Electric Power Systems Research 208 (2022) 107895

Contents lists available at ScienceDirect

Electric Power Systems Research

journal homepage: www.elsevier.com/locate/epsr

Online electricity theft detection framework for large-scale smart grid data

Soroush Omidvar Tehrani *, Afshin Shahrestani, Mohammad Hossein Yaghmaee

Department of Computer Engineering, Ferdowsi University of Mashhad, Iran

ARTICLE INFO

Keywords:
Electricity theft detection
Anomaly detection framework
Smart grid
Gradient boosting
Online processing

ABSTRACT

Smart grid gives more control and information to the utility companies. However, it can be leveraged for data manipulation, which can lead to new techniques in electricity theft. This paper presents an electricity theft detection framework, designed for handling real-time large-scale smart grid data to address these new emerging threats. It uses a hybrid approach, combining the information inferred by analyzing the reported data from distribution transformer meters with machine learning algorithms to discover fraudulent activity. We added an additional form of attack to the six previously known patterns and generated malicious variants of consumption data to solve the problem of imbalanced dataset classes, resulting in more accurate classifiers. The framework also allows for a trade-off between the detection rate and triggered false alarms by using a sliding window in the decision-making process. In the end, the proposed framework is evaluated using well-known clustering and classification methods in a practical scenario, resulting in outcomes superior or equal to the previously achieved scores while having the advantages of online and distributed processing.

1. Introduction

Electricity is an essential part of humans lives now, with almost everybody having access to and getting their lives affected by it. With its widespread usage in all parts of the world, there are some concerning matters that need to be addressed, such as electrical loss. According to the Distribution Systems Operators (DSOs) point of view, this electrical loss can be categorized into two main groups, Technical Loss (TL) and Non-Technical Loss (NTL). Technical loss is any sort of loss that can occur inside of the distribution network as a result of cables, transformers and other devices used in the transfer of electricity. Non-technical loss is any unbilled electricity attributed to several factors [1]. A prime example of NTL is energy theft, a phenomenon that imposes considerable financial damages (more than $25 billion annually [2]) and is the reason for technical consequences such as voltage violations.

Owing to the better and increased on-site inspection and the development and adoption of the smart grid, older malicious approaches in energy theft, for example, bypassing the meter, are now becoming obsolete and being replaced by tampered data transmission, created before, after, or even inside the smart meters. In [3], malicious attempts are classified using several detection techniques into three categories: cyber attacks, physical attacks, and data attacks. Physical attacks require direct manipulation and tampering of the smart meter by users. Examples of these attacks are traditional meter bypassing, disconnecting the meters, and breaking into them. Cyber attacks include fraudulent virtual events such as modifying the firmware of the meters or compromising the network through remote network exploits, which cause manipulations in communication channels and links. Lastly, data attacks are a product of cyber and physical attacks and try to change the measured values of meters. Cutting the reported values by a percentage and reporting zero consumption are examples of these types of attacks.

Although traditional methods might have previously worked in NTL detection, nowadays, as a result of the enormous amount of generated data and variety of anomalies, Machine Learning (ML) methods yield better results while being more time-efficient.

ML models performances are affected by several factors. One of these factors is the sampling rate. The sampling rate is defined as the rate at which data, in this case, power consumption data, was reported. In general, the more data there is for training, the higher ML models accuracy is. A high sampling rate provides a larger training set. However, it may threaten the privacy of customers and result in inferred details from the users personal life. Study [4] further studies the effects of different time resolutions and their impact on users privacy. Thus, we aggregated the data to acceptable sampling rates in our research to both respect the users privacy and keep the results accurate, while making our research compatible with other datasets.

Another crucial factor is the existence of malicious data in the dataset. The majority of ML models need to be trained on both malicious

* Corresponding author.
E-mail addresses: omidvar@mail.um.ac.ir (S. Omidvar Tehrani), afshin.shahrestani@alumni.um.ac.ir (A. Shahrestani), hyaghmae@um.ac.ir (M.H. Yaghmaee).

https://doi.org/10.1016/j.epsr.2022.107895
Received 9 November 2021; Received in revised form 23 February 2022; Accepted 26 February 2022
Available online 3 March 2022

and honest data to have optimum performance. Most available datasets are gathered from volunteers, and these datasets probably lack malicious samples in them. This class imbalance in the data can have significant negative effects on the performance of supervised classification models [5]. To answer this problem, researchers often use malicious sample generators to inject honest datasets with artificial anomalies. These functions take honest data as input, and after some operations, create various attack patterns from them.

A critical factor in ML models performance is the amount of computational power and time given to them. Due to the massive amount of the generated data by power consumption and the abundance of users, the traditional deployment of ML models on one system is impractical. An alternative is distributed systems. If one system is tasked with running the models on large datasets, it would require enormous computational power, coming from powerful and expensive hardware, which is both financially inefficient and, depending on the dataset, unfeasible. There is also the matter of online data processing. In this case, due to the rate of receiving data and the scale of that, centralized computing becomes near impossible. On the other hand, distributed systems can be easily scaled based on the tasks, meaning no unnecessary costs for powerful hardware. They are far more robust against system crashes and unwanted data deletion, and can be optimized for online processing. Distributed algorithms allow researchers, companies, and organizations to make informed decisions and draw meaningful conclusions from large amounts of data without inflicting unnecessary financial pressure. Although there are many benefits to distributed computing, it is not free of challenges. Some ML algorithms are not designed to be performed in a distributed manner. Besides, the data passing between different nodes in the network can be time-consuming.

Finally, the abundance of possible honest consumption patterns due to the variety of users can be challenging to ML models. Consumers based on their family size, usage, active electrical appliances, job, and many other factors can have different consumption patterns. For example, a user who works during night hours from home has vastly different consumption patterns than a three-member household. Hence, methods such as clustering algorithms can be helpful in differentiating users' consumption.

This paper aims to introduce an applicable anomaly detection framework on real customers' online power usage data. The framework is designed with the issues surrounding theft detection in consumption data and the implementation of machine learning models in them. Machine learning models are used to analyze the user's data for different consumption patterns and the detection of different types of attack patterns in the network, especially data attacks. To solve the problem of large-scale data processing and the computational and storage overhead, these machine learning models and methods are performed in a distributed space to use their advantages, such as scalability, parallelism, and reliability. Another point considered in the framework design is the capability to process the users' online data, allowing for re-evaluation of the model's understanding of a user's consumption patterns in case of major changes, hence, the applicability of the framework in industrial use cases. In this paper, we proposed a framework consisting of a training and a testing phase. In the former, the framework calculates additional features based on the available users consumption data to increase the accuracy. Next, it has a clustering phase, performed on honest data to discover various distributions. After that, the framework uses malicious sample generator functions to inject generated data and anomalies into the dataset. In the last step of the training phase, classifiers are trained using the generated data to classify the user's behavior. In the testing phase, real-time data of users are classified by the trained models to discover suspicious users. In parallel, each users neighborhood NTL is estimated to check for fraudulent activities in the user's vicinity. With the combination of these two detection methods, the users are analyzed for possible electricity theft.

2. Related works

Non-Technical Loss (NTL) detection methods can be categorized into three groups: data-oriented, network-oriented, and hybrid methods [6].

Network-oriented approaches use the data provided by the grid distribution sensors and the information related to the network, such as network topology, to detect NTL. Examples of network-oriented approaches can be seen in [7–9]. On the other hand, data-oriented methods are built solely based on data analytics and machine learning techniques. These methods are typically separated into three sub-categories: supervised methods, unsupervised methods, and semi-supervised methods. The choice of which methods can be used in detecting NTL and electricity theft mainly depends on the data itself and whether labeled data (data that has already been recognized as honest or fraudulent) is available. Most data-oriented approaches need to go through training and testing phases, while unsupervised methods only use labeled data in evaluation. Some of the examples of supervised data-oriented anomaly detection in power consumption data are explained below:

One of the main algorithms used in supervised learning is Support Vector Machine (SVM). SVMs are somewhat robust against imbalanced data and are easy to implement. In [10], a pattern-based energy theft detection system based on honest and malicious consumption patterns was proposed, in which they used the predictability of customers non-malicious and malicious behaviors. Their algorithm, called CPBETD, consists of 2 phases, training and testing. Technical Loss is estimated during the training phase, and an SVM is trained based on the malicious data generated from the honest dataset to classify anomalous and honest patterns. Next, in the training phase, the TL of the data is measured, and the SVM is performed on data. If either TL estimator or SVM detects NTL in the data, appropriate actions are taken.

Decision Trees, and in general, tree-based methods, are another group of machine learning algorithms used in NTL detection. Decision Trees generate a set of rules to classify the dataset. One of the main advantages of these methods is their understandability compared to other methods. Literature [11] uses a decision tree model to detect potentially fraudulent users based on their profiles. Two types of fraudulent behavior are discussed in this paper, reporting less power consumption than the honest value and reporting more power consumption than the honest value. After a decision tree is trained on the data, generated rules are then used to predict the consumption. Root Mean Squared Error (RMSE) is used to measure the difference between the actual value and the predicted value, and the threshold on their difference determines the existence of fraudulent behaviour on the data.

Paper [12] is another example of tree-based methods in NTL detection. It proposed a Gradient Boosting Theft Detector (GBTD), inspired by the work in [10], based on extreme gradient boosting (XGBoost), categorical boosting (CatBoost), and light gradient model (LightGBM). They did their research by focusing on feature engineering-based preprocessing and considering time complexity. It is also noteworthy that this paper made use of statistical features such as standard deviation, mean, maximum, minimum to improve Detection Rate (DR) and False Positive Rate (FPR).

Study [13] proposes an electricity theft detection method consisting of four steps: missing value interpolation, data balancing, feature extraction, and fraudulent behavior classification. The dataset used in this study was already separated into a normal and a fraudulent class, meaning there is no need for anomaly and theft injection. Three tree-based classifiers, namely Decision Tree, Random Forest, and AdaBoost, were used to classify customers' usage, and a Bayesian optimizer found the best hyperparameters for each method.

In [14], the class imbalance in energy theft in older power systems, resulting from the abundance of honest data, is addressed. Due to the lower number of energy theft data in datasets, standard models tend to ignore the malicious data and focus on honest data points. This paper combined several models, such as one-class SVMs, optimum path forest,

and C4.5 decision tree, to improve the detection rate of previous approaches. In the end, the combined approach achieved a 2%-10% improvement over individual classifiers. However, this slight performance improvement did not justify the added computational cost. Paper [10] tried to solve this problem of imbalanced data by generating malicious data to provide a dataset from different classes so the data imbalance could be fixed.

Another group of methods used in both supervised and unsupervised learning is Deep Learning algorithms and networks. Literature [15] proposed a novel hybrid method, consisting of a Convolutional Neural Network (CNN) for feature extraction and Random Forest (RF) to detect energy theft in power consumption data automatically. In their research, a convolutional neural network was designed to learn features from the different hours of the day and different days using convolution and downsampling. They also added a dropout layer to reduce the risk of overfitting. After that, a random forest is trained based on the features discovered by the CNN to detect anomalies in the power consumption. To create this hybrid system, a grid search algorithm is used to determine the optimum parameters.

Study [16] tried to improve the existing supervised anomaly detection approaches in power consumption data using feature engineering. It proposed a practical and model-agnostic feature engineering solution for fraud detection in AMI. They combined a Finite Mixture Model clustering and an evolutionary Genetic Programming Algorithm to create a set of features that can show the dynamics of demand over time, and in comparison with similar households to make anomaly and fraud detection easier. More than 4000 users half-hourly consumption data for a duration of 18 months were used in this study which was separated into five segments of households using the clustering algorithm. This new feature engineering approach was then integrated into several machine learning algorithms to test if there are any performance improvements in them. The results of using the feature engineering architecture alongside machine learning algorithms showed noticeable improvements in performance, while being computationally practical and compatible with existing supervised theft detection methods. Especially in cases of zero-day attacks, unseen attacks, and small-magnitude electricity theft, this architecture was effective.

Although supervised learning models are helpful, they might not perform optimally in NTL detection. Due to the abundance of consumption patterns, variety of malicious attacks, and lack of labeled data, most researchers use malicious attack injection into their datasets to train them. Unsupervised and semi-supervised methods require little to no labeled data at all. Some of these methods usage is NTL detection are written below:

Forecasting and predicting electricity consumption using regression is another unsupervised form of data-oriented NTL detection. In [17], the use of ARMA and ARIMA, two autoregression forecasting methods, were proposed to validate the consumption values. First, they showed the ineffectiveness of ARMA in power consumption data, due to its non-stationary nature. Then, they calculated the first-degree difference of these readings to show this makes them weakly stationary, therefore, usable by ARIMA.

The use of statistical process control has also been proposed in the field of NTL detection. Statistical process control is the processing and monitoring of process conditions to determine its performance [18]. Literature [19] analyzed power consumption data in the form of time series without seasonal data to detect fraudulent behavior. In this work, statistical process control was used, and the process was represented as electricity usage. XMR charts were used to detect significant decreases in power consumption. To test these findings, the method was tested on a set of users power consumption data caught stealing electricity.

Hybrid approaches are the last group of methods used in NTL detection. Hybrid methods adopt a combination of data and network-oriented approaches in NTL detection to increase the accuracy of the results.

Article [10] combines SVM with observations of users meters to estimate the network's technical losses and calculate the non-technical loss. When determining whether a user has committed electricity theft, both the SVM and calculated NTL are considered.

3. Proposed approach

Electricity theft is a huge financial problem for utility companies, due to the unpaid usage. It overloads the generators, and the quality of electricity supply is adversely affected by it, since utility companies cannot calculate the amount of electricity they need to supply to their honest and illegal customers [2]. Owing to this extra cost imposed by electricity theft, the detection process should both perform well and be cheap. Falsely accusing customers of electricity theft can be really expensive, since each suspect of fraudulent behavior requires on-site inspection for proof. Our framework tries to detect these fraudulent behaviors online, while minimizing the required performance and cost. It combines network oriented and data-oriented approaches to achieve a hybrid detection method, increasing the accuracy of anomaly detection and reducing the false positive rate (FPR). The process of anomaly detection by our framework can be separated into two main sections: training and testing phase.

3.1. Training

In this section, we will discuss the training process of our framework. The training phase itself consists of 3 different parts, which can be seen in Fig. 1. Each of these parts is discussed below:

3.1.1. Preprocessing

Having an unprepared dataset can ruin the result of research, even if the model is highly accurate. That is why preprocessing is an essential part of every data analytics project. Our frameworks preprocessing scheme includes these sections:

1. Data Conversion: Datasets are gathered from different sources with different standards and outputs. These different channels of data need to be converted into one pre-defined structure and format before being used. Data can be generally categorized into three groups based on its form [20]: (a) Structured Data, data typically stored in traditional databases in the form of rows, columns and dictionaries; (b) Unstructured Data, in the forms of videos, images, and audio; (c) Semi-structured Data, such as data in the form of XML, JSON and HTML files. Depending on the data sources, all of these data types could be available, requiring conversion and integration. Although power consumption data is typically gathered in structured forms, integrating data from other sources with different structures, to check the correlation between datasets, could require adjusting semi-structured and unstructured data to some sort of structure, which is a complex task.
2. Data Aggregation: Data points can be gathered at different rates. One of the deciding factors in the quality of a ML model, as mentioned before, is the size of the dataset. More data leads to training the model on a wider array of values, which can improve its accuracy and reduce the chances of overfitting. However, higher sampling rates can threaten the privacy of consumers. This data can be misused to infer private information regarding the customers, such as their work schedule, home appliances, and whether the customers are staying at their home or not [21,22]. Our framework suggests resampling in a way that we still have enough data for training the model, while preserving the privacy of consumers.
3. Removing Noises: It is common for datasets to have noisy data. Noisy data are often corrupted or distorted data points, containing values with huge differences from the expected. All noisy data are outliers, but the opposite is not correct. Our frameworks objective is to discover anomalies or meaningful outliers from the power consumption data, but the existence of noise in the dataset impacts the

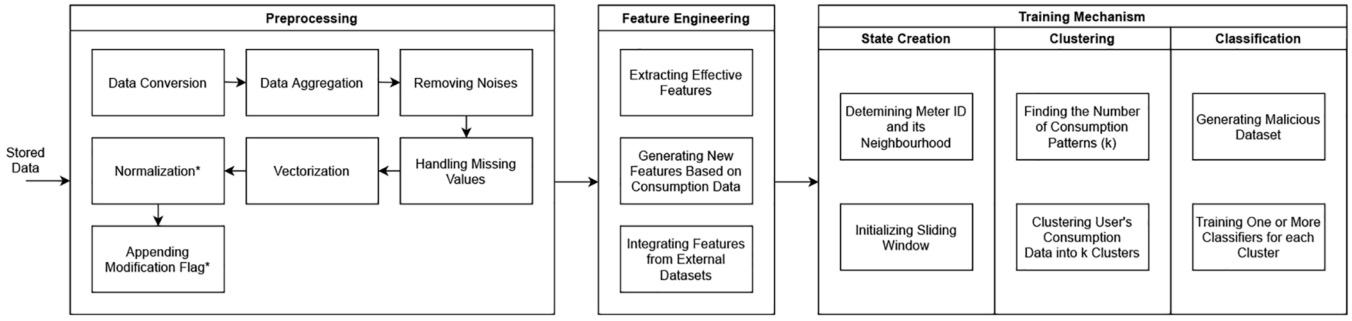

Fig. 1. Framework's training phase

model and results in lower accuracy. A good example of noise in power consumption data is incorrectly entered values into the system by operators, or negative values reported by the meter. Also, some other types of meter malfunctioning, such as buffer overflow, result in entirely different values that are considered as noise.[1] Hence, our framework removes the noisy data points from the dataset before training the models.

4. Handling Missing Values: Sometimes, due to human error, unexpected situations, or noisy data removal, we facemissing values in the dataset. Handling these missing values in the dataset is an essential step in training the system. The majority of ML models cannot function with missing values existing in the dataset. Missing values can be separated into three categories: Missing completely at random (MCAR), Missing at random (MAR), Missing not at random (MNAR). Based on what type of missing values we are dealing with, our goal, and the dataset, we choose one or more approaches to handle the missing values. Some of the missing value handling approaches are as follow [23]:
   - Case Deletion: Removing missing data points from the dataset is the most straightforward approach used in handling missing values. This approach might work when training data are abundant, and the missing values are assumed to be missing completely at random (MCAR). However, if these conditions are not satisfied, case deletion introduces biases into the data.
   - Mean Substitution: In this approach, the missing value is substituted with the mean of the variable. This approach is used with the idea that the mean is a logical estimation of a randomly selected observation in a normal distribution. Nevertheless, if the observation is not completely random or values are not normally distributed, this method may lead to inconsistency.
   - Regression Imputation: This method uses other existing observations in the dataset to predict an estimated value for the missing data. A regression model is trained on the available data to predict the observed values. Missing values from the dataset are then filled with values predicted by the model.
   - Last/Next observation carried forward/backward: In this method, the missing value is replaced with the previously observed/next available value before/after it. This method works best with datasets gathered over time with a sequential nature, such as time-series, and assumes that values observed after one another are strongly related to each other. One of the advantages of this method is its ease of understanding.

     In the proposed framework, we use these missing values handling methods based on our need to fill in the blanks in our dataset.
5. Vectorization: This step converts each users consumption data into vectors of size V. In other words, each one-dimensional channel of the time series is transformed into a two-dimensional space. Most methods use daily vectors, containing either 24 one-hour elements, or 48 half-hour elements.
6. Normalization: Normalization is a data preparation technique used to modify the values of each data column, changing them to have a common scale. There are several normalization methods, each used for different situations. Depending on the model used to process the data, normalization can significantly impact its accuracy, while some are unaffected. For example, tree-based models do not require normalization since splitting is on single features, and the scale of the data does not affect it. On the other hand, distance-based methods such as K-nearest neighbors (KNN), require normalization. In the case of KNN, since the distance of data points from each other is calculated and compared to determine the nearest neighbors, having data columns with different scales affects the calculated distance and the results of the algorithm might be inaccurate. If the ML models used by the frameworks classifiers are affected by the scale of data columns, we first normalize them.
7. Appending Modification Flag: We will add a modification flag to each row of our dataset, indicating whether the value is authentic or something calculated or modified by us. Using this flag, if needed, we can change the impact of modified values in our training process.

3.1.2. *Feature engineering*

There are some techniques that can be used to improve the quality of our dataset so we can achieve a more accurate trained model. In this framework, we use feature engineering in one of the three ways below:

- Extracting effective features: Sometimes, instead of using all the features of the training dataset, we can use a good subset of it to increase our trained model's accuracy and reduce time and resource consumption. An example of feature extraction was performed in [24], where a subset of features was selected from the RECS2015 dataset of power consumption to identify more relevant features to the yearly consumed power.
- Generating new features based on consumption data: As it was mentioned before, there are features, such as statistical features like mean, minimum, maximum, and standard deviation, that can be calculated from the vectors and then get appended to them. These calculated features are advantageous when used as a description of each user's behavior and can be helpful in clustering them together.
- Integrating features from external datasets: Sometimes, other external sources can help better explain your dataset. Features from datasets, gathered from different sources, could be integrated into one another to check their correlation together. This is especially true for time-series data. By integrating data from other sources into a time-series, we essentially add more channels to the time-series. Time-series can be univariate, as in only one channel data, or

[1] There are other types of noises, like altered values because of cosmic-ray-induced errors, which are much less common in this domain.

multivariate. Adding other features, which correspond with the timestamps in our time-series, to the dataset, is similar to having different sensors and data channels when we are gathering the data. For example, in the case of power consumption data that we are working with, other features such as weather conditions, temperature, and special occasions can increase the understandability of the changes in power consumption. However, it needs to be considered that these added features increase the time consumption of our models training phase. They also need to be individually pre-processed, since the range of their values can affect our models results.

3.1.3. Training mechanism

After completing all the previous steps mentioned above and preparing the dataset, we get to the training mechanism of the framework for detecting anomalies. This training mechanism consists of three different parts: State Creation, Clustering, Classification.

- State Creation: In the state creation phase, a representation of each user containing necessary information of that, which gets updated constantly, is created. It includes the users zone information, which are meter ID and neighborhood, a detection module, which includes the users consumption patterns in the form of clusters, and a sliding window explained in the testing phase. The sliding window is the mechanism that informs us of fraudulent behavior in users data. In the first step of the training mechanism, each users state is initialized so that it can be used and updated in the following steps of the training phase. The details of the state module are explained in the testing phase.
- Clustering: Clustering is the task of dividing the population of a group so that members of a group are similar to each other, while the difference between members of different groups is maximized. There are several general types of clustering, such as hierarchical clustering, density-based clustering, and distribution-based clustering, each of which is suited for specific purposes and datasets. The proposed framework uses clustering to find different consumption patterns in each users consumption data. Clustering algorithms have their own hyperparameters, such as the number of clusters in k-means or epsilon and the minimum number of points to form a dense region in DBSCAN, which need to be set beforehand. One way of determining the optimum value for these hyperparameters is to perform clustering algorithms several times with different values on each users data. In the end, the results of clustering algorithms are compared together based on some metrics to find out the optimum values for the hyperparameters.
- Classification: Our final phase in the training mechanism is classification. In this phase, we first inject malicious data into our users consumption data, and then we use classification algorithms to detect the injected anomalies. It is essential to know that the data used in the training process must be fully honest, and any fraudulent data in it can lower the accuracy of the model.
  - Malicious Data Generation: We used six malicious attack patterns, defined in [10], and one other pattern that we created based on the experts' opinion in Iran's electricity utility company. As mentioned before, we divided each customer's data into vectors of predetermined length. For each of these segments, we created a malicious variant with each attack pattern. These attack patterns are as follow:
    - $h_1(x_t) : \alpha x_t, \alpha = random(0.1, 0.8)$
    - $h_2(x_t) : \beta_t x_t,$

      $$\beta_t = \begin{cases} 0 & t_{start} < t < t_{end} \\ 1 & \text{otherwise} \end{cases},$$

      $t_{start} = random(0, 23 - t_{off}),$

      $duration = random(t_{off}, 24),$

      $t_{end} = t_{start} + duration,$

      $t_{off} \geq 4$
    - $h_3(x_t) : \gamma_t x_t, \gamma_t = random(0.1, 0.8)$
    - $h_4(x_t) : \gamma_t mean(x), \gamma_t = random(0.1, 0.8)$
    - $h_5(x_t) : mean(x)$
    - $h_6(x_t) : x_{24-t}$
    - $h_7(x_t) : \theta_t x_t,$

      $$\theta_t = \begin{cases} random(0.1, 0.8) & t_{\text{start peak}} < t < t_{\text{end peak}} \\ 1 & \text{otherwise} \end{cases},$$

      $$t_{\text{start peak}} = \begin{cases} 13{:}00 & \text{summer} \\ 21{:}00 & \text{other seasons} \end{cases},$$

      $$\text{duration} = \begin{cases} 6 \text{ hours} & \text{summer} \\ 3 \text{ hours} & \text{other seasons} \end{cases},$$

      $t_{\text{end peak}} = t_{\text{start peak}} + \text{duration}$

      h_1 multiplies all of the data points by a random value. h_2 is equivalent to the smart meter not sending its measurements or sending zeros for a random duration. h_3 is similar to h_1, but in this case, each of the data points is multiplied by a random number. h_4 and h_5 both send out the mean of the affected data points. h_5 sends out the exact value of the mean, while the output of h_4 is a fraction of it. h_5 and h_6 are both attacks against the load control mechanism. In some countries, such as Canada and the USA, the price of electricity usage in peak and off-peak hours are different. These load control mechanisms separate the consumption in these hours, so they can be priced differently. h_5 and h_6 attacks aim to keep the total energy consumption data equal to the actual amount while changing the calculated usage in peak hours to lower the cost. h_5 uses the mean of the data points, and h_6 reverses the power consumption of the targeted period to achieve this goal. h_7 is an attack we introduce in this paper. It is specifically designed based on the data of Irans Electrical Utility company. In this attack, similar to h_3, each power consumption value in a time frame is multiplied by a random multiplier. The difference is that the affected duration is chosen based on the start and end of the peak hours in the given season. For example, in Iran, the peak hour of power consumption in summer starts from 13:00 with a duration of 6 hours, while in other seasons, it starts at 21:00 with a duration of 3 hours. These hours change based on the country in question.
  - Training Classifiers: As mentioned above, we create malicious variants of each consumption vector. The combination of these vectors and the original vectors are used as the training data for our classifiers. The framework trains one or more different classification models for each cluster in a users data, representing the different consumption patterns in that users data. The results of these models are then combined to determine which class the consumption vector belongs to.

Now that the framework has been trained on a portion of the users power consumption data and its models learned to classify anomalous and non-anomalous behaviors, the users data can be given to the framework through an online data stream.

3.2. Testing phase

The testing phase of the proposed framework consists of two main segments. The first one includes computing the NTL of the users neighborhood and classifying its consumption vectors. The second segment of the testing phase includes the decision making unit, which uses the information from the first segment to decide if the user has committed electricity theft. The details of this phase are shown in Fig. 2.

3.2.1. Updating user's state

Before checking the users consumption vectors for fraudulent behavior, the framework first calculates the technical and non-technical

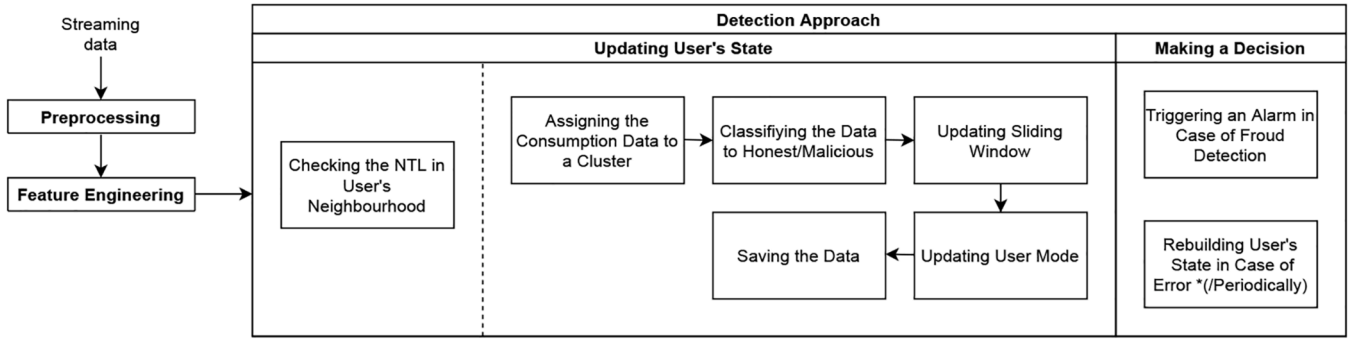

Fig. 2. Framework's testing phase

loss of the users neighborhood. For each neighborhood, one or more transformer meters report the amount of supplied electricity to that neighborhood. The measured value (E_{TM}) is then compared with the total amount of consumed energy reported by the smart meters ($\sum_i E_{SM_i}$) of the corresponding transformer. Neighborhood NTL detection formula is represented by equation 1, where E_{TL} is the estimated technical loss of the area and ε is the error in TLs calculation.

$$E_{TM}(t) > \sum_i E_{SM_i}(t) + E_{TL}(t) + \varepsilon \tag{1}$$

With the NTL value estimated for each neighborhood, high risk neighborhoods can be discovered. As new consumption vectors are given to the framework, The values for the NTL and TL are updated to include them.

The framework analyzes the users as data streams in an online manner. Similar to the training phase, the new data must first be pre-processed to meet the required format for the framework. The pre-processing stage includes data conversion, data resampling, and aggregation, removing noisy data and handling missing values, normalization, and finally, appending the modification flag. After that, the framework performs any sort of feature engineering operation done in the training phase to increase the quality of the testing portion of the dataset. Now, the data is ready to be used by the frameworks detection mechanism.

As mentioned in the training phase, consumption vectors are clustered to determine the different patterns in the users data. In the testing phase, each new consumption vector, coming through the data stream, is assigned to one of the previously created clusters. In the training phase, one or more models were trained for each of the clusters, based on the honest and malicious data associated with that cluster, to classify them. These models will now classify the newly arrived vector into one of the honest or malicious classes. Each user had a state representing its information, including a sliding window used for discovering electricity theft. Fig. 3 shows an abstract view of the state module. Every new classified consumption vector is inserted into this window. Our goal is to discover NTL in consumption data, but having one suspicious data point is not enough to determine the existence of energy theft.

Using the sliding windows, we check the consumption vectors to see if the anomalous behavior persists over a period of time. The length of this period is the same as the sliding windows size (M). To trigger an alarm of energy theft, a certain number of data points inside the sliding windows should be anomalous (N). After passing this threshold, the users mode in the state module changes to suspicious. The sliding window size and the minimum number of anomalies inside the window allow for different sensitivity levels. With an increase in M, a larger duration is considered for determining suspicious behavior. N indirectly shows the minimum percentage of the sliding window points that need to be anomalous, so the user is considered suspicious. Typically, with a higher percentage, fewer users are assumed to be fraudulent, but the probability of electricity theft committed by them is higher. This

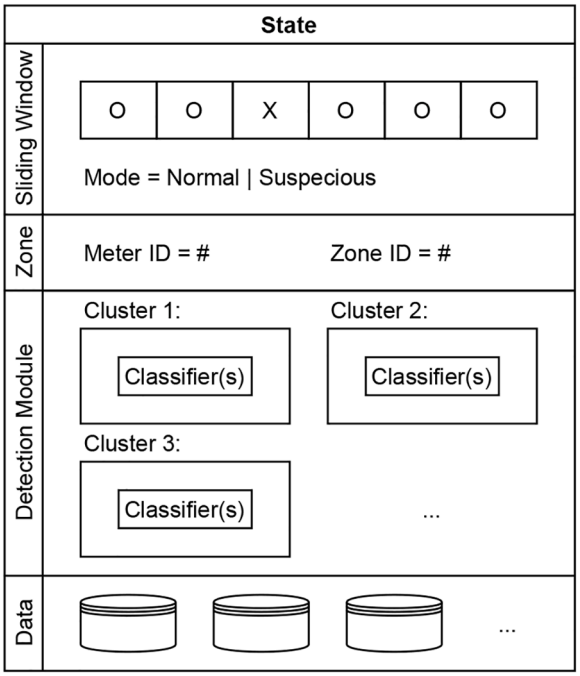

Fig. 3. User's state

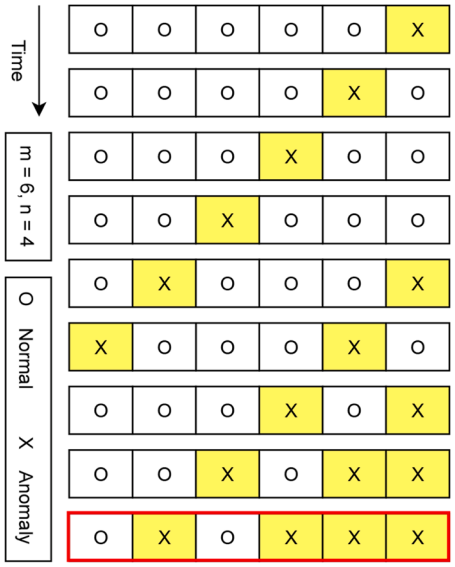

Fig. 4. An example of sliding window, with m=6 and n=4

amounts to a trade-off between the detection rate of anomalous users and false alarms. As mentioned before, suspicious users need on-site inspections, which is expensive. Based on the policies of the electrical distribution company and the situation in question, these parameters can adjust the sensitivity of the detection process. With each new data vector, the first vector in the sliding window is pushed out, and the other vectors get shifted to the left. An example of that can be seen in Fig. 4. This process is performed on the data to update the users state. In the meantime, the framework aggregates the results of the NTL estimator and the classifiers, after each new vector, to determine whether the user has committed electricity fraud or not.

3.2.2. Decision making

Now that the framework has identified suspicious behavior and calculated the NTL for the users neighborhood, we can decide if the user has committed electricity theft. There are four possible scenarios regarding electricity theft when considering the classifiers results and the neighborhood NTL detection formula.

- The first possible scenario is when neither the neighborhood NTL detection formula nor the frameworks classifier has detected anything suspicious. In this case, the observed vectors are deemed as honest data and are added to the honest power consumption data vectors of that particular user. In the future, if there is a need for retraining in the models, this data point and its malicious variants can be used.
- The second possible scenario happens when both the neighborhood NTL detection formula and the user's classifier detect fraudulent behavior. The system checks that the user's mode has changed to suspicious based on the sliding window to see if the anomalous behavior persisted over some time. If so, since both of these units suggested that the users behavior is an indicator of electricity theft, this user is reported for on-site inspection. It needs to be considered that more than one fraudulent user can be present in a neighborhood, and discovering one does not exempt others from suspicion. The fraudulent consumption vectors are added to the malicious dataset of the user for possible retraining of the models in the future.
- The third possible scenario is when the classifier detects anomalous behavior, but the neighborhood NTL detection formula shows no sign of electricity theft. There are three possible explanations for this scenario. The classifier could have misclassified the consumption vectors, or there might have been an error in the NTL calculation. Both of these cases are not supposed to happen frequently. The other explanation for this situation is an alteration in the customers consumption pattern. This alteration can be the result of using new appliances, career change, or change in residents of the household. To make sure if the consumption pattern has changed, and one of the two first explanations is not the cause of the difference in the classifier's results and the neighborhood NTL detection formula, this change should last for a period of time. If we discover that the pattern has changed, the newly observed data is kept in the database until the new database becomes large enough for the training process. With the classifiers retrained now, the framework can resume its detection process on new patterns.
- The last possible scenario in detecting electricity theft is when the NTL is detected by the neighborhood NTL detection formula, while the models classified the users consumption behavior as non-anomalous. If the condition persists, either the neighborhood NTL detection formula is at fault, or an attack is probably in progress and the classifiers cannot identify it. This might be due to contaminated data in the honest dataset used for training the models. As mentioned before, the framework presumes that all the data used in the training process is completely honest. The non-malicious part of the training dataset is checked for data contamination attacks, which slowly changed the data and polluted the dataset to cause misclassification in the models. A long-term consumption descending slope in the observed consumption data of the user can be a sign of this contamination. Another situation that can lead to this scenario is the direct attachment of electrical devices to the feeder before the meter, which is undetectable by both this framework and the previous related works.

4. Evaluation

In this section, we check the feasibility of the proposed framework and test its performance in a practical situation. We computed accuracy, the area under the Receiver Operating Characteristics (ROC), and Precision-Recall (PR) curves for the model. ROC shows the True Positive Rate (TRP) as a function of False Positive Rate (FPR) at several thresholds, and PR is an indicator of the tradeoff between precision and recall. Area Under the Curve (AUC) is a proper measure to comprise ROC and PR. Owing to the scale-invariant and threshold-invariant nature of AUC, it is a good measure for model comparison. The larger the area, the better the model is, with an area of 1 belonging to the perfect model. In the following tables, AUROC shows the two-dimensional area under the curve, and AUPR indicates that of the PR curve. Also, ACC is the overall accuracy.

In our previous work [25], we comprised decision-tree, random forest, and gradient boosting to detect anomalies in power consumption data. The models were once executed with and without clustering to see if clustering positively impacted the results. The results showed that clustering had a significant positive impact on all models, and gradient boosting achieved a better performance than the random forest and decision tree algorithms. Owing to the impact of clustering in our previous work, we decided to make clustering a permanent part of the framework.

4.1. Dataset

To test the framework in a real scenario, we chose the UMass Smart* dataset 2017 [26]. This dataset contains data for 114 single-family apartments for the period 2014-2016. The data sampling rate for the first year was every 15 minutes, and for the second year, every 1 minute. We assumed all the data in the dataset to be honest. The dataset was split into a train and a test portion. We used 75 percent of the data (consumption values until 2016-07-01) for training the framework, and the rest of the dataset was given to the framework as a data stream.

4.2. Data processing engine

Since the proposed framework is supposed to be practical and usable by utility companies, it also needed to be computationally efficient and capable of online user processing. We tested several available tools for developing the framework, concluding Apache Spark to be the best data processing engine fit for our requirements. Apache Spark is a unified analytics engine for large-scale data processing. It is flexible as a development tool, since it supports multiple languages, such as Python and Java. Spark gives us access to distributed implementations of many ML algorithms, using MLlib, and also supports deep learning via Deep Learning Pipelines by applying Keras models to incoming data. It also provides us with Spark Streaming, a component allowing Spark to process real-time data. With all of these features combined, Spark was an excellent choice for developing the proposed framework.

4.3. Preprocessing

The dataset was then preprocessed to become usable by the framework. For privacy reasons, the data was first resampled to 1 hour time frames in the preprocessing step. The missing values in the dataset were filled, and the resulting dataset was vectorized into vectors of size 24, representing the 24 hours of the day. Since we are using gradient boosting as our classification model, which is a tree-based algorithm,

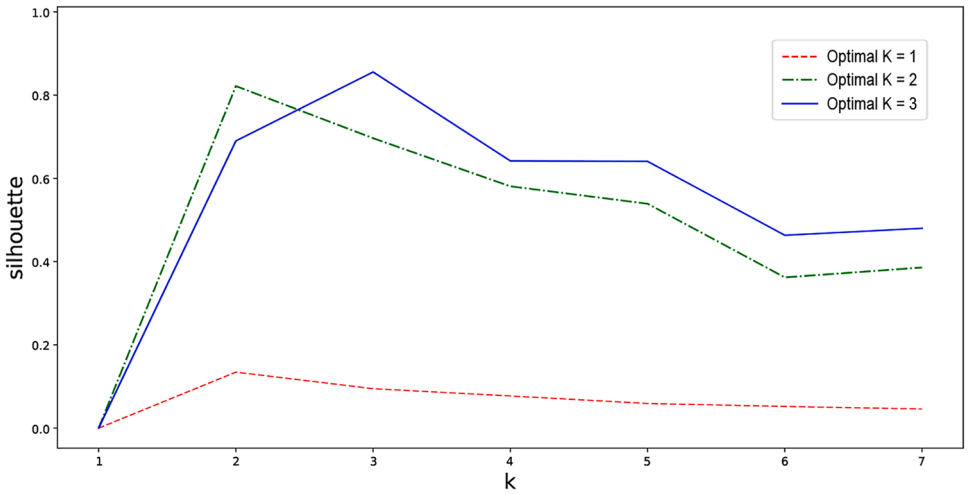

Fig. 5. Different silhouette scores for three scenarios

our dataset does not need to be normalized.

4.4. Feature engineering

The framework calculated the mean, maximum, minimum, and standard deviation for each day and appended them to the consumption vector. The newly generated vectors contain 28 elements: the consumption of the 24 hours of the day and the 4 calculated features.

4.5. Training mechanism

A state module was created and initialized for each user. Then in the clustering step, the framework used the K-Means algorithm. K-Means is a fast and reliable clustering algorithm used in previous works in this research area, such as [27]. It gives you the freedom to choose the number of clusters, in this case, the number of consumption patterns each user has and is capable of online execution. K-Means is a clustering algorithm designed to divide the dataset into K partitions based on the elements distance from the K partitions centroids. First, the value of best K for each user should be calculated; therefore, the framework ran the K-Means algorithms several times with different values for K on each users consumption vectors and measured the silhouette score of the clustering. The results showed that the majority of users gained their best silhouette score with K=1, K=2, and in some rare cases, K=3. This indicates that most users had either one or two distinct consumption patterns. After the clusters for each user have been defined, they are added to the users state. An example of different silhouette scores for three consumers can be seen in figure 5.

As mentioned above, gradient boosting was chosen as the classifier for the clusters in the classification step. Tree-based methods such as gradient boosting are well-established algorithms for classification tasks used in many different research areas, including electricity theft detection, like paper [28], for many years and have proven themselves to be comparable to more modern state-of-the-art classifiers, such as neural networks. In our previous work [25], we tested three classifiers, and gradient boosting achieved the best results. That is why we decided to implement a distributed version of gradient boosting using Apache Spark for our classifier.

4.6. Testing phase

In the testing phase, the remaining vectors for each user were given to the framework one by one. New data points are gathered and kept until they can fill a consumption vector. With the arrival of each newly filled vector, the framework updates the state of the user in question. Since we do not have access to the transformer data of users, as this data is not publicly available, we cannot use the neighborhood NTL detection formula to calculate the NTL of neighborhoods. The availability of this data could further increase the accuracy and performance of this detection system in real-world situations. The new vector is first put through the feature-engineering step, then it is assigned to one of the clusters, classified, and inserted into the sliding window. Based on one of the four possible scenarios mentioned in the testing phase section of our proposed approach, the appropriate steps are taken to identify electricity theft.

4.7. Comparison

The results of running the framework for this scenario are shown in the table 1. Gradient Boosting achieved an accuracy of 88 with an AUROC of 0.94 and AUPR of 0.95, equal to our previous work [25], performed in a non-distributed offline mode. We then compared these scores with the results of other frameworks and papers done in this area. It needs to be considered that given different datasets, the results could vary. The datasets used for models' training in the frameworks could have different characteristics, such as the percentage of missing values and sampling rate, for example, whether this period is long or includes holidays. These different characteristics affect the result of the frameworks. We also compared other aspects of the framework, including its reliability, robustness, privacy preservation, distributed execution, online processing, and sensitivity control, with other works.

Distributed Execution refers to the framework's ability to run the required algorithm in a distributed environment. As mentioned in the introduction section, distributed data processing can have many benefits, like scalability, cost-effectiveness, and more trustworthy big data handling. One of the main advantages of the proposed framework is its distributed implementation which allows for distributed Execution of the ML models on the data. Distributed Execution refers to the framework's ability to run on a cluster of connected systems. This distributed Execution also results in some secondary benefits. Reliable Execution is one of these advantages, which refers to the framework's ability to continue its work even when one or more components of the system have failed. Online Processing is the frameworks ability to process data streams as data arrives. Required Sampling Rate and Privacy Preservation are related to each other. As mentioned in the proposed approach section of this paper, higher sampling rates for data can cause a breach of the user's privacy. Sensitivity Control refers to the operator's control over the sensitivity of the proposed detection techniques offered by the framework. The sensitivity of the detection mechanism can create a tradeoff between the number of detected suspicious users and the false

Table 1
Comparison among electrisity theft frameworks

Method / Approach	This work	Work [10]	Work [3]	Work [29]
ACC	88+ [a]	89 (with 24 element vectors)	N/A	varies for different attacks [b]
AUROC	0.94	N/A	N/A	N/A
AUPR	0.95	N/A	N/A	N/A
Dataset	UMass Smart*	Irish dataset (CER)	N/A	Irish dataset (CER)
Required Sampling Rate	Low	Low	High	High
Privacy Preservation	Medium	Medium	Weak	Weak
Distributed Execution	Yes	No	No	No
Reliable Execution	Yes	No	No	No
Online Processing	Yes	No	No	No
Sensitivity Control	High	Medium	No	N/A
Robustness	High	High	Low	High

[a] based on sliding window's sensitivity level
[b] from 73.8 to 94.0 on particular attacks

alarm of the detection algorithm. The last compared metric is robustness. A detection technique is robust when the detection accuracy is not significantly affected from the baseline under various conditions, such as outliers in the dataset.

Table 1 compares this work with other frameworks in this scope. This comparison showed our framework had achieved either comparable or better accuracy, AUROC, and AUPR to other works. This acquired accuracy is subject to change based on the sensitivity of the frameworks detection mechanism, provided by the sliding window and used algorithms. Also, due to its distributed implementation, this framework offers a reliable execution on clusters of machines. However, we faced some limitations in choosing classifiers to test the framework since not every machine learning algorithm can have a distributed implementation. Besides the distributed advantages, this framework allows for the online processing of data streams, an ability essential in processing power consumption data. This approach requires a low data sampling rate, resulting in medium privacy preservation for users, and offers high robustness and sensitivity control in its detection mechanism.

Paper [10] presented a multi-class SVM-based electricity theft detection approach which got an accuracy score of 89 in 24-element vectors, tested on Irelands Commission for Energy Regulation (Irish) dataset injected with six different malicious attack patterns. Owing to the use of SVM in their detection mechanism, it is robust and offers a medium degree of control over the sensitivity of the detection system. Also, since the required sampling rate of consumption data is low, it has a medium degree of privacy preservation. However, since their proposed approach does not have a distributed implementation, it lacks a distributed and reliable execution. Work [3] used power measurement-based anomalous consumption detectors through non-intrusive load monitoring (NILM) in AMIDS, their proposed framework, which required a sampling rate of 0.5 samples/min to function properly, resulting in weak user privacy preservation. Also, this work did not focus on robustness, reliable and distributed execution, and sensitivity control; as such, it is somewhat lacking in these aspects. Lastly, Paper [29] developed an electricity theft detection framework based on Kullback- Leibler Divergence, tested on the Irish dataset, injected with specific attack patterns. This framework gained an accuracy score between 73.8 and 94, based on the injected attacks and their classes. It offered robustness in its detection but lacked distributed and reliable execution and required a high sampling rate in data, resulting in weak privacy preservation.

5. Conclusion

In this paper, we introduced a new framework for electricity theft detection in smart grids. This framework preprocesses the data, performs feature engineering, discovers consumption patterns, classifies the consumptions, and then tests the rest of the user's data to look for fraudulent behavior. This framework combines data-oriented and network-oriented approaches to detect anomalies, further increasing its accuracy. Because of this hybrid method and our novelty in using a sliding window, which allows for more control over the sensitivity of the detection process, this framework can reduce the rate of costly false alarms. We also added a new attack pattern to the previously discovered attacks in electricity theft, based on the experts' opinion of the utility company. Our proposed framework was also designed with handling real-time large-scale smart grid data in mind, which was achieved by its distributed deployment. Lastly, we evaluated the feasibility of this framework by testing it on real users' power consumption data with gradient boosting as its classifier. We compared the results of this experiment with other previous works in this field, which showed our framework to be comparable or superior to other works while having the disadvantage of limited available machine learning models in distributed form. On the other hand, the framework benefits from all the advantages of distributed deployment and the online processing of new data. The clustering and classification methods used in the evaluation of the framework have room for improvement by using techniques such as hyperparameter tuning. The mentioned clustering and classification methods were used to show the framework's ability to produce results similar or superior to the previously achieved metrics while having distributed computing capabilities.

CRediT authorship contribution statement

Soroush Omidvar Tehrani: Conceptualization, Funding acquisition, Formal analysis, Writing – original draft, Visualization. **Afshin Shahrestani:** Conceptualization, Funding acquisition, Formal analysis, Writing – original draft, Visualization. **Mohammad Hossein Yaghmaee:** Conceptualization, Writing – original draft, Visualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to in uence the work reported in this paper.

References

[1] F. de Souza Savian, J.C.M. Siluk, T.B. Garlet, F.M. do Nascimento, J.R. Pinheiro, Z. Vale, Non-technical losses: A systematic contemporary article review, Renewable and Sustainable Energy Reviews 147 (2021) 111205, https://doi.org/10.1016/j.rser.2021.111205.
[2] S.S.S.R. Depuru, L. Wang, V. Devabhaktuni, Electricity theft: Overview, issues, prevention and a smart meter based approach to control theft, Energy Policy 39 (2) (2011) 1007–1015.
[3] S. McLaughlin, B. Holbert, A. Fawaz, R. Berthier, S. Zonouz, A multi-sensor energy theft detection framework for advanced metering infrastructures, IEEE Journal on Selected Areas in Communications 31 (7) (2013) 1319–1330, https://doi.org/10.1109/JSAC.2013.130714.
[4] G. Giaconi, D. Gunduz, H.V. Poor, Privacy-aware smart metering: Progress and challenges, IEEE Signal Processing Magazine 35 (6) (2018) 59–78.
[5] N.F. Avila, G. Figueroa, C.-C. Chu, Ntl detection in electric distribution systems using the maximal overlap discrete wavelet-packet transform and random

undersampling boosting, IEEE Transactions on Power Systems 33 (6) (2018) 7171–7180.
[6] G.M. Messinis, N.D. Hatziargyriou, Review of non-technical loss detection methods, Electric Power Systems Research 158 (2018) 250–266, https://doi.org/10.1016/j.epsr.2018.01.005.
[7] M. Tariq, H.V. Poor, Electricity theft detection and localization in grid-tied microgrids, IEEE Transactions on Smart Grid 9 (3) (2018) 1920–1929, https://doi.org/10.1109/TSG.2016.2602660.
[8] E.A. Aranha Neto, J. Coelho, Probabilistic methodology for technical and non-technical losses estimation in distribution system, Electric Power Systems Research 97 (2013) 93–99, https://doi.org/10.1016/j.epsr.2012.12.008.
[9] Z. Xiao, Y. Xiao, D.H.-C. Du, Exploring malicious meter inspection in neighborhood area smart grids, IEEE Transactions on Smart Grid 4 (1) (2013) 214–226, https://doi.org/10.1109/TSG.2012.2229397.
[10] P. Jokar, N. Arianpoo, V.C. Leung, Electricity theft detection in ami using customers consumption patterns, IEEE Transactions on Smart Grid 7 (1) (2015) 216–226.
[11] C. Cody, V. Ford, A. Siraj, Decision tree learning for fraud detection in consumer energy consumption. 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), IEEE, 2015, pp. 1175–1179.
[12] R. Punmiya, S. Choe, Energy theft detection using gradient boosting theft detector with feature engineering-based preprocessing, IEEE Transactions on Smart Grid 10 (2) (2019) 2326–2329.
[13] A. Arif, N. Javaid, A. Aldegheishem, N. Alrajeh, Big data analytics for identifying electricity theft using machine learning approaches in microgrids for smart communities, Concurrency and Computation: Practice and Experience (2021) e6316.
[14] M. Di Martino, F. Decia, J. Molinelli, A. Fernández, Improving electric fraud detection using class imbalance strategies. ICPRAM (2), 2012, pp. 135–141.
[15] S. Li, Y. Han, X. Yao, S. Yingchen, J. Wang, Q. Zhao, Electricity theft detection in power grids with deep learning and random forests, Journal of Electrical and Computer Engineering 2019 (2019).
[16] R. Razavi, A. Gharipour, M. Fleury, I.J. Akpan, A practical feature-engineering framework for electricity theft detection in smart grids, Applied Energy 238 (2019) 481–494, https://doi.org/10.1016/j.apenergy.2019.01.076.
[17] V. Badrinath Krishna, R.K. Iyer, W.H. Sanders, Arima-based modeling and validation of consumption readings in power grids, in: E. Rome, M. Theocharidou, S. Wolthusen (Eds.), Critical Information Infrastructures Security, Springer International Publishing, Cham, 2016, pp. 199–210.
[18] D.S. Naidu, S. Ozcelik, K.L. Moore, Chapter 3 - gas metal arc welding: Sensing, in: D.S. Naidu, S. Ozcelik, K.L. Moore (Eds.), Modeling, Sensing and Control of Gas Metal Arc Welding, Elsevier Science Ltd, Oxford, 2003, pp. 95–145, https://doi.org/10.1016/B978-008044066-8/50005-7.
[19] J.V. Spirić, M.B. Dočić, S.S. Stanković, Fraud detection in registered electricity time series, International Journal of Electrical Power and Energy Systems 71 (2015) 42–50, https://doi.org/10.1016/j.ijepes.2015.02.037.
[20] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I.A.T. Hashem, A. Siddiqa, I. Yaqoob, Big iot data analytics: Architecture, opportunities, and open research challenges, IEEE Access 5 (2017) 5247–5261, https://doi.org/10.1109/ACCESS.2017.2689040.
[21] M. Nabil, M. Ismail, M.M.E.A. Mahmoud, W. Alasmary, E. Serpedin, Ppetd: Privacy-preserving electricity theft detection scheme with load monitoring and billing for ami networks, IEEE Access 7 (2019) 96334–96348, https://doi.org/10.1109/ACCESS.2019.2925322.
[22] M. Shateri, F. Messina, P. Piantanida, F. Labeau, Deep directed information-based learning for privacy-preserving smart meter data release. 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), 2019, pp. 1–7, https://doi.org/10.1109/SmartGridComm.2019.8909813.
[23] H. Kang, The prevention and handling of the missing data, Korean journal of anesthesiology 64 (5) (2013) 402.
[24] H.M. Sani, S.O. Tehrani, B. Behkamal, H. Amintoosi, Extracting effective features for descriptive analysis of household energy consumption using smart home data. International Congress on High-Performance Computing and Big Data Analysis, Springer, 2019, pp. 240–252.
[25] S.O. Tehrani, M.H.Y. Moghaddam, M. Asadi, Decision tree based electricity theft detection in smart grid. 2020 4th International Conference on Smart City, Internet of Things and Applications (SCIOT), IEEE, 2020, pp. 46–51.
[26] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, J. Albrecht, et al., Smart*: An open data set and tools for enabling research in sustainable homes, SustKDD, August 111 (112) (2012) 108.
[27] K. Zheng, Y. Wang, Q. Chen, Y. Li, Electricity theft detecting based on density-clustering method. 2017 IEEE Innovative Smart Grid Technologies-Asia (ISGT-Asia), IEEE, 2017, pp. 1–6.
[28] Z. Yan, H. Wen, Electricity theft detection base on extreme gradient boosting in ami, IEEE Transactions on Instrumentation and Measurement 70 (2021) 1–9.
[29] V.B. Krishna, K. Lee, G.A. Weaver, R.K. Iyer, W.H. Sanders, F-deta: A framework for detecting electricity theft attacks in smart grids. 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2016, pp. 407–418, https://doi.org/10.1109/DSN.2016.44.