Big Data-Driven Detection of False Data Injection Attacks in Smart Meters

FATIH ÜNAL, ABDULAZIZ ALMALAQ, SAMI EKICI, AND PATRICK GLAUNER

1Department of Energy Systems Engineering, Firat University, 23119 Elazığ, Turkey
2Department of Electrical Engineering, Engineering College, University of Ha’il, Ha’il 55476, Saudi Arabia
3Department of Applied Computer Science, Deggendorf Institute of Technology, 94469 Deggendorf, Germany
4Skyrocket.ai GmbH, 93055 Regensburg, Germany

Corresponding author: Fatih Ünal (funal@firat.edu.tr)

ABSTRACT Today’s energy resources are closer to consumers thanks to sustainable energy and advanced metering infrastructure (AMI), such as smart meters. Smart meters are controlled and manipulated through various interfaces in smart grids, such as cyber, physical and social interfaces. Recently, a large number of non-technical losses (NTLs) have been reported in smart grids worldwide. These are partially caused by false data injections (FDIs). Therefore, ensuring a secure communication medium and protected AMIs is critical to ensuring reliable power supply to consumers. In this paper, we propose a novel Big Data-driven solution that employs machine learning, deep learning and parallel computing techniques. We additionally obtained robust statistical features to detect the FDIs based cyber threats at the distribution level. The performance of the proposed model for NTL detection is investigated using private smart grid datasets in the Turkish distribution network for AMI-level cyber threats, and the results are compared to state-of-the-art machine learning algorithms used for NTL classification problems. Our approach shows promising results, as the accuracy, specificity, and precision metrics of most classifiers are above 90% and false positive rates vary between 0.005 to 0.027.

INDEX TERMS Advanced metering infrastructure, big data, false data injection, feature extraction, machine learning, non-technical losses, smart meter.

I. INTRODUCTION

Non-technical loss detection (NTL) plays an important role for many utilities to reduce electricity theft and take new measures to protect customers. NTL is a phenomenon that particularly occurs in emerging markets and is directly related to economic indicators. It is a significant proportion of total consumption and can vary from country to country. For example, proportions of 40% of the total electricity distributed have been estimated for Brazil [1]. The worldwide monetary value has been estimated to be around $100 billion per year. By country, electric utilities in India lose $4.5 billion each year to electricity theft [2]. Commercial losses in Brazil were reported to be $4.0 billion in 2011, according to the Brazilian Electricity Regulatory Agency [3]. In 2004, Tenaga Nasional Berhad, the electricity utility in Malaysia reported a revenue loss of $229 million per year due to faulty meters and electricity theft [4]. Apart from these examples, numerous examples can be listed worldwide, including in developed countries [1].

The literature reports various consequences of NTL for both utilities and customers [5], [6]. Among the most important are lost utility revenue and increases in electric rates for benign customers. In addition, power supply interruptions or instability problems in distribution systems may also be observed. From the utilities’ perspective, NTLs are considered a technical design problem that could be addressed through infrastructure improvement studies. Recent developments in smart grids, such as smart meters, have created new opportunities to address these issues. The smart grid enables intelligent control of the electricity supply chain via traditional and new technologies. In addition, smart meters can smoothly handle the entire bidirectional flow of data and energy by combining advanced high-resolution data collection capabilities.

However, cyberattacks and physical meter tampering can manipulate meter readings and other services in many different ways. Some of these include meter software tampering,
False Data Injection (FDI), disconnecting customers from the grid, or disrupting utility system operations. Physical meter tampering directly manipulates the physical components of smart meters, and this could be detected by expert technicians to place NTL codes. Physical tampering is common because it can be easily performed by users. However, sophisticated interventions such as FDI are more difficult to perform by individuals.

Modern energy systems have been tremendously upgraded with smart technology and advanced communication systems. Moreover, the daily activities of modern societies are heavily dependent on constant power consumption. Therefore, the problem of NTL in modern societies is an important techno-economic problem for electric utilities. In practice, the following activities have been reported in the literature [1]:

- Lowering consumption by manipulating meters.
- Bypassing meters from the power supply.
- Use of defective or faulty meters.
- Incorrect reading during data processing and billing.
- Technical or human misreading of meters.
- Access to unmetered electricity.

NTLs are very harmful to the entire power system, causing many direct and indirect negative impacts to utilities and energy providers. Loss of revenue and profit are the direct negative impacts of NTLs. Some of the impacts include expenses for irregular on-site inspections, unscheduled maintenance routes, and additional expenses for replacing faulty meters.

Most existing studies on NTL detection have focused on predicting energy consumption and customer profiles using historical energy consumption data, and a few employed Big Data techniques to improve predictive performance. However, there are still plenty of challenges when incorporating energy and load related parameters and characteristics into the NTL prediction model. In this paper, we address the following research questions:

1) How should smart power grids be modeled for NTL detection by using state-of-the-art Big Data and distributed computing technologies for classifiers?
2) How accurate is the corresponding anomaly detection by combining smart metering and observer metering with distributed computing?
3) What is the most viable way to find statistical features for NTL detection?
4) How quickly can the NTL detection model interact with the different levels of the power system, including customers and substations?

The main objective of this research is to present a new paradigm for modeling an NTL energy detection model that uses a large number of variables and features. The variables abnormal load peaks, zero falling loads and chaotic changes are FDI and are used to train the NTL detection model. The use of Big Data techniques and parallel computing provides new opportunities for analyzing large amounts of abnormal energy consumption, small fluctuations and accurate detection. The result of this work is an NTL energy detection model embedded in a Big Data framework that can be applied to many other anomaly detection problems, such as stock markets and smart cities.

Our main contributions include:

1) A novel comprehensive NTL detection model that considers parameters and characteristics of consumer load profiles.
2) A validation of Big Data techniques in conjunction with state-of-the-art classifiers and demonstration of their performance using evaluation techniques for NTL detection.

The modeling techniques presented in this paper are based on parallel computing and Big Data analysis. In this paper, we investigate the optimal statistical features for detecting FDI attacks, using Apache Spark integrated supervised learning. To validate the proposed abnormal energy consumption detection model, real data and advanced metering infrastructure (AMI) from the city of Diyarbakır, Turkey are used in this research.

II. THE STATE OF THE ART

In recent years, multiple surveys have reviewed the state of the art of NTL detection, including [1, 6]. In this section, we briefly summarize the most relevant works for this paper. In general, the conventional approach to NTL detection requires measurements of energy balance based on network topology and is therefore inefficient for modern distribution networks. Network topology evolves regularly due to the expansion of the distribution system in our modern societies. Recently, artificial intelligence (AI) and machine learning (ML) approaches have been used to detect NTL in smart grids as in by employing large computational capacities to train and test Big Data applications [7]–[10]. Several works based on different ML strategies (supervised, unsupervised and semi-supervised learning) have been studied in the field of NTL to classify fraud detection, anomaly detection, illicit connections, etc. The approaches based on ML use information about the customers’ load profiles to detect abnormal behavior, which is known to be highly correlated with NTL. Support vector machines (SVMs) have been widely used for binary classification and applied to NTL problems as in [11]–[14]. SVMs have also been combined with an evolutionary-based algorithm called Social-Spider Optimization [14]. Artificial neural networks (ANNs) have been used as a binary classifier to detect NTL in [15]–[18]. Recently, deep learning (DL) has shown promising classification performance and prediction accuracy for NTL detection problems [19]–[21]. A summary of the computing paradigms of some of the studies mentioned in this section are provided and compared to our approach in Table 1.

III. PROBLEM STATEMENT

A. NTL SCENARIOS USING DIFFERENT FDI CASES

NTL at the AMI level can be evaluated from three different aspects: physical intrusions, hardware or firmware
tampering or attacks on the communication infrastructure. In this research, we focus on the third type of communication attacks, which are widely used nowadays due to the insufficient regulation of smart grids. Identification of fraudulent consumers and on-site inspection is considered as a costly option. Moreover, the minority behaviour of these groups in the datasets leads to lower classification accuracy in NTL detection applications. Also, more sophisticated skills are required to manipulate the hardware and software equipment and are not feasible with conventional tools.

The cyber-attacks that modify measurements of smart meters are commonly referred to as FDI. Thus a fraudulent user reduces the quantity of reported electricity consumption of smart meters by simulating the tampering behaviors of NTL. In FDI attacks, there are many interventions that affect the process of calculating and reporting energy consumption. The main goal is to manipulate the reported power consumption without triggering an alarm [22], [23]. Table 2 presents a formal definition of six common FDI attacks and their mathematical formulations corresponding to the original consumption.

Variables $x, x_t, \tilde{x}_t$ represent the data point, the original and the manipulated load profile, respectively. The parameters $t, t_1, t_2$ are randomly set for each sample and the function $f(t)$ is a sequential output of $[0, 1]$. The six FDI attacks can be described as following:

- **FDI1**: A load profile is reduced using the same factor throughout the entire attack period.
- **FDI2**: Hourly readings above a threshold are clipped and the rest of values remain the same.
- **FDI3**: A threshold less than the peak value of a load profile is subtracted from all reported consumption values.
- **FDI4**: All consumption values of a load profile in a time interval longer than four hours are replaced by zero.
- **FDI5**: Each consumption value of a load profile is modified by a factor between 0.2 to 0.8.
- **FDI6**: A factor for each value of a load profile is multiplied by the average consumption value of daily load profile.

### TABLE 1. Comparison of the current study with the previous works considering computing paradigm.

| Ref. | Model | Computing Paradigm | Summary |
|------|-------|--------------------|---------|
| [4]  | SVM   | Not mentioned      | Real smart meter data of an electricity utility is used for NTL detection. The proposed system is developed as a GUI software using Microsoft Visual Basic 6.0. The computations are run on a Dell Poweredge workstation locally. The time elapsed for obtaining the results is approximately 1.8s per customer and varies based on the configuration of the workstation. |
| [22] | Consumption pattern-based energy theft detector (CPBETD) | Not mentioned | The authors used a publicly available dataset of over 5000 Irish homes and businesses. FDI samples are generated synthetically considering distribution transformers. The algorithm is designed to be run at low-sampling rates. |
| [20] | Hybrid deep neural network (HNN) | Spark integrated solutions | The authors used real smart meter data of Endesa, the largest electricity utility in Spain. PySpark is used as a distributed computing option for all the data preprocessing steps. The authors reported that the proposed methodology is currently deployed as an NTL detection tool in the utility with a precision of ~47% for new on-field inspections. |
| [10] | Edge computing enabled NTL fraud detection scheme (ENFD) | Map-Reduce | An edge computing framework is adopted to distribute the workload in parallel. Then each subtask is appointed to each smart meter to be solved locally. An edge device works as the master node and the smart meters work as the slave nodes. The authors designed a mapper and reducer function for ENFD. |

### TABLE 2. Six FDI attacks proposed in the literature [22]–[24].

| Method | Modification | Condition | Condition |
|--------|--------------|-----------|-----------|
| FDI1   | $\tilde{x}_t \leftarrow \alpha \cdot x_t$ | where $0.2 < \alpha < 0.8$ | $\alpha$ : same for all hourly readings |
| FDI2   | $\tilde{x}_t \leftarrow \begin{cases} x_t, & \text{if } x_t \leq \gamma \\ \gamma, & \text{otherwise} \end{cases}$ | $\gamma$ is a random cut-off point | $\gamma < \max x$ |
| FDI3   | $\tilde{x}_t \leftarrow \max \{x_t - \gamma, 0\}$ | $\gamma$ is a random cut-off point | $\gamma < \max x$ |
| FDI4   | $\tilde{x}_t \leftarrow f(t) \cdot x_t$ | $f(t) \leftarrow \begin{cases} 0, & \text{if } t_1 < t < t_2 \\ 1, & \text{otherwise} \end{cases}$ | $t_1 - t_2$ is a random time interval longer than 4 hours |
| FDI5   | $\tilde{x}_t \leftarrow \alpha_t \cdot x_t$ | where $0.2 < \alpha_t < 0.8$ randomly generated coefficient |
| FDI6   | $\tilde{x}_t \leftarrow \alpha \cdot \tilde{x}$ | where $0.2 < \alpha < 0.8$ randomly generated coefficient | $\tilde{x}$ : is the average consumption of daily load profile |

### B. NTL DETECTION ARCHITECTURE BASED ON BIG DATA ANALYTICS

Integrating traditional feature extraction, selection, and classification tasks with Big Data platforms is necessary in large-scale NTL discovery applications. Although the workflow of a typical NTL recognition process is well known in both industry and academia, the implementation specifications depend heavily on the knowledge and decisions of domain experts. With this in mind, and taking into account the needs of the R&D teams of distributors and the academic community of NTL, we have developed a general Big Data framework for this purpose, but also including these
specifications. Some of these specifications and expertise are briefly discussed below:

1) MASSIVE STATISTICAL FEATURE EXTRACTION PROCESS FOR SMART METER DATA

The feature extraction process for NTL generally relies on predefined handcrafted attributes. This situation is a common phenomenon observed in machine learning applications and limits the implementation of similar techniques in the broadest sense. In the NTL recognition literature, there is no specific package for this purpose and the existing solutions do not meet the requirements.

In this study, instead of explaining a particular package, we preferred to describe a set of similar packages that can be used for massive feature extraction in both local and distributed environments. In this way, practitioners can design the optimal machine learning pipeline considering their needs.

a: HIGHLY COMPARATIVE TIME-SERIES ANALYSIS (hctsa)
Includes 7764 features and allows useful and important features to be determined for various tasks. The algorithms can be run in both local and distributed modes in Matlab. This also helps researchers and practitioners to quantify and understand informative patterns in sequential datasets [25]. For detailed information, see [27], [28].

b: TIME SERIES FEATURE EXTRACTION ON BASIS OF SCALABLE HYPOTHESIS TESTS (tsfresh)
Computes a total of 794 features derived from 63 different time series characterization methods. The package supports local parallelization and distributed computing options, and allows jobs to be run on the Apache Spark framework. Also, the package includes additional feature selection methods based on automatically configured hypothesis tests. For detailed information, see [27], [28].

c: CESIUM
Extracts 112 different features from raw time series data. The package also allows the algorithms to be run in the browser via Docker containers. In addition, the package includes machine learning models and pipelines that can be used interactively within the web application.

d: FEATURETOOLS
Designed for automated feature development and allows the derivation of many features using basic transformations and mathematical operations from a set of relational database tables. Taking into account the master data of the loads connected to the smart grid, these predefined features can be suitably integrated into the package for the purposes of NTL detection, and more comprehensive features can be derived that can be easily adapted to the experts’ point of view. Detailed information can be found in [29].

2) NEIGHBORHOOD COMPONENT ANALYSIS (NCA) FOR FEATURE SELECTION PROCESS OF SMART METER DATA

Neighborhood components analysis (NCA) is a nonparametric distance learning method that aims to increase nearest neighbor classification accuracy [30]. It can also obtain a low-dimensional embedding of feature datasets, which can be used for fast classification and data visualization. The process of learning the distance metric optimizes the leave-one-out (LOO) classification error of the training dataset and helps in finding the feature weighting matrix using a regularization parameter. In this work, we opted for a similar annotation as in [30] to formulate the NCA method for feature selection.

Let $F = \{ (x_1, y_1), \ldots, (x_n, y_n) \}$ be a set of training data, where $x_i \in \mathbb{R}^p$ represents a $p$-dimensional feature vector, $y_i \in \{1, 2, \ldots, c\}$ corresponds to the true class label, $c$ is the number of classes, and $n$ is the total number of input samples. The weighting vector $w$ is a subset of features that contributes to minimizing the classification error, and the goal is to find this optimal weighting vector. The weighted distance $D_w$ between two instances $x_i$ and $x_j$ is

$$
D_w (x_i, x_j) = \sum_{k=1}^{p} w_k^2 |x_{ik} - x_{jk}|,
$$

where $w_k$ corresponds to a weight related to the $k$-th feature. The actual LOO classification accuracy of the nearest neighbor method is a non-differentiable function and chooses the nearest neighbor as the reference point. Instead, NCA prefers a differentiable cost function based on a probability distribution. The probability that $x_i$ chooses $x_j$ as the reference point is determined under the condition $i \neq j$ is

$$
p_{ij} = \frac{k (D_w (x_i, x_j))}{\sum_{m \neq i} k (D_w (x_i, x_m))},
$$

where $p_{ij} = 0$ if $i = j$ and $k (z) = \exp (-z / \sigma)$ is a kernel function. $\sigma$ is the kernel width that affects the probability that each sample is set as a reference point. As the kernel width $\sigma$ approaches 0, the nearest neighbor of $x_i$ is selected as the reference point. On the other hand, when the kernel width $\sigma$ approaches $+\infty$, all samples have the same probability. The probability that $x_i$ was correctly classified is

$$
p_i = \sum_j y_{ij} p_{ij},
$$

where $y_{ij} = 1$ if $y_i = y_j$ and $y_{ij} = 0$, otherwise. The approximate LOO accuracy is

$$
\xi (w) = \frac{1}{n} \sum_i p_i,
$$

where $\xi (w)$ is the actual LOO classification accuracy when the kernel width $\sigma$ approaches 0. The final objective function

1 https://github.com/strawlab/pyopy
2 https://github.com/cesium-ml/cesium
3 https://github.com/cesium-ml/cesium_web
4 https://featuretools.alteryx.com
conjugate with a regularization term to implement feature selection and reduce overfitting is

$$\xi(w) = \sum_i p_i - \lambda \sum_{k=1}^p w_k^2,$$

(5)

where $\lambda$ is a regularization term that can be adjusted via cross-validation (CV). Note that the regularization parameter $\lambda$ only changes the final solution vector and thus the coefficient $1/n$ in (4) can be neglected.

### C. DISTRIBUTED COMPUTING DESIGN

1) **APACHE HADOOP**

Apache Hadoop\(^5\) is a framework that uses simple programming techniques to perform large-scale distributed computation on big data. It is designed as a structure that can scale from a single node to thousands of nodes, each with local memory, storage, and CPU resources. Hadoop is an open source software platform that allows algorithms to run on clusters and has a distributed data storage capability. In addition, it has a fault-tolerant and scalable structure (e.g., similar storage, memory, and CPU resources used in parallel) for Big Data applications. Hadoop consists of three main components such as MapReduce [31], yet another resource negotiator (YARN) [32] and Hadoop distributed file system (HDFS) [33]. MapReduce is an approach that consists of two stages for parallel processing of large datasets: map() divides the dataset into independent stacks and reduce() performs the computations and combines the results. YARN manages the computational resources of the cluster and schedules the submitted jobs. YARN also aims to minimize network traffic between nodes while trying to provision computational resources as much as possible from the local disk where the data exists. YARN consists of three main components: Resource Manager, Node Manager and Application Master. In addition, the Hadoop ecosystem includes services designed to meet many other needs. Some of them can be summarized as follows: Apache Hive, which enables the use of SQL queries, Apache HBASE as a distributed NoSQL database, Apache Ambari for Big Data cluster management and code development environment, Apache Ranger for Big Data security and monitoring.

2) **APACHE SPARK**

Apache Spark is a general-purpose distributed computing framework that integrates with Java, Scala, Python, and R programming languages [34]. It has emerged as an alternative to the MapReduce paradigm of Hadoop to perform distributed computation at scale. Apache Spark is rapidly evolving as an open source project in the Big Data ecosystem with the support of contributors from academia and industry. The Spark framework consists of two main components: the Spark core, and parent libraries Spark SQL for structured data analysis and SQL, machine learning library (MLLib) for machine learning, GraphX for graph analysis, and Structured Streaming for stream processing. Spark also provides a data abstraction component, resilient distributed dataset (RDD), which allows the algorithms and pipelines to run in parallel.

### IV. MODEL

#### A. SPECIFICATIONS OF THE NTL DATASET

In this study, the load profiles of a group of consumers connected to the same substation are used to generate the NTL scenarios. The dataset covers the period from January 1, 2019 to December 31, 2019 and consists of 31 smart meter data and 1 observer meter data with hourly time resolution. The dataset contains the variables consumer ID, consumer type, time and consumption. Looking at the dataset by consumer type, 28 of the 31 smart meters are residential, 2 are schools, and 1 is a lightning rod. The output of the geographic information system (GIS) of the pilot area is shown in Fig. 1. The size of the dataset used in this study is limited to one substation taking into account the laws on privacy and data protection in Turkey [35].

Some mandatory precautions are required to analyze in depth the consumption patterns both at the distribution level and at the consumer level. These precautions include basic

\(^5\)https://hadoop.apache.org
FIGURE 3. Incoherent gaps between smart meter and observer meter measurements resulting from different phenomena.

data transformations and preprocessing steps that contribute to the formation of the final dataset. In this study, some preconditions are established to minimize potential problems that may occur as follows:

1) The total consumer data should be available more than 95% on a daily basis.
2) The total daily consumption of each consumer should be higher than 0.5 kW.
3) The hourly readings should be resampled as minutely as possible.

After the requirements are met and missing values are discarded, the dataset is a $7807 \times 1381$ dimensional matrix in which the rows correspond to the different daily load profiles and the columns correspond to the minutely time intervals. The row of the matrix is equal to 7807 after removing the daily load profiles which includes Not a Number (NaN) values. The column of the matrix is equal to 1381 after the consumption values between [00:00:00-23:00:00] time intervals are resampled using linear interpolation. This representation allows us to extract the informative hidden patterns of FDI samples that occurred in short time intervals.

Another phenomenon not often mentioned in the NTL literature is the challenge of time synchronization between smart meters and observation meters. In Fig. 2, there is an eight-hour delay between the substation measurements and the total smart meter measurements. Even if the latency (lag=0) is removed by moving the observation meter data forward on the time axis, as shown in Fig. 3, there are still non-overlapping and inconsistent gaps caused by different phenomena.

These two phenomena led us to develop an extraction-transformation-load (ETL) workflow for the dataset instead of considering the distribution line parameters. Moreover, when considering substations where loads with different consumption values coexist, as in Fig. 3, it may not be easy to integrate observer meter data at low consumption profiles into NTL discovery processes, since high consumption loads predominantly determine the substation load.

B. ADDITION OF FDI ATTACKS TO THE SMART METER DATA

The datasets used in NTL detection applications can be broadly divided into two categories: those that contain consumer master data (smart meters ID, note codes, tariff code, contract date, etc.) and monthly readings, and those that are artificially generated using various techniques. In recent years, the strict guidelines for sharing and analyzing consumer master data have created new trends such as the generation of synthetic samples using sophisticated artificial intelligence algorithms, such as generative adversarial networks (GANs) [37].

The FDI attacks mentioned in Section III-A are used to obtain NTL cases from smart meter datasets. The strategies used to generate modified profiles are as follows:

1) Selecting a random consumer on a random date and implementing a randomly selected FDI method on the original load profile.
2) Ensuring that the distribution of each method is almost equal and does not exceed half of the total value for each consumer.
3) The percentage of all modified load profiles should not exceed 30% of the total dataset.

An implementation of FDI attacks on a randomly selected consumer load profile and the modified version is depicted in Fig. 4.

C. DEVELOPMENT ENVIRONMENT

In this study, the components of Hadoop and Spark were installed on a Oracle VM VirtualBox using the Hortonworks data platform (HDP) 2.6.5 package. The HDP 2.6.5 package
runs on a virtual Linux/Red Hat (64-bit) operating system with default settings.

The server running Hadoop as a standalone application contains an AMD Ryzen 9 5900X 12-core processor, 32 GB DDR4 3000 MHz CORSAIR VENGEANCE LPX C16 RAM, a Samsung SSD 970 EVO Plus 500 GB hard drive, an NVIDIA GeForce RTX 2070 SUPER graphics card, and an MSI B450 TOMAHAWK MAX AM4 DDR4 motherboard.

Although the 2.6.5 version of HDP includes Apache Zeppelin notebook as the default development environment, we preferred to run the code in Anaconda Jupyter notebooks because it is easy to use and language independent. Jupyter notebooks were installed in the HDFS user’s file directory hdfs:/user/admin to easily manage the folders in the HDFS file system. Finally, the necessary configuration steps (e.g., exporting environment variables, setting permissions) were performed using the Apache Ambari interface. In this phase, we additionally installed and configured the necessary Python libraries (e.g. pyspark for integrating Python-Spark, Tensorflow for machine learning, elephas for parallel execution of deep learning algorithms, pyopy for massive feature extraction) in the Jupyter notebook environment.

D. NTL DETECTION USING BIG DATA ANALYTICS INFRASTRUCTURE

In this study, we developed an NTL structure integrated with the Hadoop and Spark frameworks. The NTL structure is a standalone application and consists of four main phases: data transfer from local repository to HDFS, transformations and actions using Spark, feature learning using distributed computing, and classification using traditional and state of art models. The main steps of the proposed NTL detection model are illustrated in Fig. 5.

First, the smart meter data is uploaded from the local Linux machine to the main HDFS repository and the necessary permissions are set to allow reads and writes for the standard HDFS users. The dataset is then loaded into the Spark session running on the Jupyter notebook. The Spark session is the access point to Spark functions, and YARN is the resource manager to execute the submitted jobs. In this study, the computational resources are allocated by the YARN manager and the tasks run in parallel.

The transformations and actions are the main components of Spark and help to build the dataset for a wide range of applications in a distributed manner. DataFrame and RDD transformations of the dataset were also implemented at this stage and composed using feature transformers like (VectorIndexer, VectorAssembler). We also used RDD data format in parallel DL classification stage to make iterative actions much faster.

The feature learning phase combines two processes that continue in this study to extract informative features that can be used in actual NTL applications. In the first phase, 7764 different features are computed using the hctsa package and in the second phase, the NCA feature selection algorithm is applied to the robust Sigmoid normalized dataset [26]. Since NTL detection processes are usually offline applications, we preferred the feature extraction package that provides the most features in this phase. The dataset was rearranged according to the instructions explained in the hctsa manual,\(^6\) and also tagged with labels and keywords describing the sample preconditions mentioned in Section IV-B.

Apache Sparks MLLib includes various linear methods, trees and ensemble methods, such as SVM, logistic regression (LR), random forest (RF), decision tree (DT), and naive bayes (NB). Other classifiers that are not included in Spark MLLib require additional installation and configuration steps. In this study, we explored some basic classifiers and modern classifiers that can be integrated with the Spark framework. Then, we compared the performance of the classifiers using various evaluation metrics for the FDI-based NTL applications. Moreover, we modeled a DL classifier running in parallel using the Elephas library. Using this library, we can model any neural network structure running in parallel in RDD format. In this study, the term DL classifier corresponds to an ANN model developed using the Keras library, as depicted in Fig. 6. In addition to the traditional models, we have also included the boosting methods of H\textsubscript{2}O.ai’s gradient boosting machine (GBM), eXtreme gradient boosting (XGBoost), Yandex’s CatBoost and Microsoft machine learning for Apache Spark (MMLSpark)’s LightGBM. In this way, we can study the performance of these classifiers and select the optimal solutions based on runtime, complexity and performance metrics.

V. RESULTS AND DISCUSSION

The main challenges evaluated in this section are as follows: uncertainties in feature extraction and description, class

\(^6\)https://hctsa-users.gitbook.io/hctsa-manual/
imbalance and insufficient evaluation criteria, data quality and covariate shift, scalability and comparison with multiple methods [1]. In this study, we determined 8 different evaluation metrics for performance comparison of classifiers as follows: accuracy, sensitivity (also named recall or true positive rate (TPR)), specificity, precision, FPR, $F_1$ score, Matthews correlation coefficient (MCC) [38] and Cohen’s Kappa coefficient $\kappa$ [39].

The definition of the metrics accuracy, sensitivity, specificity, precision and FPR are:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN},$$

(6)

$$\text{Sensitivity} = \frac{TP}{TP + FN},$$

(7)

$$\text{Specificity} = \frac{TN}{FP + TN},$$

(8)

$$\text{Precision} = \frac{TP}{TP + FP},$$

(9)

$$\text{FPR} = \frac{FP}{FP + TN},$$

(10)

where TP corresponds to the number of correctly classified samples for a given class, and TN is the total number of correctly classified samples of the remaining classes. In contrast, FP and FN correspond to the misclassified samples for a given class and the total number of misclassified samples of the remaining classes, respectively. $N = TP + FP + TN + FN$ represents the total number of samples in the confusion matrix.

The definition of the $F_1$ score is

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}}.$$  

(11)

The correlation coefficient between real and predicted classes, MCC, is

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}},$$

(12)

Cohen’s Kappa Coefficient $\kappa$ allows us to compare the performance of classifiers with that of a “random classifier” [39]. In the confusion matrix, a randomly selected label would be positive with probability $p_1$ and negative with probability $1 - p_1$, where $p_1 = TP + FN / N$. Similarly, a classifier generates a positive label with probability $p_2$ and a negative label with probability $1 - p_2$, where $p_2 = TP + FP / N$. Random accuracy is the probability that these two outcomes coincide by chance:

$$\text{Random accuracy} = p_1 p_2 + (1 - p_1)(1 - p_2).$$

(13)

The Cohen’s Kappa Coefficient $\kappa$ represents the relative improvement towards perfection from a random reference point:

$$\kappa = \frac{\text{Accuracy} - \text{Random accuracy}}{1 - \text{Random accuracy}}.$$  

(14)

A. FEATURE EXTRACTION AND DESCRIPTION FOR FDI ATTACKS BASED NTL SCENARIOS

The feature learning procedure of the proposed framework takes advantage of the hctsa package and its extensions. In this study, we have investigated suitable and robust statistical features in the hctsa package for detecting FDI-based NTL scenarios in detail. We computed all the features presented in the package for daily load profiles of all consumers and modified load profiles. After we cleaned missing values in the load profiles, the dataset includes a total of 7807 different load profiles from 31 consumers for 2019. As we show in Section V-B, we set the maximum NTL percentage of 30% in the dataset and then added 3000 new modified load profiles for which each FDI subset includes 500 samples. As a default option, the hctsa package computes all the features for each sample (load profile) and then removes the columns that contain special values such as (NaN, Inf, -Inf, []) or the samples that generate fatal errors, etc.) [25]. We then remove the columns with special values. Next, we apply robust Sigmoid normalization [26]

$$\tilde{sf} = \left[1 + \exp\left(-\frac{f - m_f}{1.35r_f}\right)\right],$$

(15)

where $\tilde{sf}$ represents the robust Sigmoid normalized feature values of the dataset, $f$ is the vector of non-normalized features, $m_f$ is the median of the values of $f$ and $r_f$ is its interquartile range.

We then obtain the data matrix of size $10807 \times 4269$ depicted in Figure 7. After the feature dataset is obtained, the optimal features are determined using an NCA-based feature selection strategy. The optimal regularization parameter $\lambda$ is determined using 5-fold cross-validation partition (CVP) and a stochastic gradient descent (SGD) solver. The loss function, misclassification error, is calculated for each selected $\lambda$ value as well as for each partition. In this study, the SGD based solver found a local minimum for the $\lambda$ value at trial 19 as $[\lambda = 0.0024, \text{mean loss} = 0.0699]$, as given in Table 3. The optimal value for $\lambda$ is again used for the final
FIGURE 7. Normalized feature dataset and important features of the dataset resulting from the NCA-based feature selection strategy. Each row sample of the feature dataset corresponds to a statistical output of different load profiles. Each column corresponds to a function of a predefined feature after removing special values such as (NaN, Inf, -Inf, []). The rows and columns are reordered to juxtapose similar rows using hierarchical linkage clustering.

decision using limited memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithm and the weak Wolfe line search strategy.

The regularization parameter $\lambda$ sets the weights of the irrelevant features to zero, as depicted in Fig. 7. Moreover, the magnitudes of the total weights of the features are comparable to each other, since we define only one regularization parameter for all features. This strategy allowed us to easily select the optimal features without making any assumptions about a threshold $\sigma$ value. The threshold $\sigma$ is set to 0.02 to include all features with non-zero weight in the final feature dataset.

In this study, the final feature dataset consists of a total of 32 robust statistical features obtained from hctsa. These features are listed in Table 4 with the features ID, the name of the feature and the importance of the feature. When we analyzed the content and category of the selected features in detail, we concluded that probability distributions and adjustment-based time series analysis attributes make up the majority of the group. In addition, some of the features commonly encountered in outlier analysis such as “z-score test” and “skewness of Pearson correlation coefficient” are included in the optimal feature dataset. As far as the authors are aware, there is no specified and recommended feature dataset for FDI-based NTL applications obtained from hctsa.

B. COMPARISON OF CLASSIFIERS

In this section, we examine the performance of a total of 10 classifiers against 8 evaluation metrics. Instead of using only one classifier and its variants, we preferred to increase the variety of classifiers that are integrated in the Apache Spark framework. In this section, we randomly split the feature dataset for each classifier into a training (80%) and a testing (20%) dataset and set the NTL ratio to 30% by default. We also validated the results with 5-fold CV for the classifiers that support the CV feature (XGBoost, GBM). Benchmarking NTL detection methods is challenging, as there are no publicly available NTL datasets [1]. Therefore, we used our own data set and also addressed practical implementations of classifiers in this study.

The results of the classifiers are shown in Table 5. The accuracy of most classifiers is above 90% and the NB classifier performs the worst on each performance metric. This is an expected result, as all NB classifiers make strong assumptions and assume that the value of a given feature is independent of the value of another feature. Another hypothesis can be derived from the distribution statistics of the winning classifiers for each performance criterion. Although the DL classifier is the winner when the classifiers are ranked in descending order of accuracy, $F_1$, and Kappa parameters, it can be seen that the DL classifier, which has slightly higher accuracy compared to boosting methods, is not the best on every performance criterion.

The important values of the performance criteria are highlighted in bold in Table 5. We selected the first four classifiers later in Section V-C to study the performance of the classifiers for different NTL ratios. Moreover, in this study, we focused more on finding out whether the configuration and implementation steps of the classifiers are straightforward for daily operation or not, as developing the optimal classifier model is beyond the scope.

The GBM and XGBoost classifiers have the lowest FPR rates and are the easiest to use of all the boosting algorithms thanks to the H2O.ai integration with the Spark framework. Moreover, the XGBoost classifier has the fastest training time among the boosting methods with 7.9 seconds. The training time of the remaining three classifiers is CatBoost with 13.4 seconds, LightGBM with 19.6 seconds and DL with 100.5 seconds for 100 epochs. The classifier that requires
TABLE 4. Descriptions of the selected optimal features using feature importance degree.

| No | Feature ID | Feature Name | Feature Importance | Feature ID | Feature Name | Feature Importance |
|----|------------|--------------|--------------------|------------|--------------|--------------------|
| 1  | 412        | IN-AutoMutarInfoStats-diff-20-krasovkov-1-4-amid | 1.5180      | 17          | 258         | CO-HistogramAM-quantile-5-2 | 0.4380 |
| 2  | 960        | ncut        | 1.3771              | 18          | 3870        | MF-AR-arcv-3-4-ag | 0.4744 |
| 3  | 41         | skewness-pearson | 1.3569          | 19          | 7256        | MF-armax-2-2-0-5-1-MA-1 | 0.4743 |
| 4  | 3          | harmonic-mean | 1.3003              | 20          | 6561        | MF-artif-1-8-sec-meanA | 0.4682 |
| 5  | 4127       | SC-FluctAnal-2-dfa-50-2-logi-srr | 1.1184      | 21          | 4131        | SC-FluctAnal-2-dfa-50-2-logi-exponent | 0.4613 |
| 6  | 87         | propUnique   | 0.7774              | 22          | 3334        | ST-LocalExtrema-150-minmax | 0.3873 |
| 7  | 932        | DN-FitKernelSmoothingMax | 0.7438      | 23          | 622         | SY-SlidingWindow-eat-em-5-10 | 0.3527 |
| 8  | 4156       | SC-FluctAnal-2-dfa-50-3-logi-mea | 0.7316      | 24          | 1502        | SB-TransitionAlphabet-20-ac-stdegexp-b | 0.3525 |
| 9  | 3887       | MF-AR-arcv-5-4-ag | 0.7218      | 25          | 3689        | MF-AR-arcv-4-3 | 0.3513 |
| 10 | 426        | IN-AutoMutarInfoStats-diff-20-krasovkov-1-4-amid5 | 0.7081      | 26          | 6460        | WL-coefs-dh-2-2-mean-coef | 0.3314 |
| 11 | 3245       | DN-OutlierInclude-abs-001-nexpsa | 0.6974      | 27          | 624         | SY-SlidingWindow-eat-em-10-2 | 0.2712 |
| 12 | 6511       | WL-fbm-p2    | 0.6503              | 28          | 2209        | SY-StNthDerChange-rmse | 0.2591 |
| 13 | 1454       | SB-TransitionAlphabet-40-1-maxdiagexp-rmose | 0.5916      | 29          | 607         | SY-SlidingWindow-eat-em-5-1 | 0.2333 |
| 14 | 3201       | DN-CompareKSF-it-every | 0.5705      | 30          | 3247        | DN-OutlierInclude-abs-001-nexpsc | 0.2321 |
| 15 | 3878       | MF-AR-arcv-4-4-ag | 0.5166      | 31          | 2674        | PH-Walkers-momentum-2-sw-propcross | 0.2101 |
| 16 | 3208       | DN-Compare_KSF-it-unolapant | 0.4921      | 32          | 1263        | CO-AddNoise-1-gaussian-firstUnder25 | 0.0669 |

C. EXAMINATION OF CLASS IMBALANCE AND INTERPRETATION OF EVALUATION CRITERIA

The class imbalance in NTL datasets leads to a skewed probability distribution for each class, so classifiers tend to include the minority groups in the major class. Moreover, the evaluation metrics may not correctly show the misleading results of the classifiers, which may lead to different interpretations of the results. The results of the four selected classifiers for different NTL ratios can be found in Table 6. The NTL ratio refers to the ratio of the number of modified load profiles to the total number load profiles in the datasets. The NTL ratios of (<3%, 3-5%, 5-10%, 10-20%, 20-30%) in the datasets represent additional included and randomly selected 210, 390, 750, 1500, 3000 modified load profiles, respectively. In these scenarios, the total number of real load profiles (RLPs) for each dataset is 7807.

As can be seen in Table 6, there is no single ultimate classifier that can be preferred for all NTL ratios. However, the CatBoost and DL classifiers generally performed well for NTL ratios of 5-10% and 10-20%, respectively. Also concluded that the performance of the four selected classifiers at different NTL ratios could not be accurately determined using the accuracy metric. This phenomenon, which is also mentioned in [1], is an insufficiently addressed problem of NTL applications with unbalanced datasets. In such a case, it may be helpful to examine the FPR metric to understand whether FDI samples are included in the RLP class. We found that as the NTL ratio decreases, the FPR value increases as more FDI samples are labeled as RLP in the test sets. The confusion matrices of selected classifiers at NTL ratio of 20-30% are depicted in Fig. 8. One reason that all four classifiers incorrectly labeled most of the FDI1 samples as RLP is that the modified load profiles also resemble RLP at a lower consumption level.

We examined the class imbalance problem in more detail at the substation level on a daily basis. In this scenario, we randomly selected 10 days for the year 2019 and then applied a randomly selected FDI attack to the corresponding day of a randomly selected consumer. The dataset consists of 10 FDI load profiles and 266 RLPs for the selected 10 days. As you can notice, the dataset is very unbalanced and there is only one FDI profile for each day at the substation level. These samples were completely removed from the training and testing datasets and tested separately. We then examined the performance of the DL classifier on this highly imbalanced dataset. The DL classifier misclassified 2 out of 276 samples, one FDI and one RLP. The implementation of the highly

TABLE 5. Comparison of classifiers for FDI based NTL applications.

| Classifier | Accuracy | Sensitivity | Specificity | Precision | FPR | F1 | MCC | Kappa |
|------------|----------|-------------|-------------|-----------|-----|----|-----|------|
| DL         | 0.973    | 0.939       | 0.989       | 0.958     | 0.010 | 0.948 | 0.940 | 0.892  |
| XGBoost    | 0.972    | 0.972       | 0.995       | 0.924     | 0.005 | 0.942 | 0.936 | 0.888  |
| CatBoost   | 0.972    | 0.900       | 0.985       | 0.934     | 0.014 | 0.931 | 0.930 | 0.885  |
| LightGBM   | 0.971    | 0.925       | 0.986       | 0.983     | 0.013 | 0.945 | 0.940 | 0.882  |
| GBM        | 0.969    | 0.979       | 0.994       | 0.920     | 0.005 | 0.940 | 0.935 | 0.876  |
| SVM        | 0.967    | 0.908       | 0.985       | 0.967     | 0.014 | 0.927 | 0.922 | 0.865  |
| LR         | 0.965    | 0.918       | 0.985       | 0.961     | 0.014 | 0.931 | 0.925 | 0.858  |
| RF         | 0.946    | 0.848       | 0.973       | 0.984     | 0.027 | 0.846 | 0.843 | 0.779  |
| DT         | 0.943    | 0.860       | 0.975       | 0.924     | 0.024 | 0.884 | 0.870 | 0.768  |
| NB         | 0.872    | 0.632       | 0.938       | 0.943     | 0.062 | 0.677 | 0.666 | 0.480  |

(>) : Larger value is better. (\(\downarrow\)) : Smaller value is better.
imbalanced dataset for May 30, 2019 is shown in Fig. 9 and Fig. 10.

In Fig. 10, the consumer ID represents each consumer by a number and the features are the optimal features previously described in Section V-A. As shown in Fig. 10, the consumer marked as a black rectangle represents an FDI-based NTL scenario and the remaining consumers are normal. The difference between the observer’s meter and the smart meter’s total measurement caused by the FDI-based NTL application is shown at the substation level in Fig. 9.

VI. FURTHER DISCUSSION

A. DATA QUALITY AND COVARIATE SHIFT

The data quality problem in NTL applications generally refers to mislabeled or poorly preserved information about real inspection results. In this study, we did not use consumers’ master data (e.g., meter type, connection type, tariff category). Instead, we used hourly readings from a pilot area that includes a substation and consumers connected to this substation in the Diyarbakır province, Turkey. In this pilot area, we simulated different FDI attack scenarios on smart meters of randomly selected consumers.

The issue of time synchronization between smart meter measurements and observation meter measurements and the incoherent gaps between these measurements results from the infrastructural problems at the distribution level of the city. Assuming that these obstacles are removed, the proposed robust statistical features and solutions integrated with Big Data analytics have the potential to enable widespread application at the city level.

Data quality also means the presence of data for a given time period. In this dataset, data availability is approximately 70% for 2019 and 3 out of 31 consumers have very low consumption levels in order to perform our calculations. These limitations are likely to remain an issue until full deployment and monitoring of smart meters is completed at the city level.

Covariate shift refers to the sampling bias of training and test data with different distributions [40], [41]. In this study, we were able to create as many FDI samples as we needed because we had mathematical formulations of the basic FDI attacks. Moreover, in this study we concluded that it is also possible to generate these samples using generative models such as conditional GANs [42]. In this case, it is only a matter of time before better algorithms for NTL detection are developed once publicly available NTL datasets are released by distributors.

B. SCALABILITY ANALYSIS

Assuming that the necessary installation and configuration requirements are already met before implementation, the feature extraction and selection phase is by far the most time-consuming. Since we computed all features provided in hctsa in this study, a total of approximately 83.9M different computations were performed for all samples of the dataset.
The total execution time of the feature extraction process was 27.1 hours for 7664 features. The determination of the optimal regularization parameter $\lambda$ and the feature selection process took 2.5 hours for 4269 features.

Since our goal is to determine the optimal features for FDI attack-based NTL applications, we evaluated the feature extraction and selection processes in the broadest sense. In day-to-day operations, the runtime from computations to final decision at the substation level can be reduced to minutes. In this study, we also found that most of the latency results from communication between different programming languages and additional interfaces. This latency and complexity between different programming languages can be eliminated using Spark SQL user defined functions (UDFs).

### VII. CONCLUSION

Recently, NTL detection in smart grid has become an important topic due to the increase of electricity theft worldwide. There have been many threats to consume electricity illegally, both via physical and cyber attempts. One of these attempts are FDIs that manipulate AMIs through cyber attacks. In this work, we proposed a novel framework for detecting NTLs that integrates Big Data solutions for statistical feature extraction. The proposed solution leverages several available machine learning, deep learning, and parallel computing libraries to model robust statistical feature extraction. Our robust statistical features can be used in the NTL detection phase and then be combined with the feature learning and classification phases. In the feature learning phase, two different processes are combined by using the Highly Comparative Time-Series Analysis (hctsa) and the Neighborhood Component Analysis (NCA) feature selection algorithms. Moreover, we take advantage of the transformation and action properties of the Apache Spark framework. Finally, the proposed statistical features are used in the classification phase.

We showed that most of the classifiers do not meet all the requirements, but CatBoost and Deep Learning classifiers generally performed well for the NTL ratios of 5-10% and 10-20%, and XGBoost achieved a promising classification result for the NTL ratios of 20-30%. In terms of the FPR metric, the XGBoost classifier has the lowest value as 0.005 for the NTL ratio of 20-30%. Although the highest value of the FPR metric is obtained from the CatBoost classifier for the NTL ratio of 3-5%, the specificity and the precision metrics remained within an acceptable range of over 95%. We also concluded that while the Directed Acyclic Graph (DAG) structure of Spark optimizes the execution plan of jobs submitted, it also accelerates both the learning and the convergence to optimal solutions compared to stand-alone applications. Although the Big Data solution approach is more demanding in terms of computational requirements and distribution of computational power, it has significantly outperformed the best conventional machine learning models.

### NOMENCLATURE

- $f$: vector of non-normalized features.
- $\alpha$: a randomly generated coefficient between [0.2, 0.8] for all hourly readings.
- $\bar{x}$: average consumption of daily load profile.
- $\gamma$: a random cut-off point.
- $\kappa$: Cohen’s Kappa Coefficient.
- $\lambda$: regularization parameter of Neighborhood Components Analysis (NCA).
- $\sigma$: kernel width.
- FN: total number of incorrectly classified samples of the rest of classes.
- FP: mislabeled samples for specific class.
- FP: positive probability of randomly selected label.
- TN: total number of correctly classified samples of the rest of classes.

### Table 6. Performance of selected four classifiers for different NTL ratios.

| NTL Ratio | Classifier | Accuracy (%) | Sensitivity (%) | Specificity (%) | Precision (%) | FPR (%) | F1 (%) | MCC | Kappa (%) |
|-----------|------------|--------------|-----------------|-----------------|--------------|---------|-------|-----|----------|
| <3%       | DL         | 0.995        | 0.836           | 0.981           | 0.821        | 0.019    | NaN   | 0.815 | 0.982    |
|           | XGBoost    | 0.996        | 0.816           | 0.981           | 0.836        | 0.019    | NaN   | 0.825 | 0.985    |
|           | CatBoost   | 0.991        | 0.724           | 0.974           | 0.850        | 0.026    | 0.756 | 0.761 | 0.964    |
|           | LightGBM   | 0.994        | 0.855           | 0.974           | 0.979        | 0.026    | 0.863 | 0.869 | 0.975    |
| 3-5%      | DL         | 0.990        | 0.819           | 0.980           | 0.844        | 0.019    | 0.828 | 0.815 | 0.962    |
|           | XGBoost    | 0.989        | 0.877           | 0.976           | 0.998        | 0.024    | 0.839 | 0.903 | 0.958    |
|           | CatBoost   | 0.987        | 0.817           | 0.960           | 0.954        | 0.039    | 0.824 | 0.821 | 0.947    |
|           | LightGBM   | 0.993        | 0.870           | 0.974           | 0.999        | 0.025    | 0.890 | 0.894 | 0.970    |
| 5-10%     | DL         | 0.984        | 0.881           | 0.981           | 0.938        | 0.018    | 0.903 | 0.894 | 0.935    |
|           | XGBoost    | 0.990        | 0.874           | 0.985           | 0.993        | 0.015    | 0.890 | 0.900 | 0.960    |
|           | CatBoost   | 0.994        | 0.925           | 0.991           | 0.991        | 0.009    | 0.948 | 0.948 | 0.978    |
|           | LightGBM   | 0.988        | 0.882           | 0.984           | 0.952        | 0.016    | 0.903 | 0.900 | 0.951    |
| 10-20%    | DL         | 0.981        | 0.898           | 0.986           | 0.956        | 0.014    | 0.922 | 0.915 | 0.924    |
|           | XGBoost    | 0.976        | 0.879           | 0.983           | 0.952        | 0.016    | 0.898 | 0.896 | 0.905    |
|           | CatBoost   | 0.977        | 0.863           | 0.979           | 0.963        | 0.021    | 0.882 | 0.887 | 0.905    |
|           | LightGBM   | 0.975        | 0.896           | 0.980           | 0.984        | 0.019    | 0.918 | 0.917 | 0.901    |
| 20-30%    | DL         | 0.973        | 0.939           | 0.989           | 0.958        | 0.010    | 0.948 | 0.940 | 0.892    |
|           | XGBoost    | 0.972        | 0.972           | 0.994           | 0.994        | 0.005    | 0.942 | 0.936 | 0.888    |
|           | CatBoost   | 0.971        | 0.909           | 0.985           | 0.994        | 0.014    | 0.931 | 0.930 | 0.885    |
|           | LightGBM   | 0.971        | 0.925           | 0.986           | 0.984        | 0.013    | 0.945 | 0.940 | 0.882    |

* Higher value is better. Lower value is better.
TP correctly classified samples for specific class, True Positive.

$\bar{F}$ robust Sigmoid normalized feature values of dataset.

$\tilde{x}_t$ manipulated load profile.

$\xi (w)$ leave one-out (LOO) classification accuracy.

$c$ number of classes.

$D_w$ weighted distance between two instances.

$F (t)$ a set of training dataset.

$f (z)$ sequential output of time series $[0, 1].$

$k (z)$ kernel function of $z.$

$n$ total number of input samples.

$P_{ij}$ probability that $x_i$ chooses $x_j$ as the reference point under the condition $(i \neq j).$

$p_i$ probability of $x_i$ being accurately classified.

$t$ time interval of FDI attack.

$t_1$ initial time of FDI attack.

$t_2$ end time of FDI attack.

$W_f$ weight vector of feature subset.

$w_k$ weight related to kth feature.

$x$ data point of daily load profile.

$x_i$ p-dimensional feature vector, $x_i \in \mathbb{R}^p.$

$y_i$ true class label of training dataset, $y_i \in \{1, 2, \ldots, c\}.$

$\alpha_t$ a randomly generated coefficient between $[0.2, 0.8]$ at the point of time $t.$

$\text{IQR}$ interquartile range of sf.

$\text{r}_1$ median values of sf.

$\text{r}_2$ interquartile range of sf.

$\text{L}_t$ original load profile.

ACKNOWLEDGMENT

The authors would like to thank Mahmut Sami Sarac and Tolga Koc (Department of Automatic Meter Reading (AMR) System, Dicle Elektrik Perakende Satış A.Ş.) for providing helpful insights and supports for this project.

REFERENCES

[1] P. Glauner, J. A. Meira, P. Valtech, R. State, and F. Bettinger, “The challenge of non-technical loss detection using artificial intelligence: A survey,” Int. J. Comput. Intell. Syst., vol. 10, pp. 760–775, Feb. 2017.

[2] T. Ahmad, H. Chen, J. Wang, and Y. Guo, “Review of various modeling techniques for the detection of electricity theft in smart grid environment,” Renew. Sustain. Energy Rev., vol. 82, pp. 2016–2933, Feb. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1364032117314090

[3] C. C. O. Ramos, D. Rodrigues, A. N. de Souza, and J. P. Papa, “On the study of commercial losses in Brazil: A binary black hole algorithm for theft characterization,” IEEE Trans. Smart Grid, vol. 9, no. 2, pp. 676–683, Mar. 2018.

[4] J. Nagi, K. S. Yap, F. Nagi, S. K. Ting, S. P. Koh, and S. K. Ahmed, “NTL detection of electricity theft and abnormalities for large power consumers in TNB Malaysia,” in Proc. IEEE Student Conf. Res. Develop. (SCoReD), Dec. 2010, pp. 202–206.

[5] S. S. S. R. Depuru, L. Wang, and V. Devabhaktuni, “Electricity theft: Overview, issues, prevention and a smart meter based approach to control theft,” Energy Policy, vol. 39, pp. 1007–1015, Feb. 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S030142151000861X

[6] G. M. Messinis and N. D. Hatziargyriou, “Review of non-technical loss detection methods,” Electr. Power Syst. Res., vol. 158, pp. 250–266, May 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378779618300051

[7] G. Figueroa, Y.-S. Chen, N. Avila, and C.-C. Chu, “Improved practices in machine learning algorithms for NTL detection with imbalanced data,” in Proc. IEEE Power Energy Soc. Gen. Meeting, Jul. 2017, pp. 1–5.

[8] J. Naki, K. S. Yap, S. K. Ting, S. K. Ahmed, and M. Mohamad, “Non-technical loss detection for metered customers in power utility using support vector machines,” IEEE Trans. Power Del., vol. 25, no. 2, pp. 1162–1171, Apr. 2010.

[9] P. Glauner, J. A. Meira, L. Döbbert, R. State, F. Bettinger, and Y. Rangoni, “Neighborhood features help detecting non-technical losses in big data sets,” in Proc. 3rd IEEE/ACM Int. Conf. Big Data Comput. Appl. Technol., Dec. 2016, pp. 253–261.

[10] W. Han and Y. Xiao, “Edge computing enabled non-technical loss fraud detection for big data security analytic in smart grid,” J. Ambient Intell. Humanized Comput., vol. 11, no. 4, pp. 1697–1708, Apr. 2020.

[11] J. Nagi, A. M. Mohammad, K. S. Yap, S. K. Tingon, and S. K. Ahmed, “Non-technical loss analysis for detection of electricity theft using support vector machines,” in Proc. IEEE 2nd Int. Power Energy Conf., Dec. 2008, pp. 907–912.

[12] S. S. S. R. Depuru, L. Wang, and V. Devabhaktuni, “Support vector machine based data classification for detection of electricity theft,” in Proc. IEEE/PES Power Syst. Conf. Expo., Mar. 2011, pp. 1–8.

[13] K. S. Yap, I. Z. Abidin, A. R. Ahmad, Z. F. Hussien, H. L. Pok, F. I. Iosail, and A. M. Mohamad, “Abnormalities and fraud electric meter detection using hybrid support vector machine & genetic algorithm,” in Proc. 3rd IASTED Int. Conf., Adv. Comput. Sci. Technol., 2007, pp. 388–392.

[14] D. R. Pereira, M. A. Pazoti, L. A. Pereira, D. Rodrigues, C. O. Ramos, A. N. Souza, and J. P. Papa, “Social-spyder optimization-based support vector machines applied for energy theft detection,” Comput. Electr. Eng., vol. 49, pp. 25–38, Jan. 2016.

[15] T. Xu, G. Coco, and J. Gao, “Extraction of urban built-up areas from nighttime lights using artificial neural network,” Geocarto Int., vol. 35, no. 10, pp. 1049–1066, Jul. 2020.

[16] L. A. M. Pereira, L. C. S. Afonso, J. P. Papa, Z. A. Vale, C. C. O. Ramos, D. S. Gastañado, and A. N. Souza, “Multilayer perceptron neural networks training through charged system search and its application for non-technical losses detection,” in Proc. IEEE PES Conf. Innov. Smart Grid Technol. (ISGT Latin America), Apr. 2013, pp. 1–6.

[17] V. Ford, A. Siraj, and W. Eberle, “Smart grid energy fraud detection using artificial neural networks,” in Proc. IEEE Symp. Comput. Intell. Appl. Smart Grid (CIASG), Dec. 2014, pp. 1–6.

[18] D. Labate, P. Giubbini, G. Chiocco, and F. Pigione, “Shape: The load profiling and non-technical losses modules,” in Proc. Congrès Int. Des Réseaux Electriques de Distribution (CIRED), 2015, pp. 1428–1435.

[19] X. Lu, Y. Zhou, Z. Wang, Y. Yi, L. Feng, and F. Wang, “Knowledge embedded semi-supervised deep learning for detecting non-technical losses in the smart grid,” Energies, vol. 12, no. 18, p. 3452, Sep. 2019.

[20] M.-M. Munteanu, J. Tejedor-Aguilera, P. Cruz-Romero, and A. Gomez-Exposito, “Hybrid deep neural networks for detection of non-technical losses in electricity smart meters,” IEEE Trans. Power Syst., vol. 35, no. 2, pp. 1254–1263, Mar. 2020.

[21] N. Ayub, K. Aurangzeb, M. Awais, and U. Ali, “Electricity theft detection using CNN-GRU and manta ray foraging optimization algorithm,” in Proc. IEEE 23rd Int. Multitopic Conf. (INMIC), Nov. 2020, pp. 1–6.

[22] P. Jokar, N. Arianpour, and V. C. M. Leung, “Electricity theft detection in AMI using customers’ consumption patterns,” IEEE Trans. Smart Grid, vol. 7, no. 1, pp. 216–226, Jan. 2016.

[23] M. Zanetti, E. Jamhour, M. Pellenz, M. Penna, V. Zambenedetti, and I. Chuieiri, “A tunable fraud detection system for advanced metering infrastructure using short-lived patterns,” IEEE Trans. Smart Grid, vol. 10, no. 1, pp. 830–840, Jan. 2019.

[24] Y. Wang, Q. Chen, and C. Kang, “Smart Meter Data Analytics: Electricity Consumer Behavior Modeling, Aggregation, and Forecasting,” Singapore: Springer, 2020.

[25] B. D. Fulcher and N. S. Jones, “Hetsa: A computational framework for automated time-series phenotyping utilizing massive feature extraction,” Cell Syst., vol. 5, no. 5, pp. 527–531, Nov. 2017.

[26] B. D. Fulcher, M. A. Little, and N. S. Jones, “Highly comparative time-series analysis: The empirical structure of time series and their methods,” J. Roy. Soc. Interface, vol. 10, no. 83, Jan. 2013, Art. no. 20130048, doi: 10.1098/rsif.2013.0048.
Y. Liu and S. Hu, “Cyberthreat analysis and detection for energy theft in smart grids, smart cities, and smart energy systems utilizing artificial intelligence, machine learning, and deep learning. He is the Vice-Chair of the IEEE Saudi Arabia Signal Processing Society Chapter.

ABDULAZIZ ALMALAQ (Member, IEEE) received the B.S. degree in electrical engineering from the College of Engineering, University of Ha’il, Saudi Arabia, in 2011, and the M.S. and Ph.D. degrees in electrical engineering from the Department of Electrical and Computer Engineering, University of Denver, CO, USA, in 2015 and 2019, respectively. He is currently an Assistant Professor with the Department of Electrical Engineering, University of Ha’il. He has published more than ten scientific articles in highly ranked engineering journals and contributed to several conference papers at distinguished conference proceedings. His research interests include modeling smart buildings, smart grids, smart cities, and smart energy systems utilizing artificial intelligence, machine learning, and deep learning. He is the Vice-Chair of the IEEE Saudi Arabia Signal Processing Society Chapter.

SAMİ EKİCI (Member, IEEE) received the B.S. and M.S. degrees in electrical education and the Ph.D. degree in electrical and electronics engineering from Firat University, Turkey, in 2000, 2003, and 2007, respectively. From 2000 to 2011, he was a Research Assistant with Firat University. Since 2016, he has been a Professor with the Department of Energy Systems Engineering, Firat University. He has authored more than 45 articles. His research interests include digital signal processing, machine learning methods, and electrical power systems, especially transmission line fault studies and renewable energy resources.

PATRİCİ GŁAUNER received the B.Sc. degree in computer science from the Karlsruhe University of Applied Sciences, the M.Sc. degree in machine learning from Imperial College London, the M.B.A. degree from the Quantic School of Business and Technology, and the Ph.D. degree in computer science from the University of Luxembourg. He is an Alumnus of the German National Academic Foundation (Studienstiftung des deutschen Volkes). He is currently a Full Professor of artificial intelligence with the Deggendorf Institute of Technology, Bavaria, Germany. As an expert witness, he advised the German Federal Parliament and the French National Assembly on AI. His works on AI were featured by New Scientist, McKinsey, Imperial College London, Udacity, the Luxembourg National Research Fund, Towards Data Science, and others. He was previously the Head of the Data Academy at Alexander Thamm GmbH, the Innovation Manager for artificial intelligence at Krones Group, and a Visiting Researcher with the University of Quebec in Montreal (UQAM). He was a fellow of the European Organization for Nuclear Research (CERN).

* * *