2017

An ensemble learning framework for anomaly detection in building energy consumption

Daniel B. Araya
Western University, danberara@gmail.com

Katarina Grolinger
Western University, kgroling@uwo.ca

Hany F. ElYamany
Western University, helyama@uwo.ca

Miriam AM Capretz
Western University

Girma T. Bitsuamlak
Western University

Energy and Buildings 144 (2017) 191–206

Contents lists available at ScienceDirect

Energy and Buildings

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

An ensemble learning framework for anomaly detection in building energy consumption

Daniel B. Araya [a], Katarina Grolinger [a], Hany F. ElYamany [a,c], Miriam A.M. Capretz [a,*], Girma Bitsuamlak [b]

[a] Department of Electrical and Computer Engineering, Western University, London, Ontario, Canada N6A 5B9
[b] Department of Civil and Environmental Engineering, Western University, London, Ontario, Canada N6A 5B9
[c] Department of Computer Science, Suez Canal University, Ismailia, Egypt

ARTICLE INFO

Article history:
Received 19 July 2016
Received in revised form 3 November 2016
Accepted 26 February 2017
Available online 10 March 2017

Keywords:
Anomaly detection
Ensemble learning
Autoencoder
Support vector regression
Random forest
Building energy consumption

ABSTRACT

During building operation, a significant amount of energy is wasted due to equipment and human-related faults. To reduce waste, today's smart buildings monitor energy usage with the aim of identifying abnormal consumption behaviour and notifying the building manager to implement appropriate energy-saving procedures. To this end, this research proposes a new pattern-based anomaly classifier, the *collective contextual anomaly detection using sliding window* (CCAD-SW) framework. The CCAD-SW framework identifies anomalous consumption patterns using overlapping sliding windows. To enhance the anomaly detection capacity of the CCAD-SW, this research also proposes the *ensemble anomaly detection* (EAD) framework. The EAD is a generic framework that combines several anomaly detection classifiers using majority voting. To ensure diversity of anomaly classifiers, the EAD is implemented by combining pattern-based (e.g., CCAD-SW) and prediction-based anomaly classifiers. The research was evaluated using real-world data provided by Powersmiths, located in Brampton, Ontario, Canada. Results show that the EAD framework improved the sensitivity of the CCAD-SW by 3.6% and reduced false alarm rate by 2.7%.

1. Introduction

Commercial and residential buildings account for roughly 60% of the world's electricity consumption [1]. During building operation, a significant portion of energy consumption may be wasted due to various equipment or human-related faults. By reducing building energy waste and enhancing building consumption efficiency, facilities can minimize utilities cost and reduce the associated negative impact on the environment. This can also help address the growing energy demand that the world faces today.

One promising approach to energy efficiency goals is to monitor building energy consumption with the aim of identifying abnormal consumption patterns. Once identified, this abnormal consumption behaviour can be reported to the building manager, who can subsequently perform appropriate energy-saving procedures. In recent years, with the proliferation of sensor devices, monitoring building consumption behaviour for anomaly detection purposes has become easier.

Anomaly detection refers to the process of detecting abnormal events that do not conform to expected patterns [2]. Broadly, depending on their types, anomalies can be classified as point, contextual or collective anomalies [2]. If a data instance is anomalous compared to the rest of the data, then it is referred to as a *point anomaly*. For instance, a daily lighting energy consumption value might be anomalous compared to previously recorded daily values. If a data instance is normal in one context but anomalous in another, then it is referred to as a *contextual anomaly*. For instance, an hourly school lighting consumption value might be anomalous on weekends when there are no classes, but not on weekdays when there are classes. If a group or collection of related data instances is anomalous in comparison to the rest of the dataset, then it is referred to as a *collective anomaly*. Individually, these values might not be anomalous, but collectively they represent an anomalous occurrence. For instance, the individual values of a daily profile of heating, ventilating, and air conditioning (HVAC) consumption data recorded every hour might be normal compared to previous recorded values, but collectively, the daily profile might represent an anomalous consumption pattern.

* Corresponding author.
E-mail addresses: dberhane@uwo.ca (D.B. Araya), kgroling@uwo.ca (K. Grolinger), helyama@uwo.ca (H.F. ElYamany), mcapretz@uwo.ca (M.A.M. Capretz), gbitsuam@uwo.ca (G. Bitsuamlak).

http://dx.doi.org/10.1016/j.enbuild.2017.02.058

Nomenclature

A	learner algorithm
AC	anomaly classifier
AUC	area under the curve
BAS	building automation system
C	contextual features
$CCAD$	collective contextual anomaly detection
$CCAD-SW$	collective contextual anomaly detection using sliding window
D	real dataset
D_{train}	real training dataset
EAD	ensemble anomaly detection
EPE	ensemble performance evaluator
err_neg	test output of normal dataset
err_pos	test output of anomalous dataset
err_value	test output of new sensor dataset
EVD	eigenvalue decomposition
FP	false positive
FPR	false positive rate
G	generated features
$HVAC$	heating ventilating and air conditioning
IQR	interquartile range
L	learning model
MSE	mean square error
MT	Model tester
N	real testing dataset
$OETD$	optimal ensemble threshold determinator
Opt_E_{thresh}	optimal ensemble threshold
P	artificial anomalous dataset
$pAUC$	partial area under the curve
$pAUC_c$	standardized partial area under the curve
PCA	principal component analysis
PSO	particle swarm optimization
$Q1$	first quartile
$Q2$	second quartile
$Q3$	third quartile
R	number of learning rounds
RBF	radial basis function
RF	random forest
ROC	receiver operating characteristics
S	a set of unique error values
$S_n - S_1$	difference between first and last sliding window data values
SVD	singular value decomposition
SVM	support vector machines
SVR	support vector regression
SW	sliding window
TN	true negative
TNR	true negative rate
TP	true positive
TPR	true positive rate

One of the problems of existing collective anomaly detection approaches is that there is little concern for the context of the anomaly under consideration. For example, a daily HVAC consumption pattern might be anomalous in winter, but not in summer. An important application of collective contextual anomaly detection is a *building automation system* (BAS), a built-in control system which is nowadays present in most modern buildings (smart buildings [3]). The BAS enables building managers to oversee the energy efficiency aspects of a building by providing early detection and notification of abnormal consumption behaviour. Identifying collective contextual anomalies of a facility at various granularities can be a useful tool for short-term energy saving and disaster mitigation, as well as for meeting long-term energy efficiency targets. For instance, identifying hourly collective contextual anomalies in HVAC consumption can be useful for achieving short-term energy-saving goals. Identifying anomalies in annual HVAC consumption profile is more useful for long-term energy efficiency plans such as replacing energy-wasting equipment, analyzing the cost of services over a long period of time, and planning specific energy-saving targets. For this reason, this research proposes a new pattern-based anomaly classifier, the *collective contextual anomaly detection using sliding window* (CCAD-SW) framework. The CCAD-SW framework uses overlapping sliding windows to improve significantly the anomaly detection performance of the CCAD [4] framework. In addition, it identifies anomalies earlier and substantially reduces false positives. To enhance the anomaly detection capacity of the CCAD-SW, this research also proposes the *ensemble anomaly detection* (EAD) framework. The EAD is a generic framework that combines several anomaly detection classifiers using majority voting. In this study, it is assumed that each anomaly classifier has equal weight. To ensure diversity of anomaly classifiers, the EAD framework is implemented by combining pattern- and prediction-based anomaly classifiers.

In this study, the EAD framework combines the CCAD-SW, which is implemented using autoencoder, with two prediction-based anomaly classifiers that are implemented using the machine learning algorithms support vector regression and random forest. More importantly, the EAD framework identifies an ensemble threshold that provides an anomaly classifier with optimal anomaly detection performance and minimum false positives. Results show that the EAD performs better than the individual anomaly detection classifiers.

The remaining sections of the paper are organized as follows: Section 2 provides the background information and Section 3 describes related work. Sections 4 and 5 outline the CCAD-SW and EAD frameworks proposed in this research. Section 6 presents the experimental results and discussion, and finally Section 7 concludes the paper.

2. Background information

This section first presents an overview of the learning approach used in this study, i.e., the ensemble learning approach. Moreover, it describes the machine learning algorithms used in this research: autoencoder, PCA, support vector regression, and random forest. In addition, the performance metrics used to compare anomaly detection classifiers are described.

2.1. Ensemble learning

Ensemble learning is a machine learning approach that solves a problem by training multiple learners. Unlike ordinary machine learning approaches in which a single hypothesis is learned from training data, ensemble approaches attempt to build a set of hypotheses and combine them to build a new hypothesis [5]. Previous studies show that an ensemble often performs better than the individual learners, also known as *base learners* of the ensemble [6].

Most ensemble approaches rely on a single *base learning algorithm* to produce what are referred to as *homogeneous* base learners. However, some approaches use multiple learning algorithms and are referred to as *heterogeneous* learners [5]. The primary objective of ensemble learning is to improve the performance of a model by combining multiple learners.

Normally, ensembles are constructed in two steps. Initially, several base learners are built, and then these learners are combined. Several combination techniques are used. For anomaly

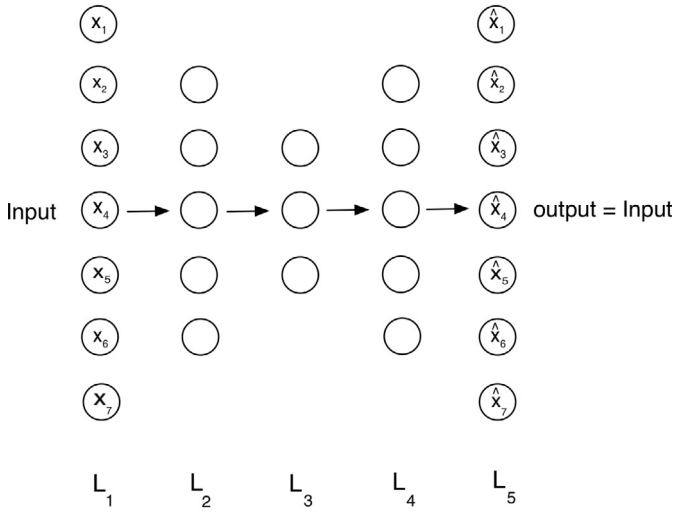

Fig. 1. Autoencoder.

classification, *majority voting* [7] is a widely used combination technique [5]. In *majority voting*, used in this study, the final decision is made based on the agreement of more than half of the base learners.

2.2. Principal component analysis

Principal component analysis (PCA) is a multivariate statistical technique that is widely used for dimensionality reduction. PCA looks for new orthogonal components (eigenvectors) that explain the largest part of the data variation by providing a measure of the dependency that exists among a set of inter-correlated features [8]. PCA is based on the Eigenvalue Decomposition (EVD) [9] of correlation or covariance matrices or the singular value decomposition (SVD) of real data matrices. The implementation in this paper is based on SVD. Compared to EVD, SVD is more stable, robust and precise and does not require calculating the correlation or covariance matrix [10].

2.3. Autoencoder

An autoencoder [11] is an unsupervised artificial neural network that is trained to reproduce input vectors as output vectors [12]. Fig. 1 represents an autoencoder; in this figure, Layer L_1 is the input layer, Layers L_2, L_3 and L_4 are the hidden layers, and Layer L_5 is the output layer. During training, the input dataset $\{x_1, x_2, \ldots, x_m\}$ is compressed through three hidden layers into a lower-dimensional latent subspace to reproduce the output $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m\}$. Assuming that each data sample $x_i \in \mathbb{R}^D$, is represented by a vector of D different variables, the training objective is to construct the outputs by minimizing the reconstruction error in Eq. (1).

$$Err(i) = \sqrt{\sum_{d=1}^{D} (x_d(i) - \hat{x}_d(i))^2} \tag{1}$$

The activation of unit k in layer l is given by Eq. (2). The sum is calculated over all neurons j in the $(l-1)$st layer:

$$a_k^{(l)} = f\left(\sum_j W_{kj}^{(l-1)} a_j^{(l-1)} + b_k^{(1)}\right) \tag{2}$$

where **b** and **W** are the bias and weight parameters, respectively. In this study, the hyperbolic tangent is used as the autoencoder's activation function.

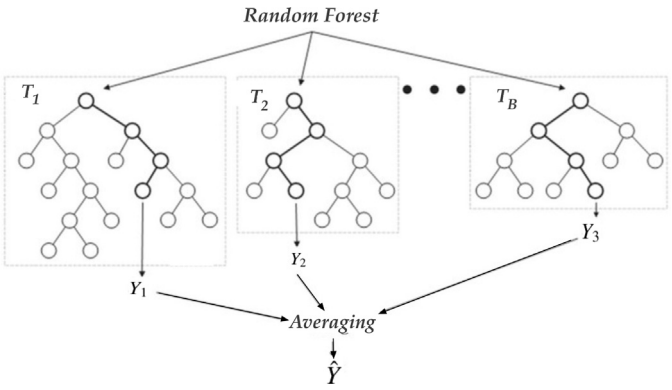

Fig. 2. Random forest structure [16].

2.4. Random forest

The *random forest* (RF), proposed by Breiman [13] is a widely used ensemble learning approach for both classification and regression problems [14,15]. RF operates by constructing a multitude of decision trees during training and outputting the class that is the mode of the classes output by individual trees. An RF is composed of an ensemble of B trees $\{T_1(F), \ldots, T_B(F)\}$, where $F = \{f_1, \ldots, f_n\}$ is an n-dimensional feature vector. The ensemble produces B outputs $\{\hat{Y}_1 = T_1(F), \ldots, \hat{Y}_B = T_B(F)\}$ where $\hat{Y}_a$, $a = 1, \ldots, B$, is the value predicted by the ath tree. The final prediction, $\hat{Y}$, is made by averaging the predicted values of each tree as shown in Fig. 2.

2.5. Support vector regression

Support vector machines (SVM) [17–20] are supervised learning models used for regression and classification purposes. *Support vector regression* (SVR) [21], a version of SVM used for regression, achieves a high degree of generalization, which implies that the model performs very accurately on previously unseen data. The support vectors in SVR are identified from the rest of the training samples by a discriminating loss function that does not penalize residuals less than a tolerance value ε. As a result, for a given hypotheses and ε, the observations constrained to the ε tube bounding the hypothesis, as illustrated in Fig. 3 do not affect the predictions.

Given a training dataset $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, suppose that y is modelled as a function of the input variables x. In SVR, the relationship between x and y is approximated as:

$$y = \omega \cdot \Psi(x) + b, \tag{3}$$

where Ψ is a non-linear kernel function that maps from the input space x to a higher-dimensional feature space. The coefficients ω and b are obtained by minimizing the following function:

$$\text{minimize} \quad \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{4}$$

$$\text{subject to} \quad \begin{aligned} & y_i - \omega \cdot x_i - b \le \varepsilon + \xi_i \\ & \omega \cdot x_i + b - y_i \le \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \ge 0 \end{aligned}$$

To achieve good generalization, the weight ω needs to be as flat as possible. The residuals beyond the ε are captured by the terms ξ_i, ξ_i^*, and the cost C is the regularization parameter that determines

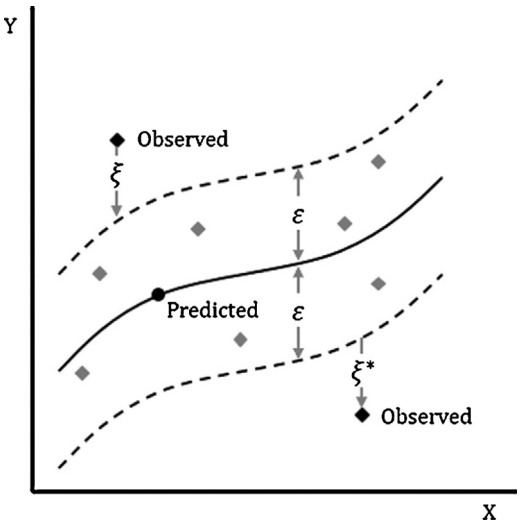

Fig. 3. Parameters of nonlinear SVR, adapted from [22].

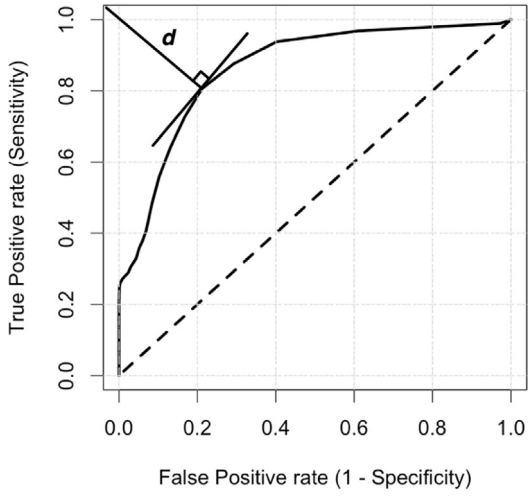

Fig. 4. Optimal threshold determination [23].

the penalty for errors greater than ε. This work uses the radial basis function (RBF) because it is a widely used kernel that is efficient to compute. Moreover the kernel has only one parameter that needs to be determined. The RBF kernel is given by:

$$K(x, \hat{x}) = \exp(-\gamma \|x - \hat{x}\|^2), \tag{5}$$

where the kernel parameter γ expresses the influence for each data point.

2.6. Performance metrics

The metrics used to analyse the anomaly detection frameworks proposed in this research are the sensitivity or true positive rate (TPR), which measures a model's capacity to identify anomalous data, and the specificity or true negative rate (TNR), which measures a model's capacity to identify normal data. The TPR and TNR are evaluated using Eqs. (6) and (7), respectively.

$$\text{Sensitivity} = \text{TPR} = \frac{\text{TP}}{\text{P}} \tag{6}$$

$$\text{Specificity} = \text{TNR} = \frac{\text{TN}}{\text{N}} \tag{7}$$

where true positive (TP) is the number of anomalous consumption patterns that are correctly identified as anomalous, true negative (TN) is the number of normal consumption patterns that are correctly identified as normal, P is the total number of positive instances and N is the total number of negative instances.

In machine learning and data mining studies, the receiver operating characteristics (ROC) curve [23,24] is widely used to analyze and visualize classifier performance. The ROC curve, as illustrated in Fig. 4, is a plot in a unit square of the true positive rate (TPR) versus the false positive rate (FPR). The FPR refers to the rate of false alarms and is given by $(1 - \text{TNR})$.

Using the ROC curve, the performance of the anomaly detection model for all possible threshold values can be evaluated, and the threshold value that optimizes specificity as well as sensitivity can be identified. Various threshold determination approaches have been examined [23,24]. In this research, the rates of both anomaly detection and false alarms are assumed to have equal weight. Based on this approach and noting that the point (0,1) on the ROC curve is the ideal point (100% anomaly detection and 0% false alarms), the shortest distance d from a point on the curve to point (0,1) as shown in Fig. 4 can be evaluated using Eq. (8) [24]:

$$d^2 = (1 - \text{sensitivity})^2 + (1 - \text{specificity})^2, \tag{8}$$

where d is the shortest distance from a point on the ROC curve to the point (0,1). This distance determines the threshold value that optimizes both the sensitivity and specificity of the anomaly detection framework.

The area under the curve (AUC) is an effective measure of accuracy which determines the overall inherent capacity of an anomaly classifier to differentiate between normal and anomalous data instances. The maximum (AUC = 1), represents a perfect anomaly classifier. Generally, an AUC closer to 1 indicates better anomaly detection performance [23].

The partial area under the curve (pAUC), defined as the area within a range of false positives or true positives [23] is a performance metric more suitable for comparing classifiers whose ROC curves cross. An anomaly classifier "A" might have better sensitivity than anomaly classifier "B" in a specific specificity range, while anomaly classifier "B" might perform better than anomaly classifier "A" in another sensitivity range. Hence instead of using AUC, which gives an overall combined metric, identifying a specific range and using pAUC provides a better comparison measure. By standardizing pAUC, regardless of the partial region defined, the value of pAUC is always 1 for a perfect ROC curve and 0.5 for a random ROC curve. pAUC can be standardized using Eq. (9) [25]:

$$pAUC_s = \frac{1}{2}\left(1 + \frac{pAUC - \min}{\max - \min}\right) \tag{9}$$

where pAUC is the partial area under the curve for the selected FPR or TPR range, *min* is the pAUC over the same region of the diagonal ROC curve, *max* is the pAUC over the same region of the perfect ROC curve, and $pAUC_s$ is the standardized partial area.

The trapezoid rule is typically used to evaluate the area under a curve by approximating the region under the curve as a trapezoid and calculating its area.

3. Related work

Anomaly detection is an important problem that has been widely researched in various application areas such as fraud detection, intrusion detection and eco-system monitoring. In this study, related work is divided into two subsections. The first subsection presents an analysis of anomaly detection studies in the building

energy domain, and the second focusses on studies that use ensemble learning to identify abnormal behaviour. To the best of our knowledge, no previous study has explored ensemble approaches for anomaly detection in the building energy domain, and therefore, in the second subsection, related ensemble approaches for anomaly detection in other domains are considered.

3.1. Anomaly detection

Several previous studies used historical building energy data to identify point anomalies [26–30]. Chou and Telaga [26] proposed a two-stage real-time point anomaly detection system. In their work, consumption value was predicted one-step-ahead, and anomalies were identified by comparing whether or not the reading deviated significantly from the predicted value. Janetzko et al. [27] outlined an unsupervised anomaly detection system based on a time-weighted prediction that used historical power consumption data to identify point anomalies. Wrinch et al. [28] detected anomalies in periodic building operations by analyzing electrical demand data using a weekly moving time window in the frequency domain. However, the techniques outlined assumed constant data periodicity which caused many false positives [31].

Hill et al. [29] proposed a data-driven modelling approach using one-step-ahead prediction to identify point anomalies. However, their study considered only sequential data and did not take contextual features into account. Considering only historical data to identify anomalies would likely create false positives when contextual information such as season and day of week was included in the anomaly detection process. Bellala et al. [30] proposed an unsupervised cluster-based algorithm that identified anomalous points based on a low-dimensional embedding of power data.

In contrast to these studies [26–30], our research introduces context to the anomaly detection process because a value might be anomalous in one context but not in another. The studies just mentioned considered only point anomalies. However, if a set of values is considered, each value might not be anomalous, but collectively, this set of values might represent anomalous behaviour. Hence, using a sliding window approach, this research identifies contextual anomalies in collective building energy consumption data.

Other studies have considered contextual attributes or behaviours to identify anomalies in a specific context. Arjunan et al. [32] proposed a multi-user energy consumption monitoring and anomaly detection technique that used partial contextual information. Besides partially available contextual features, they used the concept of neighbourhood to provide a more relevant context for anomaly detection. Zhang et al. [33] used historical data as well as weather and appliance data to compare clustering, entropy, and regression techniques for identifying unusually low energy consumption patterns in a household. Nevertheless, the model presented was static and could not adapt to changes in facility consumption behaviour, for instance, new equipment or a change in building functionality.

Zorita et al. [34] presented a methodology that used multivariate techniques to construct a model using climatic data, building construction characteristics and activities performed in the building. Ploennigs et al. [35] presented a diagnostic technique that used a building's hierarchy of sub-meters. By analyzing historical data, they identified how abnormal daily energy use is influenced by building equipment. Jiang et al. [36] presented a three-stage real-time collective contextual anomaly detection method over multiple data streams. However, the approach described identifies anomalies in the context of data streams, whereas the proposed CCAD-SW framework is flexible with regard to new contextual features.

Peña et al. [37] proposed a rule-based system developed using data mining techniques to solve energy inefficiency detection problem in smart buildings. A set of rules was developed using knowledge extracted from sensor data and contextual information. Finally, the results of the rules and energy efficiency indicators were used to construct a decision support system that identifies anomalies. Capozzoli et al. [38] proposed a methodology that uses statistical pattern recognition techniques and an artificial neural network ensemble to find anomalies in near real-time. Hayes and Capretz [39] identified sensor data anomalies using a combination of point and contextual anomaly detection approaches.

These studies [32–39] identified contextual point anomalies. In contrast, our work focusses on a set of consecutive values to identify collective anomalies contextually. Hence, by using a sliding window approach, this study identifies contextual anomalies in collective sensor data. This helps to analyze building consumption profiles contextually over a specific sliding window instead of at a specific point in time. Moreover, by varying the sliding window size, collective contextual anomaly detection can be advantageous in a number of situations. These can range from short-term energy savings and potential disaster prevention objectives to medium- and long-term building energy profile analyses which can be useful in planning long-term energy-saving targets. In addition, this research identifies an anomaly detection framework that optimizes both anomaly detection and false positive rates.

3.2. Ensemble learning for anomaly detection

Several studies have focussed on enhancing classification accuracy using an ensemble of classifiers. Some used homogeneous classifiers [40,41], whereas others used heterogeneous classifiers [42,43] or a combination of both [44–46].

Using an ensemble obtained by training multiple C4.5 classifiers, Cabrera et al. [40] evaluated these classifiers on a MANET network for two types of attacks: Denial-of-Service and Black Hole attacks.

Didaci et al. [41] proposed a pattern recognition approach to network intrusion detection based on ensemble learning paradigms. The authors categorized feature spaces and trained a neural network with separate features to create several classifiers. Subsequently, these classifiers independently performed attack detection, and their results were later combined to produce the final decision.

Folino et al. [42] introduced an architecture for a distributed intrusion detection by using ensembles that specialized in detecting particular types of attack. Similarly to our framework, the authors used different algorithms with the same dataset to build different classifiers or models. Zhao et al. [43] proposed ensemble methods to enhance the anomaly detection accuracy on unsupervised data using density-based and rank-based algorithms. Besides using these independent learners, the authors also considered sequential methods for ensemble learning in which one detection method is followed by another.

Amozegar and Khorasani [44] proposed an ensemble of dynamic neural network identifiers for Fault Detection and Isolation (FDI) in gas turbine engines. The authors first built three individual dynamic neural-network architectures, then constructed three ensemble-based techniques and compared the performance of these models. Aburomman and Reaz [45] proposed an ensemble construction method that used particle swarm optimization (PSO)-generated weights to create an ensemble of classifiers for intrusion detection. Their work used a combination of homogeneous and heterogeneous classifiers. Shoemaker et al. [46] studied an ensemble voting method for anomaly detection in supervised learning using random forests and distance-based outliers partitioning. They demonstrated that this approach provided accuracy results similar to the same methods without partitioning.

To the best of our knowledge, no previous work has explored ensemble anomaly detection techniques in the building energy

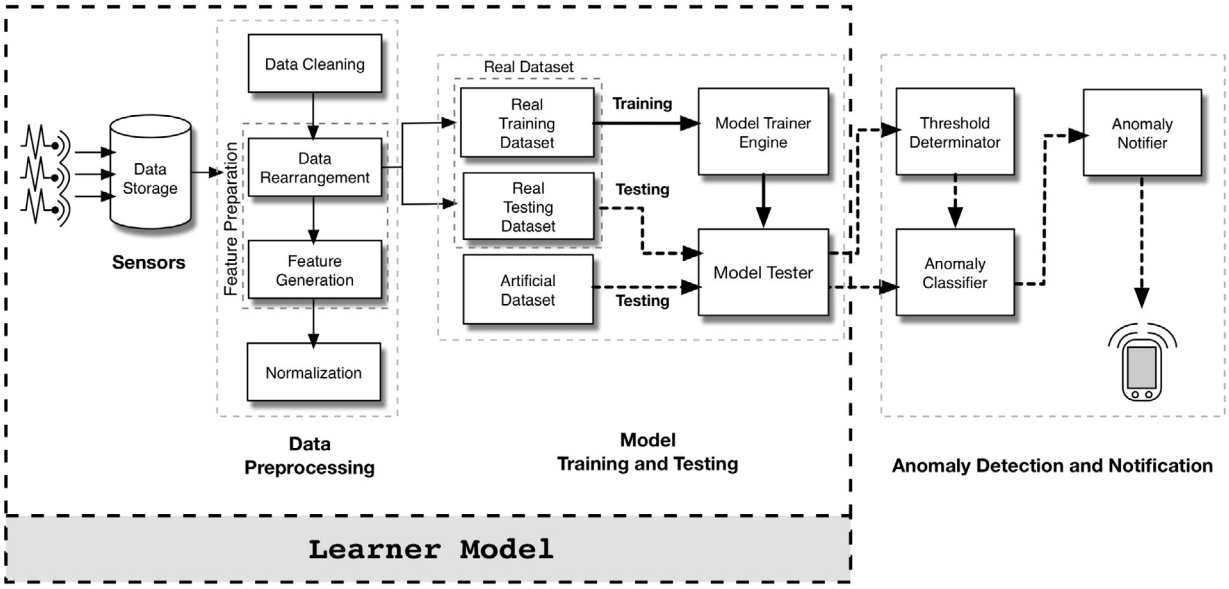

Fig. 5. Collective contextual anomaly detection using sliding window (CCAD-SW).

domain. Moreover, in contrast to the studies described above, [40–46], the EAD framework proposed in this research combines several learners and determines a combined threshold (ensemble threshold) value that yields an ensemble anomaly classifier with optimal anomaly detection performance.

4. Collective contextual anomaly detection using sliding window (CCAD-SW) framework

This research proposes the CCAD-SW framework illustrated in Fig. 5. The framework is an adaptation of the *collective contextual anomaly detection* (CCAD) framework [4]. The CCAD identifies collective contextual anomalies using a non-overlapping sliding window approach. As a result, the framework can identify anomalies only after a time t equal to the time length of a sliding window. For instance, if hourly sliding window sensor data are considered, anomalies can be identified after 1 h. Moreover, depending on the size of the sliding window, the delay can be a day, a week, and so on. This delay in anomaly detection and notification might not be suitable for services that require urgent attention (e.g., gas leaks).

By using overlapping sliding windows, the CCAD-SW addresses the shortcomings of the CCAD. As Fig. 6 shows, a sliding window is formed every time a new reading becomes available, which can be every second, every minute or whatever depending on the sensor measurement granularity. Moreover, by using flexible sliding window sizes, the CCAD-SW, accommodates both short-term urgent anomaly detection requirements and long-term building energy consumption profile analysis (monthly, annual, etc.) aimed at long-term energy-saving plans. The components of the CCAD-SW framework are described below.

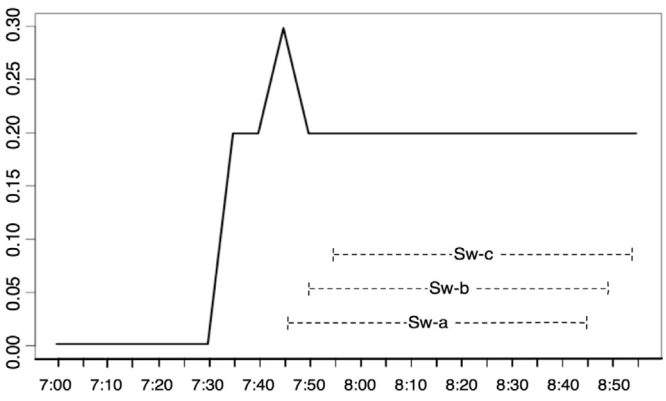

Fig. 6. CCAD-SW framework data rearrangement.

4.1. Data preprocessing

In this paper, "sensor data" refers to time-stamped consumption data recorded at regular time intervals. The dataset must be preprocessed to suit the learning algorithm used. The following sections describe the data preprocessing steps involved.

4.1.1. Data cleaning

To avoid the undesirable impact of missing and noisy real-world data, sliding windows with missing and noisy data have been removed from the dataset. In this study consumption values less than zero are considered noisy.

4.1.2. Feature preparation

After the dataset has been cleaned, it is reorganized and new features generated. The feature preparation component has two sub-steps: *data rearrangement* and *feature generation*, which are described below.

Data rearrangement: this involves rearranging the sensor data by representing each input instance using sliding window data instead of a single consumption value.

Table 1 shows a sample input dataset of the hourly sliding window data "Sw-a", "Sw-b", and "Sw-c", shown in Fig. 6. In the Table 1, columns "5", "10", . . ., "55", "60" represent an hourly consumption reading recorded every 5 min, and the consumption values in the first, second, and third rows represent the sliding window data "Sw-a", "Sw-b", and "Sw-c", respectively. For instance, "Sw-a" represents an hourly sliding window sensor dataset from 7:45 to 8:45. The next row in the sliding window represents "Sw-b" which is

Table 1
Sample preprocessed dataset for CCAD-SW.

. . .	Day of week	Hour	Minute	5	10	. . .	55	60	. . .
. . .	2	8	45	0.3	0.2	. . .	0.2	0.2	. . .
. . .	2	8	50	0.2	0.2	. . .	0.2	0.2	. . .
. . .	2	8	55	0.2	0.2	. . .	0.2	0.2	. . .

constructed when the data reading at 8:50 becomes available. "Sw-b" represents an hourly sensor dataset from 7:50 to 8:50, and so on. Hence, for instance, for the sample sliding window data shown in Fig. 6, collective contextual anomalies are identified every 5 min.

Feature generation: This component involves introducing more features to the anomaly detection process. As already mentioned, by using overlapping sliding windows, the time difference between successive sliding windows is reduced. Hence, to accommodate for this shorter gap between sliding windows, the CCAD-SW framework, introduces the feature *minute*. The contextual features *day of year*, *season*, *month*, *day of week*, and *hour of day* are also introduced to the dataset. To ensure that the CCAD-SW framework uses information contained within a sensor data time-stamp, no other contextual attributes such as weather or occupancy have been used. The following generated features were also derived: the mean of sensor data values in each window ($\bar{x}$), the standard deviation of sensor data values in each window (s), the difference between last and first elements of a sliding window ($S_n - S_1$), first quartile ($Q1$), second quartile ($Q2$), third quartile ($Q3$), and Interquartile Range (IQR). A total of 25 features were selected; a description of all the features is provided in our previous work [4].

4.1.3. Normalization

In a dataset that has features with widely differing scales, the larger values might have more influence than the smaller ones. To give the features equal weight, the dataset was normalized by rescaling the features to range to lie in [0 1] [47] using Eq. (10):

$$\acute{x} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{10}$$

where x is the original value and $\acute{x}$ is the normalized value.

4.2. Model training and testing

This section describes the datasets used for model training and testing as well as the training and testing engines. The CCAD-SW framework is based on the assumption that historical sensor data are for the most part normal. Based on this assumption, initially, a subset at the end of this dataset was set aside to assess the specificity of the model. This dataset was not part of model selection (parameter tuning) or model training. Subsequently, the remaining dataset (the "real dataset") was split into real training and real testing datasets. Moreover, an anomalous dataset was artificially generated. These datasets are described below:

- *Real dataset* (D) refers to the historical dataset.
  - *Real training dataset* (D_{train}): A subset of the historical dataset that is used to train a model to learn normal consumption behaviour.
  - *Real testing dataset* (N): A subset of the historical dataset used to test the specificity of a model.
- *Artificial dataset* (P): An anomalous dataset generated artificially in order to assess the sensitivity of a model. These anomalous data were based on the distribution of historical consumption data. Based on the observed historical data, consumption behaviour can be classified into two types of periods: low-activity and high-activity. A high-activity period has comparatively higher energy consumption, whereas a low-activity period has either low or zero consumption values. Artificial anomalous data were generated taking these two activity periods into consideration.

The training method used in this study was a form of *Bootstrap Aggregating* or *bagging* [48], which is a commonly used ensemble modelling technique [49,50]. *Bagging* is designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression.

Initially, from the historical dataset, a 10% subset at the end was set aside for final model testing. This dataset was used as a final step to test the specificity of the model and was not part of the model training or validation process. Moreover, the same size of anomalous dataset was generated artificially and used to test the sensitivity of the model.

Let the remaining part of the historical dataset be denoted by D. From D, again, a 10% subset (N) was set aside for model validation. From the remaining 90% dataset, a random 80% (D_{train}) was selected with replacement. The subset D_{train} was used to train a model. Subsequently, this model was tested using (N) as well as the artificially generated anomalous dataset (P). This training and testing process was repeated k times; most published papers suggest a k value between 25 and 50 [50,48]. After k repeated training and testing cycles, the average of the test results was evaluated, i.e., the average test result for both normal and anomalous test datasets.

Algorithm 1 describes the CCAD-SW illustrated in Fig. 5. In the following sections, each component of the figure will be described and the descriptions referred to the corresponding lines in Algorithm 1.

The algorithm starts with a loop in line 1 which represents the learning rounds R of the tasks from lines 2 to 9. Inside this loop, initially, bootstrap training samples, D_{train} are generated from the historical dataset D (line 2).

Algorithm 1. CCAD-SW framework algorithm.

Input : *New Sensor Value* (V), *Real Training Dataset* (D_{train}), *Real Testing Dataset* (N), *Artificial Dataset* (P), *Contextual Features* (C), *Generated Features* (G), *Number of learning rounds* (R), *Learner Algorithm* (A)

Output: *Notification*

```
/* Function: Learner_Model lines 1-8 */
1  for i ← 1 to R do
       /* Generate bootstrap samples */
2      D_train(i) ← Bootstrap (D);
3      normal_model ← ModelTrainer (A, SlidingWindow, D_train(i), C, G);
4      err_neg_i ← ModelTester (normal_model, SlidingWindow, N, C, G);
5      err_pos_i ← ModelTester (normal_model, SlidingWindow, P, C, G);
6  end
7  err_pos = (1/R) Σ_{i=1}^{R} err_pos_i;
8  err_neg = (1/R) Σ_{i=1}^{R} err_neg_i;
9  err_value ← ModelTester (normal_model, SlidingWindow, V, C, G);
10 thresholdValue ← ThresholdDeterminator(err_pos, err_neg);
11 A_c ← AnomalyClassifier(err_value, thresholdValue);
12 if IsAnomalous(A_c) then
13     return Notification = true;
14 else
15     return Notification = false;
16 end
```

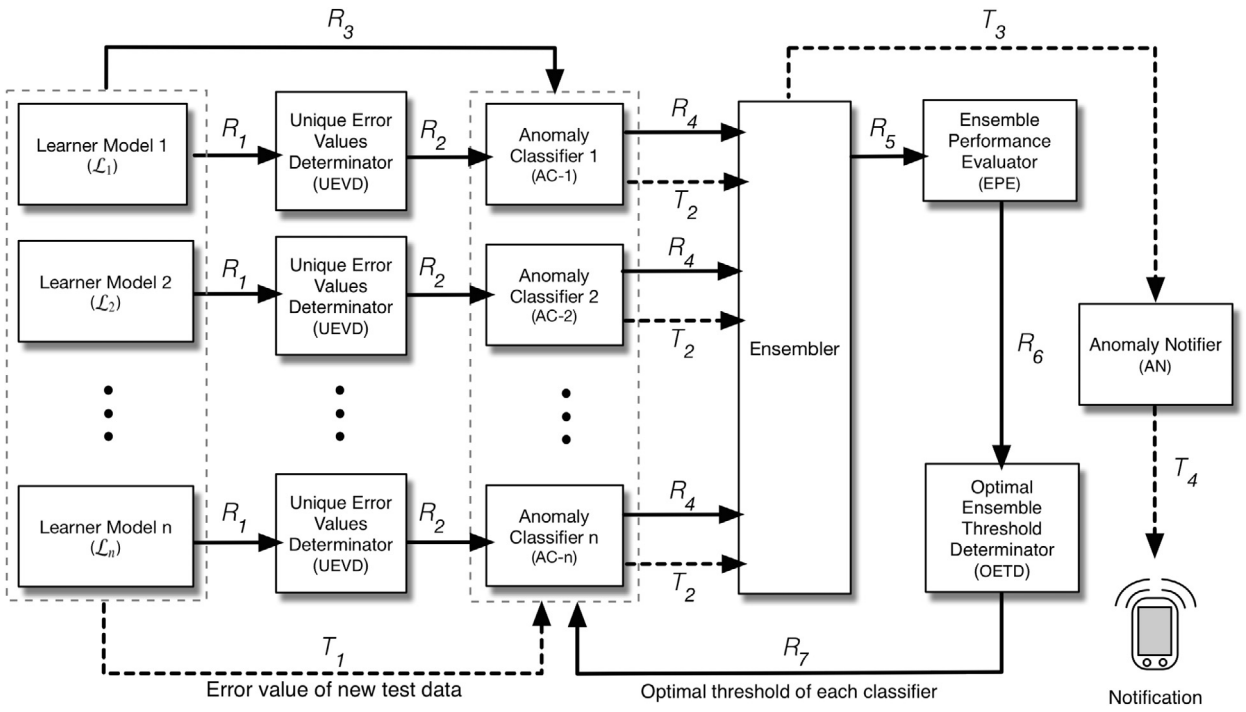

Fig. 7. Ensemble anomaly detection (EAD) framework.

4.2.1. Model trainer engine

The *Model trainer engine* in this study is a generic component that trains a pattern learning algorithm to reconstruct input data patterns. Autoencoder [11] was used in this research to capture the non-linear and complex pattern that exists between the contextual features and the sliding window of consumption data. Autoencoder provides non-linear dimensionality reduction giving the CCAD-SW framework computational efficiency gains [51] and improved classification accuracy [52] compared to other dimensionality reduction techniques such as PCA or Kernel PCA [12]. The *Model trainer engine* can be replaced by other pattern learning techniques.

In Algorithm 1, the `ModelTrainer` function is trained to recognize input data patterns consisting of historical sensor data (D_{train}), contextual features (C), and sensor data-generated features (G) (line 3).

4.2.2. Model tester

Once a model is trained using normal consumption patterns, the `ModelTester` function tests the model using the real testing dataset as well as the artificially generated anomalous dataset. The *Model tester* component tries to reconstruct the input dataset; the output of this component is a reconstruction error that measures how close the input data pattern is to the normal data pattern.

The `ModelTester` function uses the model trained in the *Model trainer engine* to reconstruct new instances of normal historical sensor data as well as artificially generated anomalous data. The reconstruction error array, *err_n eg*, which is the test output of the normal test data, as well as the reconstruction error array, *err_p os*, which is the test output of the artificially generated anomalous test data, are determined in lines 4 and 5, respectively.

After R learning rounds, the average of the test outputs of the positive and negative test results are evaluated in (line 8) and (line 9), respectively. In line 10, new sensor data are tested by the `ModelTester` function.

4.3. Anomaly detection and notification

The anomaly detection and notification section involves the threshold determination, anomaly detection, and notification steps described below.

4.3.1. Threshold determinator

The `ThresholdDeterminator` function (line 11) uses the test results of the *Model tester* component to evaluate a threshold value θ that optimizes the sensitivity and specificity of the CCAD-SW framework. In this component, the density distributions of *err_p os* and *err_n eg* are determined, and using the TP and TN values of every cut-off error value, the corresponding TPR and TNR ratios are evaluated using Eqs. (6) and (7), respectively.

The chosen threshold value determines the number of TN and TP captured. A lower threshold value yields a better anomaly detection rate while increasing the false positive rate. The ROC curve was used to determine the threshold that optimized these metrics.

4.3.2. Anomaly classifier and anomaly notifier

The reconstruction error values of new sensor data instances are determined using the trained model. These values are then compared with the threshold value θ, and the `AnomalyClassifier` function (line 12) classifies instances with a reconstruction error value greater than θ as anomalous and instances with an error value less than θ as normal. Anomalous values trigger the `notifier` function to raise an alarm that notifies the building manager, who then performs appropriate energy-saving procedures.

5. Ensemble anomaly detection (EAD) framework

To enhance the anomaly detection performance of the CCAD-SW, this research proposes the EAD framework. The EAD

Table 2
Sample dataset for prediction-based anomaly classifiers.

D. of year	Season	Month	D. of week	Hour	Min.	Consumption
142	2	5	2	8	45	2.5
142	2	5	2	8	50	2.4
142	2	5	2	8	55	2.4

framework will be described in the next few sections, but before delving into its details, this section briefly explains the motivation and reasoning behind its design.

The generalization ability of an ensemble is usually much stronger than that of a single learner [53]. One of the reasons is that a single learner might not capture enough information from the training data available. For instance, several learners might perform equally well on a given training dataset, but combining these learners might produce a better result. Another reason is that the search processes of the individual learners or base learners might not be perfect. For example, even if a unique best hypothesis exists, it might be difficult to achieve because running the base learners gives suboptimal hypotheses. Thus, ensembles can compensate for such imperfect search processes [53].

Empirical results show that, ensembles tend to have better results when there is a significant diversity among the models [54]. One way of introducing model diversity is to use models that are based on different algorithms; another is to use models that are based on the same algorithm, but trained with different subsets of the dataset. To address data shortage issues, this research focusses on the former approach because the latter requires a sizeable dataset.

Therefore, this paper proposes the EAD framework shown in Fig. 7. The EAD is a generic framework that combines several heterogeneous or homogeneous learners. The framework combines anomaly detection learners that rely on pattern and/or prediction based approaches. Moreover, by evaluating combined threshold (ensemble threshold) values, the EAD framework identifies an ensemble anomaly classifier that yields optimal sensitivity and specificity.

In this study, the prediction-based anomaly classifiers determine whether or not the sum of a sliding windows dataset is anomalous. For instance, from the sliding window dataset illustrated in Fig. 6, the CCAD-SW determines whether or not pattern of the sliding window "Sw-a" is anomalous, while the prediction-based learners determine whether or not the sum of the sliding window data "Sw-a" is anomalous. Both of these anomalous classification approaches deal with the same sliding window, but identify anomalous behaviour differently.

The EAD framework, as illustrated in Fig. 7 and outlined in Algorithm 2, has training and testing flow paths. These components with their associated lines in Algorithm 2 are described in the following sections.

5.1. Training

The training flow of the EAD framework in Fig. 7 is represented by continuous lines and the letter "*R*". The components involved in the training are described in the following sections.

5.1.1. Learner model

As shown in Fig. 7, the EAD framework has several *Learner models*, of which two types are considered in this study: pattern-based (e.g., CCAD-SW) and prediction-based learner models. The objective of a *Learner model* is to perform the following tasks: preprocess an input dataset, train a model, test the model using previously unseen normal and anomalous datasets, and finally output these test results.

The use of a *Learner model* for pattern-based anomaly detection approach has been described in the CCAD-SW section. This section describes the use of a *Learner model* for prediction-based anomaly detection classifiers.

The *Learner model*, represented by the largest dashed box in Fig. 5, is a generic component of the EAD framework (Fig. 7). From Fig. 5, the main difference between the application of the *Learner model* to pattern-based anomaly classifiers and prediction-based anomaly classifiers is, in the *Data rearrangement*, *feature generation*, *Model trainer engine*, and *Model testing* components. In the *Data rearrangement*, the dataset is reorganized so that the sliding window data shown in Fig. 6 are represented by the sum of the consumption data of a sliding window. Table 2 shows a sample reorganized dataset with corresponding temporal contextual features for Fig. 6. The first, second and third rows of the table represent the input instances for the sliding windows: "Sw-a", "Sw-b", and "Sw-c", respectively (Fig. 6). Each row contains a single consumption feature, which represents the sum of a sliding window consumption dataset.

In *Feature generation* these temporal features are also used, but the generated features such as mean and standard deviation are not included because this approach uses a single consumption value as a target variable. In the *Model trainer engine*, the underlying algorithm is trained to predict consumption value. In *Model testing*, consumption is predicted for new data instances and the difference between the actual and predicted consumption values as well as the difference between the anomalous and predicted values is evaluated. These are the outputs of the *Learner model* for prediction-based anomaly classifiers. The `Learner-Model` function is outlined in Algorithm 1, and called by Algorithm 2 (line 2) (R_1).

Algorithm 2. Ensemble anomaly detection framework.

Input : *New Sensor Value* (V), *Real Dataset* (D), *Artificial Dataset* (P), *Contextual Features* (C), *Learner Models* ($L_1, L_2, \ldots L_n$), *Number of learning rounds* (R)

Output: *Notification*

```
/* Training: Learner_Model and UEVD
   lines 1-4                                   */
1  for l ← 1 to n do
       /* Function from Algorithm 1           */
2      Learner_Model (D, A, C, G, L_l)
3      S_l ← uniqueErr (err_neg_l, err_pos_l);
4  end
   /* Training: Anomaly Classifier
   lines 5-10                                  */
5  for l ← 1 to n do
6      foreach (ε ∈ S_l) do
7          H_neg L_l,ε ← anomalyClassifier (err_neg_l, ε)
8          H_pos L_l,ε ← anomalyClassifier (err_pos_l, ε)
9      end
10 end
   /* Training: Ensembler, EPE and OETD
   lines 11-19                                 */
11 foreach (j ∈ S_1) do
12     ...
13     foreach (z ∈ S_n) do
14         H_pos E ← ensembler (H_pos L_1,j, ... , H_pos L_n,z)
15         H_neg E ← ensembler (H_neg L_1,j, ... , H_neg L_n,z)
           /* a 2-D matrix of TPR and FPR of
              all ensembles                    */
16         perf ← ensembleMetrics (H_pos E, H_neg E)
17     end
18 end
19 Opt_E_thresh ← ensembleOptimizer (perf)

   /* Testing: MT, AC, Ensembler and AN
   lines 20-29                                 */
20 for l ← 1 to n do
21     err_value_l ← ModelTester (V, C, G)
22     H_l ← anomalyClassifier (err_value_l,
       Opt_E_thresh)
23 end
24 H_E ← ensembler (H_1, H_2, ... , H_n)
25 if IsAnomalous (H_E) then
26     return Notification = true
27 else
28     return Notification = false
29 end
```

Suppose that the EAD framework contains n *Learner models* denoted by $L_l (l = 1, \ldots, n)$. As already outlined in Algorithm 1, which uses a single learner, the outputs of a *Learner model* are the test outputs for normal and anomalous test datasets, denoted by the arrays *err_neg* and *err_pos*, respectively (Algorithm 1, lines 8 and 9). The EAD framework uses multiple learners, and hence the test results L_l are denoted by err_neg_l and err_pos_l. For a pattern-based learner such as the CCAD-SW, these error values represent the reconstruction error of both normal and anomalous test datasets. For prediction-based learners, these error arrays represent the difference between predicted and actual consumption values. i.e., for real training data, err_neg_l, and for anomalous data, err_pos_l.

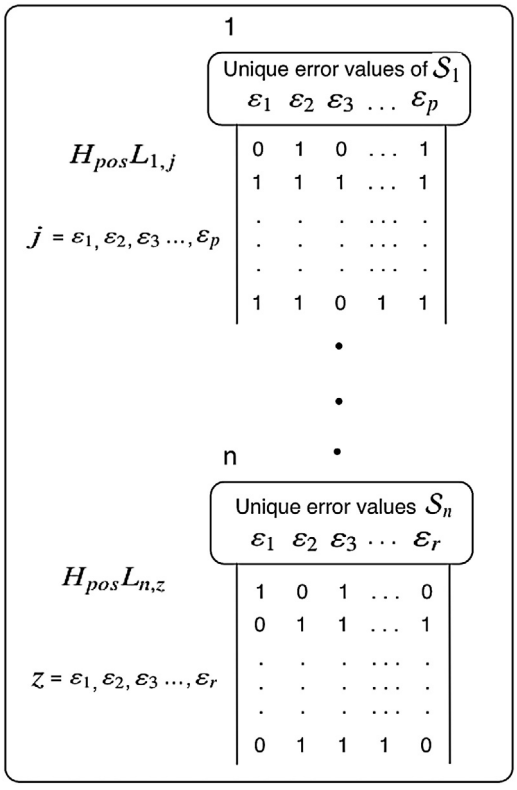

Fig. 8. *Anomaly classifier* illustration.

5.1.2. Unique error values determinator (UEVD)

Once each L_l has determined the prediction error arrays err_neg_l and err_pos_l, the `uniqueErr` function uses these error arrays to determine a set $S_l = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n\}$ that contains unique values of these errors (line 3) (R_2).

5.1.3. Anomaly classifier (AC)

The objective of the *Anomaly classifier* is to determine the anomaly class of the outputs of a *Learner Model* L_l by using each of the unique error values $\varepsilon \in S_l$ as threshold values.

The for loop in line 6 runs over all learner models and for each learner model L_l, each unique error element $\varepsilon \in S_l$ (line 7) is used as a threshold value to determine the hypotheses of the err_neg_l, which is ($H_{neg}L_{l,\varepsilon}$) (line 8) and that of the err_pos_l which is ($H_{pos}L_{l,\varepsilon}$) (line 9).

A sample illustration of the output of the *Anomaly classifier* for err_pos_l, is shown in Fig. 8. The 2D matrix in box "1" ($H_{pos}L_{1,j}$) is the hypothesis of err_pos_1 for all unique threshold values $j \in S_1$. Similarly, the 2D matrix in box "n" ($H_{pos}L_{n,z}$) is the hypothesis for all unique threshold values $z \in S_n$. More specifically, each column of the 2D matrices represents the hypothesis of the array err_pos_l determined by using the corresponding unique error value as threshold value. For instance, the first column of the 2D matrix in box "1" represents the hypothesis using threshold value $j = \varepsilon_1$.

In the same manner, the second and third and other columns are the hypotheses using threshold values $j = \varepsilon_2, \varepsilon_3, \ldots \varepsilon_p$. As a result, the lth *Anomaly classifier* creates $n(S_l)$ different anomaly classifiers, where $n(S_l)$ is the number of elements of set S_l. For instance the first *Anomaly classifier* creates p anomaly classifiers, and the last, which is the nth, creates r anomaly classifiers.

The objective of determining all possible classifiers of all learner models is to enable the EAD framework to choose the best combination of threshold values of each *Learner model* that yields optimal sensitivity and specificity from the entire ensemble.

5.1.4. Ensembler

The objective of the *Ensembler* is to combine the hypotheses of all the *Anomaly classifiers* to create a new ensemble anomaly classifier. In this research, by assuming that all learners have equal weight, all possible combinations of all *Anomaly classifiers* are used to create ensemble anomaly classifiers. The majority vote of the anomaly classifiers is used as the decision or output of the *Ensembler*.

By using all combinations of all *Anomaly classifiers*, the *Ensembler* creates $n(S_1) \times n(S_2) \times \ldots \times n(S_n)$ different ensemble anomaly classifiers. For instance, one sample ensemble from Fig. 8 is an ensemble formed by the majority vote of the hypotheses ($H_{pos}L_{1,3}$, …, $H_{pos}L_{n,5}$).

The `ensembler` function uses a combination of each *Anomaly classifier* in each *Learner model* to combine them and create new ensemble classifier (lines 16 and 17) (R_5). $H_{pos}E$ refers to the anomaly class of an ensemble model for positive test data, whereas $H_{neg}E$ refers to the anomaly class of an ensemble model for a negative test dataset.

5.1.5. Ensemble performance evaluator (EPE)

This component determines the anomaly performance of an ensemble classifier. For every ensemble classifier that the `Ensembler` combines, the `ensembleMetrics` function evaluates the performance metrics TPR and TNR of the anomaly classifier using Eqs. (6) and (7), respectively (line 18) (R_6).

By combining all possible anomaly classifiers, the *Ensembler* determines all possible ensemble anomaly classifiers, and the *Ensemble performance evaluator* determines the performance of each ensemble.

5.1.6. Optimal ensemble threshold determinator (OETD)

The *Optimal ensemble threshold determinator* (OETD) determines an ensemble threshold Opt_E_{thresh} (line 21) (R_7) that optimizes both sensitivity and specificity. The ensemble threshold, opt_E_{thresh} is a set of error values $\{\varepsilon_a, \varepsilon_b, \ldots, \varepsilon_n\}$ where, $\varepsilon_a \in S_1$, $\varepsilon_b \in S_2$, …, $\varepsilon_n \in S_n$, such that this combination of error values yields optimal ensemble performance. To identify this ensemble threshold, the ROC curve is plotted using the performance values evaluated in line 18. The ROC plot of the ensemble depends on several learners with different sets of unique error values. As a result, as shown in Fig. 9, the ROC curve is not a single curve but a scattered plot in the unit square. The reason is because multiple configurations of the base learner thresholds can yield the same FPR and TPR values.

The ensemble threshold combination that yields the optimal ensemble anomaly classifier is determined using the threshold determination technique described earlier, which assigns equal weight to sensitivity and specificity. The output of the training flow path is a trained model and a set of ensemble threshold values that yield the optimal ensemble anomaly classifier.

5.2. Testing

The testing flow path of the EAD framework in Fig. 7 is represented by the dashed lines and the letter "T".

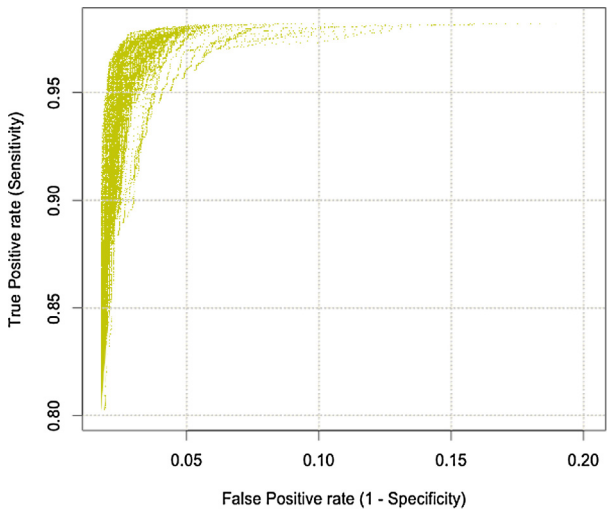

Fig. 9. Ensemble ROC.

During testing, each *Learner model* L_l initially determines its corresponding test output array err_value_l, which is the error measure for a previously unseen dataset (line 24) (T_1). Subsequently, each corresponding anomaly classifier decides about the anomaly class of the same data instance. More importantly, using the optimal threshold values of each learner determined during training by the OETD (line 21) (R_7), the *Anomaly classifier* determines the anomaly class of the *err_value* (line 25) (T_2). By using majority vote of the decisions of the anomaly classifiers, the `ensembler` function finally decides whether or not an instance is anomalous or not (line 27) (T_3). If the *Anomaly classifier* has decided that a data instance is anomalous, the `isAnomalous` function triggers the notifier (line 29)(T_4). The notification can be displayed on dashboard or sent by email, SMS or other interface. Subsequently, the responsible entity, in this case building managers, perform appropriate energy efficiency procedures.

6. Experimental results and discussion

The proposed CCAD-SW and EAD frameworks have been evaluated using datasets provided by Powersmiths [55], a company that focusses on producing sensor devices with the aim of creating a sustainable and green future. Powersmiths collects various data from sensor devices, and both of the proposed frameworks were evaluated using HVAC consumption data (kWh) for a school recorded every 5 min from 2013 to 2015.

The experiments in this study had two objectives: the first was to evaluate the performance of the CCAD-SW framework using both autoencoder and PCA. The second, was to determine the anomaly detection performance of the EAD framework by combining the aforementioned CCAD-SW framework with two prediction based learners. The experiments in this research were divided into two sections, the CCAD-SW framework and the EAD framework experiments, and are described below.

6.1. CCAD-SW framework experiments

The CCAD-SW framework was implemented using autoencoder and PCA. The performance of these two anomaly classifiers was compared with that of models already implemented in the CCAD [4] framework (CCAD-17 and CCAD-26).

The dataset was initially preprocessed, subsequently generated and contextual features integrated with it. The final dataset consisted of 337640 samples. Next, a 10% subset at the end of the dataset was set aside to assess the specificity of the CCAD-SW

Table 3
Autoencoder model parameters.

Parameter	Value
Hidden layers	3
Neurons in hidden layers	20, 10, 20
L1 (regularization parameter)	1E−04
Number of epochs	400
Activation function	Hyperbolic tangent

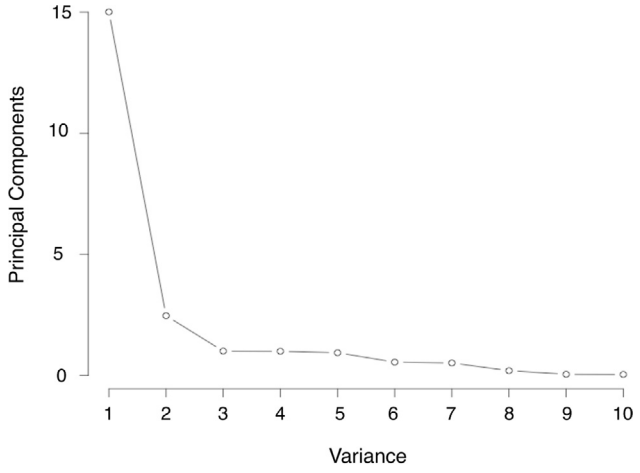

Fig. 10. CCAD-SW (PCA) screen plot.

framework. Subsequently two experiments were performed, which are described below:

Experiment 1: The objective of this experiment was to determine the sensitivity and specificity of the CCAD-SW framework using autoencoder. The autoencoder used in this research was based on a deep-learning autoencoder implementation in H_2O [56], which is a scalable and fast open-source machine learning platform. The experiment was performed within the R [57] programming environment using an H_2O API.

Initially, the autoencoder model was tuned. Various configurations of the regularization parameter (L1), number of epochs, and the activation function as well as both shallow and deep networks were considered. A model that resulted in a minimum stable Mean Square Error (MSE) was finally selected. The parameters selected for the model are given in Table 3.

The next step was repeated training and testing; to perform this, from the remaining dataset, a 10% subset was set aside for testing (the normal dataset). Subsequently, from the remaining 90% of the dataset, a random 80% training dataset was selected with replacement. After each training cycle, the model was tested using normal and artificially generated anomalous test datasets. This training and testing cycle was repeated 25 times, and the average values of the reconstruction errors of the normal and anomalous test datasets were evaluated.

Experiment 2: In this experiment, the sensitivity and specificity of the CCAD-SW framework was evaluated using PCA. The dataset that was already used in *Experiment 1* was also used for this experiment. Initially, PCA was used for dimensionality reduction purposes; it was found that the first 10 principal components described 99% of the variance of the dataset, as shown in the *scree plot* in Fig. 10. A *scree plot* is a line segment plot that shows the fraction of total variance in the data as explained or represented by each principal component.

During training, the *component loadings* were determined. The *component loadings* are the weight by which each standardized original variable should be multiplied to obtain the component score. These values show how much of the variation in a variable is explained by the component. Subsequently, the principal components that could explain 99% of the variance were selected. During testing, these *component loadings* were used to try to reconstruct previously unseen normal and anomalous test datasets, and the reconstruction error of the normal and anomalous test datasets were determined.

Table 4
CCAD-SW performance comparison.

Model	Threshold	TPR (%)	FPR (%)	AUC
CCAD-SW[b]	0.63	52.1	50.4	0.513
CCAD-17[a]	0.07	68.6	12.7	0.842
CCAD-26[a]	0.05	80.2	21.1	0.862
CCAD-SW[a]	0.001	94.5	4.7	0.981

[a] Autoencoder.
[b] PCA.

The outputs of both these experiments were the reconstruction errors for positive and negative test datasets. Using these reconstruction errors, the TN and TP density distributions for both experiments were determined. Subsequently, for each experiment, the TPR and FPR = (1 − TNR) were evaluated using Eqs. (6) and (7), respectively. These values were used to plot the ROC curves of the anomaly classifiers. Moreover, assuming that both sensitivity and specificity have equal weight, a threshold value that optimizes these two metrics was determined using Eq. (8).

Finally, each model was tested using previously unseen normal and anomalous datasets. Subsequently, using the threshold values determined in each experiment, the test outputs of these new datasets were classified as either anomalous or not.

To compare the performance of all these models, the following metrics were used: TPR, FPR, and AUC. In this experiment the ROC curves of the CCAD-SW (autoencoder) and the CCAD-SW (PCA) did not cross the other curves, and hence the pAUC was not considered. To evaluate the AUC, a function *AUC* from the R package that approximates the area under the curve using the trapezoid rule was used. An AUC = 1, which is the maximum value, represents a perfect anomaly classifier, where as an AUC = 0.5 represents a non-discriminant or random classifier.

6.2. CCAD-SW experimental results and discussion

The results of *Experiments 1* and *2* are shown in Fig. 11a and b, respectively. These are the density distributions of the normalized values of the reconstruction errors. For each error value on the *x*-axis, the plots show the proportions of TP and TN for the anomaly classifiers. This distribution was referred to as the *TP–TN density distribution* in these experiments. From the figures, the peak of the TN and the peak of the TP for the CCAD-SW (autoencoder) were more separated while for the CCAD-SW(PCA) they overlapped. Intuitively, this indicates that the latter implementation has a lower anomaly detection performance.

Fig. 12 shows the ROC curve of both CCAD-SW implementations as well as the CCAD-17 and CCAD-26 models which were already implemented using the CCAD [4] framework. Intuitively, from Fig. 12, the CCAD-SW (autoencoder) can be seen to have a larger AUC than any of the other anomaly classifiers. Moreover, the CCAD-SW (PCA) has a curve that almost matches the *line of non-discrimination*, which is the linear diagonal line shown in the figure.

The performance metrics of the four anomaly detection classifiers are shown in Table 4. They confirm the observations made earlier: CCAD-SW (autoencoder) had the largest AUC followed by CCAD-26, CCAD-17, and lastly CCAD-SW (PCA). If the optimal values of TPR and FPR of each anomaly classifier are compared, with a

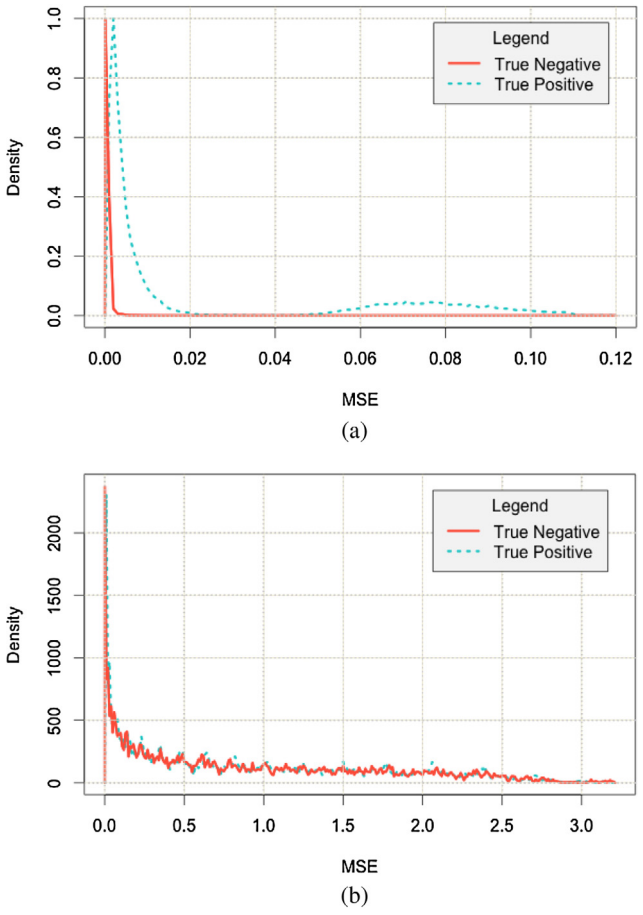

Fig. 11. TP–TN distribution: (a) CCAD-SW (autoencoder), (b) CCAD-SW (PCA).

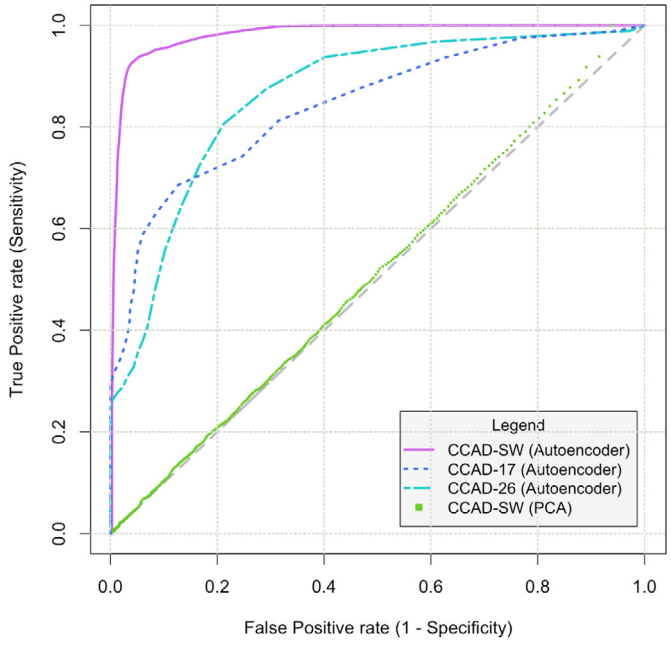

Fig. 12. ROC of the CCAD-SW and CCAD models.

TPR of 94.5% the CCAD-SW (autoencoder) had the highest anomaly detection rate while still maintaining a FPR of 4.7%, which was the lowest of all the other anomaly detection classifiers. As for the CCAD-SW (PCA), the optimal TPR and FPR measures also showed

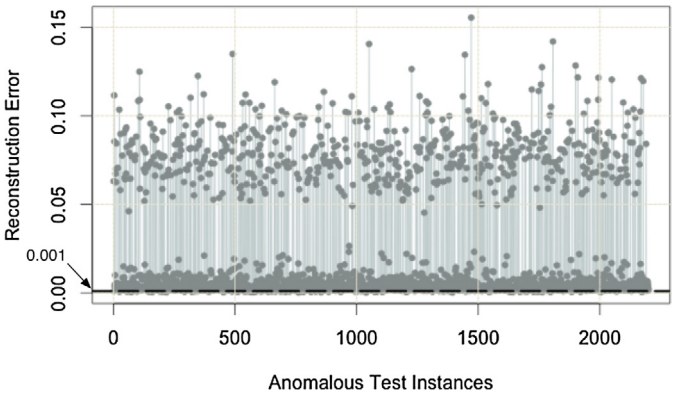

Fig. 13. Reconstruction error of anomalous test data.

the same non-discriminant behaviour as indicated by the AUC. Both had values close to 50%. Although the CCAD-SW (PCA) performed slightly better than the random classifier for larger values of FPR (greater than 80%), false positives in this range would not be acceptable for a workable anomaly detection system.

Overall, the experiments done showed that for the dataset used, the CCAD-SW (autoencoder) is well-suited for both lenient and strict FPR requirements. Fig. 13 shows the reconstruction error values for artificially generated anomalous test data using CCAD-SW (autoencoder), and 94% of these anomalous instances are greater than the optimal threshold value 0.001 indicated in the figure.

6.3. EAD framework experiment

The objective of this experiment was to evaluate the anomaly detection performance of the EAD framework by combining the CCAD-SW (autoencoder) framework described above with two prediction-based anomaly detection classifiers. These two classifiers were implemented using random forest and SVR. A comparison was also made between the anomaly detection performance of the EAD framework and the three anomaly classifiers selected.

The CCAD-SW is a neural network-based learning framework that uses autoencoder. A random forest, is an ensemble learning algorithm that combines the hypotheses of several decision trees, and SVR is a version of support vector machine (SVM) for regression.

The experiment was subdivided into four steps and these are:

- *CCAD-SW framework based learner model*: In this step, a CCAD-SW anomaly detection classifier based on autoencoder was implemented and the anomaly class of normal and anomalous test datasets determined.
- *SVR-based learner model*: In this step, a prediction-based anomaly detection classifier was implemented using SVR, and the anomaly class of the normal and anomalous test datasets used in the previous step was determined.
- *Random forest-based learner model*: In this step, a prediction-based anomaly detection classifier was implemented using random forest, and the anomaly class of the normal and anomalous test datasets used in the previous steps was determined.
- *EAD framework anomaly classifier*: Using majority voting, this step combined the decisions of the three anomaly classifiers.

CCAD-SW framework-based learner model: This step was already implemented in *Experiment 1*, which was described in the CCAD-SW experimental section. The output of this experiment was the reconstruction error for previously unseen positive and negative test datasets.

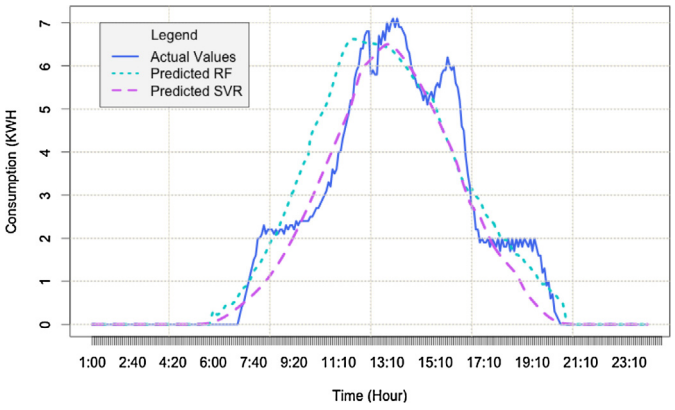

Fig. 14. Predicted and actual consumption values.

SVR-based learner model: The objective of this step was to implement a prediction-based anomaly detection classifier using SVR. The dataset was initially prepared as described in Section 5, and the temporal features mentioned in Section 4.1.2 were introduced into the dataset. The final dataset included a total of 7 features: 6 contextual features and one consumption feature. The same normal and anomalous test datasets used for the experiment described in Section 6.1 were preprocessed and used for final model testing. After preparing the dataset, the parameters of the SVR were tuned by considering various configurations of the parameters C (cost) and γ. By holding one constant and varying the other, a configuration that yielded the minimum MSE was selected; $C = 10$ and $\gamma = 0.1$.

Next, using the remaining subset of the dataset, the same training and testing technique was applied as described in the CCAD-SW experimental section. The average predicted values of the test runs were determined and using these values, the difference from the actual values were evaluated. The difference between the actual (normal dataset) and the predicted values is the error of the negative test dataset, whereas the difference between the predicted values and the artificially generated anomalous test datasets is the error of the positive test dataset.

Random forest-based learner model: The next step was to perform procedures similar to those in the previous, *SVR-based learner model* step, using the random forest algorithm. The same test datasets used in the previous step were also used.

Various random forests with varying tree sizes were considered. For the dataset used, a random forest configuration with 400 trees yielded the minimum MSE value and was selected for the experiment. Using the selected random forest model selected, consumption was predicted and the error values determined as described in the *SVR-based learner model* step.

EAD framework anomaly classifier: The last part of the experiment was to combine the decisions of the anomaly detection classifiers based on the three learner models described so far. The final output of the EAD framework was the anomaly class of the test data as determined by a majority vote of the three anomaly detection classifiers described above.

6.4. EAD experimental results and discussion

The *Random Forest-based learner model* and *SVR-based learner model* were implemented to predict consumption value of each collective data as shown in Fig. 14. The figure shows a sample daily energy consumption prediction for the month of June, and the models describe the data fairly well. Subsequently, the TP–TN density distributions of the *Random Forest-based learner model* and *SVR-based learner model* are illustrated in Fig. 15a and b, respectively.

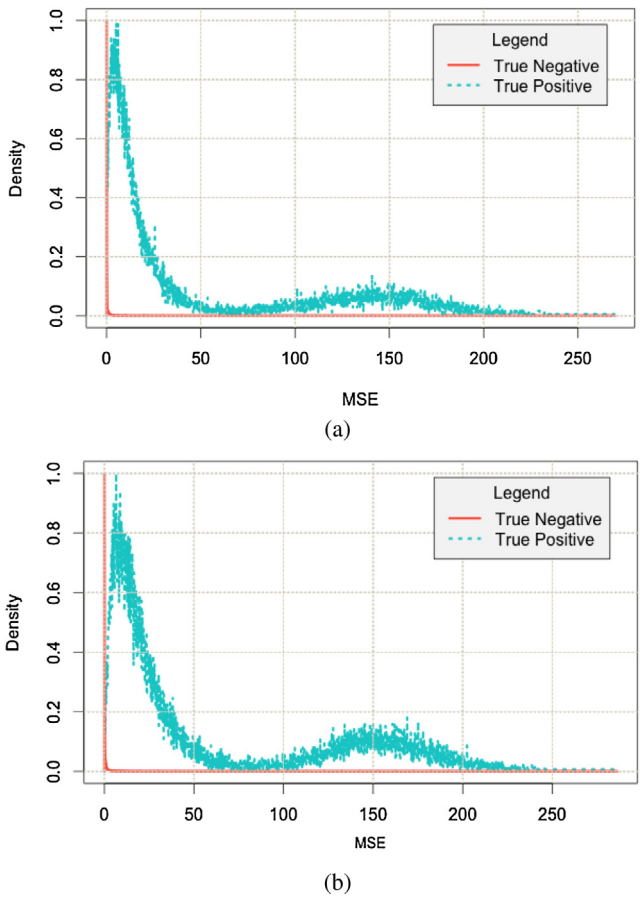

Fig. 15. TP–TN distribution: (a) random forest, (b) SVR.

It is clear that the the peak of the densities of TP and TN are separated, which intuitively shows that the models have a reasonable anomaly detection capacity. The ROC curves of the anomaly classifiers are shown in Fig. 16. The ROC plot of the EAD framework is not a single curve because the framework relies on three different learner models. For instance, from the ROC curve of the CCAD-SW (autoencoder) in Fig. 12, it can be observed that each FPR value corresponds to one and only one TPR value.

The mean TPR value for each FPR is plotted in the top zoomed figure of Fig. 16. Intuitively, the ROC plot shows that the EAD framework outperformed the rest of the individual anomaly classifiers. The ROC curves in Fig. 16 cross, and hence, for the reasons discussed earlier, the pAUC was also used as a metric to evaluate and compare the anomaly detection classifiers. Moreover, as in the previous experiments, the optimal values of TPR and TNR as well as the AUC were used as performance measuring metrics.

As shown in Fig. 16, the ROC curves cross at an FRP of 6%, and hence the pAUC was analysed for FPR ranges of (0–6%) and (6–20%). In this study, because the anomaly classifiers are performed well even for lower false positive rates, only low false positive rates were considered. Moreover, the trend of the curves did not significantly change beyond the ranges considered.

Table 5 shows the optimal TPR and FPR values as well as the threshold values that yielded these optimal values. For the EAD framework, the optimal TPR and FPR was achieved at a combined threshold values (ensemble threshold) of CCAD-SW = 0.0032, random forest = 0.3, and SVR = 2. This shows that for the dataset used in this research, optimal ensemble anomaly classifier is not attained by combining the base anomaly classifiers at their respective optimal thresholds which is CCAD-SW = 0.001, random forest = 1.7 and

Table 5
EAD performance comparison.

Model	Threshold	TPR (%)	FPR (%)	pAUC[b]	
				FPR (0–6%)	FPR (6–20%)
SVR	3.2	95.7	7.10	0.82	0.955
CCAD-SW[a]	0.001	94.5	4.70	0.89	0.960
Random forest	1.7	94.9	6.60	0.87	0.965
EAD	[2, 0.0032, 0.3][c]	98.1	1.98	0.95	0.97

[a] Autoencoder.
[b] pAUC standardized [0–1].
[c] Threshold at these values of SVR, CCAD-SW and random forest, respectively.

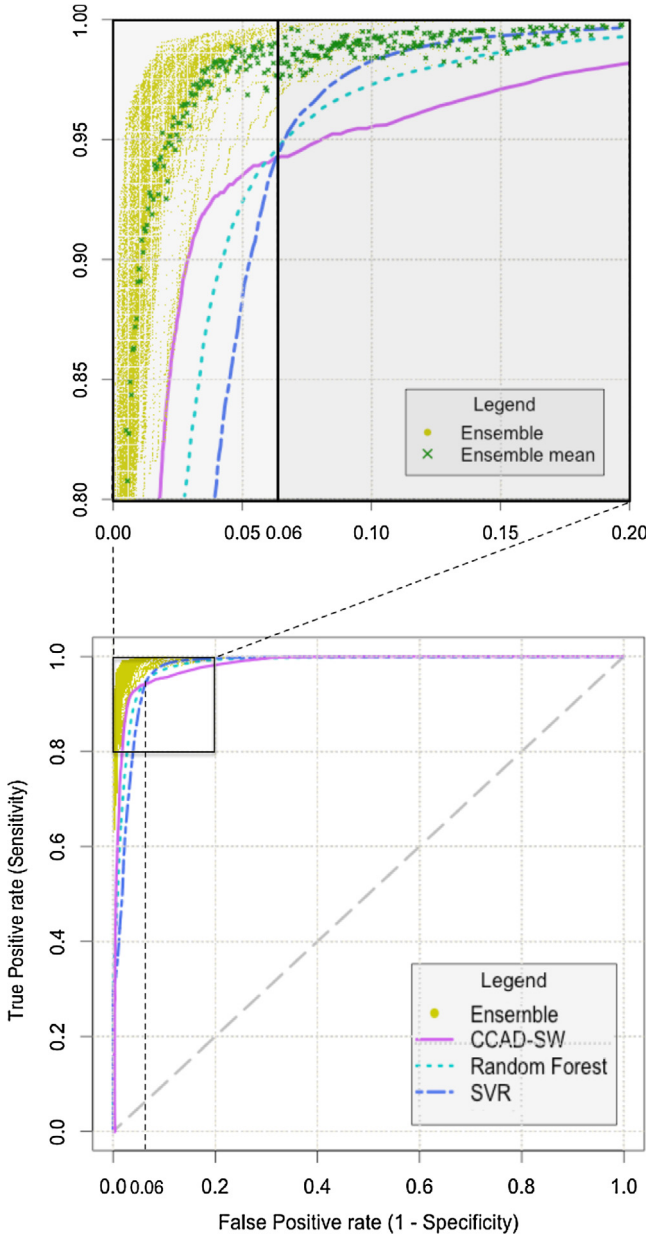

Fig. 16. ROC of the EAD framework.

SVR = 3.2, but instead at the values CCAD-SW = 0.0032, random forest = 0.3, and SVR = 2.

The table also shows that for FPR range (0–6%) with a pAUC of 0.95, the EAD framework performed better than any of the base anomaly classifiers. The CCAD-SW (autoencoder) was the second best in this FPR range followed by the random forest based and SVR based anomaly classifiers. Moreover, with a pAUC of 0.97, the EAD framework still outperformed the base learners in the higher FPR range, i.e., (6–20%). Although the SVR was the worst-performing anomaly classifier for this sensitivity range, the optimal TPR and FPR values also indicate that the EAD framework outperformed the other classifiers not only in anomaly detection (higher TPR), but also in reducing false positives (lower FPR). Although the SVR-based anomaly classifier had a TPR of 95.7% which is the second best, its has the highest FPR of all the classifiers, 7.10 %. From these experiments, it can be concluded that for all the FPR ranges considered, the EAD is a better anomaly classifier than any of the base learners.

7. Conclusions and future work

In this research, two generic anomaly detection frameworks have been proposed: CCAD-SW and EAD. The CCAD-SW framework, which is a pattern-based anomaly classifier was implemented using autoencoder. The EAD framework combines the CCAD-SW with prediction-based anomaly detection classifiers. In this research, the prediction-based classifiers were implemented using support vector regression and random forest.

The results show that the CCAD-SW improved the TPR of the CCAD by 15% and reduced the FPR by 8%. Moreover, the EAD framework further improved the TPR of the CCAD-SW by 3.6% and reduced the FPR by 2.7%. These results show that the EAD framework is suitable both for stringent anomaly detection and demanding false positive requirements. In this study, it was found out that the optimal combined threshold of the EAD was not achieved at the optimal threshold values of the base learners. Instead, the optimal ensemble threshold was achieved by searching through the threshold space.

Future research will explore more robust voting techniques such as weighted voting. In addition, hybrid base classifiers that are trained on different subsets of a dataset, as well as classifiers that are trained with different features of a dataset, will be studied.

Acknowledgments

This research was supported in part by an NSERC CRDat Western University (CRDPJ 453294-13). In addition, the authors would like to acknowledge the support provided by Powersmiths.

References

[1] UNEP, United Nations Environment Program, http://www.unep.org/sbci/aboutsbci/background.asp.
[2] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection, ACM Comput. Surv. 41 (3) (2009) 1–58.
[3] M. Drăgoicea, L. Bucur, M. Pătraşcu, A service oriented simulation architecture for intelligent building management, in: Exploring Services Science, Springer, 2013, pp. 14–28.
[4] D.B. Araya, K. Grolinger, H.F. ElYamany, M.A. Capretz, G. Bitsuamlak, Collective contextual anomaly detection framework for smart buildings, in: International Joint Conference on Neural Networks (IJCNN), IEEE, 2016, pp. 1–8.
[5] Z.-H. Zhou, Ensemble learning, Encycl. Biom. (2015) 411–416.

[6] D. Opitz, R. Maclin, Popular ensemble methods: an empirical study, J. Artif. Intell. Res. (1999) 169–198.
[7] R. Polikar, Ensemble learning, in: Ensemble Machine Learning, Springer, 2012, pp. 1–34.
[8] J.C. Lam, K.K. Wan, K. Cheung, L. Yang, Principal component analysis of electricity use in office buildings, Energy Build. 40 (5) (2008) 828–836.
[9] P. Geladi, B.R. Kowalski, Partial least-squares regression: a tutorial, Anal. Chim. Acta 185 (1986) 1–17.
[10] P. Praus, SVD-based principal component analysis of geochemical data, Cent. Eur. J. Chem. 3 (4) (2005) 731–741.
[11] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.
[12] M. Sakurada, T. Yairi, Anomaly detection using autoencoders with nonlinear dimensionality reduction, in: MLSDA 2014, 2nd Workshop on Machine Learning for Sensory Data Analysis, 2014, p. 4.
[13] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[14] K. Ellis, J. Kerr, S. Godbole, G. Lanckriet, D. Wing, S. Marshall, A random forest classifier for the prediction of energy expenditure and type of physical activity from wrist and hip accelerometers, Physiol. Meas. 35 (11) (2014) 2191–2203.
[15] A. Kusiak, M. Li, F. Tang, Modeling and optimization of hvac energy consumption, Appl. Energy 87 (10) (2010) 3092–3102.
[16] C. Nguyen, Y. Wang, H.N. Nguyen, Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic, J. Biomed. Sci. Eng. 6 (5) (2013) 551–560.
[17] T. Hastie, R. Tibshirani, J. Friedman, J. Franklin, The elements of statistical learning: data mining, inference and prediction, Math. Intell. 27 (2) (2005) 83–85.
[18] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[19] V.N. Vapnik, S. Kotz, Estimation of Dependences Based on Empirical Data, vol. 40, Springer-Verlag, New York, 1982.
[20] R.Ž Jovanović, A.A. Sretenović, B.D. Živković, Ensemble of various neural networks for prediction of heating energy consumption, Energy Build. 94 (2015) 189–199.
[21] A. Smola, V. Vapnik, Support vector regression machines, Adv. Neural Inf. Process. Syst. 9 (1997) 155–161.
[22] R.K. Jain, K.M. Smith, P.J. Culligan, J.E. Taylor, Forecasting energy consumption of multi-family residential buildings using support vector regression: investigating the impact of temporal and spatial monitoring granularity on performance accuracy, Appl. Energy 123 (2014) 168–178.
[23] R. Kumar, A. Indrayan, Receiver operating characteristic (ROC) curve for medical research., Indian Pediatr. 48 (17) (2011) 277–287.
[24] K. Hajian-Tilaki, Receiver operating characteristic (ROC) curve analysis for medical diagnostic test evaluation, Casp. J. Internal Med. 4 (2) (2013) 627–635.
[25] D.K. McClish, Analyzing a portion of the ROC curve, Med. Decis. Mak. 9 (3) (1989) 190–195.
[26] J.-S. Chou, A.S. Telaga, Real-time detection of anomalous power consumption, Renew. Sustain. Energy Rev. 33 (2014) 400–411.
[27] H. Janetzko, F. Stoffel, S. Mittelstädt, D.A. Keim, Computers and graphics anomaly detection for visual analytics of power consumption data, Comput. Graph. 38 (2013) 1–11.
[28] M. Wrinch, T.H.M. El-Fouly, S. Wong, Anomaly detection of building systems using energy demand frequency domain analysis, in: 2012 IEEE Power and Energy Society General Meeting, 2012, pp. 1–6.
[29] D.J. Hill, B.S. Minsker, Anomaly detection in streaming environmental sensor data: a data-driven modeling approach, Environ. Model. Softw. 25 (9) (2010) 1014–1022.
[30] G. Bellala, M. Marwah, M. Arlitt, Following the electrons: methods for power management in commercial buildings, in: 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining – KDD '12, 2012, pp. 994–1002.
[31] R. Fontugne, J. Ortiz, N. Tremblay, P. Borgnat, P. Flandrin, K. Fukuda, D. Culler, H. Esaki, Strip, bind, and search: a method for identifying abnormal energy consumption in buildings, in: 12th International Conference on Information Processing in Sensor Networks, 2013, pp. 129–140.
[32] P. Arjunan, H.D. Khadilkar, T. Ganu, Z.M. Charbiwala, A. Singh, P. Singh, Multi-user energy consumption monitoring and anomaly detection with partial context information, in: Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments, ACM, 2015, pp. 35–44.
[33] Y. Zhang, W. Chen, J. Black, Anomaly detection in premise energy consumption data, in: 2011 IEEE Power and Energy Society General Meeting, 2011, pp. 1–8.
[34] A.L. Zorita, M.A. Fernández-Temprano, L.-A. García-Escudero, O. Duque-Perez, A statistical modeling approach to detect anomalies in energetic efficiency of buildings, Energy Build. 110 (2016) 377–386.
[35] J. Ploennigs, B. Chen, A. Schumann, N. Brady, Exploiting generalized additive models for diagnosing abnormal energy use in buildings, in: 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings, 2013, pp. 17:1–17:8.
[36] Y. Jiang, C. Zeng, J. Xu, T. Li, Real time contextual collective anomaly detection over multiple data streams, in: SIGKDD Workshop on Outlier Detection and Description under Data Diversity, 2014, pp. 23–30.
[37] M. Peña, F. Biscarri, J.I. Guerrero, I. Monedero, C. Léon, Rule-based system to detect energy efficiency anomalies in smart buildings: a data mining approach, Expert Syst. Appl. 56 (2016) 242–255.
[38] A. Capozzoli, F. Lauro, I. Khan, Fault detection analysis using data mining techniques for a cluster of smart office buildings, Expert Syst. Appl. 42 (9) (2015) 4324–4338.
[39] M.A. Hayes, M.A. Capretz, Contextual anomaly detection framework for big sensor data, J. Big Data 2 (1) (2015) 1–22.
[40] J.B. Cabrera, C. Gutiérrez, R.K. Mehra, Ensemble methods for anomaly detection and distributed intrusion detection in mobile ad-hoc networks, Inf. Fusion 9 (1) (2008) 96–119.
[41] L. Didaci, G. Giacinto, F. Roli, Ensemble learning for intrusion detection in computer networks, in: Workshop Machine Learning Methods Applications, Siena, Italy, 2002.
[42] G. Folino, F.S. Pisani, P. Sabatino, A distributed intrusion detection framework based on evolved specialized ensembles of classifiers, in: Applications of Evolutionary Computation, Springer, 2016, pp. 315–331.
[43] Z. Zhao, K.G. Mehrotra, C.K. Mohan, Ensemble algorithms for unsupervised anomaly detection, in: Current Approaches in Applied Artificial Intelligence, Springer, 2015, pp. 514–525.
[44] M. Amozegar, K. Khorasani, An ensemble of dynamic neural network identifiers for fault detection and isolation of gas turbine engines, Neural Netw. 76 (2016) 106–121.
[45] A.A. Aburomman, M.B.I. Reaz, A novel SVM-KNN-PSO ensemble method for intrusion detection system, Appl. Soft Comput. 38 (2016) 360–372.
[46] L. Shoemaker, L.O. Hall, Anomaly detection using ensembles, in: Multiple Classifier Systems, Springer, 2011, pp. 6–15.
[47] D. Pyle, Data Preparation for Data Mining, Morgan Kaufmann, 1999.
[48] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[49] J. Torres-Sospedra, C. Hernández-Espinosa, M. Fernández-Redondo, Using bagging and cross-validation to improve ensembles based on penalty terms, in: Neural Information Processing, Springer, 2011, pp. 588–595.
[50] Y. Grandvalet, Bagging equalizes influence, Mach. Learn. 55 (3) (2004) 251–270.
[51] F.S. Tsai, Dimensionality reduction techniques for blog visualization, Expert Syst. Appl. 38 (3) (2011) 2766–2773.
[52] C. Orsenigo, C. Vercellis, Linear versus nonlinear dimensionality reduction for banks' credit rating prediction, Knowl. Based Syst. 47 (2013) 14–22.
[53] T.G. Dietterich, Ensemble methods in machine learning, in: Multiple Classifier Systems, Springer, 2000, pp. 1–15.
[54] L.I. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Mach. Learn. 51 (2) (2003) 181–207.
[55] Powersmiths, Powersmiths: Power for the Future, http://ww2.powersmiths.com/index.php?q=content/powesmiths/about-us.
[56] H_2O, H_2O.ai, http://h2oworld.h2o.ai/#about.
[57] Team, R Core, R: A Language and Environment for Statistical Computing, http://www.R-project.org/.