EPiC Series in Computing

Volume 79, 2021, Pages 54–63

Proceedings of ISCA 34th International Conference on Computer Applications in Industry and Engineering

Automating Predictive Maintenance for Energy Efficiency via Machine Learning and IoT Sensors

Paul M. Bodily[1], Isaac D. Griffith[1], Mary Hofle[2], Omid Heidari[2], Safal Lama[2], Avery Conlin[2], Andrew Christiansen[1], Delaney Moore[1], Kellie Wilson[2], Anish Sebastian[2], and Marco P. Schoen[2]

[1] Department of Computer Science, Idaho State University, Pocatello, ID, 83209
[2] Department of Mechanical Engineering, Idaho State University, Pocatello, ID, 83209
{bodipaul,grifisaa,hoflmary,heidomid,lamasafa,conlaver,
chriandr,moordela,wilskell,sebaanis,schomarc}@isu.edu

Abstract

The arise of maintenance issues in mechanical systems is cause for decreased energy efficiency and higher operating costs for many small- to medium-sized businesses. The sooner such issues can be identified and addressed, the greater the energy savings. We have designed and implemented an automated predictive maintenance system that uses machine learning models to predict maintenance needs from data collected via data sensors attached to mechanical systems. As a proof of concept, we demonstrate the effectiveness of the system by predicting several operating states for a standard clothes dryer.

1 Introduction

A significant portion of energy losses and inefficiencies among small- to medium-sized businesses and consumers arise due to a common set of maintenance-related issues that can be assessed and mitigated through the application of predictive modeling using data collected both manually and automatically via sensors. Historically, the keys to saving energy include the implementation of energy management techniques, specifically equipment maintenance and monitoring techniques [1]. In addition, predictive maintenance uses equipment sensors (manually or automatically operated) that indicate and predict when maintenance will be required [1].

Both sensors and a commodity Internet of Things (IoT) platform that can serve as the basis for these sensors are readily available. Additionally, machine learning has been shown to be highly effective at predictive modeling [2]. Combined, these are capable of automatically collecting, propagating, and assessing underlying maintenance data, all of which are necessary to develop the tools required by managers to effectively plan and manage energy efficient maintenance [3]. In this paper we describe the design and implementation of cost-effective, automated solutions for overcoming maintenance-related energy losses in small- to medium-sized businesses. Our objective in this application is to perform assessments of existing operational infrastructure and constraints that represent many of the systems found in small- to medium-sized manufacturing businesses, such as material/product handling, fluid flow, electric motor

drive systems, and other systems. Maintenance issues caused by the failure or degradation of system subcomponents (e.g., vibration causing wear in bearings) can be identified by a change of sound, movement, or temperature, indicating possible changes within a subcomponent that are outside the required operational range.

Much recent attention has focused on automative prediction using machine learning as an integral part of broadly emergent fields of Industrial IoT (IIOT) and Industry 4.0. Building information modeling (BIM) and IoT have been suggested as a means of facility maintenance management (FMM) [4]. In particular the proposed system uses artificial neural networks (ANNs) and support vector machines (SVMs) to perform condition monitoring and fault alarming, condition assessment, condition prediction, and maintenance planning. Their findings suggest that the future condition of mechanical, electrical, and plumbing (MEP) components for maintenance planning can be efficiently predicted, particularly in the architecture, engineering, construction, and facility management (AEC/FM) industry.

Published in 2019, a systematic literature review of machine learning methods applied to predictive maintenance asserts that the performance of predictive maintenance applications depends on the appropriate choice of the ML method [5]. A second systematic literature review published in 2020 provides a similar overview of machine learning algorithms used for predictive maintenance (including ANNs, SVMs, Decision Trees, Random Forests, and Linear, Logistic, and Symbolic Regression) [6]. This review includes a review not only of the types of algorithms, but also of the equipment, data acquisition devices, and most common commercial ML platforms used in predictive maintenance architectures.

As predictive maintenance capabilities have broadened, other work has focused on the optimal management of tasks that result from predictive maintenance systems. One comparison of optimization algorithms used in tandem with predictive machine learning in this domain found that a genetic algorithm-based resource management algorithm outperformed MinMin, MaxMin, FCFS, and RoundRobin algorithms in execution time, cost and energy usage [7].

Related work has specifically looked at the implementation of automated predictive maintenance systems in the so-called “brownfield” which refers to technologically-outdated industrial or commercial sites. As an example, this work looks at the process of retrofitting a heavy lift Electric Monorail System at the BMW Group sites with low-cost sensors, an IIoT architecture and cloud-based machine learning to avoid unplanned downtime, increase availability and efficiency, and save costs through optimized maintenance strategies [8].

In the present study we collect data for use in the design, development, and testing of an IoT sensor platform and cloud-based smart decision-support tool incorporating predictive machine learning to improve and automate decisions for energy efficiency and curtailment.

2 Methods

Figure 1 shows a high-level overview of the system we have designed. Sensors are attached to mechanical systems. Data from the sensors is collected by an IoT device (i.e., a Raspberry Pi). The IoT device sends data to a cloud server which acts as both a data warehouse and as a platform for data analysis using machine learning models. A user interface (UI) provides access to data and system configuration information at both the IoT and server levels. The following subsections go into each of these components in detail.

Table 1: Catalog of sensors

Sensor name	Attribute measured	Data communication protocol
MPU6050	Vibration	I2C
1528-2526-ND	IR Break Beam	GPIO
MLX90614ESF	IR Temperature	I2C
DHT22	Temperature & Humidity	Proprietary
MAX446	Sound	SPI (via ADC)
YHDC-SCT-013-000	Current	SPI (via ADC)

2.1 Mechanical Systems

Much of the energy consumption of small- to medium-sized businesses comes from various mechanical systems (e.g., pumps, motors, etc.). As with all mechanical systems, the energy efficiency of these systems depends on maintenance needs being met in a timely and routine manner. Breakdown and degradation in performance of motors, belts, and pumps can lead to energy losses. In our study we included four mechanical systems that collectively included a variety of motors, pumps, and belts. This included two dryers, one blender, and a water pump.

2.2 Sensors

Automated assessment of maintenance needs is conducted by measuring attributes of the mechanical systems. Such attributes may include temperature (both that of the ambient and particular system elements), sound, vibration, rotation speed, and electric current. To measure these attribute we attach several sensors to each mechanical system. A catalog of the sensors attached to each of our four mechanical systems can be found in Table 1. For each sensor the data communication protocol is also listed.

Each sensor is attached to the mechanical device in a position that optimizes the quality of data collection for the sensor. Figure 2 illustrates the placement of sensors on the clothes dryer we used for the experiment we describe below.

To collect data from the sensors, we connected all of the sensors for a particular mechanical

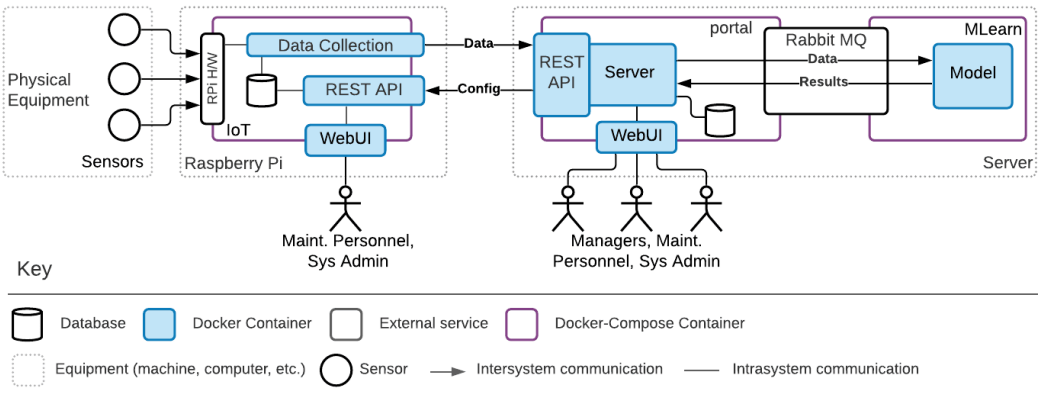

Figure 1: System overview

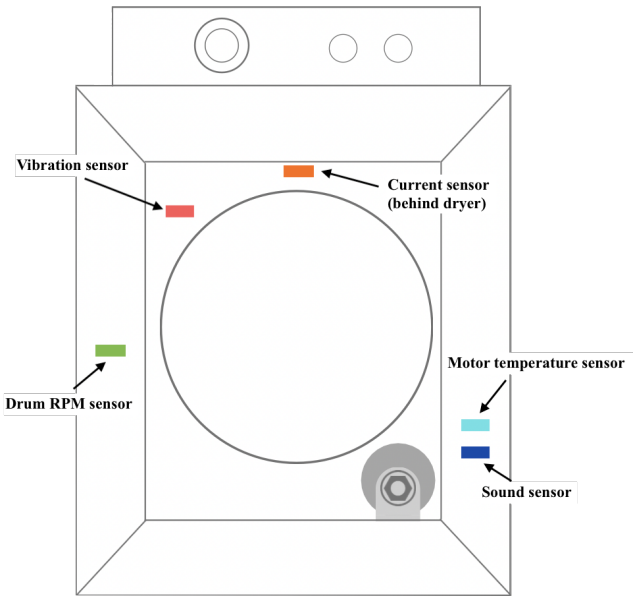

Figure 2: Diagram of sensor placement on disassembled drying machine. We attached a MAX446 sound sensor near the motor to observe variance in sound produced by motor. We attached an IR temperature sensor on the wall of the casing to read the temperature of the motor windings. We attached a MPU6050 vibration sensor on the rear wall for direct contact with the motor and the rotating drum. A current sensor was attached in the rear of the dryer to measure electric current used by the dryer. We attached an IR break beam sensor on the side wall to detect the rotations per minute of the dryer drum. Not shown is a DHR temperature/humidity sensor for measuring aspects of the contextual environment.

system to a single Raspberry Pi by means of an attached hardware board called a Pi HAT (short for "Hardware Attached on Top"). A diagram of the configuration of sensor wires to the Pi HAT is shown in Figure 3. Sensor wires are soldered onto the hardware board, being careful not to burn or damage the connection or the board itself.

2.3 Software overview

Software for the automated predictive maintenance system is divided primarily between A) software local to the IoT sensor platform and B) portal software on a cloud server. Software on the IoT sensor platform is designed to collect data from the sensors; temporarily store small quantities of data; and send data in batches to the portal software. The portal software is designed to receive batches of data from one or more IoT devices; to act as a data warehouse for data from multiple devices across multiple locations; and to perform data analysis using machine learning for anomaly detection for the automation of predictive maintenance needs. Both the IoT and portal software are developed with an accompanying browser-based UI that reports on the state of the system and allows for configuration of connected sensors/servers.

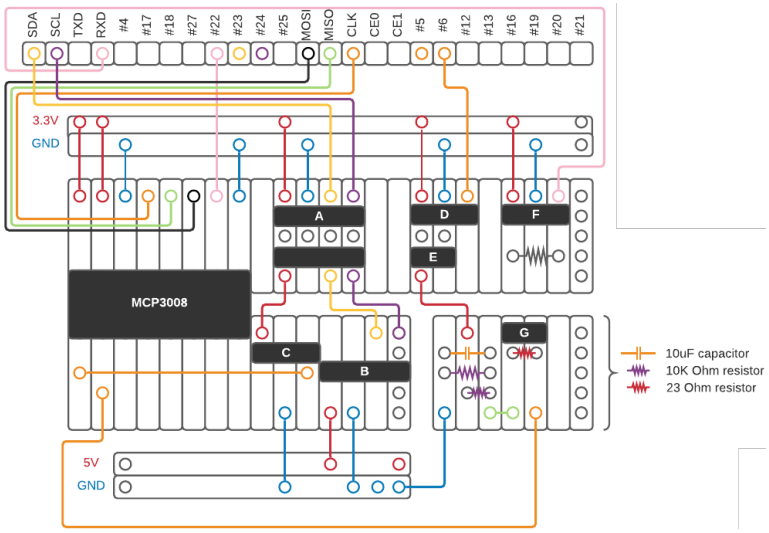

Figure 3: Diagram of the configuration of sensor attachments to a Pi HAT showing pin connectors for a 3-axis MPU6050 accelerometer (A), a MLX90614ESF IR temperature sensor (B), a MAX446 electret microphone amp (C), a 1528-2526-ND IR break beam receiver (D), a 1528-2526-ND IR break beam transmitter (E), a DHT22 humidity/temperature sensor (F), and a YHDC-SCT-013-000 current sensor (G). Note that A-F are pin connectors soldered to the board.

All software is implemented in Python 3[1].

From an implementation standpoint, much of the software implemented is common to both the IoT and the portal platforms, so it is more appropriate to consider the software organization in terms of functional needs. Broadly speaking there are six functional modules in the system which we describe in the following paragraphs.

The *Base-Server* module provides a common architecture to be inherited by the IoT and Portal servers. It includes the basic functionality and protocol software needed for communication of devices across the web. It also contains the logic for setting up the database and managing the flask web-service. This module also stores some web-service routes shared between IoT and Portal servers that handle creating users, authorization and configuration.

The *IoT* module runs on the Raspberry Pi and specifically includes software needed to collect data and provides a web-based UI for configuration. This software manages a web server and client. The web server inherits from the base-server module. The client web page can be used to manage the IoT platform by configuring sensors, servers, and database settings. This platform can also be used to view the data of a single machine. The sole responsibility of this module and the hardware it rests on is to read the data from the sensors that are monitoring machinery. The data that is read is then stored in a database where it can be used in other modules. This data will also be displayed on the client web page.

The *Portal* module handles data aggregation and provides a web based UI for displaying sensor data and managing users and sensors. The web server in this module inherits from the base-server module. The portal module is responsible for receiving and aggregating data that is collected from all the IoT devices. The portal software runs on its own server machine where

[1]Software download available at https://github.com/isu-avista

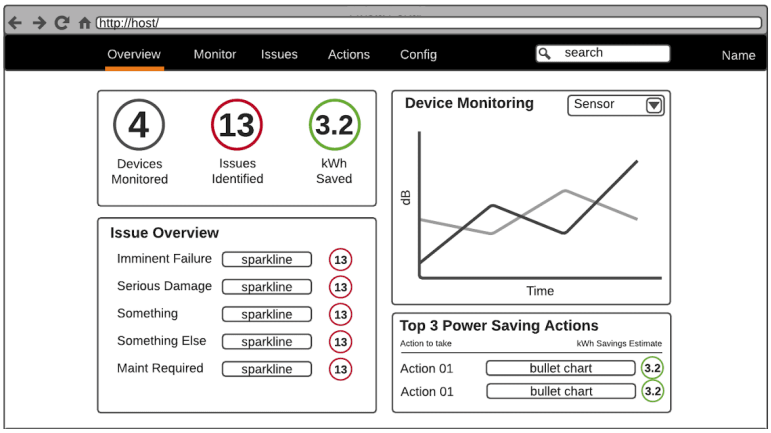

Figure 4: Mock-up of the user interface for the portal web page, allowing users to view problems with machinery, look at aggregated data, and record fixed issues.

data collected from all IoT devices can be stored. The portal web page will provide the end user with a UI to view problems with machinery, look at aggregated data, and record fixed issues if needed (see Figure 4). Users, depending on their role permissions, may also edit their profile; edit, add, and delete sensors; and edit, add, and delete users.

The *Data* module houses an interface to our database schema through an object-relational mapping for use in the other system modules. This allows both the IoT and portal devices to store and transfer data. The data module also manages users. Users can have different roles which include devices, administrators, managers, and maintenance workers. The data module controls what tasks certain roles have access to. For an example, a maintenance worker can edit their profile on the portal, but they may not add or delete other users like a manager is able to do. The data module also manages API keys.

The *Sensors* module contains implementations for each of the physical sensors used for hardware data collection. Each sensor-specific implementation is customized according to the protocol defined for the sensor. The module periodically retrieves data from all the sensors and stores the data locally. When a specified period of time has been reached then the data is sent to and recorded on the main server where other modules can make use of it.

The *Control* module handles message queuing with RabbitMQ to send data and predictions respectively to and from the machine learning module. The Control module provides an interface through which we can interact with RabbitMQ using a python library called Pika. Specifically, the Portal service employs a docker container with a Publisher that will continually check the database for new data. If there is new data it will be sent to a Consumer. This Consumer will then use learners from the MLearn module that will make predictions on the data based on some machine learning algorithm. The resulting prediction is then returned to the Publisher and provided to the Portal if there is a predicted issue. The idea here is that the Portal web page will then display to the end user that something is wrong with a machine being monitored by an IoT device.

The machine learning or *mlearn* module is responsible for loading pre-trained machine learning models and making automated predictions on data. For this phase of the project, there are four different classifiers that have been used: Perceptron, Naive Bayes, Random Forest, and Multilayer Perceptron. These classifiers are built using Waikato Environment for Knowledge

Analysis (WEKA) [9].

3 Results

As a proof of concept and to test the design of the system so far, we conducted an initial experiment in which we equipped a clothes dryer with all of the sensors listed in Table 1 minus the current sensor (for reasons described below). The sensors gathered the rotations per minute of the dryer belt, the internal temperature, the external temperature, sound made by the dryer, humidity inside the dryer, and the vibration of the dryer. The system was set to collect data from the sensors at 30-second intervals. We ran the dryer for approximately 10 minutes under 5 different experimental conditions: off; on but with the belt removed; on with the belt attached and the drum empty; on with the belt attached with a load of dry towels; and on with the belt attached with a load of wet towels. Data for a total of 89 training instances was collected (18 off; 18 no belt; 17 empty; 18 dry towels; and 18 wet towels)[2]. We trained four different classifiers—Perceptron, Naive Bayes, Random Forest, and Multilayer Perceptron—with the goal of determining how well each model could be used to predict whether or not the dryer was off, on with no belt, on with nothing inside, on with dry towels, or on with wet towels (see Fig. 5)[3].

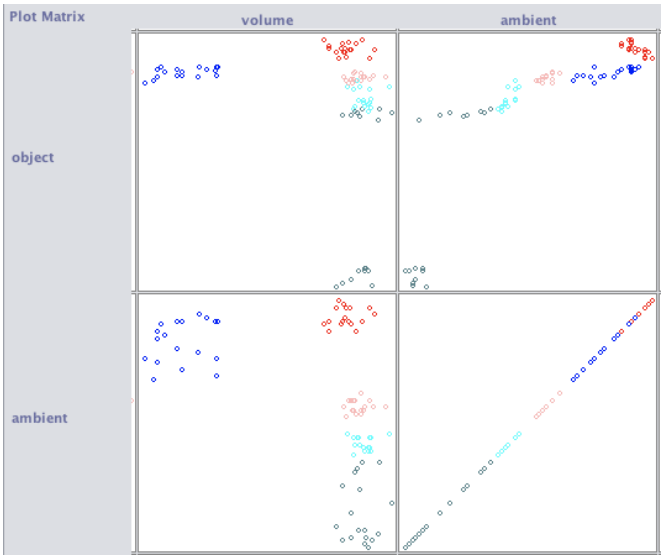

Figure 5: Scatterplot submatrix generated by WEKA showing correlation between pairs of input features. Instance labels are as follows: off (blue); no belt (red); empty drum (cyan); dry towel load (grey); wet towel load (pink). As expected, volume readings are lower when the dryer is off. It is suspected that the variation in object (i.e., motor) and ambient temperature readings may be caused more by the time of day in which data was collected than by the experimental condition of the dryer.

Of the four trained models, the Random Forest and Multilayer Perceptron models achieved the highest predictive accuracy (see Table 2). That these models showed improvement over the

[2]The dataset is available in ARFF format at https://tinyurl.com/dryerarff.
[3]A video demonstrating experimental design available at https://tinyurl.com/dryerexperiment.

Table 2: Classifier results

Classifier	Correctly Classified	Predictive Accuracy
Perceptron	70/89	78.65%
Naïve Bayes	87/89	97.75%
Random Forest	**88/89**	**98.88%**
Multilayer Perceptron	**88/89**	**98.88%**

Table 3: Confusion matrix for the Perceptron classifier

		Predicted class				
		off	**no belt**	**empty**	**dry towels**	**wet towels**
Actual class	**off**	**18**	0	0	0	0
	no belt	0	**18**	0	0	0
	empty	0	0	**8**	2	7
	dry towels	0	0	1	**17**	0
	wet towels	0	0	9	0	**9**

Perceptron and Naïve Bayes models suggests that predicting mechanical system conditions as a function of the configured sensors will require a classifier capable of modeling non-linearly-separable data classes. An examination of the confusion matrix for the Perceptron classifer shows that the model struggled to discriminate between when the dryer was on and empty versus when the dryer was on with a load of dry/wet towels (see Table 3). Misclassifications by the other three models followed this same pattern.

A decision tree generated via the Random Forest method is shown in Fig. 6. The first split attribute is ambient temperature, suggesting that the outside temperature (which correlates with the time of day and/or the order in which data for the several experimental conditions was collected) most effectively discriminates which experimental condition is predictable. While this results in high accuracy for this dataset, it will likely not generalize. In future experiments, we will collect data across a wide variety of environmental conditions.

Other intuitive insights come from examining nodes further down in the tree. For ambient temperature above or equal to 24.63°C, the volume feature very effectively discriminates between when the dryer is off versus on with the belt removed. For ambient temperature below 24.63°C, the "object" (i.e., motor) temperature is used to broadly discriminate between wet towels and dry/no towels, possibly indicative of the fact that greater energy is required to turn the drum with wet towels. These initial findings broadly suggest that predicting the operating conditions of a mechanical system from sensor data is achievable.

Our initial findings provide critical insights into several issues that should be addressed moving forward. First, as mentioned above, models must be trained on data collected under a variety of environmental conditions (i.e., ambient temperature, humidity, sound, etc.). Second, it is critical to ensure that sensors are accurately collecting and reporting data. We found from visualizing the data that the break beam RPM sensor and the 3-axis vibration sensor are currently not providing any meaningful data to the classifier. This finding prompted the subsequent addition of a full sensor sweep feature to the system combined with a battery of tests designed to indicate when individual sensors are failing to report meaningful data. Third, whereas we had initially assumed that predictive classes would be linearly separable from the data, the relatively poor performance of the Perceptron classifier suggests that a more sophisticated model will be necessary for optimal system diagnosis.

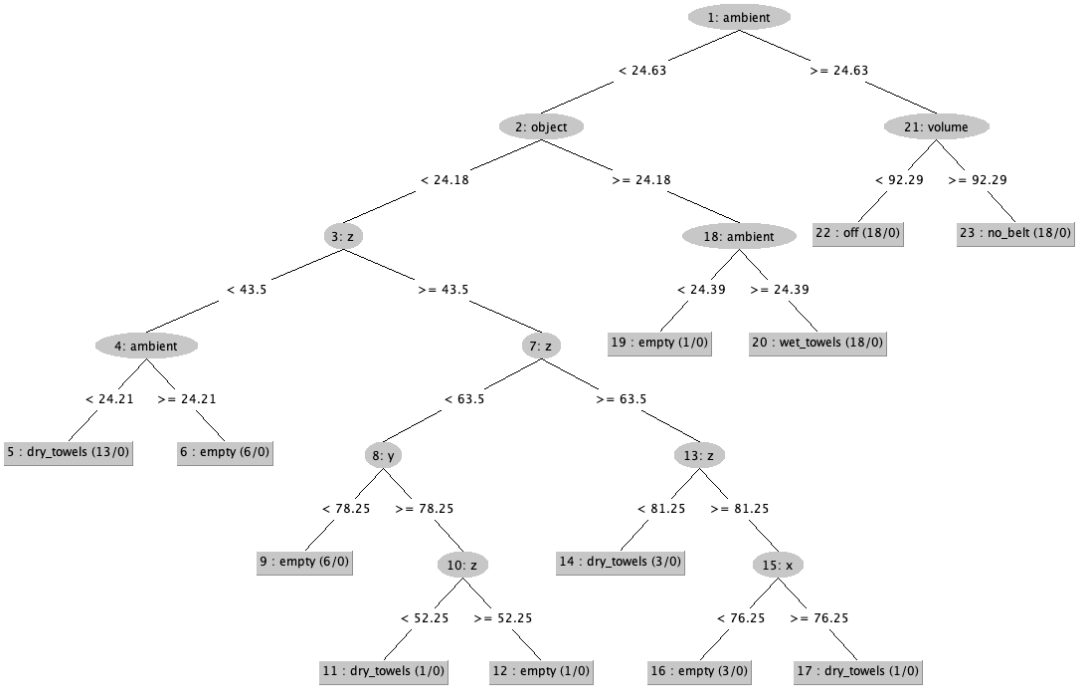

Figure 6: Decision tree generated by WEKA for predicting dryer status. Intermediate nodes represent input features (i.e., sensor readings). The features "ambient" and "object" refer to the temperature values in °C of the environment and motor respectively. The "volume" feature represents the sound volume in decibels. The "x", "y", and "z" features represent vibration movement along 3 axes. Leaf nodes are labeled with predicted class labels together with the number of training instances associated with the node that are accurately/inaccurately associated with the node's label.

4 Discussion and Conclusion

Our initial experiment serves to validate some of the critical aspects of our central hypothesis. It demonstrates that the state of a mechanical system—at a sufficiently nuanced level to be able to detect the difference between a dryer with load of wet towels versus a dryer with a load of dry towels—is well within the capability of the system we have designed. Furthermore it validates that our implementation thus far of the designed system is (with some minor reparable issues) working as anticipated from the sensor functions to the data collection to the data transmission to the data analysis. This initial experiment serves to highlight areas of needed improvement in the system, most notably the need to verify proper functionality of the sensors.

As mentioned, the current sensor was not included in our initial experimental design. This was for several reasons. First, the thought to add a current sensor came later in the project as a result of discussions about the means by which we might begin to estimate or measure energy savings in the system. Second, the YHDC-SCT-013-000 current sensor is not innately designed to interface with a Raspberry Pi, and the software programming necessary to interpret the data from the sensor proved more involved than that for the other sensors. Since our initial

experiment, this sensor has been fully implemented and future research will assess its usefulness for predicting maintenance needs.

Our work thus far has targeted a single machine (a clothes dryer). To expand on these results, we have already undertaken to expand consideration to several other machines, including a blender, a water pump, a water treadmill, and a second clothes dryer. This larger volume of IoT devices will allow assessment of a more real-world configuration of the system we have implemented. Our work thus far has also focused on classic classification methodology, that is, delivering data into remote cloud center for further processing. Future work will aim to convert this approach to more of an online, anomaly detection system in order to address concerns about latency and the overhead of the system.

In this paper we have summarized the initial design and implementation of an automated predictive maintenance system that uses machine learning and IoT sensors. Our focus has been on mechanical systems commonly employed in small- to medium-sized businesses. Having developed and tested an initial prototype of this system on a conventional clothes dryer and having demonstrated the ability of this system to effectively classify the operating state of the dryer, we look to subsequently expand our focus to challenges in implementing a network of such systems for improved generalization and learning across systems.

5 Acknowledgements

The authors gratefully acknowledge that this work was supported by a grant from Avista Corporation [AER R-43127].

References

[1] S. Backlund, P. Thollander, J. Palm, and M. Ottosson, "Extending the energy efficiency gap," *Energy Policy*, vol. 51, pp. 392–396, 2012.

[2] A. Mosavi and A. Bahmani. (2019) Energy consumption prediction using machine learning; a review.

[3] A. Lewis, A. Elmualim, and D. Riley, "Linking energy and maintenance management for sustainability through three American case studies," *Facilities*, 2011.

[4] J. C. Cheng, W. Chen, K. Chen, and Q. Wang, "Data-driven predictive maintenance planning framework for MEP components based on BIM and IoT using machine learning algorithms," *Automation in Construction*, vol. 112, p. 103087, 2020.

[5] T. P. Carvalho, F. A. Soares, R. Vita, R. d. P. Francisco, J. P. Basto, and S. G. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106024, 2019.

[6] Z. M. Çınar, A. Abdussalam Nuhu, Q. Zeeshan, O. Korhan, M. Asmael, and B. Safaei, "Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0," *Sustainability*, vol. 12, no. 19, p. 8211, 2020.

[7] Y. K. Teoh, S. S. Gill, and A. K. Parlikad, "IoT and Fog Computing based Predictive Maintenance Model for Effective Asset Management in Industry 4.0 using Machine Learning," *IEEE Internet of Things Journal*, 2021.

[8] P. Strauß, M. Schmitz, R. Wöstmann, and J. Deuse, "Enabling of predictive maintenance in the brownfield through low-cost sensors, an iiot-architecture and machine learning," in *2018 IEEE International Conference on Big Data*. IEEE, 2018, pp. 1474–1483.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.