Energy & Buildings 185 (2019) 326–344

Contents lists available at ScienceDirect

Energy & Buildings

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

Early detection of faults in HVAC systems using an XGBoost model with a dynamic threshold

Debaditya Chakraborty*, Hazem Elzarka

Department of Civil and Architectural Engineering and Construction Management, University of Cincinnati, OH 45221-0071, USA

ARTICLE INFO

Article history:
Received 15 August 2018
Revised 16 October 2018
Accepted 24 December 2018
Available online 5 January 2019

Keywords:
Energy modelling
Fault detection
XGBoost
Dynamic threshold

ABSTRACT

Growing demand for energy efficient buildings requires robust models to ensure efficient performance over the evolving life cycle of the building. Energy management systems can prevent energy wastage in buildings without sacrificing occupant's comfort. However, their full capabilities have not been completely realized, partly due to their inability to quickly detect faults in HVAC systems. An accurate model and an appropriate threshold are the key factors in fault detection. The traditional method of setting a fixed threshold often leads to missed opportunities to detect faults, delayed detection of faults or false alarms. To improve the effectiveness of fault detection algorithms, we have first developed a data-driven model using extreme gradient boosting (XGBoost). We have then applied the proposed dynamic threshold method to determine occurrences of faults in real time. This method adjusts the threshold value dynamically according to the real-time moving average and moving standard deviation of the predictions. The results demonstrate the usefulness of our proposed method to detect faults early in the course. An average increase of 8.82% and 117.65%, in the F1 score, is achieved with the proposed method in comparison to the traditional fixed threshold method and an existing dynamic residual method.

1. Introduction

Building energy management and control systems (EMCS) collects and stores energy consumption data, which is used to maintain indoor environmental conditions and ensure efficient operation of the HVAC systems. Fault detection algorithms also use EMCS data to determine HVAC operation that deviates from expected behavior. Research on automated fault detection of HVAC systems has been active over several decades. Methods of detecting fault can be divided into three categories: Data-driven model-based, qualitative model-based, and quantitative model-based methods. As per a recent review article by Kim and Katipamula [15], data-driven models are the most popular choice for automated fault detection due to numerous advantages. In a data-driven fault detection model the actual performance of the HVAC system is compared to the predicted values so that erroneous operation and poor performances can be detected. An accurate predictive model and an appropriate threshold are the key factors in data-driven model-based fault detection. Over the past decade, various data-driven models with fixed thresholds have been proposed by researchers to detect faults in HVAC systems. The key problem with the existing data-driven fault detection methods are related to modeling limitations such as inaccurate predictions [10,19]. We are proposing a tree-based ensemble learning algorithm in this study to overcome these limitations. Another pertinent problem is related to the traditional technique of setting a fixed threshold to isolate faults from normal data. Statistical techniques or experience or rules-of-thumb are used to set fixed thresholds. Detecting faults early in the course is necessary to reduce energy wastage, prevent further deterioration, and prolong equipment life. The traditional method of setting a fixed threshold often leads to missed opportunities to detect faults, delayed detection of faults or false alarms [13,15,23]. If the threshold is set too high faults may be missed or if it is too low then false alarms will occur. Also, a threshold that is high may lead to a delayed identification of faults, which may cause significant damage to the equipment due to its extended use under faulty conditions. Therefore, we also propose a novel dynamic threshold method in this study to overcome the problems associated with fixed thresholds. The proposed dynamic threshold automatically adjusts its value according to the real-time moving average and moving standard deviation of the predictions. This paper explains the development of the proposed fault detection model using the extreme gradient boosting (XGBoost) algorithm and the proposed dynamic threshold method. The feasibility of the proposed fault detection model is tested in Section 6. The proposed dynamic threshold method is benchmarked relative

* Corresponding author.
E-mail address: chakrada@mail.uc.edu (D. Chakraborty).

https://doi.org/10.1016/j.enbuild.2018.12.032
0378-7788/

to the traditional fixed threshold method and an existing dynamic residual method in Section 7.

2. Background

HVAC performance degrades due to operational misuse, sensor error, instrument error, system degradation and system faults, etc. It is essential to identify these anomalies early in the course so as to avoid energy wastage, maintain the desired indoor conditions, and extend equipment life. Over several decades, researchers have proposed different algorithms to automate the detection of faults in HVAC systems. This section reviews some prominent research papers on HVAC fault detection in a chronological way.

Hyvarinen and Karki [13] edited a source book that formally introduced the basic concepts and approaches of fault detection in HVAC systems. Different techniques for fault detection ranging from physical models to black box models were discussed by various authors in this book. Fault detection indicates a deviation of performance from expectation by a prespecified threshold to establish whether a fault has occurred or not. The simplest threshold method involves range checking, where the measurements are expected to be within certain fixed bounds. These thresholds are determined based upon heuristics or statistical methods. In a statistical approach, the standard deviation about its mean is often used to check if the measurements fall within the specified bound. For example, assuming that the data is normally distributed, a measurement can be labeled as fault-free, with 99.5% confidence, if it falls within 3 standard deviation from the mean. In the past, better performance has been achieved using such statistical methods. They mentioned that tests with different sets of simulated and experimental data must be carried out to analyze the pros and cons of each method. The sensitivity (fault detection rate) can be dramatically improved through the use of energy models for expected performance. The energy model is used to predict the output of a process under normal conditions. Faults are indicated when the difference between the predicted and measured output is greater than the threshold. In Section 7, the proposed dynamic threshold method is benchmarked relative to the statistical fixed threshold method mentioned above.

Pang et al. [25] developed an online strategy to detect sensor faults in centrifugal chillers. They grouped a set of correlated variables and captured the systematic trends of the chillers using two models based on principal component analysis. Subsequently they calculated the Q-statistics of the two models and compared them to a fixed threshold based on the analysis of the residual subspace. This method is based on the premise that data points making a large contribution to the Q-statistics correspond to faulty system behavior. However, it is well know that large variance in the data is also possible due to the ever changing weather conditions and occupancy status, which may also make large contribution to the Q-statistics. Thus increasing the number of false alarms. Also, setting a fixed threshold may exacerbate the detection of faults if it is set too high or too low.

Navarro-Esbri et al. [21] proposed a dynamic model suitable for real-time refrigerant leakage detection in vapor compression chillers. They suggested an auto-regressive Kalman filter-based model to predict the suction pressure using the inlet temperature of the secondary fluids to the evaporator and condenser and the compressor rotation speed as the independent variables. The fault detection was accomplished by evaluating the residual associated to the output variable for each data point. The residuals were calculated using Eq. (1). They mentioned that the residual defined as the quadratic error underestimates the small errors and, in this way, it makes the methodology less sensible to false alarms.

$$r(t) = |x(t) - \bar{x}(t)|^2 \tag{1}$$

Where $x(t)$ is the value provided by a sensor and $\bar{x}(t)$ is the value calculated by the model. Subsequently, they proposed dynamic thresholds for the residuals, being obtained in a recursive way using Eq. (2).

$$L(t+1) = r(t) + \alpha\sigma(t)$$
$$\alpha = \frac{max[r(t_0), ..., r(t_{init})]}{\sigma[r(t_0), ..., r(t_{init})]} \tag{2}$$

where $\sigma(t)$ is the standard deviation of the prediction error actualized each time a new residual is computed until time t, and α is an adjustable parameter that is computed during the initialization process. The results showed that the dynamic residual evaluation method was able to detect refrigerant leakage in steady-state or transient operation, being robust enough to avoid false alarms due to transients associated to cooling load variations. In Section 7, the proposed dynamic threshold method is benchmarked relative to the dynamic residual evaluation method mentioned above.

Seem [24] used a generalized extreme studentized deviate (GESD) to detect outliers in daily energy consumption data. However, the GESD test assumes that the data come from a normal distribution, if not then non-normality of the data is detected rather than the presence of outliers. It is shown in this paper that energy consumption data is not necessarily normally distributed (see Section 3). In contrast, data may be highly skewed that may remain unaffected by transformations to modify it to a normal distribution. Therefore, more robust fault detection methods are required that do not assume the distribution of the data apriori.

Reddy [23] highlighted the reasons for limited implementation of automated fault detection though there had been a widespread proliferation of research papers in this field for several years. They also reviewed various functions and capabilities of automated fault detection systems and summarized some pertinent issues, which are listed below.

- A basic issue with fault detection is the specification of threshold levels to flag deviations between real and modeled system performance. Assigning a fixed threshold is problematic because there is a trade-off between detection sensitivities and false alarm rates. Low threshold limits may detect smaller faults but are also likely to lead to more false-alarms and vice-versa.
- The behavior of buildings and HVAC systems is more difficult to predict due to the lack of accurate models. Since most HVAC designs are unique, the amount of time, money and effort required to derive a model is significant.

Najafi [19] discussed some of the key issues relevant to fault detection in HVAC systems. The methods available, back then, to detect faults or anomalous performance in building HVAC systems were labor intensive. Building operators or engineers had to use intuition and various rules of thumb to identify problems. Due to the intensive nature of such tasks, they were not routinely performed in buildings. From a technical standpoint, modeling limitations and measurement constraints were the main challenges facing fault detection. They proposed a top-down diagnostic approach based on Bayesian network, which dealt with various issues related to fault detectability in air handling units.

Wu and Sun [27] presented a spatial-temporal partition strategy to create summer, spring, day and night partitions. They individually calculated the fixed thresholds using abnormal data for each partition. Although separate thresholds are set for each partition, it may not be sufficient to handle the inter-partition variations. For example, some days in summer may be much cooler than the typical, therefore the inter-partition variation may also be higher than usual. Therefore, a fixed threshold may lead to numerous false alarms or missed detection opportunities. The authors also mentioned that the spatial and temporal conditions are related to different environmental conditions and human occupancy profiles.

Since data-driven models require a considerable amount of data, the partitioning strategy resulting in multiple smaller datasets may affect the accuracy of the trained models. An alternative is to include the spatial-temporal variables, such as the month, day of the week, weekday and weekend indicators, in the modeling process rather than partitioning the dataset. Subsequently, setting a dynamic threshold may be more appropriate. In this paper, we explore the feasibility of such an alternative strategy.

Pang et al. [22] presented a framework to infer the presence and location of faults by comparing the actual performance of buildings with the expected performance in real time. They showed that the deviation between the actual and modeled performance is dynamic in nature, which varies throughout the prediction window depending on the weather condition, occupancy status, noise, and model error. Magoulès et al. [16] used a recursive deterministic perceptron neural network model with a fixed threshold to detect abnormal energy consumption faults. They used a synthetic dataset of an office building in France, which is simulated in EnergyPlus. They were able to achieve high accuracy of fault detection by using a fixed threshold of 1000 Joules. However, the building used to generate the synthetic data had only one floor and one room. In reality buildings are much more complex in nature exhibiting energy consumption variation that is extremely complicated. Therefore, the task of isolating unusual variation due to faults from variation due to weather, spatial, and temporal conditions is comparatively more demanding. Bonvini et al. [4] presented an algorithm called Unscented Kalman Filtering for both whole-building and component-level energy fault detection. Although a fixed threshold value is used to detect faults, they have mentioned that a dynamically adaptive threshold can further improve the effectiveness of the fault detection algorithm. However, it was outside the scope of their research.

An important issue in developing efficient fault detection algorithms is that data associated with the faulty operation of systems is unavailable apriori. Beghi et al. [2] used a semi-supervised data-driven approach to detect faults in water chiller systems. They have exploited a principal component statistical model with a fixed threshold to identify anomalies relative to normal operating conditions. They mentioned that detecting faults early in the course is necessary to reduce energy wastage, prevent further deterioration, and prolong equipment life. Frank et al. [11] proposed a hybrid data-driven model based fault detection for commercial buildings. According to the authors, conventional rule-based fault detection requires expensive instrumentation and valuable engineering labor that limits implementation opportunities. Since data-driven methods require large and comprehensive training data sets that are often unavailable from real buildings, a synthetic database of faults were generated using energyplus for training the models. For fault detection, seven different approaches were compared including two regression methods and five classification methods. They tested the fault detection methods using a 880 feet^2, single-story security building located in Golden, Colorado, USA. The training dataset consisted of 14,965 mixed observations of daily normal working behavior and seven faults at a variety of severity levels. Initially the testing dataset consisted of 10,585 mixed observations. However, the presence of an active fault model does not necessarily produce faulty behavior in the building. By defining faulty behavior as any behavior that produces a 5% deviation (minimum 100 W) from the simulated baseline daily energy consumption, they reduced the test dataset size to 4,805 observations. For the regression models, a threshold of 5% was set for fault detection. They concluded that both regression and classification methods showed great potential for whole-building fault detection.

The main points from the literature review are summarized below:

1. Fault detection indicates a deviation of performance from predicted values by a specified threshold.
2. Thresholds are mostly determined based upon heuristics or statistical methods. These thresholds are fixed and do not change as per system dynamics.
3. In a statistical approach, the standard deviation about its mean is often used to check for faults. These statistical methods assume normality of the data.
4. Assigning a fixed threshold is problematic because there is a trade-off between detection sensitivities and false alarm rates. Low threshold limits may detect smaller faults but are also likely to lead to more false-alarms and vice-versa.
5. A fixed threshold may also lead to a significant delay in fault identification.
6. The deviation between the actual and modeled performance is dynamic in nature. Therefore, setting a dynamic threshold may be more appropriate to detect faults early and reduce the false alarm rate.
7. A novel dynamic residual evaluation method was proposed by Navarro-Esbri et al. [21] to detect faults in in vapor compression chillers. Since then it has not been re-implemented. Most researchers have used fixed threshold methods to detect faults.
8. Modeling limitations and measurement constraints are the main challenges facing fault detection.
9. Data driven fault detection methods are easy to develop and use, therefore, they are suitable for virtually any kind of problem where sufficient data is available.

We are presenting a tree-based ensemble learning algorithm, which is described in Section 4, to overcome the modeling limitations identified in this section. We also show, in this paper, that a fixed threshold, which is universally used by researchers, is insufficient to make early fault detection. Therefore, a novel dynamic threshold method, which is described in Section 5, is proposed to determine occurrences of faults in real-time. In this method, the threshold is adjusted and updated in real-time as per the moving average and moving standard deviation of the predicted values. The feasibility of the proposed fault detection model is tested in Section 6. In Section 7, the proposed dynamic threshold method is comparatively analyzed relative to fixed thresholds and the existing dynamic threshold method by Navarro-Esbri et al. [21].

3. Description of faults and synthetic dataset

Building faults are operating abnormalities that degrade building performance, which include using more energy than normal operation or failing to maintain building temperatures according to the thermostat set points [8]. Synthetic data signifying building faults, simulated in EnergyPlus, is used in this research to develop and evaluate the performance of the proposed fault detection model. Synthetic databases provide a consistent basis for research, energy model development, and setting benchmarks. Furthermore, the synthetic data can be generated in a matter of minutes, rather than years required to collect a similar data from an actual building. Therefore, synthetic data enables data science and machine learning efforts that, due to a lack of access to real data, may have otherwise not left the ground.[1]

The synthetic datasets are obtained from EnergyPlus simulations with prototype IDF representing a medium- and large-size commercial office building located in Chicago, U.S.A. The IDF file is developed by the U.S. Department of Energy's Building Technologies Program in collaboration with the Pacific Northwest National Laboratory, Lawrence Berkeley National Laboratory, and

[1] http://news.mit.edu/2017/artificial-data-give-same-results-as-real-data-0303.

Table 1
Basic information of medium-size office building used to generate the synthetic dataset.

Standard	ASHRAE 90.1-2013.
Locations	Zone 5A: Chicago (Cold,humid).
Total Floor Area	4979.6 m^2.
Thermal Zoning	Each floor has four perimeter zones and one core zone. Percentages of floor area: Perimeter 40%, Core 60%.
Cooling Type	Packaged air conditioning unit.
Distribution and terminal units	VAV terminal box with damper and electric reheating coil.
HVAC Sizing	Autosized to design day.
Weather files	TMY3 customized with real weather data for years 2015 to 2016.

Table 2
Basic information of large-size office building used to generate the synthetic dataset.

Standard	ASHRAE 90.1-2013.
Locations	Zone 5A: Chicago (Cold,humid).
Total Floor Area	46321.5 m^2.
Thermal Zoning	Each floor has four perimeter zones and one core zone and one IT closet zone. Percentages of floor area: Perimeter 29%, Core 70%, IT Closet 1% The basement has a data-center zone occupying 28% of the basement floor area.
Cooling Type	Water-source DX cooling coil with fluid cooler for data-center in the basement and IT closets in other floors. Two water-cooled centrifugal chillers for the rest of the building.
Distribution and terminal units	VAV terminal box with damper and hot-water reheating coil except non data-center portion of the basement and IT closets that are served by CAV units.
HVAC Sizing	Autosized to design day.
Weather files	TMY3 customized with real weather data for years 2015 to 2016.

Table 3
The faults introduced in the buildings.

Dataset	Building	Parameter	Change
D1	Large-size office	Water cooled centrifugal chiller - Reference COP	6.28 to 4.3
D2	Medium-size office	Packaged air conditioning unit - Reference COP	3.4 to 2.38
D3	Large-size office	Variable air volume Motor efficiency Fan total efficiency	 0.94 to 0.65 0.62 to 0.42

National Renewable Energy Laboratory.[2] Basic properties of the selected buildings are given Tables 1 and 2.

The synthetic database consists of the daily cooling energy consumption over a period of two years (2015–2016). The corresponding data for the year 2015 represent normal system behavior, whereas the data for the year 2016 represent both normal and faulty system behavior. These faults are induced in EnergyPlus by modifying the efficiency parameters of a chiller, packaged air conditioning unit (PACU), and variable air volume (VAV) distribution system as prescribed by Basarkar et al. [1]. The changed parameters are listed in Table 3. The normal energy consumption dataset for the year 2015 is used to train the XGBoost model. Subsequently, the dataset for the year 2016, comprising of both normal and faulty conditions, is used to test the effectiveness of the proposed fault detection model. The days on which the cooling equipment is not used are excluded from the test dataset.

[2] https://www.energycodes.gov/development/commercial/prototype_models.

Table 4
The features used to develop the models.

Month
Day of the week.
Holiday indicator.
Daylight savings indicator.
Diffuse radiation - daily average, sum, maximum, minimum, and variance.
Direct radiation - daily average, sum, maximum, minimum, and variance.
Dew point temperature - daily average, sum, maximum, minimum, and variance.
Dry bulb temperature - daily average, sum, maximum, minimum, and variance.
Relative humidity - daily average, sum, maximum, minimum, and variance.
First lag of daily energy consumption.

As mentioned in Section 2, energy consumption data is not necessarily normally distributed. This property is illustrated using distribution plots overlaid with a kernel density estimation (KDE) functions. A KDE is an useful tool to visualize the shape of data that is a continuous replacement for the discrete histogram. As shown in Fig. 1, the energy consumption data generated by EnergyPlus, for both the medium- and large-size buildings, are highly skewed. The data is skewed to the extent that transformations, such as log-transform or box-cox transform, cannot modify the data to a normal distribution, as shown in Fig. 2. For this reason, statistical anomaly detection methods, such as generalized extreme studentized deviate that assumes normally distributed data, is not going to work well. However, ensemble machine learning models, such as XGBoost, is insensitive to skewness because it is based on the principle of decision tree boosting that use a sampling technique. It has been proven that tree based models, supplemented with sampling technique, is robust to skewed datasets [9].

Employing substantial input variables (also known as Features) is crucial for machine learning based building energy use prediction. The features supplied to the XGBoost models are given in Table 4. These features are chosen based on the review article by Wang and Srinivasan [26]. The first lag of the daily energy consumption is also included in the model development. This lagged feature is introduced to transform the time-series forecasting problem into a supervised machine learning problem, as suggested by Chakraborty and Elzarka [5].

4. Developing the predictive model using extreme gradient boosting (XGBoost)

Choosing an effective machine learning algorithm is critical to achieve good performance with the resulting fault detection model. Most review articles, published in the recent past, suggest that ensemble algorithms are more effective than single prediction algorithms [10,18]. Recently, Chakraborty and Elzarka [5] showed that an ensemble algorithm, known as extreme gradient boosting (XGBoost), produces more accurate energy models compared to artificial neural networks and degree-day ordinary least square regression. The high accuracy of XGBoost can be attributed to the machine learning theory stating that multiple weak learners can produce better models than a single strong learner. They used a synthetic dataset generated in EnergyPlus. The process of feature engineering, feature selection, hyper-parameter optimization is explicitly described in the paper. In addition, the importance of these intermediate steps is highlighted by adopting a comparative analysis technique. In this study, we have followed the process of energy model development adopted in [5].

Gradient boosting decision tree is a popular machine learning algorithm and has quite a few effective implementations such as XGBoost and LightGBM. In this research, we have used

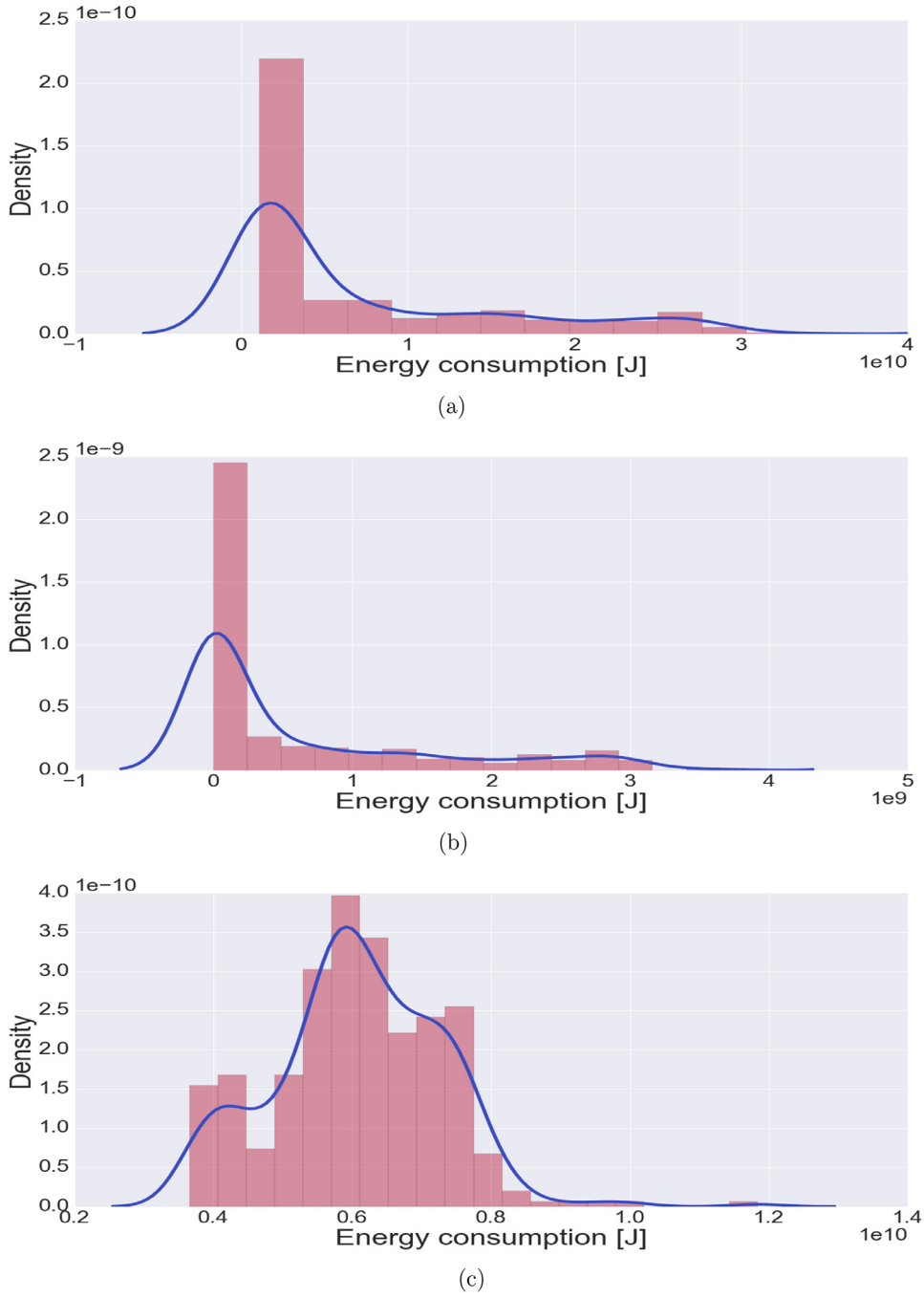

Fig. 1. Distribution of the training data (Year 2015) (a) Data distribution for the water cooled centrifugal chiller. (b) Data distribution for the packaged air conditioning unit. (c) Data distribution for the variable air volume system..

XGBoost to generate the predictive energy consumption models. Predictions from the XGBoost model are used in tandem with the proposed dynamic threshold method to isolate the faulty behavior from normal behavior. Boosting is a general term in machine learning where multiple weak learners such as regression trees are ensembled to create a single strong learner [17, p. 154]. The basic idea behind this procedure is to learn sequentially in which the current regression tree is fitted to the residuals (errors) from the previous trees. This new regression tree is then added to the fitted model to update the residuals. It is worth noting that statistical learning approaches that learn slowly such as boosting tend to perform well [14, p. 322]. The principle of gradient boosting further enhances the flexibility of the boosting algorithm by constructing the new regression trees to be maximally correlated to the negative of the gradient of the loss function. This helps in convergence of the loss function and allows arbitrary differentiable loss functions to be used in the model building process [6,20]. Readers may refer to Hastie et al. [12, chaps. 9 & 10] for further details about regression trees and boosting. In general, gradient boosting can be represented by Eq. (3).

$$\widehat{y}_i = \phi(x_i) = \sum_{k=1}^{K} t_k(x_i), \quad t_k \in T \tag{3}$$

Where y_i are the predicted response, x_i are the inputs and K is the number of functions in the function space T. In XGBoost these functions are introduced as parameters which allows to find functions t_k that fit the data very well while training, thus finding

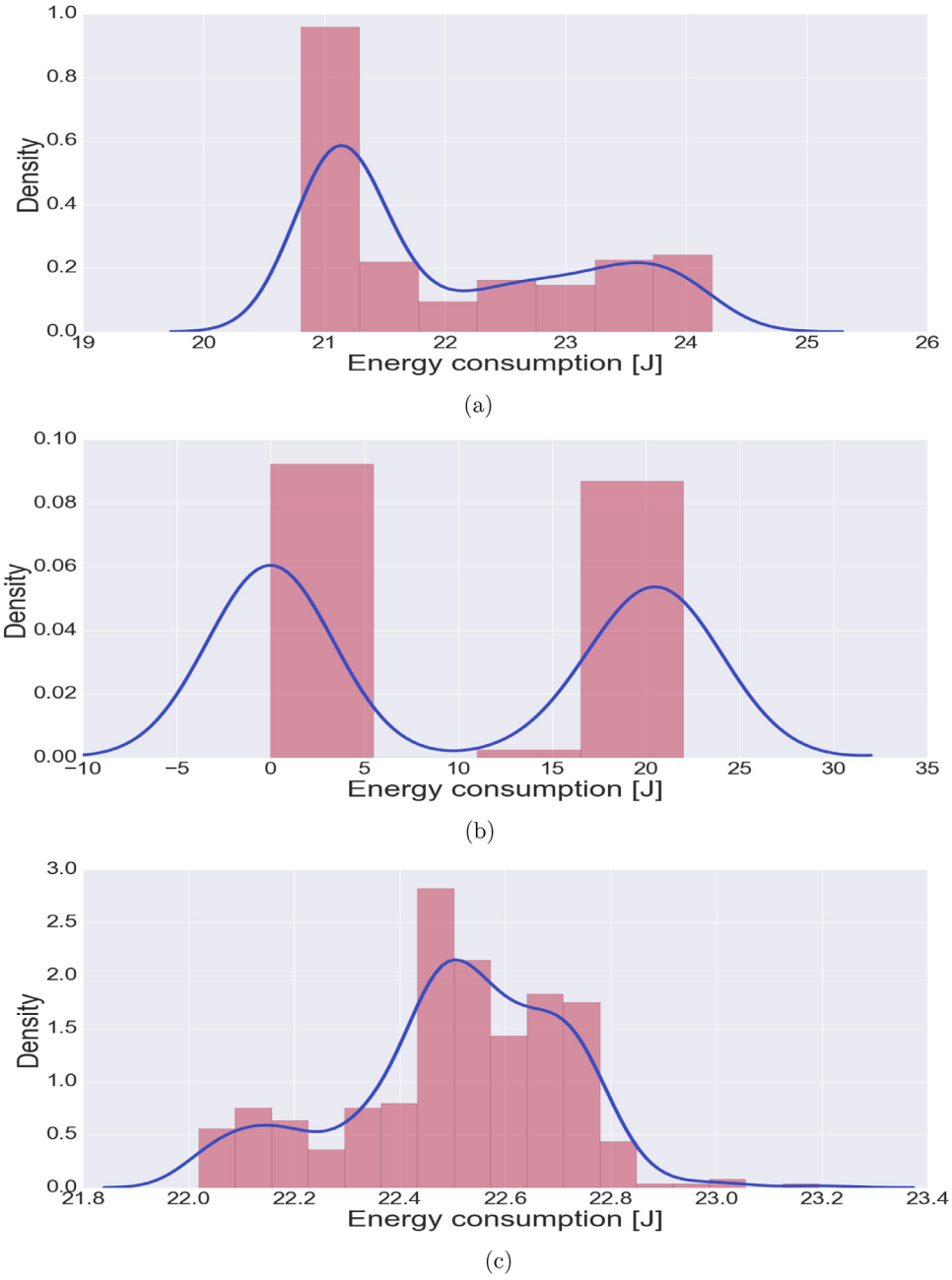

Fig. 2. Distribution of the training data after log transformation (Year 2015) (a) Data distribution for the water cooled centrifugal chiller after log transformation. (b) Data distribution for the packaged air conditioning unit after log transformation. (c) Data distribution for the variable air volume system after log transformation..

the corresponding regions automatically [7]. The functions t_k are learned by minimizing the following objective function.

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(t_k) \tag{4}$$

where l is a differentiable loss function and Ω is the regularizing function that penalizes overly complicated models. It is worth noting that XGBoost is built with OpenMP support, which is a shared memory multiprocessing application programming interface. In other words, OpenMP allows XGBoost to efficiently use all the CPU cores in parallel while training, making it extremely fast. In addition, XGBoost presorts the independent variables at the beginning of the training process, which further reduces the training complexity and computational time [7].

The following steps are carried out to generate the predictive energy model using XGBoost. Chakraborty and Elzarka [5] have shown that these steps are absolutely essential to generate accurate predictive models of building energy consumption.

- Feature selection - The purpose of model learning is to find out the proper relationship between the features and the target. It is imperative to reasonably choose a subset of appropriate features, which improves the accuracy and practicality of the resulting models [28]. The most useful and important input features are selected based on cross validated recursive feature elimination, which is a wrapper-based feature selection method. The function of recursive feature elimination is to select features by recursively considering smaller and smaller sets of features. First, the algorithm is trained on the initial set of

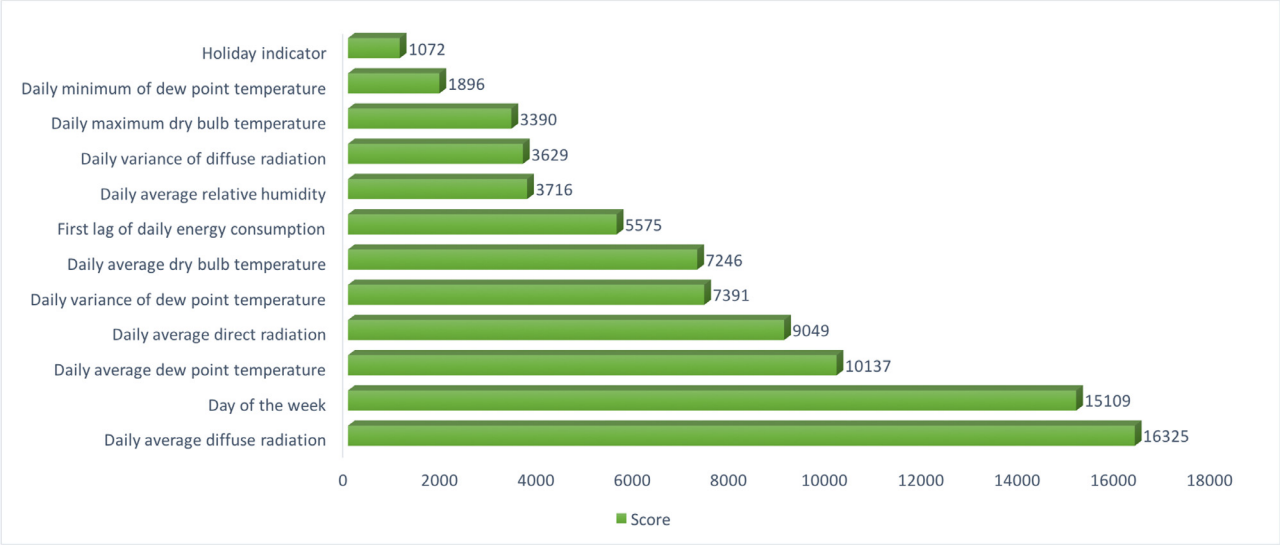

Fig. 3. Feature importance assigned by the XGBoost model for dataset D1.

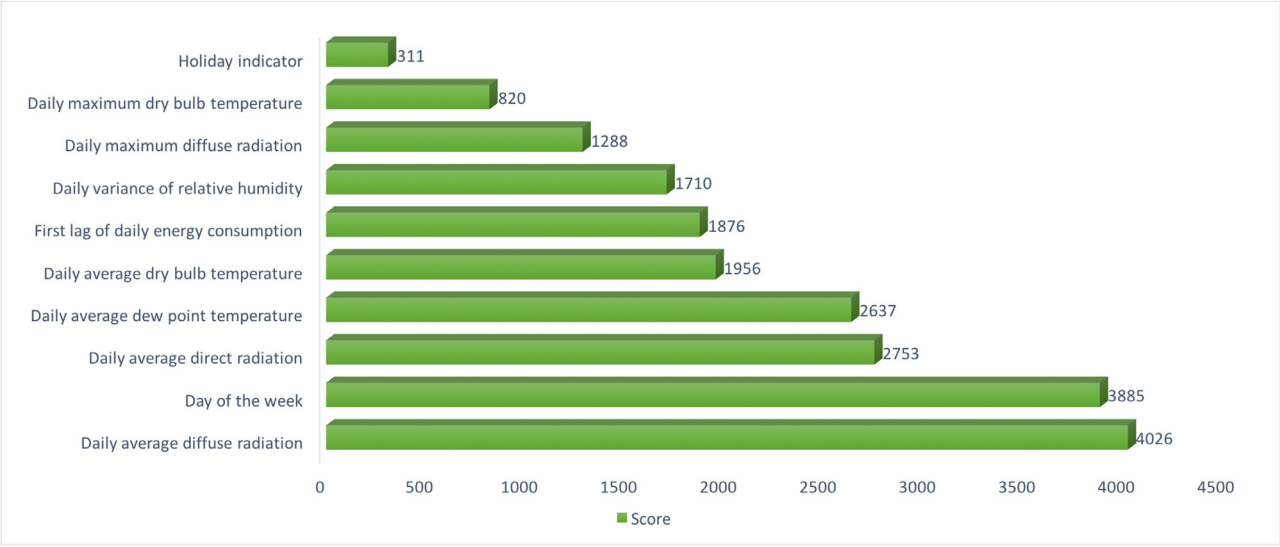

Fig. 4. Feature importance assigned by the XGBoost model for dataset D2.

features and the importance of each feature is obtained through the feature importance attribute of XGBoost. Then, the least important features are pruned from current set of features. This procedure is recursively repeated on the pruned set until the optimal features are obtained. The final set of features selected for each model and their corresponding scores (generated by XGBoost) are shown in Figs. 3–5. The best set of features is obtained using a scikit-learn class called "RFECV".

- Hyper-parameter optimization - In the context of machine learning, hyperparameters are parameters (e.g. number of trees in XGBoost, number of neurons in ANN, etc.) whose values are set prior to the commencement of the learning process. Hyperparameter search is the process of identifying an optimal hyperparameter set to improve the accuracy of the resulting models. Independent sets of hyperparameters from a prespecified search space are evaluated by grid search during k-fold cross-validation (cv). Each set of the independent hyperparameters are subjected through k-folds cv in the pursuit of finding the optimal hyperparameter. A k-fold cv is chosen because it is suitable for hyperparameter search of machine learning models with time series data [3]. The best set of hyper-parameters is obtained using a scikit-learn class called "GridSearchCV". Subsequently, the best set of hyperparameters is used for final model generation, which is then used to make predictions on the testing dataset.

5. The dynamic threshold method

Model-predicted values are never perfect, due to the presence of noise and modeling error, even in the fault-free state. Therefore, these predicted values must be further treated before identifying the occurrence of faults in the system. The most common method of treatment is to apply a fixed threshold to the difference between the predicted and actual energy consumption values. If the difference is more than the threshold then a fault is identified. However, applying a fixed threshold may lead to several missed opportunities to identify faults, false alarms, and delayed identification of faulty conditions. In this research, we propose a dynamic threshold method to overcome these problems.

Let P be the predicted variable of a moving window with finite mean μ and finite non-zero variance σ^2. According to Chebyshev's

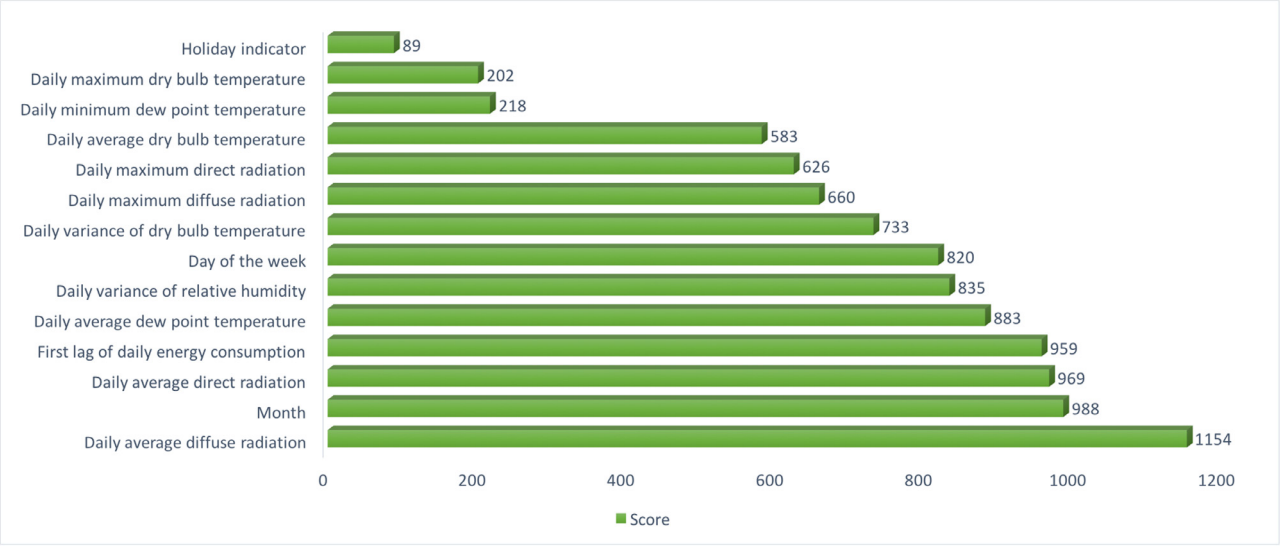

Fig. 5. Feature importance assigned by the XGBoost model for dataset D3.

inequality, for any real number $k > 0$,

$$Pr(|P - \mu| \geq k\sigma) < \frac{1}{k^2} \tag{5}$$

In probability theory, Chebyshev's inequality guarantees that, for a wide class of probability distributions, no more than $\frac{1}{k^2}$ of the distribution's values can be more than k standard deviations away from the mean. Therefore, according to Eq. (5), the threshold $T^i \in [\mu^i - k\sigma^i, \mu^i + k\sigma^i]$, where T^i, μ^i, and σ^i are the threshold, moving mean, and moving standard deviation of the ith data point in the dataset. Since we are only concerned when the actual energy consumption values are greater than the threshold, we consider the upper threshold $\mu^i + k\sigma^i$. The mean μ^i and standard deviation σ^i are updated, using Eqs. (6) and (7) respectively, in real-time to account for the dynamic variation of the energy consumption data over time.

$$\mu^i = \frac{P_i + P_{i-1} + \ldots + P_{i-j}}{j+1} \tag{6}$$

$$\sigma^i = \sqrt{\frac{\sum_{n=i-j}^{i} \left(P_n - \mu^i\right)^2}{j}} \tag{7}$$

where $j+1$ is the total number of data points in the moving window. Thus we obtain the threshold T^i which is updated, using Eq. (8), in real-time.

$$T^i = \mu^i + k\sigma^i \tag{8}$$

The parameters j and k are the free parameters, which controls the sharpness of the threshold curve. For example, when j is set very large, then the moving mean approaches the sample mean, which makes the threshold curve rather smooth. k specifies the maximum number of standard deviations away from the mean that is allowable for a data point to be considered normal. j and k must be controlled by the user to create the desirable balance between the number of false alarms and the number of missed opportunities for fault detection. The influence of the parameters j and k on the accuracy of the fault detection method is explored in Section 6.

6. Feasibility of the proposed fault detection model

The feasibility of the proposed fault detection model is tested in this section. The synthetic datasets described in Section 3 are used to evaluate the feasibility of the proposed fault detection model. We have implemented the algorithm in Python and the source code is provided in Appendix A. As mentioned in Section 5, the proposed dynamic threshold method has two free parameters (j and k) that are user-specified. These free parameters are used to control the sharpness of the threshold curve. Setting the value of j too high will result in a smoother curve, which will decrease the number of false alarms but increase the number of missed opportunities to detect faults. The dynamic threshold tends to a fixed threshold as j approaches N (the total number of data points in the dataset). Similarly, setting a higher k will result in a more liberal threshold curve that is less punishing to the actual data points deviating away from the predicted values. The relationship between j and k with the F1 score of the resulting fault detector is illustrated in Figs. 6–8. These figures show that the F1 score starts high at $j = 2$ and $k \in (1, 2)$. The F1 score is relatively low for $k = 3$. Although there are intermittent peaks in the F1 score at $(j = 16, k = 1)$, $(j = 15, k = 1)$, and $(j = 5, k = 1)$ in Figs. 6–8, applying $k = 1$ will result in a very conservative threshold curve. A conservative threshold curve may detect more faults, thereby, increasing the F1 score, but it will lead to numerous false alarms, which is undesirable. Also, it is better to choose a lower value of j so as to create a sharper threshold curve when the variance in the dataset is high. This will prevent missed opportunities to detect faults. In contrast, when the variance in the dataset is low it is better to choose a higher value of j so as to reduce the number of false alarms. Therefore, considering the fact that the synthetic data exhibits high variance and a threshold curve that generates numerous false alarms is undesirable, the values of j and k are set at 2.

As mentioned in Section 2, the prime purposes of a fault detection model are: It must detect faults early in the course, and it must cut down the number of missed opportunities to detect faults without generating numerous false alarms. The feasibility of the proposed fault detection model is visually inspected in Figs. 9–11. These plots clearly indicate that the proposed dynamic threshold method can identify a system fault early in its course.

As shown in Figs. 9–11, the first fault appears in the system on April 15th. The proposed dynamic threshold method immediately identified the fault as soon as it entered the system on April 15th, as shown in these figures. The fault is fixed on August 31st and normal operation resumed from September 9th. As shown, there were few false alarms during the periods of normal

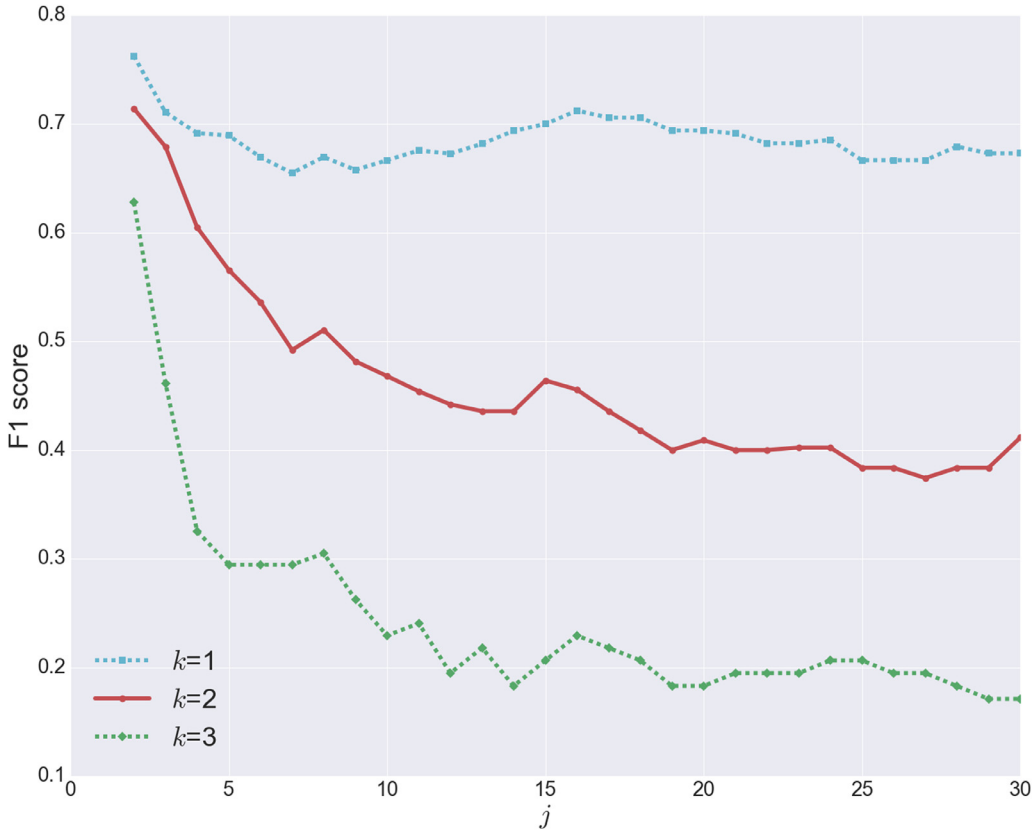

Fig. 6. Relationship between free parameters *j*, *k*, and F1 score for the large-size office building water cooled chiller.

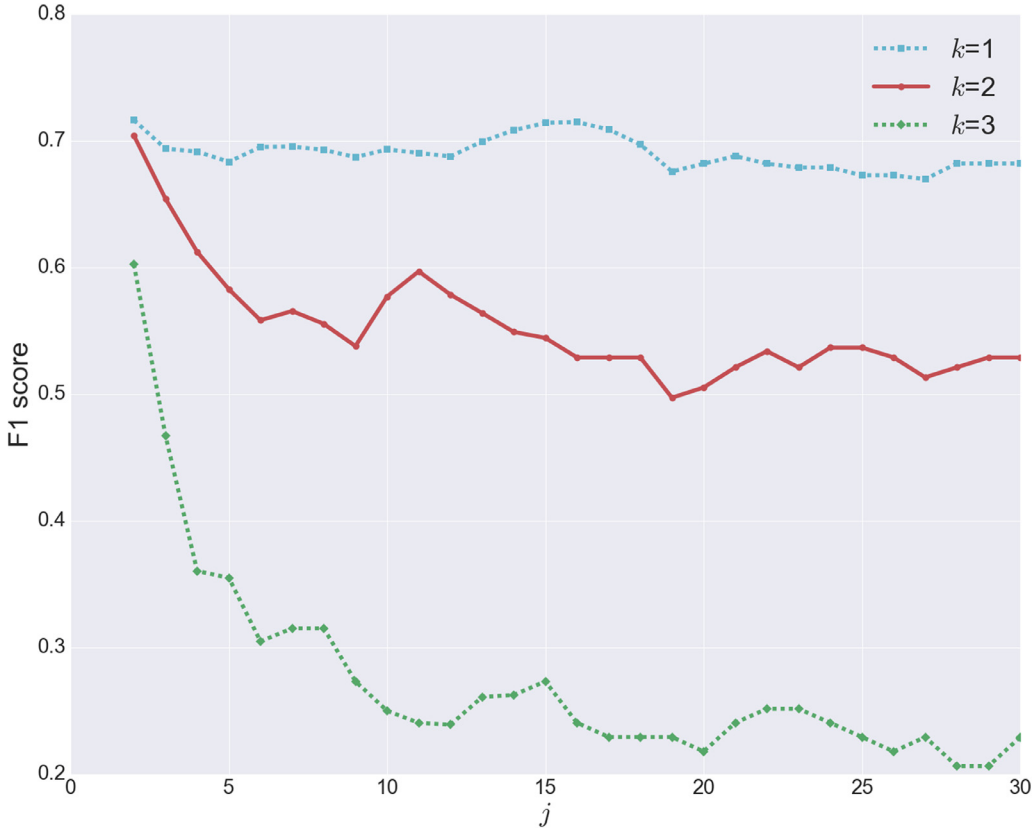

Fig. 7. Relationship between free parameters *j*, *k*, and F1 score for the medium-size office building PACU system.

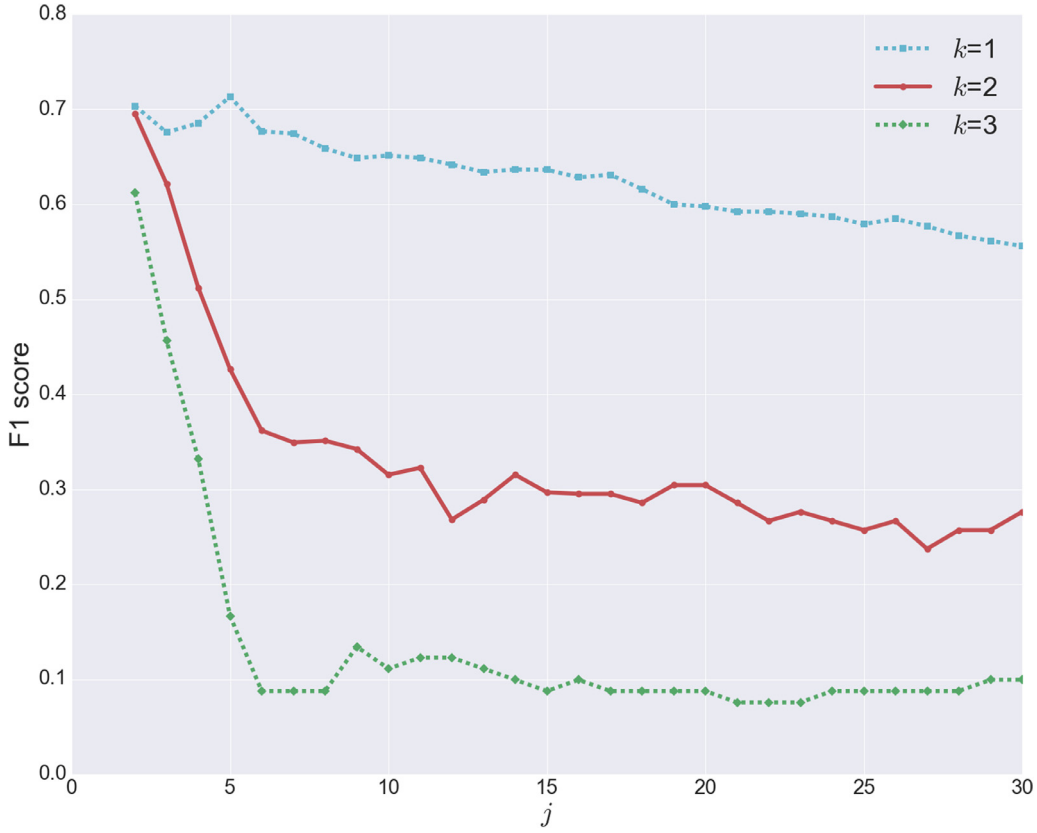

Fig. 8. Relationship between free parameters *j*, *k*, and F1 score for the large-size office building VAV system.

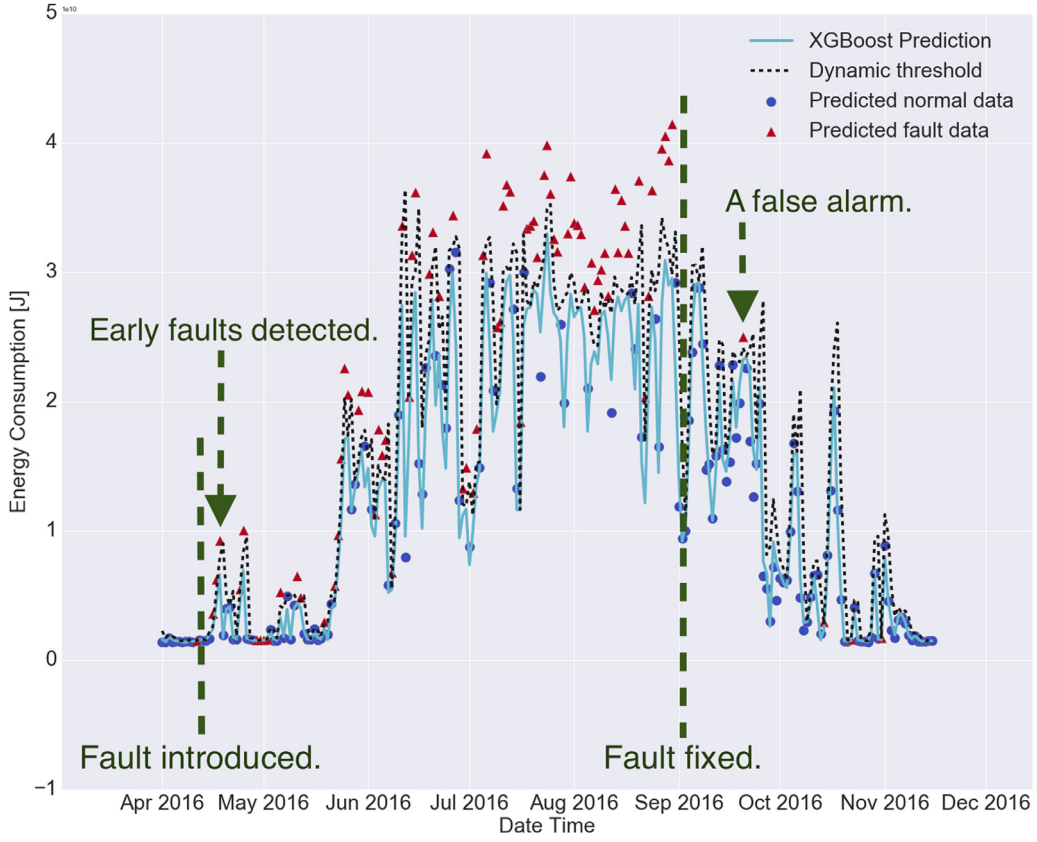

Fig. 9. Water-cooled centrifugal chiller fault detection results.

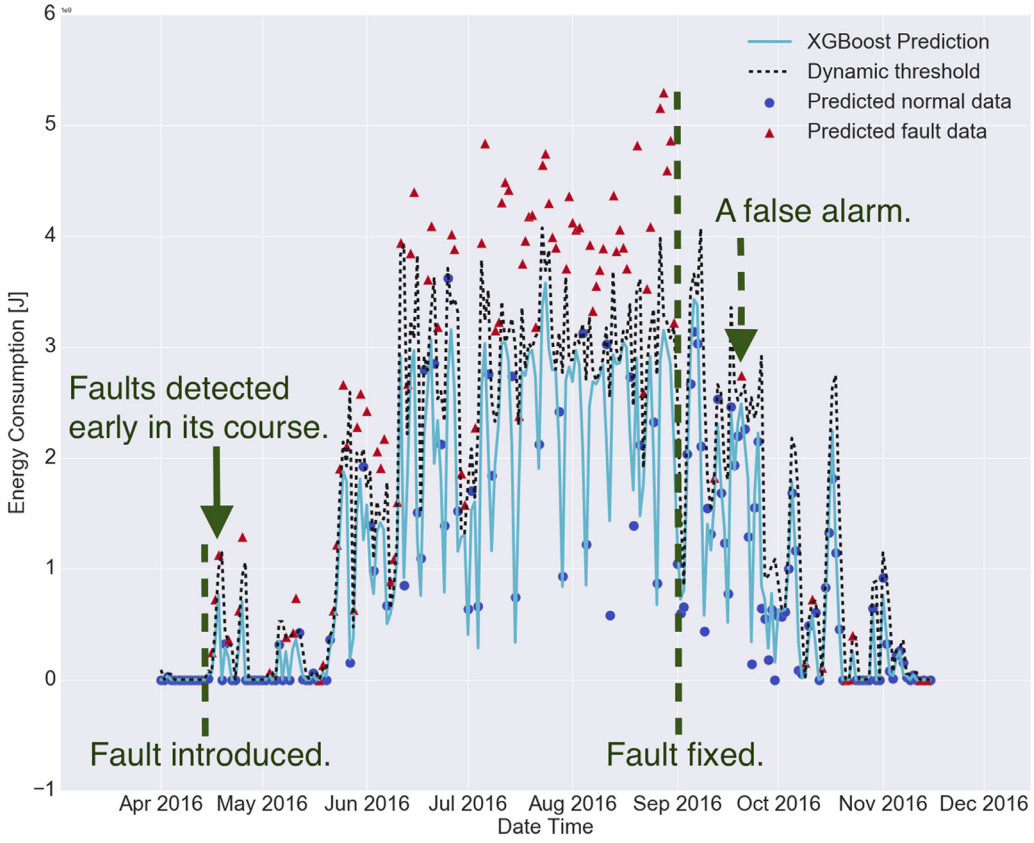

Fig. 10. Packaged air conditioning unit fault detection results.

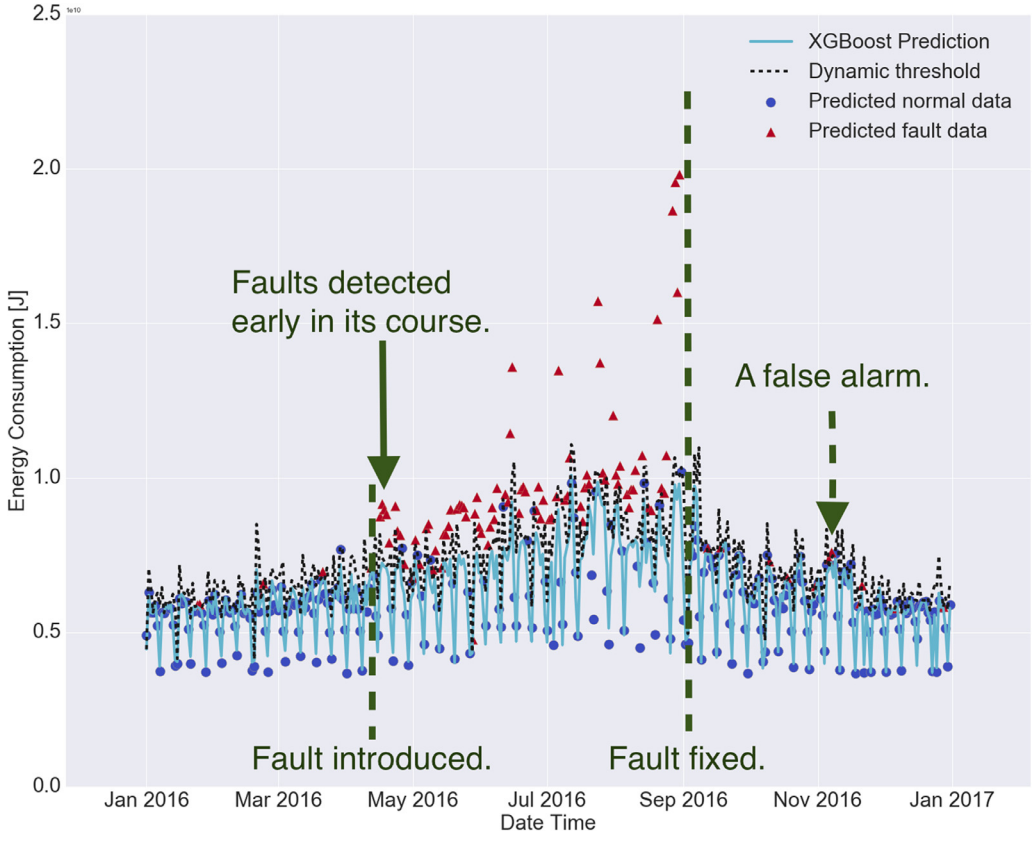

Fig. 11. Variable air volume distribution system fault detection results.

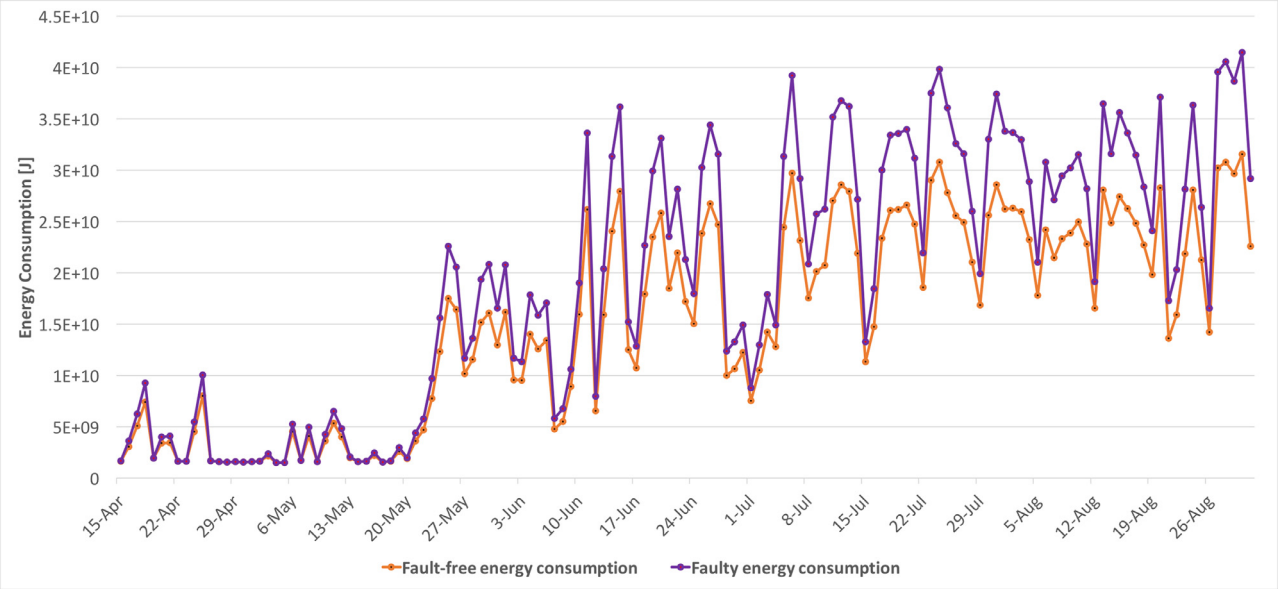

Fig. 12. Illustrating the difference between fault-free and faulty dataset for the water-cooled centrifugal chiller.

operation, which is a drawback of setting relatively low values for j and k. The sensitivity $(100 \times \frac{\#\text{ of faults detected}}{\text{Total \# of faults}})$ and specificity $(100 \times \frac{\#\text{ of normal points falsely detected as faults}}{\text{Total \# of normal points}})$ of theses models are 57.6% and 5.6% for the fault detection model shown in Fig. 9, 58.3% and 11.1% for the fault detection model shown in Fig. 10, and 60.4% and 7.5% for the fault detection model shown in Fig. 11. Readers must note that the low sensitivity values are due to the some of the missed opportunities to detect faults, as shown in the figures. These missed opportunities mostly occur in the lower ranges of energy consumption values. This is due to the fact that these energy systems are highly efficient, therefore, there is hardly any difference between faulty and fault-free energy consumption, in the lower ranges, as shown in Fig. 12. However, these missed faults are trivial because even if a fault is missed on a particular day, it is generally identified on the following day by the dynamic threshold method. Note that the false alarms mostly occur close to the dotted black line representing the dynamic threshold. Therefore, the facility manager may decide, based on visual inspection, whether or not to further investigate the faults that appear like false alarms. The advantage of the proposed dynamic threshold method is that the free parameters 'j' and 'k' can be utilized by the facility manager to control the number of false alarms with respect to the faults detected. Therefore, the facility manager can adjust j and k according to the cost of a false alarm or a misdiagnosed fault in that particular facility.

7. Comparative analysis

As mentioned previously in Section 2, an appropriate threshold is a key factor in fault detection. The traditional method of setting a fixed threshold often leads to missed opportunities to detect faults, delayed detection of faults or false alarms. However, very little work, related to dynamic threshold methods, has been done in the field of HVAC fault detection. Navarro-Esbri et al. [21] proposed a dynamic residual evaluation method. Since then most researchers have adopted fixed thresholds to evaluate the feasibility of their proposed fault detection algorithms. Therefore, the performance of the proposed dynamic threshold method in this research is benchmarked against fixed thresholds and the dynamic residual evaluation method proposed by Navarro-Esbri et al. [21].

A decisive factor in applying a fixed threshold is to specify how far is "far enough" for a data point to be considered a fault. In this research, three different fixed threshold values are evaluated, which are 1-, 2-, 3- standard deviations away from the mean. The results, shown in Figs. 13–15, indicate that the percentage of faults detected and false alarms decrease as one moves further away from the mean of the data. The sensitivity $(100 \times \frac{\#\text{ of faults detected}}{\text{Total \# of faults}})$ and specificity $(100 \times \frac{\#\text{ of normal points falsely detected as faults}}{\text{Total \# of normal points}})$ of the models, shown in Figs. 13–15 respectively, are 60.4% and 21.1%; 41.7% and 5.6%; 23% and 0%. The percentage of faults detected and false alarms increased significantly when the fixed threshold is at 1 standard deviation away from the mean in comparison to the fixed threshold at 3 standard deviations away from the mean. Also, the first fault appears in the system on April 15th. However, the fixed threshold could not identify the fault until mid-June (Fig. 15) or end-of-May (Fig. 13). Therefore, there is always a significant delay in fault identification. This behavior confirms the theory mentioned previously in Section 2. That is, if the threshold is set too high faults may be missed or if it is too low then false alarms will occur. Also, a fixed threshold may lead to a delayed identification of faults, which may cause significant damage to the equipment due to its extended use under faulty conditions. It is necessary to find the right balance between faults detected and false alarms. In this research, a fixed threshold of $T = \mu + 2\sigma$ is applied to the predicted values for comparative analysis with the proposed dynamic threshold method.

The effectiveness of the proposed dynamic threshold, described in Section 5, is benchmarked relative to the fixed threshold and the existing dynamic residual evaluation method described in Section 2. The results of the comparative analysis is given in Table 5. The performance is measured in terms of the precision, recall, and F1 score given in Eq. (9). Note that True positives (TP) are data points correctly labeled as faults. False positives (FP) refer to normal data points incorrectly labeled as faults. True negatives (TN) correspond to data points that are correctly labeled as normal. Finally, false negatives (FN) refer to faulty data points incorrectly labeled as normal.

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

$$Recall = \frac{TP}{TP + FN}$$

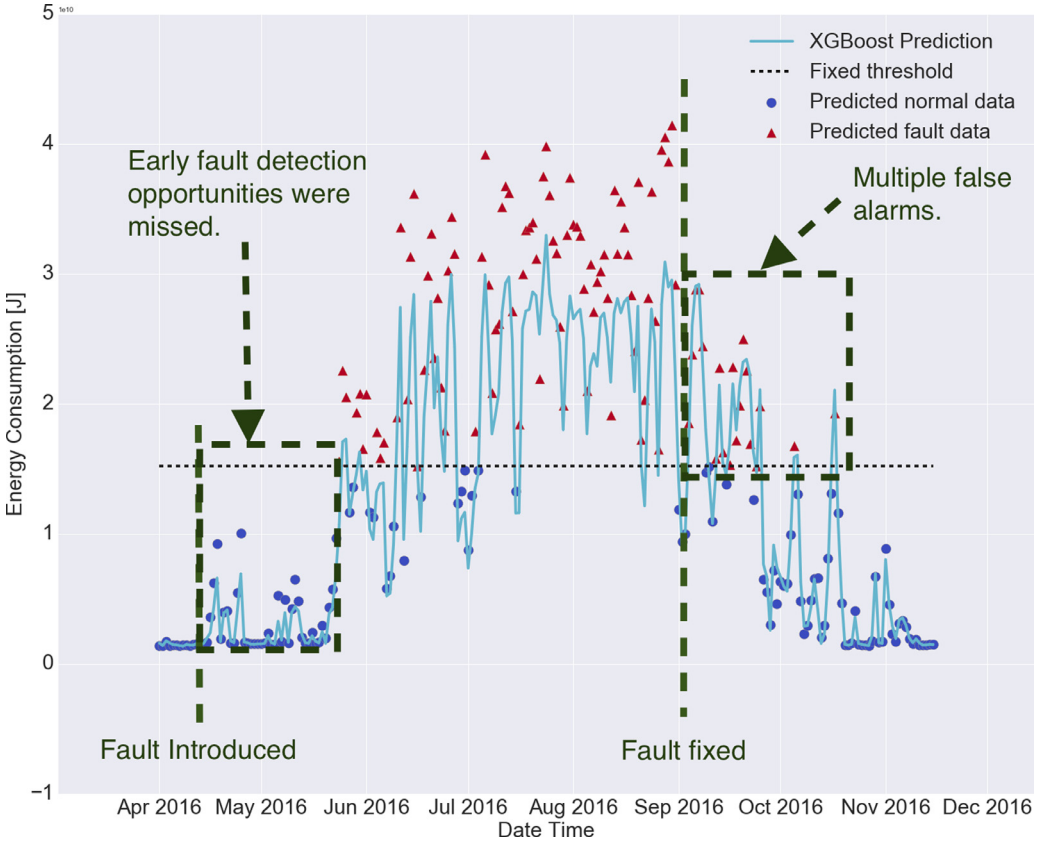

Fig. 13. Fault detection with a fixed threshold set 1 standard deviation away from the mean.

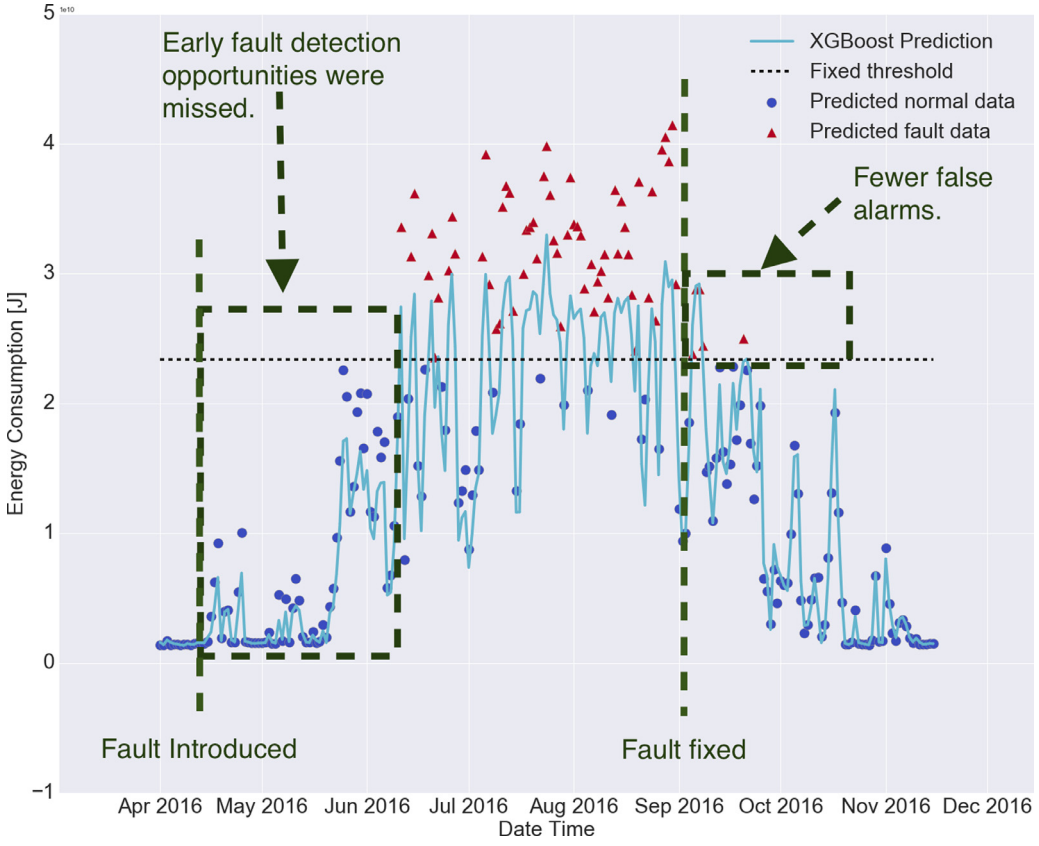

Fig. 14. Fault detection with a fixed threshold set 2 standard deviations away from the mean.

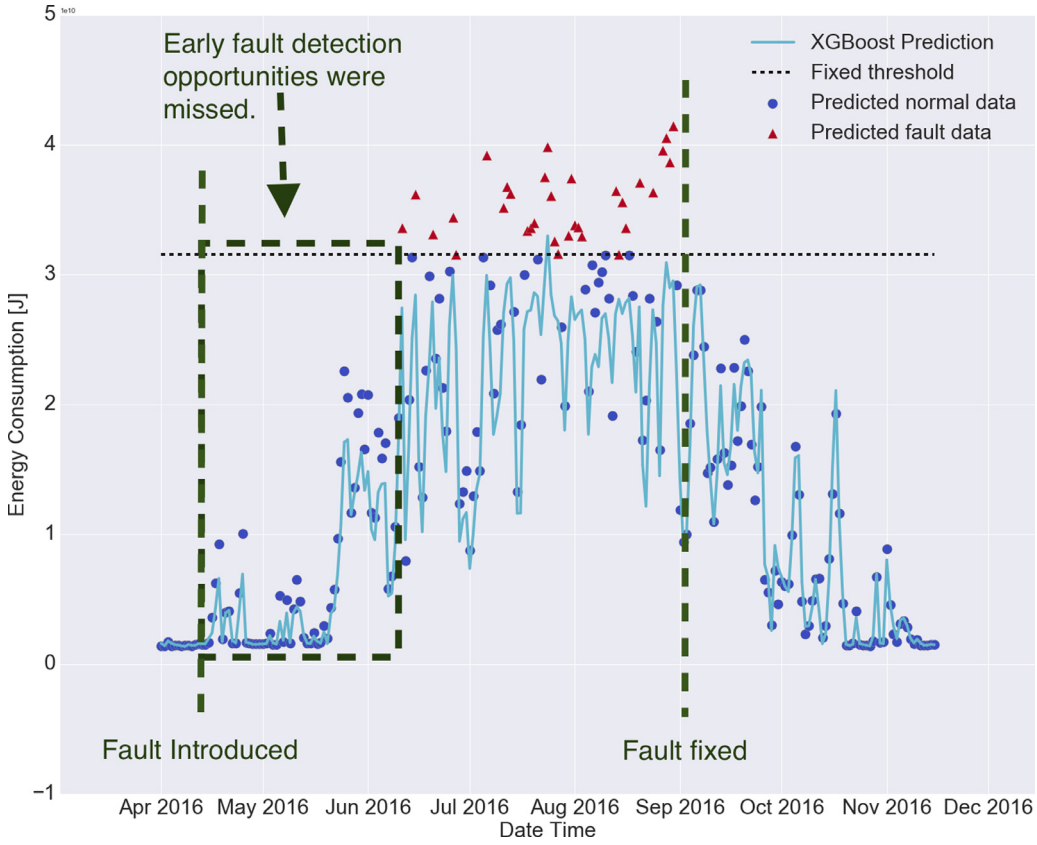

Fig. 15. Fault detection with a fixed threshold set 3 standard deviations away from the mean.

Table 5
Comparison of the proposed dynamic threshold relative to a fixed threshold and an existing dynamic residual threshold method proposed by Navarro-Esbri et al. [21].

Dataset	Threshold method	Precision	Recall	F1 score	Early fault detected
D1	Proposed dynamic	0.8	0.72	0.72	Yes
	Fixed	0.76	0.62	0.61	No
	Dynamic residual	0.76	0.41	0.26	Yes
D2	Proposed dynamic	0.77	0.7	0.7	Yes
	Fixed	0.75	0.63	0.62	No
	Dynamic residual	0.76	0.4	0.24	Yes
D3	Proposed dynamic	0.81	0.8	0.79	Yes
	Fixed	0.86	0.82	0.8	Yes
	Dynamic residual	0.72	0.64	0.51	Yes

$$F1\ score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{9}$$

The results in Table 5 clearly indicate that the dynamic residual evaluation method proposed by Navarro-Esbri et al. [21] performs worse, in terms of both precision and recall, compared to the fixed threshold and the proposed dynamic threshold. This seems to be the reason why researchers have avoided the use of dynamic residual evaluation method since its inception in 2006. As reviewed in Section 2, most researchers have adopted fixed thresholds to evaluate the feasibility of their proposed fault detection algorithms. The fixed threshold did not perform as well as the proposed dynamic threshold, in terms of both precision and recall, for datasets D1 and D2. However, the fixed threshold performed better than the other two methods for dataset D3. Also, readers must note that, for the same dataset (D3), when the faults are introduced in August, instead of mid-April then the precision and recall values with the fixed threshold decreases to 0.77 and 0.69 respectively. In contrast, the precision and recall values with the proposed dynamic threshold remains relatively stable at 0.8 and 0.79. This indicates that a fixed threshold is not flexible enough, in comparison to the proposed dynamic threshold method, for handling different types of datasets and faults.

The predictions are visually inspected in Figs. 16–18 to identify whether faults are detected early in their course by the three different threshold methods. Although the dynamic residual evaluation method proposed by Navarro-Esbri et al. [21] is able to detect an early fault, it gets uncalibrated soon after the model errors start increasing due to high summer demands, as shown in Fig. 18. This behavior is due to the procedure of routinely updating the threshold with the squared of the residuals (see Eqs. (1) and (2)), which suggest that this method is not mathematically robust. The reader can argue that, maybe, such behavior of a threshold method is harmless as long as it detects a fault early, correctly, and does not produce numerous false alarms. To answer such arguments, the dynamic residual evaluation method is also applied to a year-long fault-free dataset, as shown in Fig. 19. It can be seen that the threshold, in this case too, gets uncalibrated soon after misdiagnosing two normal data points as faults. In reality, implementing such a method would cause extra burden on the facility

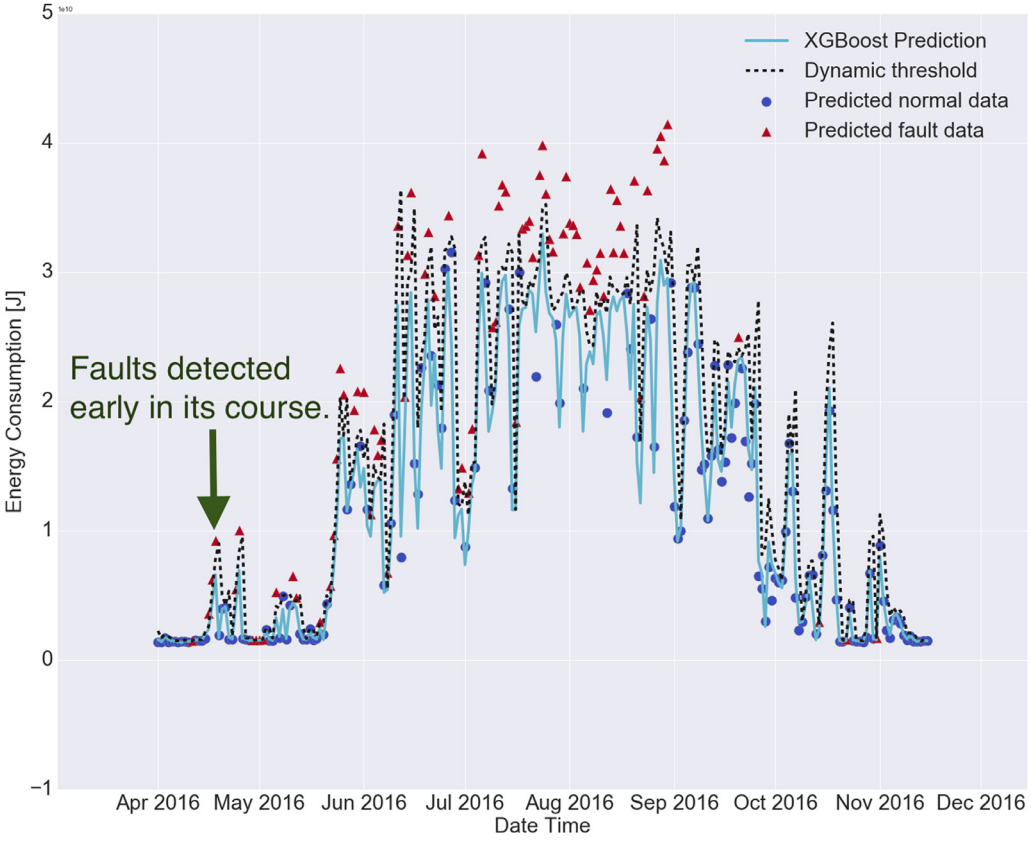

Fig. 16. Performance evaluation of the proposed XGBoost + dynamic threshold method with dataset D1.

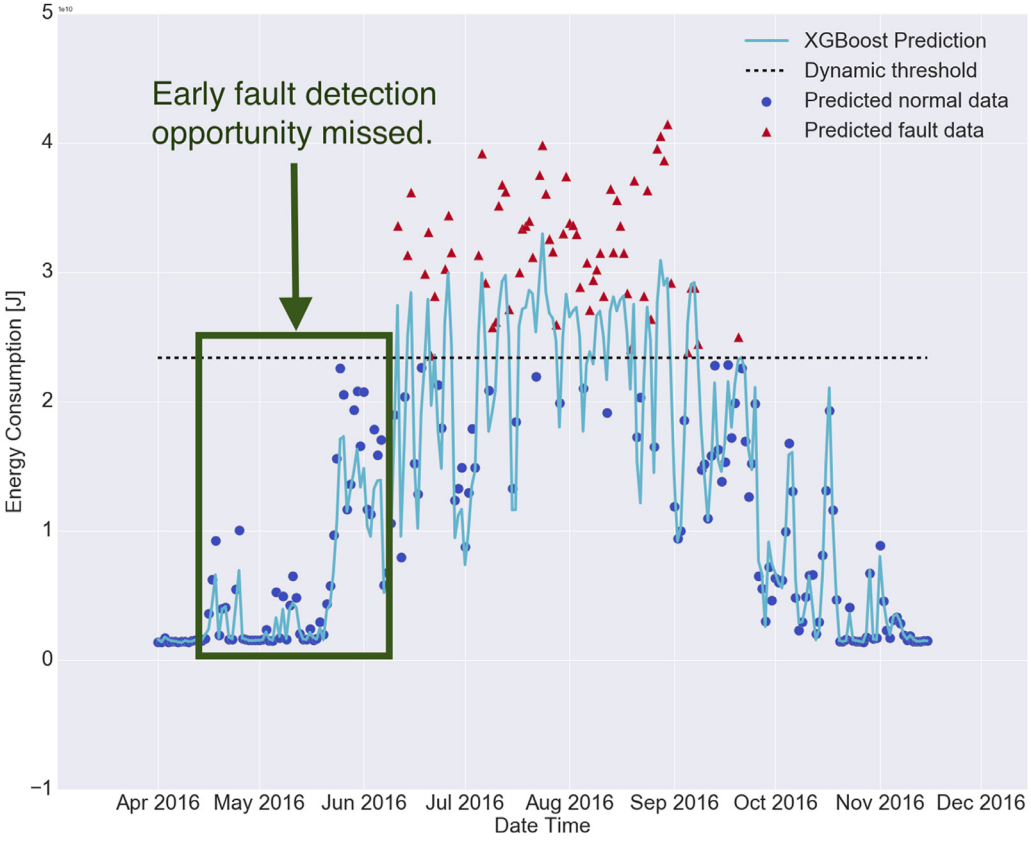

Fig. 17. Performance evaluation of the proposed XGBoost + fixed threshold method with dataset D1.

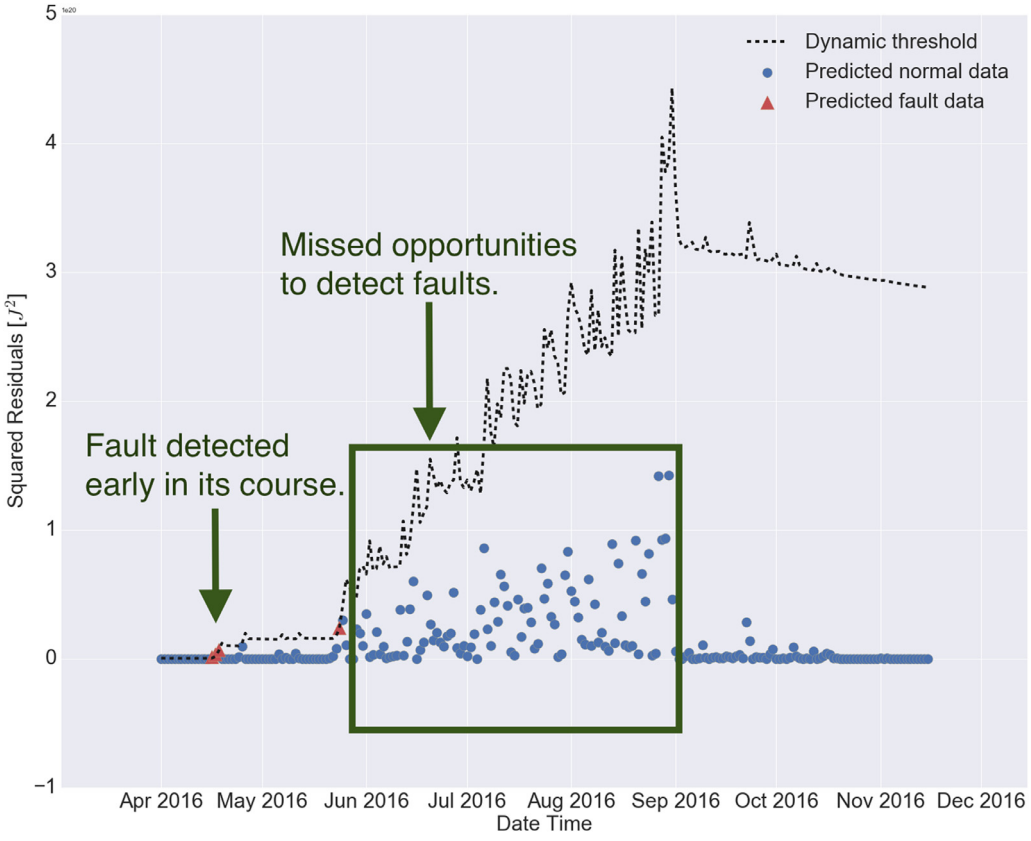

Fig. 18. Performance evaluation of the proposed XGBoost + residual evaluation method, proposed by Navarro-Esbri et al. [21], with dataset D1.

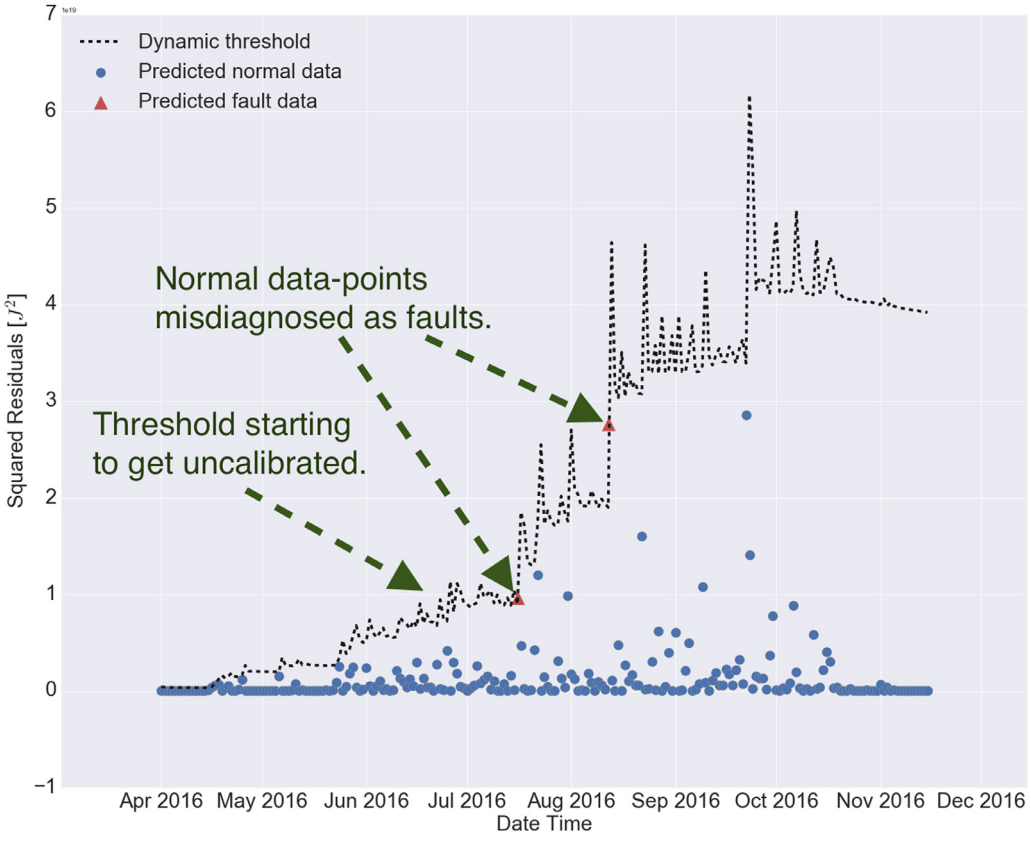

Fig. 19. Performance evaluation of the dynamic residual evaluation method, proposed by Navarro-Esbri et al. [21], with fault-free data.

manager to decide whether or not to further investigate a fault that has been detected. Needless to say that one must re-calibrate the threshold every-time it gets uncalibrated by manually removing the data-point/s (both faulty or fault-free) that caused the problem. As shown in Fig. 17, the first fault appears in the system in mid-April but, the fixed threshold could not identify the fault until mid-June, which is a significant drawback of this method. As reviewed in Section 2, researchers in the past have repeatedly mentioned this to be a key problem facing the fault detection of HVAC. In contrast, the proposed dynamic threshold method immediately identified the fault as soon as it entered the system in mid-April, as shown in Fig. 16. In addition, the proposed dynamic threshold method does not suffer from the problems associated with the dynamic residual evaluation method by Navarro-Esbri et al. [21].

8. Conclusion and future work

In this paper, we presented a model, based on an XGBoost algorithm with a dynamic threshold, for automatic fault detection in real-time. XGBoost is used to generate predictive models for energy consumption using fault-free datasets. Subsequently, for fault detection, a novel dynamic threshold method is applied to the predictions generated by the XGBoost model using a dataset that consists of both fault and fault-free data. Applying Chebyshev's inequality, our proposed method dynamically adjusts the threshold value in terms of the moving average and the moving standard deviation. In this paper, the fault detection model has been applied to detect faults in a water cooled chiller, a packaged air conditioning unit, and a variable air volume distribution system. We have benchmarked the performance of our proposed dynamic threshold method relative to fixed thresholds and an existing dynamic residual evaluation method. We have shown that our proposed model is adequately robust to work well under highly skewed datasets. Our results suggest that the proposed dynamic threshold method has a fast reaction speed, i.e. there is no time delay between the occurrence of a fault and its detection. The proposed method also decreased the number of missed opportunities to detect faults without generating numerous false alarms.

The proposed dynamic threshold method has two free parameters that can be controlled in real-time to maintain a desirable balance between the number of false alarms and missed fault detection opportunities. Establishing desirable ranges of these free parameters for different building groups and HVAC systems could be an interesting topic for future research. The proposed fault detection model is expected to reduce energy wastage, prevent further deterioration of equipment, reduce equipment downtime due to major maintenance, and prolong the equipment life. This model can be used by the building operators to ensure that no energy is wasted in its operation that could excessively increase the energy cost of the building. This model can also be used by the maintenance team to locate and repair faults in its early stage before it can cause any major damage to the system.

Appendix A. Python source code

```
##########################################################################
# This is a comment! Comments start with the hash character, '#', and
# written in italics.
# Comments explain the purpose of a line or a block of code.
##########################################################################
# Import the necessary packages.
import numpy as np
import pandas as pd
from sklearn.model_selection import GridSearchCV, KFold, train_test_split
from sklearn.feature_selection import RFECV
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, make_scorer
##########################################################################
# Read data from file - Here we have the data in an excel file.
Data = pd.read_excel('Data_File_Name.xls')
Features = Data[Specify_Feature_Names]
Target = Data[Specify_Target_Name]
Target = Data[Specify_Label_Name]
X = Features.as_matrix()
y = Target.as_matrix()
True_Labels = Labels.as_matrix()
##########################################################################
# Divide the dataset into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5,
  shuffle=False)
TL_train, TL_Test = train_test_split(True_Labels, test_size=0.5, shuffle=
  False)
##########################################################################
# XGBoost modeling
np.random.seed(7)
kf = KFold(n_splits=5, shuffle=True)
scoring_param = make_scorer(mean_squared_error,greater_is_better=False)
rfecv = RFECV(estimator=XGBRegressor(n_jobs=-1), step=1, cv=kf, scoring=
  scoring_param)
FS_model = rfecv.fit(X_train, y_train)
ranks = FS_model.ranking_
FN =[]
for i in range(len(ranks)):
  if ranks[i] == 1:
    FN.append(Feature_Names[i])
X = Energy_Data[FN].as_matrix()
X_train_transformed, X_test_transformed = train_test_split(X, test_size
  =0.5, shuffle=False)
MD = [int(i) for i in np.linspace(1,20,num=5)]
LR = np.linspace(0.001,0.1,num=5)
NE = [int(i) for i in np.linspace(100,1000,num=5)]
p_grid = dict()
p_grid = dict(max_depth = MD, learning_rate = LR, n_estimators = NE)
model = GridSearchCV(estimator = XGBRegressor(n_jobs=-1), param_grid =
  p_grid, scoring = scoring_param, cv = kf)
model.fit(X_train_transformed, y_train)
Y_Test_Pred = model.predict(X_test_transformed)
# Applying the dynamic threshold method
Labels, Threshold = calc_dyn_threshold(y_test, Y_Test_Pred, 2, 2)
##########################################################################
```

Listing 1. XGBoost model development in Python.

```
##########################################################################
def calc_dyn_threshold(A, P, I, N):
  threshold = np.zeros(I-1)
  threshold[0:(I-1)] = P[0:(I-1)]
  labels = np.zeros(I-1)
  for k in np.arange(I,len(P)+1):
    mu = np.mean(P[(k-I):k])
    sigma = np.std(P[(k-I):k])
    T = mu + N*sigma
    threshold = np.append(threshold,T)
    if (A[k-1] > threshold[k-1]) :
      labels = np.append(labels,1)
    else:
      labels = np.append(labels,0)
  return labels, threshold
##########################################################################
```

Listing 2. Implementing the dynamic threshold method in Python.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.enbuild.2018.12.032.

References

[1] M. Basarkar, X. Pang, L. Wang, P. Haves, T. Hong, Modeling and simulation of HVAC faults in energyplus, in: Proceedings of Building Simulation, IBPSA, 2011, p. 28972903.

[2] A. Beghi, R. Brignoli, L. Cecchinato, G. Menegazzo, M. Rampazzo, F. Simmini, Data-driven fault detection and diagnosis for HVAC water chillers, Control Eng. Prac. 53 (2016) 79–91.

[3] C. Bergmeir, J.M. Benítez, On the use of cross-validation for time series predictor evaluation, Inf. Sci. 191 (2012) 192–213.

[4] M. Bonvini, M.D. Sohn, J. Granderson, M. Wetter, M.A. Piette, Robust on-line fault detection diagnosis for HVAC components based on nonlinear state estimation techniques, Appl. Energy 124 (2014) 156–166.

[5] D. Chakraborty, H. Elzarka, Advanced machine learning techniques for building performance simulation: a comparative analysis, J. Build. Perform. Simul. 0 (0) (2018) 1–15, doi:10.1080/19401493.2018.1498538.

[6] T. Chen, Introduction to Boosted Trees, University of Washing Computer Science. University of Washington vol. 22 (2014).

[7] T. Chen, T. He, Higgs boson discovery with boosted trees, in: NIPS 2014 Workshop on High-energy Physics and Machine Learning, 2015, pp. 69–80.

[8] H. Cheung, J.E. Braun, Development of Fault Models for Hybrid Fault Detection and Diagnostics Algorithm: October 1, 2014 – May 5, 2015, Technical Report, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2015.

[9] D.A. Cieslak, T.R. Hoens, N.V. Chawla, W.P. Kegelmeyer, Hellinger distance decision trees are robust and skew-insensitive, Data Min. Knowl. Discov. 24 (1) (2012) 136–158.

[10] A. Foucquier, S. Robert, F. Suard, L. Stéphan, A. Jay, State of the art in building modelling and energy performances prediction: A review, Renew. Sustain. Energy Rev. 23 (2013) 272–288.

[11] S. Frank, M. Heaney, X. Jin, J. Robertson, H. Cheung, R. Elmore, G. Henze, Hybrid model-based and data-driven fault detection and diagnostics for commercial buildings, Proceedings of the 2016 ACEEE Summer Study on Energy Efficiency in Building (2016). 12–1

[12] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, second ed., Springer-Verlag, New York, 2009.

[13] J. Hyvarinen, S. Karki (Eds.), Building optimization and fault diagnosis source book, IEA Annex 25, 1996.

[14] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning, vol. 6, Springer, 2013.

[15] W. Kim, S. Katipamula, A review of fault detection and diagnostics methods for building systems, Sci. Technol. Built Environ. 24 (1) (2018) 3–21.

[16] F. Magoulès, H.-x. Zhao, D. Elizondo, Development of an RDP neural network for building energy consumption fault detection and diagnosis, Energy Build. 62 (2013) 133–138.

[17] S. Marsland, Machine Learning: An Algorithmic Perspective, CRC Press, 2009.

[18] F. Martínez-Álvarez, A. Troncoso, G. Asencio-Cortés, J.C. Riquelme, A survey on data mining techniques applied to electricity-related time series forecasting, Energies 8 (11) (2015) 13162–13193.

[19] M. Najafi, Fault detection and diagnosis in building HVAC systems (2010).

[20] A. Natekin, A. Knoll, Gradient boosting machines, a tutorial, Front. Neurorob. 7 (2013).

[21] J. Navarro-Esbri, E. Torrella, R. Cabello, A vapour compression chiller fault detection technique based on adaptative algorithms. application to on-line refrigerant leakage detection, Int. J. Refrig. 29 (5) (2006) 716–723.

[22] X. Pang, M. Wetter, P. Bhattacharya, P. Haves, A framework for simulation-based real-time whole building performance assessment, Build. Environ. 54 (2012) 100–108.

[23] T.A. Reddy, Automated fault detection and diagnosis for HVAC&R systems: functional description and lessons learnt, in: ASME 2008 2nd International Conference on Energy Sustainability collocated with the Heat Transfer, Fluids Engineering, and 3rd Energy Nanotechnology Conferences, American Society of Mechanical Engineers, 2008, pp. 589–599.

[24] J.E. Seem, Using intelligent data analysis to detect abnormal energy consumption in buildings, Energy Build. 39 (1) (2007) 52–58.

[25] S. Wang, J. Cui, Sensor-fault detection, diagnosis and estimation for centrifugal chiller systems using principal-component analysis method, Appl Energy 82 (3) (2005) 197–213.

[26] Z. Wang, R.S. Srinivasan, A review of artificial intelligence based building energy use prediction: contrasting the capabilities of single and ensemble prediction models, Renew. Sustain. Energy Rev. 75 (2017) 796–808.

[27] S. Wu, J.-Q. Sun, Cross-level fault detection and diagnosis of building HVAC systems, Build. Environ. 46 (8) (2011) 1558–1566.

[28] H.-X. Zhao, F. Magoulès, Feature selection for predicting building energy consumption based on statistical learning method, J. Alg. Comput. Technol. 6 (1) (2012) 59–77.