, HybridForest provides a high ratio of correctly predicted positives to the total number of real positive samples, suggesting a high recall and precision. The F1-Score performance confirms the benefits of HybridForest predictor compared to the predictors analyzed for NTL detection. Hence, the F1-Score results mean that HybridForest is efficient for both detecting frauds and for correctly identifying honest data samples because HybridForest considers the temporal nature of energy consumption data in the pre-processing, training, testing, and validation steps, and also employs different TSC classifiers to create an ensemble predictor. The F1-Score average results of HybridForest is 7% and 4% higher compared to CatBoost (*i.e.*, the best-performing conventional classifier) and Signature (*i.e.*, the best-performing TSC classifier), respectively. On the other hand, in the best results of standard deviation, HybridForest provides F1-Score 4% and 3% higher compared to CatBoost and Signature, respectively. Finally, in the worst results, HybridForest is 1.5% higher with the average standard deviation compared to Signature and 3% to CatBoost. As the results go by, we realize that algorithms with a low number of features extracted from the data tend to obtain lower values for attacks with more sensitive data to be detected.

Figure 17 shows accuracy for the analyzed NTL detection predictor. Based on the results, we can conclude that the HybridForest with your better accuracy than the classifiers analyzed. Specifically, HybridForest provides an average accuracy results of 77.33%, which is around 8% and 5% higher compared to CatBoost (*i.e.*, the best-performing conventional classifier) and Signature (*i.e.*, the best-performing TSC classifier), respectively. On the other hand, in the best results of standard deviation, HybridForest provides accuracy 4% and 3% higher compared to CatBoost and Signature,

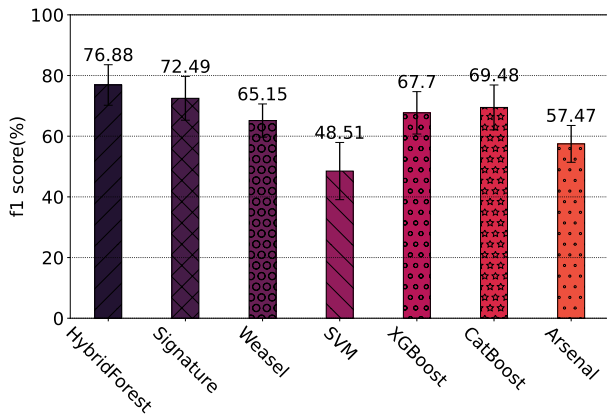


Figure 16. F1-score for different TSC algorithm for fraud detection

respectively. Finally, in the worst results, we are higher 2% with the average standard deviation compared to Signature and 4% to CatBoost.

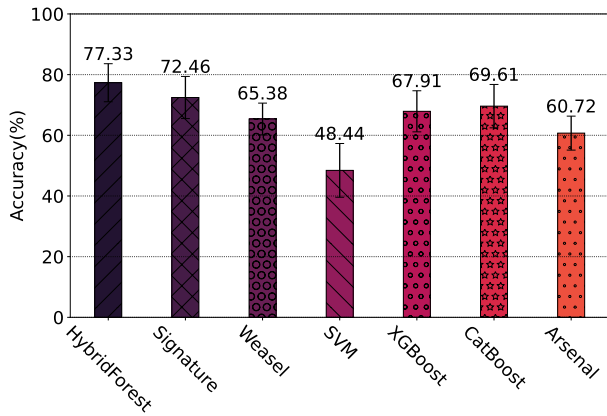


Figure 17. Accuracy for different TSC algorithm for fraud detection

Figure 18 illustrates the FPR results, which directly correlate to the inspection costs incurred by utility companies. A small FPR value represents a small number of false detection. The FPR results also show the benefits of HybridForest for fraud detection compared to the predictors analyzed because HybridForest consider the temporal nature of energy consumption data and uses different TSC classifiers to create an ensemble predictor. For instance, HybridForest provides an average FPR results is 33 % and 128 % lower compared to CatBoost (*i.e.*, the best-performing conventional classifier) and Weasel , respectively. HybridForest achieved a significantly lower FPR, in the best results of the standard deviation of 0.001 compared to 0.004 of Signature, four times lower. Those results corroborate our method’s validity because it achieved very low FPR values compared to other NTL detectors, regardless of the classifier.

Figure 19 shows the recall performance of all the classifiers implemented for NTL detection. The recall is the most used performance metric for NTL detectors, and high recall values indicate that the predictor is efficient for fraud detection. By analyzing the results, we observed that all individual TSC-based predictors (*i.e.*, Catch, Weasel, KNN, SVM, and Arsenal) performed worst when compared to conventional

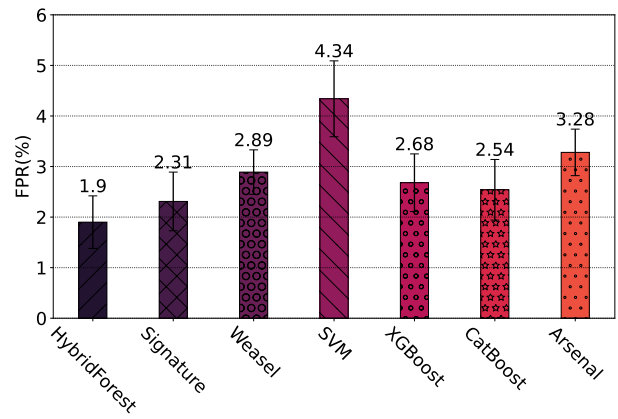


Figure 18. FPR for different TSC algorithm for fraud detection

classifiers (*i.e.*, XGBoost, CatBoost, and LightBoost), and HybridForest yields the highest recall performance. HybridForest has an average recall results of around 8 % and 5 % higher compared to CatBoost (*i.e.*, the best-performing conventional classifier) and Signature , respectively. HybridForest provides Recall in the best results of the standard deviation of 3% higher compared to Signature and 4% to CatBoost. In the worst results, 1.5% to Signature and 4% to CatBoost. Based the evaluation results, we can conclude an ensemble predictor (combining different TSC classifiers) produces a more robust and accurate predictor. Consequently, we can classify user samples with ease based on the recall.

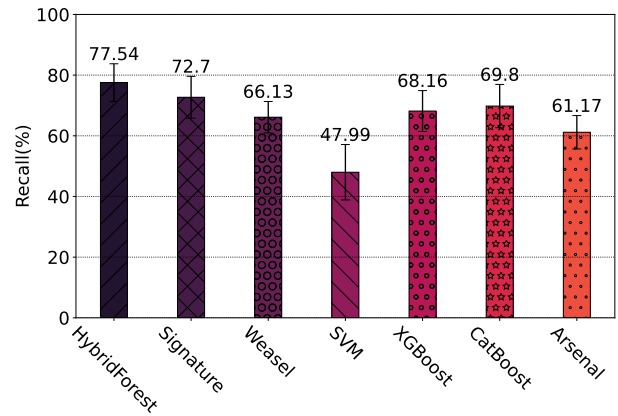


Figure 19. Recall for different TSC algorithm for fraud detection

4.3 Analysis for each classifier to detect each kind of fraud

We created two categories to analyze the ROC curve, *i.e.*, easy attacks to be classified and difficult ones. We divide it into two categories to enable better visualization of ROC results and to show the impact of HybridForest to classify each kind of fraud regardless of their characteristics. It is because there are frauds that are very different from regular consumption, thus facilitating their detection, and others are called smart frauds that try to imitate the characteristics of regular consumption. Figures 20 and 21 show the ROC curves for each class predicted for HybridForest, TSC-based predictors

with the best performance (*i.e.*, Signature), and the conventional classifiers (*i.e.*, XGBoost). The ROC curve illustrates the relationship between recall and FPR, where an optimal result should achieve the highest possible recall value while maintaining the lowest possible FPR. We can observe that the performance varies among the different predictors and also among the different classes (*i.e.*, honest data and twelve different types of Attacks described by Eqs. 1-12). It is due to each class having different patterns and characteristics. Honest and fraudulent fl data follow the same consumption pattern but with different amplitudes. It makes it difficult for predictors to distinguish one class from the other, especially when it started before the observation window Chuwa and Wang [2021]. Attack 4 has the worst performance regardless of the predictor. It is important to highlight that some predictors yield different performance results for attacks, *e.g.*, HybridForest results in an AUC of over 80% to identify Attack 7, while XGBoost and Signature yield an AUC value of under 70% for the same class.

In Attack 9, the actual samples are replaced by their mean value Jokar *et al.* [2016], which facilitates prediction. In addition, Attack 12 can be easily detected because it occurs when an attacker switches their consumption pattern to a user with a low consumption pattern Yip *et al.* [2018]. In this way, for both Attacks 12 and 9, the Attack patterns yield the best results, *i.e.*, high recall and low FPR (close to 1), regardless of the predictor. Finally, Attack 3 multiplies each meter reading x_t by a different real pseudo-random number β_t between *min* and *max* values in a predetermined interval. Specifically, the Attack level increases as β_t value decrease. It means the NTL might not occur continuously, and there may be some discontinuous reporting of “fraudulent” values. Therefore, the user produces different reduction rates across measurements in this attack pattern. It confuses conventional classifiers incapable of accounting for the time-dependent nature of the data and processing it point by point. Therefore conventional classifiers will not be able to discern between this type of Attack and honest samples readily.

We can also observe that honest data and Attack patterns 1, 4, 6, 8, and 11 have the worst results compared to other classes (*i.e.*, 2, 3, 5, 7, 9, 10, and 12), regardless of the predictors. For instance, the XGBoost predictor in Figure 20c and 21c shows the worst results for honest data and Attack patterns in the 3 for easy and 4 for hard classes compared to the remaining classes. The curves for these classes take longer to reach the maximum value on the y-axis, which means that these classes have a lower recall than the others, *i.e.*, a lower detection rate. On the other hand, detecting Attack patterns 2, 5, 7, and 9 for easy classes and Attack patterns 1, 6, and 8 for challenging classes show better performance, as their curves illustrate, in most cases, higher than the average curve of all classes. It demonstrates that it is easier to make predictions for those classes, *i.e.*, they have a higher detection rate.

By analyzing the results of each predictor, we conclude that HybridForest obtained high performances regardless of the classes, reaching the maximum y-axis value very quickly, especially for Attack patterns 2, 5, and 9 for ROC easy and 1, 6, 8 for ROC hard. It is attributed to HybridForest relying entirely on TS data processing combined with an ensemble predictor. On the other hand, the individual TSC-based

predictor (*i.e.*, Signature) also obtained reasonable results to detect each Attack pattern because they were specifically designed to process TS data. Lastly, the Conventional predictor (*i.e.*, XGBoost) showed the worst performances because they treated the data in a tabular format, which can hinder classification for this application.

We conclude from the analysis of the results obtained that HybridForest achieved a higher Recall, precision, and F1-score compared to the other predictors. In addition, HybridForest outperformed all the other detectors in terms of FPR (*i.e.*, 1.9), which directly translates into tangible benefits (such as the reduction of inspection costs) to utility companies as an expected trade-off for a more comprehensive and practical NTL predictor. Usually, binary NTL detectors group all Attack patterns in a single class, which can confuse the predictor by having different patterns being part of the same class, which increases FPR values. In this context, HybridForest took advantage of the time dependence inherent to time series in the classification process. Therefore, our methodology not only brings benefits to utility companies but also improves NTL detection. It is achieved by exploring the available data in novel ways and by employing and combining the most recent resources for classification problems more efficiently to detect specific types of NTL.

5 Conclusion

Despite many efforts to detect fraud, the challenge remains an open issue. In this article, we presented a heterogeneous ensemble predictor for NTL detection as a use case in our IoSGT system with ML framework at the network edge. HybridForest relies on the heterogeneous Ensemble to perform a multi-classification of fraudulent users (*i.e.*, classifying samples as honest or as a specific type of fraud) with high performance against other methods such as SVM, XGBoost, CatBoost, LightBoost to identify specific types of fraud. In addition, we consider that frauds in the electrical system are not only binary, and our predictor can also classify these variations among frauds and their different unique aspects. In this context, the Edge Cloud DC hosting HybridForest identifies specific types of fraud and considers the temporal nature of energy consumption data in the pre-processing, training, testing, and validation stages while considering different TSC classifiers to create an ensemble predictor.

For evaluation, we considered the Irish dataset as input into our IoSGT system, which has a considerable number and type of customers, and a long duration of measurements. This database stored in IoSGT includes only honest consumption data because it was generated in a scenario of uninterrupted monitoring of known customers who earlier accepted the terms of the compact and who had to respond to a questionnaire before and after the measuring period. The EMLF counted twelve types of fraud already described in the literature in this dataset to create a synthetic dataset with honest and fraudulent consumers. In this context, we can determine different types of fraud, whereas existing related works only make binary categories (fraudulent and non-fraudulent). We also followed the method adopted in similar works, where the fraudulent data generated were randomly selected among

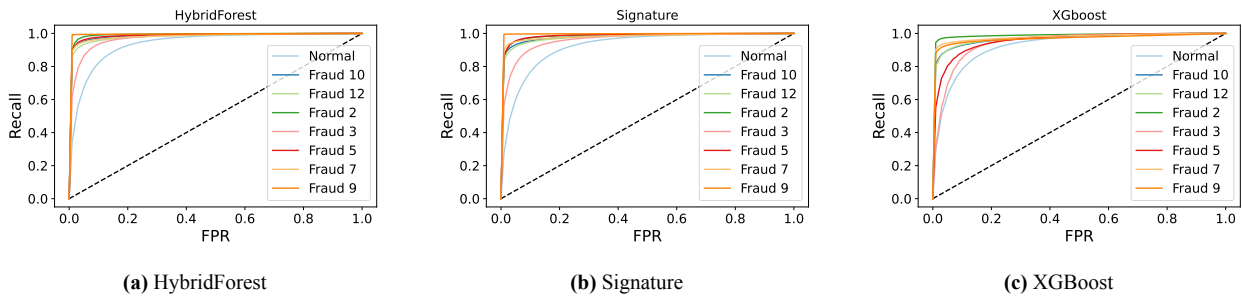


Figure 20. ROC for different TSC algorithm to detect each type of easy Attack

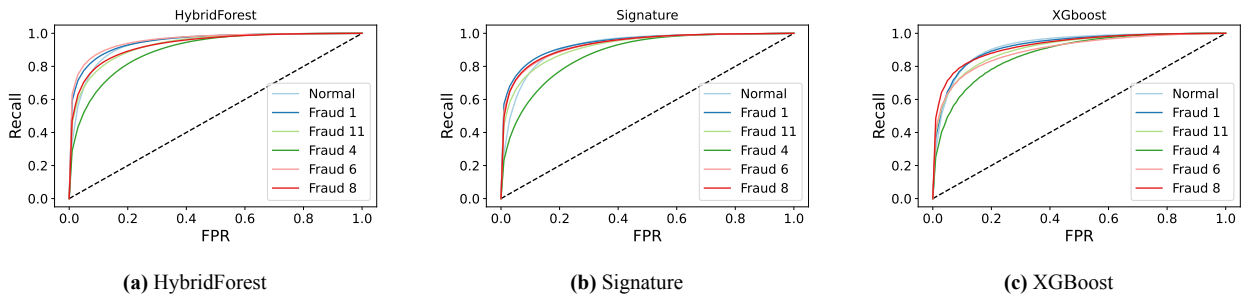


Figure 21. ROC for different TSC algorithm to detect each type of hard Attack

users. It is essential because fraudulent samples are heavily unbalanced and must be used by ML algorithms.

The algorithm employs TS of user consumption data to build a predictor capable of classifying samples as honest or a specific type of fraud. We tested and compared multiple TSC algorithms in our experiments. The TSC algorithms performed better than the conventional classifiers for all metrics, demonstrating the benefits of using this classifier to create a prediction for NTL detection. By employing TSC classifiers to build an ensemble predictor, we obtained a performance improvement with an FPR value equal to 1.9% and precision of 80.5% for heterogeneous data and kinds of frauds. The algorithm focuses on TS data, which enables the development of a method that better interprets real-world scenarios and is more error-resistant. For future work, we aim to evaluate other aspects from the network point of view, such as memory (execution and persistent data storage), CPU, and time response. In addition, we could evaluate other aspects of the cloud can directly impact the training time of the ensemble.

Acknowledgments

This study was financed by the R&D project entitled “Sistema IoT-Cloud de Medição Centralizada de Energia Voltado a Rede CEA - 001/2021”, and in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

Declarations

Authors’ Contributions

LB contributed to the conception of this study. BM, HS, IM, PE, LM, DS, EN, EC, MK and AN performed the experiments. LB is the main contributor and writer of this manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

Data can be made available upon request.

References

- Araujo, F., Araújo, F., Machado, K., Rosário, D., Cerqueira, E., and Villas, L. (2020). Ensemble mobility predictor based on random forest and markovian property using lbsn data. *Journal of Internet Services and Applications*, 11. DOI: 10.1186/s13174-020-00130-7.
- Archive, I. S. S. D. (2012). Commission for energy regulation (cer) smart metering project - electricity customer behaviour trial, 2009-2010 [dataset]. Available at: <https://www.ucd.ie/issda/data/commissionforenergyregulationcer/>.
- Barja-Martinez, S., Aragüés-Peñalba, M., Ingrid Munné-Collado, Lloret-Gallego, P., Bullich-Massagué, E., and Villafila-Robles, R. (2021). Artificial intelligence techniques for enabling big data services in distribution net-

- works: A review. *Renewable and Sustainable Energy Reviews*, 150:111459. DOI: 10.1016/j.rser.2021.111459.
- Belhadi, A., Djenouri, Y., Djenouri, D., Srivastava, G., and Lin, J. C.-W. (2022). Group intrusion detection in the internet of things using a hybrid recurrent neural network. *Cluster Computing*, pages 1–12. DOI: s10586-022-03779-w.
- Belhadi, A., Djenouri, Y., Srivastava, G., Djenouri, D., Lin, J. C.-W., and Fortino, G. (2021a). Deep learning for pedestrian collective behavior analysis in smart cities: A model of group trajectory outlier detection. *Information Fusion*, 65:13–20. DOI: 10.1016/j.inffus.2020.08.003.
- Belhadi, A., Djenouri, Y., Srivastava, G., Jolfaei, A., and Lin, J. C.-W. (2021b). Privacy reinforcement learning for faults detection in the smart grid. *Ad Hoc Networks*, 119:102541. DOI: 10.1016/j.adhoc.2021.102541.
- Chevyrev, I. and Kormilitzin, A. (2016). A primer on the signature method in machine learning. DOI: 10.48550/arXiv.1603.03788.
- Chuwa, M. G. and Wang, F. (2021). A review of non-technical loss attack models and detection methods in the smart grid. *Electric Power Systems Research*, 199:107415. DOI: 10.1016/j.epr.2021.107415.
- de Souza, M. A., Pereira, J. L., Alves, G. d. O., de Oliveira, B. C., Melo, I. D., and Garcia, P. A. (2020). Detection and identification of energy theft in advanced metering infrastructures. *Electric Power Systems Research*, 182:106258. DOI: 10.1016/j.epr.2020.106258.
- de Souza Savian, F., Siluk, J. C. M., Garlet, T. B., do Nascimento, F. M., Pinheiro, J. R., and Vale, Z. (2021). Non-technical losses: A systematic contemporary article review. *Renewable and Sustainable Energy Reviews*, 147:111205. DOI: 10.1016/j.rser.2021.111205.
- Dempster, A., Petitjean, F., and Webb, G. I. (2020). Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495. DOI: 10.1007/s10618-020-00701-z.
- Deng, D., Li, J., Jhaveri, R. H., Tiwari, P., Ijaz, M. F., Ou, J., and Fan, C. (2022). Reinforcement learning based optimization on energy efficiency in uav networks for iot. *IEEE Internet of Things Journal*. DOI: 10.1109/JIOT.2022.3214860.
- Deng, H., Runger, G., Tuv, E., and Vladimir, M. (2013). A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153. DOI: 10.1016/j.ins.2013.02.030.
- Fersini, E., Messina, E., and Pozzi, F. (2014). Sentiment analysis: Bayesian ensemble learning. *Decision Support Systems*, 68:26–38. DOI: 10.1016/j.dss.2014.10.004.
- Gunturi, S. K. and Sarkar, D. (2021). Ensemble machine learning models for the detection of energy theft. *Electric Power Systems Research*, 192:106904. DOI: 10.1016/j.epr.2020.106904.
- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O’Reilly Media, Inc. Book.
- Jokar, P., Arianpoo, N., and Leung, V. C. M. (2016). Electricity theft detection in ami using customers’ consumption patterns. *IEEE Transactions on Smart Grid*, 7(1):216–226. DOI: 10.1109/TSG.2015.2425222.
- Krishna, V. B., Lee, K., Weaver, G. A., Iyer, R. K., and Sanders, W. H. (2016). F-deta: A framework for detecting electricity theft attacks in smart grids. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 407–418. DOI: 10.1109/DSN.2016.44.
- Lobato, E., Prazeres, L., Medeiros, I., Araújo, F., Rosário, D., Cerqueira, E., Tostes, M., Bezerra, U., Fonseca, W., and Antloga, A. (2023). A monitoring system for electric vehicle charging stations: A prototype in the amazon. *Energies*, 16(1):152. DOI: 10.3390/en16010152.
- Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., and Király, F. J. (2019). sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872*. DOI: 10.48550/arXiv.1909.07872.
- Lubba, C. H., Sethi, S. S., Knaute, P., Schultz, S. R., Fulcher, B. D., and Jones, N. S. (2019). catch22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery*, 33(6):1821–1852. DOI: 10.1007/s10618-019-00647-x.
- Mahato, V., O’Reilly, M., and Cunningham, P. (2018). A comparison of k-nn methods for time series classification and regression. In *AICS*, pages 102–113. Available at: https://www.researchgate.net/profile/Vivek-Mahato-3/publication/329702259_A_Comparison_of_k-NN_Methods_for_Time_Series_Classification_and_Regression/links/6250341e4f88c3119cea19a6/A-Comparison-of-k-NN-Methods-for-Time-Series-Classification-and-Regression.pdf.
- Maleki, F., Muthukrishnan, N., Ovens, K., Reinhold, C., and Forghani, R. (2020). Machine learning algorithm validation: from essentials to advanced applications and implications for regulatory certification and deployment. *Neuroimaging Clinics*, 30(4):433–445. DOI: 10.1016/j.nic.2020.08.004.
- Meena, R. C., Bhatia, S., Jhaveri, R. H., Shukla, P. K., Kumar, A., Varshney, N., and Malibari, A. A. (2023). Enhancing software-defined networks with intelligent controllers to improve first packet processing period. *Electronics*, 12(3):600. DOI: 10.3390/electronics12030600.
- Messinis, G. M. and Hatziaegyriou, N. D. (2018). Review of non-technical loss detection methods. *Electric Power Systems Research*, 158:250–266. DOI: 10.1016/j.epr.2018.01.005.
- Messinis, G. M., Rigas, A. E., and Hatziaegyriou, N. D. (2019). A hybrid method for non-technical loss detection in smart distribution grids. *IEEE Transactions on Smart Grid*, 10(6):6080–6091. DOI: 10.1109/TSG.2019.2896381.
- Modesto, W., Bastos, L., Venâncio Neto, A., Rosário, D., and Cerqueira, E. (2022). Towards automating the integration of legacy ieds into edge-supported internet of smart grid things. *Journal of Internet Services and Applications*, 12(1):33–46. DOI: 10.5753/jisa.2022.2374.
- Passos Júnior, L. A., Oba Ramos, C. C., Rodrigues, D., Pereira, D. R., de Souza, A. N., Pontara da Costa,

- K. A., and Papa, J. P. (2016). Unsupervised non-technical losses identification through optimum-path forest. *Electric Power Systems Research*, 140:413–423. DOI: 10.1016/j.epr.2016.05.036.
- Punmiya, R. and Choe, S. (2019). Energy theft detection using gradient boosting theft detector with feature engineering-based preprocessing. *IEEE Transactions on Smart Grid*, 10(2):2326–2329. DOI: 10.1109/TSG.2019.2892595.
- Ramana, K., Revathi, A., Gayathri, A., Jhaveri, R. H., Narayana, C. L., and Kumar, B. N. (2022). Wogru-ids—an intelligent intrusion detection system for iot assisted wireless sensor networks. *Computer Communications*, 196:195–206. DOI: 10.1016/j.comcom.2022.10.001.
- Ramos, C. C., Rodrigues, D., de Souza, A. N., and Papa, J. P. (2018). On the study of commercial losses in brazil: a binary black hole algorithm for theft characterization. *IEEE Transactions on Smart Grid*, 9(2):676–683. DOI: 10.1109/TSG.2016.2560801.
- Santos, H., Eugênio, P., Marques, L., Oliveira, H., Rosário, D., Nogueira, E., Neto, A., and Cerqueira, E. (2022). Internet of smart grid things (iosgt): Prototyping a real cloud-edge testbed. In *Proceedings of the 14th Brazilian Symposium on Ubiquitous and Pervasive Computing (SBCUP)*, pages 111–120, Porto Alegre, RS, Brazil. SBC. DOI: 10.5753/sbcup.2022.223205.
- Schäfer, P. and Leser, U. (2017). Fast and accurate time series classification with weasel. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 637–646. DOI: 10.1145/3132847.3132980.
- Wang, Y., Chen, Q., Hong, T., and Kang, C. (2019). Review of smart meter data analytics: Applications, methodologies, and challenges. *IEEE Transactions on Smart Grid*, 10(3):3125–3148. DOI: 10.1109/TSG.2018.2818167.
- Xi, X., Keogh, E., Shelton, C., Wei, L., and Ratanamahatana, C. A. (2006). Fast time series classification using numerosity reduction. In *Proceedings of the 23rd International Conference on Machine Learning, ICML '06*, page 1033–1040, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/1143844.1143974.
- Yip, S.-C., Tan, W.-N., Tan, C., Gan, M.-T., and Wong, K. (2018). An anomaly detection framework for identifying energy theft and defective meters in smart grids. *International Journal of Electrical Power & Energy Systems*, 101:189–203. DOI: 10.1016/j.ijepes.2018.03.025.
- Yip, S.-C., Wong, K., Hew, W.-P., Gan, M.-T., Phan, R. C.-W., and Tan, S.-W. (2017). Detection of energy theft and defective smart meters in smart grids using linear regression. *International Journal of Electrical Power & Energy Systems*, 91:230–240. DOI: 10.1016/j.ijepes.2017.04.00.
- Zanetti, M., Jamhour, E., Pellenz, M., Penna, M., Zambenedetti, V., and Chueiri, I. (2019). A tunable fraud detection system for advanced metering infrastructure using short-lived patterns. *IEEE Transactions on Smart Grid*, 10(1):830–840. DOI: 10.1109/TSG.2017.2753738.
- Zheng, Z., Yang, Y., Niu, X., Dai, H.-N., and Zhou, Y. (2017). Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids. *IEEE Transactions on Industrial Informatics*, 14(4):1606–1615. DOI: 10.1109/TII.2017.2785963.