|  |
| --- |
| Urban Data Labs |
| **Real-Time Anomaly Detection for Building Sensors** |
| Master of Data Science Capstone |

|  |
| --- |
| Nathan Smith, Mitch Harris, Ryan Koenig  5-9-2021 |

# Executive Summary

Table of Contents

[Executive Summary 1](#_Toc73612011)

[1 – Introduction 4](#_Toc73612012)

[1.1 - Urban Data Lab 4](#_Toc73612013)

[1.2 – Project Scope and Objective 4](#_Toc73612014)

[2 – Study Data 6](#_Toc73612015)

[2.1 – Database 6](#_Toc73612016)

[2.3 – Campus Energy Centre Sensors 6](#_Toc73612017)

[2.4 – Data Anomalies 7](#_Toc73612018)

[3 – Anomaly Detection Framework 9](#_Toc73612019)

[3.1 – Options Considered 9](#_Toc73612020)

[3.1.1 – In-Line with Telegraf 9](#_Toc73612021)

[3.1.2 – Python Client Library 9](#_Toc73612022)

[3.1.3 – Flux Tasks 10](#_Toc73612023)

[3.2 – Framework Selected 10](#_Toc73612024)

[3.3 - InfluxDB Schema 11](#_Toc73612025)

[3.3.1 READINGS Measurement 12](#_Toc73612026)

[3.3.2 TRAINING\_ANOMALY Measurement 12](#_Toc73612027)

[3.3.3. PREDICT\_ANOMALY Measurement 12](#_Toc73612028)

[4 – Anomaly Detection Model 13](#_Toc73612029)

[4.1 – Model Review 13](#_Toc73612030)

[4.2 – LSTM-ED Model 13](#_Toc73612031)

[4.3 – Model Pipeline 15](#_Toc73612032)

[4.3.1 – Data Cleaning 15](#_Toc73612033)

[4.3.2 – Model Training 16](#_Toc73612034)

[4.3.3 – Anomaly Detection 16](#_Toc73612035)

[4.4 – Additional Model Comments 16](#_Toc73612036)

[5 – Anomaly Detection Model Testing 17](#_Toc73612037)

[5.1 – Model Testing Approach 17](#_Toc73612038)

[5.1.1 – Manual Anomaly Labelling 17](#_Toc73612039)

[5.1.2 – Performance Criteria 18](#_Toc73612040)

[5.1.3 – Sensors Tested 19](#_Toc73612041)

[5.2 – Phase 1 Testing Results 20](#_Toc73612042)

[5.3 – Phase 2 Testing Results and Recommended Model 21](#_Toc73612043)

[6 – Dashboard and Notifications System 22](#_Toc73612044)

[6.1 – Dashboard 22](#_Toc73612045)

[6.2 Notifications System 22](#_Toc73612046)

[7 – Conclusion and Recommendations 24](#_Toc73612047)

[7.1 – Conclusion 24](#_Toc73612048)

[7.2 – Recommendations 24](#_Toc73612049)

[7.2.1 – Anomaly Detection Framework 24](#_Toc73612050)

[7.2.2 – Anomaly Detection Model 24](#_Toc73612051)

[7.2.3 – Dashboard and Notification System 25](#_Toc73612052)

[7.2.4 – Performance Monitoring 25](#_Toc73612053)

[8 - References 27](#_Toc73612054)

[Appendix A – Whatever 0](#_Toc73612055)

# 1 – Introduction

## 1.1 - Urban Data Lab

The Urban Data Lab (UDL) was founded in 2019 to advance data access, data management and data analytics capabilities on the University of British Columbia (UBC) campus with the goal of addressing campus-wide sustainability challenges. It accomplishes this by providing open access of UBC sustainability data to researchers, policymakers and operational staff. It also supports the monitoring and measurement of sustainability performance for buildings, transportation, and specifically as it relates to the policy commitments of UBC Sustainability Initiative and Campus and Community Planning [1].

UDL has access to the SkySpark platform managed by UBC Energy and Water Services (EWS). SkySpark is an analytics platform for smart devices and equipment systems and collects data from buildings on the UBC campus including information such as heating, ventilation and air conditioning (HVAC) equipment and energy data. UDL stores data from SkySpark their own database using InfluxDB and the data are made available to students, researchers, and operational staff at UBC. These data are used to support UDL’s collected research and operational interests including energy profiling, fault detection, spatial visualizations, and dashboard decision support systems [2].

## 1.2 – Project Scope and Objective

The project goal was to develop a real-time anomaly detection system using open-source tools that could be used with InfluxDB. UDL have noticed potentially erroneous data reporting to InfluxDB from SkySpark and there is currently no system in place with UDL’s InfluxDB instance to flag these data. The detection system would allow users to understand when data are potentially erroneous such that the anomalous data can be removed from analyses or used to understand if sensors or systems should be investigated. The anomaly detection system would also provide visualization of data identified as potentially anomalous on a dashboard and will ideally provide automated notifications to users. Ideally, the detection system could be improved and used to provide campus and building managers with the real-time or near real-time notifications of potential issues in system operations reducing operational costs, downtime, and unexpected maintenance.

A high-level schematic of the system is provided below.

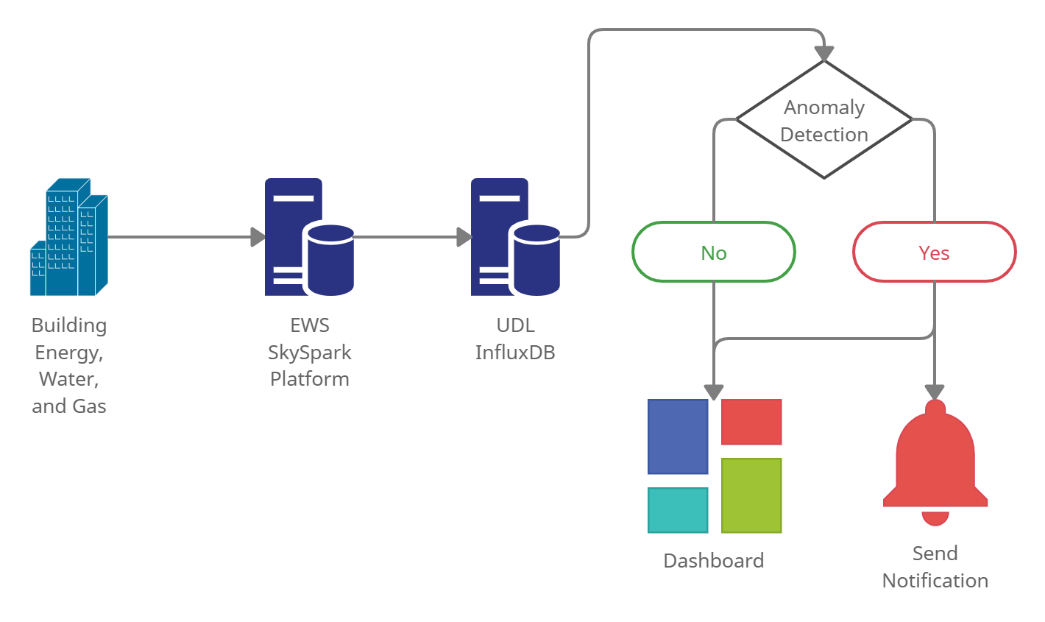


Figure 1: High-Level Anomaly Detection Schematic

The project used the following approach described in this report:

* Select a subset of SkySpark sensor data for the study.
* Review the data subset to understand the variation in sensor patterns and anomalous data.
* Identify a real-time anomaly detection framework that could work with the UDL InfluxDB instance.
* Research and select an anomaly detection model.
* Test performance of the detection model with the selected data subset.
* Implement the real-time anomaly detection model with a dashboard and notification system.

It was also a goal of the project to identify an anomaly detection system general enough that it could be applied to sensors not used in the study subset of data.

# 2 – Study Data

The Campus Energy Centre (CEC) hot water boiler facility sensor data was selected as the subset of SkySpark data for this project. The selection was based on recommendations from UDL as a dataset that would provide a variety of sensor types.

## 2.1 – Database

UDL uses InfluxDB to store data provided from the EWS SkySpark platform. InfluxDB is a timeseries optimized database [3] and UDL is using the open-source implementation of InfluxDB. The database currently only contains limited historical SkySpark data from approximately June to September 2020 and live streaming of SkySpark data to InfluxDB is not functional. UDL is working on implementing live streaming with the database using Telegraf. Telegraf is a plugin driven server agent and is used to listen to http posted data provided by EWS and parse/direct the data to InfluxDB. Support was also provided to UDL during this project for the Telegraf parsing and implementation. Note that UDL is currently working with v2 of InfluxDB.

Data from InfluxDB can be accessed directly through a user interface or using command line interface (CLI) or client libraries. The InfluxDB Python client library [3] was used as the main tool to read and write from InfluxDB for this project.

However, the majority of the data used in the study was downloaded directly from the EWS SkySpark platform user interface due to the limited data available in InfluxDB during the project timeline. This is very time consuming and a limited amount of data had to be selected.

## 2.3 – Campus Energy Centre Sensors

SkySpark platform currently has 72 numerical sensors for the CEC. The sensors have different data resolutions on SkySpark varying from approximately one to 15-minute reading intervals. During the exploratory data analysis (EDA) phase of the project, data were visually assessed using the SkySpark user interface and samples of data were also downloaded for additional exploration (approximately 2-months for sensors with 1 minute interval data and 4 months for sensors with longer intervals).

The sensors can be generally categorized into three main groups based on the exploration:

* Group A: Sensors related to power output of the boilers. This includes power and energy as well as firing rate of the boilers and speed of the pumps. These sensors can have large fluctuations related to the operation of the boilers. This group contains 22 sensors.
* Group B: Sensors that are less likely to experience sharp fluctuations in data compared with Group A. These include temperature (water temperature, exhaust temperature), pressure, concentration, and tank levels. This group contains 41 sensors.
* Group C: Water flow rate sensors. This group contains 9 sensors.

The main purpose of grouping the sensors was simply to provide an initial understanding of the variety of sensors available in the exploration phase. It also provided guidance for selecting a variety of sensors for anomaly detection testing recognizing the project timeline did not provide allowance to individually look at all sensors.

A selection of sensor traces for a two-month period is shown below. The plots include the median as well as 1.5x and 3x the interquartile range (IQR) calculated using the period of record shown on the plots. The IQR values are shown to help provide an understanding of data variability.

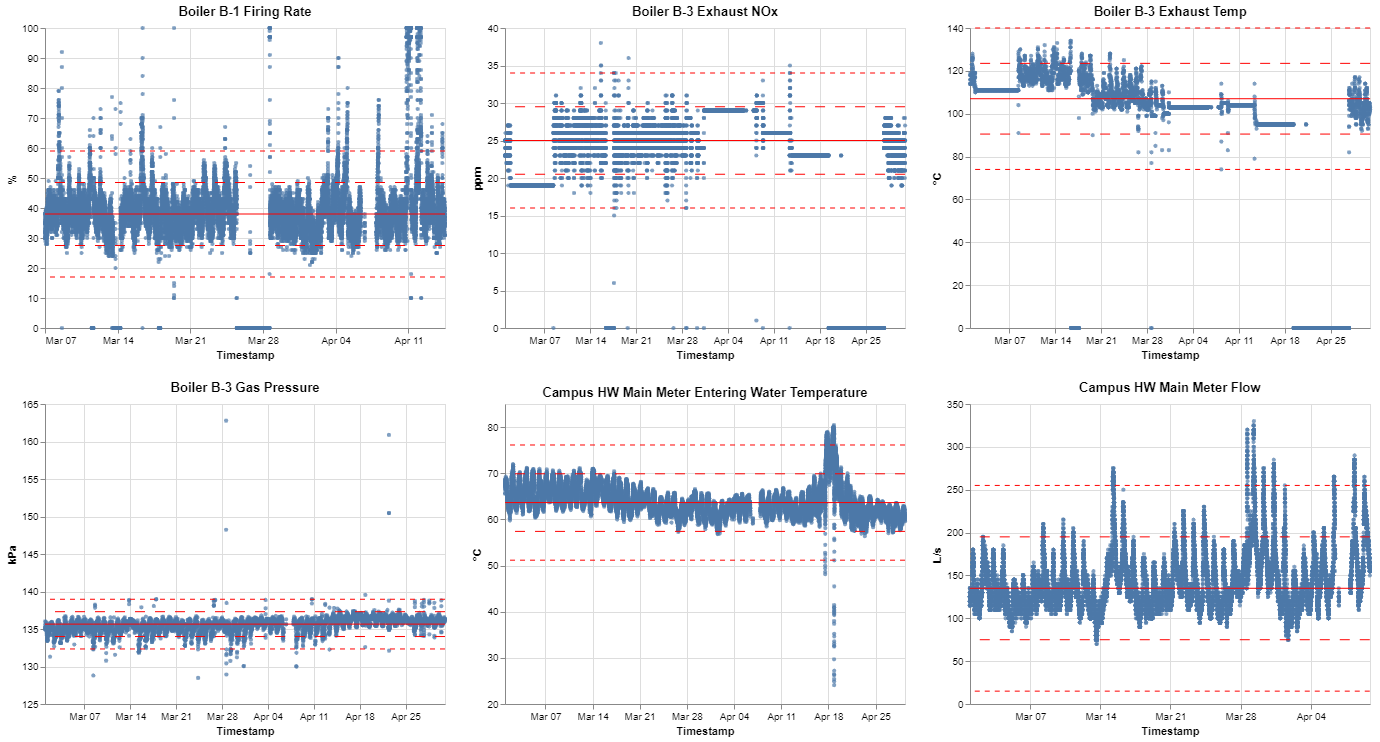


Figure 2: CEC Sensor Data – 6 Sensors, 2-Month Duration

Note: Lines indicate the median, 1.5xIQR and 3xIQR.

The figure shows there is a variety of patterns between the sensors. Some observations from the subset of data shown above include:

* Campus HW Main Meter Entering Water Temperature is generally contained with 3xIQR whereas the Campus HW Main Meter Flow has high fluctuations that appear to be normal data exceeding 3xIQR.
* Boiler B-3 Gas Pressure has short duration spikes that are well out of 3xIQR compared with a longer duration spike that occurs at the Campus HW Main Meter Entering Water Temperature.
* Boiler B-3 Exhaust Temp appears to have steps in the data compared with the Boiler B-1 Firing Rate and Boiler B-3 Gas Pressure which appear to follow a constant trend.
* Boiler B-3 Exhaust and Boiler B-3 Exhaust Temp have periods of flatline data that appear to be erroneous.

Figure 3 shows two sensors over a two-year duration.

Figure 3: CEC Sensor Data – 2 Sensors, 2-Year Duration

Figure 3 shows a seasonal change in the data for asdfasdf while asdfasdfas is constant year-round.

This variety in sensor patterns indicates that a flexible anomaly detection model is likely required if a single model is to be applied to all sensors. It also suggests that model training may be required for individual sensors (or groups of very similar sensors) versus training a single model that could be applied to all sensors.

Additionally, Figures 2 and 3 indicate there appear to be different types data irregularities between the sensors.

## 2.4 – Data Anomalies

Labelled anomalies are not readily available for the CEC sensors. Accordingly, any data that did not appear to follow the typical pattern of data for a sensor was considered as anomalous for the study. These anomalies could be a function of sensor malfunction, equipment malfunction, abnormal system operations, or network related issues resulting in erroneous recordings.

Sensor anomalies could include the following and many of these are shown on Figure 2 and 3:

* Single point anomalies: one or two points that fall well above or below the trend of data.
* Repeated values (flatline) anomalies: the same value repeated for a period of time, not inline with surrounding data patterns.
* Data spikes: a rapid increase/decrease in data compared with the typical data pattern and could represent an operational issue.
* Erroneous values: Sensors recording a value that does not appear correct. For example, a sensor recording zeros or values such as -999 or 999.
* Contextual anomalies: a pattern of data that does not appear standard based on the historical sensor data [4].

These would all be considered anomalous data within this study.

As labelled data was not available for the sensors, the project was framed as an unsupervised modelling approach. However, an exercise of manually reviewing and flagging data that appeared as visually anomalous was completed to help provide an understanding of model performance. Additional details on the manual labelling and performance criteria used are described in Section 5.1.

It should be noted that it is not expected that sensors would have labelled anomalies except in cases when detailed records have been kept (the occurrence of this is unknown). Accordingly, viewing the project as an unsupervised modelling problem is likely representative of how the anomaly detection model would be used.

# 3 – Anomaly Detection Framework

UDL is currently implementing live streaming from the SkySpark platform to InfluxDB using Telegraf. Telegraf is a plugin driven server agent [5] and is used to listen to http posted JSON data provided by EWS. Telegraf parses the data into the line protocol format required by InfluxDB and writes to the database.

The project included identifying an open-source framework for real-time (or near real-time) anomaly detection that works with the InfluxDB/Telegraf. Several options were explored and the selected framework was implemented within a test InfluxDB environment. The test environment was used as live streaming from the SkySpark platform to the UDL InfluxDB instance was not operational during the project. A description of the test environment is provided in Appendix A.

## 3.1 – Options Considered

Several anomaly detection frameworks that could work with InfluxDB/Telegraf are described below. The selected framework is discussed in Section 3.2.

### 3.1.1 – Inline with Telegraf

Telegraf can output data to multiple sources and continuously run executables through its exec and execd plug-ins. This functionality could allow Telegraf to send parsed data directly to InfluxDB as well as to an external program used for anomaly detection. This was initially identified as the option that would provide the most real-time anomaly detection capability on incoming data from SkySpark.

However, this option was not pursued in this project as Telegraf was still being configured by UDL and testing of the system would not have been possible. It is recommended that this option be considered in the future to potentially provide faster detection. There are also several challenges that will need to be considered with this option including:

* Configuring the Telegraf exec/execd plug-ins to run an external program. The external program could include a continuously running python script that provides anomaly detection, or could send data to a server used for prediction.
* Determining how the external program would keep the window of data required for anomaly detection in memory as the program would not be reading data from InfluxDB but only receiving new data streamed from SkySpark.
* Implementing error handling.

### 3.1.2 – Python Client Library

The InfluxDB python client library ‘influxdb-client-python’ [6] allows reading and writing directly from InfluxDB. A python script could be used on a timed interval to read data from InfluxDB, detect anomalies, and write back to InfluxDB. For example, the script could be run every 1-minute to predict the most recent data written to InfluxDB. The script would be configured to read the window of data required by the anomaly detection algorithm.

This approach would not be as real-time as the inline Telegraf option described in Section 3.1.1 but would instead detect using a near real-time approach on a timed interval. This option was selected for this study as it does not rely on operational Telegraf streaming.

A variation on this approach could include running a continuous program (for example using the ReactiveX RxPY package) that would read, detect, and write on a timed basis. This would allow the program to keep a recent window of data required for anomaly detection in memory without having to continuously re-read the same data from InfluxDB. This variation was explored at a high-level basis during this project and is believed to be viable, but was not tested within the project timeline.

### 3.1.3 – Flux Tasks

InfluxDB includes functionality to set automated tasks. These tasks can be used to send data out of the database on a schedule which could be picked up by Telegraf or an API for anomaly detection. This option provides a reasonable approach and uses built-in InfluxDB functionality but was considered similar to the approach using the python client library to read data from InfluxDB. This option was not pursued in favor of using the python client library.

## 3.2 – Framework Selected

The python client library ‘influxdb-client-python’ was selected for implementing near real-time anomaly detection. A schematic of the framework is shown below and the components are described.

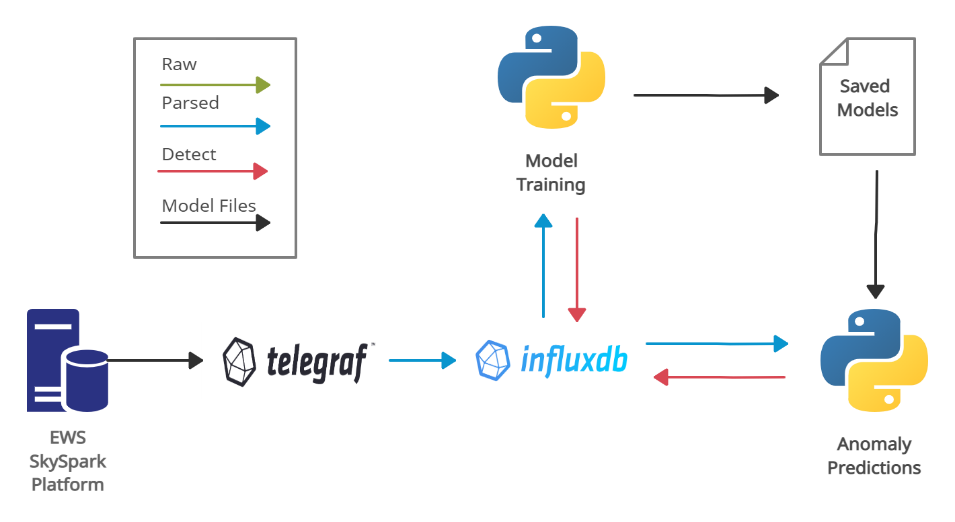


Figure 4: Anomaly Detection Framework

* **Telegraf:** EWS posts SkySpark data in JSON format to an http port. Telegraf listens to the port, parses the JSON data, and write the data to InfluxDB. These data are labelled as READINGS measurements in InfluxDB.
* **InfluxDB:** UDL InfluxDB instance. All data are stored in multiple ‘measurements’ within a single InfluxDB ‘bucket’. A description of the InfluxDB measurements is provided in Section 3.3.
* **Model Training:** A model training python script is run on a selected interval (for example every month) to update the anomaly detection models. A description of the model is provided in Section 4. Anomalies identified during the model training process are labelled as TRAINING\_ANOMALIES measurements in InfluxDB.
* **Saved Models:** The trained models from the model training script are stored in hdf5 files. These files are updated each time the model training script is run.
* **Anomaly Predictions:** The anomaly prediction python script is run on a selected high frequency interval (for example every 1-minute or every 5-minutes). The script reads the most recent data in InfluxDB including the window of data required to make a prediction, loads saved model files, makes anomaly prediction, and writes the result to InfluxDB. These predictions are labelled as REALTIME\_ANOMALIES measurements in InfluxDB.

As discussed in Section 3.1.1, anomaly detection inline with Telegraf likely provides a more real-time detection framework. A schematic of this alternate framework (not tested in this project) is shown below.

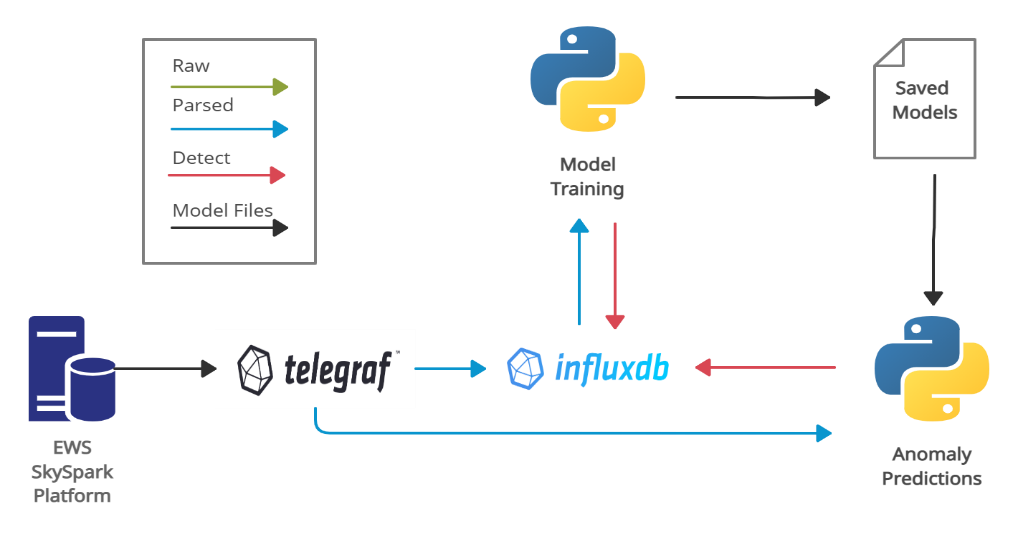


Figure 5: Alternate Anomaly Detection Framework

The only difference with the alternate detection framework is that the anomaly prediction script (or API) would now be sent data directly from Telegraf instead of reading the latest data from InfluxDB.

## 3.3 - InfluxDB Schema

A bucket is a named location where time series data is stored in InfluxDB. Within an InfluxDB bucket, a measurement category act as a container for tags, fields, and timestamps. Tags provide metadata and are typically used in queries while fields represent data value. Each record in InfluxDB has a timestamp [3]. The UDL InfluxDB instance uses a single bucket to store all SkySpark related data. The SkySpark bucket measurements used in the anomaly detection framework are described in the following sections and a schematic showing the relationship between the measurements is provided below. Note that the val\_num field is currently duplicated in the TRAINING\_ANOMALY and PREDICT\_ANOMALY measurements and was done for ease of investigating the dashboard and notification system described in Section 6. This duplication can likely be removed in the future.

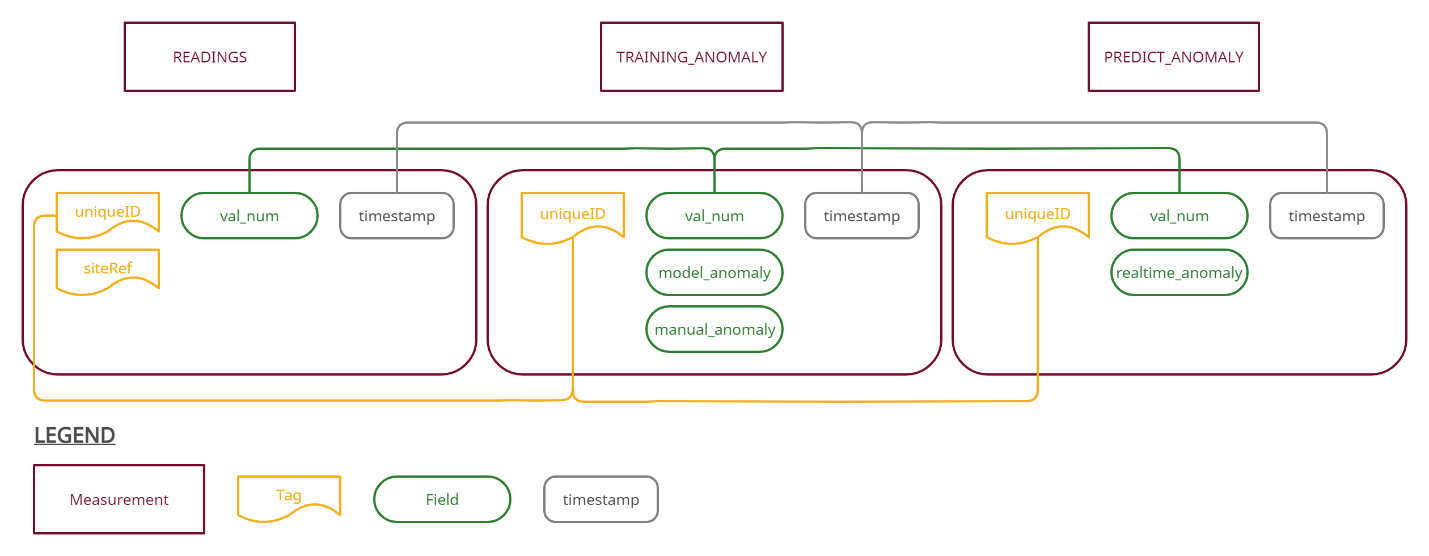


Figure 6: InfluxDB SkySpark Bucket Measurements Related to Anomaly Detection

### 3.3.1 READINGS Measurement

Contains the SkySpark data streamed from InfluxDB. The READINGS measurement has the following tags and fields relevant to the anomaly detection framework:

* **siteRef:** Tag key that specifies a UBC building. This can be used to query all sensors from a specific building on campus.
* **uniqueID**: Tag key that specifies unique sensor IDs. This can be used to query a specific sensor.
* **val\_num**: Field key used for numeric values with sensor data stored as the field value. This study only looks at sensors that have val\_num as the field key.

### 3.3.2 TRAINING\_ANOMALY Measurement

Contains the anomalous data predictions from the training process. Every time model training is completed for a sensor (for example on a monthly basis), the predictions are rewritten. Also contains a field for manually labelled anomalies. The TRAINING\_ANOMALY measurement contains the uniqueID tag key, val\_num field key with the sensor values and:

* **model\_anomaly**: Field key used to store the boolean predictions from model training (False = normal, True = anomalous). The field values are rewritten every time model training is completed.
* **manual\_anomaly:** Field key used to store manually labelled anomalous data (False = normal, True = anomalous). Any field values labelled as anomalous (True) are removed from the training process as described in Section 4.

### 3.3.3. PREDICT\_ANOMALY Measurement

Contains the anomalous data predictions from the anomaly predictions script. Only contains predictions that were made in real-time (does not include any data from the training process). The PREDICT\_ANOMALY measurement contains the uniqueID tag key, val\_num field key with the sensor value and:

* **realtime\_anomaly**: Field key used to store the boolean predictions from the anomaly prediction script (False = normal, True = anomalous).

# 4 – Anomaly Detection Model

## 4.1 – Model Review

The model criteria for this study included the following:

* Provide anomaly detection on univariate data such that the model could be applied to individual sensors without a requirement of additional data.
* Be general/flexible enough that the model could be applied to any building sensor. While the project study used the CEC boiler sensors, the goal was to provide an anomaly detection method that could be applied to many sensors.
* Useable in a continuous predict-detect approach such that model training was not required every time a prediction on new data was made.
* Trainable in an unsupervised approach recognizing that there is unlikely to be labelled data available for sensors.

An initial review of anomaly detection models for sensor data was initially completed with a focus on the above criteria. Internet of Things (IoT) sensor studies was used for the review as they represent a wide variety of univariate data and appear to be the focus of recent sensor anomaly detection studies. General categories of anomaly detection models identified based on the review include [4, 7, 8]:

**Statistical:** Simple statistical rules include using a set number of standard deviations or multiplication of the inter-quantile range (IQR) to identify anomalous data. More sophisticated approaches include using a model such as Autoregressive Integrated Moving Average (ARIMA) to predict expected future data and compare if actual data is sufficiently different from the predicted value to be labelled as anomalous.

**Machine Learning:** Machine learning models typically provide an approach that typically provides less inferential information compared with the statistical model. Models include using k-Nearest Neighbour (KNN), One-Class Support Vector Machine (OCSVM), Isolation Forest, and Gaussian Mixture Models. Many of these models are used in anomaly detection by identifying clusters of data and labelling anomalies as data that are not near these clusters.

**Deep Learning:** Several neural network approaches have been used successfully with IoT sensors for anomaly detection. Examples include one-dimensional convolutional neural networks (CNN) or long short-term memory recurrent neural networks (LSTM RNN). These represent very flexible models but provide more of a black-box modelling approach.

The above categories represent a high-level classification and Cook et al. [4] provides a good discussion of recent anomaly detection methods. There are also methods that use a combination of approaches or even ensembles of model. For example, Microsoft’s SR-CNN Anomaly Detector accessible through the Azure cloud platform uses a spectral residual transformation combined with a CNN [9].

There are also many variations that have been studied for individual anomaly detection models. For example, a recent study on IoT-based vertical plant wall for indoor climate control sensors [7] looked at the performance of multiple LSTM architecture variations including bi-directional LSTM (bi-LSTM) and an LSTM with an encoder-decoder (LSTM-ED). The study also discussed variations of the anomaly detection approach used with the model. For example, the LSTM model could be used to recreate a sequence of input data for comparison with the original input data (sequence reconstruction) or could be used to predict the next point (point prediction) for comparison with the actual next data point. A poor sequence reconstruction or point prediction would indicate potential anomalous data.

In summary, there are multiple categories of anomaly prediction model and each category has many options. Furthermore, there are approaches that include using multiple models or data transformations and there are variations on each of the models as noted in the LSTM example above.

## 4.2 – LSTM-ED Model

The LSTM was selected for the study based on the model review completed for the following reasons:

* Good performance on recent IoT sensor studies and provides a flexible model that should be applicable to a variety of sensors types. Minimal project data was available during the model review process and selecting a flexible model was an important consideration. The project goal is also to provide an approach/model that can be used on sensors not used in this study.
* The model can be used in an unsupervised approach.
* Model parameters can be saved after training and loaded during model prediction.
* The model is sufficiently simple enough that it could be understood and built within the project timeline. Microsoft’s SR-CNN anomaly detection model looks like it could provide good performance but is more complex than the LSTM and the method requires anomaly injection to allow the model to be used in an unsupervised approach.

Specifically, the LSTM-ED was selected for the study based on a comparison of various LSTM architectures where the LSTM-ED provided the highest performance [7].

The LSTM-ED includes the following architecture with 128 LSTM cells for each of the encoder and decoder layers [7]:



Figure 7: Long Short-Term Memory Encoder-Decoder Model

The first layer of the model includes LSTM cells that act as an encoder on the model input sequence. The second layer also consists of LSTM cells and acts as a decoder from the first layer to the model output sequence. A dense layer is used to connect the LSTM decoder layer and the output. The LSTM model for the project was built using Keras [10, 11, 12].

Two variations on the LSTM-ED model were used in this study: next point prediction, and sequence reconstruction. These variations are described by Liu et al. [7] and are shown in the following figure.

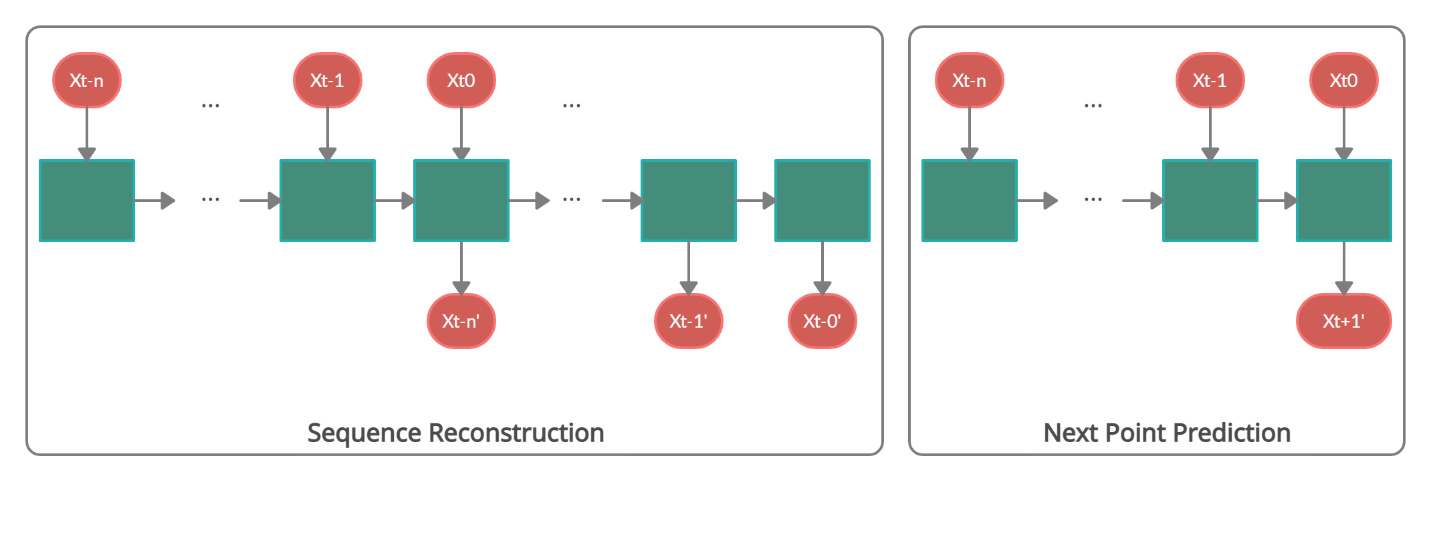


Figure 8: Sequence Reconstruction and Next Point Prediction

Next Point prediction uses a window of input data (xt-n to xt0) and trains the LSTM-ED to predict the next point in the data (xt+1’). The actual value (xt+1) is then compared with the predicted point and if there is a high difference in the value, the point may be flagged as an anomaly. This study looked at various data window sizes (described in Section 5) and trained models by sliding the window by the entire length of the window.

Sequence reconstruction uses a window of input data (xt-n to xt0) and trains the LSTM-ED to predict the same sequence of data (xt-n’ to xt0’). If there is a high difference in the actual and predicted sequence, the data may be flagged as anomalous. There are variations on the sequence reconstruction method including training the model by sliding the window of data one step at a time (overlapping window method), or sliding the entire window by the window length (non-overlapping window), or sliding by several data point (semi-overlapping window) [7]. This study uses the non-overlapping window training approach. Predictions are made on a point of interest by passing the data point as the latest point in the window to the model. If the sequence reconstruction error is high, the point is flagged as anomalous. There are also variations on how to identify data as anomalous with the sequence reconstruction including flagging any data within a window as anomalous, or using a more sophisticated method to track windows with high reconstruction error and only flagging a point as anomalous if it occurs within a certain number of windows with high error [12].

## 4.3 – Model Pipeline

A schematic of the LSTM-ED model pipeline is shown below and descriptions of each component are described in the following sections.

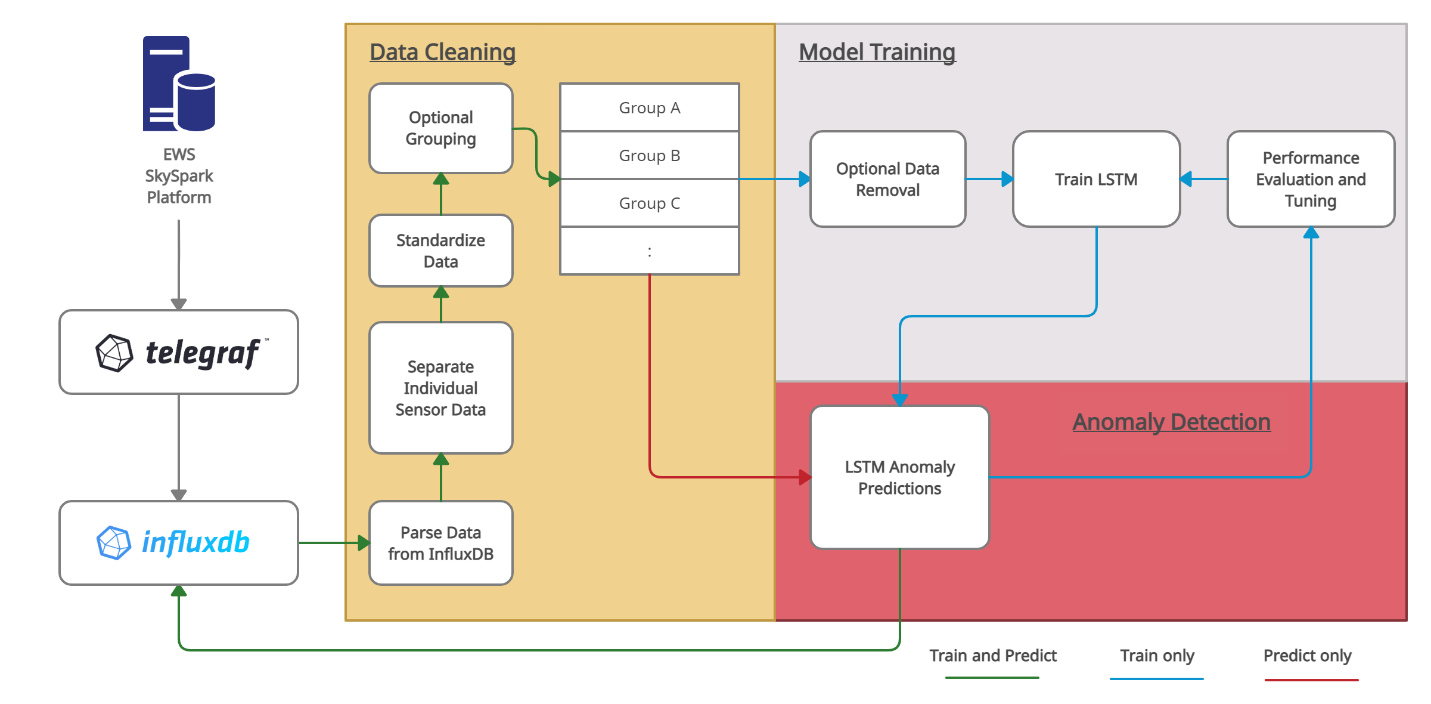


Figure 9: Model Pipeline Schematic

### 4.3.1 – Data Cleaning

The data cleaning component of the model pipeline applies to both model training and model prediction. Data are queried from InfluxDB, parsed and separated into dataframes for individual sensors, and each sensor dataframe is standardized. Sensors can also be optionally grouped. This allows multiple sensors to use a single model and also allows a new sensor with minimal data (insufficient for model training) to use an existing model trained on one or more existing sensors. Note that, while the pipeline includes functionality to allow this sensor groupings, the performance of using a single model for multiple sensors was not been tested in this study.

### 4.3.2 – Model Training

The model training section of the pipeline only applies to the model training script described in Section 3.2. The pipeline includes removing any data manually labelled True (anomalous) in the TRAINING\_ANOMALY measurement manual\_anomaly field. This removal is unnecessary for a typical sensor with small periods of anomalous data, but provides functionality if there are large periods of erroneous data at a sensor. For example, if a sensor has data from March to July but the data in March to May are erroneous, these data can be manually flagged such that they are removed from model training. This can be important as the LSTM-ED is trying to learn the historical pattern of sensor data.

After any manual\_anomaly flagged data are removed, the LSTM-ED is trained on the remaining data. A default window size of x data points is used in training based on the model testing described in Section 5. However, the window size can be adjusted for a sensor if required.

After the model is trained, the model is saved in an hdf5 file. The schematic in Figure 9 shows a loop model training, to LSTM anomaly predictions, to performance evaluation and tuning, and back to model training. This is not an automated process and is just shown to indicate that tuning may be required if the default model values do not provide the required performance.

### 4.3.3 – Anomaly Detection

The anomaly detection section of the pipeline applies the trained LSTM models to the sensor data. The model results are compared with the actual data as described in Section 4.2 (using next point prediction or sequence reconstruction). Data is flagged as anomalous if the error between the model output and actual data exceeds a threshold.

NEED TO EXPLAIN ERROR THRESHOLD METHOD HERE

In the model training script, this anomaly detection process is applied to all data on record for a sensor with the results written to the TRAINING\_ANOMALY measurement model\_anomaly field. In the anomaly prediction script (used for near real-time detection) this is only applied to the period of recent data read from InfluxDB and results are written to the to the PREDICT\_ANOMALY measurement realtime\_anomaly field.

## 4.4 – Additional Model Comments

The LSTM-ED anomaly detection model selected provides a model that should be applicable to a variety of sensors. This generalizability provides a detection approach useable for a variety of systems and was a goal of the project.

However, it should be noted that there may be models or rules that can provide better anomaly detection performance on the specific CEC boiler sensors looked at in this study. The LSTM-ED model looked at in this study is not considered as a replacement for any specific and better performing models or rules for individual sensors or systems. Instead, this model/framework represents a general approach that could be put in place parallel to existing model or rules, or useful for sensors without any existing anomaly detection system. It should also be relatively easy to integrate additional models or rules within the pipeline discussed above.

The model discussed above also uses a univariate sensor approach and does not consider data at multiple sensors. A system (as opposed to sensor) based anomaly detection approach may be more appropriate in some cases. For example, a piece of equipment with multiple sensors may require anomaly detection considering the relationship between sensors. The model in this study could potentially be used as part of a system-based detection approach.

# 5 – Anomaly Detection Model Testing

## 5.1 – Model Testing Approach

The LSTM-ED model was tested on the CEC sensors. These sensors are a selection from the groupings noted in Section 2.3 and include a variety of data patterns and anomalies. Streaming from the SkySpark platform to InfluxDB was not available during the project and these data had to be manually downloaded from the SkySpark platform user interface which is a time-consuming process. It is recognized that this represent a small subset of the 71 CEC boiler sensors available in SkySpark but was considered a reasonable representation of the sensors for this study. The selected sensors are summarized in Appendix X.

The model testing was split into two phases. Phase 1 included testing five of the ten sensors. As the model uses an unsupervised approach and anomaly labels are not available, a manual anomaly labelling process was completed on these five sensors as described in the following section. The labels were used to help asses performance of the model. This first testing phase was used to assess various input data window sizes and compare the next point predictions versus sequence reconstruction methods described in Section 4.2.

A second phase of model testing was completed using the remaining five sensors. These sensors were not manually labelled. The results of Phase 1 testing were used for model setup for Phase 2. The purpose of Phase 2 was to 1) test anomaly detection on additional sensors and 2) assess if the best model setup identified in the Phase 1 still provided good results.

### 5.1.1 – Manual Anomaly Labelling

Manual anomaly labelling was completed on five sensors to support model performance assessment for Phase 1 testing. Records of anomalies for the sensors were not available and the labelling process was based on a visual assessment of data that appeared abnormal. This could include large spikes, flatline data, or irregular patterns based on surrounding data. This process was subjective but still useful for model assessment without the availability of anomalous data records that could provide a quantitative evaluation. Examples of manually identified anomalies are shown in the figure below. A summary of the labelled anomalies is provided in Appendix B.

Figure 10: Phase 1 Sensor Data – Anomaly Examples

A Shiny R application was built as an internal tool to support this manual labelling process. The application allows loading, interactively viewing, and selecting data for labelling as normal or anomalous. The application uses the WebGL implementation of plotly allowing a user to interactively view a large amount of data without performance issues. The application is described in Appendix B as it may be useful for future interactive exploration of data.

### 5.1.2 – Performance Criteria

As discussed, anomalous data labels are not available for the sensors and the LSTM-ED model uses an unsupervised approach. The initial reason for manually labelling anomalous data was to provide some level of quantitative evaluation for the project study. However, it was recognized that labelling process was subjective and it would be difficult to assess model performance quantitatively using the manual labels. It was also observed that the model identified data that while not initially identified and manually labelled as anomalous, did appear to be potentially abnormal data.

Accordingly, it was decided to focus on a qualitative assessment instead of trying to use quantitative measures. This was considered reasonable given the unsupervised modelling approach and the uncertainty in the manual labelling process. The following qualitative criteria was used to assess model performance and allow comparison between model tests:

* **Event Identification**: If the anomaly detection model is identifying events that are considered anomalous. This does not include consideration for the proportion of data points within the event flagged, but just whether some data within the event is identified or not. A poor anomaly detection model would miss many events.
* **Event Coverage**: This includes consideration for the proportion of data identified in an anomalous event Note that it is sometimes difficult to even visually determine the duration of an anomalous event. This performance measure is mainly to provide a general understanding of whether the model is flagging what appears to be the majority of the event, or just one or two points within the event.
* **Initial Event Detection:** How early the model identifies anomalous data point within the event. Ideally the model would label the first point as anomalous.
* **False Anomaly Identification:** If the model is incorrectly identifying normal data as anomalous.

In summary, the qualitative measures are aimed at assessing if the model is identifying events that appear to be anomalous, how much of the event is identified, how early it is identifying the event, and if normal data is being incorrectly labelled as anomalous. These measures were assessed using interactive visualizations of the data. An example of a qualitative assessment is provided in the following table.

Table 1: Qualitative Model Assessment Example

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sensor** | **Test** | **Threshold Used** | **Event Identification** | **Event Coverage** | **Initial Event Detection** | **False Anomaly Identification** |
| HW Main Meter Power | 1a | 25% | Most manually labelled data is identified. | Provides complete coverage for most events but sometimes continues to identify past the event. | Typically delayed by 1 or 2 data points. | Minimal |

It is recognized that this model assessment method is subjective but it was found to provide a reasonable understanding of model performance and comparison of models. The anomaly detection model performance will likely also be assessed on a qualitative basis when in operation using a similar criterion.

### 5.1.3 – Sensors Tested

Model testing of the LSTM-ED was completed on the subset of ten CEC boiler sensors described in Table 2 and shown on Figure 11. These sensors were selected as they provide a selection from each sensor group described in Section 2.3, have varying data intervals, and provide a variety of data patterns and anomalies.

Table 2: Sensors used in Model Testing

* HW Main Meter Power
  + Sensor from Group A related to power output.
  + Includes rapid changes in values related to operation of the boilers.
  + Interval: 1 minute
  + Period: x to y
  + Additional comments: none
  + Testing Phase: 1
* HW Main Meter Entering Water Temperature
  + Sensor from Group B related (temperature sensor)
  + Daily fluctuations, less rapid changes that observed in the HW Main Meter Power sensor
  + Interval: 1 minute
  + Period: x to y
  + Additional comments: includes periods of abnormal data in summer of x, y, z but not summer a. Correspondence with EWS indicates that these periods correspond with the periods with the CEC was not in use.
  + Testing Phase: 1
* HW Main Meter Flow
  + Sensor from Group C related to flow.
  + Daily fluctuations, less rapid changes that observed in the HW Main Meter Power
  + Interval: 1 minute
  + Period: x to y
  + Testing Phase: 1
* Boiler B-1 Gas Pressure
  + Sensor from Group B (pressure sensor).
  + Daily fluctuations with short but high spikes in values.
  + Interval: 15 minutes
  + Period: x to y
  + Testing Phase: 1
* Boiler B-1 Exhaust O2
  + Sensor from Group B (concentration sensor)
  + Constant data that changes values infrequently.
  + Interval: 15 minutes
  + Period: x to y
  + Testing Phase: 1

Figure 11: Sensors Used in Model Testing

## 5.2 – Phase 1 Test Results

The following model variations were looked during the first phase of model testing:

* Window Size: this typically varied between x and y data points (corresponding to x to y minutes depending on the sensor recording interval).
* Next Point Prediction vs Sequence Reconstruction
* Anomalous Data Removal: removal of large periods of erroneous sensor data to understand the impact on model performance.

The model tests are summarized in Table 3.

Table 3: Phase 1 Model Tests

A summary of the qualitative model assessment results is provided in Appendix C. Appendix C provides a table of the qualitative assessment and corresponding figures.

The results of Phase 1 testing indicates that:

* Sequence reconstruction appears to perform better than next point prediction. Both methods are good at identification of anomalous events but next point prediction typically only identifies a subset of data points within an event.
* A shorter window size appears to result in good event identification performance while having less false positives compared with longer windows. A window of x was selected as a good default value. A shorter window size also results in a faster model training time.
* Removal of large periods of erroneous data (for example months of data) improves model performance but removal of small periods does not have an impact.

Examples of the difference in performance is shown in the following figure. Additional figures from the tests can be found in Appendix C.

Figure 12: Phase 1 Model Test Examples

## 5.3 – Phase 2 Test Results

Phase 2 testing used sequence reconstruction and a 1-hour window size for the LSTM-ED. A summary of the tests is shown in the following table.

Table 4: Phase 2 Model Tests

A summary of the qualitative model assessment results is provided in Appendix D. Appendix D provides a table of the qualitative assessment and corresponding figures.

The second phase tests indicate that the model achieved good performance using the model parameters based on Phase 1 test results. Accordingly, sequence reconstruction will be implemented in the anomaly detection framework with a default window of 1-hr. The window size can be modified as required.

# 6 – Dashboard and Notifications System

A dashboard and notification system were implemented with the test InfluxDB environment described in Appendix A.

## 6.1 – Dashboard

InfluxDB version 2 provides a built-in dashboard system. The dashboard allows visualization of data stored in the database and templates can be saved for future use. Grafana provides an alternate option to InfluxDB and appears to currently have slightly more functionality with many available plug-ins. Grafana can easily be connected to InfluxDB but is a separate program. UDL indicated they would prefer to use the built-in InfluxDB dashboard to store and view data within a single platform. Accordingly, the InfluxDB dashboard was used for this project. Grafana should be considered in the future if additional functionality is required.

The dashboard discussed with UDL and created for the project provides a simple interface to view sensor data with anomalous data highlighted. Figure 13 below shows an example view of the dashboard with the Phase 1 test sensors. Anomalies are shown as highlighted zones on the plots. The user can select if they want to view the manual labelled (TRAIN\_ANOMALY measurement manual\_anomaly field), model labelled (TRAIN\_ANOMALY measurement model\_anomaly field), or real-time labelled (PREDICT\_ANOMALY measurement realtime\_anomaly field) anomalies on the plots.

Figure 13: Dashboard Example

It is recommended to create multiple dashboards for sensors of interest as there are a large number of sensors (the CEC alone has 71 sensors available in SkySpark). Creating a dashboard in InfluxDB is fairly simple and a description is provided in Appendix E.

It should be noted that the dashboard implemented in this project provides a simple solution directly integrated with InfluxDB. More sophisticated options that provide dropdowns or use of multiple tabs for sensor group views could be investigated.

## 6.2 – Notifications System

A notification system was also setup with the test InfluxDB environment used in this study. The notification system uses built-in InfluxDB functionality. InfluxDB can be configured to provide notifications to multiple sources including x, y, and z. This study tested InfluxDB notifications sent to Slack. Notifications were setup to provide slack messages when any of the ten CEC sensors looked at in this study had True (anomalous) values written to the PREDICT\_ANOMALY measurement realtime\_anomaly field. The notification provided the sensor name, value, and timestamp associated with the anomalous flag. The system can easily be modified within InfluxDB and a brief description of how-to setup a notification is provided in Appendix F.

This component of the real-time detection framework likely requires additional testing and configuration before it could be used in an operational setting. This study was limited to testing the capability of InfluxDB to send notification on anomalous data identified at a sensor and was found to be successful. More sophisticated notification systems could include suppressing notifications when equipment is not operational or only providing notifications when multiple sensors detected anomalous data. The capability of the InfluxDB notification system to do this was not assessed in this study.

It should also be noted that the system does not provide notifications on missing data including long duration intervals when data are not provided to InfluxDB. This may represent operational or sensor issues but the model applied in this study currently does not detect this. It should be possible to implement this functionality within the anomaly detection framework discussed in this study.

# 7 – Conclusion and Recommendations

## 7.1 – Conclusion

This project resulted in a proposed approach for a near real-time anomaly detection system with UDL’s database. The approach included a framework for model anomaly detection model training and prediction with InfluxDB using open-source software. Model training is completed by querying sensor data from InfluxDB on an infrequent basis while anomaly detection occurs on a continuous basis by reading recent data from InfluxDB, loading and running the previously trained models, and writing the results back to InfluxDB. This framework was implemented in a test InfluxDB environment.

The anomaly detection model selected is an LSTM-ED using a short-duration (approximately 1-hr window) using sequence reconstruction in an unsupervised approach. The LSTM-ED provides a model that should be applicable to a variety of sensors. This generalizability provides a detection approach useable for a variety of systems and was a goal of the project. The model was tested on ten CEC boiler sensors available from the EWS SkySpark platform. A qualitative performance assessment was completed as anomaly labels for these sensors were not available. The results indicate that the detection model is capable of providing good performance.

A dashboard and notification system were also implemented with the anomaly detection framework in the test InfluxDB environment. The dashboard is built directly in InfluxDB and provides a simple display of sensor data overlaid with anomalous flagged data. The notification system also uses built-in InfluxDB functionality and was configured to send notifications to a Slack channel for data predicted as anomalous.

## 7.2 – Recommendations for Future Study

Ideally, the detection system could ultimately be implemented and used to provide campus and building managers with real-time or near real-time notifications of potential issues in systems operations reducing operational costs, downtime, and unexpected maintenance.

This study resulted in a proposed approach that appears useful for near real-time anomaly detection. The following sections discuss improvements that should be considered to understand the capability for potential operational use.

### 7.2.1 – Anomaly Detection Framework

The anomaly detection framework in this study operates by reading recent data from InfluxDB, providing predictions, and writing results back to InfluxDB. This can be implemented on a timed near real-time basis (for example, every 1 minute or every 5 minutes). Improvements to the current approach include providing a method to avoid re-reading the same data from InfluxDB each time the prediction script is executed. This could either be done by writing data to a temporary file or running the program continuously and keeping the data in memory.

A more real-time approach includes providing predictions inline with Telegraf as shown on Figure 5 in Section 3.2. There are several challenges associated with this including configuration of Telegraf to send the data to an external program, implementing streaming error handling, and keeping the required window of data for predictions in memory.

### 7.2.2 – Anomaly Detection Model

The LSTM-ED model was selected as a model believed to provide good generalization to a variety of systems and sensors based on recent studies. The results indicate good performance on a subset of CEC boiler sensors but the following additional studies should be considered:

* Different types models (for example, bi-LSTM or ARIMA) were not tested in this study based on the limited project timeline. Testing additional models may result in identification of models with higher performance or similar performance but with simpler approaches.
* The study did not have time to assess the use of additional models or rules that could be used in conjunction with the LSTEM-ED. It is expected there are rules and models that have high anomaly detection performance with specific systems and sensors. It would make sense to apply these (where they are known) with this more general model.
* The model pipeline presented in Section 4.3 has the capability to group multiple sensors for use with a single trained model. This could allow a sensor with minimal data to use a model trained on other sensors. This functionality would also fewer models to be trained instead of creating an individual model for each sensor. This grouping should be viable on sensors that are very similar but testing of this was not completed.
* The LSTM-ED in this study only considers univariate data. Modification of the LSTM-ED to use multi-variate data could be considered.

### 7.2.3 – Model Performance

A qualitative performance assessment was completed on ten CEC boiler sensors in this study. A more rigorous model performance assessment could be completed using a quantitative approach. This would likely require identifying sensors with labelled anomalous data or selecting sensors where a simple model of rule could be used to identify anomalous data for comparison.

The study also only looked at a subset of boiler system sensors from a single building. Additional testing should be completed on additional sensors/systems to understand the generalizability of the approach. This could be done in an online approach by implementing the anomaly detection model on additional sensors and monitoring performance.

### 7.2.4 – Dashboard and Notification System

The dashboard and notification system assessed in this study are simple implementations using built-in InfluxDB functionality. The purpose of the assessment was to understand viability and it is expected that additional effort would be required to improve the dashboard and provide a useful notification system.

If it is found that the built-in InfluxDB dashboard does not provide the functionality required for a useable interface, it is recommended that Grafana is further explored. Grafana provides easy integration with InfluxDB and currently appears to have more functionality.

The notification system tested in this study provides notifications to a Slack channel when anomalous data are identified in near real-time anomaly. To use the notification system for an operational system likely requires bringing the anomaly detection system online and monitoring performance before the notification system is turned on for a sensor to ensure the detection system is providing meaningful results. As discussed in Section 4.2, there are several adjustments that can be made to the default model parameters on an as-needed basis including the detection threshold and input data window size.

More sophisticated settings may also be required such as only triggering a notification if multiple sensors detect an anomaly, or turning off notifications when equipment is offline. These were not explored in this study but may be required to improve the operational usefulness of a notification system.

### 7.2.5 – System Anomaly Detection

The anomaly detection framework in this study uses a univariate sensor approach and does not consider the relationships between sensors. The LSTM-ED model tries learn the historical pattern of data for a sensor and identify data that does not follow the normal pattern. There may also be interest in anomaly detection on a system instead of sensor basis. An example could be identification of a potential boiler malfunction versus identification of anomalous data at individual boiler related sensors. The LSTM-ED anomaly detection model could help support identification of system malfunctions/anomalies but was not considered in this study. This task is likely system specific requiring an understanding of the relationships between sensors and input from domain experts. For example, a sequence of sensor anomalies may be meaningful and indicate a potential malfunction, while identification of an anomaly at a single sensor may not indicate an issue depending of the system.

# 8 - References

[1] Wei, J. (n.d.). *About.* Urban Data Lab. Retrieved May 2021, from https://urbandatalab.io/about

[2] Wei, J. (2020, January 9). UBC Building Data. Retrieved May 2021, from <https://urbandatalab.io/news/ubc-building-energy-data/>

[3] <https://www.influxdata.com/products/influxdb/>

[4] Anomaly Detection for IoT Time-Series Data: A Survey

[5] https://www.influxdata.com/time-series-platform/telegraf/

[6] InfluxDB 2.0 python client. Retrieved June 2021, from https://influxdb-client.readthedocs.io/en/latest/index.html

[7] Anomaly detection based on machine learning in IoT-based vertical plan wall for indoor climate control

[8] Detecting the Onset of Machine Failures using Anomaly Detection Methods

[9] Time-Series Anomaly Detection Service at Microsoft

[10] Machine Learning Architecture

[11] Keras general

[12] Keras anomaly detection

# 9 – Acknowledgements

# Appendix A – Whatever