smart cities

MDPI

Article

Enabling Alarm-Based Fault Prediction for Smart Meters in District Heating Systems: A Danish Case Study

Henrik Alexander Nissen Søndergaard *, Hamid Reza Shaker * and Bo Nørregaard Jørgensen

SDU Center for Energy Informatics, Maersk Mc-Kinney Moeller Institute, The Faculty of Engineering, University of Southern Denmark, Campusvej 55, DK-5230 Odense, Denmark; bnj@mmmi.sdu.dk
* Correspondence: heso@mmmi.sdu.dk (H.A.N.S.); hrsh@mmmi.sdu.dk (H.R.S.)

Abstract: District heating companies utilize smart meters that generate alarms that indicate faults in their sensors and installations. If these alarms are not tended to, the data cannot be trusted, and the applications that utilize them will not perform properly. Currently, smart meter data are mostly used for billing, and the district heating company is obligated to ensure the data quality. Here, retrospective correction of data is possible using the alarms; however, identification of sensor problems earlier can help improve the data quality. This paper is undertaken in collaboration with a district heating company in which not all of these alarms are tended to. This is due to various barriers and misconceptions. A shift in perspective must happen, both to utilize the current alarms more efficiently and to permit the incorporation of predictive capabilities of alarms to enable smart solutions in the future and improve data quality now. This paper proposes a prediction framework for one of the alarms in the customer installation. The framework can predict sensor faults to a high degree with a precision of 88% and a true positive rate of 79% over a prediction horizon of 24 h. The framework uses a modified definition of an alarm and was tested using a selection of machine learning methods with the optimization of hyperparameters and an investigation into prediction horizons. To the best of our knowledge, this is the first instance of such a methodology.

Keywords: fault prediction; machine learning; district heating; consumer installations; smart meter

Citation: Søndergaard, H.A.N.; Shaker, H.R.; Jørgensen, B.N. Enabling Alarm-Based Fault Prediction for Smart Meters in District Heating Systems: A Danish Case Study. *Smart Cities* **2024**, *7*, 1126–1148. https://doi.org/10.3390/smartcities7030048

Academic Editor: Pierluigi Siano

Received: 12 April 2024
Revised: 6 May 2024
Accepted: 10 May 2024
Published: 14 May 2024

1. Introduction

Sensors in smart meters on the consumer side in district heating systems (DHSs) produce a plethora of data. Having the data validated is therefore critical for applications utilizing them.

As it is now, the data from smart meters are not used for many purposes [1]. They are mainly used for billing [1,2]. However, for billing, robust data collection is critical. Having smart meters that are calibrated and maintained properly is therefore key. In Denmark, the DH company has a direct vested interest in meters operating properly, as they are the owners of the meters and responsible for them operating properly [3]. This includes self-control of the meters [3] to make sure they stay within tolerances. Furthermore, for individual consumers, wrong measurements in their system can potentially affect the operation and therefore comfort in the building, given that the measurements are used for control locally [2]. Improvements in the identification of issues with smart meters can therefore help the DH company uphold its obligation to maintain the smart meters. This can also help them reduce the number of complaints received through the independent Danish "Energy Supplies Complaint Board" or general complaints sent directly to them. Identification of errors in sensors is currently performed using alarms; however, quicker notification about issues is preferred, as it could increase the reliability of the smart meters. A significant improvement in that current setup could be achieved through predicting when faults are bound to happen in the sensors. Achieving this can help fix problems arising in the sensors earlier, removing the risk of erroneous bills being issued and avoiding having

to perform data reconstruction to the same extent. This is a use case of fault prediction that can benefit the DH company today.

However, in the future, if the flexibility of buildings that have district heating is to be leveraged or other data-driven smart solutions are to be implemented using smart meter data, robust readings are necessary [1]. These solutions will rely on historical data and also on real-time (recent historical) data being dependable. Real-time data are needed to know the current operation of consumers to allow proper prediction of, e.g., how much and when to bid into a flexibility market, which can change in real time. Trusting current data is therefore essential. Work on leveraging the flexibility of DHSs on the grid side has been investigated in [4], where increasing and decreasing temperatures throughout the grid can provide flexibility to the electricity system. To illustrate another potential issue that can occur today, consider the following scenario (although not all district heating companies engage in this practice): When conducting a day-ahead simulation using inaccurate historical heat consumption data, including very recent figures, it can lead to suboptimal set point determinations in the system's operation. This could result in energy wastage and, in certain cases, failure to meet the heat demand due to, for instance, excessively low forward temperatures caused by the discrepancy between the data and actual conditions.

As mentioned, detection of these faults in sensors already happens in smart meters via the use of alarms, but these alarms are not utilized properly or tended to. Additionally, all of the data from the smart meter for a given day are transmitted after the day ends. Furthermore, even the current proactive real-time detection of faults (when utilized properly) is not sufficient for future applications that rely on real-time data being trustworthy. Moving towards predicting sensor faults is therefore necessary to unlock such applications.

1.1. Related Works

Work within fault or event prediction in general and for energy systems specifically has seen an increase in interest due to the increased available data. This can be attributed to more sensors in the systems [5] (which also measure more frequently) and developments in machine learning. In industry, work with fault prediction can be seen in examples such as [6], in which they utilize alarm logs to predict hardware faults in telecommunication, and for power systems. Betti et al. [7] perform fault prediction in large photovoltaic plants using self-organizing maps. Zhang et al. [8] perform line trip fault prediction using long short-term memory networks and support vector machines using real-world data. Alqudah et al. [9] develop a methodology for fault prediction in power systems using multimodal data and multi-instance learning.

While the electricity system field has seen considerable advancements, fault prediction within the district heating (DH) field remains significantly underrepresented in research. An illustrative case of this research gap can be found in Mortensen et al.'s work [10], where machine learning techniques are applied to predict the relative fault vulnerability of pipes—a rare exploration in DH research.

In contrast, DH research has extensively delved into predictive models for other aspects, such as heat load prediction, as evidenced by studies like [11–14]. These load predictions can be used to understand the nature of future heat loads for better day-ahead planning in the system from both a production and a grid standpoint to avoid faults due to misconfiguration of set points, which could lead to not enough supply to consumers. In a sense, the outputs of these methods can help avoid future faults in the system. However, the limited attention given to fault prediction, despite its critical relevance in ensuring system reliability and minimizing downtime, underscores the pressing need for further dedicated research in this specific area of DH.

The step down in maintenance functionality from fault prediction (predictive maintenance) is fault detection (proactive maintenance), which does not look into the future and only detects faults when they occur (or when they develop, i.e., early fault detection due to symptoms of a fault developing being detected as the fault). The field of fault/anomaly

detection and diagnosis in DHSs has seen much attention in works such as [15], which focuses on fault detection in district heating substations using a gradient boosting regressor. Zhang and Fleyeh [16] also perform fault detection at the substation but apply long short-term memory. Gadd and Werner [17] introduce various thresholds to substations and find that 74% displayed faults. Guelpa and Verda [18] detect fouling in heat exchanges in substations. Yliniemi et al. [2] detect sensor faults in district heating substations, improving the billing using threshold-based detection. Examples of improving the operation of the heating supply of a building that uses an HVAC system instead of DH using fault detection can be seen in works such as [19,20].

The papers presented above for DH fault detection mostly focus on detecting faults in the system and not errors in the data themselves. It can be difficult to distinguish between faults in the sensors themselves and faults in the system. However, there are some edge cases in which some measurements are an impossibility for the system. This could be, in the case of DH consumers, forward temperatures above what the supplying pump substation (i.e., substation between the transmission system and distribution system) supplies (assuming that measurement is correct). Work concerning data validation (faults in data and thereby the sensor) for DH data can be seen in papers such as [21], in which they transform the consumption data from district heating and apply k-means clustering for analysis of patterns. Leiria et al. [22] propose a framework that, in essence, detects outliers and errors in data and imputes those outliers and errors using, e.g., linear imputation. Schaffer et al. [1] develop a framework for detecting errors and outliers in data, which then carries out linear interpolation to match the temporal resolutions and finally applies imputation. Pedersen et al. [23] detect errors in data using, e.g., a frozen test, ranged test, Shewhart chart, sliding window, and principal component analysis. For wireless networks, in general, refs. [24,25] are good examples of both detection of errors in data and then of imputation. One example of prediction can be found in [26] for wireless sensor networks. These methodologies are the preprocessing steps carried out before applying other methods. If errors in the data are identified, reconstruction is needed, which can be carried out with imputation. Moving towards predicting that future errors will happen in the data can reduce the need for reconstructing data. The increased reliability of smart meter readings will in turn also increase the usefulness of fault detection methods for consumer installations. Another area of interest in the field of smart meters is communication errors, which can affect the data recorded. Not all of the data errors may be due to faults in smart meters but due to data transmission. The paper [27] develops a decision support system to identify bad transmission of data in electric smart meters based on various quality parameters.

If a fault prediction framework is to be created for district heating smart meters, it is critical to discuss the various types of prediction approaches that are available in the field. The following paragraph, therefore, describes the overall thinking behind how fault, or specifically event, prediction is categorized based on research problems. The identification of which category our specific problem belongs to is a function of the data available and the end goal of the prediction. The identified category determines which types of techniques are appropriate for the problem. Event prediction problems can be categorized into several categories according to the taxonomy by Zhao et al. [28]. The first main category is time prediction, in which you predict when or if an event will occur in the future. This category can be subdivided as follows:

- Occurrence-time prediction: does an event happen or not in a future period?
- Discrete-time prediction: in which among several time slots does an event occur?
- Continuous-time prediction: at which exact time does an event occur in the future?

The second main type is location prediction, in which you predict where an event will occur and which is separated into raster-based, where the space is divided into cells, or point-based, where each location is an abstract point. The third category is semantic prediction, in which future topics, descriptions, or general metadata are on top of the usual time and/or event locations. The last category comprises ways to jointly combine the

other categories such as predicting time and semantics at the same time. A more thorough description of each of the problems is available in [28].

As observed in the descriptions above, most of the focus has been on describing time prediction. That is because for this work, given the available time series data from smart meters and the overall goal, the use of occurrence-time prediction is the most fitting. For this, the appropriate techniques used for solving the problem that align with our goal and the data available are machine learning methods that can perform (binary) classification [28]. This is because the data contain labels from the alarms generated, and supervised methods are therefore very appropriate as they can leverage that aspect. By using classification, you also explicitly tell the model what to learn and can be certain the model has the same goal as you. You provide it with the solutions in training. Other techniques are anomaly detection and regression. They could be utilized as well, but given the available labels, it would be illogical to use them. For these two types of methods, you also assume that the models have the same goals as you and will try to predict what you want, and more thorough testing is needed to validate that. The methods used for the chosen technique, with example applications in parentheses, include support vector machines (prediction of solar flares [29]), decision trees (customer churn prediction [30]), and neural networks (crime prediction [31]).

1.2. Contributions

A gap has been identified within predictive maintenance in the Related Works subsection in DHSs for sensor faults as well as system faults. This paper tackles one of those areas by proposing a predictive sensor fault a maintenance framework for alarms in customer installations in the DHS that utilizes the alarms and data from sensors in training and then the sensor data for future event prediction of alarms in a given period. The use of classification machine learning methods was identified to allow for the prediction of the problem at hand [28], based on the available data and the problem at hand. The work within this paper will be able to distinguish between the system faults and sensor faults (wrong data), as it will utilize alarms that are meant to detect implausible measurements in sensors. The methodology in this work, therefore, detects faults in sensors and therefore also performs predictive data validation. Again, by performing predictive maintenance (compared to proactive, which is currently performed with the alarms), the goal is to avoid having to perform a comprehensive reconstruction of the incorrect data, help the DH company uphold its obligation to have proper readings for their billing, and enable smart solutions in the future that rely on the data being error-free in real time. This work also acts as a guideline for implementation in smart meters for other fields. The novel scientific contributions are as follows:

- A sensor fault prediction framework based on alarms generated in smart meters and operational data from the smart meters.
- First application of fault prediction in smart meter sensors in the DH domain, to the best of our knowledge.
- The application of the framework to a case study with real-world data.

To summarize the above, the contributions of this paper are not new alterations to machine learning methods, how they predict, or how they are tuned. Instead, contributions lie in what is specifically predicted (the application itself) and also the use of real-world data.

This paper is structured in the following manner. Section 2 presents the data available for performing fault prediction in sensors and determines an alarm candidate for prediction and changes to the definition of the alarms. It closes off with a description of how to structure the data for use in machine learning methods (e.g., prediction horizon). Section 3 presents the framework for the development of the fault prediction, which includes which machine learning methods will be used, how the methods will be tuned, and which key performance indicators (KPIs) to use. Additionally, a framework for real-world online application of the trained model is proposed. The results from the application of the framework on the prepared data are in Section 4, along with tuning of the various parameters in

the framework. This includes the selection of the best machine learning algorithm and its hyperparameters and further investigation into how changing the various horizon length parameters affects the KPIs.

2. Case Study and Data Preprocessing

Before the framework for fault prediction is described in detail, the case study and data must be introduced, as they lay the foundation for the framework. The next steps are the identification of which alarms will be used for prediction as well as structuring the data for prediction.

The main data set utilized in this paper is from a suburb in Odense, Denmark, which is part of a larger DHS on the island of Fyn. The data consist of consumption data from 914 consumers over 3 years (October 2019 to October 2022) at an hourly resolution. The consumers are a mix of various buildings, but the focus of this paper for showcasing the framework will be on detached homes. This results in a final 578 consumers for potential analysis. The data are collected by Kamstrup A/S [32] smart meters of the Multical line of products. Kamstrup is a smart meter manufacturer and supplies a large proportion of smart meters utilized in Denmark for heating, power, and water systems. The smart meters utilize external sensors that are connected to the smart meter for flow and temperature measurements (two flow sensors can be used). The collection of data is therefore extremely dependent on the accuracy of the external sensors. Moreover, they can utilize various communication modules to transmit the data to the district heating company. In the event of no transmission of data, there does not seem to be an alarm for this purpose (of course the alarm would not be able to be transmitted), but there is surely an alarm that is raised at the district heating company when data are not received. The data used in this development are historical and are extracted from an SQL server.

For missing data points, linear interpolation is carried out, and beforehand, any infinities are set to 0.

2.1. Consumption Data

The sensor data from each consumer for each hour consist of the following observations seen in Table 1. The data sampling frequency is 1 h.

The forward (T_f) and return (T_r) temperatures are calculated according to Equations (1) and (2). They are added as new features in the data.

$$T_f = FT_f / F \tag{1}$$

$$T_r = FT_r / F \tag{2}$$

Table 1. Description of raw data utilized. Each sample contains these data.

Variable Name	Variable	Unit	Description
ConsumerID	ID	-	A unique ID for each consumer.
DateTime	DT	-	It is in the format DD-MM-YYYY HH:MM.
Energy consumption	E	[GJ]	The heat energy utilized over a one-hour period.
Forward flow · forward temperature	FT_f	[$m^3 \cdot °C$]	The total water flow for each hour multiplied by the forward temperature. One can extract the temperature again by dividing by the flow, F. This is the setup, as you then can extract the mean forward temperature for that hour instead of an instantaneous value.
Return flow · return temperature	FT_r	[$m^3 \cdot °C$]	The total water flow for each hour multiplied by the return temperature.
Flow	F	[m^3]	The total flow of water that hour.

2.2. Alarm Data

The other data set is the information code or alarm data generated from the smart meters. These alarms are based on various rules. Most of the smart meters in use in the case study area can be categorized into two categories. The old smart meters include, for example, the Multical 66 to 602, and the new include the Multical 403 and 603 [33]. These generate very similar types of alarms with a few differences. The newer ones have more alarm types. The alarms for both new and old smart meters can be subdivided into alarms that are caused by sudden faults or events, which are very unlikely to be predictable by machine learning as they do not manifest in past data. The others have the potential to be observed in past data. The alarms are of higher temporal resolution than the consumption data, and they trigger at a specific time. The alarms are not turned off again; they are an event at a specific time, not over a period of time. The smart meter keeps raising alarms repeatedly at a specified interval until the issue is tended to. This can be called the minimum trigger frequency.

The alarm data consist of the attributes DateTime, the consumer ID, and the alarm number generated, and they do not contain a data point for each hour, only when alarms occur. Therefore, when the alarm data are merged with the consumption data using the DateTime variable, it is the case that for hours with no alarms, the alarm column is set to "0". The alarm occurrences thereby lose some resolution in order to conform to the consumption data due to the merger. Alarms are originally saved in a 1-min resolution.

2.3. Identification of Fault Prediction Possibilities

There are a few contenders for fault prediction among the alarms. They are seen in Table 2. All rely on there being a manifestation in the data such as drift or potential bias if large enough. A description of all alarms generated can be found in the technical description for the smart meters by Kamstrup [33].

Table 2. Alarm contenders for fault prediction.

Alarm #	Description	Potential Manifestation in Data
8	When the temperature sensor either violates a specified upper threshold (165 °C degrees in these data) or is disabled.	Forward temperature increasing values over time (drift).
128	Wrong ΔT (forward-return), meaning the difference between forward and return temperature. $\Delta T < 0$.	ΔT decreasing over time until it reaches zero and goes negative.
2048	The water flow rate violates a specified threshold for more than one hour.	Heightened flow values for each consecutive hour until reaching the set limit.

According to initial model development and inspection of data, the only good contender for fault prediction is alarm 8. In total, only 5 consumers of all the consumers have this alarm generated throughout the investigation period. Some of the consumers have many occurrences and some few. This creates a great imbalance in the data set for prediction. The imbalance is very high (data windowing explained later will improve the imbalance). On top of this, the model will only be trained and tested on these few consumers (you can use the healthy data from other consumers for training and testing, but that worsens the imbalance even more) and is thereby not a generalized model. For prediction, the identified sensor or variable used will be forward temperature (T_f). None of the other sensors can predict this alarm according to initial pretesting. An example of one of the consumers is seen in Figure 1. Here, it can be seen that the fault persists over time before dipping below the threshold again.

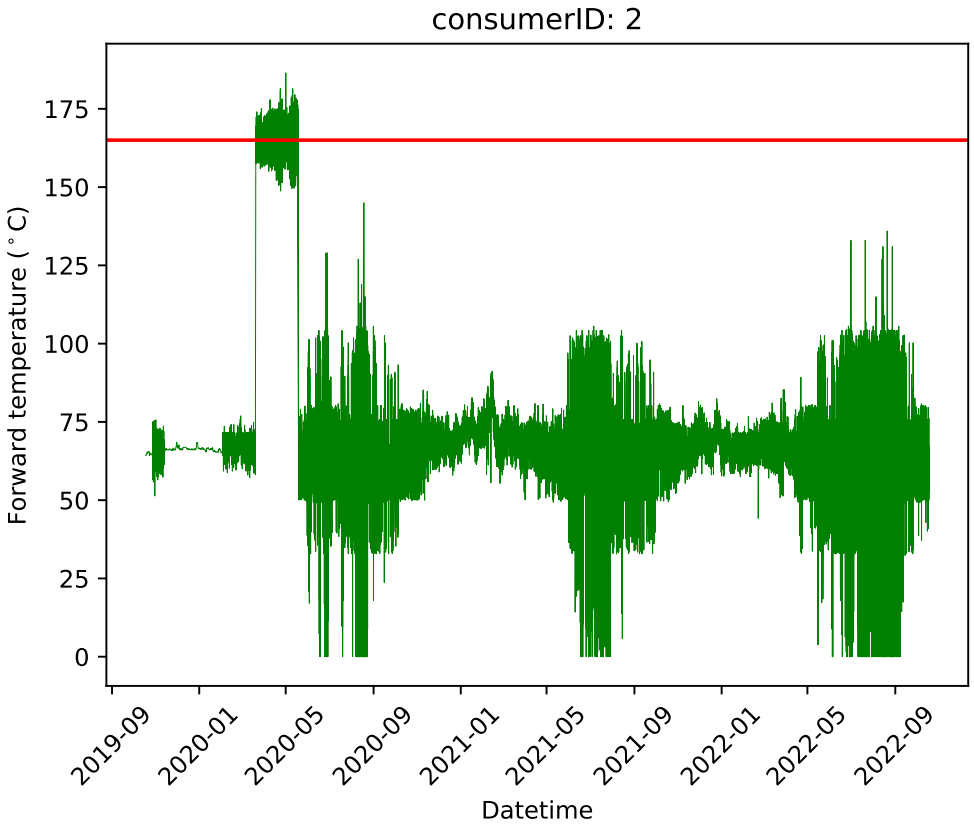

Figure 1. Forward temperature for one of the 5 consumers using the original fault definition. Here, it can be seen how long the first fault persists. The triggering threshold for alarm 8 is 165 °C, depicted by the horizontal red line.

2.4. Modification of Alarm Definition

Due to how alarms persist over time by triggering indefinitely, if they are not remedied, the smart meters keep generating the alarm every hour. Creating a model based on this results in predicting alarms when they are far into a faulty state, often by many hours or days. We are interested in the transition period from there not being a fault to there being one present and the machine learning methods identifying those transition patterns to predict alarms/faults. This means that for this paper, the definition of alarms is altered to create a more useful predictive model suiting our needs.

The fault definition will change to only triggering an alarm once when the temperature goes above the predefined threshold, and it is only able to trigger again after reentering normal operation. This will drastically decrease the number of alarms in the original data. Therefore another change is carried out.

The threshold for alarm 8 is currently set to 165 °C; however, even temperatures lower than this should never happen (it is set very conservatively). Therefore, a parameter or self-defined threshold is created for triggering alarms. This new parameter overrules, along with the previously mentioned change, the alarms in the original data. This makes it possible to dynamically expand the consumer base and therefore decrease the imbalance, as more consumers violate lower thresholds while still detecting errors in the measurements/smart meters. This equates to more alarms in the data. The number of consumers that are used after the modification is 150, compared to the 5 previously.

The threshold is set to 100 °C in this paper, as forward temperatures can never exceed the forward temperature from the supplying pumping station. A pumping station is located between the transmission grid (transmission generally operating at higher temperatures and pressures) and the distribution grid. The pumping station can reduce or increase pressures and mix return water with supply water to reduce forward temperatures. The forward temperatures are always lower than 100 °C after such a pumping station in the grid in question. They are typically higher in winter and lower in summer but usually in the range of 60 °C to 80 °C. The threshold (100 °C) is therefore set with a certain margin above that range to allow for some unusual operation of the grid. The goal is to not identify operation that is inefficient but to identify data that are very likely impossible for the system, indicating that there is something wrong in the measurement of the data.

Large drops in forward temperature readings in the summer are expected due to low consumption and therefore low flow rates. The water in the pipes can, therefore, cool considerably. However, lower temperatures than the ambient are not possible either. The temperatures in the ground at one meter vary between 3 °C in winter and 16 °C in summer on Fyn, where the data originate [34]. However, in this paper, we focus on detecting the alarms on too high temperatures and not too low, but this explains some of the dips in temperature in the summer.

Figures 2 and 3 show examples of one consumer where the forward temperature is extremely high (and low) and one that operates regularly within the new threshold depicted.

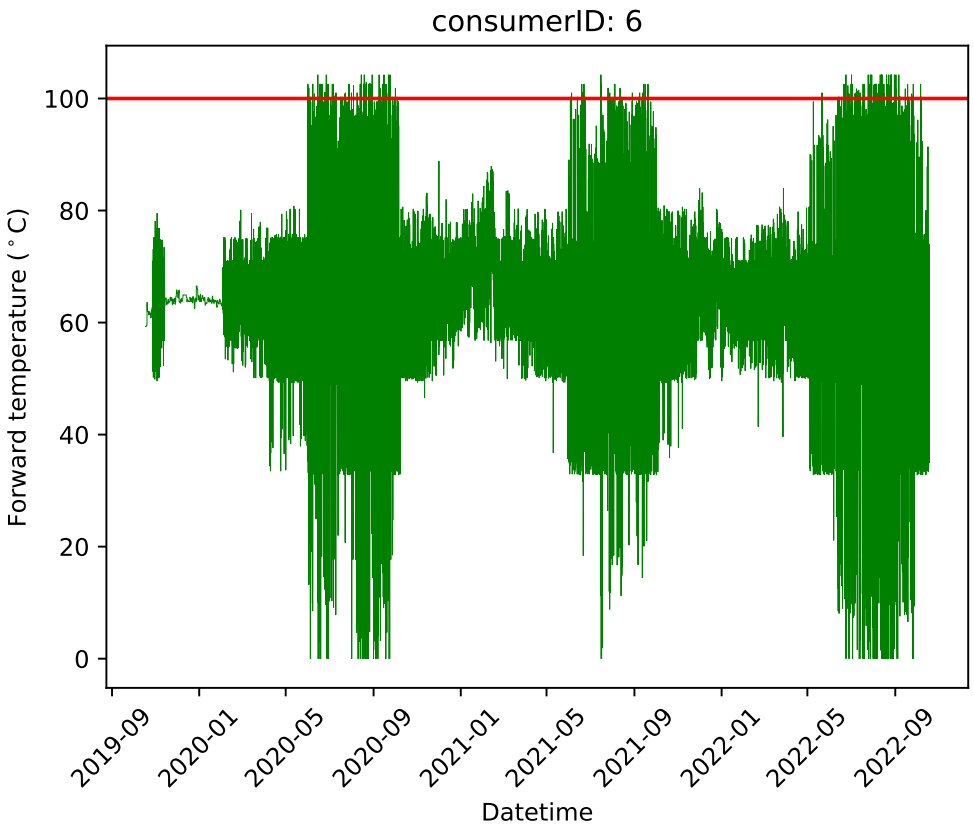

Figure 2. Forward temperature depicted by the green line with both extremely fluctuating forward temperature (which can be acceptable, as the water cools in the heating system if no heat is used, seen especially in the summer months) and violations of the new threshold depicted by the red line.

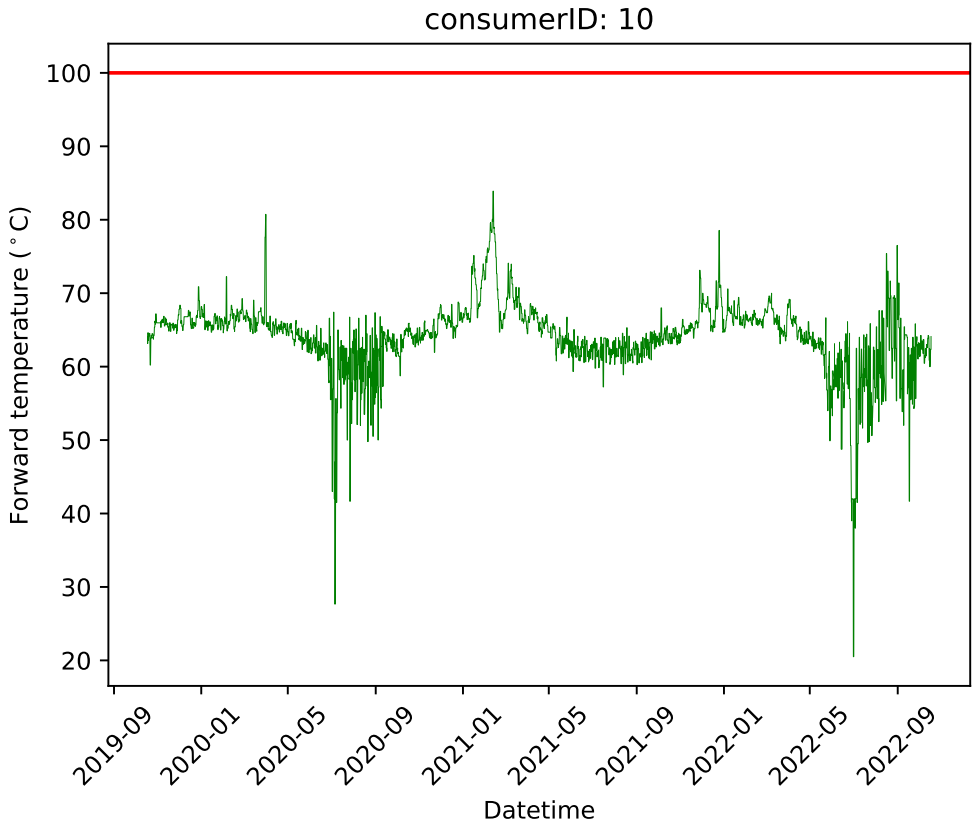

Figure 3. Forward temperature for a consumer with no irregular operation depicted by the green line. It displays low forward temperatures in the summer months due to no consumption in some hours leading to natural cooling of water in the system. The new threshold is depicted by the red line.

2.5. Preparation of Data for Prediction

Lastly, a windowing method is used to create samples for the training and testing of the model. A sample consists of p number of past values for the sensor variable in question (p number of variables are created as time shifts of previous measurements). Then we want to predict if an alarm happens in the next f future time steps. As we are not interested in when it happens in those future time steps, the alarm is boiled down to one label. It represents the following: Does an alarm happen (label = 1) or not (label = 0) in the next f hours. This means that if just one time step contains an alarm, it results in the whole period being labeled as an alarm period. Lastly, the future prediction window can be shifted further into the future using s. This is created due to the previous notion of the data currently being transmitted to the servers the next day. This means that we are not interested if a fault happens in the next hour, as that has already passed. However, if this barrier is removed in the future, this shift is not needed. This can be visualized in Figure 4. As mentioned in the Introduction, the prediction methodology used is labeled as an occurrence-time prediction problem [28], where one predicts whether or not a fault will occur in a future period. It is one of the most classic and widely used approaches to time-focused fault prediction. This windowing is carried out independently for each consumer and then inserted into the same data set again. In the Results Section, an investigation into the effects of changing the parameters p, s, and f on the prediction performance is carried out. Setting these correctly is a mixture of increasing the performance of the model while still achieving a prediction output that can be utilized for the goal of predictive maintenance. For example, having a very low shift (s) will likely increase the performance, but due to the nature of how the data are transmitted currently, that low value is not beneficial for the goal of predictive maintenance.

The imbalances in the data after the modifications to how alarms are raised with varying prediction horizons can be seen in Figure 5. Here it can be observed that increasing the future prediction horizon greatly increases the balance in the data. This is because more samples created by the windowing method have the output label 1, as just one label within the future window has to be an alarm.

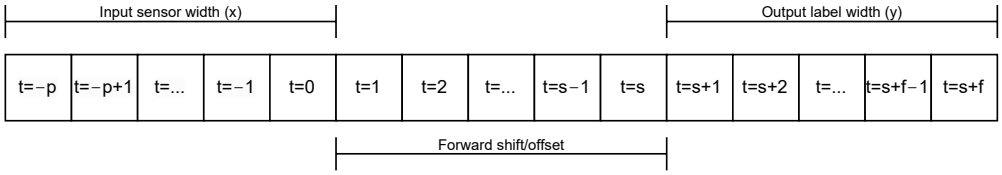

Figure 4. Data windowing example, with p indicating how many time steps of past values to utilize, s indicating the shift into the future of which labels to predict, and finally, 'f' as the number of time steps into the future to be predicted. These are combined into one value to be predicted by taking the maximum value of all output labels. This results in the method predicting whether or not a fault happens at all in that interval. We are not interested in the specific time it happens but if it will happen in a given future period.

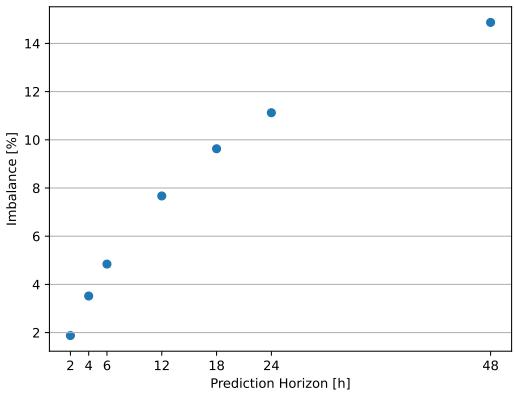

Figure 5. The effect of increasing the prediction horizon on the imbalance in the data (higher is better). 0% = no samples in the minority class. Imbalance = samples in minority class/samples in majority class.

3. Methodology

Firstly, a framework for the development phase, where the machine learning algorithm is determined along with the accompanying parameters of the framework, is presented. Secondly, the online implementation phase of the framework is presented, which outlines how the determined parameters and model from the development phase would be used in a real-world implementation using online unseen data, where results are provided to maintenance crews. The latter cannot be showcased in this paper as that requires a pilot project, but a flowchart is presented at the end of the section. The former is the main subject of this paper, to determine if it is possible to predict alarms and if so, determine the values of the underlying parameters to improve the prediction.

All code is written in Python v3.11, and the main packages are mentioned when necessary.

3.1. Development Phase Framework

The proposed fault prediction framework for the development phase can be seen in Figure 6. With this development framework, it is possible to assess if it is possible to predict alarms with the available data. An essential part of development is determining hyperparameters, which also includes picking a machine learning method. The procedure for determining the parameters that arise with this framework is outlined at the end of this section. This procedure also dictates the rough structure of the Results Section. This

framework evaluates the method based on unseen historical data, splitting the historical data set into training and testing. The testing data could be called validation data, but in this paper, they will be called test data. The framework includes the preprocessing steps described in Section 2 marked with blue, and descriptions of those can be found there.

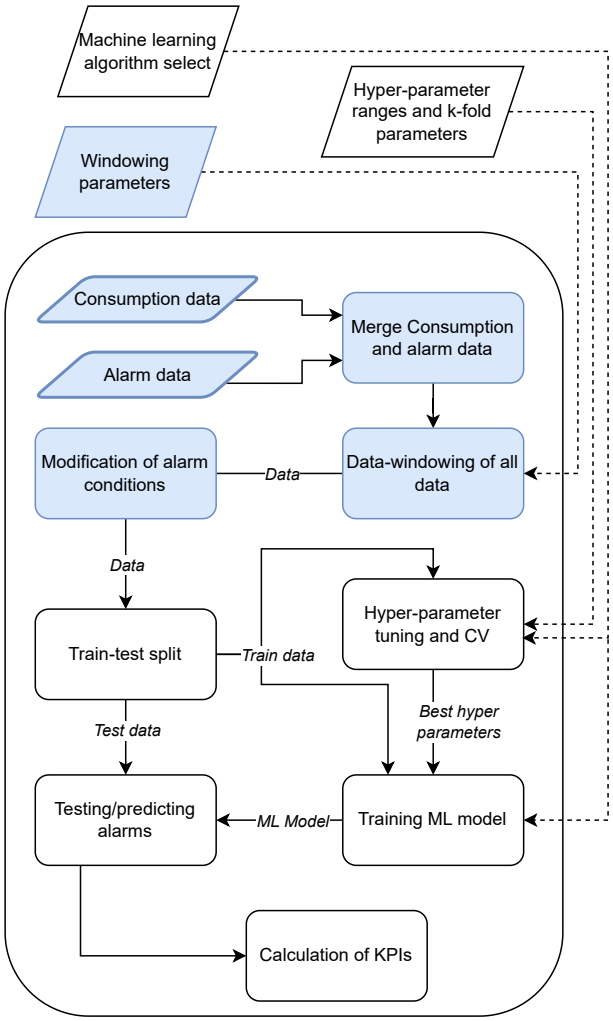

Figure 6. Fault prediction framework for the development phase. Blue indicates that the step was explained in the Case Study and Preprocessing Section. KPI = key performance indicator.

A handful of machine learning algorithms are applied to the framework for a more comprehensive analysis. The machine learning algorithms used are from the scikit-learn package [35], which contains the functions used in the "Training ML model" and "Testing/predict alarms" processes on the flowchart; these are discussed shortly. How each machine learning algorithm works is presented separately in Section 3.1.1.

The following paragraph goes through the processes on the flowchart. The "Train–test split" process uses the "train_test_split" method from [35]. The "train" function is used for the given machine learning algorithm in the "Training ML model" process, where the inputs are the independent variables (a vector of past values of forward temperature), and the target/dependent variable is the alarm (represented by 0 and 1). Additionally, hyperparameters are also supplied. The output of the "train" function is the trained machine learning model. The values of the hyperparameters for "Training ML model" are determined in the "Hyperparameter tuning and CV" process and are discussed separately in Section 3.1.3 in depth.

The "testing/predict alarms" process is carried out using the "predict" function from the machine learning algorithm by supplying only the independent variables from the

unseen test data and the ML model from training, which then outputs the predicted alarms (labels) for the testing data. The methodology can then be evaluated using the KPIs calculated in "Calculation of KPIs", which are based on comparing the actual known labels in the test data with the predicted ones. These KPIs are discussed separately in Section 3.1.2.

For a given execution of the framework, a set of "outside" parameters must be provided, which are described by the boxes connected by dashed lines to various processes within the framework. The ranges for the parameters in hyperparameter tuning are described in Section 3.1.3 and are static throughout the paper, and the hyperparameter tuning process determines the best values from the ranges provided automatically, whereas the other two outside parameters, that is, the choice of machine learning algorithm and picking windowing parameters, are currently not tuned automatically in the model, which leads to the following procedure:

1. Determining machine learning algorithm: Execute the framework on each of the proposed machine learning methods calculating the KPIs with the static windowing parameters, p = 24, s = 0, and f = 24. This identifies the hyperparameters for each machine learning method given the ranges it can pick from (also applying to the default hyperparameters). The best-performing machine learning method is then used onward.
2. Determining windowing parameters: Carry out three types of experiments where the three windowing parameters are adjusted individually to determine the effects on the KPIs. The best-performing mix of windowing parameters is determined (as mentioned in Section 2.5) by a mixture of maximizing the KPIs while also considering the usefulness of the prediction for maintenance crews, as that changes when adjusting the windowing parameters (this is why it cannot be performed automatically).

3.1.1. Description of Machine Learning Algorithms

This paper proposes applying multiple machine learning classifiers to the framework to determine the most appropriate for the given data in conjunction with hyperparameter tuning. The following machine learning algorithms will be tested:

- Random forest (RF).
- k-nearest neighbor (KNN).
- Support vector machine (SVM).
- Adaboost—using decision stumps.
- Gaussian naive Bayes (GNB).

Random Forest

RF is based on decision trees [36] and consists of various nodes of tree structures. It consists of a root node (the first node), decision nodes, and leaf nodes (the last nodes). For decision nodes, the sample data are put against a condition; it can then go either of two ways based on whether the condition is met or not (true or false). Then it can proceed to another decision node. In the end, the leaf nodes indicate which class the data sample belongs to. The decision nodes create splits to best separate the classes based on the data themselves. The basic structure of a decision tree is seen in Figure 7. Random forest creates *n* instances of decision trees and uses majority voting to decide on the prediction. Each tree is based on bagging (using a randomized sample from the data set for each tree, with replacement) and improving it by not supplying all variables when training for each decision node [36].

Adaboost

Instead of creating independent trees (bagging), AdaBoost can, if using decision trees as a method, utilize boosting, in which a series of small decision trees are created and each is dependent on the previous and is fitted based on the previous decision trees' residual [36].

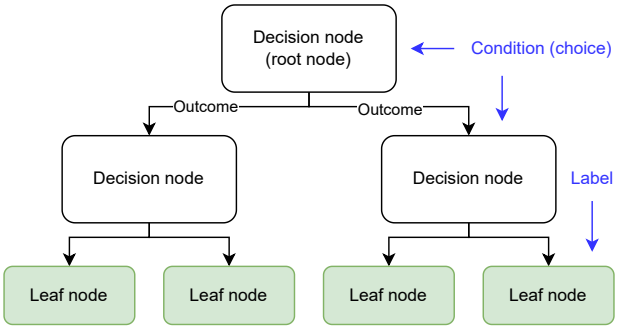

Figure 7. Visualization of a decision tree.

K-Nearest Neighbor

KNN looks for the k-nearest training data points to a new sample (with an unknown class), and based on which class the training data belong to, it determines which class the new sample point belongs to. This is based on majority voting, and the Euclidean distance is used to calculate the distance between each training data point and the new sample to determine the ones with the shortest distance [37]. In the standard setup, each point has the same weight. However, another weight method can be chosen where the data points closer to the new sample have a higher weight. A basic visualization of KNN for classification can be seen in Figure 8.

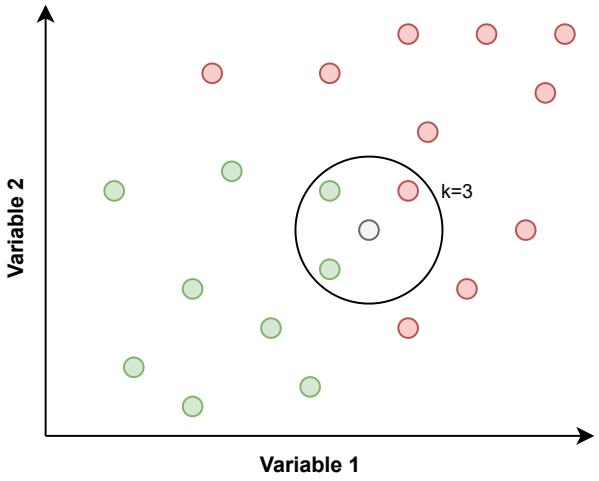

Figure 8. Visualization of KNN methodology. The green and red colors indicate the class of the data points. The white dot is a new sample. Looking at the 3 nearest neighbors, it belongs to class green.

Support Vector Machine

The goal of an SVM is to separate classes with hyperplanes and maximize the margin of the hyperplanes with the use of support vectors. Maximization of the margin improves the prediction, as the two classes are better separated [38]. The support vectors are the data points close to the hyperplane and influence the position and orientation of the hyperplane. In testing, the decision boundary, which determines which class new data belong to, is between the two hyperplanes. The minimization problem for SVM is seen in Equations (3)–(5).

$$\min_{w,b,\xi_i} \frac{1}{2}||w||_2 + C\sum_{N} \xi_i \tag{3}$$

$$\text{s.t. } y_i[w^T x_i + b] \geq 1 - \xi_i \tag{4}$$

$$\xi_i \geq 0 \tag{5}$$

The width of the margin is equal to $\frac{2}{||w||_2}$, and minimizing w will increase the margin as it is performed in the objective function. The last term in the objective function containing ξ_n allows for misclassification with punishment. Equation (4) represents that the data points must be located on or outside the hyperplane for each class. It is a combination of Equations (6) and (7), which describe the two hyperplanes and limiting values to be on either side of them. This can be seen in Figure 9. For the training of the SVM model, the radial basis function (RBF) kernel is used.

$$w^T x_i - b \geq 1, \text{if } y_i = 1 \tag{6}$$

$$w^T x_i - b \leq -1, \text{if } y_i = -1 \tag{7}$$

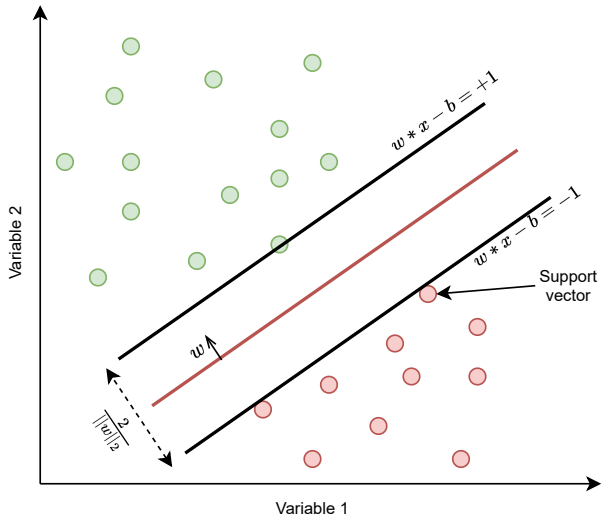

Figure 9. Visualization of the SVM methodology.

Gaussian Naive Bayes

This method is based on applying Bayes's theorem with the assumption of conditional independence between all pairs of features given the value of the class variable [39]. Equation (8) describes the probability of a new observation (x_i) belonging to class 1 (the fault class). Here, μ_y is the mean of the training data points belonging to class 1, and σ_y is the standard deviation of the training data points belonging to class 1 (it assumes a Gaussian distribution of the data). The same probability calculation can be set up for class 0. The class with the highest probability of the two for a new data point is then attributed to that data.

$$P(x_i|y=1) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{8}$$

3.1.2. Key Performance Indicators

Due to the imbalance in the data set between faulty and nonfaulty states, using accuracy as a metric for evaluating the performance of the model is not appropriate. If the model is bad at predicting the minority class, this has a low effect on the accuracy, as the minority class represents a low percentage of total labels. The chosen metrics are therefore the true-positive rate (TPR) (degree of missed detections) and precision (degree of false alarms). Higher is better for both. In tandem, they give an indication of how good the model is at predicting the minority class. The precision–recall curve can therefore be used for displaying the results.

3.1.3. Hyperparameter Tuning and Cross-Validation

Hyperparameter tuning is performed on each of the ML algorithms to determine parameters that improve the prediction performance of each algorithm, using the KPIs' precision and TPR in the tuning. Table 3 provides an overview of the hyperparameters that are changed for each method, what the chosen range for the hyperparameter tuning is, and what the default values are. Results for applying the models with both default and tuned parameters are presented in the Results Section. Randomized Grid Search from scikit-learn is utilized [35] for this purpose. The ranges are not set too large due to computational complexity, and the number of parameter combinations (n_{iter}) that are sampled is set to the default 10 (which is also a trade-off between quality and runtime). Hyperparameter tuning also employs 5-fold cross-validation. There is no hyperparameter tuning for SVM, as it took too long to compute.

Independent cross-validation is also carried out using 5-fold cross-validation using the K-fold function from scikit-learn [35] and applied to the training data. It is carried out to ensure the model is not overfitted.

Table 3. Parameters supplied to randomized grid search for hyperparameter tuning. Where appropriate, intervals are set as continuous.

RF	**Parameters**	**n_estimators**	**max_depth**	**min_samples_split**	**min_samples_leaf**	**bootstrap**
	Values	10–150	3–50	2–10	1–5	True, False
	Default	100	None	2	1	True
KNN	**Parameters**	**n_neighbors**	**leaf_size**	**p**	**weights**	**metric**
	Values	1–50	2–50	1–2	Uniform, Distance	Minkowski, Chebyshev
	Default	5	30	2	Uniform	Minkowski
AdaBoost	**Parameters**	**n_estimators**	**learning_rate**			
	Values	10–200	0.1–1			
	Default	50	1			

3.2. Online Phase Framework

A proposal for real-world online employment of the determined model can be seen in the flowchart in Figure 10. The picked ML model and windowing parameters are determined and are inputs along with new unseen online consumption data. This is presented to give a rough idea of how the machine learning model would be used.

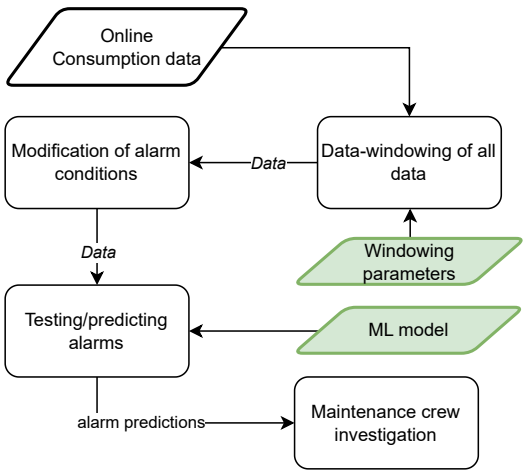

Figure 10. Fault prediction framework for the online phase in a real-world implementation. Green indicates the model and parameters from the development/training phase.

4. Results

This section applies the framework to the described data from the case study. A total of 70% of the data are used for training and the remaining for testing. The reported values from the machine learning models for precision, TPR, etc., are for the test data unless otherwise specified. The following Results Section consists of tuning the hyperparameters for the chosen machine learning algorithms, cross-validation, and general results from each algorithm. Thereafter, we can tune the windowing parameters for the input data using the best model from the selection of the machine learning algorithm.

4.1. Selection of Machine Learning Algorithm and Hyperparameter Tuning

In addition to selecting the machine learning algorithm and the accompanying hyperparameters, a short investigation on whether it makes sense to apply this framework to the data is carried out.

Figure 11 shows a t-distributed stochastic neighbor embedding (T-SNE) visualization of the input data and the corresponding correct label, where 0 is no alarm and 1 is an alarm. Here, a good separation of the data is present. However, due to how the data are plotted, a large number of green dots are below the middle region, meaning there is not a perfect separation. However, it seems there is potential for fault prediction.

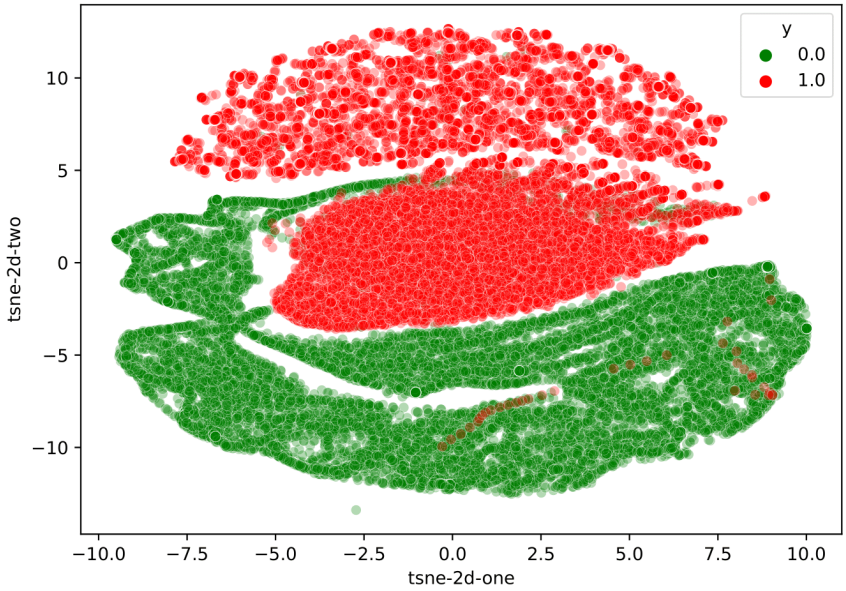

Figure 11. T-SNE visualization of the data. T-SNE can indicate if prediction methods can perform well if there is a good separation of labels of the data. According to the figure, there is a significant separation of label 0 (green) and label 1 (red).

The next part applies various machine learning algorithms to the problem in their default state and also with a randomized grid search applied to them with five-fold cross-validation. It is also set to maximize precision and TPR. The results can be observed in Table 4. A confusion matrix for default RF is seen in Figure 12. Here, it is seen as in the table that the models have lower TPR than precision. A higher precision is wanted, as false alarms can lead to less confidence in the model. However, missed detections are not good either but are more acceptable to have than false alarms. This means we might not catch all alarms, but for the ones we do, around 90% are labeled correctly. Many of these alarm events also overlap, meaning we might not predict an alarm now for the next 24 h, but as we approach it, it might be predicted. The samples that are set as faulty in the data set are often for the same alarm due to how the windowing functions. Another final note regarding the hyperparameter tuning results: it is very apparent, especially in using KNN, that the optimization heavily tries to improve precision while disregarding TPR.

Table 4. Overview of experiment results for prediction horizon of 24 h, no future shifting, and using the past 24 h of data for prediction. Showing default hyperparameter results and results using hyperparameters found via randomized grid search. AP = average precision.

ML Method	Setup	Precision	TPR	AP
RF	Default	88%	79%	91%
	RGS	93%	66%	89%
KNN	Default	90%	57%	79%
	RGS	95%	4%	63%
SVM	Default	93%	72%	91%
AdaBoost	Default	89%	75%	90%
	RGS	91%	72%	90%
GNB	Default	90%	74%	89%

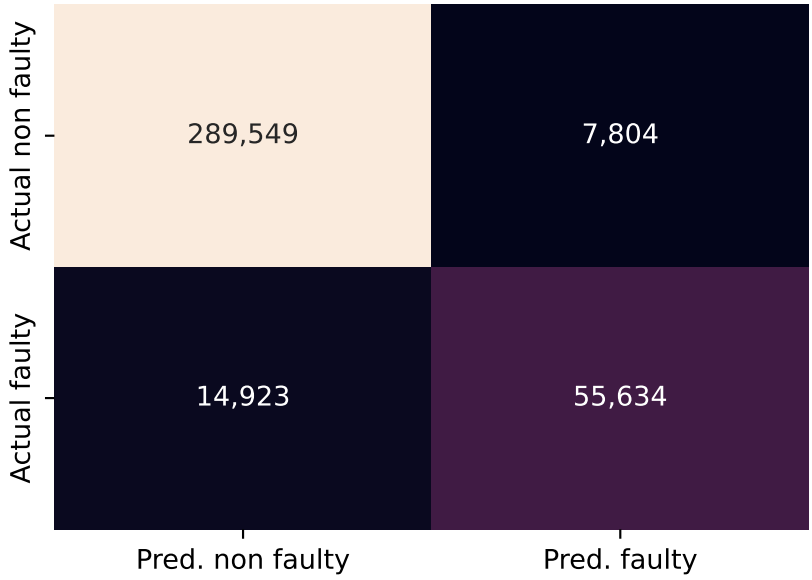

Figure 12. Confusion matrix for RF with prediction horizon at 24 h.

Overall, RF performs the best and will be used in the tuning of data input in the next section. Five-fold cross-validation of KNN and RF are presented in Table 5. There are quite consistent precision values across the folds, except for fold three which is in the mid-50% range.

Table 5. Five-fold cross-validation results for KNN and RF. The scoring is precision.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
KNN	78%	71%	56%	80%	73%
RF	75%	74%	54%	78%	73%

4.2. Determination of Horizons

The ML method has been established, and the next step is investigating the effects of increasing and decreasing the prediction horizon on the selected metrics, the effect of shifting the prediction horizon into the future, and the effect of how many hours of data are utilized for prediction. Figure 13 shows precision–recall curves for varying prediction horizons. As seen in the figure, increasing the prediction horizon improves the prediction capabilities. This can be attributed to two factors: first, the imbalance in the data set decreases with longer prediction horizons, and second, it generally becomes an easier task.

Imagine if the horizon was 2 months. In the data, alarms happen quite regularly, therefore the model achieving a prediction of there being a fault in those 2 months is almost always a yes and it will get it correct, as just one alarm has to happen. Of course, a prediction horizon of 2 months is too long here, but a very short one is also not good (and has bad performance). Or put another way, previous no-alarm windowed samples can be converted to actual alarms when increasing the horizon due to just needing one alarm present in the horizon for it to be labeled as an alarm as a whole. This can convert some previously false alarms to actual true alarms. A balance must be struck. We suggest a prediction horizon of around 24–48 h based on the reaction time for a maintenance crew. If the fault is expected to happen the following hour, there is no time to rectify it. The precision is also low for shorter horizons because the footprint is not large enough in the data for next-hour prediction. Very often, sudden temperature increases do not always cross the threshold of 100 °C. Increasing the horizon alleviates this problem. During a period of erratic temperatures, it is expected that at some point, the spike will go above 100 °C, as seen in Figure 2. So it is expected that the models learn that when there are a lot of spikes in the temperature, a fault is expected to happen soon (or when faults already are happening). It may not see the slow increase in the temperature leading up to the fault due to not high enough data resolution. The problem of data resolution is discussed in Section 5. It can also be observed that it is the precision that is improved by increasing the prediction horizon.

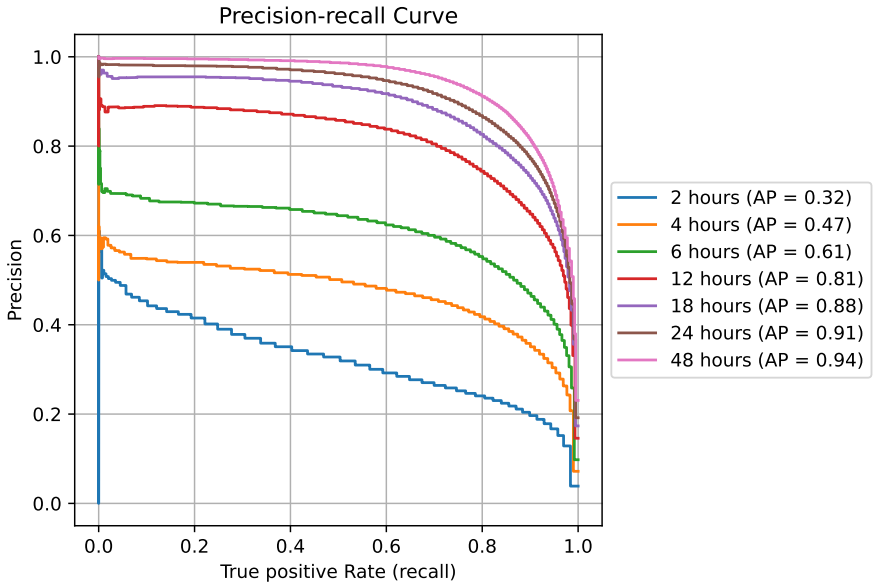

Figure 13. Precision–recall curve for default RF model with various prediction horizons (f), shift (s) of 0, and past data usage (p) of 24. The figure shows that increasing the horizon improves the prediction capabilities of the model.

Figure 14 shows it for varying shifting of the prediction horizon. Due to the conclusion that prediction in the first few hours into the future does not provide much value, a shift in the prediction horizon is useful. However, as seen in the figure, shifting it decreases performance, which is expected, but not by a large amount.

Figure 15 displays the precision–recall curve with varying numbers of past data utilized. Here, using fewer hours shows a large decrease in performance when utilizing under 12 h of past data. Increasing it by more than 24 h does not yield substantial increases in performance.

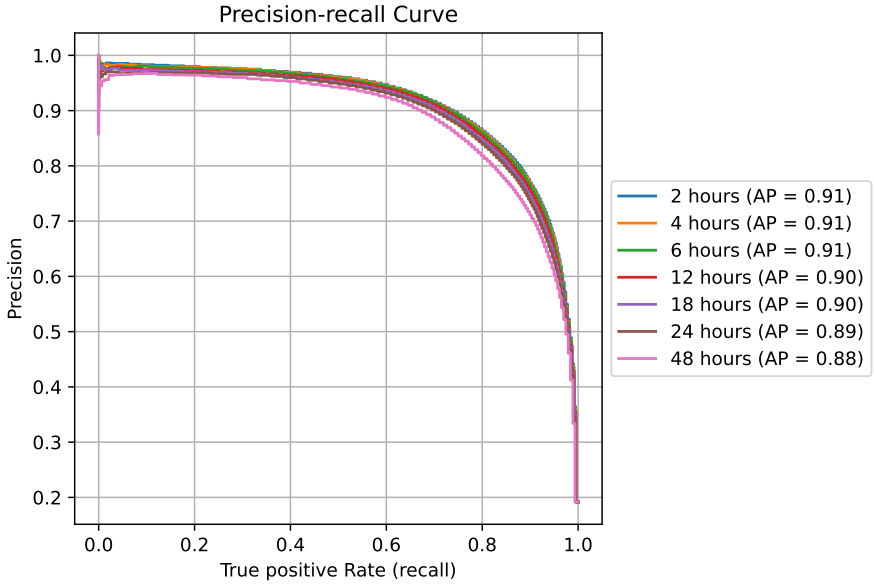

Figure 14. Precision–recall curve for default RF model with various shifts into the future (s) of the prediction horizon (prediction horizon is fixed at 24 h). This shows that increasing the shift slightly worsens the prediction capabilities of the model.

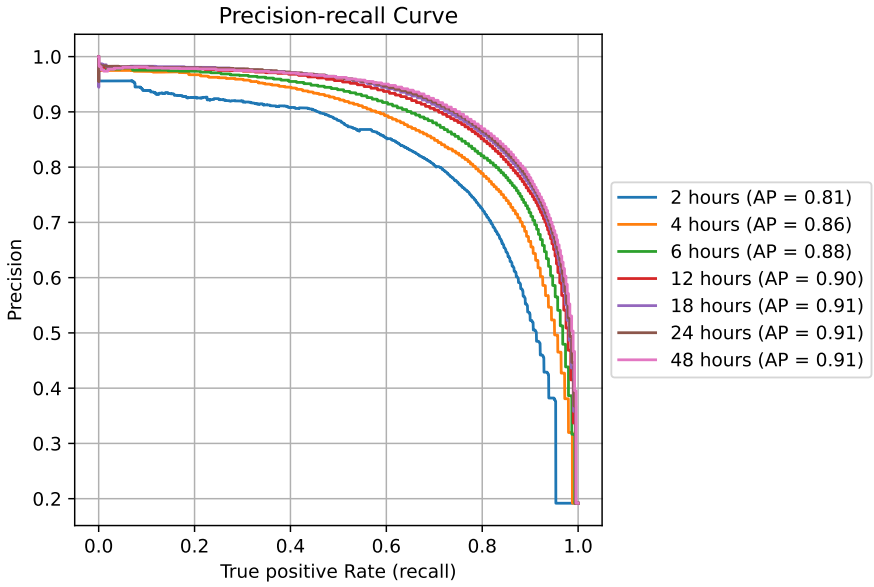

Figure 15. Precision–recall curve for default RF model with various usage of past values (p) for prediction (prediction horizon is fixed at 24 h). This shows that increasing the amount of past hours of data improves the prediction capabilities of the model.

5. Discussion

The current temporal resolution of the data could be improved; this could result in better predictive performance, as the increase in forward temperature would have a larger footprint in the data. At the moment, when data are averaged over an hour, the fault developing might not always be captured. However, this change demands a lot from the DH companies and their infrastructure, as the amount of data collected and transmitted would drastically increase. Moreover, the question of when it is possible to see the fault in data developing in relation to the time until the alarm triggers is a limiting factor concerning reaction time for maintenance crews. If the maintenance crew is not able to react quickly

enough because the fault symptoms happen just before an alarm is raised, it can be difficult to utilize the fault prediction capabilities.

Another aspect is that if the data are collected more frequently, there might not be enough data capacity at the district heating company (or its provider of that service) to store them, and secondly, this would increase electricity consumption, which could be a problem for smart meters utilizing batteries. However, according to the manufacturer of the new smart meters used, they have an estimated battery life of 16 years with a measurement data resolution of 10 s and daily transmission of data [33]. More frequent transmission of data is preferred, which would lower the battery life. Moreover, on their website, regarding transmission technology for data, they mention that smart meters utilizing wireless technologies such as 4G networks or narrow-band Internet of Things networks increase energy consumption significantly [32]. There is also the issue of the accuracy of the sensors due to bad calibration or the inherent nature of the sensors: even if the frequency of measurements is increased, would it even be beneficial if the sensors are not accurate enough in their measurements? The measurement accuracy of the Multical 603 is between almost 3% at low water flows and 2% at higher flows [33]. Another problem is the sensitivity of the sensors when moving to higher resolution. Very often, there is only a minuscule flow of water in 1 min, so it might report it as zero due to thresholds or just its ability to measure such low quantities. The nominal flow rate can be configured within a large interval for a Multical 603 [33] (0.6–15,000 $\frac{m^3}{h}$), but regarding the lower threshold for flow rates it can detect, it depends on which sensor is attached to the smart meter.

The fault prediction ideally happens, as with the current alarms, in a distributed manner in the smart meter instead of at a centralized location. This can minimize the data transmission and potentially remove the problem of the delayed daily transmission of consumption data (but the need for immediate transmission and notification when the framework predicts an alarm is needed). However, they are currently very constrained in their computational capabilities and cannot host such algorithms. The solution would be to increase the data transfer rate and perform the computation centrally. However, that would be a large strain on data communication but could be an initial temporary solution and require less change to the current infrastructure.

The data contain many smart meters that keep displaying faults according to the new alarm threshold defined in this paper. This was even the case using the original alarm definition. This results in data that are dissimilar from data where the faults are rectified quickly. If they changed their attitude and started fixing smart meters quickly, it would change the data foundation for the methodology and likely result in different performance.

6. Conclusions and Future Work

Alarms are a critical way of identifying nonfunctioning smart meters and installations. Detecting these faults in the smart meters before they occur improves the maintenance strategy and can help ensure more robust billing of consumers. Prediction can also eliminate the need for as much reconstruction of data, as errors can be detected before they affect the data greatly. This work will also help unlock the possibility of implementing future smart solutions leveraging the consumption data from consumers by ensuring robust real-time data. Trustworthy data are a necessity.

A fault prediction framework was therefore proposed in this paper that, with the use of alarms for training and consumption data from smart meters, predicts when the forward temperature reaches values that should not occur. It can predict if it reaches above that value or not over a future period (e.g., the next 24 h). It achieves this with a precision of 88% and a true positive rate of 79% using the random forest classifier. The methodology was tuned to improve the precision as it is critical to make sure that the alarms that are generated are very actual true alarms, whereas the second objective was to have satisfactory recall, which is an indication of how many of the true alarms are caught. Missing some detections is tolerated as long as the ones we do detect are true alarms. The idea of shifting

the future period of prediction was also introduced, meaning we do not, for example, predict the following 24 h but instead 24 h the next day.

Another important aspect of the methodology is how to implement it in the future. Where the prediction should be carried out, either at the smart meters themselves or centrally, is critical. As it is now, smart meters do not possess the computational power, but in the future, this could change. Choosing the location has a large effect on how to implement the methodology in the real world. A related problem to this is the transfer times for data from the smart meters. If the data are only transmitted daily to the central database and prediction is to occur there, it would not work as efficiently. This is the reason for introducing shifting the future time span of prediction, as it allows for performing predictions centrally with delayed data. However, ideally, we would predict not far into the future as performance improves. Another aspect is how the methodology will perform in the future if district heating companies react more quickly to alarms; this would significantly decrease the basis for training methods later on, as there would be significantly fewer instances of alarms (greater imbalance).

Future work includes training and testing the methodology on higher-resolution data, as it is expected that a larger footprint of the alarm is observed before it triggers. This will likely improve performance at shorter prediction horizons. Application of neural networks such as long short-term memory (LSTM) can also be carried out in the future. Lastly, a real-world pilot project must be carried out to test the online phase framework.

Author Contributions: Conceptualization, H.A.N.S., H.R.S. and B.N.J.; methodology, H.A.N.S.; software, H.A.N.S.; validation, H.A.N.S. and H.R.S.; formal analysis, H.A.N.S.; investigation, H.A.N.S.; resources, H.A.N.S.; data curation, H.A.N.S. and H.R.S.; writing—original draft preparation, H.A.N.S.; writing—review and editing, H.A.N.S., H.R.S. and B.N.J.; visualization, H.A.N.S.; supervision, H.R.S. and B.N.J.; project administration, H.R.S. and B.N.J.; funding acquisition, H.R.S. and B.N.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Danish Energy Agency under the Energy Technology Development and Demonstration Program grant number 64020-2102 and 134-22011.

Data Availability Statement: The data presented in this study are not available as they contain sensitive information about energy consumption patterns.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TPR	True-positive rate
DH	District heating
DHS	District heating system
KPI	Key performance indicator
KNN	k-nearest neighbor
RF	Random forest
GNB	Gaussian naive Bayes
SVM	Support vector machine
T-SNE	T-distributed stochastic neighbor embedding

References

1. Schaffer, M.; Tvedebrink, T.; Marszal, A. Three years of hourly data from 3021 smart heat meters installed in Danish residential buildings. *Sci. Data* **2022**, *9*, 420. [CrossRef] [PubMed]
2. Yliniemi, K.; van Deventer, J.; Delsing, J. Sensor fault detection in a district heating substation. In Proceedings of the 10th IMEKO TC10 International Conference on Technical Diagnostics, Budapest, Hungary, 9–10 June 2005.
3. The Danish Safety Technology Authority Regulation on Usage of Data Recording Instruments for Measuring of Usage of Water, Gas, Power or Heat. 2018. Available online: https://www.retsinformation.dk/eli/lta/2018/582 (accessed on 8 May 2024).
4. Zheng, W.; Xu, S.; Liu, J.; Zhu, J.; Luo, Q. Participation of strategic district heating networks in electricity markets: An arbitrage mechanism and its equilibrium analysis. *Appl. Energy* **2023**, *350*, 121732. [CrossRef]

5. Andresen, C.; Torsæter, B.N.; Haugdal, H.; Uhlen, K. Fault Detection and Prediction in Smart Grids. In Proceedings of the 2018 IEEE 9th International Workshop on Applied Measurements for Power Systems (AMPS), Bologna, Italy, 26–28 September 2018; pp. 1–6. [CrossRef]
6. Massaro, A.; Kostadinov, D.; Silva, A.; Obeid Guzman, A.; Aghasaryan, A. Predicting Network Hardware Faults through Layered Treatment of Alarms Logs. *Entropy* **2023**, *25*, 917. [CrossRef] [PubMed]
7. Betti, A.; Tucci, M.; Crisostomi, E.; Piazzi, A.; Barmada, S.; Thomopulos, D. Fault Prediction and Early-Detection in Large PV Power Plants Based on Self-Organizing Maps. *Sensors* **2021**, *21*, 1687. [CrossRef] [PubMed]
8. Zhang, S.; Wang, Y.; Liu, M.; Bao, Z. Data-Based Line Trip Fault Prediction in Power Systems Using LSTM Networks and SVM. *IEEE Access* **2018**, *6*, 7675–7686. [CrossRef]
9. Alqudah, M.; Kezunovic, M.; Obradovic, Z. Automated Power System Fault Prediction and Precursor Discovery Using Multi-Modal Data. *IEEE Access* **2023**, *11*, 7283–7296. [CrossRef]
10. Mortensen, L.K.; Shaker, H.R.; Veje, C.T. Relative fault vulnerability prediction for energy distribution networks. *Appl. Energy* **2022**, *322*, 119449. [CrossRef]
11. Chen, S.; Friedrich, D.; Yu, Z.; Yu, J. District Heating Network Demand Prediction Using a Physics-Based Energy Model with a Bayesian Approach for Parameter Calibration. *Energies* **2019**, *12*, 3408. [CrossRef]
12. Dang, L.M.; Shin, J.; Li, Y.; Tightiz, L.; Nguyen, T.; Song, H.K.; Moon, H. Toward Explainable Heat Load Patterns Prediction for District Heating. *Sci. Rep.* **2023**, *13*, 7434. [CrossRef]
13. Dahl, M.; Brun, A.; Kirsebom, O.S.; Andresen, G.B. Improving Short-Term Heat Load Forecasts with Calendar and Holiday Data. *Energies* **2018**, *11*, 1678. [CrossRef]
14. Sakkas, N.P.; Abang, R. Thermal load prediction of communal district heating systems by applying data-driven machine learning methods. *Energy Rep.* **2022**, *8*, 1883–1895. [CrossRef]
15. Månsson, S.; Kallioniemi, P.O.J.; Sernhed, K.; Thern, M. A machine learning approach to fault detection in district heating substations. *Energy Procedia* **2018**, *149*, 226–235. [CrossRef]
16. Zhang, F.; Fleyeh, H. Anomaly Detection of Heat Energy Usage in District Heating Substations Using LSTM based Variational Autoencoder Combined with Physical Model. In Proceedings of the 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA), Kristiansand, Norway, 9–13 November 2020; pp. 153–158. [CrossRef]
17. Gadd, H.; Werner, S. Fault detection in district heating substations. *Appl. Energy* **2015**, *157*, 51–59. [CrossRef]
18. Guelpa, E.; Verda, V. Automatic fouling detection in district heating substations: Methodology and tests. *Appl. Energy* **2020**, *258*, 114059. [CrossRef]
19. Dey, M.; Rana, S.P.; Dudley, S. A Case Study Based Approach for Remote Fault Detection Using Multi-Level Machine Learning in A Smart Building. *Smart Cities* **2020**, *3*, 401–419. [CrossRef]
20. Du, Z.; Fan, B.; Jin, X.; Chi, J. Fault detection and diagnosis for buildings and HVAC systems using combined neural networks and subtractive clustering analysis. *Build. Environ.* **2014**, *73*, 1–11. [CrossRef]
21. Johra, H.; Leiria, D.; Heiselberg, P.; Marszal, A.; Tvedebrink, T. Treatment and analysis of smart energy meter data from a cluster of buildings connected to district heating: A Danish case. *E3s Web Conf.* **2020**, *172*, 12004. [CrossRef]
22. Leiria, D.; Johra, H.; Marszal-Pomianowska, A.; Pomianowski, M.Z.; Kvols Heiselberg, P. Using data from smart energy meters to gain knowledge about households connected to the district heating network: A Danish case. *Smart Energy* **2021**, *3*, 100035. [CrossRef]
23. Pedersen, A.; Ustrup, S.; Mortensen, L.; Shaker, H. Data Validation for Digitally Enabled Operation & Maintenance of District Heating Systems. In Proceedings of the 2nd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Maldives, Maldives, 16–18 November 2022. [CrossRef]
24. Liu, Y.; Dillon, T.; Yu, W.; Rahayu, W.; Mostafa, F. Missing Value Imputation for Industrial IoT Sensor Data with Large Gaps. *IEEE Internet Things J.* **2020**, *7*, 6855–6867. [CrossRef]
25. Dzaferagic, M.; Marchetti, N.; Macaluso, I. Fault Detection and Classification in Industrial IoT in Case of Missing Sensor Data. *IEEE Internet Things J.* **2022**, *9*, 8892–8900. [CrossRef]
26. Ara, T.; M, P.; Bali, M. Fault Prediction in Wireless Sensor Networks using Soft Computing. In Proceedings of the 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), Bengaluru, India, 9–10 October 2020; pp. 532–538. [CrossRef]
27. Siryani, J.; Tanju, B.; Eveleigh, T.J. A Machine Learning Decision-Support System Improves the Internet of Things' Smart Meter Operations. *IEEE Internet Things J.* **2017**, *4*, 1056–1066. [CrossRef]
28. Zhao, L. Event Prediction in the Big Data Era: A Systematic Survey. *ACM Comput. Surv.* **2021**, *54*, 1–37. [CrossRef]
29. Inceoglu, F.; Jeppesen, J.; Kongstad, P.; Hernández Marcano, N.; Jacobsen, R.; Karoff, C. Using Machine Learning Methods to Forecast if Solar Flares Will Be Associated with CMEs and SEPs. *Astrophys. J.* **2018**, *861*, 128. [CrossRef]
30. De Caigny, A.; Coussement, K.; De Bock, K.W. A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees. *Eur. J. Oper. Res.* **2018**, *269*, 760–772. [CrossRef]
31. Lin, Y.L.; Yen, M.F.; Yu, L.C. Grid-Based Crime Prediction Using Geographical Features. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 298. [CrossRef]
32. Kamstrup A/S. 2023. Available online: https://www.kamstrup.com/en-en (accessed on 12 September 2023).

33. Kamstrup A/S. MULTICAL 603—Data Sheet. 2023. Available online: https://www.kamstrup.com/en-en/heat-solutions/meters-devices/meters/multical-603/documents (accessed on 8 May 2024).
34. Wang, P.G.; Scharling, M.; Wittchen, K.B.; Kern-Hansen, C. *Teknisk Rapport 13–18: 2001–2010 Dansk Design Reference Year Supplerende Datasæt*; Technical Report; Danish Metrological Institute: Copenhagen, Denmark, 2013.
35. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
36. Rokach, L.; Maimon, O. *Data Mining with Decision Trees: Theory and Applications*; World Scientific Publishing Co. Pte. Ltd.: River Edge, NJ, USA, 2015.
37. Witten, I.H.; Frank, E. *Practical Machine Learning Tools and Techniques*, 2nd ed.; Elsevier: Amsterdam, The Netherlands, 2016.
38. Alpaydin, E. *Introduction to Machine Learning*; MIT Press: Cambridge, MA, USA, 2010.
39. Mitchell, T.M. Machine Learning: Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression. 2005. Available online: https://api.semanticscholar.org/CorpusID:482346 (accessed on 8 May 2024).