Evaluating Concept Drift Detectors on Real-World Data

Ufuk Erol
University of Bristol
u.erol@bristol.ac.uk

Francesco Raimondo
University of Bristol
f.raimondo@bristol.ac.uk

James Pope
University of Bristol
james.pope@bristol.ac.uk

Samuel Gunner
University of Bristol
sam.gunner@bristol.ac.uk

Vijay Kumar
Toshiba Europe Ltd.
University of Bristol
vijay.kumar@bristol.ac.uk

Ioannis Mavromatis
Toshiba BRIL, Bristol
ioannis.mavromatis@toshiba-bril.com

Pietro Carnelli
Toshiba Europe Ltd.
Pietro.Carnelli@toshiba-bril.com

Theodoros Spyridopoulos
Toshiba Europe Ltd.
Cardiff University
spyridopoulost@cardiff.ac.uk

Aftab Khan
Toshiba Europe Ltd.
aftab.khan@toshiba-bril.com

George Oikonomou
University of Bristol
g.oikonomou@bristol.ac.uk

Abstract

Cloud-IoT deployments are ubiquitous and employed in various application domains, including smart buildings. Often employed in public spaces, IoT devices are exposed to various security threats. One such attack is “anomalous concept drift”. It occurs when an attacker tampers with a device causing it to report realistic sensor data that slowly deviates from the correct value. Evaluating concept drift detectors on real-world data is ideal. Though many indoor datasets exist, our real-world dataset provides a natural, long-term collection of indoor environmental sensor readings over six months. The dataset consists of environmental sensor samples collected via eight IoT devices in a real office setting. The dataset is particularly useful for evaluating concept drift detection algorithms as spatial aspects can be used along with the signals. The dataset has been made openly available, and in this paper we use it to inject malicious concept drifts and to evaluate the performance of several drift detection techniques. The injection tool's source code is also publicly available.

1 Introduction

Internet of Things (IoT) networks have become part of daily life, and usage cases are increasing. IoT sensing applications are widely adopted, and interconnected IoT devices are common in various fields, such as industrial monitoring, smart cities, and environmental monitoring. As the number of applications increased, billions of IoT devices have been deployed and generated immense amounts of data. Use cases include indoor air quality [1], occupancy estimation [15, 2], drift detection, and network planning [13]. However, real-world datasets are hard to obtain, and most real-world data is only accessible to the organisations involved in the collection process. Getting access to sensor information facilitates testing, standardisation, and comparison of sensor-related technologies is a significant limitation/obstacle. Such datasets are particularly helpful for research and analysis. For example, Chimamiwa et al. [4] recently provided a dataset over six months for smart homes. Our dataset has been produced over a similar period with similar sensors, however, it is deployed in a working office environment and provides continuous monitoring data (unlike “rolling” deployed sensors in the smart home dataset).

Open access to real-world sensor data can be cooperative and fulfil the requirements of those researchers who do not have the facility or time to generate such comprehensive datasets. Real-world indoor environmental datasets can also accelerate the development of algorithms designed for smart buildings and home automation. However, having access to such a dataset is not enough in isolation. Essential information about the environment and how the dataset has been processed is required to provide context. We provide a publicly accessible dataset collected over six months with various sensors on various endpoints located in different rooms in a busy environment. In addition, we provide details about the environment, giving additional helpful information to derive meaning from the data. Finally, we provide an open-access tool in a Git repository[1] for concept drift analysis.

However, the volume of data comes with reliability and accuracy concerns. The appeal of gathering and studying

[1]https://github.com/jpope8/synergia_datadrift_dataset

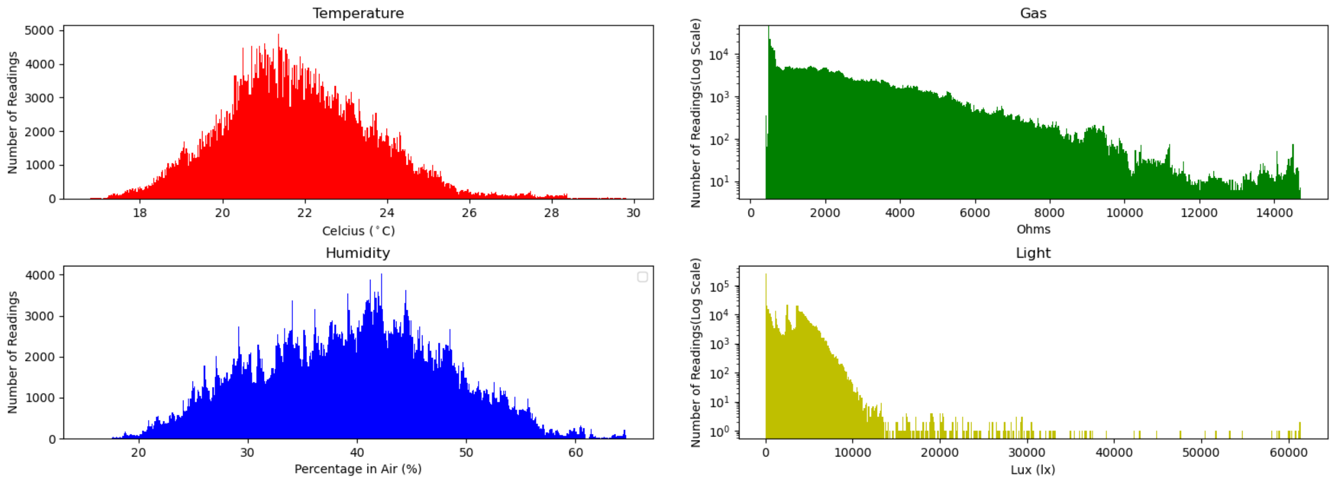


Figure 1: Histogram of Four Sensor Readings of a Single IoT Endpoint

real-world data is that it supports the development of new techniques for data processing. These techniques must be able to cope with the fact that real-world data is not always perfect and can be impacted by issues with the sensors, malicious attacks, and dynamic settings, among other things. In order to maintain healthy and accurate information, distinguishing erroneous and harmful data is essential.

Data drift algorithms play a crucial role in detecting and mitigating malicious attacks using real-world datasets [9]. These algorithms monitor the ever-changing patterns and characteristics of data over time, enabling the identification of any anomalous behavior that may indicate a potential attack. By analyzing the drift in data distribution, these algorithms can detect and raise alerts when data deviates significantly from the expected values. Real-world datasets provide a valuable resource for training and evaluating such algorithms, as they capture the complexities and dynamics of actual environments, including the presence of subtle and sophisticated attacks. Leveraging these datasets, data drift algorithms can effectively enhance the security of systems by promptly detecting and responding to malicious activities, helping organizations safeguard their sensitive information and maintain the integrity of their data infrastructure

The remainder of this paper is structured as follows: Section 2 provides details about the dataset, how the data was collected and the hardware used in the data collection process. Section 3 provides the analysis of the data for malicious and natural data drift detection. Finally, we conclude this paper in Section 4.

2 Dataset and Experiment Overview

To collect real-world data, we deployed an end-to-end IoT network in our office space and laboratory at the University of Bristol. The office area has a maximum occupancy of 28 people and is actively used by a significant number of academic staff, research and taught students. It is located on the second floor thus, it gets exposed to environmental changes such as seasonal temperature, humidity and light fluctuations. The network consists of eight severely resource-constrained IoT endpoints installed at fixed locations, an additional device acting as the "edge", and a server for data collection and controlling the experiment. Each IoT endpoint is equipped with sensing elements providing temperature, humidity, pressure, gas, 3-axes acceleration, and light readings. The endpoints are located in different locations/rooms in the lab to collect varying data due to differentiation between the areas. We collected two additional pieces of information: the measurements' accuracy value, calculated by the environmental sensors and the received signal strength indicator (RSSI) [16]. Sensing elements are sampled every 10 seconds, and sent from endpoints to the edge device. The experiment started in February 2022, and we collected in excess of six months of data (Figure 1). Sensor readings are collected by an application developed in-house, and stored on a cloud server in CSV file format. In the analysis presented in this manuscript we have only used three sensor readings (Temperature, Humidity, and Light), but the full dataset with all sensor readings is openly available [5].

2.1 Resource-Constrained Wireless Sensor Network

Each endpoint device of the network is a data collecting unit and consists of a Nordic nRF52840 DK board [11] and the following sensors:

1. "ISL29125" Light Sensors: Collects intensity of the light.
2. "MMA8452Q" Accelerometer Sensors.
3. "BME680" Environmental Digital Sensors: Sense gas (VOC/CO_2), pressure, temperature and humidity.

Endpoints are identified using both the MAC address and a unique identifier provided by the chip vendor. To easily locate every sensor deployed in the network, a map marking device installation locations is available as part of the data set. The DK board and sensors are connected to every endpoint device using a breadboard. The communication is implemented using the I2C interface. Endpoints are connected in a mesh network topology, where the destination of the endpoints' data traffic is a device acting as the edge of the network. To enable communication between

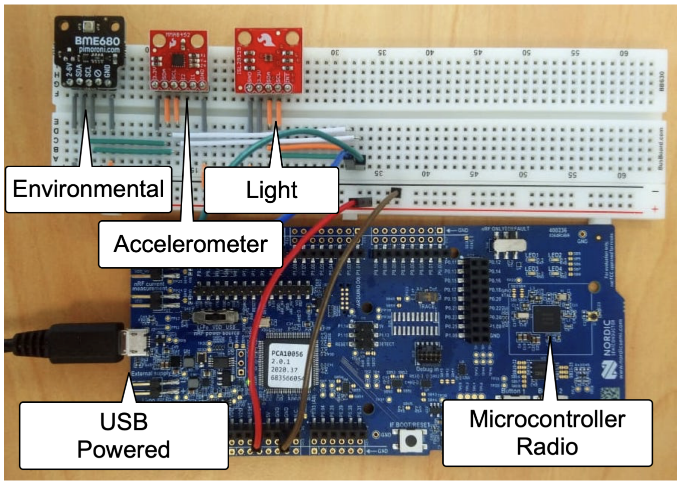


Figure 2: IoT Endpoints with Sensors

the endpoints and the edge of the network, we deployed, on the endpoints' DK board, the Contiki-NG operating system [12]. This provides a full stack implementation for forming mesh networks using IEEE 802.15.4 Time Slotted Channel Hopping (TSCH) MAC protocol [8], an IPv6 network layer and a UDP transport layer. The adoption of TSCH provides a very reliable MAC layer and contributes to obtain healthy continuous data. The device used as the edge of the mesh network is an UMBRELLA edge [3], equipped with a Nordic nRF52840 SoC and a Raspberry Pi. The Contiki-ng border router implementation has been deployed on the nRF52840 SoC. In this configuration, data are received by the nRF52840 SoC, using the IEEE 802.15.4 communication standard, and transferred to the Raspberry Pi.

On the UMBRELLA edge, we execute a series of software services, implemented in Python, providing three functions: 1) control of the experiment, 2) monitoring of the experiment, 3) data file format and storage. The control function communicates with the connected nRF52840 SoC, extracting the incoming stream of sensor readings. Monitoring functions verify that all endpoints are sending sensor data correctly. The detection of an endpoint failure will be reported, providing the date of the failure and the identifier of the endpoint. Finally, the data file format and storage function is responsible for writing the received data in text files, using a *comma-separated values* (CSV) format. Moreover, the same component periodically transfers data to the cloud.

3 Analysis - Drift Detection

To demonstrate the utility of the dataset, we show how it can be used to evaluate drift detection algorithms that might be used to identify compromised IoT devices reporting malicious data. We assume that a device has been compromised and is reporting sensor values intended to misrepresent the environmental state. Malicious data manipulation could be sudden or gradual. We only consider gradual manipulation as we believe sudden changes are easier to detect. There are also natural drifts in the data, that have not been caused by a malicious actor. These natural drifts include seasonal changes (e.g. light, temperature, humidity). Some natural drifts could indicate anomalous scenarios such as temporary heating, ventilation or air conditioning (HVAC) failures, causing deviations from ideal indoor temperatures. Therefore, we consider the two scenarios of *malicious data drift* and *natural data drift*. Before proceeding, we note other use cases of the dataset including occupancy detection, indoor air quality estimation, and evaluation of time-series data generation techniques to address missing data (e.g. long-short term memory neural networks).

For both scenarios, we use the temperature signal for ease of illustration. However, the analysis generalises to the other sensor data. We use the Concept Drift Detection modules from the scikit-multiflow package [10], a machine learning package for streaming data. We chose to use the HDDM_W [7] drift detector which uses an exponentially weighted moving average along with McDiarmid's bounds. We found this implementation was the easiest to tune and produced the most plausible results. The only hyperparameter adjusted was the *lambda_option* that determines the weight given to recent data. We had to adjust this for different sensors and noted the values used in our analysis. In addition to the HDDM_W, we re-test anomaly detection with the Long Short-Term Memory (LSTM) model [14], which is a type of recurrent neural network (RNN) [6].

LSTM networks incorporate memory cells and gates that help them capture and propagate information over longer time intervals, making them well-suited for tasks involving sequential data and dependencies. However, LSTM can be used the detect attacks using a prediction model. We mark the detected abnormal values that are filtered by an algorithm by comparing the predicted values by LSTM algorithm with real received sensor measurement values.

It can be practical to deploy LSTM algorithm to either an edge device or in the cloud to detect malicious attacks.

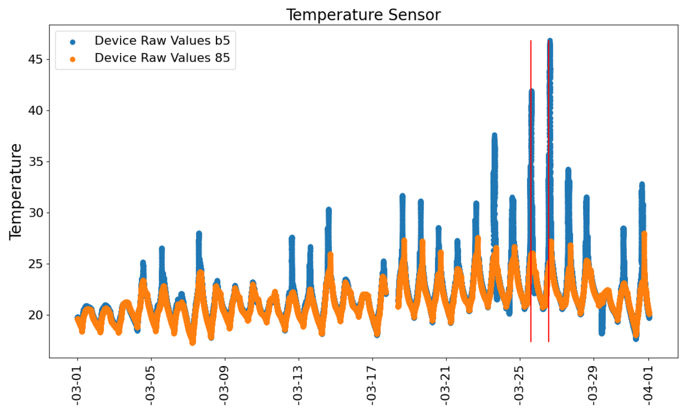
3.1 Natural Data Drift

We consider the two devices, B285 and 2DB5 co-located in the same room. We would expect the two devices to report very similar temperatures and would not expect them to significantly deviate. Figure 3a shows the original temperature samples for the month of March. Interestingly, there is a notable difference between the values towards the end of the month. We believed this was because device 85 was nearer to the window and received more direct sunlight. Figure 5 confirms this to be the case where there is notably more light for device b5 than for device 85 later into March.

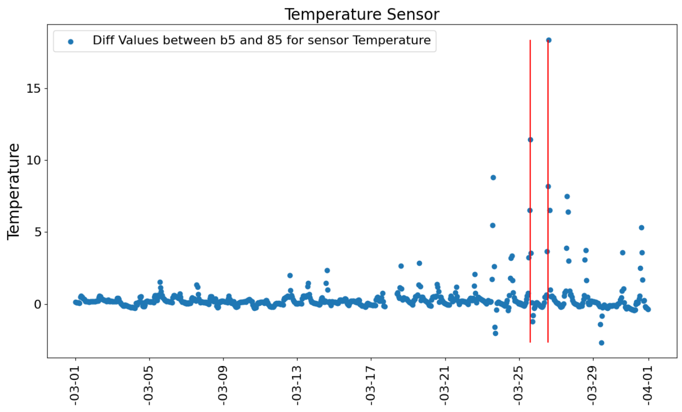
Because the samples have different timestamps, we group them into time intervals of one hour and take the average for each interval. We then subtract one of the signals from the other to produce a difference. Figure 3b shows the results of running the drift detector on the difference signal (using lambda_option=0.01). Two drift events are identified, highlighting the points where the two original signals appear to deviate the most. These detected drifts are shown in both figures by the red line.

3.2 Malicious Data Drift

We developed a simple utility to modify sections of the signals based on date/time ranges. This allows the data in that range to be changed with strictly artificial data. How-



(a) Devices b5 and 85, Original Temperature



(b) Devices b5 and 85, Temperature Difference

Figure 3: Natural Data Drift between Devices 85 and b5 temperatures, March 2022

ever, we choose to inject data that is a function of the original data. This approach will make detection difficult and more indicative of a sophisticated attacker. We take a weighted average of the original data with a constant slope, linear signal to adjust the trend. We modify the signal as follows, where *ts* is the starting time to modify, *t* denotes time, α is a weighting given to original data, and β is a weighting given to the artificial data (α plus β should equal 1.0).

$$data[t] = \alpha * data[t] + \beta * (data[ts] - slope * t) \quad (1)$$

Figure 4a shows the original temperature data collected by device 61 through the month of March. There is a steady rise, most likely caused by the increasing outside temperature expected at this time of year (although a decrease is visible at the end of the month, due to either air-conditioning or the onset of a cold period). The detector (using lambda_option=0.0001) identifies five drifts thought to be natural. We then injected malicious data from March 10 - March 13, with $slope = 0.00001$ and $\alpha = 0.5$. Figure 4b shows the modified temperature data. The detector identifies six drifts when run on this modified data. Notably, a drift is detected near the end of the injected data. We note that injection did not create a significant discontinuity but rather lowered the average value so that when injection ceased on 14 March a detection was generated. Though an attacker would likely continue to modify the data, other techniques, for example the difference signal for nearby devices, could be used to detect the compromised device. Moreover, this demonstrates the utility of the dataset for drift detection.

In case of LSTM detection model, we selected humidity and temperature readings and injected drift with the injection code we provided. We had various LSTM configurations and build the model according to computation as follows:

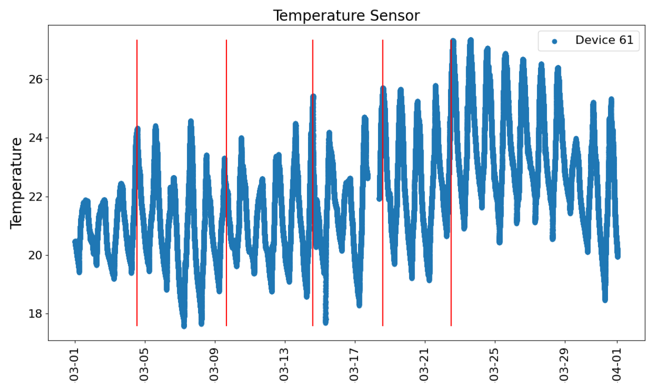
Given an input sequence $\mathbf{X} = \{x_1, x_2, \ldots, x_T\}$, where $x_t \in \mathbf{R}^d$, and the hidden states $\mathbf{h} = \{h_1, h_2, \ldots, h_T\}$ and cell states $\mathbf{c} = \{c_1, c_2, \ldots, c_T\}$ of the LSTM network, the LSTM prediction at time step t can be computed as follows:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, x_t] + \mathbf{b}_f), \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, x_t] + \mathbf{b}_i), \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c \cdot [\mathbf{h}_{t-1}, x_t] + \mathbf{b}_c), \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, x_t] + \mathbf{b}_o), \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),
\end{aligned}$$

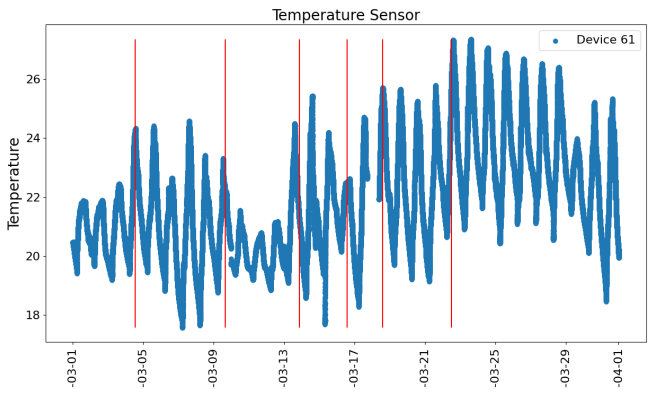
where σ denotes the sigmoid function, $\odot$ denotes element-wise multiplication, and $\mathbf{W}$ and $\mathbf{b}$ represent the weight matrices and bias vectors of the LSTM network, respectively.

- **ft**: The “forget gate” controls the amount of information to discard from the previous cell state, based on the previous hidden state and current input.
- **it**: The “input gate” regulates the amount of new information to be added to the cell state, based on the previous hidden state and current input.
- $\tilde{\mathbf{c}}_t$: The “candidate cell state” represents the new information to be considered for updating the cell state, based on the previous hidden state and current input.
- **ct**: The “cell state” combines the forget gate, input gate, and candidate cell state to update the previous cell state, retaining relevant information and discarding irrelevant information.
- **ot**: The “output gate” determines the amount of hidden state information to expose as the output, based on the previous hidden state and current input.
- **ht**: The “hidden state” is the final output of the LSTM, obtained by applying the output gate to the cell state after passing it through a non-linear activation function.

To detect anomalies, an LSTM model is trained on the provided dataset, where normal instances are represented as the majority class, and anomalies as the minority class. The network learns to model normal behavior by capturing the sequential patterns and dependencies in the data. During



(a) Device 61, Original Temperature



(b) Device 61, Drift (10-13 March) Temperature

Figure 4: Malicious Data Drift, Device 61, March 2022

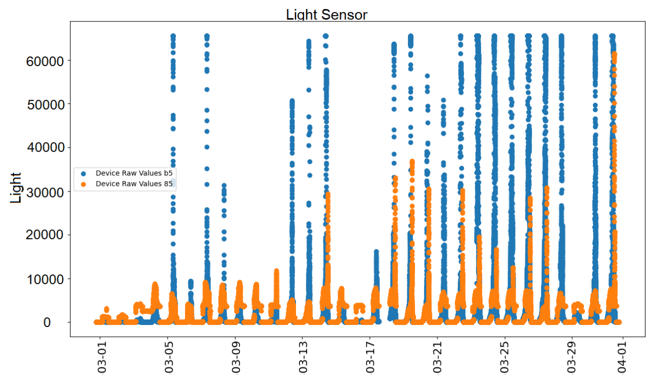


Figure 5: Devices b5 and 85 Light, March 2022

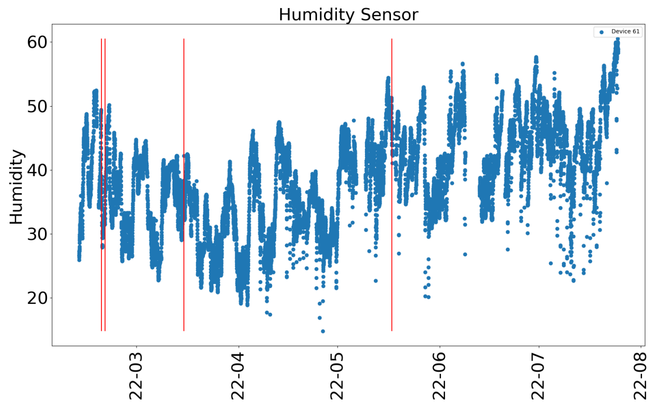


Figure 6: LSTM Data Drift Detection, Humidity

training, the LSTM, RNN adjusts its internal states, known as cell states and hidden states, to remember long-term dependencies and forget irrelevant information.

After we applied the drift-injected dataset and trained the model, the abnormal data was detected by LSTM. We achieve that by comparing the predicted data values with the real measurements provided by the dataset. The detections with red vertical lines can be seen in figure 6 and figure 7.

Once the LSTM model is trained, it can be used to predict the next data value based on the previous sequence of data. When presented with a new data point, the model calculates a prediction error, which measures the discrepancy between the predicted and actual values. Anomalies often result in high prediction errors, indicating deviations from the learned normal behavior.

By setting a threshold for the prediction error, the LSTM can flag instances with errors above the threshold as anomalies. This approach allows for the detection of both point anomalies (sudden deviations) and contextual anomalies (deviations within a specific context). Additionally, LSTM RNNs can handle time-series data with irregular time intervals, missing values, and noisy signals, making them versatile for anomaly detection in various applications such as network intrusion detection, and predictive maintenance.

4 Conclusion

In this paper we have used an open dataset with 6 months of environmental sensor readings to investigate the problem of anomalous data drift in sensor networks. We have presented a technique, and accompanying open source tool, for anomalous data injection, and we have investigated anomalous data drift detection techniques including HDDM_W and LSTM models. Our study concludes that, although the data collected by constrained IoT devices could be manipulated due to their simplicity, it is possible to detect maliciously-injected anomalies.

5 Acknowledgments

This work was supported by UK Research and Innovation, Innovate UK [grant number 53707].

6 References

[1] R. S. Abdul Wahhab. Air quality system using iot for indoor environmental monitoring. In *Proceedings of the 2019 5th International Conference on Computer and Technology Applications*, ICCTA 2019, page 184–188, New York, NY, USA, 2019. Association for Computing Machinery.

[2] S. Ahmed, U. Kamal, T. R. Toha, N. Islam, and A. B. M. A. Al Islam. Predicting human count through environmental sensing in closed indoor settings. MobiQuitous '18, page 49–58, New York, NY, USA, 2018. Association for Computing Machinery.

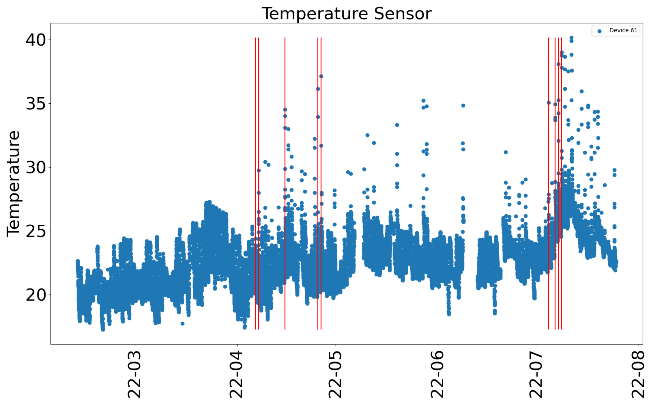


Figure 7: LSTM Data Drift Detection, Temperature

[3] BRIL Toshiba Europe Ltd. UMBRELLA node. `https://www.umbrellaiot.com/what-is-umbrella/umbrella-node/`, 2021. Accessed: 2021-09-06.

[4] G. Chimamiwa, M. Alirezaie, F. Pecora, and A. Loutfi. Multi-sensor dataset of human activities in a smart home environment. *Data in Brief*, 34:106632, 2021.

[5] U. Erol, F. Raimondo, J. Pope, S. Gunner, V. Kumar, I. Mavromatis, P. Carnelli, T. Spyridopoulos, A. Khan, and G. Oikonomou. Multi-sensor, multi-device smart building indoor environmental dataset. https:/doi.org/10.5523/bris.fwlmb11wni392kodtyljkw4n2, 2023.

[6] A. Fathalla, K. Li, A. Salah, and M. F. Mohamed. An lstm-based distributed scheme for data transmission reduction of iot systems. *Neurocomputing*, 485:166–180, 2022.

[7] I. Frías-Blanco, J. d. Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, 2015.

[8] IEEE. Ieee standard for low-rate wireless networks. *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pages 1–709, 2016.

[9] I. Mavromatis, A. Sanchez-Mompo, F. Raimondo, J. Pope, M. Bullo, I. Weeks, V. Kumar, P. Carnelli, G. Oikonomou, T. Spyridopoulos, and A. Khan. LE3D: A Lightweight Ensemble Framework of Data Drift Detectors for Resource-Constrained Devices. In *Proc. CCNC*, pages 611–619, 2023.

[10] J. Montiel, J. Read, A. Bifet, and T. Abdessalem. Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, 19(72):1–5, 2018.

[11] Nordic Semiconductor. *nRF52840*, 11 2021. Rev. 7.

[12] G. Oikonomou, S. Duquennoy, A. Elsts, J. Eriksson, Y. Tanaka, and N. Tsiftes. The contiki-ng open source operating system for next generation IoT devices. *SoftwareX*, 18:101089, 2022.

[13] N. Raj. Indoor rssi prediction using machine learning for wireless networks. In *2021 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 372–374, 2021.

[14] A. Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[15] L. Walmsley-Eyre and R. Cardell-Oliver. Hierarchical classification of low resolution thermal images for occupancy estimation. In *2017 IEEE 42nd Conference on Local Computer Networks Workshops (LCN Workshops)*, pages 9–17, 2017.

[16] R.-H. Wu, Y.-H. Lee, H.-W. Tseng, Y.-G. Jan, and M.-H. Chuang. Study of characteristics of rssi signal. In *2008 IEEE International Conference on Industrial Technology*, pages 1–3, 2008.