BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 73(3), 2025, Article number: e153832
DOI: 10.24425/bpasts.2025.153832

Description and comparison of fault detection algorithms based on a selected building automation device

Sandra WŁOSTOWSKA*, Bartłomiej KAWA and Piotr BORKOWSKI

Department of Electrical Apparatus, Faculty of Electrical, Electronics, Computer and Control Engineering, Lodz University of Technology, Łódź, Poland

Abstract. The article compares selected classification algorithms and those dedicated to anomaly detection. The models used temperature measurements in four rooms simulated in the MATLAB Simscape environment as test signals. The empirical part of the work consists of two parts. In the first one, an example data from the simulated building heating model object, models were built using unsupervised and supervised machine learning algorithms. Then, data from the facility was collected again with changed parameters (failures occurred at times other than the test ones, and the temperature patterns differed from those recorded and used to train the models). The algorithm effects and test signals (temperature changes) were saved in the database. The results were presented graphically in the Grafana program. The second part of the work presents a solution in which the analysis of the operating status of the heating system takes place in real time. Using an OPC server, data was exchanged between the MATLAB environment and the database installed on a virtual machine in the Ubuntu system. The conclusions present the results and collect the authors' suggestions regarding the practical applications of the discussed classification models.

Keywords: fault detection; machine learning; building automation; smart metering.

1. INTRODUCTION

In today's dynamic development of recent technologies, work on fault detection systems is essential for maintaining the reliability and safety of various processes and devices. Detecting potential problems or damage enables quick corrective action before a total failure occurs. It helps to effectively exclude high repair costs and unfavourable equipment downtime. The early detection model of potential threats contributes to a higher reliability level and extends the durability of the device. Instead of routine, constant inspections detecting damage on an ongoing basis facilitate optimal planning of maintenance works. For some devices, failures may pose a serious threat to user safety. The fault detection model can allow for quick intervention and appropriate corrective measures, minimizing risk to people using the device [1].

In the context of this challenge, the presented study focuses on analyzing and comparing various fault detection algorithms. Thanks to the use of advanced analysis techniques, the presented research allows for the selection of solutions that can significantly improve the ability, speed, and effectiveness of detecting, locating, and diagnosing irregularities in building automation systems [2].

*e-mail: sandra.wlostowska@p.lodz.pl

Manuscript submitted 2024-10-07, revised 2025-01-25, initially accepted for publication 2025-02-03, published in May 2025.

2. BUILDING AUTOMATION

Building automation is a field of technology that uses control and automation systems in buildings to optimize their performance, energy efficiency, user comfort, and safety. It includes a network of devices that manage various installations in the building and its surroundings. This automated connection system creates the so-called intelligent building that enables control of many of its functions. The benefits of building automation include improved installation operation, increased comfort, financial savings, security, and remote control. Fault detection and analysis in building automation systems involves identifying, locating, and understanding the causes of device malfunctions or failures. Many diagnostic methods and fault location techniques support this process, which include data analysis, tests and inspections, signaling techniques, diagnostics and prediction, laboratory tests, fault location techniques, and cause and effect analysis [3, 4].

3. MACHINE LEARNING

Machine learning is one of the areas of artificial intelligence that includes a set of algorithms capable of independently improving their performance by gaining experience and analyzing data. In other words, it is a process in which the software can learn from the information provided and gradually improve its skills through subsequent operations. In the 1950s and 1960s, Artur Samuel from IBM developed a program to train chess players, which can be considered the beginning of machine learning. Samuel introduced the term "machine learning" as "the ability

of computers to learn without having to program new skills." In 1965, the Dendral program was created at Stanford University, which was a breakthrough in analyzing and identifying molecules of organic compounds based on data [5, 6].

Today, there are many types of machine learning, which can be divided into two main categories: supervised and unsupervised learning.

Supervised learning [7] involves the presence of a human in the learning process. In this case, the algorithms learn data based on input that is labeled, i.e., assigned appropriate markings or classes. Algorithms learn to recognize patterns and create predictive models based on labeled data. Examples of supervised learning techniques include classification, regression, and prediction.

On the other hand, unsupervised learning [8] occurs without labeled data and does not require human supervision. Algorithms learn from the structure and patterns of input data, identifying relationships and grouping similar objects. Examples of unsupervised learning techniques include clustering, dimensionality reduction, and association.

3.1. Data classification

Classification involves assigning objects to specific, known categories, where each can be assigned to only one class. The classification process consists of two stages. The first stage is learning, generating the knowledge necessary to conduct the process. The second stage is the actual determination of the result, during which the knowledge generated in the learning phase is used [9].

3.2. Data classification algorithms

Data classification algorithms include various methods and techniques for creating classification models. An algorithm is a specific set of clearly defined steps that a computer can perform to achieve a specific result. With machine learning models, the goal is to establish or discover patterns that humans can use to make predictions or classify data [6].

3.3. Anomaly detection algorithms

Anomaly detection algorithms are unsupervised learning algorithms and are data analysis techniques that aim to identify unusual, rare, or outlier observations in a dataset. Anomalies may indicate observations that differ from the normal pattern or represent potential abnormalities, failures, fraud, or unusual behavior [10].

3.4. Criteria for comparing classification algorithms

When analyzing the results obtained by the algorithms, classification efficiency, i.e., the ability of the algorithm to correctly assign samples to the correct classes, was chosen as the main evaluation criterion. Although accuracy is the most commonly used indicator, sensitivity, i.e., the ability of the model to detect the most minor deviations from the norm, was also considered here [11]. It will also be essential to compare the performance of all algorithms, considering the time required for training and classification. This time is crucial, especially with large datasets or real-time applications [12].

4. METHODOLOGY

For this work, a model illustrating the heating system of a building (Fig. 1) offered in the MATLAB-Simulink [13, 14] library was used. The facility consisted of four rooms equipped with radiators. The simulation calculated the room temperature values based on the heat exchange with the environment through its external walls, roof, and windows. Each path was simulated as a combination of thermal convection, heat conduction, and thermal mass. It was assumed that no heat was transferred internally between rooms. The heater consisted of a furnace, a boiler, an accumulator, and a pump that circulated hot water in the system. A PI controller with an active integral limit, anti-windup, started fuel flow to the furnace if the average room temperature fell below 21°C and stopped it if the temperature exceeded 25°C. The disturbance in the example was the external temperature with a sinusoidal waveform. It was important

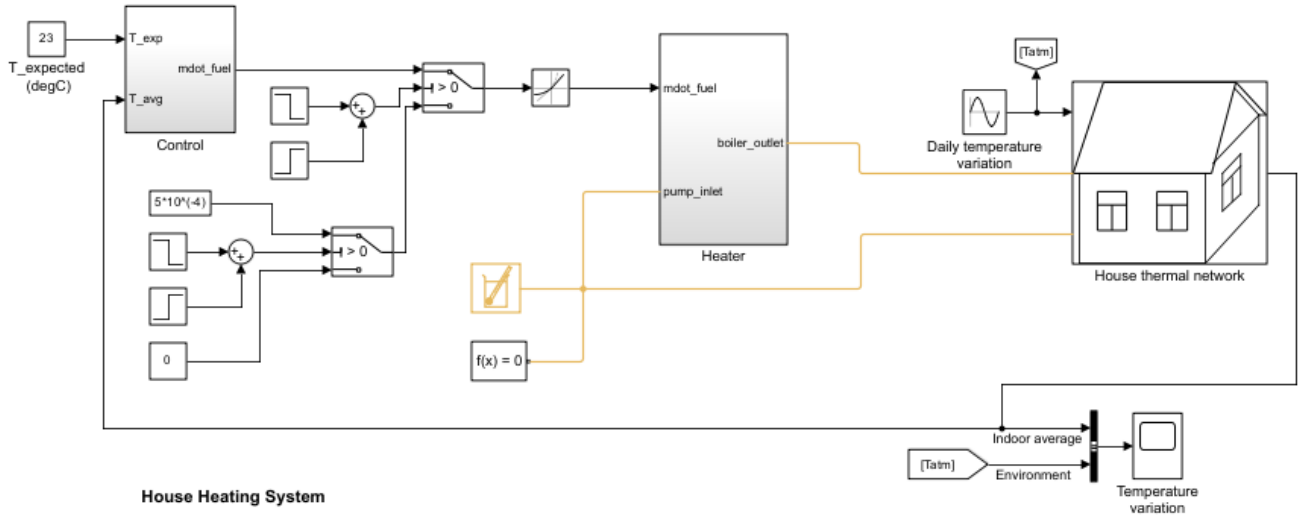

Fig. 1. MATLAB building heating model

that the waveform changes gradually so that the model could be evaluated for subtle changes in temperature values. For this reason, a sine wave was used instead of abrupt jumps in values. A fault was manually introduced into the simulation in the form of open windows, damaged radiators, and a damaged stove to disturb the temperature inside the rooms.

Using the Repeating Sequence block, time series were built for combinations of different simulation variances of radiator damage and open windows, manipulating the value of the furnace efficiency and the leakage of warm air through the windows. Time series include damage in each room, then combinations of damaged radiators in two rooms, and for open windows additionally in three and all rooms. Damage to all radiators at once was not possible in the simulation. Figure 2 shows the Repeating Sequence blocks, and the object transmittance added to the simulation.

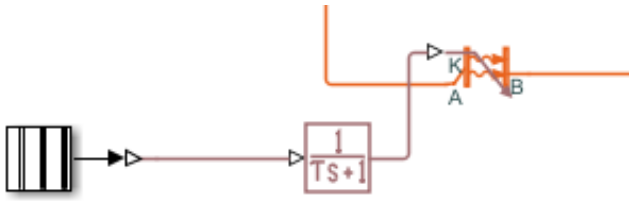

Fig. 2. Repeating Sequence block and object transfer

The work was then divided into two parts. The first involved simulating the operating time of the facility, while the second presented a data exchange solution between the real-time test facility and the database. In the second, an OPC server solution (KEPSerwerEX) was used [15].

4.1. Supervised learning of algorithms

A run was created with the training data, where all possible variances of the interference were simulated and then given labels (Fig. 3).

The data was divided into temperature values for individual rooms. The runs were divided into parts presenting data for individual faults and those representing the correct operation of the model (Fig. 4).

```
p0=1; p0e=290; p1=291; p1e=685; p2=686; p2=686; p2e=967; p3=968; p3e=1345;
```

Fig. 4. Determining the division points of the route – section

Each fault was assigned an individual variable, and then using the "for" command, each data sample was associated with a label describing a specific fault, e.g., Open window in room 1, Damaged radiator in room 2. The purpose of this procedure is to later indicate to the algorithms an appropriate rule that will map the input data to the desired output. In the next step, the "species" matrix was declared, containing all variables describing the faults. Based on this matrix and data from all rooms, a table with training data was created (Fig. 5).

```
T_tren=table(species,Room1,Room2,Room3,Room4,Outside);
```

Fig. 5. Training data table

The above steps were repeated to prepare the test data. It was necessary to remember that the order of faults in the run for the test data should be different from that for the training data (Fig. 6).

```
T_test=table(species,Room1,Room2,Room3,Room4,Outside);
```

Fig. 6. Test data table

The data was standardized and, using the Classification Learner tool available in the Statistics and Machine Learning Toolbox [16, 17], classification models for supervised learning algorithms were created. It is an interactive and graphical user interface that allows easy and fast exploration, analysis, and

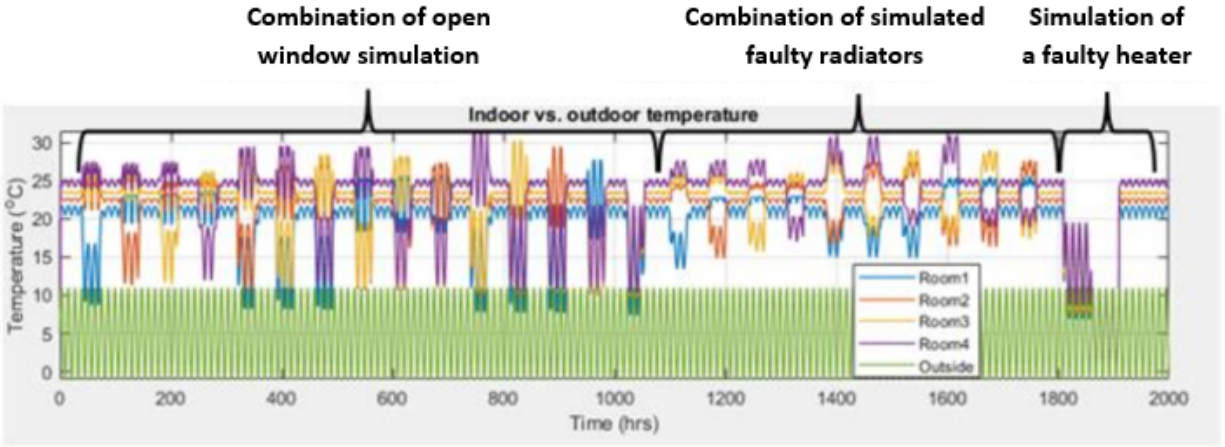

Fig. 3. Temperature waveforms recorded

training of classification models. The first step in the process of testing algorithms in MATLAB is to collect the data on which classification will be performed. The data must be imported into the Classification Learner environment. As mentioned earlier, the dataset is divided into two main subsets: training set and test set. The training set is used to train the model, and the test set is used to evaluate the model performance on new data. In the initial comparison, all algorithms offered by MATLAB were used. In the model training process, the algorithms are fitted to the training data to teach the models to recognize patterns and their associated class labels. After training is completed, it is possible to evaluate the performance of each model on the test set. Different performance evaluation metrics are available in Classification Learner, such as accuracy, precision, and recall.

In the initial analysis, the accuracy of prediction was considered, and five algorithms were selected for further analysis (Table 1).

Table 1
Algorithms with the highest failure detection efficiency

No.	Algorithm type	Subtype	Accuracy
1.	Logistic regression	Efficient logistic regression	91.10%
2.	Support vector machine	Quadratic SVM	91.74%
3.	K-nearest neighbors	Cosine KNN	89.99%
4.	Multiple classifiers	Bagged trees	90.80%
5.	Neural network	Trilayered neural network	92.33%

The accuracy of classification models is calculated based on the model prediction results as compared to known class labels in the test data set. It is described by the following equation:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of samples}}.$$

In the initial phase, it is often crucial to filter algorithms in terms of their overall quality, and accuracy is remarkably effective in doing this. Only in the subsequent stages of analysis were the models assessed in terms of precision and recall.

- Efficient logistic regression – this classifier works based on classical logistic regression. Still, it was optimized to better cope with large data sets. It uses efficient optimization techniques that allow the model to fit faster on large data sets. By optimizing the learning process, this type of logistic regression achieves higher performance, resulting in reduced time and resources needed to train the model on big data [18]. Figure 7 shows the algorithm training parameters. The parameters include, among others, data such as the accuracy of validation and test data. The prediction speed is expressed in the number of observations per second, which means the number of predictions that the classifier can make in one second, with the total training time and the size of the model.
- Quadratic SVM – an extension of the standard SVM algorithm that allows quadratic interactions between features to be considered during the classification process. This means

Training Results

Accuracy (Validation)	92.8%
Total cost (Validation)	1251
Prediction speed	~2400 obs/sec
Training time	162.98 sec
Model size (Compact)	~4 MB

Test Results

Accuracy (Test)	91.1%
Total cost (Test)	1461

Fig. 7. Efficient logistic regression algorithm training parameters

that the Quadratic SVM model can detect more complex relationships between features, particularly useful for data with nonlinear decision boundaries. By considering quadratic interactions, this algorithm can better manage nonlinear patterns and provide greater generalizability for more complex datasets [19]. Figure 8 shows the training parameters of the Quadratic SVM algorithm. Compared to the previous algorithm, it achieved a higher value of accuracy in both validation and training data. However, the prediction speed is much lower, and the training time of this algorithm is almost eight times longer than efficient logistic regression. What can affect the advantage of the Quadratic SVM algorithm is the smaller size of the model.

Training Results

Accuracy (Validation)	96.3%
Total cost (Validation)	641
Prediction speed	~1700 obs/sec
Training time	894.33 sec
Model size (Compact)	~2 MB

Test Results

Accuracy (Test)	91.7%
Total cost (Test)	1356

Fig. 8. Quadratic SVM algorithm training parameters

- Cosine KNN – a variant of the classic KNN algorithm that uses cosine similarity. The operation of this classifier is based on measuring the cosine similarity between vectors representing objects in the feature space. Unlike traditional KNN, where Euclidean distance determines similarity, cosine similarity measures the angle between two feature vectors. This is particularly useful when the features have different amplitudes or do not have a fixed scale. Cosine KNN can be used, for example, in text analysis, where feature vectors represent the frequency of words, and cosine similarity facilitates the determination of the similarity degree between documents [20]. Figure 9 shows the training parameters of the Cosine KNN algorithm. It achieved comparably high accuracy compared to previous algorithms, but what distinguishes it is the high prediction speed and small model size.

Training Results

Accuracy (Validation)	94.9%
Total cost (Validation)	886
Prediction speed	~7600 obs/sec
Training time	1513.1 sec
Model size (Compact)	~979 kB

Test Results

Accuracy (Test)	90.0%
Total cost (Test)	1643

Fig. 9. Cosine KNN algorithm training parameters

- Bagged trees – is one of the compound classifiers that uses a set of decision trees to improve classification performance. By using multiple trees and various training data, bagged trees tend to show improved generalization ability and reduced risk of over-fitting compared to a single decision tree. At the time of prediction, each tree in the set predicts the test data and the final class is selected based on a majority vote. In binary classification, most trees unanimously decide to assign the data to one of the classes [21]. Figure 10 shows the training parameters of the bagged trees algorithm. It achieved the highest accuracy of all the validation data. The accuracy of the test data is similar to the rest of the algorithms. What affects the algorithm disadvantage is the exceptionally heavy weight of the model, approximately 16 MB. The algorithm is characterized by an extremely high prediction speed and a short model training time.

Training Results

Accuracy (Validation)	97.1%
Total cost (Validation)	494
Prediction speed	~14000 obs/sec
Training time	1578.7 sec
Model size (Compact)	~16 MB

Test Results

Accuracy (Test)	90.8%
Total cost (Test)	1510

Fig. 10. Bagged trees algorithm training parameters

- Trilayered neural network – is one of the fundamental types of neural networks. It is also called a unidirectional neural network (feedforward neural network) because data flows through the network in only one direction, i.e., from the input layer to the output layer. It consists of three main layers: an input layer, where input data is received; a hidden layer, where feature values are processed by neurons that perform calculations using weights and activation functions; and an output layer, where results are generated based on the processed data from the hidden layer [22, 23]. Figure 11 shows the parameters of the trilayered neural network training. Compared to the previous algorithms, the quality of the neural network is better in almost every respect. The classifier achieved the highest accuracy on the test data, and its prediction speed reached 190 000 obs/sec. What is also worth noting is the minuscule size of the model, reaching only 21 KB.

Training Results

Accuracy (Validation)	96.7%
Total cost (Validation)	579
Prediction speed	~190000 obs/sec
Training time	3638.9 sec
Model size (Compact)	~21 kB

Test Results

Accuracy (Test)	92.3%
Total cost (Test)	1259

Fig. 11. Training parameters of the trilayered neural network algorithm

4.2. Unsupervised learning of algorithms

Unlike classification models in supervised learning, unsupervised learning does not require initial training on labeled data or the assignment of labels. The process of training the algorithm occurs automatically, without user intervention. The data generated by the algorithm can identify those that differ from the overall trend. Models were built using K-means, one-class SVM, and iForest algorithms.

- K-means (Fig. 12) – is called the centroid algorithm; its learning process works iteratively, aiming to find K groups (centroids). Each point is assigned to the nearest centroid, minimizing the sum of squares of the distances between the data and the centroids. The algorithm was configured to display two clusters of data, i.e., normal values and anomalies [24]. The "Display" parameter was set to "final", which means that only the final progress information will be displayed during the execution of the K-means algorithm. The "Distance" parameter specifies the distance metric used to calculate the distance between points and cluster centroids. In this case, after testing several, the most suitable option for the given case was the "cityblock" option, i.e., Manhattan Distance, where the sum of the absolute values of the differences between corresponding features is calculated. The "Replicates" parameter specifies how many times the K-means algorithm should be run with different random initial centroid positions. The final result will be the result of the replication that achieved the best result. This value was chosen experimentally.

```
load('daneTempTest2.mat');
Room1=ytest.Data(500:end,1);
Room2=ytest.Data(500:end,2);
Room3=ytest.Data(500:end,3);
Room4=ytest.Data(500:end,4);
Outside=ytest.Data(500:end,5);

X = [Room1, Room2, Room3, Room4, Outside];

opts = statset('Display','final');
[idx,C] = kmeans(X, 2, 'Distance', 'cityblock', 'Replicates', 50, 'Options', opts);
```

Fig. 12. K-means algorithm classification model

- One-class SVM (Fig. 13) – a variant of the standard SVM algorithm, it is used to identify unusual observations in the data by training the model on only one class of data (without labeled outliers). The model creates a hyperplane (or hyper curve in the case of nonlinear kernels) in a multidimensional feature space that attempts to constrain most data points inside the boundaries (hyperplane) or at the periphery of the boundaries. Points outside these boundaries are considered unusual and may be classified as outliers from the rest of the data [25]. The efficiency of the model depends on the selection of parameters, which were determined experimentally. The "ocsvm" function was used. The function itself, without the selection of parameters, gave unsatisfactory results – only a few samples were classified as anomalies. To make the model more efficient, the data were standardized "StandardizeData = true". The automatic kernel scale function was selected, which is an element in the SVM algorithm and controls the transformation of the data space. "Auto" suggests that the kernel scale will be adjusted automatically. The "Lambda" parameter is a regularization parameter that affects the flexibility of the model decision boundary, also set automatically and the most crucial element influencing the efficiency of the "ContaminationFraction" model, i.e., the expected percentage of anomalies in the training data. Knowing the training data, the parameter was set to 25%. This parameter should be selected carefully because too high a value leads to overfitting the model.

```
load('daneTemp.mat');
Room1=y.Data(500:end,1);
Room2=y.Data(500:end,2);
Room3=y.Data(500:end,3);
Room4=y.Data(500:end,4);
Outside=y.Data(500:end,5);
Xdane = [Room1, Room2, Room3, Room4, Outside];

load('daneTempTest2.mat');
Room1=ytest.Data(500:end,1);
Room2=ytest.Data(500:end,2);
Room3=ytest.Data(500:end,3);
Room4=ytest.Data(500:end,4);
Outside=ytest.Data(500:end,5);
Xtest = [Room1, Room2, Room3, Room4, Outside];

iforestanomaly = iforest(Xdane,ContaminationFraction=0.25,NumLearners=150);
isanomalyiforest = isanomaly(iforestanomaly,Xtest);
```

Fig. 13. One-class SVM algorithm classification model

- iForest (Fig. 14) – is one of the commonly used unsupervised learning algorithms belonging to the family of decision trees. It is one of the most popular models for detecting anomalies in data, i.e., observations that deviate significantly from the predicted norm. The algorithm builds decision trees based on the features of the data and identifies those points that do not fit the rest of the set. The syntax of the algorithm is quite similar to that of a one-class SVM. Equally important is selecting an appropriate percentage of anomalies in the training data. Since the training data set remained constant. In addition, a parameter was introduced to determine the number of trees (learning classifiers) in the iForest algorithm. Each tree is a certain number of samples from which the model attempts to isolate anomalies. Automatically, this value is set to 100 trees, but due to the large dataset, it was increased to 150, which improved the classification result by several percent [26].

```
load('daneTemp.mat');
Room1=y.Data(500:end,1);
Room2=y.Data(500:end,2);
Room3=y.Data(500:end,3);
Room4=y.Data(500:end,4);
Outside=y.Data(500:end,5);
Xdane = [Room1, Room2, Room3, Room4, Outside];

load('daneTempTest2.mat');
Room1=ytest.Data(500:end,1);
Room2=ytest.Data(500:end,2);
Room3=ytest.Data(500:end,3);
Room4=ytest.Data(500:end,4);
Outside=ytest.Data(500:end,5);
Xtest = [Room1, Room2, Room3, Room4, Outside];

OneClassSVM=ocsvm(Xdane,StandardizeData=true,KernelScale="auto",Lambda="auto",ContaminationFraction=0.25);
isanomalysvm = isanomaly(OneClassSVM,Xtest);
```

Fig. 14. iForest algorithm classification model

Unlike supervised learning algorithms, which produce a constant result each time they are run, unsupervised learning algorithms perform a series of calculations in the classification process, and their result always differs by a few percent. This can negatively affect the usability of these algorithms.

4.3. OPC server

MATLAB-Simulink provides many possibilities to compare the performance of algorithms, first of all, you can freely "damage" model elements to detect any potential damage and teach algorithms to recognize it, which is not the case with real systems. However, this program also has several difficulties, which are often impossible to solve. First of all, it is a real-time simulation. To prepare training and test data, it is necessary to finish the simulation and then save it in a file, which is later used to train algorithms, which forces a fixed simulation time.

The paper proposes a solution to the above problem using a communicator, which is the OPC server. However, this requires the use of Statistic and Machine Learning Toolbox blocks because all simulation-related activities for test data are performed directly in Simulink without using a script in MATLAB. This toolbox offers only blocks related to supervised machine learning. The scope of these algorithms is diminutive, which is not the case in the previously discussed Classification Learner. First, the procedure is standard, i.e., a training set with class labels of possible failures should be prepared. For the needs of the paper, a simulation with one failure was created, because at this point the algorithms will not be compared, and only a way to solve the simulation problem in real mode will be shown. Attempts were made to prepare a model with all possible failures, but during real-time simulation, despite the lack of compilation errors, the time was very slow, until finally, the simulation stopped at the level of microseconds, which could be caused by the program version or internal limitations of the computer system.

After preparing the test data and labeling the time samples, the classes were changed to numerical values to avoid later errors in the data type in the database. The classification model was trained using the "fitcecoc" function, which is used to train a multiclass classifier using the error-correcting output codes (ECOC) coding strategy. This technique solves the multiclass classification problem by transforming it into a series of binary classification problems. The ECOC strategy is based on the fact that each class is represented as a combination of the binary outputs of the classifiers. Each of these binary classifiers is responsible for recognizing one pair of classes: the "target class" (the class the user wants to recognize) and the "other class" (the remaining classes that are not the target class). The results of these classifiers are then combined to obtain the final decision.

Then, a preliminary simulation with a damaged furnace was prepared for the test data to check the algorithm performance. The available ClassificationECOC Predict block was used and Scope was connected to it (Fig. 15).

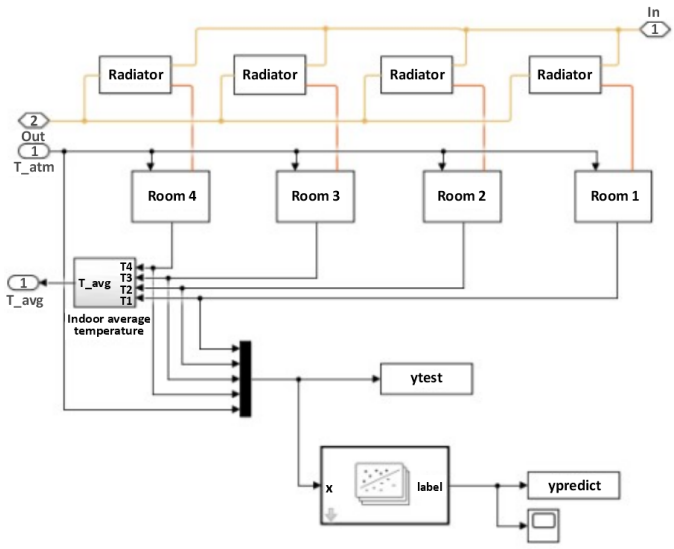

Fig. 15. Real-time simulation using the predict block

Figure 16 shows the response of the classification model to the simulation of the heating model operation, which indicates correct fault detection.

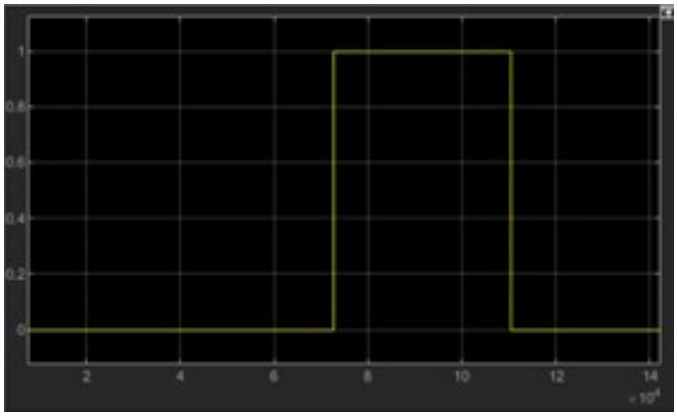

Fig. 16. Classification model response

4.4. Data import into InfluxDB and PostgreSQL databases

The InfluxDB [27, 28] database is one of the most popular open-source time series databases. It was designed as a solution for projects that generate substantial amounts of data over time, especially for smart metering. For this reason, it was chosen as the most suitable tool for the issues addressed in the article.

A PostgreSQL [28, 29] database was used for the stage using the OPC server. A relational database was used because of the ease of communication of such a database with MATLAB.

4.5. Visualization in the Grafana system

Grafana [30] is an open-source tool for data visualization and monitoring that visualizes the results obtained. It facilitates the creation of attractive data visualizations such as graphs, bar charts, dot plots, pie charts, heat maps, and much more. This allows users to understand better and analyze data. Grafana supports various data sources, such as InfluxDB and PostgreSQL databases.

5. RESULTS

5.1. Simulated time operation

After the data was stored in the database, InfluxDB was communicated to the Grafana visualization system. The simulation ran as a function of time with different combinations of faults and the responses of the supervised learning algorithms are shown in Fig. 17. At first glance, the neural network and quadratic SVM performed best with the classification. At some points, they misclassified the damage at the beginning of the detection, while after providing more measurements, they eventually recognized the correct damage. The other algorithms happened to detect damage in the correct data, which can confuse real objects and waste time detecting damage that does not occur. All algorithms mostly confused the values for open windows and classified them as damaged radiators. This is due to the similar behavior of the temperature values for these types of damage.

The algorithms managed the heater damage effectively, as such damage results in temperatures in all rooms dropping to reach the outside temperature, which cannot be mistaken for any other fault. Other than that, the operation of the algorithms is relatively correct - they reacted to every anomaly present in the data. To better illustrate the performance of the algorithms, a graph of the misclassified data against the manual description of the data sample labels is also shown (Fig. 18). The fewest errors were observed in the performance of the trilayered neural network algorithm.

Another visualization was performed for the unsupervised learning algorithms. The results of the one-class SVM and iForest algorithms were compared simultaneously, as the syntax of their models is similar. They produced comparable results, with a slight advantage in favor of one-class SVM (Fig. 19). They reacted to every anomaly present. In contrast, with most of the anomalies present, they had a problem making a final decision, which is why, as seen in the graphs, the markers jump from 0 to 1. The performance of the K-means algorithm did not give satisfactory results. It detected about 50 percent of anomalies in the data, only where the temperature in the facility reached the outside temperature. When it fell by a few °C, e.g., due to open windows, the algorithm did not classify this as a fault.

However, it is essential to note that for the one-class SVM and iForest algorithms, a definite disadvantage in building classification models based on these algorithms is the need for a percentage of anomalies present in the data. Without this, the performance of the algorithms was decidedly inferior to K-means – they classified only a few data samples for the damaged heater as abnormal.

Fig. 17. Diagram showing the result of the supervised learning algorithms

Fig. 18. Sum of incorrect classifications of supervised learning algorithms

Fig. 19. Diagram showing the result of the one-class SVM algorithm

5.2. Real-time operation

For real-time simulation, the alerting offered by Grafana has also been added to the data visualization. This is the process of monitoring and detecting anomalies or abnormal behavior in systems and metrics and generating notifications or alerts when certain thresholds or conditions are exceeded. As for previous visualizations, Grafana was connected to a database, this time, PostgreSQL. Alerting was then configured, depending only on the values returned by the classification model. A value of 0 indicates no damage in the system, while 1 indicates damage. Therefore, a condition was set so that for all values above 0, an alert about a detected anomaly is triggered. Figure 20 shows the real-time simulation run along with the response of the classification model. The system worked as expected – it correctly detected the anomaly at a time that agreed with the simulation run.

6. CONCLUSIONS

Analyzing the obtained results, the popular supervised learning algorithms remain reliable: the trilayered neural network and support vector machine SVM (Quadratic SVM). However, two unsupervised learning algorithms – one-class SVM and iForest – achieved comparable results.

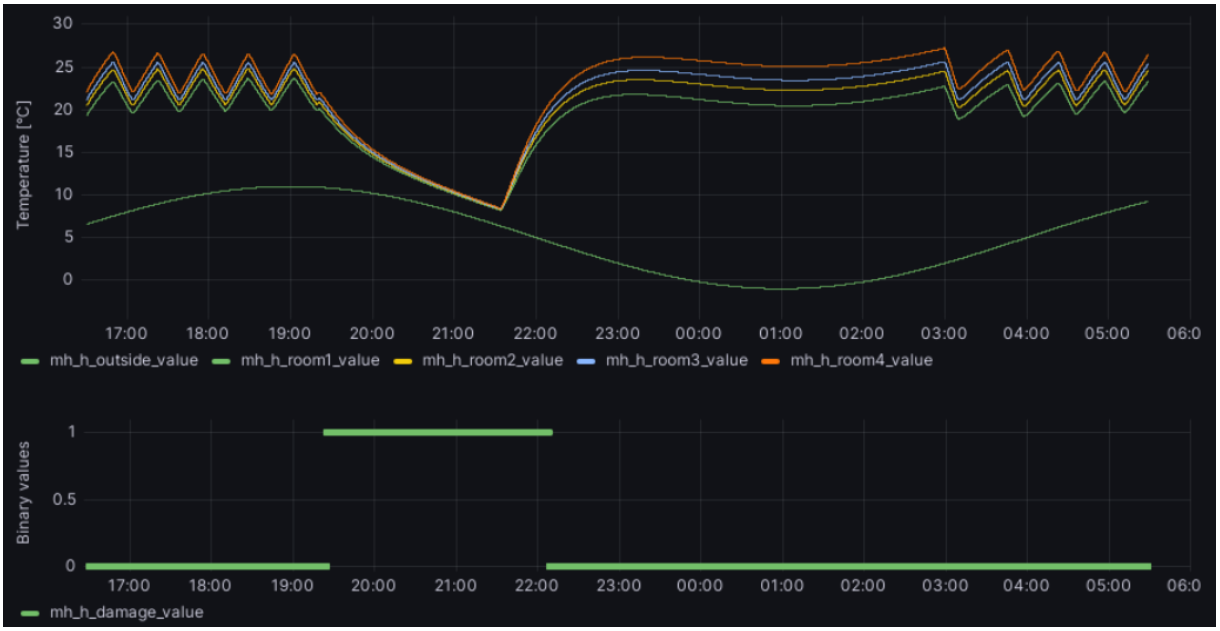

Fig. 20. Real-time simulation diagram

The goal of supervised learning is to train a model to predict or classify data based on the labels provided in the training data. The model is trained by examples to assign input data to the appropriate categories accurately. Such learning algorithms require labeled training data that contains input-output (feature-label) pairs. They find their uses in classification, regression, and anomaly detection where there are clearly defined labels or predictive goals. The ability to evaluate is relatively easier than unsupervised learning algorithms because there are comparison labels. Measures such as accuracy, precision, sensitivity, and confusion matrix are commonly used for evaluation [7, 9, 31].

Unsupervised learning aims to discover hidden patterns, structures, or relationships in data without using labels. There are no labels, and unsupervised algorithms try to group similar data or reduce the dimensionality of the data. Activities take place on unlabeled or unlabeled data. They are often used to group data, reduce dimensionality, detect anomalies, and generate data structures. Assessing their effectiveness is more difficult because there are no comparative labels. Other evaluation methods must be used, such as internal measures [32].

The choice between supervised and unsupervised machine learning depends on the data characteristics, the availability of labels, and the purpose of the analysis. In practice, these two methods can often be combined to obtain better results and a deeper understanding of the data. In this case, supervised algorithms work better when dealing with an object consisting of several rooms. They detect specific damage in a specific room, which will save time wasted on locating damage in the case of unsupervised algorithms. They are also much more accurate; they always detect an anomaly and, although at the beginning, the measurements delivered frequently were wrong about the type of damage, after providing more samples, they classified correctly.

The advantage of unsupervised learning models is the classification time. In the case of this facility, where the data determines the temperature values in the rooms, it affects the thermal comfort of the inhabitants. Therefore, it is important to react as quickly as possible. This time is only disturbed by the lack of knowledge in which room the damage occurred. Unsupervised algorithms can discover hidden patterns in new data, which is impossible with supervised learning. However, due to the lack of labels, it is more difficult to understand why the model makes certain decisions [33].

An exciting aspect of the work was an attempt to directly communicate the MATLAB Simulink program with the database so that it was possible to control the temperature values in real time constantly. For this purpose, an OPC server was used to function as an intermediary in data transfer. The OPC server collected data from the Simulink process, then processed it into specified data types and then sent it to a previously selected database communicated with the server. However, this method does not allow you to test all the algorithms discussed by everyone due to the lack of appropriate blocks in Simulink. The complexity of the simulated processes and the substantial number of labels in the training data also slowed down and blocked the system [15].

6.1. Possible development

The topics discussed in the above article provide ample opportunities for further development. First of all, application in a real environment. Selected fault detection algorithms can be tested in a real building automation environment. By carefully analyzing test results, operating conditions, confounding factors, and potential challenges in implementing algorithms can be considered. It is also possible to further combine the algorithms to increase their efficiency and integrate with existing systems, adapt to various devices, and monitor performance in the long run. Further tests of the algorithms on different devices will facilitate seeing how the algorithms behave in different conditions and environments. In the further development of this issue, it will be essential to conduct a cost-benefit analysis of the implementation of the developed fault detection algorithms and compare the estimated benefits with the implementation and maintenance costs to assess the effectiveness of the technology in practice.

REFERENCES

[1] V. Chandola, D. Cheboli, and V. Kumar, "Detecting anomalies in a time series database," Computer Science & Engineering (CS&E) Technical Reports, 2009.

[2] K. Różanowski, "Sztuczna inteligencja rozwój, szanse i zagrożenia," *Zeszyty Naukowe Warszawskiej Wyższej Szkoły Informatyki*, vol. 2, no. 2, pp. 109–135, 2007, (in Polish).

[3] H. Chaouch, A.S. Bayraktar, and C. Çeken, "Energy Management in Smart Buildings by Using M2M Communication," *International Istanbul Smart Grids and Cities Congress and Fair (ICSG)*, Istanbul, Turkey, 2019, pp. 31–35, doi: 10.1109/SGCF.2019.8782357.

[4] P. Borkowski, B. Bolanowski, and E. Walczuk, *Inteligentne systemy zarządzania budynkiem*, Wydawnictwo Politechniki Łódzkiej, 2011

[5] M. Szeliga, "Data Science i uczenie maszynowe", *Wydawnictwo Naukowe PWN*, 2017.

[6] A. Duraj, E. Korzeniewska, and A. Krawczyk, "Classification algorithms to identify changes in resistance," *Prz. Elektrotechn.*, vol. 91, no. 12, pp. 80–83, 2015.

[7] V. Nasteski, "An overview of the supervised machine learning methods," *Horizons B*, vol. 4, no. 51–62, p. 56, 2017.

[8] F. Hahne, W. Huber, R. Gentleman, S. Falcon, R. Gentleman, and V.J. Carey, "Unsupervised machine learning," in *Bioconductor Case Studies*, Springer, 2008, pp. 137–157.

[9] P. Fatyga and R. Podraza, "Klasyfikacja danych – przegląd wybranych metod," *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej*, 2010, (in Polish).

[10] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv. (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[11] P. Lula, "Wykorzystanie sztucznej inteligencji w prognozowaniu," *Seminarium Statsoft "Prognozowanie w przedsiębiorstwie"*, Warszawa, Poland, 2000, pp. 39–69, (in Polish).

[12] M. Walesiak, "Wybór grup metod normalizacji wartości zmiennych w skalowaniu wielowymiarowym," *Prz. Stat.*, vol. 63, no. 1, pp. 7–18, 2016, (in Polish).

[13] "MATLAB – MathWorks". [Online]. Available: https://www.mathworks.com/products/matlab.html, [Accessed: 13. Sep. 2024].

[14] "House Heating System - MATLAB & Simulink". [Online]. Available: https://www.mathworks.com/help/simulink/ug/model-a-house-heating-system.html, [Accessed: 13. Sep. 2024].

[15] "KEPServerEX/ThingWorx Kepware Server," INEE Sp. z o.o. [Online]. Available: https://inee.pl/produkty/kepserverex-thingworx-kepware-server, [Accessed: 13. Sep. 2024].

[16] "Simscape." Mathworks. [Online]. Available: https://www.mathworks.com/products/simscape.html [Accessed: 13. Sep. 2024].

[17] "Simulink – Simulation and Model-Based Design." Mathworks. [Online]. Available: https://www.mathworks.com/products/simulink.html [Accessed: 13. Sep. 2024].

[18] M. Mamczur, "Jak działa regresja logistyczna?," 20/03/2023. [Online]. Available: https://miroslawmamczur.pl/jak-dziala-regresja-logistyczna/, [Accessed: 13. Sep. 2024].

[19] M. Jukiewicz, "Wykorzystanie maszyny wektorów nośnych oraz liniowej analizy dyskryminacyjnej jako klasyfikatorów cech w interfejsach mózg-komputer," *Poznan Univ. Technol. Acad. J.-Electr. Eng.*, vol. 79, pp. 25–30, 2014.

[20] T. Pamuła, "Ocena predykcji natężenia ruchu pojazdów z użyciem algorytmu kNN-najbliższych sąsiadów i sieci neuronowych," *Logistyka*, vol. 6, pp. 8313–8320, 2014, (in Polish).

[21] E. Szpunar-Huk, "Pozyskiwanie wiedzy z danych przy wykorzystaniu klasyfikatorów złożonych," *Prace Naukowe Akademii Ekonomicznej we Wrocławiu*, pp. 268–279, 2005, (in Polish).

[22] J. Tchórzewski and K. Leszko, "Artificial neural networks as models neuronal electronic state offices," *Inf. Syst. Manag.*, vol. 4, no. 3, pp. 219–227, 2015.

[23] P. Lula and R. Tadeusiewicz, *Introduction to Neural Networks*. Kraków: StatSoft Polska Sp. z o.o., 2001.

[24] B. Całka, "Grupowanie nieruchomości lokalowych za pomocą metody K-średnich," *Prz. Geodezyjny*, vol. 84, no. 11, pp. 3–8, 2012, (in Polish).

[25] Y. Chen, X.S. Zhou, and T.S. Huang, "One-class SVM for learning in image retrieval," in *Proc. 2001 International Conference on Image Processing*, vol. 1, pp. 34–37, 2001.

[26] X. Zhao, Y. Wu, D.L. Lee, and W. Cui, "iForest: Interpreting random forests via visual analytics," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 1, pp. 407–416, 2018.

[27] "InfluxDB Time Series Data Platform," InfluxData. [Online]. Available: https://www.influxdata.com/, [Accessed: 13. Sep. 2024].

[28] S. Włostowska, J. Szabela, A. Chojecki, and P. Borkowski, "Comparison of SQL NoSQL and TSDB database systems for smart buildings and smart metering applications," *Prz. Elektrotechn.*, vol. 99, pp. 7–12, 2023.

[29] "PostgreSQL," P.G.D. Group. [Online]. Available: https://www.postgresql.org/, [Accessed: 13. Sep. 2024].

[30] "Grafana: The open observability platform," Grafana Labs. [Online]. Available: https://grafana.com/, [Accessed: 13. Sep. 2024].

[31] L. Deri, S. Mainardi, and F. Fusco, "TSDB: A compressed database for time series," in I *Traffic Monitoring and Analysis: 4th International Workshop, TMA 2012*, Vienna, Austria, 2012, pp. 143–156.

[32] Y. Ma, Q. Zhang, J. Ding, Q. Wang, and J. Ma, "Short term load forecasting based on iForest-LSTM," *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2019, pp. 2278–2282.

[33] D. Conway, *Uczenie maszynowe dla programistów*, Helion, 2014, (in Polish).