applied sciences

MDPI

Article

Sensor Fault Detection and Classification Using Multi-Step-Ahead Prediction with an Long Short-Term Memoery (LSTM) Autoencoder

Md. Nazmul Hasan [1,†], Sana Ullah Jan [2,†] and Insoo Koo [1,*,†]

[1] Department of Electrical, Electronic and Computer Engineering, University of Ulsan, 93 Daehak-ro, Nam-gu, Ulsan 44610, Republic of Korea; hasan01@mail.ulsan.ac.kr
[2] School of Computing, Engineering and Built Environment, Edinburgh Napier University, Edinburgh EH10 5DT, UK; s.jan@napier.ac.uk
* Correspondence: iskoo@ulsan.ac.kr
† These authors contributed equally to this work.

Abstract: The Internet of Things (IoT) is witnessing a surge in sensor-equipped devices. The data generated by these IoT devices serve as a critical foundation for informed decision-making, real-time insights, and innovative solutions across various applications in everyday life. However, data reliability is often compromised due to the vulnerability of sensors to faults arising from harsh operational conditions that can adversely affect the subsequent operations that depend on the collected data. Hence, the identification of anomalies within sensor-derived data holds significant importance in the IoT context. This article proposes a sensor fault detection method using a Long Short-Term Memory autoencoder (LSTM-AE). The AE, trained on normal sensor data, predicts a 20-step window, generating three statistical features via SHapley Additive exPlanations from the estimated steps. These features aid in determining potential faults in the predicted steps using a machine learning classifier. A secondary classifier identifies the type of fault in the sensor signal. Experimentation on two sensor datasets showcases the method's functionality, achieving fault detection accuracies of approximately 93% and 97%. It is possible to attain a perfect fault classification performance by slightly modifying the feature calculation approach. In a univariate prediction scenario, our proposed approach demonstrates good fault detection and classification performance.

Keywords: sensor faults; LSTM autoencoder; SHAP; multi-step prediction

Citation: Hasan, M.N.; Jan, S.U.; Koo, I. Sensor Fault Detection and Classification Using Multi-Step-Ahead Prediction with an Long Short-Term Memoery (LSTM) Autoencoder. *Appl. Sci.* **2024**, *14*, 7717. https://doi.org/10.3390/app14177717

Academic Editors: Alex Fragoso, Mahdi Zareei and Jesús Arturo Pérez-Díaz

Received: 23 July 2024
Revised: 22 August 2024
Accepted: 30 August 2024
Published: 1 September 2024

1. Introduction

The Internet of Things (IoT) is a marvelous integration of several technologies that enable interconnectivity among everyday objects and devices, allowing data and intelligence sharing among them. In recent times, the applications of the Internet of Things (IoT) have expanded rapidly in diversified domains. The IoT is providing more flexible and automated patient monitoring in critical scenarios [1,2]. In agriculture, IoT-enabled systems can perform efficient irrigation [3], plant disease detection and control [4], and monitoring of soil and environment states [5,6]. In power systems, the IoT can ensure reliable operation of smart grids [7], analysis of electricity demand [8], and anomaly detection in grids [9]. IoT systems can bring us a smarter environment for living by automating lighting systems, waste disposal and management; parking allocation utilizing electric vehicles; and home automation [10,11].

At its core, there are several underlying components, such as sensors, micro-controllers and embedded systems, wireless networking, data analytics, and edge computing, that bring the IoT into its functional existence. Among the core elements, sensors and sensor networks play a pivotal role because they are the means to interact with the physical world. The majority of IoT applications rely on data-driven decision-making, analysis, and prediction based on the data collected by sensors. Thus, the reliability of sensor data is of paramount importance in IoT applications. Depending on the specific application,

sensors can be deployed in rugged environments like industries or in open environments with extreme weather conditions [12]. Aside from the environmental conditions, a sensor device can deviate from its normal functionality because of prolonged operation, hardware malfunctions, and improper installation. A sensor is considered faulty when the data stream generated demonstrates a significant deviation from its specified range. Such deviations are deemed anomalous or faulty [13]. In a broad context, sensors can be affected by two types of fault: incipient and abrupt [14]. Incipient faults are gradual in nature, develop slowly over time, and are hard to detect immediately. On the other hand, abrupt faults appear suddenly in sensor readings and are marked by a rapid and sometimes significant deviation from the normal readings. An elaborate classification is provided in Figure 1.

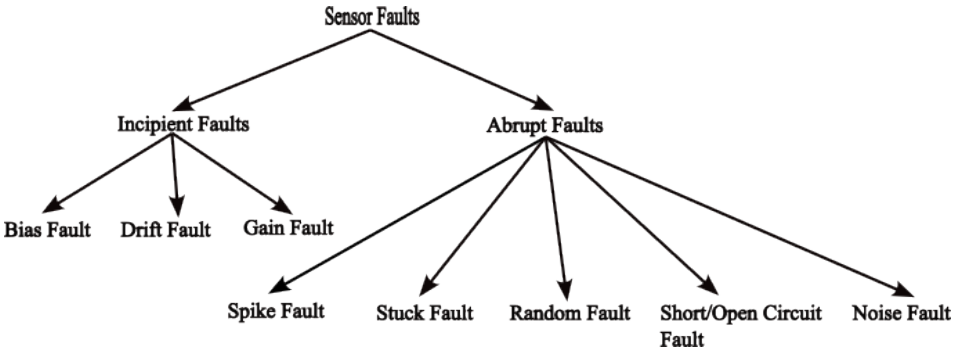


Figure 1. Sensor fault types.

In the current data-driven world, sensor fault detection and classification tasks carry great significance. In critical systems like those in the automotive and aerospace industries, undetected faulty sensors can compromise the safety and reliability of a system and could lead to catastrophic accidents. The timely detection of a sensor fault can prevent unwanted downtime and unnecessary maintenance, and ensuring sensor data integrity helps make data-based control and decision-making more accurate. Wu and Zhao [15] classified sensor fault analysis into three broad categories: knowledge-based, model-based, and data-driven approaches. Among these approaches, data-driven techniques are now widely explored by the research community due to the availability of unprecedented amounts of data generated from the integration of the IoT. The improvements in sensor technology, networking, and efficient computation power have given the data-driven sensor fault the upper hand over the other approaches. Several popular machine learning algorithms are utilized in sensor fault classification and analysis tasks. The support vector machine (SVM) [16–18], K-nearest neighbor (KNN), random forest (RF)-based classifiers, Gaussian Naive Bayes (GNB), and Artificial Neural Networks (ANNs) have demonstrated compelling results in sensor fault classification [19–21].

Several deep convolutional neural network (CNN) architectures are also employed for this purpose. As two-dimensional CNN networks require image data for processing, when CNNs are used for sensor fault classification, the one-dimensional sensor signals are converted to images using methods like Short-Time Fourier Transform (STFT) [22], scalograms [23], and wavelet transforms [24]. As sensor recordings are sequential in nature, recurrent neural networks (RNNs) are more suitable for them as these deep networks have the ability to capture long-term sequential dependencies and can handle variable sequence lengths.

Although machine learning (ML) algorithms such as SVM, RF, KNN, and others have been suggested for sensor fault analysis, they come with certain limitations. ML algorithms require careful feature extraction from raw sensor data, which can be both challenging and time-consuming. Additionally, ML algorithms do not inherently capture the temporal dependencies in time series data, which is crucial in sensor fault detection, where understanding the relationship between data points as they change during fault occurrences is essential. In real-world scenarios, faulty instances in sensor signals are expected to occur

less frequently, leading to an imbalance between faulty and non-faulty classes in practical datasets. Unfortunately, ML models often struggle with imbalanced data.

In contrast, deep learning (DL) models have the capability to automatically extract features, and models such as RNN, LSTM, and TCN can capture the temporal dependencies in time series data. These models are also suitable for multivariate time series scenarios. However, deep learning models require considerable computational resources, and power consumption is a significant concern in real-world deployments.

Both ML and DL approaches have their strengths and weaknesses in developing a methodology for sensor fault detection and classification. Given the nature of the data, DL models with recurrent structures are well suited for capturing temporal dependencies. However, for simpler implementations, ML models are more desirable. In this work, we aim to combine both approaches to leverage their strengths while keeping the overall approach less complex by maintaining a simple DL model architecture and considering a small number of features necessary for an ML model.

It is observed in the literature that when recurrent models are used for fault or abnormality detection in time series, multivariate data are typically considered. However, for fault detection and classification in individual sensors deployed in real environments, a univariate approach is often more practical. Multi-step prediction is a relatively newer concept for time-series-related tasks. While several multi-step prediction models have been proposed for other applications, their use in sensor fault analysis has not been widely explored. Therefore, in this work, we investigate how this approach can be applied to sensor fault detection and classification. The major contributions of this paper are as follows.

- A Long Short-Term Memory autoencoder (LSTM AE)-based univariate multi-step-ahead forecasting approach is proposed for sensor fault detection and classification. This method uses only normal sensor data for training.
- A feature extraction step is integrated into the prediction model to enable the detection of multiple types of faults. Additionally, three influential statistical features are reported that better represent bias, drift, and stuck faults in sensors.
- Results obtained from two different datasets are presented to demonstrate the functionality of the proposed approach.

This sensor fault-detection and classification scheme can be integrated into real-world systems where sensing activity is critical, such as in IoT networks. The inclusion of sensor fault analysis is essential for ensuring data reliability. In scenarios like smart cities, smart agriculture, smart grids, smart environmental monitoring, smart homes and healthcare, and smart industry, many applications today heavily depend on sensor data for effective analysis and informed, intelligent decision-making. Therefore, to ensure the credibility of the sensed data, implementing sensor fault analysis approaches is crucial in real-world applications.

The remainder of this paper is organized as follows. A brief summary of recent research focused on sensor fault analysis using an RNN and similar models is presented in Section 2. The proposed methodology is described extensively in Section 3. Experimental results and discussions are presented in Section 4, and Section 5 concludes the paper.

2. Related Works

For fault detection in systems, components, or sensors, autoencoder-based models are used in a wide range of applications such as bearing fault classification [25], smart grids [26], industrial processes [27], and healthcare [28]. Based on the number of features utilized in forecasting and the number of features in the output, time series forecasting tasks can be grouped into three categories: univariate, covariate, and multivariate. Considering the forecasting window, a time series forecasting problem can be categorized into single-step or multi-step scenarios. A single-step forecasting model predicts only a single time step in the future, whereas, in a multi-step scenario, the forecasting horizon comprises multiple time steps as depicted in Figure 2.

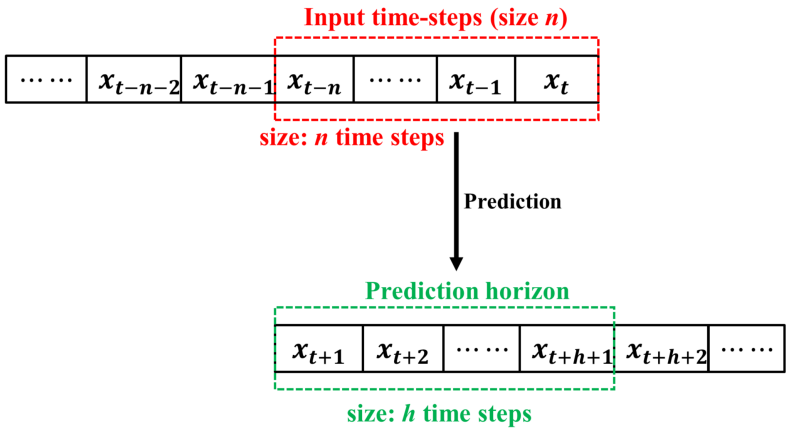


Figure 2. Univariate multi-step forecasting.

Both multivariate and univariate schemes are addressed in the literature for multi-step time series forecasting. Liu and Lin in [29] proposed a bidirectional LSTM (Bi-LSTM) model using multiple features to analyze how COVID-19 impacts electricity demand. Those authors achieved good performance while forecasting demand for a 20-day duration in terms of root mean square error (RMSE) and mean squared logarithmic error (MSLE). A modified encoder–decoder architecture was proposed in [30], where the encoder section is based on LSTM cells, and the decoder section is based on Bi-LSTM cells. Between the encoder and decoder, a temporal attention layer is included to obtain the latent space variables. The model was tested on five different datasets to predict up to six time steps in the future. The authors in [31] proposed a novel deep learning architecture called the Spatiotemporal Attention Mechanism (STAM) for multivariate time series prediction and interpretation. Spatial and temporal embeddings are computed using feed-forward networks and LSTM networks, respectively. Autoencoder-based deep learning models are particularly suitable for anomaly or fault detection in sensor signals, which is a semi-supervised approach where only normal signals are used; later, the reconstruction error is used to identify faults. Fault detection in a diesel engine of a maritime vessel using an LSTM-based variational autoencoder (VAE) was proposed in [32]. The authors used a modified parameter (log reconstruction probability) as the anomaly score to detect faults in an engine component. The drift detection of a chemical vapor deposition process in a semiconductor manufacturing system was studied in [33] using a variational autoencoder. The autoencoder model was trained to learn the normal process drift, and it later detected abnormal process drift from the sensor data by comparing the reconstruction error. An end-to-end framework for sensor fault analysis was presented in [34], in which the authors implemented two deep learning architectures: a CNN and a convolutional autoencoder (CAE). The CNN model was used to detect faults that occur in a single sensor within a collection of 10 sensors. Upon fault detection, the CAE is used to reconstruct a normal estimation of the faulty sample.

Detecting unusual patterns in time series data utilizing the concept of multi-step forecasting was implemented in several works. Bai and Zhao [35] proposed a multi-variable transformer architecture to predict incipient faults in chemical processes. Their model predicts the target process variable multiple steps in advance by utilizing multiple transformer models, and the multi-horizon prediction is compared with a fault threshold to detect faults in the process. In [36], anomalous pattern detection in internet traffic data is carried out using an LSTM-based encoder–decoder model. The authors predicted different horizons from three to twelve time steps with increments of three steps. An ANN-based multi-step forecasting model is described in [37] to detect leaks in a water distribution network. The difference between multi-step forecast data and the actual measured data is analyzed to check for anomalous patterns in the data under leakage conditions. A voltage fault diagnostic scheme was proposed by Zhao et al. [38] using a Gated Recurrent Unit

(GRU) neural network multi-step-ahead voltage prediction. The GRU predicts the battery cell voltages six time steps (one minute) in advance from 30 previous time step values. Predicted values are compared with a predefined threshold to make decisions about fault occurrences. Another attention-based multi-step prediction model was investigated in [39] to predict tool wear in a CNC router machine. Performance from various prediction horizons ranging from five steps to twenty steps was examined. Sensor fault diagnosis in a wind turbine blade by mapping spatiotemporal relationships among sensors was proposed in [40] by using a CNN model. The CNN predicts individual sensor readings employing all sensor readings, and the predicted value is compared with the actual reading to detect faults. More advanced variants of convolutional neural network-based models, such as Generative Adversarial Networks (GANs), have also been implemented for sensor fault analysis [41]. Given that sensor signals are highly prone to noise, other adversarial techniques, as proposed in [42,43], could be explored in future research for sensor fault analysis. A summary of significant works in sensor fault analysis is presented in Table 1.

Table 1. Summary of approaches for sensor fault analysis in the existing literature.

Reference	Dataset	Time Series Type	Model	Features	Task
[16]	Temperature signal collected by Arduino	Univariate	SVM	Statistical features	Classification
[17]	Gas turbine sensor data collected from simulator	Univariate	SVM	EMD based features	Classification
[18]	Sensor data published by Intel lab	Multivariate	SVM with Grey-Wolf Optimization	Feature extracted by Kernel Principle Component Analysis	Classification
[19]	Various sensor data collected from pressurized water reactor	Univariate	SVM, KNN, NN	-	Classification
[21]	Temperature and humidity sensor signal from a wireless sensor network setup	Multivariate	SVM, RF, DT, Extra-Trees	-	Classification
[22]	Altitude barometer sensor signal collected from an UAV	Univariate	CNN	Time frequency images	Classification
[23]	Various sensor data collected from aeroengine control system	Univariate	CNN	CWT scalograms	Classification
[33]	Sensor signal from semiconductor manufacturing process	Univariate	VAE	-	Classification
[34]	Synthetic data generated from shear-type structure and experimental data from an arc bridge structure	Mutivariate	CNN and CAE	-	Detection, classification, and correction
[44]	Sensor data derived from autonomous driving dataset	Univariate	1D CNN and DNN	Time domain statistical features	Detection, classification, and isolation
[45]	Sensor data from air quality dataset, WSN dataset, and Permanene Magnet Synchronous Dataset	Multivariate	ANN based digital twin concept	-	Detection, classification, and accommodation

3. Materials and Methods

The proposed methodology is presented in Figure 3. The complete fault detection and classification framework can be viewed as a combination of three major segments: multi-step prediction, fault detection, and fault classification. We consider a fault detection and classification framework for a single node in a sensor network. The multi-step prediction segment forecasts multiple future time steps, referred to as the prediction window. This segment is implemented using an LSTM autoencoder architecture. As shown in the figure, the LSTM autoencoder takes N time steps as input, from time instance t to $t - N$ as input. It then predicts M future time steps, from t to $t + M$.

In the next step, features selected during the feature-selection stage are computed from the predicted time steps. The prior feature-selection stage ensures that the most important features are chosen to minimize computational load during the fault-detection and classifi-

cation stages. Once these key features are computed, the feature vector is provided to the first-stage classifier, which performs fault detection. If the classifier identifies a potential fault, the features are then utilized by a second-stage classifier to classify the type of fault. Therefore, the first-stage classifier functions as a binary classifier, while the second-stage classifier functions as a multi-class classifier. A detailed description of each step in the proposed methodology is provided in the following subsections.

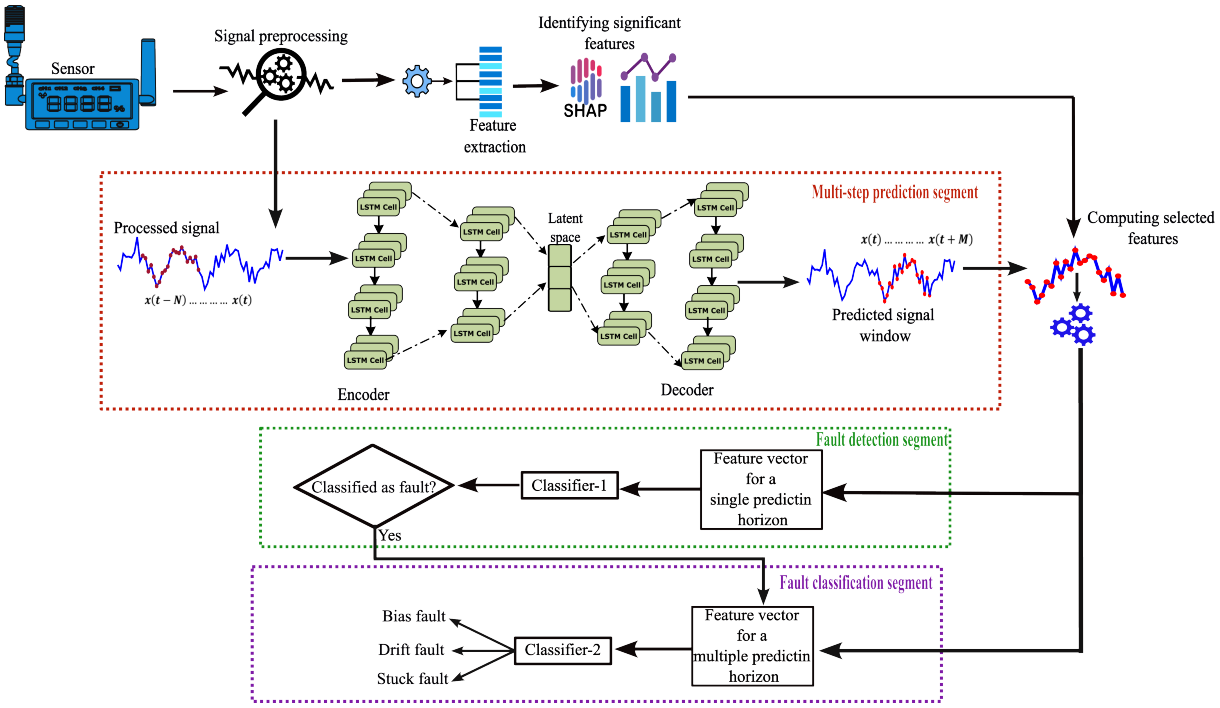


Figure 3. Univariate multi-step forecasting-based sensor fault detection and classification.

3.1. Datasets

In this work, sensor data from two different datasets are utilized. The first dataset consists of temperature sensor recordings from the dataset presented in [46]. This dataset contains load information and oil temperature data from two transformers over two years, recorded at 15-minute intervals. The oil temperature of a transformer is a significant indicator of its condition, and since this reading comes from a real sensor, we considered the temperature sensor signal from one transformer. The second dataset consists of sensor signals collected using a wireless sensor network (WSN) [47]. This WSN dataset includes temperature signals collected under both single-hop and multi-hop network settings. In this work, we considered the data collected using the multi-hop scenario. In order to perform sensor fault analysis, a dataset that includes sensor measurements for both normal and faulty measurements is required. Unfortunately, no open dataset offers sensor recordings of real sensor faults. Thus, researchers have adopted an approach to synthetically injecting faults into normal sensor signals [34,44,48]. To create the fault dataset in our work, we injected three types of artificial sensor faults. These three types of faults are commonly found in most of the articles on this topic. Researchers in [44,45,48] have included these faults in their studies. Additionally, drift, bias, and stuck faults have been studied in several real-world scenarios, such as aeroengines [23], UAVs [22], chiller systems [49], and hydrogen sensors [50]. The faults and the equations used to generate them are briefly described below.

3.1.1. Bias Fault

When the sensor output deviates to a higher value than normal, the sensor is said to exhibit a bias fault. This type of fault can be artificially generated by adding a constant bias term to the normal sensor output.

$$S^N_{bias} = S^N_{normal} + \tau, \tau = constant. \quad (1)$$

3.1.2. Drift Fault

In a drift fault, the sensor signal pattern is offset from the normal in a linear pattern over time. Sensors may experience this type of fault due to external factors or if circuit parameters are changed for any reason. This type of fault can be generated from a normal signal using the following equation:

$$S^N_{drift} = S^N_{normal} + \beta_N. \quad (2)$$

Here, $\beta_N = \alpha \times N$ to represent drift that increases over time along slope α, with N denoting the index of the data point.

3.1.3. Stuck Fault

A sensor reading that remains constant for considerably longer than anticipated implies the sensor is stuck. Such a fault might occur due to signal clipping, hardware issues, or battery malfunction. A stuck fault can be artificially generated with the following equation:

$$S^{t:N}_{stuck} = \tau, \tau = constant. \quad (3)$$

Figure 4 depicts samples of each type of fault considered in this work. As seen in Figure 4a, the bias faults were injected at random locations. Several samples were created where the location of the bias was kept random. Similarly, in drift fault and stuck fault are shown in Figure 4b,c the location of fault initiation is also random for the different samples.

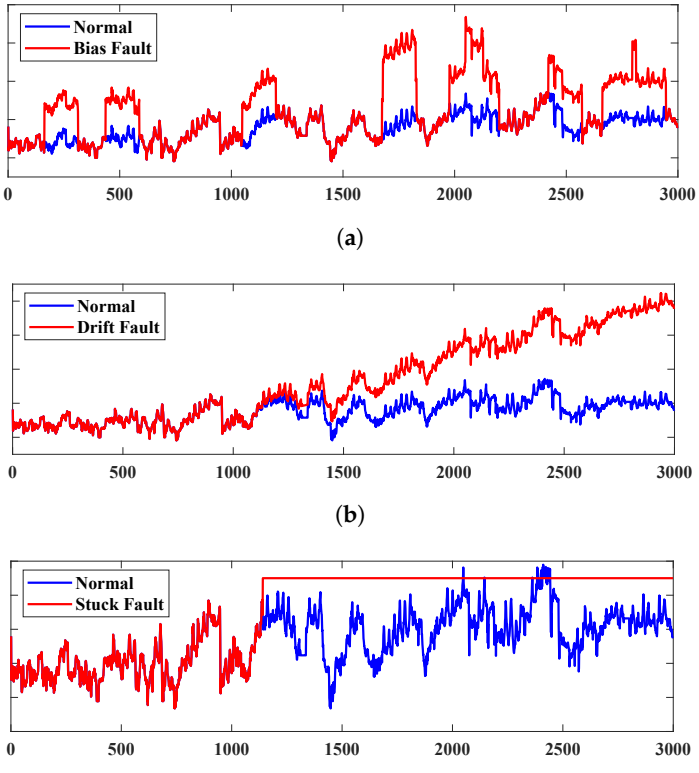


Figure 4. Sample of (**a**) bias fault, (**b**) drift fault, and (**c**) stuck fault.

3.2. Signal Pre-Processing:

After creating additional samples, the next step is to normalize the data. It is significant to bring the instances of the dataset within the same scale for proper training of deep learning models. In this study, we implemented the min–max normalization approach for

rescaling the samples. A time step value x_t in the sensor signal X_N is normalized using the following equation.

$$x_t{}' = \frac{x_t - min(X_N)}{max(X_N) - min(X_N)} \quad (4)$$

The first major block of our proposed scheme is the LSTM auto-encoder-based forecasting model. This model takes the sensor signal as its input and performs prediction for a horizon in the future. For training the auto-encoder model, it is necessary to format the raw sensor signal to properly train the model. The signal pre-processing stage involves scaling the signal and modifying its shape according to the shape that a recurrent neural network accepts.

3.3. Feature Importance Analysis

After creating the simulated dataset that contains three types of fault patterns and the normal sensor fault samples, we performed some prior investigation to identify the significant features from a number of statistical features. To proceed, in the first step, we computed 16 statistical features from the dataset. The statistical features that are computed from the sensor signals are listed in Table 2.

Table 2. Statistical features.

Feature Name	Mathematical Expression
Minimum	$min(y_i)$
Maximum	$max(y_i)$
Mean	$\mu = \frac{1}{N}\sum_{i=1}^{N} y_i$
Standard deviation	$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(y_i-\mu)^2}{N}}$
Kurtosis	$\mathcal{K} = \frac{1}{N}\sum_{i=1}^{N}\frac{(y_i-\mu)^4}{\sigma^4}$
Skewness	$\mathcal{S} = \frac{1}{N}\sum_{i=1}^{N}\frac{(y_i-\mu)^3}{\sigma^3}$
Root mean square (RMS)	$y_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\lvert y_i{}^2\rvert}$
Crest factor	$\mathcal{CF} = \frac{max(y_i)}{y_{i\,RMS}}$
Shape factor	$\mathcal{SF} = \frac{y_{RMS}}{\frac{1}{N}\sum_{i=1}^{N}\lvert y_i\rvert}$
Impulse factor	$\mathcal{IF} = \frac{y_{peak}}{\frac{1}{N}\sum_{i=1}^{N}\lvert y_i\rvert}$
Clearance factor	$\mathcal{CLF} = \frac{y_{peak}}{(\frac{1}{N}\sum_{i=1}^{N}\sqrt{\lvert y_i\rvert})^2}$
Variance	$\mathcal{S}^2 = \frac{\sum_{i=1}^{N}(y_i-\mu)}{n-1}$
Energy	$\mathcal{E} = \sum_{n=-\infty}^{\infty}\lvert y_n\rvert^2$
Power	$\mathcal{P} = \lim_{N\to\infty}\frac{1}{N}\sum_{n=0}^{n=N-1}\lvert y(n)\rvert^2$
Peak to rms	$\frac{y_{peak}}{y_{rms}}$
Range	$y_{max} - y_{min}$

Once the features are calculated, the next step is to observe the importance of these features in classifying the faults on the basis of Shapley values. The concept of Shapley values originates in game theory, where these values help to determine the contribution of each player to the total gain in a collaborative game. In a machine learning paradigm considering each feature in the dataset as a player, Lundberg and Lee in their seminal paper proposed the SHapley Additive exPlanations (SHAP) method [51] for interpreting the features' importance in a model's predicting capability.

The SHAP values are obtained by comparing the model's predictions, first by including the particular feature and again by excluding the feature. For a dataset consisting of $\mathbb{N}$ features, the SHAP value of a particular feature, ϕ_i, can be expressed as

$$\phi_i = \sum_{\mathbb{S}\subseteq\mathbb{N}-i}\frac{|\mathbb{S}|!(|\mathbb{N}|-|\mathbb{S}|-1)!}{|\mathbb{N}|!}[\mathcal{F}(\mathbb{S}\in\{i\})-\mathcal{F}(\mathbb{S})] \quad (5)$$

Here, $\mathbb{S}$ represents a subset excluding feature i, $\mathcal{F}(\mathbb{S} \in \{i\})$ denotes the model's prediction considering feature i, and $\mathcal{F}(\mathbb{S})$ is the prediction when feature i is not present. Finally, the summation is carried out over all the possible subsets of $\mathbb{S}$. These SHAP values can then be utilized to express a feature's importance. Evaluating the features at an early stage will be beneficial in fault detection and classification in later stages because it will be enough to consider only the highly important features instead of all the features.

3.4. Long Short-Term Memory

First introduced by Hochreiter and Schmidhuber [52] in 1997, the LSTM architecture excels compared to the other recurrent structures in processing sequential data. The inclusion of the memory cell and the gating mechanism enables LSTM to address long-term dependencies efficiently. LSTM also mitigates the vanishing gradient problem, thus making the training of deep architectures on sequential data more effective. The structure of a single LSTM unit is shown in Figure 5.

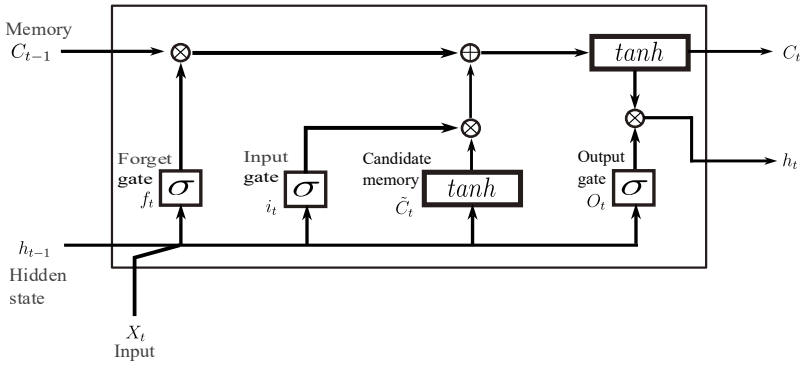


Figure 5. An LSTM cell.

At each time step, an LSTM unit manages a hidden state and a memory state by utilizing three gating mechanisms. The input gate decides whether new information should be stored in the cell state or not. The input gate uses the current input and the previous hidden state in its computation, as expressed by the following equation:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{6}$$

The candidate memory state, $\tilde{C}_t$, is also computed using the same input, and it represents new information that could be added to the cell state. The candidate cell state equation is

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{7}$$

The forget gate decides what information from the previous cell state should be discarded or retained. The input to this gate is the previous hidden state and the current input:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{8}$$

The cell state at the core of the LSTM is responsible for retaining long-term information, which is updated according to the following equation:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{9}$$

Based on the previous hidden state and the current input, the output gate controls what the output would be as the new hidden state. The output and the hidden state relations are as follows:

$$O_t = \sigma(W_0 \cdot [h_{t-1}, x_t] + b_0) \tag{10}$$

$$h_t = O_t * tanh(C_t) \tag{11}$$

3.5. LSTM Autoencoder

Autoencoders are a specific kind of deep neural network and are quite efficient at sequential data analysis and prediction [53,54]. As discussed in the previous subsection, to leverage LSTM's ability to adapt to temporal dependencies in the data, an LSTM-based autoencoder architecture is considered in our work. The principle of such autoencoder models is to encode the input sequence into a compressed representation through the encoder section, and the subsequent decoder segment decompresses the latent representation to reconstruct the original sequence. In this work, we utilize an LSTM autoencoder network to learn the pattern in a sensor signal through training, so it can make predictions for multiple time steps into the future.

During training, the encoder will take a particular input window of the sensor signal, $x_t \in \mathbb{S}^{t-m}$, that represents the m historical data points, including the current time step. The encoder network will transform this input into a latent vector of length l: $\mathbb{Z} = \mathbb{E}(x_t) \in \mathbb{S}^l$. In the final stage, the decoder produces a sequence of length n that represents the prediction for n time steps in the future: $\tilde{x_t^\star} \in \mathbb{S}^{t+n}$. The autoencoder is trained with the objective being to minimize the mean squared error for the prediction horizon, $\mathbb{L} = \frac{1}{2}\sum_n \|x^\star - \tilde{x}^\star\|$, where $x^\star$ is the actual value for future time steps.

3.6. Fault Detection and Classification

The fault detection approach in this paper is envisioned as a binary classification problem. Two key elements in fault detection are multi-step prediction by the LSTM autoencoder and the selected features based on their importance from the feature importance analysis described in earlier segments. The multi-step-ahead prediction of a sensor signal is performed by the LSTM autoencoder, which works as a sequence-to-sequence model. Using this autoencoder model, n time steps will be predicted using the past m time step values.

The autoencoder is trained with normal data following the steps listed in Algorithm 1. Once the model is trained, it is evaluated using fault observations from the sensor signals. For an input window of n time steps, the trained model will predict m time steps in the future. In the next step, these predicted data points are used to compute the features that were selected on the basis of their importance using the SHAP analysis described earlier. Utilizing the calculated features for multiple windows in the fault samples and information on the point of fault injection from the fault dataset generation step, a dataset is created that consists of only two classes: no-fault and fault. The fault class encompasses all three types of fault considered in this work. This makes fault detection a binary classification scheme that classifies the feature vector resulting from the future prediction horizon from the LSTM multi-step prediction model. The calculation and utilization of the features for the prediction horizon are necessary because of the range in the sensor signal amplitude. For example, if we consider the bias fault scenario, at the fault's location, the signal increases to a value higher than normal. Now, there may be some instances where the amplitude after the occurrence of the fault may equal or even be smaller than the maximum sensor reading observed in a sample of the sensor recordings. This has been observed in a few research papers that compare some threshold value with the multi-step predicted value to identify the occurrence of a fault. As mentioned, normal sensor recordings might acquire higher or similar values for some instances of a fault, and this reason prevents us from setting a threshold, because in such a case, some fault instances might go undetected. If the threshold is set to a higher or lower value, a normal instance might be determined as faulty. Additionally, three different types of fault are considered in this work, so it is hard to set a fixed threshold for different types of fault because fault patterns differ from each other. Once a fault is detected, the next step is fault classification, and again, ML classifiers are used. However, at this stage, it becomes a multiclass problem. Utilizing prior information about fault-injected time steps from synthetic fault data generation, another dataset is constructed in which the faults are given different labels according to their type. Therefore, this dataset contains three classes: bias, drift, and stuck faults. As a predicted horizon

is detected as a fault by the preceding classifier, it is provided to the second classifier to determine the type of fault that occurred.

Algorithm 1 Multi-step prediction-based fault detection and classification.

Inputs: Training data: Time series sequence with $\mathcal{T}$ data points, $\mathcal{T} = \{x_1, x_2, x_3, \cdots\cdots, x_{\mathcal{T}}\}$, Encoder: $\mathcal{E}(\cdot)$, Decoder: $\mathcal{D}(\cdot)$, Window size: ω, Prediction horizon: $\hbar$

Outputs: Decision on fault detection and fault type

Training Autoencoder:

Create input and output sequences (X_i, Y_i)

for $i = 1, 2, \cdots, \mathcal{T} - \omega + 1$ **do**

$X_i = \{x_i, x_{i+1}, \cdots, x_{i+\omega-1}\}$

$Y_i = \{x_{i+\omega}, x_{i+\omega+1}, \cdots, x_{i+\omega+\hbar-1}\}$

end for

for Each training epochs **do**

Encoder output: $\mathcal{d}_i = \mathcal{E}(X_i)$

Decoder output: $\hat{Y_{\hbar}} = \mathcal{D}(\mathcal{d}_i)$

Calculate loss: $\mathcal{L}\left(\hat{Y}_i, Y_i\right) = \frac{1}{\hbar}\sum_{t=1}^{\hbar}(\hat{y_{i,t}} - y_{i,t})^2$

Optimize the model parameters, φ: $\underset{\varphi}{minimize}\mathcal{L}\left(\hat{Y}_i, Y_i\right)$

end for

Prediction, Fault detection and classification:

Predict the output sequence: $\hat{Y_{\hbar}} = \mathcal{D}(\mathcal{E}(X_i))$

Compute the four selected features: $f_1\left(\hat{Y_{\hbar}}\right), f_2\left(\hat{Y_{\hbar}}\right), f_3\left(\hat{Y_{\hbar}}\right), f_4\left(\hat{Y_{\hbar}}\right)$

Fault detection: $C_1(f_1, f_2, f_3, f_4) = fault_{detect}$

if $fault_{detect} = True$ **then**

Classify the fault type: $C_2(f_1, f_2, f_3, f_4) = fault_{type}$

end if

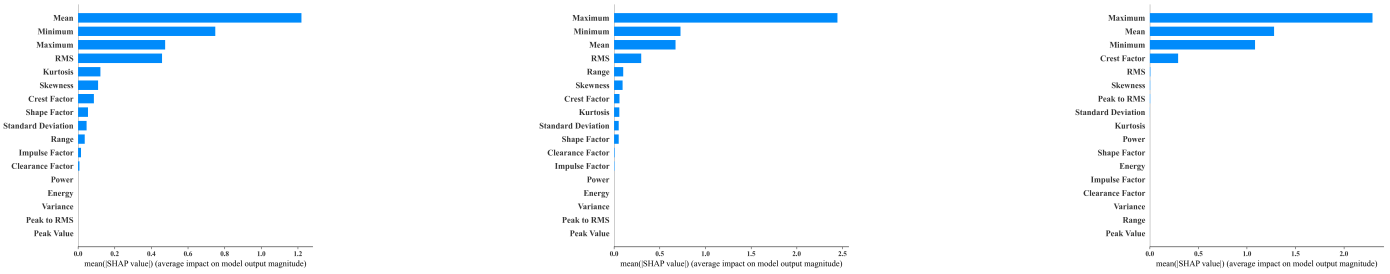
Return: Fault detection decision: $fault_{detect}$ Fault classification result: $fault_{type}$

4. Results and Discussion

In this section, the performance of the autoencoder in multi-step forecasting along with fault detection and classification performance by our proposed methodology are discussed.

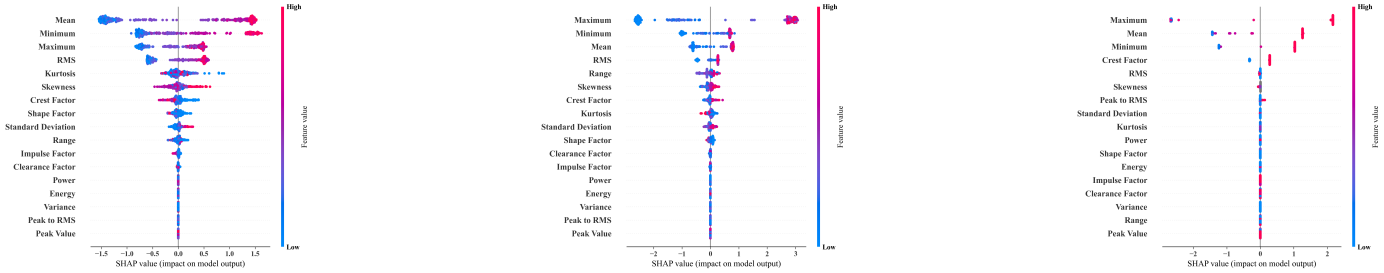
4.1. Feature Selection Using SHAP

As mentioned earlier, based on the Shapley values, the important features for classifying the types of fault are determined. This prior feature importance investigation will help in observing only a few features for fault detection and classification at a later stage. For finding feature importance, a tree-based explainer approach is used, and the XGBoost classifier model is chosen as the tree-based model. The significant features are presented using the SHAP feature importance plots and SHAP summary plots in Figures 6 and 7.



(**a**) (**b**) (**c**)

Figure 6. SHAP feature importance plot for (**a**) bias faults, (**b**) drift faults, and (**c**) stuck faults.



(**a**) (**b**) (**c**)

Figure 7. SHAP summary plot for (**a**) bias faults, (**b**) drift faults, and (**c**) stuck faults.

In the bar charts presented in Figure 6, the features are arranged according to their importance for each type of fault. We can see that maximum, minimum, and mean are the top three important features that distinguish these three sensor faults from the normal scenario. The summary plots in Figure 7 provide more insights into the significance of the features. The summary plot shows that the more positive SHAP values of the features' maximum, minimum, and mean correspond to the fault class, and negative values pertain to the normal class. Based on the SHAP analysis, the top five features are selected for use in the subsequent stages of fault detection and classification.

4.2. Multi-Step-Ahead Prediction by the Autoencoder

For predicting a multi-step prediction horizon, we experimented with two different LSTM autoencoder models. In the first model, the encoder and decoder have three LSTM layers with 75, 50, and 25 units, respectively, and in the second model, the encoder and decoder have two LSTM layers consisting of 50 and 25 units each. The size of the latent space was set to 15 in each variant. The input window size (w) and the prediction horizon $(\hbar)$ were both set to 20. To evaluate prediction performance by the model, root mean square error was used as the performance metric, which is defined in the following equation:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{\hbar}\left(\hat{Y}_i - Y_i\right)^2}{\hbar}} \tag{12}$$

Out of the 17,000 data points, the first 14,000 data points were used to train the model, and the rest were used for testing. The training and testing segments are presented in Figure 8.

Several experiments were performed by varying the batch size and the number of epochs, as listed in Table 3. The lowest test RMSE of 0.0410 was obtained for a batch size of 64 when the model was trained for 100 epochs. The model was trained using normal sensor signals only. After the model was trained, faulty sensor signal samples are provided as input to predict 20 future data points.

Table 3. List of parameters for model training.

Model 1				
Input Window	**Prediction Horizon**	**Batch Size**	**Epochs**	**MSE**
20	20	64	200	0.0588
20	20	32	200	0.0572
20	20	20	200	0.0582
Model 2				
20	20	64	100	**0.0410**
20	20	32	100	0.0596
20	20	16	100	0.0531

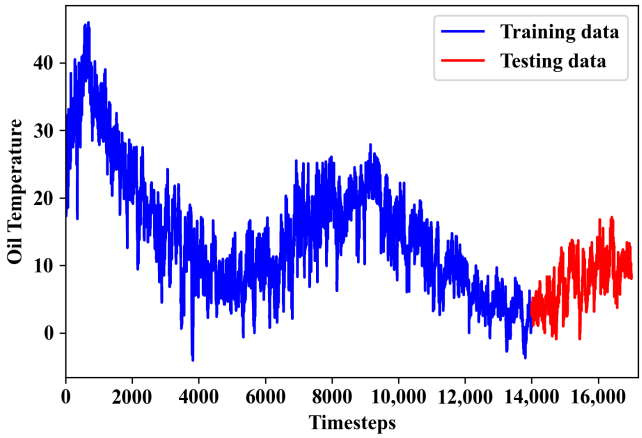


Figure 8. Training and testing segments.

The prediction performance of the autoencoder model for the test data segment is presented in Figure 9, where it is evident that the prediction by the model follows the general pattern of the test data. At this point, the trained model is employed to detect faults by utilizing its ability to predict a window of 20 time steps in the future.

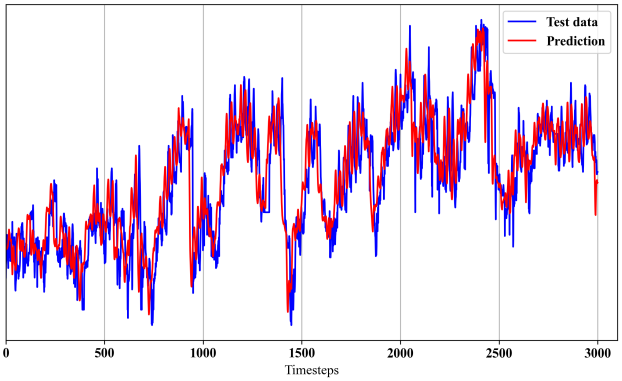


Figure 9. Sensor signal prediction performance by the autoencoder.

4.3. Sensor Fault Detection

The fault detection mechanism can be explained with the help of Figure 10, which illustrates several time steps of a bias fault sample and the corresponding prediction by the model. As discussed in the methodology section, our model takes input window w_1 and predicts the next 20 time steps in $\hat{w}_1$, as indicated in Figure 10. This implies that with this model, we can estimate the future trends in the sensor data, and therefore, if there is an unusual change in the sensor data stream due to the possible presence of a fault, the model can predict the trend a few time steps ahead; thus, the presence of a fault can be determined. Considering input window w_i in the same figure, a bias fault occurred immediately after the segment. Once the model has window w_i as input, it will predict change due to a bias fault, and by calculating the four selected features from the predicted values, decisions can be made as to whether there is a fault in segment $\hat{w}_i$ with the help of the first classifier mentioned in the proposed methodology.

From prior information on fault locations recorded during injection of synthetic faults into the normal sensor signals, we created a labeled dataset consisting of the four selected features calculated from the prediction window, in which the corresponding time steps in the predicted signal are labeled as normal or faulty. Utilizing this dataset with some of the most widely used classifiers to identify whether the future predicted segment is faulty or normal, we exceeded 90% accuracy for all the classifiers. The commonly used classification metrics are listed in Table 4.

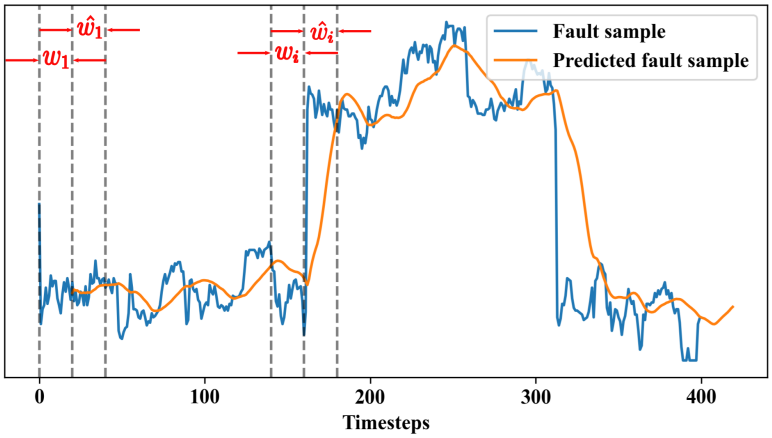


Figure 10. Concept of fault detection with multi-step-ahead prediction.

Table 4. Fault detection performance by the classifiers.

Classifier	Accuracy	Precision	Recall	F1 Score
Random forest	93.39%	95.47%	91.98%	93.70%
XGBoost	93.84%	95.12%	93.25%	94.18%
SVM	92.34%	92.00%	93.81%	92.90%
KNN	93.24%	95.467%	91.70%	93.54%
LightGBM	93.47%	94.83%	92.83%	93.82%

Fault detection performance can be observed more clearly from the confusion matrices in Figure 11. The performance metrics indicate that the boosting algorithm classifiers performed slightly better in comparison to the other classifiers. KNN had comparable performance in detecting the presence of faults in the predicted trends of the sensor signals.

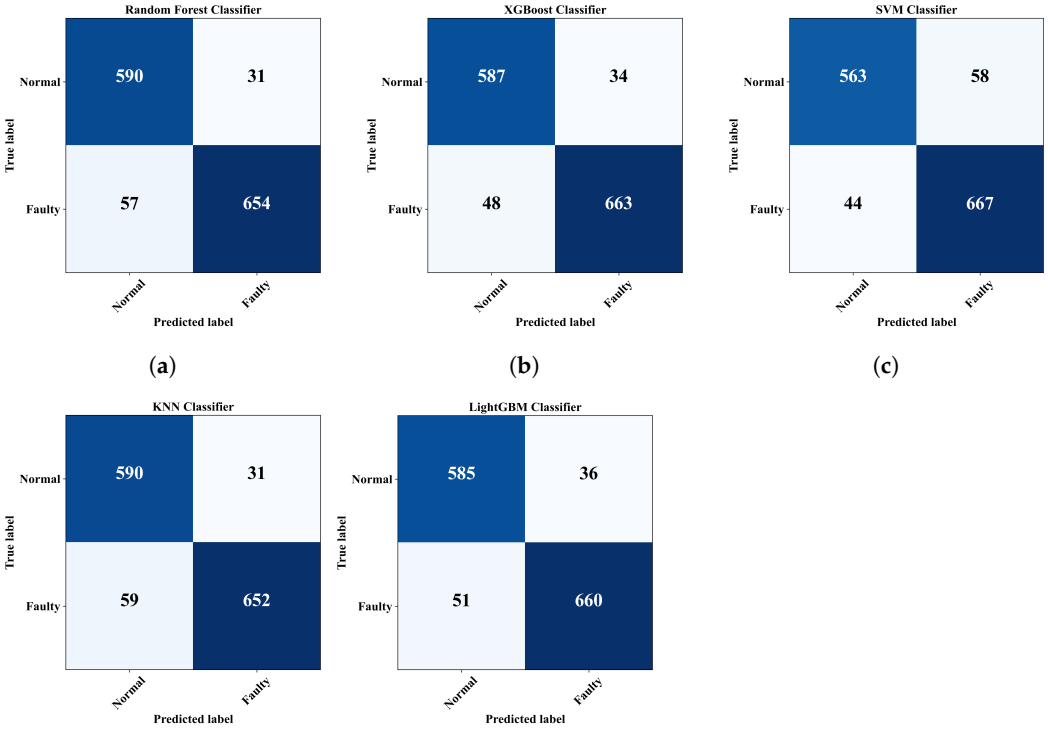


Figure 11. Fault detection performance for different classifiers: (**a**) random forest, (**b**) XGBoost, (**c**) SVM, (**d**) KNN, and (**e**) LightGBM.

4.4. Fault Classification

After the detection of a fault, the next step is to identify the type of fault using ML classifiers. In the dataset that was used for fault detection in the previous step, any of the three types of fault are considered under a single class as a fault. In this step, a second dataset is used, in which the injected time steps are labeled according to the type of fault: bias, drift, or stuck. Fault classification performance by the five ML classifiers is presented with confusion matrices in Figure 12. As can be seen from the confusion matrices, every classifier could recognize the stuck fault accurately, but all of them performed poorly in distinguishing between the bias fault and the drift fault.

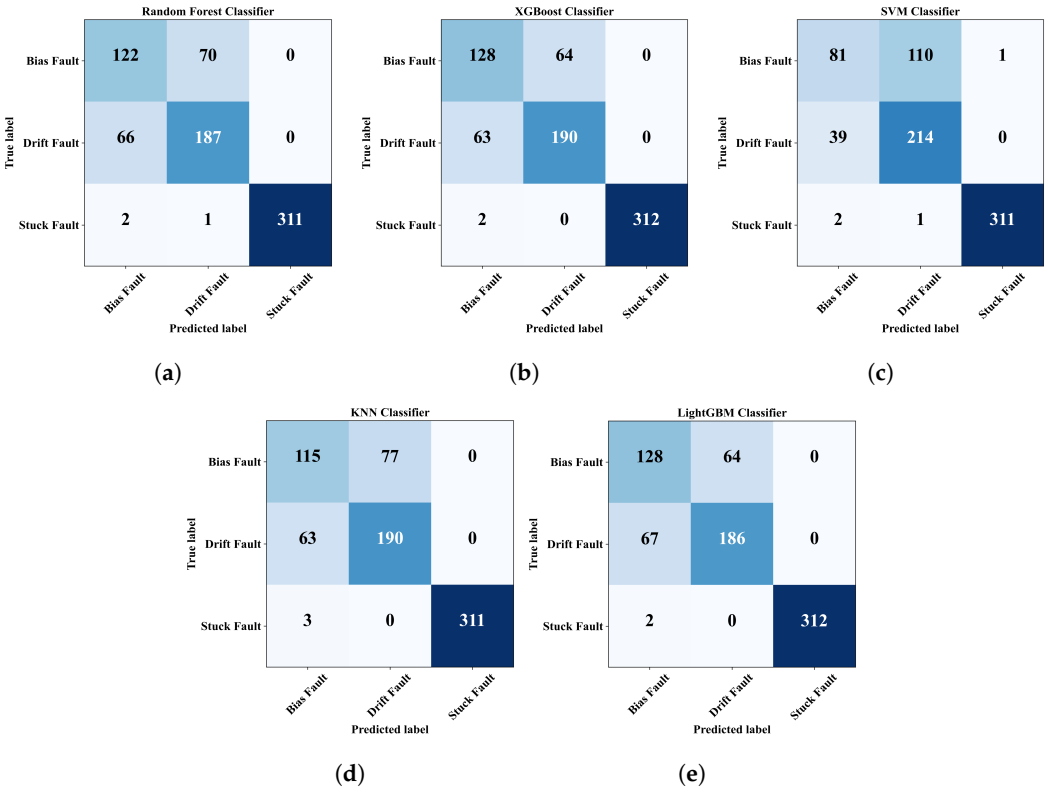


Figure 12. Fault classification performance for different classifiers: (**a**) random forest, (**b**) XGBoost, (**c**) SVM, (**d**) KNN, and (**e**) LightGBM.

This deterioration in classification performance between these two classes is because of the gradual increase in the drift fault signal, which after a certain point equals the amplitude of the signal during a bias fault. The same signal amplitude makes the signature of both faults similar, and thus, it becomes difficult for the classifier to distinguish between these classes. The corresponding metrics are listed in Table 5 for the fault classification task.

Now, to improve fault classification performance, we propose an approach to calculate the feature for more time steps rather than computing features for the prediction horizon of 20 time steps. In the proposed approach, once the presence of a fault is identified by the first classifier, the system will continue to observe the signal for more than 20 time steps and will then extract the selected features for a wider time span. It has been observed that when 100 time steps instead of 20 time steps are considered after the occurrence of a fault, the classifiers can more accurately classify the type of fault with the feature extracted from the extended time steps. The improvement in fault classification is indicated by the confusion matrices in Figure 13.

Table 5. Fault classification performance metrics.

	Bias Fault	Drift Fault	Stuck Fault	Average
Random Forest				
Precision	64.21%	72.48%	100%	78.90%
Recall	63.54%	73.91%	99.04%	78.83%
F1-Score	63.87%	73.19%	99.52%	78.86%
Accuracy	81.82%	78.89%	99.60%	86.77%
XGBoost				
Precision	66.32%	74.80%	100%	80.37%
Recall	66.67%	75.10%	99.36%	80.38%
F1-score	66.49%	74.95%	99.68%	80.37%
Accuracy	83.00%	83.27%	99.74%	88.67%
SVM				
Precision	66.39%	65.85%	99.68%	77.31%
Recall	42.19%	84.58%	99.04%	75.27%
F1-score	51.59%	74.05%	99.36%	75.00%
Accuracy	79.97%	80.24%	99.47%	86.56%
KNN				
Precision	63.54%	71.16%	100%	78.23%
Recall	59.90%	75.10%	99.04%	78.01%
F1-score	61.60%	73.08%	99.52%	78.07%
Accuracy	81.16%	81.55%	99.60%	87.44%
LightGBM				
Precision	64.97%	74.40%	100%	79.79%
Recall	66.67%	73.52%	99.36%	79.85%
F1-score	65.81%	73.96%	99.68%	79.81%
Accuracy	82.48%	82.74%	99.74%	88.32%

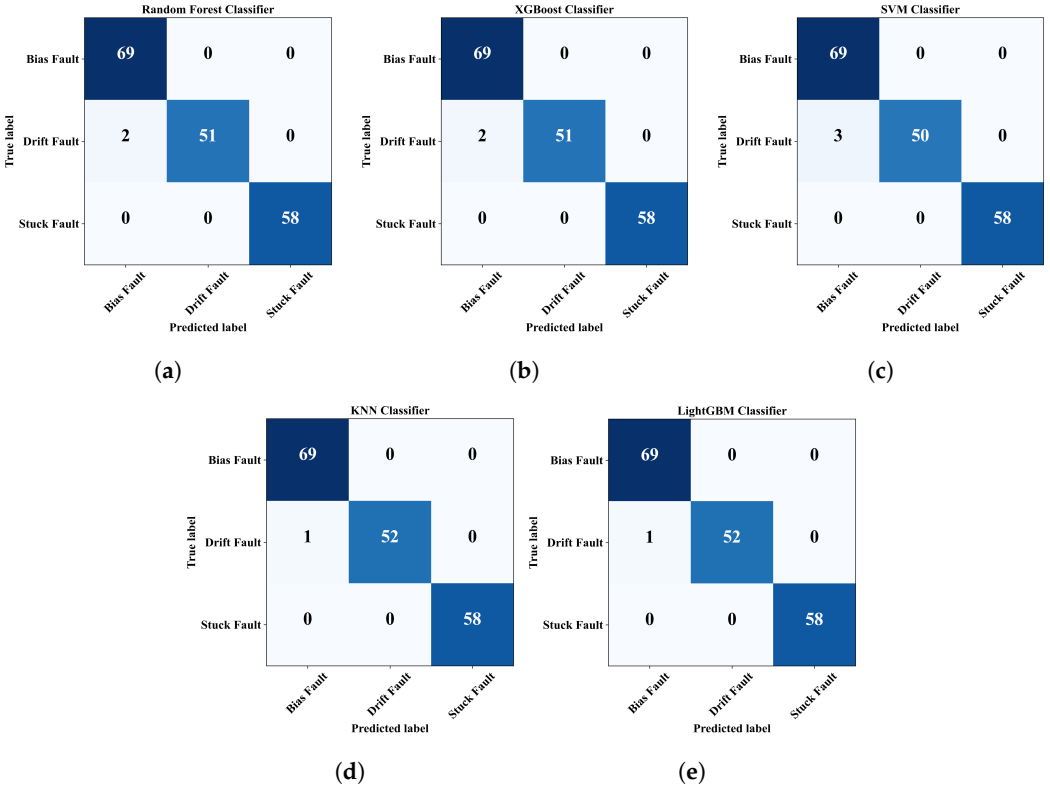


Figure 13. Improved fault classification performance for different classifiers: (**a**) random forest, (**b**) XGBoost, (**c**) SVM, (**d**) KNN, and (**e**) LightGBM.

Turning now to the analysis of the second dataset, we focused on applying the same methodology without reapplying SHAP feature selection. This decision was based on our ability to demonstrate that the features selected from the first dataset for the three different types of faults are equally appropriate for the new sensor signals. Therefore, from the

predicted time steps, only the top three features, as determined previously, were calculated and sent to the first classifier for fault detection. Figure 14 depicts the fault detection performance in the form of confusion matrices obtained from the different classifiers.

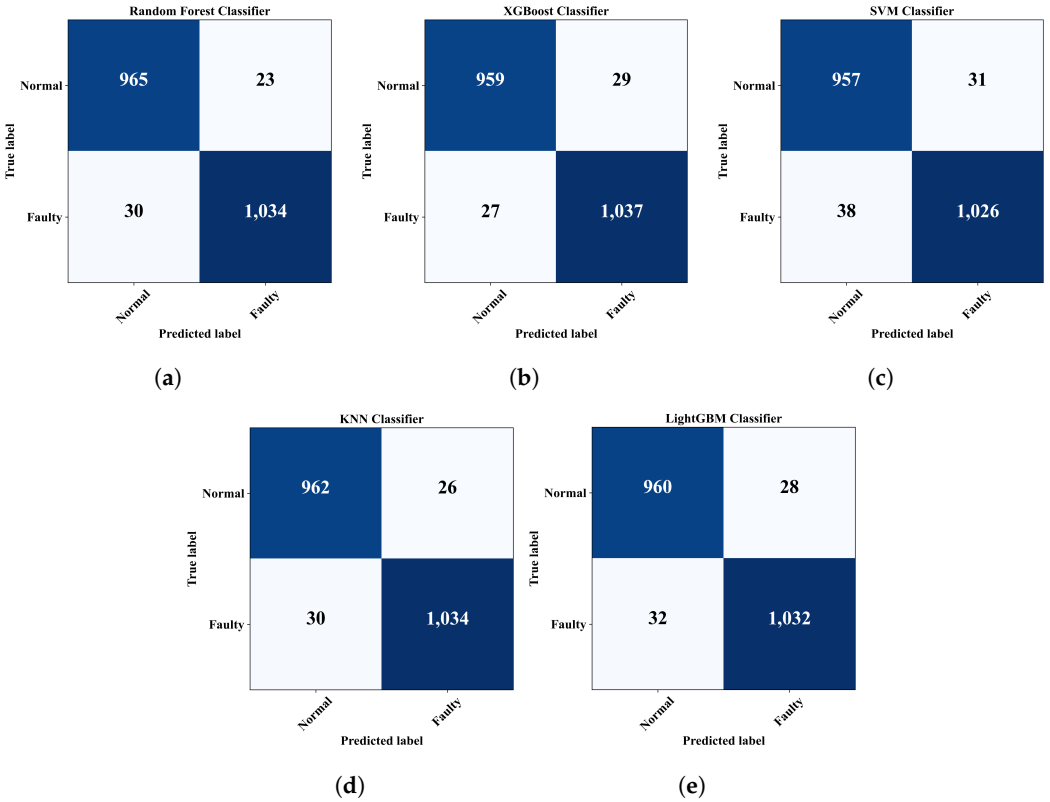


Figure 14. Fault detection performance for different classifiers with the WSN dataset: (**a**) random forest, (**b**) XGBoost, (**c**) SVM, (**d**) KNN, and (**e**) LightGBM

Table 6 lists values from the performance in the fault detection task. The table shows that every classifier can recognize the pattern of the prediction horizons for normal or fault classes with an accuracy of around 97%, except for the SVM classifier, which had a slightly lower accuracy at 96.64%.

During fault type classification, a pattern similar to the previous case was observed. The classifiers could recognize the stuck fault successfully, but for bias and drift faults the classifiers were not very accurate, as demonstrated in the confusion matrices of Figure 15.

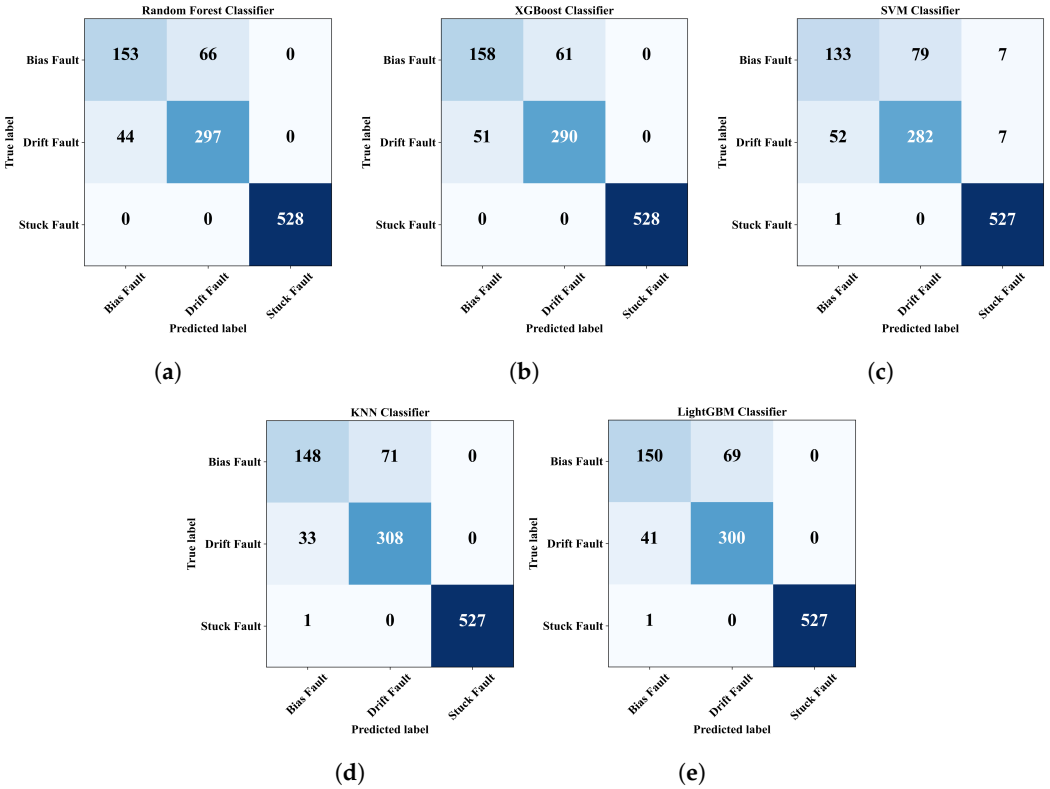
The bar chart in Figure 16 shows the average values for precision, recall, F1-score, and accuracy from the five classifiers considered in this work. The last group of bars shows all the classifiers had above 93% accuracy in classifying the types of fault except for the SVM, which had an accuracy of 91% (about 2% less than the others). Although the average classification accuracy is quite promising, the confusion matrices of Figure 15 show the classifiers did not perform definitively with bias faults and drift faults. These misclassified instances could be reduced by considering multiple prediction horizons after detecting a fault in order to calculate features and then using the classifier.

The improvement in classification when five prediction horizons are considered, rather than one, is visible in the confusion matrices in Figure 17. The improvement in fault classification is because when multiple prediction windows are considered, more time steps are enlisted for calculating features. Thus, it is possible to capture the distinct patterns in two types of faults. In the previous case, the associated features were computed for only 20 time steps, and the misclassified samples aligned with prediction windows where the sensor signal values for bias and drift faults exhibited similarity. Due to this,

the features calculated from those segments were not significantly different, and thus, wrongly classified.

Table 6. Fault detection performance for the WSN dataset.

	Normal	Faulty	Average
Random Forest			
Precision	96.98%	97.82%	97.40%
Recall	97.67%	97.18%	97.43%
F1-score	97.33%	97.50%	97.41%
Accuracy	97.42%	97.42%	97.42%
XGBoost			
Precision	97.29%	97.28%	97.28%
Recall	97.09%	97.46%	97.28%
F1-score	97.19%	97.37%	97.28%
Accuracy	97.28%	97.28%	97.28%
SVM			
Precision	96.18%	97.07%	96.62%
Recall	96.86%	96.43%	96.65%
F1-score	96.52%	96.75%	96.63%
Accuracy	96.64%	96.64%	96.64%
KNN			
Precision	96.98%	97.55%	97.26%
Recall	97.37%	97.18%	97.27%
F1-score	97.17%	97.27%	97.27%
Accuracy	97.27%	97.27%	97.27%
LightGBM			
Precision	96.77%	97.36%	97.07%
Recall	97.17%	96.99%	97.08%
F1-score	96.97%	97.18%	97.07%
Accuracy	97.08%	97.08%	97.08%



(**a**) (**b**) (**c**)

(**d**) (**e**)

Figure 15. Fault classification performance for the different classifiers with the WSN dataset: (**a**) random forest, (**b**) XGBoost, (**c**) SVM, (**d**) KNN, and (**e**) LightGBM.

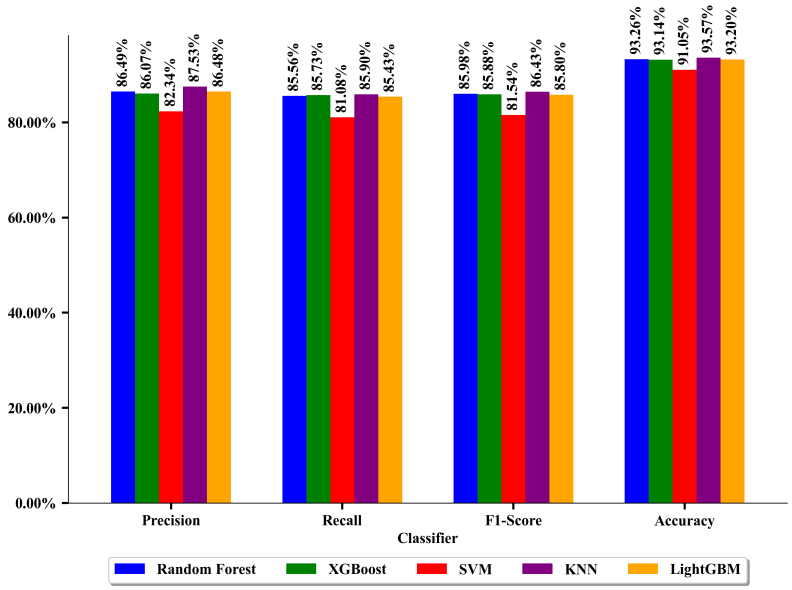


Figure 16. Performance metrics for WSN dataset fault classifications.

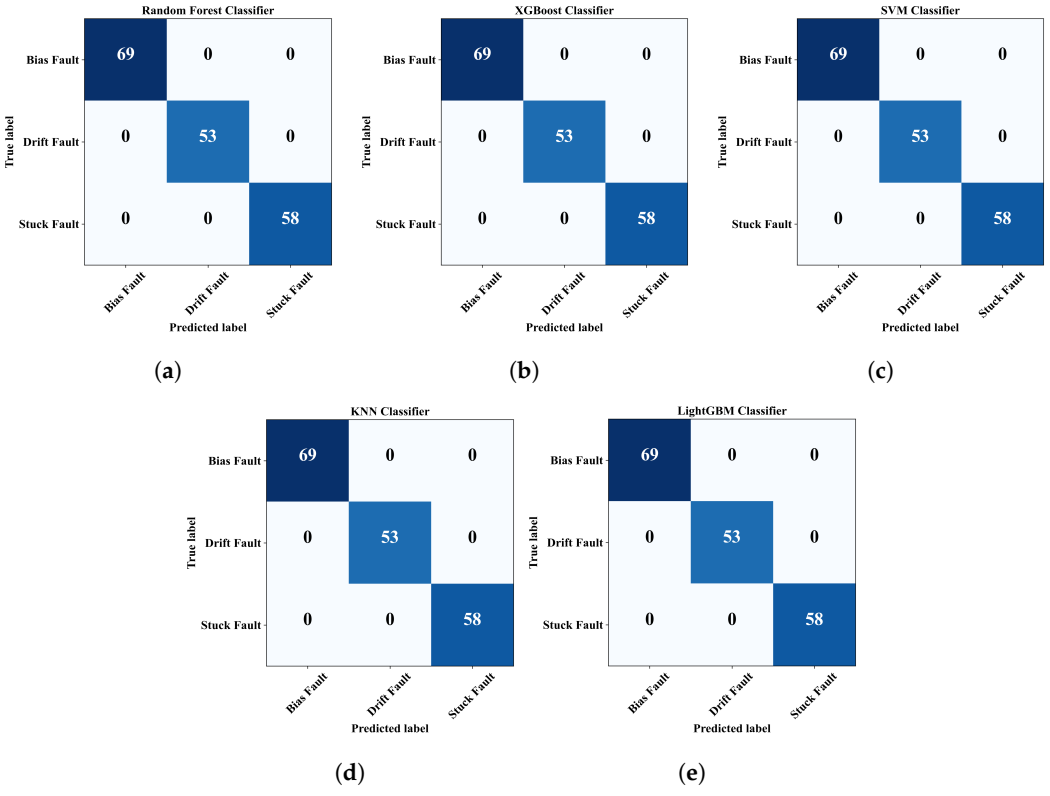


Figure 17. Fault classification performance from the different classifiers with the WSN dataset when considering multiple prediction horizons: (**a**) random forest, (**b**) XGBoost, (**c**) SVM, (**d**) KNN, and (**e**) LightGBM.

In Table 7, a brief comparison of our proposed model with some existing works on sensor fault analysis is presented. As multi-step prediction-based approach is not implemented for sensor fault classification and detection tasks; for that reason, we select works that considered the WSN dataset and common fault types as in this paper. The accuracy score provided for comparing the performance represents the highest accuracy reported in the paper for drift, bias, or stuck fault. The authors of [16,21,21] used the same WSN dataset with SVM and ExtraTree (ET) algorithm. It can be seen that our proposed model classifies

the faults with higher accuracy than ET. The SVM model-based method achieves 100% accuracy, which is comparable with our work but they only considered the classification task. Compared with the results reported in [45], our proposed model achieved higher accuracy scores for both fault detection and classification. Although deep bidirectional Long Short-Term Memory recurrent-neural-network-based deep recurrent canonical correlation analysis (BLCCA) [49] model and Fuzzy Deep Neural Network (FDNN) [55] used different datasets, the classification accuracy for the fault types is significantly higher in our proposed model.

Table 7. Comparison of the proposed method with other works.

Reference	Model	Faults Considered	Task	Performance Metric (Accuracy)
[16]	SVM	Drift, bias, stuck, spike, erratic	Fault classification	100%
[21]	ET	Drift, bias, stuck, spike, erratic, dataloss, random	Fault classification	81%
[48]	ET	Drift, bias, stuck, spike, erratic, dataloss, random	Fault classification	90%
[45]	Multi-layer perceptron	Drift, bias	Fault detection and classification	89.8% and 86%
[22]	DNN	Drift, bias	Fault classification	99.6%
[49]	BLCCA	Drift	Fault classification	92.40%
[55]	FDNN	Drift, bias, stuck, spike, degradation	Fault classification	92.89%
This paper	LSTM-based multi-step prediction	Drift, bias, stuck	Fault detection and classification	93∼97% and 100%

The proposed methodology can be applied to anomalous behaviour detection for all real-world applications where sensors are deployed, for instance, self-driving cars, smart homes, smart energy meters, smart cities, and Internet-of-things. The challenge lies in the collection and use of sufficient realistic for training ML algorithms because of the infrequent occurrence of fault for short intervals of time. Nevertheless, we have designed the LSTM model to be small and identified only four features to work with the ML algorithm, thereby simplifying the approach. Consequently, our proposed method could be a good candidate for real-world implementation.

5. Conclusions

In this paper, a sensor fault detection and classification approach based on the multi-step-ahead prediction technique is explored. An LSTM autoencoder model is used to predict multiple time steps to capture the trend of a sensor signal. In the autoencoder training stage, normal sensor signals are used. As the autoencoder model predicts a window of 20% time steps, estimated time steps are utilized to compute features. In the initial phase of data preparation, feature importance is determined using SHAP analysis to select the most significant features from a set of statistical features that effectively discern bias, drift, and stuck fault patterns in the sensor signals. Based on SHAP analysis, only three features are selected and calculated using an estimated prediction window that has a length of 20% time steps. Considering only three features will ensure less computation complexity and faster classification of faults in the next stage. Once a feature is obtained from the predicted windows, two classifiers are used in the subsequent stages. The first classifier takes the feature point and tries to determine whether the estimated future time steps are faulty or normal. Since the feature point represents characteristics of future data points, the classifier output based on the feature point provides early fault detection. If the next prediction window is classified as faulty by the first classifier, a second classifier in the next stage is used to classify the type of fault. For the two classifiers (fault detection and

classification), two datasets were created for training. In the dataset for fault detection, all the time steps corresponding to the fault were labeled as faulty, irrespective of the type of fault, and the rest were labeled as normal. But the dataset for fault classification consisted of only faulty time step information and had three classes: bias, drift, and stuck faults. The labels are determined from the location of faults in the time steps extracted from synthetic fault injection in the data preparation stage. The proposed methodology was conducted on two different datasets, and five common classification algorithms were used in fault detection and classification. For the first dataset, most of the classifiers achieved about 93% accuracy in fault detection, whereas with the second dataset, around 97% accuracy was attained. In the fault classification step, the identification of bias and drift faults was uncertain, unlike for stuck faults. However, fault classification performance improved if an extended number of time steps were considered for calculating features after fault detection. The multi-step prediction approach has recently gained attention and is mostly implemented in multivariate scenarios. In this work, we experimented with a univariate scenario, and multiple faults were considered. The results indicate that our proposed approach can provide considerable sensor fault detection and classification performance with a single sensor signal. As a future extension of this work, the LSTM-based autoencoder model could be further optimized using advanced optimization methods. Additionally, a broader range of sensor faults could be considered. Furthermore, a more accurate and robust multi-step prediction model could be developed to minimize prediction errors across the horizon, potentially eliminating the need for feature computation and the subsequent machine learning classifier segment. In this scenario, the presence of faults could be determined by observing the difference between the actual values and the predicted time step values.

Author Contributions: Conceptualization, M.N.H. and I.K.; methodology, M.N.H. and S.U.J.; software, M.N.H.; validation, S.U.J. and I.K.; writing—original draft preparation, M.N.H.; writing—review and editing, S.U.J. and I.K.; visualization, M.N.H. and S.U.J.; supervision, I.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Regional Innovation Strategy (RIS) through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE) under Grant 2021RIS-003.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

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

References

1. Famá, F.; Faria, J.N.; Portugal, D. An IoT-based interoperable architecture for wireless biomonitoring of patients with sensor patches. *Internet Things* **2022**, *19*, 100547. [CrossRef]
2. Filho, I.d.M.B.; Aquino, G.; Malaquias, R.S.; Girão, G.; Melo, S.R.M. An IoT-Based Healthcare Platform for Patients in ICU Beds During the COVID-19 Outbreak. *IEEE Access* **2021**, *9*, 27262–27277. [CrossRef]
3. Forcén-Muñoz, M.; Pavón-Pulido, N.; López-Riquelme, J.A.; Temnani-Rajjaf, A.; Berríos, P.; Morais, R.; Pérez-Pastor, A. Irriman platform: Enhancing farming sustainability through cloud computing techniques for irrigation management. *Sensors* **2021**, *22*, 228. [CrossRef] [PubMed]
4. Mendes, J.; Peres, E.; Neves dos Santos, F.; Silva, N.; Silva, R.; Sousa, J.J.; Cortez, I.; Morais, R. VineInspector: The vineyard assistant. *Agriculture* **2022**, *12*, 730. [CrossRef]
5. Ahmed, M.A.; Gallardo, J.L.; Zuniga, M.D.; Pedraza, M.A.; Carvajal, G.; Jara, N.; Carvajal, R. LoRa based IoT platform for remote monitoring of large-scale agriculture farms in Chile. *Sensors* **2022**, *22*, 2824. [CrossRef]
6. Bates, H.; Pierce, M.; Benter, A. Real-time environmental monitoring for aquaculture using a LoRaWAN-based IoT sensor network. *Sensors* **2021**, *21*, 7963. [CrossRef]

7. Khan, F.; Siddiqui, M.A.B.; Rehman, A.U.; Khan, J.; Asad, M.T.S.A.; Asad, A. IoT based power monitoring system for smart grid applications. In Proceedings of the 2020 International Conference on Engineering and Emerging Technologies (ICEET), Lahore, Pakistan, 22–23 February 2020; pp. 1–5.
8. Shashank, A.; Vincent, R.; Sivaraman, A.K.; Balasundaram, A.; Rajesh, M.; Ashokkumar, S. Power analysis of household appliances using IoT. In Proceedings of the 2021 International Conference on System, Computation, Automation and Networking (ICSCAN), Puducherry, India, 30–31 July 2021; pp. 1–5.
9. Kumaran, K. Power theft detection and alert system using IOT. *Turk. J. Comput. Math. Educ. (TURCOMAT)* **2021**, *12*, 1135–1139.
10. Alam, T. Cloud-based IoT applications and their roles in smart cities. *Smart Cities* **2021**, *4*, 1196–1219. [CrossRef]
11. Padmanaban, S.; Samavat, T.; Nasab, M.A.; Nasab, M.A.; Zand, M.; Nikokar, F. Electric vehicles and IoT in smart cities. In *Artificial Intelligence-Based Smart Power Systems*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2023; pp. 273–290.
12. Sunny, A.I.; Zhao, A.; Li, L.; Sakiliba, S.K. Low-cost IoT-based sensor system: A case study on harsh environmental monitoring. *Sensors* **2020**, *21*, 214. [CrossRef] [PubMed]
13. Aryal, S.; Baniya, A.A.; Santosh, K. Improved histogram-based anomaly detector with the extended principal component features. *arXiv* **2019**, arXiv:1909.12702.
14. Li, D.; Wang, Y.; Wang, J.; Wang, C.; Duan, Y. Recent advances in sensor fault diagnosis: A review. *Sens. Actuators A Phys.* **2020**, *309*, 111990. [CrossRef]
15. Wu, H.; Zhao, J. Deep convolutional neural network model based chemical process fault diagnosis. *Comput. Chem. Eng.* **2018**, *115*, 185–197. [CrossRef]
16. Jan, S.U.; Lee, Y.D.; Shin, J.; Koo, I. Sensor fault classification based on support vector machine and statistical time-domain features. *IEEE Access* **2017**, *5*, 8682–8690. [CrossRef]
17. Peng, D.; Yun, S.; Yin, D.; Shen, B.; Xu, C.; Zhang, H. A sensor fault diagnosis method for gas turbine control system based on EMD and SVM. In Proceedings of the 2021 Power System and Green Energy Conference (PSGEC), Shanghai, China, 20–22 August 2021; pp. 682–686.
18. Cheng, X.; Wang, D.; Xu, C.; Li, J. Sensor fault diagnosis method based on-grey wolf optimization-support vector machine. *Comput. Intell. Neurosci.* **2021**, *2021*, 1956394. [CrossRef]
19. Naimi, A.; Deng, J.; Shimjith, S.; Arul, A.J. Fault detection and isolation of a pressurized water reactor based on neural network and k-nearest neighbor. *IEEE Access* **2022**, *10*, 17113–17121. [CrossRef]
20. Abed, A.M.; Gitaffa, S.A.; Issa, A.H. Quadratic support vector machine and K-nearest neighbor based robust sensor fault detection and isolation. *Eng. Technol. J* **2021**, *39*, 859–869. [CrossRef]
21. Saeed, U.; Jan, S.U.; Lee, Y.D.; Koo, I. Fault diagnosis based on extremely randomized trees in wireless sensor networks. *Reliab. Eng. Syst. Saf.* **2021**, *205*, 107284. [CrossRef]
22. Huang, J.; Li, M.; Zhang, Y.; Mu, L.; Ao, Z.; Gong, H. Fault detection and classification for sensor faults of UAV by deep learning and time-frequency analysis. In Proceedings of the 2021 40th Chinese Control Conference (CCC), Shanghai, China, 26–28 July 2021; pp. 4420–4424.
23. Gou, L.; Li, H.; Zheng, H.; Li, H.; Pei, X. Aeroengine control system sensor fault diagnosis based on CWT and CNN. *Math. Probl. Eng.* **2020**, *2020*, 5357146. [CrossRef]
24. Zhao, T.; Zhang, H.; Zhang, X.; Sun, Y.; Dou, L.; Wu, S. Multi-fault identification of iron oxide gas sensor based on CNN-wavelelet-based network. In Proceedings of the 2021 19th International Conference on Optical Communications and Networks (ICOCN), Qufu, China, 23–27 August 2021; pp. 1–4.
25. Toma, R.N.; Piltan, F.; Kim, J.M. A deep autoencoder-based convolution neural network framework for bearing fault classification in induction motors. *Sensors* **2021**, *21*, 8453. [CrossRef]
26. Majidi, S.H.; Hadayeghparast, S.; Karimipour, H. FDI attack detection using extra trees algorithm and deep learning algorithm-autoencoder in smart grid. *Int. J. Crit. Infrastruct. Prot.* **2022**, *37*, 100508. [CrossRef]
27. Jang, K.; Hong, S.; Kim, M.; Na, J.; Moon, I. Adversarial Autoencoder Based Feature Learning for Fault Detection in Industrial Processes. *IEEE Trans. Ind. Inform.* **2022**, *18*, 827–834. [CrossRef]
28. Thill, M.; Konen, W.; Wang, H.; Bäck, T. Temporal convolutional autoencoder for unsupervised anomaly detection in time series. *Appl. Soft Comput.* **2021**, *112*, 107751. [CrossRef]
29. Liu, X.; Lin, Z. Impact of Covid-19 pandemic on electricity demand in the UK based on multivariate time series forecasting with Bidirectional Long Short Term Memory. *Energy* **2021**, *227*, 120455. [CrossRef]
30. Du, S.; Li, T.; Yang, Y.; Horng, S.J. Multivariate time series forecasting via attention-based encoder–decoder framework. *Neurocomputing* **2020**, *388*, 269–279. [CrossRef]
31. Gangopadhyay, T.; Tan, S.Y.; Jiang, Z.; Meng, R.; Sarkar, S. Spatiotemporal attention for multivariate time series prediction and interpretation. In Proceedings of the ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; pp. 3560–3564.
32. Han, P.; Ellefsen, A.L.; Li, G.; Holmeset, F.T.; Zhang, H. Fault detection with LSTM-based variational autoencoder for maritime components. *IEEE Sens. J.* **2021**, *21*, 21903–21912. [CrossRef]
33. Kim, Y.; Lee, H.; Kim, C.O. A variational autoencoder for a semiconductor fault detection model robust to process drift due to incomplete maintenance. *J. Intell. Manuf.* **2021**, *34*, 529–540. [CrossRef]

34. Jana, D.; Patil, J.; Herkal, S.; Nagarajaiah, S.; Duenas-Osorio, L. CNN and Convolutional Autoencoder (CAE) based real-time sensor fault detection, localization, and correction. *Mech. Syst. Signal Process.* **2022**, *169*, 108723. [CrossRef]
35. Bai, Y.; Zhao, J. A novel transformer-based multi-variable multi-step prediction method for chemical process fault prognosis. *Process. Saf. Environ. Prot.* **2023**, *169*, 937–947. [CrossRef]
36. Saha, S.; Haque, A.; Sidebottom, G. Analyzing the Impact of Outlier Data Points on Multi-Step Internet Traffic Prediction using Deep Sequence Models. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 1345–1362. [CrossRef]
37. Wan, X.; Farmani, R.; Keedwell, E. Gradual Leak Detection in Water Distribution Networks Based on Multistep Forecasting Strategy. *J. Water Resour. Plan. Manag.* **2023**, *149*, 04023035. [CrossRef]
38. Zhao, H.; Chen, Z.; Shu, X.; Shen, J.; Liu, Y.; Zhang, Y. multi-step-ahead voltage prediction and voltage fault diagnosis based on gated recurrent unit neural network and incremental training. *Energy* **2023**, *266*, 126496. [CrossRef]
39. Guo, B.; Zhang, Q.; Peng, Q.; Zhuang, J.; Wu, F.; Zhang, Q. Tool health monitoring and prediction via attention-based encoder–decoder with a multi-step mechanism. *Int. J. Adv. Manuf. Technol.* **2022**, *122*, 685–695. [CrossRef]
40. Liu, W.X.; Yin, R.P.; Zhu, P.Y. Deep Learning Approach for Sensor Data Prediction and Sensor Fault Diagnosis in Wind Turbine Blade. *IEEE Access* **2022**, *10*, 117225–117234. [CrossRef]
41. Hasan, M.N.; Jan, S.U.; Koo, I. Wasserstein GAN-based digital twin-inspired model for early drift fault detection in wireless sensor networks. *IEEE Sens. J.* **2023**, *23*, 13327–13339. [CrossRef]
42. Kwon, H.; Lee, S. Friend-guard adversarial noise designed for electroencephalogram-based brain–computer interface spellers. *Neurocomputing* **2022**, *506*, 184–195. [CrossRef]
43. Ko, K.; Kim, S.; Kwon, H. Multi-targeted audio adversarial example for use against speech recognition systems. *Comput. Secur.* **2023**, *128*, 103168. [CrossRef]
44. Safavi, S.; Safavi, M.A.; Hamid, H.; Fallah, S. Multi-sensor fault detection, identification, isolation and health forecasting for autonomous vehicles. *Sensors* **2021**, *21*, 2547. [CrossRef]
45. Darvishi, H.; Ciuonzo, D.; Eide, E.R.; Rossi, P.S. Sensor-fault detection, isolation and accommodation for digital twins via modular data-driven architecture. *IEEE Sens. J.* **2020**, *21*, 4827–4838. [CrossRef]
46. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. *AAAI Conf. Artif. Intell.* **2021**, *35*, 11106–11115. [CrossRef]
47. Suthaharan, S.; Alzahrani, M.; Rajasegarar, S.; Leckie, C.; Palaniswami, M. Labelled data collection for anomaly detection in wireless sensor networks. In Proceedings of the 2010 sixth international conference on intelligent sensors, sensor networks and information processing, Brisbane, QLD, Australia, 7–10 December 2010; pp. 269–274.
48. Saeed, U.; Lee, Y.D.; Jan, S.U.; Koo, I. CAFD: Context-aware fault diagnostic scheme towards sensor faults utilizing machine learning. *Sensors* **2021**, *21*, 617. [CrossRef]
49. Gao, L.; Li, D.; Yao, L.; Gao, Y. Sensor drift fault diagnosis for chiller system using deep recurrent canonical correlation analysis and k-nearest neighbor classifier. *ISA Trans.* **2022**, *122*, 232–246. [CrossRef] [PubMed]
50. Sun, Y.; Zhang, H.; Zhao, T.; Zou, Z.; Shen, B.; Yang, L. A new convolutional neural network with random forest method for hydrogen sensor fault diagnosis. *IEEE Access* **2020**, *8*, 85421–85430. [CrossRef]
51. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 4765–4774.
52. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
53. Suresh, V.; Aksan, F.; Janik, P.; Sikorski, T.; Revathi, B.S. Probabilistic LSTM-Autoencoder Based Hour-Ahead Solar Power Forecasting Model for Intra-Day Electricity Market Participation: A Polish Case Study. *IEEE Access* **2022**, *10*, 110628–110638. [CrossRef]
54. Aksan, F.; Li, Y.; Suresh, V.; Janik, P. Multistep Forecasting of Power Flow Based on LSTM Autoencoder: A Study Case in Regional Grid Cluster Proposal. *Energies* **2023**, *16*, 5014. [CrossRef]
55. Jan, S.U.; Lee, Y.D.; Koo, I.S. A distributed sensor-fault detection and diagnosis framework using machine learning. *Inf. Sci.* **2021**, *547*, 777–796. [CrossRef]