Methodology for Predicting Failures in a Smart Home based on Machine Learning Methods

Viktoriia Zhebka[1], Pavlo Skladannyi[2], Serhii Zhebka[1], Svitlana Shlianchak[3], and Andrii Bondarchuk[1]

[1] State University of Information and Communication Technologies, 7 Solomenskaya str., Kyiv, 03110, Ukraine
[2] Borys Grinchenko Kyiv Metropolitan University, 18/2 Bulvarno-Kudriavska str., Kyiv, 04053, Ukraine
[3] Volodymyr Vynnychenko Central Ukrainian State University, 1 Shevchenka str., Kropyvnytskyi, 25006, Ukraine

Abstract

The article presents a platform for predicting failures in a smart home. A detailed algorithm of the predicting platform has been described. An algorithm for integrating the fault prediction platform into the smart home system has been developed. An algorithm for the functioning of a smart home with a failure prediction program based on machine learning has been presented. The software has been developed using the JHipster1 generator and the Java programming language. The use of machine learning methods in a smart home system expands its ability to analyze large amounts of data and identify patterns that may precede failures. This allows the system to predict possible problems and respond to them in advance. The use of preventive measures allows the system to automatically take measures to avoid failures, such as automatically adjusting the operation of devices or performing backups based on predictions.

Keywords

Failures, machine learning methods, predicting, methodology, information technology, IoT, smart home.

1. Introduction

The rise of smart homes, facilitated by advancements in the Internet of Things (IoT) and smart technologies, offers convenient automation and control over various household systems like lighting, heating, security, and energy efficiency [1]. Yet, with increased complexity comes a higher risk of malfunctions or issues [2]. Factors such as network instability, software glitches, and faulty devices can create unpredictable scenarios, compromising both the functionality and security of a smart home [3].

Anticipating these failures has become a pertinent concern, prompting the application of machine learning techniques for prediction. By leveraging machine learning algorithms, it becomes feasible to sift through vast datasets, identifying deviations and trends indicative of impending failures. This proactive approach enables preemptive measures to mitigate potential problems before they manifest.

Presently, most smart home failure prediction relies on reactive analysis, meaning anomalies or failures are detected after they occur, making prevention challenging. However, employing machine learning methods such as classification, clustering, and prediction algorithms holds promise in developing systems capable of forecasting failures in advance.

These systems utilize data from various sources including sensors, IoT devices, energy consumption records, etc., to discern patterns and anomalies preceding disruptions [4, 5]. Armed with this insight, machine learning systems construct predictive models that respond to specific signals or deviations from

CPITS-2024: Cybersecurity Providing in Information and Telecommunication Systems, February 28, 2024, Kyiv, Ukraine
EMAIL: viktoria_zhebka@ukr.net (V. Zhebka); p.skladannyi@kubg.edu.ua (P. Skladannyi); szhebka@hotmail.com (S. Zhebka); shlanchaksveta@gmail.com (S. Shlianchak); dekan.it@ukr.net (A. Bondarchuk)
ORCID: 0000-0003-4051-1190 (V. Zhebka); 0000-0002-7775-6039 (P. Skladannyi); 0009-0007-4620-9888 (S. Zhebka); 0000-0001-9893-5709 (S. Shlianchak); 0000-0001-5124-5102 (A. Bondarchuk)

normal operation, alerting users to potential issues or taking corrective actions autonomously [6–16].

While still a nascent field, ongoing research in this domain aims to enhance smart home control systems by enabling them to foresee and forestall potential failures, thereby enhancing reliability and security for users [17–24].

2. Research Results

A fault prediction platform assumes the existence of an IoT platform, as shown in the upper part of Fig. 1. In particular, it assumes the existence of two data sources, one for operational data and the other for historical and static data, which is reminiscent of the Lambda architecture, the main idea of which is that data is first processed in real time (operational stream) and then sent for storage and analysis (historical stream). This allows the system to work efficiently in both real-time and historical contexts, providing analysis and conclusions based on the accumulated data.

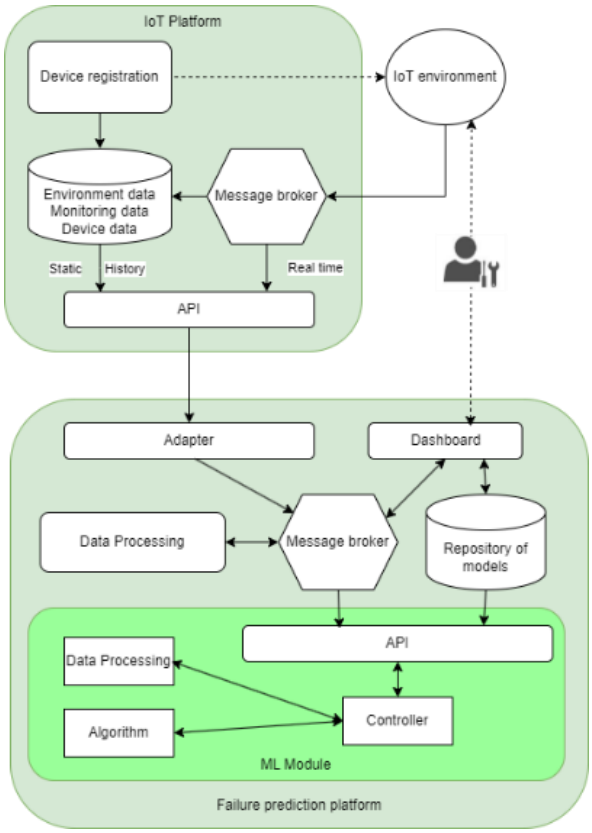

Figure 1: Architecture of the failure prediction platform. Solid lines indicate control and data flow and dashed lines for manual tasks

An IoT platform receives data from the IoT environment, which can be obtained from sensors and actuators (environmental data) or directly from devices, such as CPU and memory usage (monitoring data). In addition, the IoT platform stores general information about devices (device data). Device data are obtained when a device is registered in the IoT environment using, for example, a device registration component. The process can be automated to a certain extent, but it still requires manual intervention. IoT platforms usually use a message broker that receives data from all the different devices in the IoT environment. It can be used as a source of real data. In addition, there may be a database that permanently stores the received data from the environment and the static data of the devices. Usually, these two main components are part of IoT platforms, as seen in the example of a multi-purpose binding and provisioning platform that implements the Mosquito broker and the MongoDB database. However, it is important to secure only these two data sources regardless of the types of components used in the background mode. The IoT platform can explicitly provide the API component with access to its data, or it can establish a direct connection to the broker and database. The lower part of Fig. 1 shows the network composition, which should be able to establish connections to different types of IoT platforms and data sources. The adapter component decouples the fault prediction platform from specific implementations and receives the data to be used by the models. The message broker is used to coordinate and permanently store data after individual processing steps.

After the data has been received by the adapter, they are sent to the broker and stored in a specific group without changes. In the next step, the data can be processed in the data processing component before they will be used by the models. This step can be standardized using a predefined data template, enriched with missing information, converted to a specific format, or adapted in any other way. Standardizing data using a template leads to data consistency, which further facilitates the implementation of models. The processed data are sent and stored back to another group where they can be reused by the models. This component is necessary when working with heterogeneous data and performs the processing

required by all models. Assuming that the data comes in the form of time-stamped records, it can be used to extract meaningful information from the time stamp, such as hour of the day, day of the week, holiday, etc. Another potential option of usage, in the case of split device groups, is to filter the data by device and send it to the appropriate group.

Each machine learning model is integrated as a module consisting of four main components: API, data processing component, controller, and the machine learning algorithm itself. The API has the same composition for all modules and consists of a message source and two message consumers. There is a data consumer that consumes pre-processed data and a command consumer that consumes commands sent from the dashboard. The message source sends model results and status or progress information to the message broker. The messaging interface separates the machine learning modules from the platform. However, there is a common data processing component in the previous step. Therefore, an additional data processing component is integrated into each module to prepare data for a specific algorithm. The algorithm itself is the core of the module. It goes through a training process that results in a model used for forecasting. The algorithm can be implemented in different ways and with different libraries, but it must provide an interface for training and predicting. Finally, all components are managed by a controller that performs several functions. It processes commands, received from the outside and performs the appropriate actions, receives data, sends them to the processing component, and then to the algorithm, if necessary. The controller can implement automated hyperparametric tuning to find the optimal parameters, i.e. the most optimal algorithm configuration based on the data provided. The controller also stores and loads data into a database or file system. It can implement a scheduler to periodically create backups, retrain the algorithm if online learning is not possible, or reconfigure the parameters. It can evaluate the current processing status, and send model predictions to the message producer.

A model repository describes a general-purpose storage shared by all models. The repository can be a database, a file system, or a combination of both. The main purpose is to store, describe, and configure algorithms, and hyperparameters after tuning, or complete machine learning models, i.e. internal parameters after training. It can also store any other information related to the model. It is accessed through the API components of the modules and the dashboard.

Once the machine learning model is ready and starts making predictions, the corresponding module sends the results to the group it has defined. The dashboard collects all the predictions from different models, processes them, and visualizes the results. In addition to the groups for predicting, each module has a group dedicated to commands sent from the dashboard. In addition, there may be additional groups that are used by modules to send information about the status and progress of work. The dashboard should provide the ability to interact with the modules and perform individual steps manually. These steps can include: importing history, starting automatic hyperparameter tuning, training individual algorithms, starting and stopping individual forecasting models, saving and loading models, etc. In addition, the dashboard has access to the model repository to retrieve and display relevant model information in the user interface. The dashboard can also be used to insert information during the model registration process. Finally, the attendant monitors the dashboard and performs repairs or replacements in the IoT environment. An alert system can also be implemented to inform technical support if a device fails or reaches critical thresholds based on predictions.

3. Integration of a Fault Prediction Platform into a Smart Home System

At the time the predictive platform is connected to the IoT platform, the IoT platform may already exist and collect some data. The data can be used for initial algorithm training.

The prediction platform allows us to retrieve old data from the database, go through a general processing stage, and save it by a specific topic. After that, the process of training algorithms can be started individually. Initial training is important because it allows you to smoothly start model forecasting with meaningful results right from the start. But it can also be skipped, especially if no data has been collected up to this point. This will lead to

instant training, either in the form of online training or retraining at regular intervals. Another important advantage of old data is that they can be used to customize the scalers required by some models. Scalers typically require the calculation of internal parameters such as minimum, maximum, or average values before they can be applied to the incoming live data. Performing this configuration on the go is a more cumbersome process that is likely to lead to poor results in the beginning. In addition to these benefits, there is a limitation that must be considered. The potentially huge amount of data that could be stored in the IoT platform's database could not be loaded into memory immediately. Nevertheless, in most cases, processing data in batches or even selecting a portion of the data can be enough to set up scalers as well as train algorithms to provide meaningful results in the beginning.

In the second step, before any prediction model is made, the forecasting platform establishes a connection to the message broker on the IoT platform and starts receiving live data. Again, the data go through a processing stage and are assigned to a specific group. At this stage, predicting can be started separately for each model. The module receives data and processes them further if necessary. Then the model makes a prediction, and the controller sends it to the appropriate prediction group. Finally, the predictions are extracted and visualized in a dashboard. The data passing through the system is always saved for each group, except for groups designated for teams and status or progress information.

So, the entire algorithm for integrating a fault prediction platform into a smart home system can be represented in four steps:

Step 1. Connection to the IoT system and usage of the available data:

- Connecting the predicting platform with the IoT system.
- Using available data for initial algorithm training.

Step 2. Obtaining new data:

- Connecting the forecasting platform to the message broker on the IoT platform.
- Receiving and processing new live data.
- Structuring data by a specific topic.

Step 3. Forecasting with models:

- Starting forecasting using individual models.
- Receive, process, and transfer forecasts to the appropriate forecast groups.

Step 4. Visualization and storage of results:

- Extracting and visualizing predictions in a dashboard.
- Storing data for further use and updating models.

This algorithm allows us to work efficiently with existing data and obtain new data to continuously improve predictive models.

The process of integrating an information system into a smart home begins with the analysis of functional requirements and the detailing of needs (Fig. 2). The next stage is the development of specialized modules that ensure the interaction of the information system with the smart home through the establishment of specific links and interfaces.

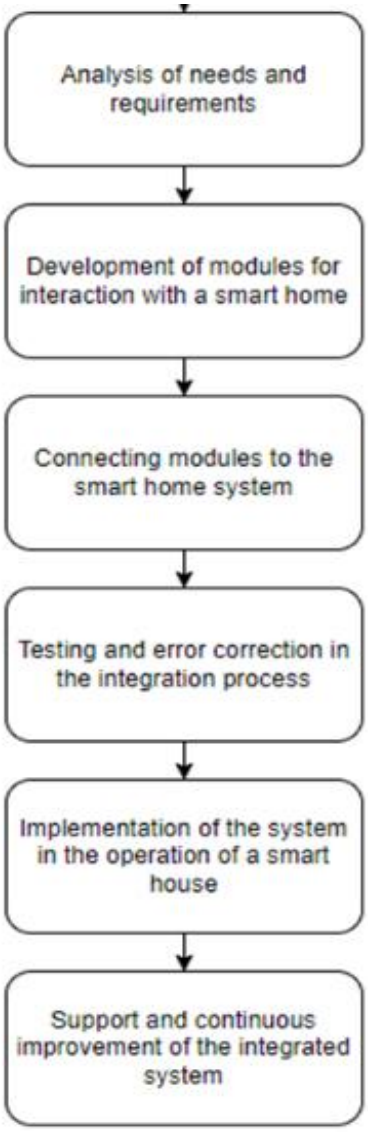

Figure 2: The process of integrating an information system into a smart home

These modules are being integrated with existing devices and systems in the smart home, using networks and communication protocols for data exchange and control. An important step is to test the integrity and efficiency of the interaction, as well as to correct possible errors and malfunctions.

After the system is successfully integrated into the smart home, the implementation process takes place, when the system becomes an active part of the home environment. However, this is only the beginning: further support, optimization, and continuous improvement of the system play a key role in ensuring its long-term and efficient operation in the smart home, adapting to changing needs and conditions.

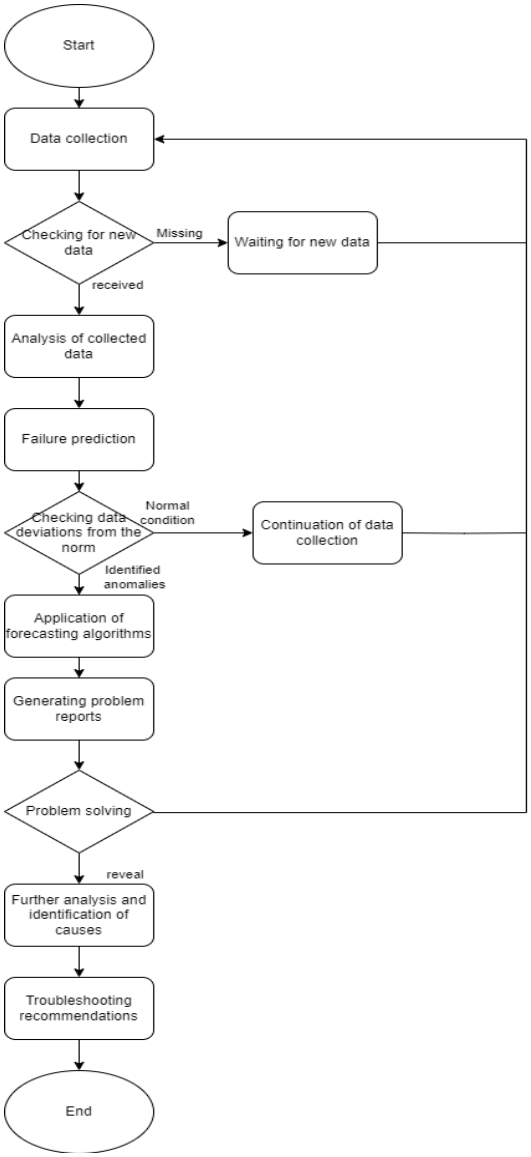

Figure 3: Smart home functioning algorithm with a machine learning-based fault prediction program

Fig. 3 shows a block diagram of the smart home functioning algorithm with a machine learning-based fault prediction program. The proposed algorithm can be divided into the following blocks:

1. Start—usually means the beginning of a sequence of actions.
2. Data collection is the block responsible for checking and collecting new data. If there are no data, the system waits for new data. If the data are received, are being analyzed.
3. Failure prediction—this is where the data are checked for abnormalities. If everything is normal, the system continues to collect data. If anomalies are detected, prediction algorithms are used to identify problems and generate notifications about them.
4. Problem-solving: if problems are detected, the system conducts further analysis, and determines the causes and recommendations for their solution.

Each unit is responsible for a specific part of the functionality of the smart home fault prediction system. They help in collecting, analyzing, and processing data, identifying possible problems, and providing recommendations for their elimination. Such a system allows the smart home to be more autonomous and respond to potential failures or problems in real-time. After at least one day of data collection, the system can start learning and identifying patterns. The dataset is divided into training and evaluation sets. The training set makes up 75% of the total data, and the remaining 25% is the evaluation set. This split allows us to estimate the model's accuracy. Once a model is created, it is saved and can be used to evaluate new data. The system continues to collect data about the environment. The diagram of the flow in Fig. 4 shows the entire system workflow when integrating data collection with model implementation. The "motion", "light", "temperature and humidity", "device states", and "main script" blocks represent the main process of monitoring the environment in the workflow. Monitoring is used both for data collection, for the "action in the environment" block, "data sets", and for making system decisions, indicated by the "every minute, movement or change of the device", "LSTM model", and "environment control" blocks.

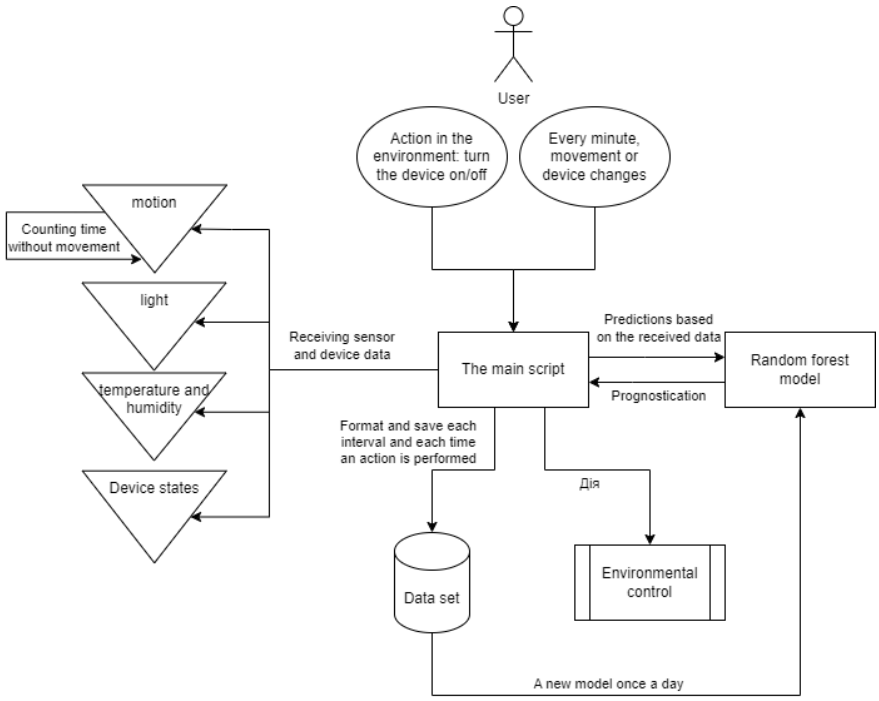

Figure 4: Diagram of the flow

4. Software Description

The failure prediction platform has been implemented using the JHipster1 generator. JHipster creates a complete web application that provides user management, a responsive user interface, monitoring, and other benefits. The prototype implementation uses a Java backend, a Spring2 framework, and an Angular3 interface. In addition, it stores data in a MongoDB database and uses Kafka4 as a message broker. The failure prediction dashboard is shown in Fig. 5.

The smart home failure prediction software uses data sources in the form of a multi-purpose binding and provisioning platform, including MongoDB and Mosquitto. The proposed development includes a multi-purpose platform adapter that creates a connection to the MongoDB database and a separate connection to the Mosquitto broker. To show the learning and prediction processes, the generated dataset was integrated into the database and the broker. The panel shows the models and their functions, as well as two common functions that apply to all models.

The first general function is responsible for importing old data from the database and storing it according to a specific group without changes. For this purpose, Kafka is used, a fast streaming platform that provides high-quality data storage and allows you to create and exchange streaming data between different systems and modules, providing the ability to process and analyze this data in real-time.

After that, the data are processed in a common component implemented in Java. Given the use of a generated dataset, it is already preprocessed, except for the removal of MongoDB identifiers from each record. This is done by defining a data template without a MongoDB ID and matching the input data accordingly when removing the ID.

The processed data are again stored in another group. The second general function activates the monitoring process by establishing a connection with the Mosquito broker and receiving the current data. The real-time data follow the same processing path as the old data until they are saved again after processing. Unfortunately, there is currently no synchronization mechanism and these two functions can create duplicates if they are performed at the same time.

A machine learning model can be integrated into a fault prediction platform in three steps. A snapshot of the architecture of the fault prediction platform and the components involved in integration is shown in Fig. 5.

When it comes to integrating a machine learning model into a fault prediction platform, there are three steps involved. First, the model is developed as a separate machine-learning module. This module has a built-in Kafka

producer to deliver predictions and two Kafka consumers to exchange data and commands. It is important to choose adequate names for the groups used in this context.

At the next stage, the model is registered in the platform dashboard, providing various information about the model: name, description, type, and names of groups used for communication and transmission of forecasts. The model data are stored in the model repository, which allows us to conveniently manage, view, edit, and delete them. The model registration area provides a convenient interface for managing registered models and adding new ones.

At the final stage of the module development, it is independently integrated, which makes it possible to interact with the dashboard through the Kafka system. For the demonstration, we used a simple Long Short-Term Memory model with a sequence of one failure value. This model has been chosen because it supports online learning and time series prediction. An equally important indicator of model selection is the analysis of the effectiveness of machine learning methods conducted in the second section, which showed that the Long Short-Term Memory model gives the best performance in the predicting process. This module is implemented in Python and integrated using the steps described above. It uses an abstract API component with Kafka producers and consumers, as well as predefined abstract methods to create the interface. This ensures a uniform API implementation across all models. In addition, for the Long Short-Term Memory model, a special data processing component has been created that transforms functions, prepares sequences, and uses a scaler to normalize data. The last two components of the model are the controller and the algorithm. The controller processes commands and performs appropriate actions, managing the components in the model, as well as starting and stopping the algorithm's learning and prediction process.

When the dashboard is initialized, all existing models and their forecast histories are loaded, as shown in Fig. 6. Each model has four interrelated functions. To train the models, they move to the group with the processed data. In this implementation, a limited number of data records are used to load all the data simultaneously. However, records are being received in limited quantities or batches. Once training is complete, predicting can begin. If online training is provided for a particular model, forecasting can be started instantly. In addition, each model, along with the corresponding scaler, can be saved as files and retrieved at any time.

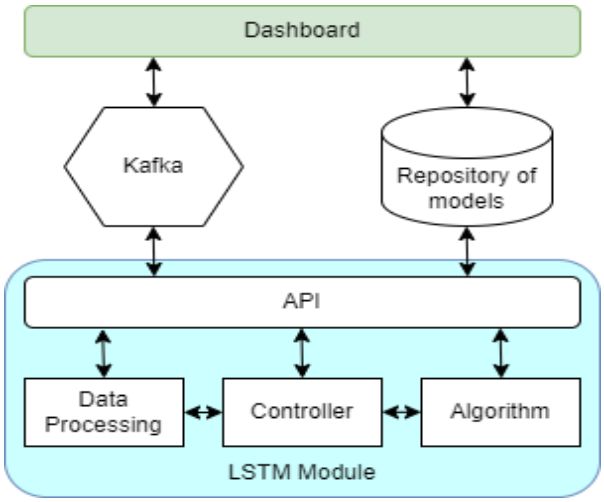

Figure 5: Failure prediction platform architecture and components

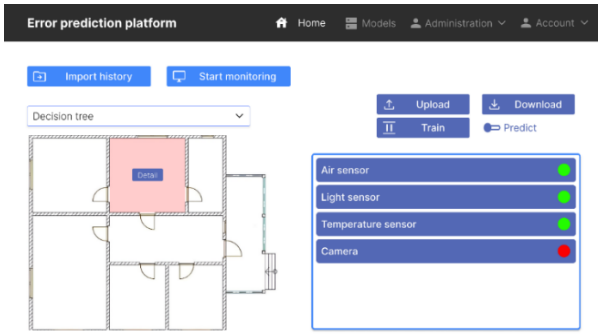

Figure 6: Developed software

The basic algorithm of the information system can be described as follows:

1. Initialization and loading of the models:
   - When the system starts up, all available models and their predictive histories are loaded.
   - This is done for further analysis and use of these models in the decision-making process.
2. Model training:
   - Models can be trained by moving to the appropriate group with pre-processed data.
   - A limited number of data records are used for simultaneous training, but the ability to receive data in limited quantities or batches ensures the efficiency of the process.

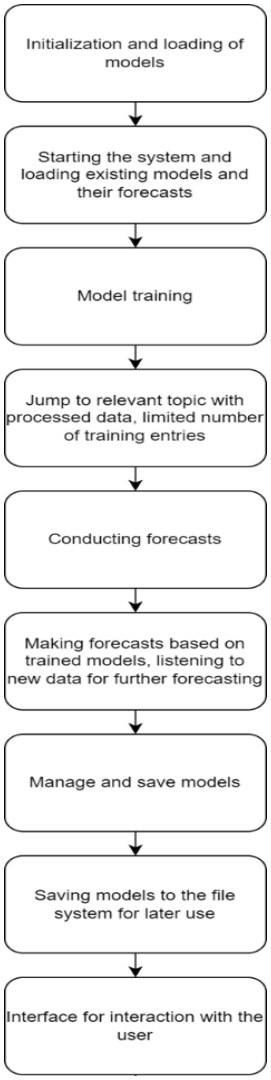

Figure 7: Algorithm of the information system operation

3. Making predictions:
   - Once the training process is complete, the system is ready to run predictions.
   - New data entering the same group is recorded for further prognostication.
   - If online training is provided for a particular model, the prediction can be performed instantly.
4. Model management and their storage:
   - Each model and its corresponding scaler can be saved as files.
   - This allows us to download already trained models at any time for further use.
5. Interface for user interaction:
   - The dashboard provides an opportunity to interact with the models, their predictions, and the management of the training/predicting process.
   - The user can interact with the system through this interface, performing operations with the models and receiving prediction results.

Each block represents a separate stage of the information system. The flowchart in Fig. 7 shows the sequence of steps in the system's operation, where each step leads to the next, managing various aspects such as model loading, training, forecasting, management, and user interaction.

5. Study of the Effectiveness of Introducing a Machine Learning Block

The utilization of machine learning methods in a smart home system offers significant benefits. They provide the system with the ability to analyze large amounts of data and identify patterns that may precede failures. This predicts possible problems and makes it possible to avoid them or prepare in advance. The assigned preventive action allows the system to automatically take measures to avoid disruptions, such as adjusting device operations or performing backups based on predictions.

Machine learning systems detect and correct failures faster, allowing the system to respond more quickly, and reducing the impact on user experience. These algorithms can also learn to adapt to changes in the environment, such as automatically adjusting the temperature in a building based on weather changes or occupant preferences. Optimizing energy consumption is another benefit—systems can adapt to the habits of residents and use energy efficiently, which can lead to significant cost savings. This approach improves the quality of service, ensures efficient system operation, and increases user comfort.

Table 1
Comparative characteristics of smart home systems with and without the use of machine learning methods

Characteristics	With machine learning	Without machine learning
Failure prediction	Predicting possible problems and avoiding them in advance	Reactive response to failures after they occur
Preventive measures	Automated actions to avoid failures	No automated measures before failure
Quick detection and correction	Prompt response to failures and their correction	Response to failures after they occur
Adaptation to changes	The ability of the system to adapt to changes in the environment	Lower ability to adapt to changes
Optimization of energy consumption	Efficient use of energy through automation	Limited capacity for energy efficiency
Resource requirements	Greater requirements for computing resources and data quality	Lower requirements for computing resources and data

Table 2
Indicators for comparing a smart home system with and without the use of machine learning methods

Indicator	With machine learning	Without machine learning
Amount of data (per week)	10 000	1 000
Number of failures (per week)	3	5
Failure detection rate	5 minutes after occurrence	30 minutes after the occurrence
Response to a failure	Automatic correction	Manual reaction
Failure prediction	Yes	No

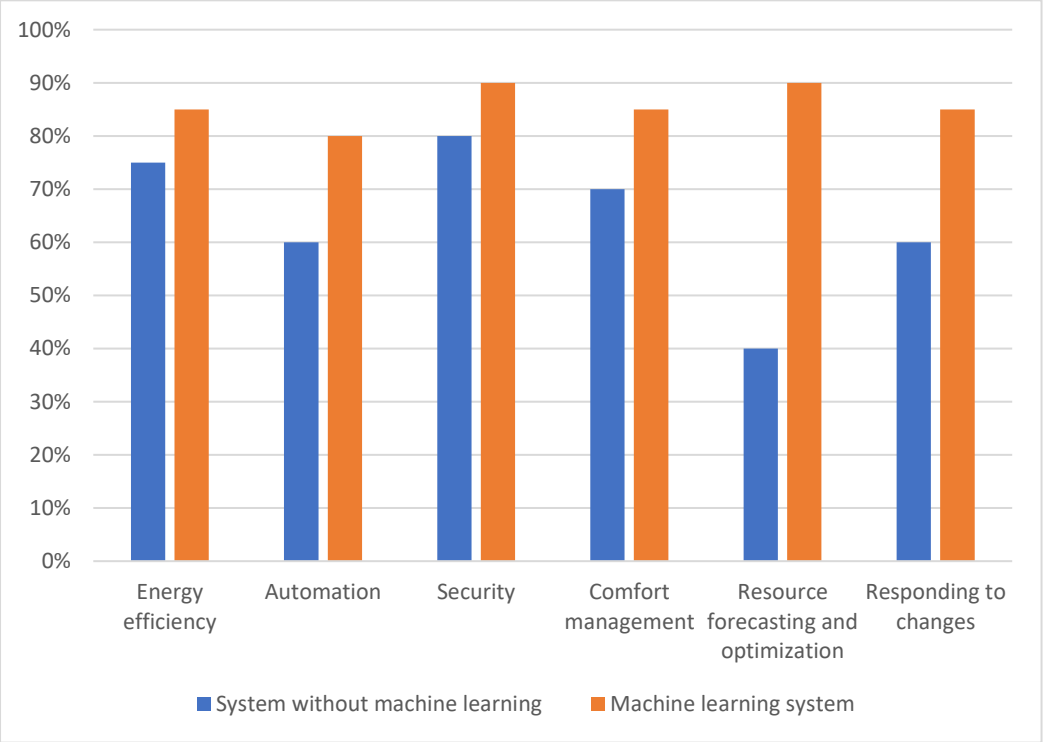

Figure 8: Smart home system performance with and without machine learning methods

So, without machine learning, the system made 5 failures:

1. Loss of connection—problems with connectivity in wireless networks.
2. Sensor failure—damage to the temperature sensor, resulting in missing data.
3. Software errors—caused the system to malfunction.
4. Automation malfunctions—the automation system did not respond to signals.
5. Power problems—problems with the power supply of the devices.

The system with machine learning avoided:

1. Predicting loss of connectivity—by analyzing previous network and signal data, the machine learning model predicted possible connectivity issues and activated mechanisms to restore connectivity before they occurred.
2. Early detection of sensor faults—the model detected anomalies in the usual sensor readings, indicating a possible

malfunction. Timely response to these signals helped to take action before serious problems occurred.

The system could not avoid:

1. Software bugs—Machine learning can improve the detection of some software bugs, but it cannot always predict or avoid the occurrence of complex software bugs or critical vulnerabilities.
2. Automation malfunction—Certain types of malfunctions or blockages in automated processes may be beyond the scope of machine learning, especially if they involve complex interactions between devices.
3. Energy issues—Loss of power or energy issues can occur without warning signals or changes in normal metrics, making them difficult to predict.

Machine learning helps to avoid certain problems by analyzing previous data and recognizing patterns, but it cannot predict absolutely all possible scenarios, especially if they arise from certain unpredictable factors or third-party interventions.

A smart home system that uses machine learning methods proves to be better than a system without this technology (as the study results show, the performance of a smart home using failure prediction methods gives an average of 22% better result compared to a similar system without prediction—Fig. 8). Machine learning allows the system to adapt to changes in the environment and user requirements, respond more quickly to new conditions, and optimize resource use. This helps to improve the system's efficiency in managing energy, comfort, safety, and user satisfaction.

Machine learning allows the system to predict and avoid failures, which ensures greater reliability and durability of the system. This approach also allows for increased automation, helping the system perform routine tasks without user intervention. Overall, a machine learning system remains the preferred choice due to its ability to predict, optimize, and adapt to changes, enabling it to provide more efficient and convenient smart home management.

6. Conclusions

The study results showed a wide range of modern technologies, sensors, and control systems used in smart homes. The survey showed that the available technologies have the potential to improve convenience, security, and energy efficiency.

The study identified several potential failures in smart systems, including those related to connectivity, sensors, and software. This makes it possible to prepare the system in advance to manage such situations.

The analysis of available machine learning methods indicates their potential in predicting and managing risks in smart homes. The considered models have shown high accuracy in predicting failures.

The developed methodology for predicting failures based on machine learning methods allows for effectively predicting and managing possible risks.

The overall analysis showed a positive impact of the introduction of machine learning systems in the operation of a smart home. Increased reliability and ability to adapt to changing conditions are noted.

The developed software is of great practical importance and allows the implementation of the failure prediction platform presented in this paper.

References

[1] O. Bahatskyi, V. Bahatskyi, V. Sokolov, Smart Home Subsystem for Calculating the Quality of Public Utilities, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 3421 (2023) 168–173.

[2] F. Kipchuk, et al., Assessing Approaches of IT Infrastructure Audit, in: IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (2021). doi: 10.1109/picst54195.2021.9772181.

[3] I. Kuzminykh, et al., Investigation of the IoT Device Lifetime with Secure Data Transmission, Internet of Things, Smart Spaces, and Next Generation Networks and Systems, vol. 11660 (2019) 16–27. doi: 10.1007/978-3-030-30859-9_2.

[4] V. Sokolov, et al., Method for Increasing the Various Sources Data Consistency for IoT Sensors, in: IEEE 9th International Conference on Problems of Infocom-munications, Science and Technology (PICST) (2023) 522–526. doi: 10.1109/PICST57299.2022.10238518.
[5] Z. Hu, et al., Bandwidth Research of Wireless IoT Switches, in: IEEE 15th Int. Conf. on Advanced Trends in Radioelectronics, Telecom-munications and Computer Engineering (2020). doi: 10.1109/tcset49122.2020.2354922.
[6] Y. Bazak, Comparison of machine learning methods for predicting failures in a smart home, Problems of Computer Engineering, Collection of Abstracts (2023) 125–127.
[7] A. Boiko, Modeling of the Automated System of Operational Control of the Parameters of the "Smart Home" in the Proteus Environment, Technol. Design 2(35) (2020).
[8] Y. Bondarenko, Manual for the Study of the Discipline "Statistical Analysis of Data" (2018).
[9] K. Hureeva, O. Kudin, A. Lisnyak, A Review of Machine Learning Methods in the Task of forecasting Financial Time Series, Comput. Sci. Appl. Math. (2) (2018) 18-28.
[10] V. Zhebka, Y. Bazak, K. Storchak, Features of Predicting Failures in a Smart Home Based on Machine Learning Methods, Telecommun. Inf. Technol. 4(81) (2023) 4–12.
[11] K. Kononova, Machine Learning: Methods and Models: A Textbook for Bachelors, Masters and Doctors of Philosophy in Specialty 051 "Economics", V. N. Karazin Kharkiv National University (2020).
[12] I. Puleko, A. Yefimenko, Architecture and Technologies of the Internet of Things: A Textbook, State University "Zhytomyr Polytechnic" (2022).
[13] I. Ruban, V. Martovytskyi, S. Partyka, Classification of Methods for Detecting Anomalies in Information Systems, Systems of Armament and Military Equipment 3(47) (2016) 47.
[14] Smart Home Technology: How AI Creates a Space that is Comfortable for Life. URL: https://www.everest.ua/tehnologiya-rozumnogo-budynku-yak-ai-stvoryuye-prostirkomfortnyj-dlya-zhyttya/
[15] V. Kharchenko, Fundamentals of Machine Learning: A Textbook, Sumy State University (2023).
[16] T. Arsan, Smart Systems: From Design to Implementation of Embedded Smart Systems (2016). URL: http://ieeexplore.ieee.org.focus.lib.kth.se/document/7753420/
[17] X. Sun, et al., System-Level Hardware Failure Prediction Using Deep Learning, 56th Annual Design Automation Conference (2019).
[18] V. Malinov, et al., Biomining as an Effective Mechanism for Utilizing the Bioenergy Potential of Processing Enterprises in the Agricultural Sector, in: Cybersecurity Providing in Information and Telecommunication Systems Vol. 3421 (2023) 223–230.
[19] V. Zhebka, et al., Optimization of Machine Learning Method to Improve the Management Efficiency of Heterogeneous Telecommunication Network, in: Cybersecurity Providing in Information and Telecommunication Systems Vol. 3288 (2022) 149–155.
[20] B. Zhurakovskyi, et al., Coding for Information Systems Security and Viability, in: CEUR Workshop Proceedings Vol. 2859 (2021) 71–84.
[21] M. Moshenchenko, et al., Optimization Algorithms of Smart City Wireless Sensor Network Control, in: Cybersecurity Providing in Information and Telecommunication Systems II Vol. 3188 (2021) 32–42.
[22] O. Shevchenko, et al., Methods of the Objects Identification and Recognition Research in the Networks with the IoT Concept Support, in: Cybersecurity Providing in Information and Telecommunication Systems Vol. 2923 (2021) 277–282.
[23] B. Chen, Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges, IEEE Access 6 (2018) 6505–6519.
[24] J. Mineraud, et al., A Gap Analysis of Internet of Things Platforms, Comput. Commun. 89–90 (2016) 5–16. doi: 10.1016/j.comcom.2016.03.015.