INL/EXT-19-55629

Light Water Reactor Sustainability Program

Subtle Process-Anomalies Detection Using Machine-Learning Methods

September 2019

U.S. Department of Energy

Office of Nuclear Energy

INL/EXT-19-55629

Subtle Process-Anomalies Detection Using Machine-Learning Methods

Ahmad Al Rashdan, Michael Griffel, Roger Boza, and Donna Guillen

September 2019

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy

ABSTRACT

Operators at a nuclear power plant (NPP) are responsible for several tasks that can result in the operations organization becoming the most burdened organization in the plant. In addition to visually monitoring plant instruments in digital or analog form to keep track of plant conditions, operators are tasked with changing the plant configuration when needed, performing operator rounds, conducting and documenting surveillances, issuing clearance orders (including tagging in and out equipment), coordinating with other plant organizations, and performing random walks and checks of the plant. With this level of tasking, operators tend to adopt a reactive model to plant monitoring—i.e., responding to alarms and events—rather than a proactive model, under which small signs of anomalies can be detected, tracked, and mitigated before the issue escalates. Escalation of an anomaly to a functional failure can result in severe economic or safety consequences to the plant and demand a responsive effort with costs that often far exceed what would have been invested to mitigate the issue.

Techniques that have been developed to proactively detect anomalies in a plant are based on manual inspection through visual or physical checks on equipment by surveillances, preventive maintenance, and operator rounds. The nuclear power industry has recently become interested in implementing automated monitoring through retrofitting equipment that are not currently instrumented with permanent condition-monitoring sensors capable of transferring data to custom-developed methods, revealing the condition of the equipment. Deploying this approach on a large scale is an investment that requires resources the industry is reluctant to allocate under current economic conditions, and would require significant time to adopt— time that industry does not have. This research targets an alternative approach that is both inexpensive and faster-to-deploy: leveraging enormous amounts of existing process data that are archived on a plant computer. These data can be used to detect process anomalies based on recognized behaviors that have been identified over decades of operations. The aim of this approach is to enable operations to detect anomalies that are hidden in the process data, foresee catastrophic failures, and prepare for or mitigate them.

A specific scenario was targeted in this effort in collaboration with a United States NPP. On May 11, 2018, and May 26, 2018, two drywell fan-coil units (FCUs) at Cooper Nuclear Station failed unexpectedly, resulting in a 6-day plant outage. A lack of vibration sensors on the FCUs prevented the plant from predicting and preparing for this failure. Because installing new sensors was not an option for the plant, new means to detect anomalies associated with this failure mode were needed. Thirty-six process-instrument data points from the plant computer related to the failure were instead used to achieve this objective. This required studying minor deviations in the data correlation in time to determine whether a change has occurred due to an external factor (such as environment or load change) or an equipment or process anomaly because of a failure precursor. The operator brain cannot perceive such correlations beyond a few degrees of freedom, especially when tasked with many competing priorities. As a result, machine-learning methods were developed and applied. Two unsupervised learning methods—K-Means and isolated forests—were tested and yielded a large number of false-positive alarms. They were therefore excluded. Long short-term memory (LSTM) networks were used instead to introduce the

time and memory element. The LSTM method was customized with an anomaly-detection criterion that yielded satisfactory results.

The resulting method was able to predict the failure of FCU D 8 days before it occurred and, coincidentally, detected a sensor failure 2 days before its occurrence. The failure of FCU A occurred around two weeks after FCU D. This caused the FCU A failure to not be detected because the model has not been trained for a scenario that has only three fans running under similar plant conditions (i.e., this new behavior could not be predicted without more time for the machine to learn). In addition to directly using the method to monitor fans for the collaborating NPP, this pilot will feed into two directions of research: 1) to develop equipment-agonistic anomaly-detection methods 2) to create physics-based anomaly detection using a digital twin.

ACKNOWLEDGEMENTS

The authors would like to thank the Light Water Reactor Sustainability (LWRS) program for funding this effort and Cooper Nuclear Station for collaborating on this effort as part of cooperative research and development agreement 19-CR-15 to reduce the workforce cost at Cooper Nuclear Station using online monitoring and streamlined work processes. The author would like to also thank Brandon Rice for supporting the data acquisition process, and Shawn St. Germain and Jadallah Zouabe for their contributions.

CONTENTS

FIGURES

TABLES

ACRONYMS

ANN	artificial neural networks
BWR	boiling water reactor
CNS	Cooper Nuclear Station
CSV	comma-separated value
DP	Data point
FCU	fan-coil units
GPM	gallons per minute
GPU	graphical processing unit
HI	health index
HTM	hierarchical temporal memory
HVAC	heating, ventilation and air-conditioning
LSTM	long short-term memory
LWRS	Light Water Reactor Sustainability
ML	machine learning
MSE	mean squared error
NASA	National Aeronautics and Space Administration
NPP	nuclear power plant
PCA	principle components analysis
REC	reactor equipment cooling
ReLU	rectified liner unit
RMSE	root mean squared error
RNN	recurrent neural network
RUL	remaining useful life
SVM	support vector machine

SUBTLE PROCESS-ANOMALIES DETECTION USING MACHINE-LEARNING METHODS

1. INTRODUCTION

Operators at a nuclear power plant (NPP) are responsible for several tasks that can result in the operations organization becoming the most burdened organization in the plant. In addition to visually monitoring plant instruments in digital or analog form to keep track of plant conditions, operators are tasked with changing the plant configuration when needed, performing operator rounds, conducting and documenting surveillances, issuing clearance orders (including tagging in and out equipment), coordinating with other plant organizations, and performing random walks and checks of the plant. With this level of tasking, operators tend to adopt a reactive model to plant monitoring—i.e., responding to alarms and events—rather than a proactive model, under which small signs of anomalies can be detected, tracked, and mitigated before the issue escalates. Escalation of an anomaly to a functional failure can result in severe economic or safety consequences to the plant and demand a responsive effort with costs that often far exceed what would have been invested to mitigate the issue.

There are four main techniques to proactively detect anomalies in the plant:

- *Manual inspection.* This is the model currently used by most of the nuclear power industry (through preventive maintenance and surveillance). Temporary and localized measurement of a certain physical phenomena (such as vibration) is performed manually, and the data are analyzed to determine the health of an equipment.
- *Plant instrumentation with set-point based alarms.* This model is also used by much of the nuclear power industry through legacy plant computer and localized systems. The monitoring system alarms when the monitored parameter falls outside given set-points. This method is limited because it does not detect anomalous behavior within the set-point range and can give false positive indications for normal behavior that crosses a set-point value.
- *Automated monitoring.* Through retrofitting equipment that are not currently instrumented with permanent condition-monitoring sensors, data can be transferred to custom-developed methods that determine the condition of the equipment. This is the model to which the industry is moving. However, deploying this approach on a large scale is an investment that requires resources the industry lacks under current economic conditions, and would take significant time to adopt—time that industry does not have.
- *Process-anomaly detection.* This method is used to detect process data anomalies based on recognized behaviors that have been identified over decades of operations. The nuclear power industry has an enormous amount of process data that are archived in plant computers. These data can often reveal some condition anomalies, but are not utilized because analyzing data is challenging. Process-anomaly detection can be applied in parallel to manual inspection and automated monitoring, but is mostly valuable when the equipment monitored cannot be manually inspected or equipped with automated monitoring (this report targets an example of this scenario).

These four levels of defense are aimed towards a common objective, which is to reduce the level of unexpected equipment failure. This results in a lower probability of operations interruption and plant outage and fewer alarms and events (i.e., reduced operator workload in monitoring the plant).

Adopting a proactive model through process-anomaly detection requires studying minor deviations of many variables correlation in time to determine whether a change has occurred due to an external factor (such as environment or power generation) or due to an equipment or process anomaly caused by a failure precursor. The human brain cannot perceive variables' correlations beyond a few degrees of freedom; this is especially true for an operator, tasked with many competing priorities. High-dimensional feature spaces

are well beyond human capabilities to understand: the interactions occurring on a continual temporal basis and how they relate to ongoing equipment performance or upcoming failure events is complex. A machine, equipped with intelligent methods that can distinguish between an anomaly and normal plant behavior, is therefore a better candidate. Advanced machine-learning (ML) algorithms processing continual data streams can learn complex patterns and be configured to flag anomalies indicative of pending failure events in order to support an operator's decision process. However, extensive research is needed to develop these sophisticated models that can span supervised (i.e., informed by a training set of data that represent prior knowledge) and unsupervised (i.e. prior knowledge is not provided through the training data set) learning environments drawing from an ever-growing pool of algorithms and structural approaches. Several tools, designed by online-monitoring vendors, are offered for anomaly detection using pattern recognition. These tools, however, mainly provide libraries of methods to use and require the user to configure them for the specific problem being addressed.

The aim of this effort is research and develop data-analysis methods to detect anomalies based on data that is already present in an NPP plant computer. While a method agnostic to equipment— i.e., one method that can work for any equipment in the plant—is desirable, achieving this objective requires an incremental approach. The aim of this effort is to tackle a critical equipment failure as a step towards more generalized anomaly detection.

1.1 Problem Statement

A drywell is the containment structure enclosing the vessel and recirculation system of a boiling-water reactor (BWR, U.S. Nuclear Regulatory Commission 2019). Figure 1 shows where the drywell space lies relative to the reactor vessel. Because the reactor vessel is completely enclosed within the drywell, heat must be continuously removed from the drywell atmosphere by the drywell's ventilation system. The design temperature limit of the drywell is 135°F (General Electric 2011). Typical temperatures during operation from fall to spring are ~111°F whereas, in the summer months, the temperature ranges from ~125 to 130°F due to the higher temperature of river water. The plant intakes Missouri River water through a shell-and-tube heat exchanger. The river water provides cooling for the reactor-equipment cooling (REC) water. The REC water flows through shell-and-tube heat-exchanger coils, which have a 10°F approach (i.e., the difference between the water-inlet and the air-outlet temperatures). The REC-water flow rate is approximately 1,650–1,700 gallons per minute (GPM).

During normal plant operation, a closed loop of flowing nitrogen within the drywell provides cooling. Nitrogen flows through four fan coil units (FCUs)—each comprising a shell-and-tube heat exchanger and a centrifugal fan and located at the bottom of the drywell—surrounding the drywell. Each FCU consists of an electric motor driving a shaft directly coupled to a fan. The fan and motor units are connected to coils that pass heat from the air to recirculating fluid. The fan pulls hot air from the top of the drywell space above the nuclear reactor to the FCU. The REC loop provides cooling to all four FCUs. Hot nitrogen gas is cooled as it passes over the cooling-coil heat exchanger, and the fan directs the cool nitrogen upwards through a supply duct.

Figure 2 shows a schematic overview of FCU placement within the drywell environment from both the lateral and top-down perspectives. The FCUs are placed adjacent to FCU recirculation pumps (A and B) that control the fluid recirculation to the FCU coils. The reactor vessel is in the middle of the configuration. The drywell fans are identical except for the direction of rotation, but the enclosures are different given the custom designs for each of the four fan units. The fan units are designated FC-R-1A, B, C and D and are each allocated to one of four quadrants of the drywell (Figure 2).

On May 11 and May 26, 2018, two drywell FCUs at the Cooper Nuclear Station (CNS, see Figure 3) failed in a catastrophic manner, resulting in a plant outage. Specifically, Unit FC-R-1D failed on May 11, followed by Unit FC-R-1A on May 26, 2018 (see Figure 2 for location and Figure 4 for one of the failures). Both units failed when outboard fan bearings failed, damaging shafts and the mechanical infrastructure supporting the units. The bearings were installed in 2016, and failure occurred after

approximately 18 months of service. After the failure, the fan shafts were found to have broken at the fan side of the inboard bearings. The outboard bearings and support structures were severely damaged. The outboard end of the fan shaft had ground down through the bearing, lowering that end of the shaft. The fan wheels and inlet cones were somewhat damaged due to a lowering of the outboard end of the fan shaft. The failure of the second FCU resulted in an unscheduled plant shutdown lasting 6 days to facilitate repairs, resulting in a significant loss of revenue for the plant. This shutdown was necessary to protect the integrity of the drywell environment and stay in compliance with regulatory standards relative to drywell temperature limits.

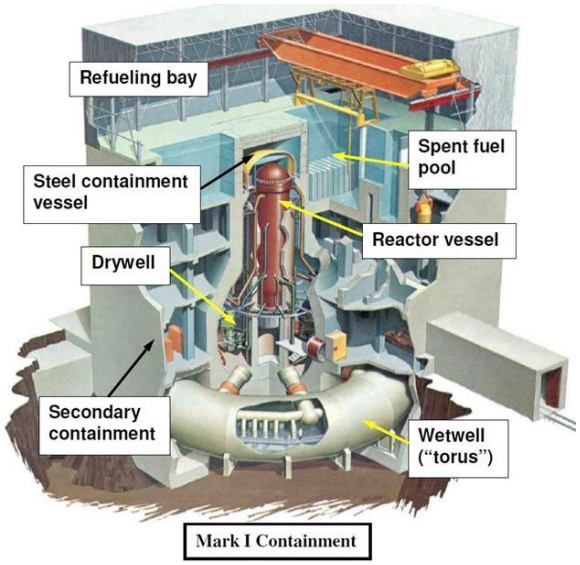
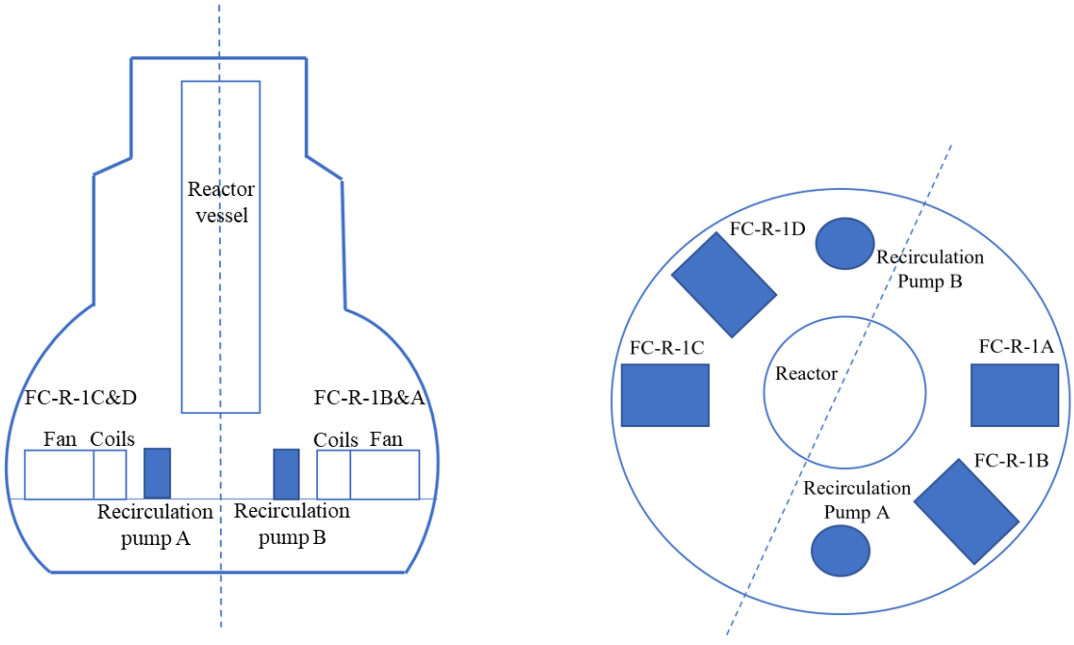
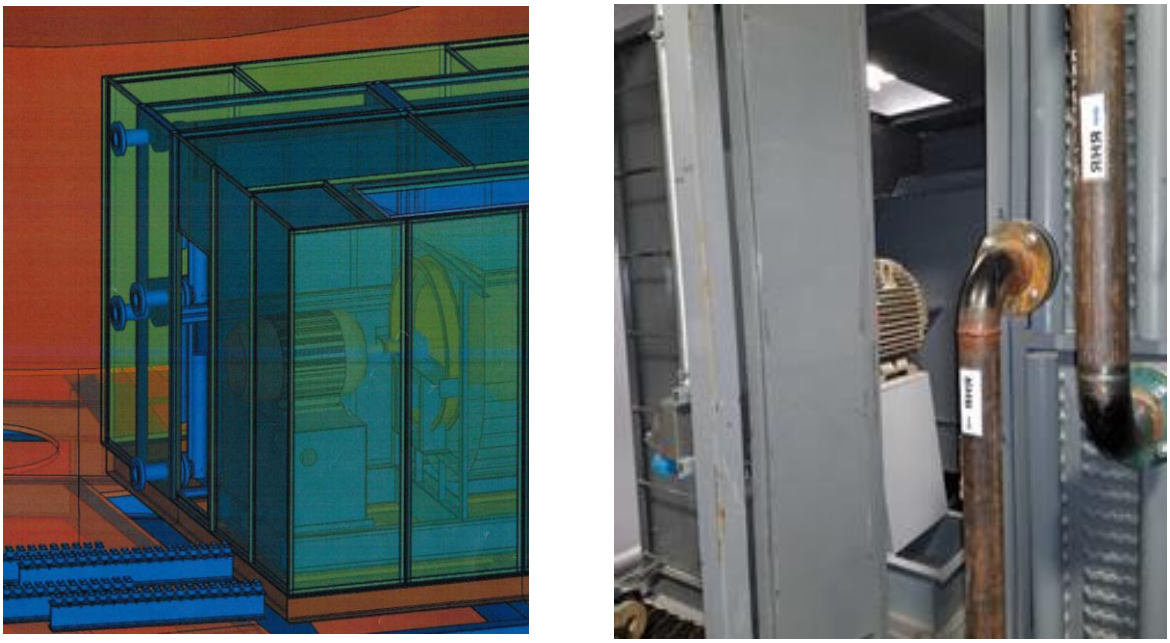


Figure 1. Configuration of drywell within a typical BWR, from Lochbaum (2013).



(a) Lateral perspective (b) Top-down perspective

Figure 2. FCU placement and arrangement of FCUs within CNS's drywell containment structure from lateral (left) and top-down (right) perspectives.



(a) Three-dimensional model of fan (b) Fan mockup

Figure 3. Drywell FCU at Cooper Nuclear Station.

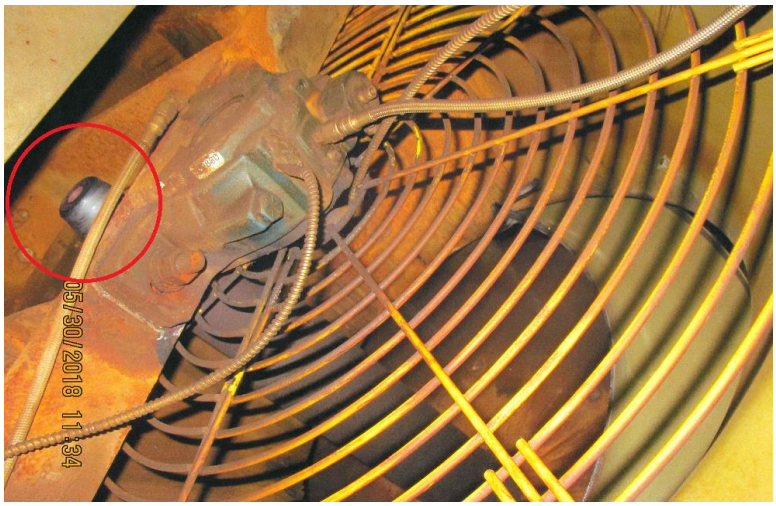


Figure 4. FCU failure showing the shaft displaced from the bearing.

The FCU system is not instrumented with sensors monitoring motor or bearing metrics, which could have been used for operators to track equipment health. New sensors cannot be installed without significant effort due to a lack of needed infrastructure and the complexity of an installation. This necessitates approaching this problem using process-anomaly detection. The CNS facility captures extensive sensor data from various plant environments and processes via the plant computer to monitor, in real-time, system and equipment performance. Sensor data points are archived at varying timescales, allowing for users to retrieve data via a software interface displaying time-series plots, data tables, and varying aggregate statistics. This effort identified thirty-six individual data points from the plant computer at CNS that relate to the drywell environment and, therefore, to the fan. These data are logged at roughly

a 1-minute temporal resolution, and include temperature, humidity, and fluid-flow rates. Each data point is assigned a unique data point identification (DP ID) in this effort for traceability.

1.2 Literature Review

Throughout multiple industries, traditional remaining-useful-life (RUL) metrics have been used to track equipment degradation guiding preventative maintenance and economic decisions. RUL is based on known equipment parameters, extensive operations testing, and known operating-environment constraints to track progress on degradation curves. However, faults occurring because of a combination of degradation and other factors or due to unknown defects can occur outside of known degradation curves. This makes them unreliable for fault prediction. Instead, researchers have turned to ML algorithms configured for unsupervised or supervised scenarios to identify anomalies that can be useful predictors for oncoming or already-occurring faults. This section will present some of the relevant work that was captured in literature.

Fault detection via associated data anomalies has been studied using various approaches. Farber et al. 2019 developed a method to detect small-break loss of coolant at a nuclear power plant using kernel density estimation. The method resulted in an average detection delay of one-seventh the time that a reactor usually takes to trip. A system for fault detection occurring within variable-air-volume air-conditioning systems based on system behavior rules was evaluated in Wang and Chen (2016). Specifically, they incorporated a residual-based exponentially weighted moving-average control-chart method in conjunction with twenty-six system behavior rules and validated this approach on air-conditioning systems data, incorporating artificial faults. Despite promising results, this approach requires extensive subject-matter expertise to develop rules, which limits its implementation to already well-known systems.

Other researchers utilized support vector machines (SVM) (Cortes and Vapnik, 1995) model to detect anomalies. In specific, an unsupervised single-class SVM was used to model anomalies in heating, ventilation, and air conditioning (HVAC) systems (Beghi et al. 2014). The lack of labelled HVAC data streams often prohibits supervised approaches. A single-class SVM learns to identify objects belonging to a class defined by the features of a single-class training data set. In this case, it was used to develop a "reference" model by training on data derived from a unit operating under normal conditions with no faults. The model was applied to an HVAC data set with known anomalies, and it returned good classification results. Although single-class SVMs are ideally suited to work with extremely unbalanced data sets like those studied for fault detection, it is only suitable as a grouping mechanism and may become less effective as variance or noise within training data expands the grouping boundaries. This could make the model less effective at detecting faults. This issue can also be aggravated by "concept drift," a phenomenon that occurs in time-series data exhibiting long-term changes or temporal dependence.

Fault detection based on signal-feature extraction and a decision tree was evaluated for vibration data from roller bearings on low-speed equipment (Song et al. 2018). Feature extraction was performed by combining a statistic filter, wavelet-package transform, and a moving-peak-hold method. Resultant features were passed to a decision tree for fault diagnosis. A method such as this requires directly coupled sensors and tuning signal-feature extraction using extensive training data for both normal and abnormal states.

ML algorithm development supported by engineering subject-matter expertise was used to detect faults in a commercial-building chiller unit, achieving an 80% accuracy during testing (Hu et al. 2019). The approach outlined specific questions asked of the experts to guide model development and reduce features. SVMs were trained on well-curated and labeled data. The analysis shows it is possible to significantly reduce the number of features while maintaining model performance metrics. But the approach requires extensive access to experts with deep domain expertise.

A framework was proposed and tested where anomalous data in HVAC temperature- and electricity-sensor data were identified and then pruned using contextual information to reduce false positives (Hayes and Capretz 2015). A univariate Gaussian predictor model was used to build a historical model of the data; this was used to predict and compare new content anomalies. Because the content detection would likely flag false positives as well, a contextual anomaly detector was used to prune the final output. Contextual anomaly detection was based on assigning incoming content anomalies to defined sensor-group profiles derived from K-means (MacQueen 1967) clustering of sensor information (i.e., location, date, time, climate, etc.) to make a final classification determination based on the distance of the content value from the group average. Although the research shows the importance of contextual information, in some scenarios, context may not always be available.

Isolation forests were developed to explicitly isolate anomalies in numerical data sets using random subsampling versus profiling normal instances, thereby improving algorithm efficiency and reducing memory requirements (Liu et al. 2008). When compared to other ML algorithms, including neural networks and random forests, the isolation forest algorithm performed well in benchmark tests and excelled for large data sets in use of memory and computation time (Carrasquilla 2010). However, the algorithm depends upon two assumptions: that anomalies make up a minority of the instances of a data set and have attribute values that are very different from those of normal instances. From the perspective of detecting subtle anomalies preceding faults, it is not known whether an approach using an isolation forest would be suitable to flag such instances without incurring excessive false positives.

K-means clustering was applied to multiple telemetry data streams typically monitored by human operators to track equipment health (Azevedo et al. 2012). In most cases, the algorithm was effective in identifying anomalous instances preceding known equipment faults. However, in cases of subtle anomalies, where variation in the data was small, anomaly-classification accuracy was reduced. K-means clustering can be impacted by background variance or "noise" in the data, which can reduce sensitivity to smaller anomalies preceding equipment faults or failures. Also, it is more challenging to implement as an online learner given the model carries with it the distance matrix for all features on which it was trained. This matrix continually increases with training. In some cases, data dimensionality-reduction strategies, such as principle-components analysis (PCA) (reviewed in Jolliffe & Cadima, 2016) can be leveraged to reduce noise and improve model sensitivity.

A sequence-learning framework called hierarchical temporal memory (HTM) can be implemented to process data streams in real time and to incorporate continual learning to account for concept drift in a way that can detect anomalies. This method was evaluated using the Numenta anomaly benchmark data in Ahmad et al. (2017). The results indicated that the errors of HTM are not always correlated.

Long short-term memories (LSTMs) (Hochreiter and Schmidhuber, 1997) were used to predict fault occurrences and track RUL based on performance metrics of simulated aircraft-turbofan data sets provided by the National Aeronautics and Space Administration (NASA, Yuan et al. 2016). Their analysis showed LSTMs outperformed the standard recurrent neural network (RNN) architecture. Although their results are encouraging, the researchers had the benefit of the simulated and well-curated data necessary for supervised models. This included multiple fault events on multiple engine components and given output responses of the measured engine variables or features. This level of labelling and curation is not always available when working with complex, multivariate time-series data derived from equipment or industrial facilities.

Approaches have been developed allowing for the inclusion of LSTMs when labeled anomalies or faults are not available or are very limited. This is done by training an LSTM to reconstruct time-series data of a "healthy" system by using input features and labels associated with normal activity. After training, any time the model is tasked to reconstruct time-series data based on features with anomalies or faults, the reconstruction error or difference from the "healthy" baseline increases, indicating that the model is seeing something different than the healthy baseline. Researchers demonstrated this by

developing a health index (HI) calculated by an LSTM encoder-decoder (LSTM-ED) (Malhotra et al. 2016). Because the LSTM-ED trained only on the healthy baseline system, error would occur in its predictions when anomalies occurred impacting the input features during predictions. The reconstruction error was then used to adjust the RUL estimation. This framework was developed and validated using publicly available turbofan-engine and milling-machine data sets and was successfully tested on data from a pulverizer mill where the HI showed high correlation with maintenance costs. This approach is similar to earlier work performed by Petsche et al., (1996) who used neural-network-based auto associators (encoders) trained to reconstruct electrical-motor data derived from healthy systems. This method requires extensive historical data prior to implementation.

2. DATA ACQUISITION AND PREPARATION

The plant computer of CNS continually logs data from various sensors, including the REC inlet- and outlet-water temperatures and the air bulk temperature. However, the individual flow rates through each of the four REC heat exchangers are not tracked. Individual FCU inlet and outlet temperatures are also tracked, along with FCU dew points. Table 1 shows the complete list of plant computer data points relative to the CNS drywell environment. The DP ID column shows the unique ID associated with the data stream in the plant computer. The Description column provides the descriptive text associated with the DP. The Units column shows the sensor measurement units and the Environment column shows the specific FCU associated with the sensor. Each FCU has dedicated inlet, outlet, and dew-point sensors. The remaining sensors are placed in varying locations throughout the drywell environment and are associated with all FCUs. Figure 5 shows a schematic diagram of sensor locations and associated DP IDs. To have enough data with which to develop exploratory models, the data span from January 1, 2017, through May 12, 2019. This provides data before and after the known FCU failure events.

2.1 Data Challenges

The challenges associated with the data collected from the plant computer can be summarized:

- No vibration data related to the fans were found to validate the method performance. Vibration is a key element in identifying rotary equipment failure (Al Rashdan et al. 2018). This, however, was the main motive behind using a process-anomaly detection approach because the overarching goal of this analysis is to show CNS-drywell environmental-data streams can be used to detect anomalies preceding equipment fault or failure events.
- The specific FCU data are limited to FCU-inlet and outlet temperatures and inlet dew point. The rest of the measurements (heat-discharge rate, power, and drywell dew point and temperature) in Table 1 are common among all FCUs. This reduces the unique dimensionality of the individual FCU—i.e., it makes it harder to determine which of the FCUs is experiencing the anomaly.
- Each data stream was logged at an approximate 1-minute resolution with date-time instances logged at either a second or decimal-second precision. At this resolution for thirty-six separate data points, tens of million records or data instances were available for a given year. Because multiple years of data were available for the analysis, the number of instances increased to several tens of millions, which was computationally demanding.

Table 1. CNS drywell plant computer DP IDs and associated data.

DP ID	Description	Units	Environment
DP 1	Primary Containment Heat Exchanged	KBTU/H	ALL
DP 2	Plant Power Output	%	ALL
DP 3	Reactor Core Flow	MLB/H	ALL
DP 4	FC-R-1A Inlet Air Temperature	°F, Dry bulb	A
DP 5	FC-R-1B Inlet Air Temperature	°F, Dry bulb	B
DP 6	FC-R-1C Inlet Air Temperature	°F, Dry bulb	C
DP 7	FC-R-1D Inlet Air Temperature	°F, Dry bulb	D
DP 8	Recirculation Pump A Area Temperature	°F, Dry bulb	ALL
DP 9	Recirculation Pump B Area Temperature	°F, Dry bulb	ALL
DP 10	FC-R-1A Outlet Air Temperature	°F, Dry bulb	A
DP 11	FC-R-1B Outlet Air Temperature	°F, Dry bulb	B
DP 12	FC-R-1C Outlet Air Temperature	°F, Dry bulb	C
DP 13	FC-R-1D Outlet Air Temperature	°F, Dry bulb	D
DP 14	FC-R-1A Inlet Dew Point	°F, Dew Point	A
DP 15	FC-R-1A Inlet Dew Point	°F, Dew Point	B
DP 16	FC-R-1A Inlet Dew Point	°F, Dew Point	C
DP 17	FC-R-1A Inlet Dew Point	°F, Dew Point	D
DP 18	Drywell Inlet Supply Temperature	°F	ALL
DP 19	Drywell Outlet Temperature	°F	ALL
DP 20	Flow Rate to Drywell	GPM	ALL
DP 21	Average Bulk Drywell Temperature	°F	ALL
DP 22	Average Bulk Drywell Temperature (20 Data Points)	°F	ALL
DP 23	Average Bulk Drywell Temperature (5 Data Points)	°F	ALL
DP 24	Return Air Ring Temperature	°F	ALL
DP 25	Return Air Ring Temperature	°F	ALL
DP 26	Return Air Ring Temperature	°F	ALL
DP 27	Zone 2B Temperature	°F	ALL
DP 28	Zone 2B Temperature	°F	ALL
DP 29	Zone 2B Temperature	°F	ALL

Continued Table 1 CNS drywell plant computer DP IDs and associated data

DP 30	Zone 2B Temperature	°F	ALL
DP 31	Zone 2B Temperature	°F	ALL
DP 32	Zone 2C Temperature	°F	ALL
DP 33	Zone 2C Temperature	°F	ALL
DP 34	Zone 2C Temperature	°F	ALL
DP 35	Zone 2C Temperature	°F	ALL
DP 36	Zone 2C Temperature	°F	ALL

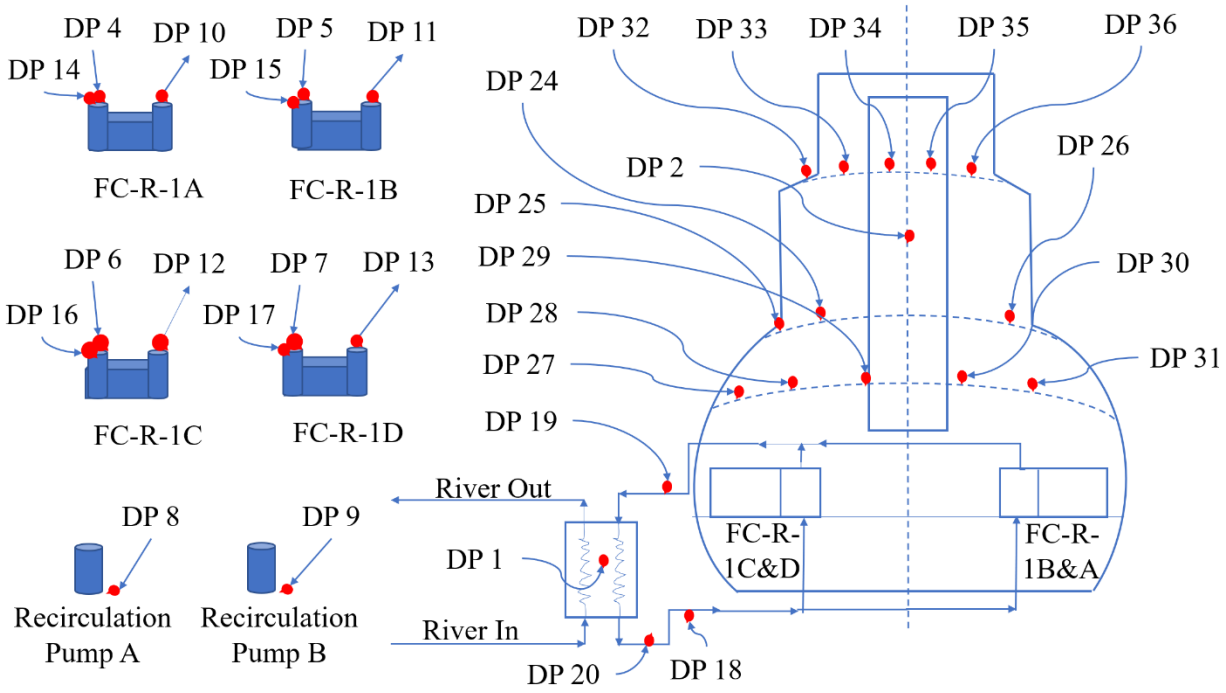


Figure 5. Location of selected sensors shown in Table 1.

- Inspection of the data revealed occurrences of missing or "0" values and potential sensor or logging problems resulting in repeating data points that sometimes spanned months. An example of this is shown in Figure 6. The figure's Y axis shows CNS plant power output (DP 2) and FCU B input (DP 5) and output (DP 11) temperatures from data downloaded from the plant computer for 2018. Starting in early May, DP 11 likely suffered a sensor malfunction as it continually logged the exact same value until late September of 2018.
- Another significant challenge is the sparse failures, which are required for training the ML approaches. Out of tens of millions of instances, only two mechanical-failure events are known along with two additional sensor failures.

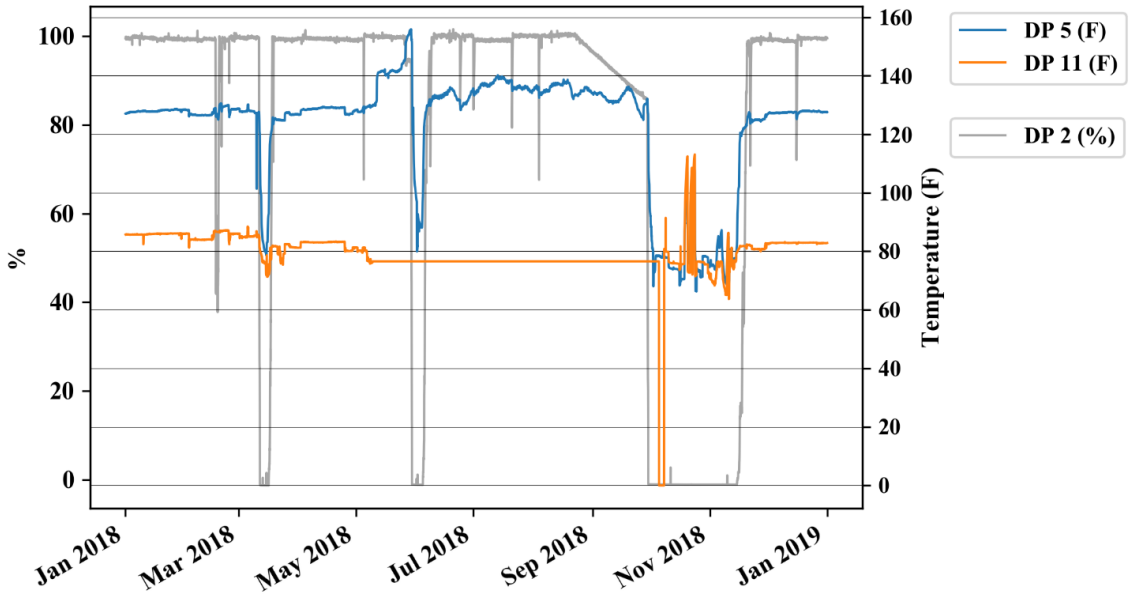


Figure 6. Data logged from three data points from Table 1 over one year of operations.

2.2 Data Preparation

Because of the size and complexity of the data used for this analysis, the data sets were prepared using Python 3.6 (Python development team, 2019), which incorporates multiple libraries suited for preprocessing and modelling data. Common data-handling software (such as Microsoft Excel) are not capable of handling or aggregating tables and supporting the complex-feature engineering tasks necessary for this effort. The large amount of data also challenged computer-hardware resources given the increased requirements for computer memory needed to load the data sets. The data-preparation process in Python followed the following chronological process:

1. Time formatting must be homogenized. Inspection of data showed some data files' date-time instances contained a mixture of date-time precision (second and decimal-second precisions) which required correction prior to time-series analysis.
2. Although each data stream is logged at a 1-minute resolution, date-time instances are not logged at the exact same moment. Temporal alignment was performed. For example, DP 20 shows an instance was logged on December 1, 2016 at 00:00:02. For that same date, hour, and minute, DP 1 shows a logging time of 00:00:00 (see Figure 7). Data sets were prepared with time granularity in minutes, hours, days, and weeks. This decision was made to find different abstract patterns in the data as well as to decrease model computation time during the learning process. The time-series data resampling performed was based on the arithmetic mean, and each data set was saved in an individual file for faster processing.[a]

a The files were serialized using Python's "pickling" algorithm for object serialization.

	Datetime	DP 3	DP 20	DP 2	DP 1	DP 14	DP 15	DP 16	DP 17	DP 18	...
0	2016-12-01 00:00:00	NaN	NaN	NaN	5.6099	NaN	NaN	NaN	NaN	NaN	...
1	2016-12-01 00:00:02	NaN	1710.9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
2	2016-12-01 00:00:03	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
3	2016-12-01 00:00:04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
4	2016-12-01 00:00:05	NaN	NaN	NaN	NaN	NaN	77.0	NaN	NaN	NaN	...
5	2016-12-01 00:00:06	NaN	NaN	NaN	NaN	NaN	NaN	75.0	NaN	NaN	...
6	2016-12-01 00:00:08	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
7	2016-12-01 00:00:09	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
8	2016-12-01 00:00:17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
9	2016-12-01 00:00:21	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...

Figure 7. A screenshot of drywell-sensor data, arranged in columns with time steps as rows, for Minute 0 of December 1, 2016.

3. As indicated earlier, multiple temperature-sensor failures recorded "flat line" values or zero values as the time-series progressed. Furthermore, some sensors sporadically recorded a temperature below 60°F; these readings were advised to be incorrect and needed to be removed. Given the data had multiple occurrences of logging issues or sensor failures, data where all sensors appeared to be functioning appropriately were identified, labeled, and flagged. This represented a significant section of data prior to the known mechanical-failure events (FCU A and D) and a sensor failure (FCU B) that occurred just prior to the mechanical failures. Figure 8 shows the flat-line value segments and the zero value segments. Three main data interruptions occurred due to:

- FCU A inlet-temperature sensor (DP 4) failure, recorded a flat line twice consecutively, followed by zero values beginning December 1, 2016.
- FCU B outlet-temperature sensor (DP 11) failure, recorded a flat line beginning May 9, 2018.
- FCU D inlet-temperature sensor (DP 7) failure, recorded a flat line and then zero values beginning September 9, 2017.

A custom defined function[b] was created to handle the curation process and remove inconsistent data from the data.

3. METHODS EVALUATED

After consideration of the literature review described in Section 1.2, three approaches were selected for evaluation. The first is a clustering unsupervised machine approach using the K-Means method. The second uses another unsupervised machine approach: an isolation forest that is customized for anomalies detection. Last, LSTM was used to incorporate the time element into consideration. The following sections describe the implementation and finding of each method.

3.1 K-Means

3.1.1 Method Description

K-Means is an unsupervised ML method for cluster analysis in data mining (MacQueen 1967). It is one of multiple commonly used methods for clustering, such as density-based spatial clustering of applications with noise (Ester et al. 1996), ordering points to identify the clustering structure (Ankerst et al. 1999), and agglomerative clustering, a subset of hierarchical clustering (Ward 1963).

b Using Jupyter Notebook.

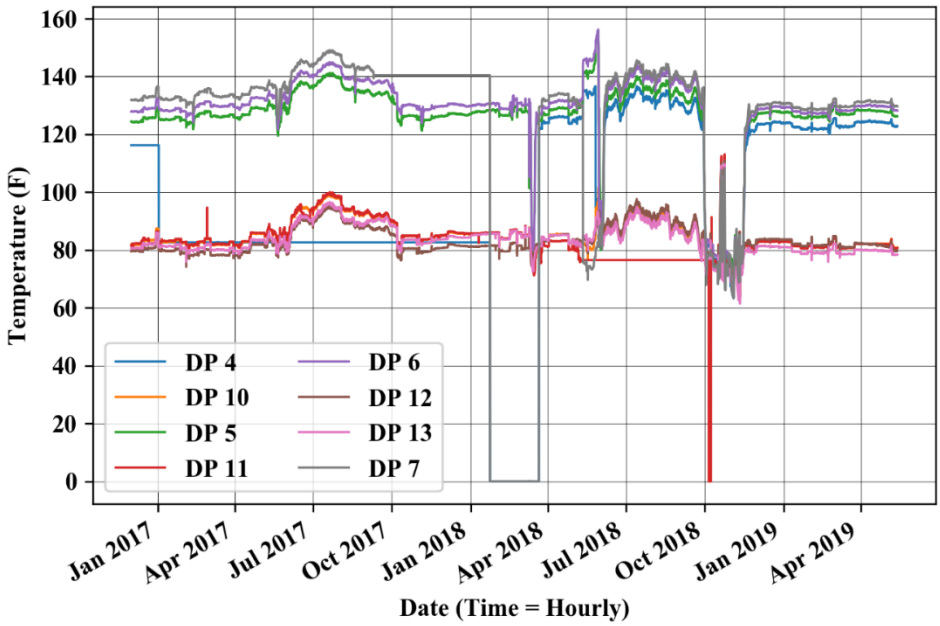


Figure 8. Graph showing inlet and outlet temperature recorded by sensors for FCUs A, B, C, and D.

K-Means is most often used with data sets that have no predefined categories or groups. The goal of the method is to make *k* distinct clusters from *n* total observations. It works by assigning each data point in the data set to one of the *k* groups based on all the features available. Each one of the *k* clusters is defined by its centroid. The centroids are the mathematically computed arithmetic mean of the points within the individual clusters (geometric center). The K-means algorithm operates by iterating between two main steps. At the initiation of the process, each cluster gets a random computed centroid or a randomly selected one from the original data set. The first step is to assign data points to the nearest centroid. The second step is to recalculate the centroid with all the data points assigned to the cluster. These two steps are done until a stopping condition or criterion is met. Common stopping conditions include reaching a set limit for the amount iterations, summing until the distances have been minimized, or reaching a point where no data points have changed cluster assignment. In summary, the K-Means pseudocode proceeds through these tasks:

- Choose the number of *k* clusters
- Obtain the data points from the data set
- Place centroids c_1, c_2, …, c_k randomly
- Repeat the following steps iteratively until the stopping condition is met
- For each data point x_i, find the nearest centroid (c_1, c_2, …, c_k)
- Assign the data point to that cluster
- For each cluster c_1, c_2, …, c_k, find a new centroid = mean of all points currently assigned to that cluster.
- Repeat until stop condition is met.

The K-Mean method is computationally demanding and falls under the category of non-deterministic polynomial-time (NP)-hard problems. Although K-Means falls under such category, an efficient heuristic algorithm can converge on a solution rather quickly and find a local optimum. In most cases the number of clusters, *k*, is not known a priori and must be found through empirical observations. For this task a

technique called the "elbow method" was used for validation. The elbow method is heuristic and designed to help find the appropriate number of clusters in a data set. The analysis begins by determining the minimum and maximum number of clusters for the algorithm. The minimum number of clusters is trivially chosen and set to one ($k = 1$). The maximum number of clusters is the total number of features present in the data set ($k = 26$, which is the number of variables in Table 1 after removing the data points that are specific to other FCUs). After the domain of clusters has been chosen, the analysis proceeds by calculating a score value for each number of k clusters. The score is implemented[c] where the objective function is to minimize the sum of squares of the distances of all data points in their respective clusters. All the scores are then graphed in a line plot and analyzed for the elbow criterion. The optimal number of k clusters is where the graph shows minimal gain by increasing the number of clusters, the elbow in the graph. In summary, the K-Means cluster analysis pseudocode:

- Determines the total number of features n in data set
- For each cluster $j = 1 \ldots n$, calculates a score for K-Means cluster using $k=j$
- Graphs the number of clusters and its respective score
- Looks for elbow criteria on the line-graph, optimum value sets k to optimum value.

The K-Means algorithm can be used as an anomaly detector by analyzing the distance of each data point from its centroid. Closely related points are concentrated around the centroid and are considered normal. Data points that fall far away from the centroid are still considered to be related to their neighboring points, but with a lower affinity; thus, each can be classified as anomalous. The anomaly detector could work by using the distance as a notion of relationship and apply it as a discriminator between normal and abnormal. A threshold on the distance can then be set to determine whether a data point is normal or an anomaly. An example is shown of two cluster in Figure 9. The farthest point in this example is considered anomalous.

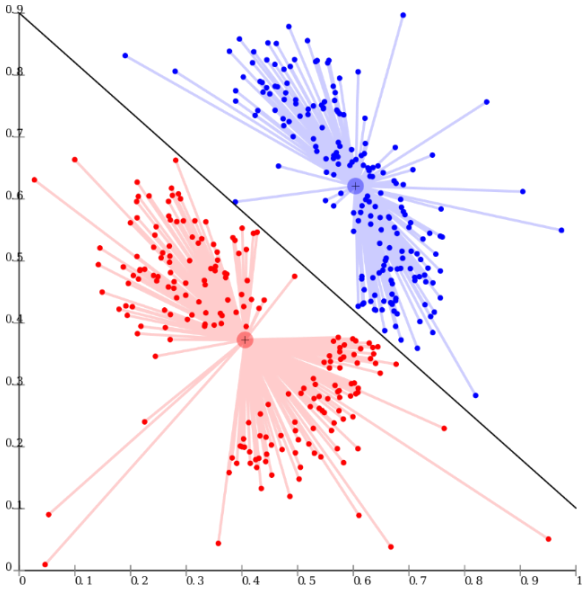


Figure 9. Sample of two-dimensional data set to visualize the distance from points to centroid (Wikimedia Commons contributors 2016).

Two approaches were implemented to evaluate the K-Means method as an anomaly detector using the distance metric. The first was to sort all members of each cluster in descending order based on the

[c] Implemented in the Scikit-learn Python library (Pedregosa et al., 2011)

distance. Thresholds (*p%*) of 1, 2, 3, and 5% were applied to select the top *p*% from each cluster and classify them as anomalies. The second approach was also to sort all members of each cluster in descending order again, but the threshold would be applied globally. In this version, the same values were used, and the top *p*% were selected across all clusters. The approaches pseudocode is:

- Run K-Means algorithm to find *k* clusters
- Set threshold space $t = (0.01, 0.02, 0.03, 0.05)$
- For $j = 1 \ldots k$, sort members of cluster number *j* in descending order (Approach 1) or sort members of cluster number *j* in descending order (Approach 2)
- For each threshold *t*, select top *t* members
- Classify them as anomalous

3.1.2 Tools

To execute the K-Means algorithm, an environment was created within the Anaconda distribution for the Python programming language. Anaconda is entirely open source and aims to simplify package management and deployment through the conda system. It provides Python and R software products that include commonly used data science and ML libraries for desktop computing systems. The environment for this analysis was setup with Python 3.6.8 and Jupyter Notebook (the code-editing tool), Version 4.4.0. Multiple libraries were installed using the conda system to facilitate data manipulation. Numpy and Pandas were mainly used to store and manipulate the time-series data sets as both incorporate unique functionalities supporting data engineering. Scikit-learn was used to leverage the ML algorithm and the internal preprocessing ability. Matplotlib was used extensively to visualize the analysis and show the results of the anomaly detector.

3.1.3 Results

The goal was for the anomaly detector to catch mechanical failures of FCUs A and D before the actual breakdown and with a low misclassification for the other data points. A search space for the threshold was created that contained the top 1, 2, 3, and 5%, as described in the previous section. A cluster analysis (shown in Figure 10) revealed that a high number of clusters is needed. However, to keep the number of clusters at minimum due to computational constraints, five clusters were selected. Except for the 5% threshold, none of the results (from 1, 2, and 3%) detected any anomaly. Figure 11 shows the resultant clusters prior to the known failure events for FCU D using 5% as the outlier threshold. The failure of FCU D was experienced on May 11, 2018, coinciding with the last point in the time series.

The 5% threshold could capture the mechanical failure one day ahead of its occurrence. However, it contained several false positives (shown in red in Figure 11). A reduction technique was applied using principal-component analysis (PCA). PCA is another unsupervised learning technique, and it is used to reduce the dimensionality of the data set while maintaining the loss of information at a minimum (Wold et al. 1987). The approach was used to reduce the number of dimensions in the data set and use fewer clusters in the K-Means method. For optimization purposes, a component analysis was done using 1 to *n*-1 components, where *n* was the number of dimensions in the data. The result of the cluster analysis with PCA showed that the first 5 components create an elbow and explain 99% of the variance in the data set (Figure 12). The K-Means application after PCA provided better results in terms of fewer false positives (because of the 2% stricter threshold in this case), but it did not remove all false positives (Figure 13). It is important to note that the false positives were not investigated as a potential actual anomaly.

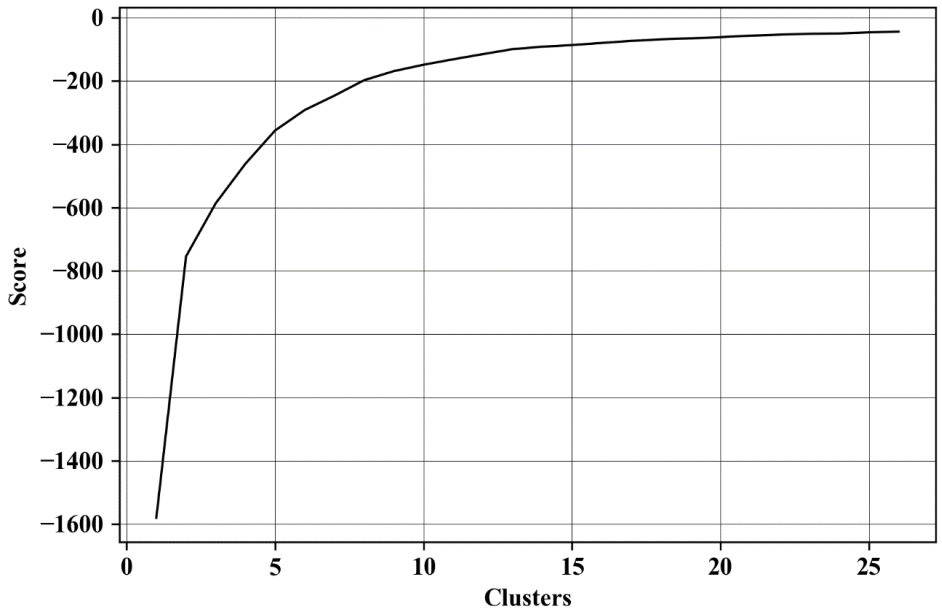


Figure 10. Cluster analysis for the data set with the elbow forming at around ten clusters

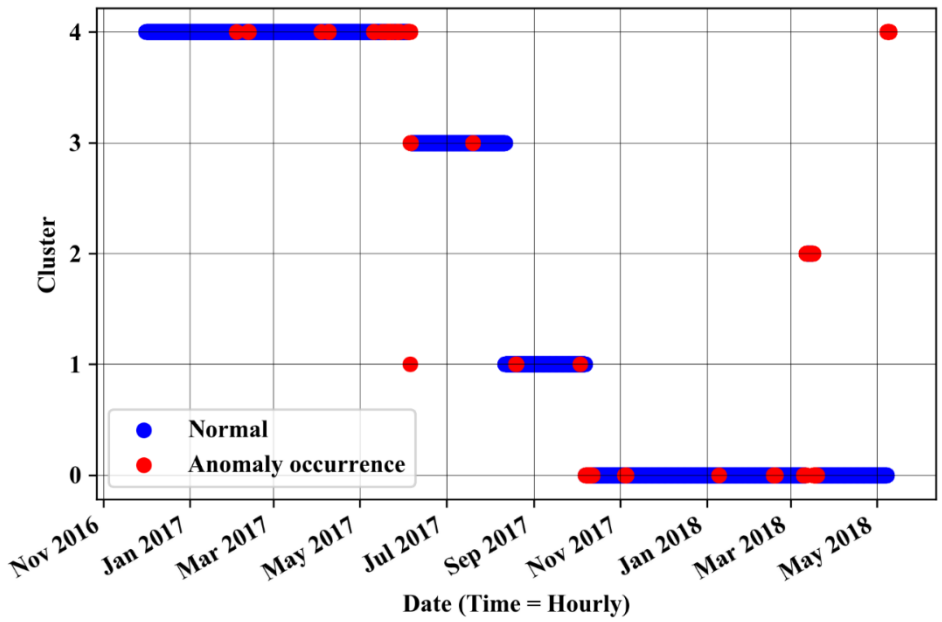


Figure 11. K-means algorithm using five clusters with an outlier fraction of 5% for FCU D, from December 1, 2016, to May 11, 2018.

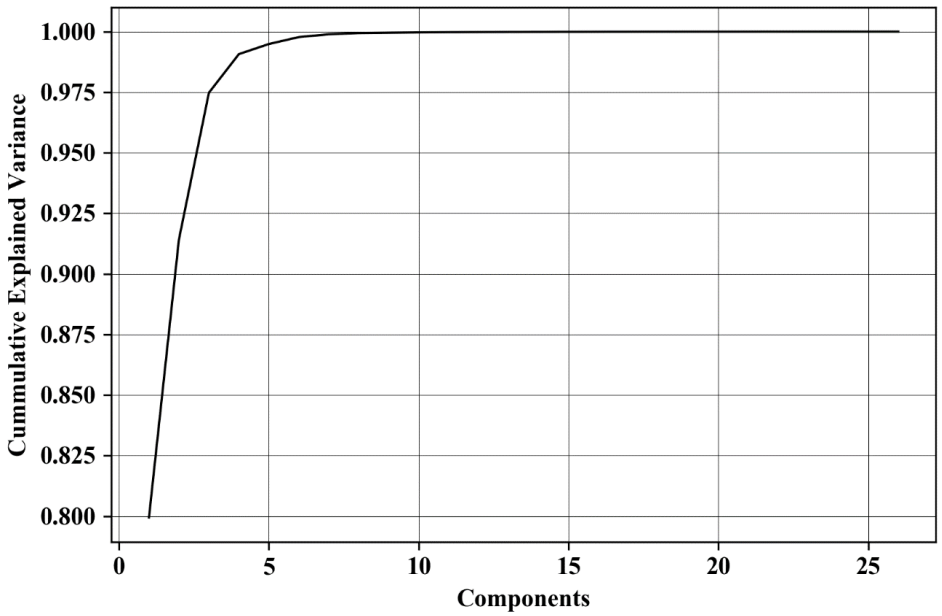


Figure 12. Cluster analysis for the data set with the elbow forming at around five clusters.

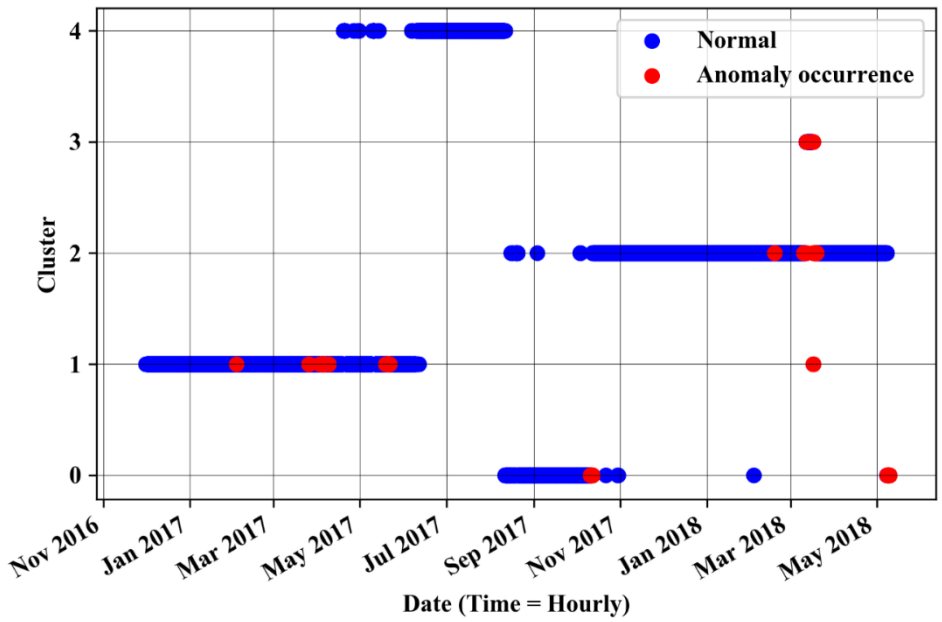


Figure 13. K-means algorithm with PCA using five clusters and an outlier fraction of 2%, for FCU D, from December 1, 2016, to May 11, 2018.

3.2 Isolation Forest

Isolation forest is another unsupervised ML algorithm that is built based on decisions trees. The main idea, which is very different in representation from many other popular outlier and anomaly-detection algorithms, is that an isolation forest explicitly focuses on identifying anomalies instead of characterizing normal data points in the data set. Isolating anomalies is easier as fewer conditions are needed to separate them from the rest of the data set. In contrast, normal observations require more conditions to isolate them.

3.2.1 Method Description

The isolation forest algorithm isolates observations by randomly selecting a feature from the feature space and then randomly selecting a split point from the maximum and minimum values of the selected feature. In principle, outliers have much-less-frequent occurrence than regular observations in the data and have subtle differences in terms of values. Because recursive partitioning can be natively represented by a tree structure, the number of splits required to isolate an observation is equivalent to the length of the path from the root node of the tree to the leaf on which the observation lies. The path length, when averaged over all random trees in the forest, is a measure of anomaly in the observation. For example, Figure 14 shows that, when the point is in the cluster (Figure 14 left), it requires significant portioning to identify while, when it lies outside the cluster (Figure 14 right, showing an anomaly), few partitions are sufficient to isolate it.

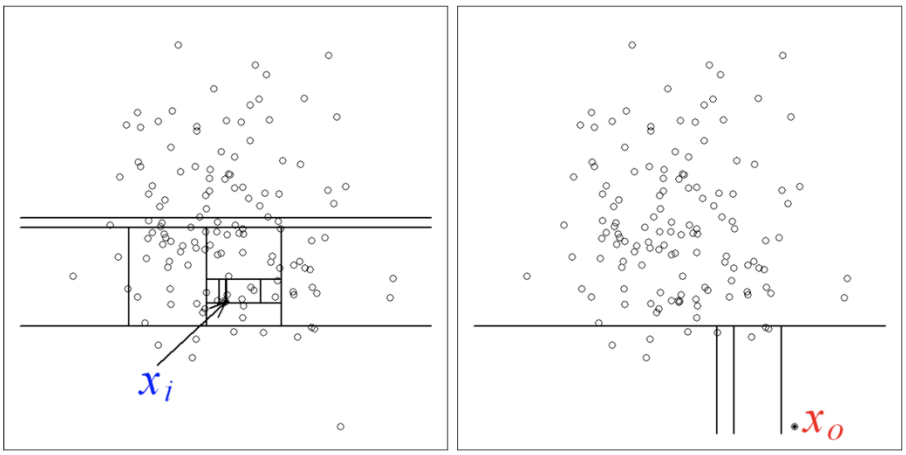


Figure 14. A visualization showing how many partitions (lines) are required to find a point when it's a normal point (left) and an anomaly (right) (from Lewinson 2018).

Figure 15 shows a visualization of a single isolation forest tree defined by partitions resulting from the random selection of features and random split values. In this case, the red line extends from the root of the tree (top) and has the shortest path, indicating it is an anomaly. The blue path is longer and indicates a normal value. When all the trees of an isolation forest are combined into one averaged tree, the classification of anomaly considers all trees in the forest, instead of one (Figure 16).

The approach taken and implemented was to use the path-length metric in the isolation forest algorithm and apply it over a search space for multiple time frames and contamination rates. The contamination rate is a parameter that describes the proportion of outliers (similar to thresholds in K-means in the data set).[d] It is used when fitting the data to the model to define the threshold on the decision function. The contamination-rate search space was composed of the values 0.01, 0.05, and 0.1. The isolation forest pseudocode is:

1. Select a point from the data set to isolate.
2. For each feature f_i, (representing one of the twenty-six data points) set the minimum in the range and the maximum in the range.

d Passed to the Scikit-learn ML module in Python.

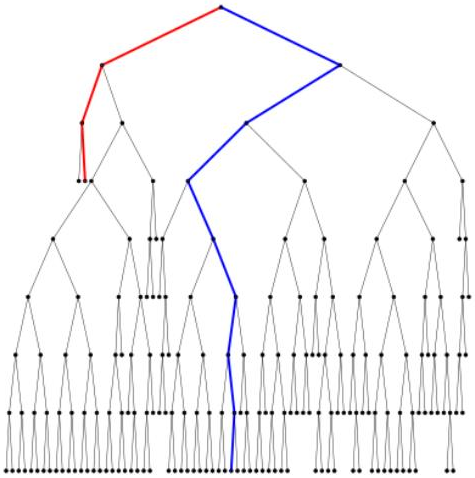


Figure 15. A single tree representation of the isolation forest (from Hariri et al. 2018).

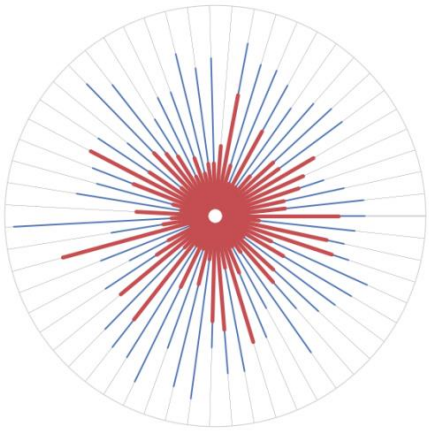


Figure 16. Visual representation of a full forest in which each radial line represents the path length of a single tree. Red represents anomalies while blue represents normal values (from Hariri et al. 2018).

3. Choose a feature at random.
4. Pick a value within the range at random.
5. Use the value as a partition (path in tree).
6. Evaluate the stop condition (e.g., how many points above or below the partition to find if the point is the only one within the range for all features).
7. Repeat steps 3 to 6 until the stop condition is met.
8. Count how many times steps 3 to 6 (the loop) were done. This is the path length (p_{length}).
9. Repeat steps 2 to 6 for each tree t_i in the forest. Each tree will be started from scratch and end up with a different tree structure because of the randomness of points.
10. Sum p_{length} for all t_i and get the average a_{pl}.
11. Find the anomaly based on the smallest $a_{\text{pl.}}$ and the contamination rate.

PCA was also applied to the data, and an additional post-PCA analysis was conducted to evaluate the effects of dimensionality reduction on the algorithm. These values were selected specifically to fine tune the model and to calibrate the sensitivity to anomalous observations. The Implementation pseudocode is:

- Use different time granularity t = (minutes, hours, days, weeks)
- Resample original data set using arithmetic mean
- Set contamination rate r = (0.01, 0.05, 0.1)
- For each $t_{i,}$ create isolation forest:
  - For each contamination rate $r_{i,}$ select data points average $p_{length} < r_i$
  - Classify data points with $p_{length} < r_i$ as anomalous.

3.2.2 Tools

The Anaconda environment used for the K-Means algorithm was reused to execute the isolation forest implementation. One additional library within the Scikit-learn package, IsolationForest, was imported to handle the creation and logic for the algorithm. This library handled the creation of all random trees in the forest and returned the binary (-1 or 1) classification for the data set.[e] Matplotlib was used to visualize the mapped results as a color-coded scatter plot.

3.2.3 Results

The results are shown in Figure 17 for the pure isolation forest approach and Figure 18 for the preprocessing using PCA. The first approach in Figure 17 failed to catch the actual mechanical failure (shown in red) regardless of the contamination rate (0.05 was the highest and most relaxed). The second approach, shown in Figure 18, was able to catch and label the mechanical failure for FCU D as anomalous, but resulted in an increase in misclassification rate. Unfortunately, neither method (i.e., with or without PCA) was able to detect the failure without an enormous misclassification rate. While it was assumed that both methods identified various anomalies that did not relate to known equipment-failure events, this was not verified, and these anomalies could have been associated with actual plant events.

3.3 Long Short-Term Memory Recurrent Neural Networks

Artificial neural networks (ANNs), specifically RNNs are a supervised learning model capable of carrying forward temporal information or learning over time via feedback loops, which can account for concept drift. An RNN is a class of ANNs where the connections between nodes form a directed graph (digraph). The digraph is made up of a set of vertices which are connected by edges. All the edges have a start and end vertex, as well as a direction associated with them. In the case of the RNN, the digraph is formed along a temporal sequence which allows the network to exhibit a dynamic temporal behavior. RNNs are exceptionally well suited to process sequences of inputs due to their inherent internal state (memory). This makes them applicable to tasks dealing with time-series data set such as the one for this project.

LSTM networks are a modern variant of RNNs and were first proposed in 1997 (Hochreiter and Schmidhuber 1997). LSTMs are better suited to handle long-term temporal dependencies because the architecture includes a memory cell to maintain temporal information (i.e., LSTM extends the memory of a regular RNN). The short-term memory in the LSTM is exhibited through persistent previous information that is used in the current neural network. It also mitigates the vanishing-gradient problem, which is the network's inability to learn because the updates to the weights within the nodes become too small (insignificant) to alter the output. The LSTM does the mitigation by using a series of gates

[e] A one-to-one mapping was done from -1 to 1 and 1 to 0, where the mapped 0 meant normal and the mapped 1 meant anomalous.

contained in memory blocks that regulate the flow of information. Gates are also are used to control what information goes into memory cells and how long that information is maintained.

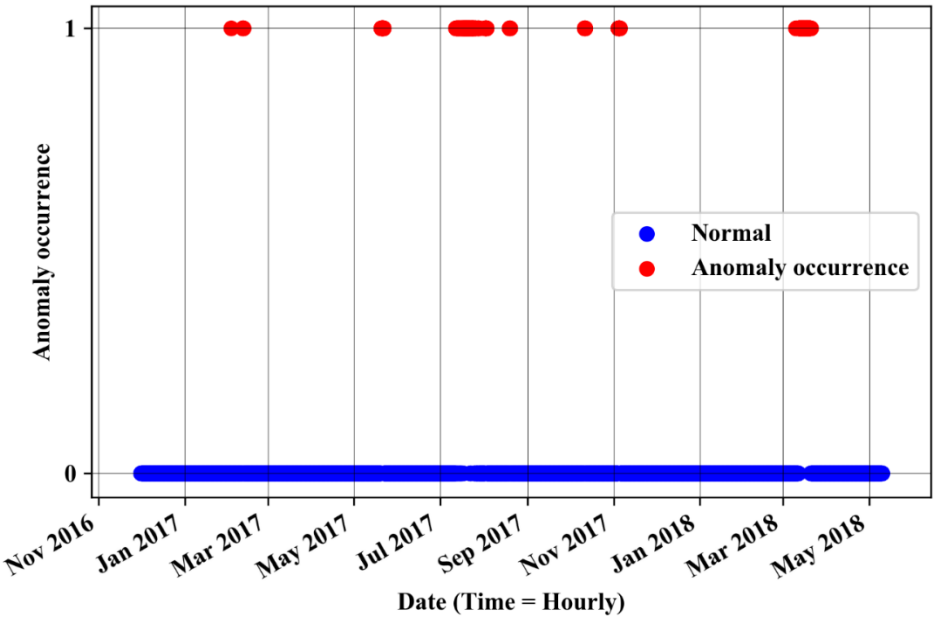


Figure 17. Random forest algorithm performing anomaly detection on hourly time-series data set for FCU D.

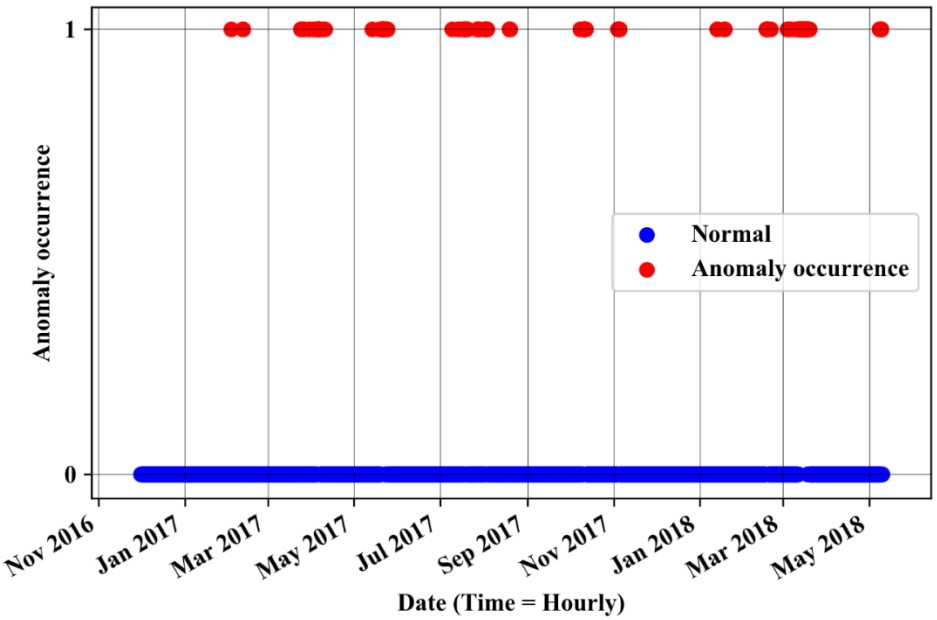


Figure 18. Random forest algorithm performing anomaly detection on hourly time-series data set with PCA for FCU D.

3.3.1 Method Description

A typical LSTM unit is composed of a cell, an input gate, an output gate, and a forget gate. The input gate scales input to the cell and behaves as a write operation. The output gate scales output to cells and behaves as a read operation. The forget gate scales old cell values and behaves as a reset operation. Each of the gates in the LSTM unit behaves as a switch to control read/write operations and thus gives the effect of long-term memory to the model.

Memory allows LSTM networks to excel at finding complex relationships within a multivariate feature space. LSTM's ability to process sequential data and have internal memory-cell units made it perfect to model the time-series data set for the project. The memory attribute of the architecture would be able to remember seasonal patterns as well as cyclic patterns that constitute normal operating conditions. Given enough data, LSTM has the potential to learn patterns from one year to the next as well.

Most approaches that use LSTM networks as anomaly detectors focus on training the models on historical data and predicting a future outcome *y*. The predicted outcome *y* is then compared to the actual observation and, if the value is more than *d* standard deviations away, then it is considered anomalous. This approach would not suffice because the algorithm must wait for an observation to be significant to determine whether a data point was an anomaly. In the case of this research, this would mean waiting until just before a mechanical failure occurred to confirm an anomaly. This identification would come too late.

The approach taken was to analyze the LSTM network's ability to learn as new dynamic information was being added. Ideally, the model would be trained with historical data that were assumed to be normal. Once the model was properly fitted, meaning it learned the training data set, a network architecture was in place, capable of learning normal data. Afterwards, new data was added to the training set, and the model was retrained. The concept is that if the new data added were normal, the model would be able to learn it and not predict erroneously from the training set. On the other hand, if new data were anomalous, then the model would begin to predict erroneous values in the training set as it tried to minimize the internal-loss function. This approach would consequently be able to catch anomalies before actual mechanical failures and behave as a preventative system.

The method took in a training data set and the predicted values from the model for the training set. The data were processed sequentially, one time unit at a time. At each time unit, the actual and predicted values were compared with the actual and predicted values *t* timesteps ahead respectively. A line was formed between the two actual values, and another line was formed between the two predicted values. The angle between the two lines was used as an indication of the anomaly. A divergence between the lines indicates that the actual and predicted values were falling apart (i.e., an anomaly was occurring). The pseudocode for this process:

- Load data with actual and predicted values
- Set timestep *t* (in mins, hours, etc)
- Set anomaly counter $c = 0$
- For each record r_i in the file
    - Form a line between actual value at r_i and r_{i+t}
    - Form a line between predicted value at r_i and r_{i+t}
    - If lines diverge from each other then increment *c* by 1
- If c >= 1
    - Mark the newly added data as anomalous.

To train the LSTM, the data set had to be properly curated and processed, as described in Section 2.2 . After the original data set was curated, it had to be formatted accordingly to be used as input to the LSTM network. It had to be structured in time-windows of size *l* where the granularity was in hours. The variable *l* is a hyperparameter, and choosing the optimum value of *l* is not trivial. A value of ten (*l* = 10 hours) was used in this initial research. To construct each time window, a moving frame of size *l* was applied to the data set iteratively, with a step of one unit. Each time-window required verification for usability because all datapoints inside had to be temporally sequential in reference to the time unit. There are cases in which a time-window contains a set of data points which are not temporally sequential as a result of the data-curation process (i.e., removing bad data as described in Section 2.2) and must be disregarded as an input to the LSTM.

The time-window formatted data set was given to the LSTM as training, with the purpose to output the outlet temperature in the following time step. The internal architecture of the network was composed of three stacked LSTM layers followed by a dense layer (i.e., the last layer that receives input from all neurons and generates one output) which was responsible for the output of the model. The LSTM layers use a rectified liner unit (ReLU) activation function (i.e., determinative of when the neuron fires) to speed computation. A value of 20% was used for the dropout rate. Optimizing the dropout rate is used to prevent overfitting and excessive memorization. ReLU activation functions also perform exceptionally well when the model never has to predict a negative value. The used LSTM network architecture is

- Layer 1 with 64 LSTM units
- Layer 2 with 128 LSTM units
- Layer 3 with 256 LSTM units
- Layer 4 with 1 dense layer unit.

The last layer in the network is the dense layer, which flatten out to a single output for prediction. The dense activation function was linear because the predicted target was a continuous non-discrete value (outlet temperature of FCU A, B, C, or D). The loss function that needs to be minimized in the learning process was set to mean squared error (MSE). [f]

3.3.2 Tools

To execute the LSTM algorithm, an additional environment was created within the Anaconda distribution. The environment was set up with Python 3.6 and Jupyter Notebook, Version 4.4.0. The Keras ML library was installed and used to handle the architecture of the LSTM. Keras is a wrapper library for the TensorFlow library that provides ML capabilities. Because this approach for anomaly detection is very computationally expensive, the graphical processing unit (GPU) version of Keras and TensorFlow were activated. GPU processing speeds up the computation time exponentially when compared to using the central processing unit. Numpy and Pandas were used to store the time-series data sets and create time-windows for input to the network. Matplotlib was used to visualize how the model was learning during the training and to check the loss function as time advanced. The graphing abilities were also used to show the actual values of the training set, compared to the predicted values of the final model after training.

3.3.3 Results

The LSTM model showed promising results during the training phase: it was able to learn what normal behavior was. Training and validation split were set to 80/20 respectively. The model's loss function consistently decreased for both the training set and the validation set (Figure 19). More importantly, the loss function for both sets didn't deviate from one another and converged at a low value.

f The optimizer used in the network was root mean square propagation (rmsprop), which has the benefit of automatically adjusting the learning rate.

Convergence of the function loss for the data sets means that the predictive model extracted sufficient information to learn and did not simply memorize the relations.

The LSTM model also had a very low root mean squared error (RMSE) for both the training-data set and internal-validation set. The RMSE for training was 0.05 while the RMSE for the validation was 0.16. The low values in RMSE are indications that the model also learned some natural operating behaviors and cycles. Figure 20 shows that once new data, in this case an additional day closer to the mechanical failure of FCU D, was introduced to the LSTM model, the ability to predict its own training set diminished. This is visible as it forms a divergence in the prediction value from the actual values in the training set in Figure 20, with the red line representing predicted, blue representing actual, and black representing the actual data including the future behavior. The model was able to detect an anomaly before the mechanical failure of FCU D using this approach as early as May 3, 2018 (i.e., 8 days ahead of time). The same method was also inadvertently able to detect a FCU B outlet-temperature sensor failure on May 7, 2018 (2 days ahead of time), as shown in Figure 21. The failure of FCU A was impacted by the failure of FCU D because the model had never been trained for a scenario that had three fans running under similar plant conditions (especially since twenty-four tags are common among all fans), so this was a new behavior, and it needed more time to learn before it could predict the next failure.

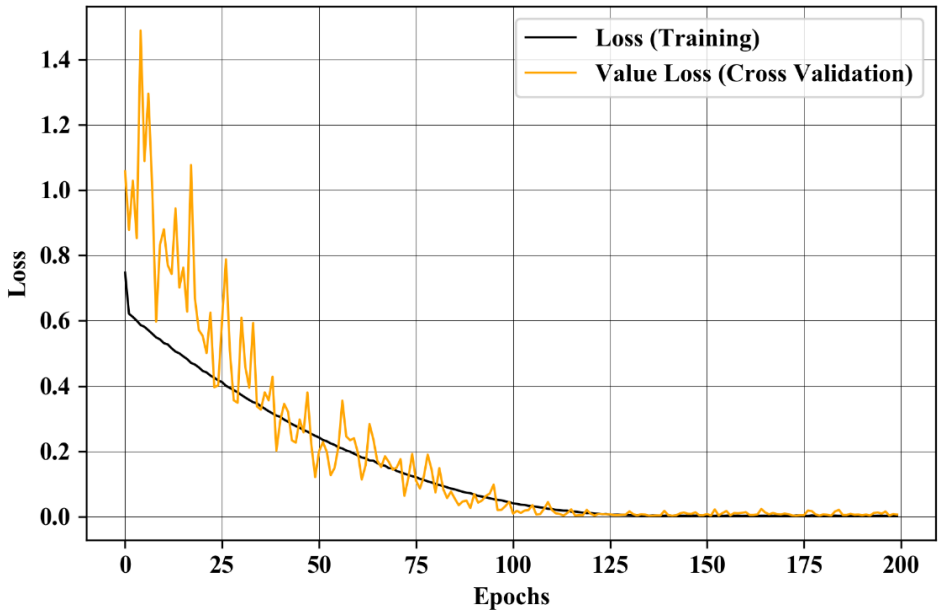


Figure 19. LSTM-model loss function showing the convergence of the training and validation data sets. Graph shows the LSTM's ability to learn.

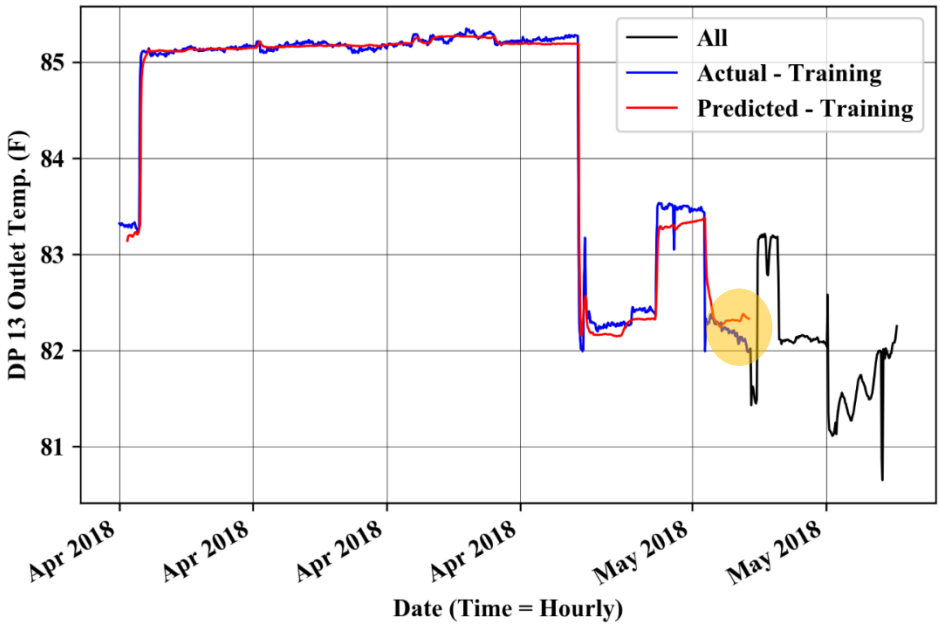


Figure 20. LSTM model prediction diverging from actual value on May 3, 2018, training data for FCU D (8 days ahead of mechanical failure).

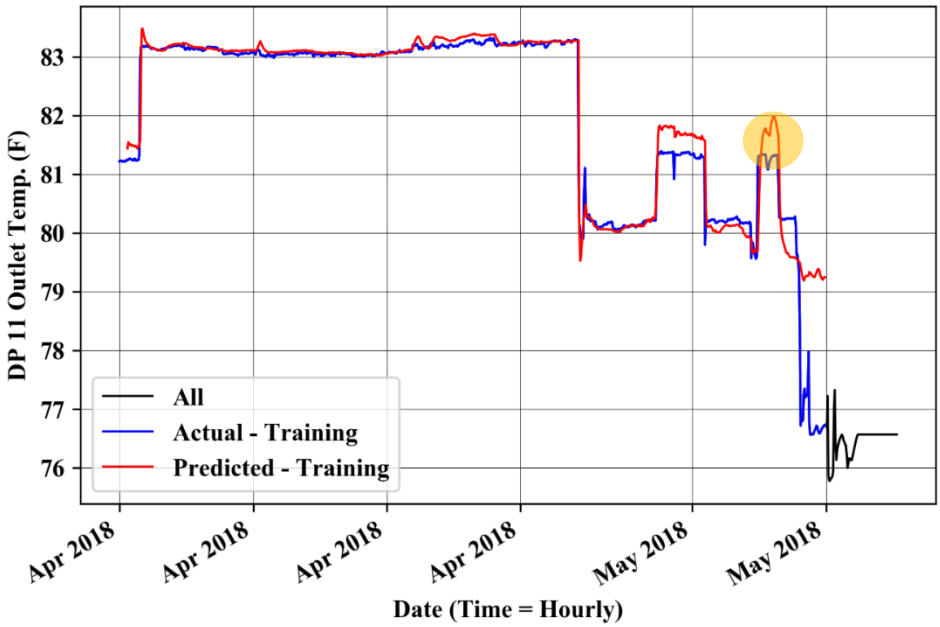


Figure 21. LSTM model prediction diverging from actual value on May 7, 2018, training data for FCU B (2 days ahead of sensor failure)

4. CONCLUSIONS AND FUTURE WORK

In May 2018, two drywell FCUs at the CNS failed in a catastrophic manner, resulting in plant outage for 6 days. The lack of vibration sensors on the equipment required finding new means to predict this failure through detection of small process anomalies. These anomalies can be found in data that are not directly related to the failure. The detection of small anomalies lies in small deviations that are

challenging to detect especially when the time element is introduced and small anomalies become part of the new norm. Ignoring the time aspect and considering data measured at any point of time using K-means did not yield any valuable insight until the failure did occur. Dimensionality reduction via PCA was also applied as a means of reducing data dimensionality to improve the anomaly classification. This did not yield valid results. An isolation forest method was also implemented as an unsupervised learner to predict anomalies because they are designed for similar applications, but was unsuccessful due to a bias towards declaring normal states as anomalous (i.e., signaling false alarms). Both K-means and isolation forest method are unsupervised ML methods.

LSTM was also applied. A baseline LSTM was developed by training a model on CNS data associated with normal drywell operations and examining reconstruction-error differences in temporal windows associated with known failure events. LSTMs were selected because they are suited to handle long-term temporal dependencies because the architecture includes a memory cell to maintain temporal information. The LSTM was customized with a new deviation/anomaly criterion that is based on comparing the training and actual model before and after introducing new temporal data. It was able to predict one of the FCU failures 8 days before it occurred and, coincidentally, detected a sensor failure 2 days before its occurrence. The failure of the second FCU occurred around two weeks after the first. Because the model had never been trained for a scenario that had three fans running under similar plant conditions, this was a new behavior, and the LSTM network required more time to learn it before it could predict the next anomaly. This caused the method to miss the second failure.

Throughout this effort, data size, quality, complexity, availability, and curation all had significant impacts on potential modelling pathways and outcomes. The sparsity of failures drove the research into detection of norms, rather than failures. The next scope of work will apply the developed method periodically to data from the plant to determine if anomalies are present. This will involve acquiring the data from the plant computer, running scripts to prepare data (as was performed in this work), and analyzing the results for anomalies. Future research will branch in two directions. One will explore expanding this scope to develop equipment-agonistic methods for anomalies detection. Achieving this objective is challenging, but would be very rewarding. The second approach will be a physics-informed ML model. A thermal-hydraulic model will be created using the Reactor Excursion and Leak Analysis Program (RELAP5) code representing a digital twin of the FCU to simulate the fan at normal operating conditions and under different failure scenarios. RELAP5 is a light-water reactor transient-analysis code developed for the U.S. Nuclear Regulatory Commission for simulation of a wide variety of hydraulic and thermal transients in nuclear power plants (Mesina 2016). The purpose of augmenting the ML model with simulation data is to compare the actual nitrogen supply and return temperatures based upon plant computer data, with predictions based upon mass, momentum, and energy conservation. Differences between the simulated and predicted FCU temperatures will then be examined for anomalies indicative of incipient fan failure.

5. REFERENCES

Ahmad, S., A. Lavin, S. Purdy, and Z. Agha, 2017. "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing* Vol. 262, pp. 134–147.

Al Rashdan, A., J. Smith, S. St Germain, C. Ritter, V. Agarwal, R. Laurids, and T. Ulrich, 2018. *Development of a Technology Roadmap for Online Monitoring of Nuclear Power Plants*, INL/EXT-18-52206, Idaho National Laboratory.

Ankerst, M., M. Breunig, H. Kriegel, and J. Sander, 1999. "OPTICS: Ordering points to identify the clustering structure," *ACM SIGMOD Record* Vol. 28, No. 2, pp. 49–60.

Azevedo, D., A. Ambrósio, and M. Vieira, 2012. "Applying data mining for detecting anomalies in satellites," in *2012 Ninth European Dependable Computing Conference*, pp. 212–217.

Beghi, A., L. Cecchinato, C. Corazzol, M. Rampazzo, F. Simmini, and G. Susto, 2014. “A One-Class SVM Based Tool for Machine Learning Novelty Detection in HVAC Chiller Systems” in *IFAC Proceedings* Vol. 47, No. 3, pp. 1953–1958.

Carrasquilla, U., 2010. “Benchmarking algorithms for detecting anomalies in large data sets,” *MeasureIT*, Nov, pp.1–16.

Cortes, C., and V. Vapnik, 1995. "Support-vector networks." *Machine learning* Vol. 20, No. 3, pp. 273-297.

Ester, M., H. Kriegel, J. Sander, and X. Xu, 1996. “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)* Vol. 96, No. 34, pp. 226–231.

Farber, J., D. Cole, A. Al Rashdan, and V. Yadav, 2019. "Using Kernel Density Estimation to Detect Loss-of-Coolant Accidents in a Pressurized Water Reactor." *Nuclear Technology* Vol. 205, No. 8 pp. 1043-1052.

General Electric, 2011. *General Electric Systems Technology Manual- Chapter 4.1: Primary Containment System,*, USNRC HRTD Rev 09/11. Last accessed: August 30, 2019. Link: https://www.nrc.gov/docs/ML1125/ML11258A322.pdf

Hariri, S., M. Kind, and R. Brunner, 2018. “Extended Isolation Forest,” Link: https://arxiv.org/abs/1811.02141

Hayes, M. and M. Capretz, 2015. “Contextual anomaly detection framework for big sensor data,” *Journal of Big Data*, Vol. 2, No. 2.

Hochreiter, S., and J. Schmidhuber, 1997. “Long Short-term Memory,” *Neural computation* Vol. 9, pp. 1735–1780.

Hu, R., J. Granderson, D. Auslander, and A. Agogino, 2019. “Design of machine learning models with domain experts for automated sensor selection for energy fault detection,” *Applied Energy*, Vol. 235**,** pp. 117–128.

Jolliffe, I., and J. Cadima, 2016. "Principal component analysis: a review and recent developments." *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences.* Vol. 374, No. 2065, ID. 20150202.

Lewinson, Eryk, 2018. Outlier Detection with Isolation Forest. Last accessed: August 30, 2019. Link: https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e

Liu, F. K. Ting, and Z. Zhou, 2008. “Isolation forest,” in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422.

Lochbaum, D., 2013. *Containment Venting is Cool(ing)–But Needs Filters*, [webpage], Last accessed: August 30, 2019. Link: https://allthingsnuclear.org/dlochbaum/containment-venting-is-cooling-but-needs-filters

MacQueen, J., 1967. “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, No. 14, pp. 281–297.

Malhotra, P., V. TV, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, 2016. “Multi-sensor prognostics using an unsupervised health index based on LSTM encoder-decoder,” Presented at *1st ACM SIGKDD Workshop on Machine Learning for Prognostics and Health Management, San Francisco, CA, USA*. Link: https://arxiv.org/abs/1608.06154

Mesina, G. 2016. “A History of RELAP Computer Codes,” *Nuclear Science and Engineering* Vol. 182, pp. 5–9.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel et al, 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, Vol. 12(Oct), pp. 2825-2830.

Petsche, T., A. Marcantonio, C. Darken, S. Hanson, G. Kuhn, and I. Santoso, 1996. “A neural network auto-associator for induction motor failure prediction,” *Advances in Neural Information Processing Systems* Vol. 8, No. 8, pp. 924–930.

Python development team, 2018. “Python Language Reference”, *Python Software Foundation*, Release 3.6.5.

Song, L., H. Wang, and P. Chen, 2018. “Vibration-Based Intelligent Fault Diagnosis for Roller Bearings in Low-Speed Rotating Machinery,” *IEEE Transactions on Instrumentation and Measurement* Vol. 67**,** pp. 1887–1899.

U.S. Nuclear Regulatory Commission,2019. *Drywell*, [webpage], August 30, 2019. Link: https://www.nrc.gov/reading-rm/basic-ref/glossary/drywell.html

Wang, H., and Y. Chen, 2016, “A robust fault detection and diagnosis strategy for multiple faults of VAV air handling units,” *Energy and Buildings*, Vol. 127, pp. 442–451.

Yuan, M., Y., Wu, and L. Lin, 2016. “Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network,” in *2016 IEEE/CSAA International Conference on Aircraft Utility Systems* (AUS), pp. 135-140.

Ward Jr, J. 1963, “Hierarchical grouping to optimize an objective function,” *Journal of the American Statistical Association* Vol. 58, No. 301, pp. 236–244.

Wikimedia Commons contributors, 2016. *KMeans-density-data*[webpage], August 30, 2019. Link: https://commons.wikimedia.org/w/index.php?title=File:KMeans-density-data.svg&oldid=214849130

Wold, S., K. Esbensen, and P. Geladi, 1987. “Principal component analysis, *Chemometrics and intelligent laboratory systems* Vol. 2, Nos. 1–3, pp.37–52.