6-24-2015

Fault detection and diagnostics of an HVAC sub-system using adaptive resonance theory neural networks

Christian Birk Jones

Christian B. Jones

Candidate

Engineering

Department

This dissertation is approved, and it is acceptable in quality and form for publication: *Approved by the Dissertation Committee:*

Dr. Andrea Mammoli, Chair

Dr. Thomas Caudell, Member

Dr. Lydia Tapia, Member

Dr. Francesco Sorrentino, Member

Fault detection and diagnostics of an HVAC sub-system using adaptive resonance theory neural networks

by

Christian Birk Jones

B.S., Civil Engineering, University of California, Davis, 2004
M.S., Civil Engineering, University of New Mexico, 2009

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Engineering

The University of New Mexico

Albuquerque, New Mexico

May, 2015

Dedication

To my wife Shannon for her encouragement, grace, and love.
To my parents, Tina and Larry, for their support, and love.
To my brother Henry for his intellect and humor.

"When in doubt go higher" – Anonymous

Acknowledgments

I would like to thank my advisor, Dr. Andrea Mammoli, for his support, inspiration, and encouragement. He has also had some brilliant ideas!! I would also like to thank Dr. Tom Caudell for teaching me about artificial neural networks and for his patience, and significant contributions to the present work.

I would like to thank my other dissertation committee members Dr. Lydia Tapia for her aid in the development of the data routing and storage for this project. To Dr. Francesco Sorrentino for his review, questions, and help.

Also, thank you to Dr. Manel Martinez-Ramón for his time, and key contributions that helped define procedures and analysis methodologies.

To my fellow student, Matt Robinson, thank you for your help in development of the IT framework and data collection system. Also, thank you to Matt and Yasser for the lounge discussions. Most importantly, thanks for the coffee!!

Thanks to Matt Lopez for his support and generous contributions.

Last but not least, thank you to Yearout Mechanical Inc. for their financial support. A special thanks to Kevin Yearout for his vision and desire to stretch the limits.

Fault detection and diagnostics of an HVAC sub-system using adaptive resonance theory neural networks

by

Christian Birk Jones

B.S., Civil Engineering, University of California, Davis, 2004
M.S., Civil Engineering, University of New Mexico, 2009

Ph.D., Engineering, University of New Mexico, 2015

Abstract

The commercial building sector consumed about 20% of the total primary energy in the U.S. in 2008 [2]. A significant yet avoidable portion of the energy consumption is due to inefficient system operations. The inefficiencies can be attributed to degrading HVAC sub-systems, and undetected abnormal conditions. Recognition and remediation of these conditions through advanced data analytics can reduce energy consumption by 5% to 20% [101]. This could save about $9 billion in utility costs in the U.S. alone.

Modern buildings are constantly sending messages in the form of sensor data. However, this data is only as good as the system that collects it. Therefore, the present work explores fault detection and diagnostics (FDD) of an HVAC sub-system, in particular an air handling unit (AHU), through the evaluation of various methods. The detection methods include a controls alarm threshold, rule-based expressions,

regression, one-class support vector machine (SVM), back-propagation, adaptive resonance theory (ART), and lateral priming adaptive resonance theory (LAPART). The diagnosis of AHU faults were performed using a multi-class SVM and LAPART algorithms. The results from the fault detection experiments were reviewed based on the two-class classification where the number of false positives and false negatives where compared. The diagnostic results were evaluated based on the comparison of precision and probability of detection values.

Contents

List of Figures

List of Figures

List of Tables

Glossary

ART	Adaptive Resonance Theory
AHU	Air Handling Unit
ANN	Artificial Neural Network
AUC	Area Under the Curve
BP	Back-propagation
BACnet	Building Automation and Control network
BAS	Building Automation System
B.E.R.T.	Building Energy Retrofit Testbed
CFM	cubic feet per minute
CHW	chilled water
CVRMSE	Coefficient of Variation of the Root Mean Squared Error
DMS	Database Management System
DDC	Direct Digital Controls
ER	Entity-relational diagram

FD	Fault Detection
FDD	Fault Detection and Diagnostics
FN	False Negative
FP	False Positive
HW	hot water
HVAC	Heating, Ventilating, Air Conditioning
IP	Internet Protocol
IT	Information Technology
kW	kilowatt electrical
kW_{th}	kilowatt thermal
kWh	kilowatt hour electrical
kWh_{th}	kilowatt hour thermal
LAN	Local Area Network
LAPART	Lateral Priming Adaptive Resonance Theory
LMS	least mean square
$\dot{m}$	mass flow rate
MEBldg	Mechanical Engineering Building
MSE	Mean Squared Error
ML	Machine Learning

NMBE	Normalized Mean Bias Error
PHP	Hypertext Preprocessor
PID	Proportional Integral Control
$\dot{Q}$	rate of heat transfer
Q	amount of heat transferred
RSME	Root Mean Squared Error
ROC	Receiver Operator Characteristic
S/F	Supply Fan
SaaS	Software as a Service
SQL	Structured Query Language
SVM	Support Vector Machines
TN	True Negative
TP	True Positive
TRNSYS	**Tra**n**s**ient **Sys**tem Simulation Tool
UNM	University of New Mexico
VAV	Variable Air Volume
VFD	Variable Frequency Drive
XML	Extensible Markup Language

Chapter 1

Introduction

In recent years, the commercial building sector consumed about 20% of the total primary energy in the U.S. [2]. As a result, the CO_2 emissions associated with building operations accounted for about 20% of the total [4]. The high contribution to consumption and emissions can be attributed to the fact that 90% of the typical American's daily life occurs inside of a building [1], and buildings require significant amounts of energy to provide the desired comfort and amenities. However, a significant yet avoidable portion of the energy consumption is due to inefficient heating and cooling system operations. The inefficiencies can be attributed to degrading infrastructure, and undetected abnormal conditions. Recognition and remediation of these conditions can reduce energy consumption by 5% to 20%, and even 30% in some buildings [101] which could save about $9 billion in utility costs in the U.S. alone. The aim of this work is to find an effective means to identify opportunities for reducing energy consumption in buildings.

1.1 Problem: HVAC subsystem faults

Heating, ventilating, and air conditioning (HVAC) systems are comprised of sub-systems that work together to provide heating and cooling throughout a building's thermal zones. These sub-systems have multiple components that can fail or degrade over time. Faults such as duct leakage, dampers not working properly, misbalanced airflow, control software programming errors, and valves not closing properly can arise and remain undetected for long periods [101]. Several examples of these type of faults were discovered during a recent upgrade to the University of New Mexico (UNM) Mechanical Engineering Building (MEBldg), which involved the replacement of pneumatic controls with Direct Digital Controls (DDC) at zone terminal boxes. A common issue were actuators, used to modulate air flow dampers, that were stuck or jammed at many of the zone boxes (Figure 1.1). The fault shown in Figure 1.1 went undetected for an undetermined amount of time, probably several years. During that time, there were few or no complaints of uncomfortable temperatures by the building occupants, which indicated that zone temperatures were unaffected by the actuator fault. However, because the actuator was in an uncontrolled state, it most likely demanded more fan power or chilled or hot water flows at the

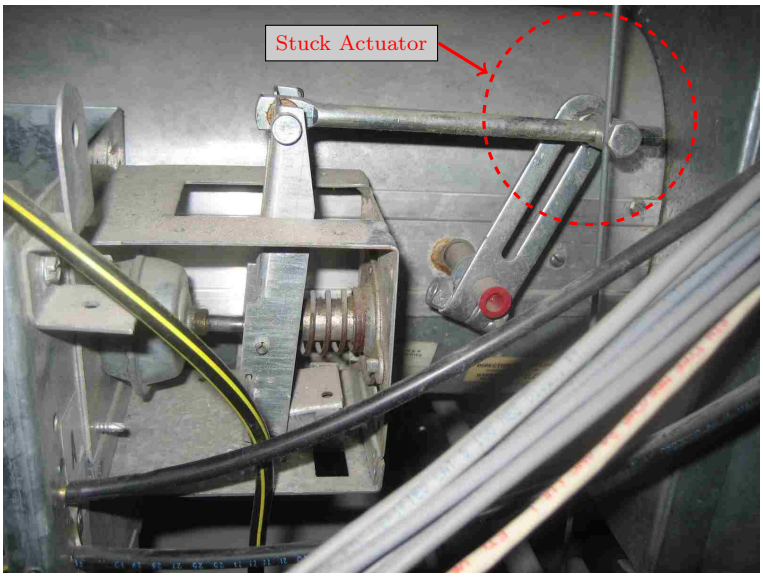

Figure 1.1: Zone box damper actuator that was stuck on a hanger wire that supported the ceiling.

air handling unit (AHU) in order to maintain the zone temperatures.

A fault such as the stuck actuator, shown in Figure 1.1, can have a negative impact on system performance. The impact can be more severe if the fault occurred at a larger system, such as an AHU. AHU systems are prone to inefficiencies and have the potential to remain unnoticed for long periods of time without proper oversight. An engineer or technician who casually observers the system may not recognize these types of faults because parameter thresholds are not exceeded, or occupants are not complaining about uncomfortable indoor temperatures. Furthermore, maintenance staff or building managers may not have the time to visually inspect each component of the entire HVAC system. Instead, they rely on a complete system failure or an occupant complaint before they address an issue, which can lead to waste and inefficiency. A more proactive approach can be implemented that will identify and help remediate issues through **automated** fault detection and diagnostics (FDD). An automated approach can provide valuable information quickly and reliably so that maintenance staff can manage preventative maintenance and repair more effectively.

1.2 Solution: Automated fault detection

A significant portion of the energy waste is due to poorly maintained, degraded, and improperly controlled equipment [58]. These issues can be fixed and/or mitigated by increasing the amount of oversight. Oversight, via a sophisticated FDD platform, can provide valuable information that will improve maintenance activity and control operations. This work proposes a platform that is automated, inexpensive, and self-learning.

The detection of system degradation and even immediate failures can be a tedious and cumbersome process for humans. More specifically, engineers and building managers do not have the time and/or resources to constantly monitor data that

define system performance. Therefore, the only viable option is where the process is performed by a computer where monitoring and evaluations can be automated, intelligent, and not require intensive human interaction.

The ideal platform for automated FDD requires minimal effort to implement and maintain, and at the same time it must provide precise results. To do this, it needs to learn and adapt on its own. The present work hypothesizes that machine learning (ML) algorithms, and in particular Adaptive Resonance Theory (ART) artificial neural network (ANN), can learn and evaluate equipment conditions accurately. ML algorithms have the ability to learn through statistics or with ANN architectures. Statistical learning has been used to interpret, predict, and assess human health issues, stock market prices, hand writing recognition, and perform novelty detection [42]. ANN learning algorithms emulate how animal brains operate [80], and have been used for advanced pattern recognition, classification, and many other applications. The present work investigates the application of ML architectures for FDD of building systems; the algorithms can learn patterns, adapt to normal changes, and detect novelties that would otherwise go unnoticed.

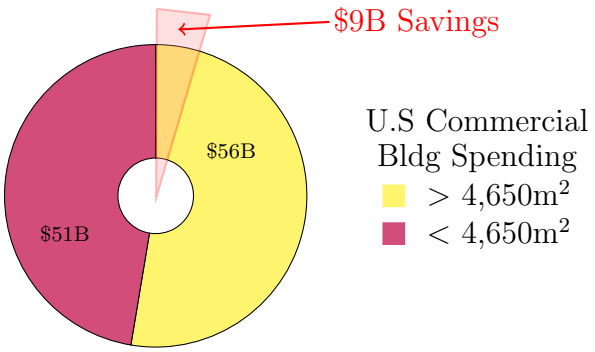

Figure 1.2: The total expenditures for U.S commercial buildings is about $107B. Buildings over 4,600m^2 cost about $56B and under 4,600m^2 are about $51B. The estimated savings is about $9B which is 8.4% of the total.

The implementation of the proposed platform can potentially be achieved at a very large scale. There are about 5.6 million commercial buildings in the U.S., over half of which have an area that is over 4,650m^2 [89]. Moreover, 70% of the buildings

above this threshold are equipped with Building Automation Systems (BAS) [61]. This means that there are existing sensor networks in about 2 million buildings that can be accessed and used for FDD. The proposed platform can access the sensor networks and extract the necessary sensor values with little to no upfront cost. Furthermore, the HVAC systems in the target market, which are buildings that have an area of 4,650m^2 and above, consume about 4.52 quads per year [4]. The savings technical potential ranges from 0.67 to 1.36 quads. This equates to an annual cost savings of about $9 billion.

This potential annual savings is significant. The energy expenditure for commercial buildings is $107 billion. The buildings larger than 4,650m^2 cost $56 billion and the ones that are smaller than 4,650m^2 cost $51 billion for energy as shown graphically in Figure 1.2. The estimated savings of $9 billion is about 8.4% of the total energy expenditures for commercial buildings in the U.S.

1.3 Research Question

The previous section (Section 1.2) introduces the concept of a real-time FDD platform that uses ML algorithms. This concept has been discussed in existing literature which is described in Chapter 2. However, the neural network algorithms known as ART and Lateral Priming ART (LAPART) have not been considered in comparison to other ML tools for HVAC equipment sensor data sets. The ART and LAPART algorithms are expected to perform even better on the time-series data acquired form the HVAC equipment.

In this work, ART and LAPART were applied to HVAC data, in particular data from an AHU. The results from the two algorithms were also compared with the outputs from other techniques. The other techniques included rule-based models, back-propagation (BP) ANN, and support vector machines (SVM). *The question is,*

can the ART and LAPART ANN algorithms detect and diagnose AHU faults with minimal false alarms and false normals? False alarms refer to the case where the tool detected a fault that did not occur. The false detection scenario was when a fault was incorrectly defined.

1.4 Significance of this study

People are prone to catching a cold or the flu; similarly, buildings are susceptible to faults and often operate for many years in an nonoptimal condition. However, people have the ability to monitor themselves in real-time and make adjustments to improve their behavior, health, etc. But, what can buildings do? Buildings do not have a brain that can provide real-time feedback to help them improve their comfort level, and energy use. This study considers this issue, and evaluates ways that could allow for buildings to have a self-checking mechanism that will find and diagnose faults in real-time.

Currently, there is not an effective real-time FDD system that is self-learning and can provide accurate, reliable, and specific information to help manage a building's HVAC systems. It is claimed here that ART and LAPART algorithms can provide a significant impact on building performance. This could be achieved by learning equipment behaviors and then defining its status in real-time. This can provide building managers with a powerful tool that can reduce energy consumption and improve maintenance productivity. This dissertation provides details regarding the data acquisition and storage platform as well as results from experiments that were used to evaluate the performance of ML algorithms.

Existing literature describe ART, LAPART, and other FDD methods individually. Never before have these algorithms been applied and compared with other like methods within an information technologies platform that can evaluate the perfor-

mance of HVAC sub-systems. The platform, developed in the present work, considers the existing building controls network and provides a custom interface for performance review. Additionally, a new product, used to measure the air flow rate was created, tested, and integrated into the existing network that can provide feedback of fan performance. Ultimately, this work presents the structure and implementation procedures for an overall platform that can provide continuous and automated learning for effective FDD of HVAC sub-systems. It also provides a thorough review of ART and LAPART with commonly used methods. The review considered the different algorithm's abilities to detect and diagnose faults found in a very common HVAC sub-system.

1.5 Structure of Dissertation

This paper begins with the introduction (Chapter 1) and literature review (Chapter 2). Next, the methodology describes the experimental set-up (Chapter 3), type of analysis tools (Chapter 4), and experiments (Chapter 5). The first set of results, documented in Chapter 6, describe the outputs from physical and rule-based model calibration experiments. Chapters 7, 8, and 9 describe the outputs from two fault detection and one fault diagnostics experiments respectively. For instance, the ART, LAPART, and other tool's outputs for fault detection are described in Chapter 7. Chapter 8 discuss the outputs from a adaptive fault detection experiment. Chapter 9 described the fault diagnostics test results.

1.5.1 Methodology Overview

The experimental methodology defines the overall set-up of the experiments. It defines the analysis tools used to detect and diagnose faults. In addition, it defines

the experiments for evaluating the effectiveness of each tool on the test apparatus data set.

Set-up (Chapter 3)

The overall experimental set-up is described in Chapter 3 including the test apparatus system, components, and sensors. It also describes the physical model of apparatus used to run the experiments. Finally, the Information Technology (IT) framework was defined to transfer and store data.

Apparatus Components & Sensors (Section 3.1)
The test apparatus for the experiments was a single AHU located in the basement of the UNM MEBldg. It includes mixed air dampers, a chilled water cooling coil, and a centrifugal fan. The unit supplies cold air to cool the second, third, and fourth floors of the building. The AHU, shown in Figure 1.3, has many sensors, including duct pressure, filter pressure drop, and fan speed. Three sensors used in the analysis: (1) supply air flow, (2) supply air temperature, and (3) mixed air temperature. These sensors where picked because they are typical in all commercial AHUs and therefore the framework can be applied to the majority of buildings.

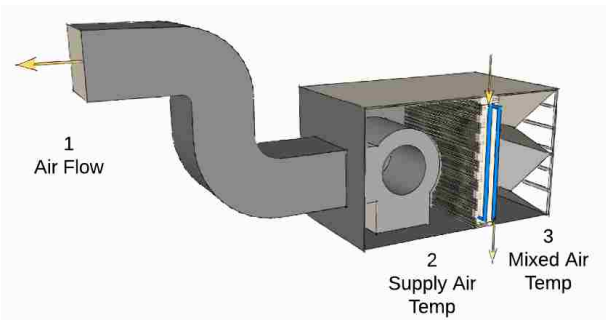

Figure 1.3: AHU 2 was used in this experiment to test the different FDD techniques. Faults will be detected by analyzing the measured air flow, supply air temperature, and mixed air temperature.

Physical Model (Section 3.3)
The physical model, developed in the TRNSYS simulation studio, was used to simulate the AHU system. Model compliance with the actual system was based on ASHRAE calibration standards [6]. Once the model was calibrated simulations were executed to produce data with and without faults. The outputs from these simulations were used to train and then test the FDD tools.

IT Framework (Section 3.2)
The platform presented in the current work required the collection of sensor values from the test apparatus as well as from the building. Therefore, an IT framework was created to collect, route, and store data. Data collection was performed by accessing the existing building controls network. Python and Hypertext Preprocessor (PHP) codes were executed automatically to route the data to a MySQL database. Additionally, the data were available for access and analysis by the FDD tools.

Analysis Tools (Chapter 4)

The FDD tools are defined in Sections 4.1 and 4.2. Section 4.1 defines the algorithms based on ART. Section 4.2 describes other tools also tested in the four experiments.

ART & LAPART (Section 4.1)
In the present work, the fault detection capabilities of ART and LAPART were evaluated. The evaluation process considered the individual performance of each algorithm that are defined in Section 4.1. The work also compared the results from ART and LAPART with other FDD tools.

Non-ART (Section 4.2)
Wolpert and Macready described the *no free lunch* theorem. This theorem states that no one learning algorithm can outperform all others for every problem set [127].

Therefore, the present work introduced other algorithms besides ART and LAPART to compare and contrast results. The other methods include threshold, rule-based model, SVM, and BP neural network.

Experiments (Chapter 5)

Four experiments were performed in the present work. Each of the experiments followed a similar procedure to validate, train, and then test the tool. The tools were presented with data that contained either normal or four types of fault conditions. The experiments ranged from physical and rule-based model validation to fault detection and finally fault diagnostics. The review of results considered the number of false alarms and the precision for each of the tools.

Experiment Procedures (Section 5.1)
The experimental procedures are defined in Section 5.1. The procedures included validation that the model represented the actual well, training, fault detection, and fault diagnostics. Each of the tools were validated through an optimization process that defined it's free parameters. For example, the machine learning algorithms were subjected to a k-fold cross validation process. The tools were then trained by presenting a training data set so that knowledge could be acquired. The tool was then ready to perform fault detection and diagnostics by analyzing previously unseen data samples.

Fault Types (Section 5.2)
Four types of AHU faults were reviewed by the FDD tools. The first fault occurred due to a malfunction in the chilled water system and caused the supply air temperature and flow rate to increase. The second fault was due to a malfunction in the mixed air section. The third was caused by a failure in the fan system and the fan operated at a constant rate. The final fault was due to a control error that caused

the system to operate during off-schedule times.

Experiment Types (Section 5.3-5.6)

The first experiment was the calibration of the physical and rule-based models and is described in Section 5.3. The rule-based model was developed and calibrated to be used as a fault detection tool. The physical model was used in Experiments 2, 3, and 4 as an input to represent normal as well as abnormal behavior. The physical model allowed controlled fault detection experiments to be conducted. The second experiment, defined in Section 5.4, reviewed the situation where two months of training data were collected from a retro-commissioned (Retro-Cx) AHU. The third experiment (Section 5.5) tested the adaptability of ART, LAPART, and SVM. The three tools were retrained with normal data that was statistically different than the original training set. The tools were then introduced to a new testing set that included faults and normal data. In the fourth experiment training data that included normal and fault conditions were presented to the SVM and LAPART algorithms. The training data were labeled as either certain fault or as normal behavior. After knowledge had been acquired from the training process, the two algorithms were then presented with new data, and each performed fault diagnostics (Section 5.6).

Analysis of Results (Section 5.7)

Fault detection can be considered a binary decision problem with a two-class classification. These types of problems have two types of error, which are false positive (false alarm) and false negative (missed detection). To review these errors, a confusion or general loss matrix was created for each FDD method. Also, false positive and true positive rates were plotted in Receiver Operator Characteristic (ROC) space. The fourth experiment, in which a multi-class classification was performed, the F_1 score statistic was used to evaluate performance.

Chapter 2

Literature Review

FDD methods have been applied to many different fields. Katipamula *et al.* performed an exhaustive assessment of current techniques [58]. Their research article described fundamental FDD methods and defined them as either quantitative, qualitative, or process history based. The three approaches are similar in that they involve the use of models and data for the development of their general structures. However, knowledge is acquired differently by each of the methods. For example, quantitative and qualitative FDD models used priori to gain knowledge. This means that knowledge could be gained through a deduction, and not through empirical evidence. The process history method can be used to acquire knowledge through the presentation of past data. The three methods, quantitative, qualitative, and process history, can be applied to FDD for commercial building HVAC systems.

Katipamula *et al.* followed the initial review of FDD with a second part that concentrated on HVAC systems, such as refrigerators, air conditioners, heat pumps, chillers, and AHUs [59]. In the paper, various methods for FDD of AHU performance including parametric models [87], autoregressive algorithms [96, 132, 133], back-propagation ANN [69, 83], classification algorithms [51], and a simplified physical

model [102] were described. However, ART and LAPART were not discussed in the review article for FDD.

Venkatasubramanian *et al.* published a three-part review of FDD for process control applications that was similar to the work performed by Katipamula *et al.*. In the first part, research was discussed that applied quantitative FDD methods that included analytical redundancy and residual generation in dynamic systems [119]. Qualitative model-based methods that were typically developed through a fundamental understanding of the physics [117] were discussed in the second part. Part three focused on process history methods that were based on process knowledge [118]. The review categorized the process history methods based on quantitative or qualitative feature extraction. The quantitative feature extraction section included a discussion of expert systems and trend modeling approaches. It also discussed the qualitative process history method that applied statistical feature extraction from the process data and used neural network pattern recognition.

In the third review paper, Venkatasubramanian *et al.* discussed the application of various ANN for FDD [118]. It was evident that, to date, the most popular supervised learning algorithm for FDD has been the back-propagation neural network. Other methods such as radial basis function ANN [20], self-organizing maps [64], K-means clustering [57], and ART networks [11] have also been implemented. Venkatasubramanian *et al.* did not cite any papers in which the ART algorithm was implemented on HVAC systems, but did provide an example where the ART2 algorithm was used to interpret sensor data for chemical processing [124].

The application of the ART algorithm can also be found in another review paper by Markou and Singh that focuses on novelty detection using neural network based approaches [77]. In the review paper, multiple ANN including the multi-layer perceptron, SVM, ART, radial basis function, auto-associator, hopefield, oscillatory, self-organizing maps, habituation based, and a neural tree were considered. It was

found that the ART algorithm can outperform other classifiers and cites a paper where ART2-A was used to classify military target locations [84]. In addition to the ART2, Markou and Singh reference papers in which the ARTMAP neural network was used to perform familiarity discrimination and tested the application on a simulated radar target recognition task [17, 18]. However, none of the papers cited by Markou and Singh considered the ART or LAPART algorithm for FDD of HVAC sub-systems.

2.1 Qualitative Method

The qualitative method for FDD has been used for simple evaluations that consider basic descriptions of system performance. The evaluations involve the review of system status, such as equipment on or off, and has not been used to provide a numerical analysis of the system's sensor data. A common type of qualitative analysis is the threshold method. This method is based on a concept where a magnitude must be exceeded for a fault to be recognized. This approach has been used to monitor an AHU's fan. In this case, the fault detection analysis would consider the actual fan status in comparison to the pre-set desired status. If the actual status deviated from the desired then a fault was detected.

The threshold method used in the present work was based on the alarm system embedded in the existing BAS. No literature was found in support of this type of FDD approach despite the fact that it is the most common FDD approach used by practitioners. However, other qualitative approaches for assessing AHU faults have been implemented. For example, a trend analysis-based framework for incipient faults was discussed by Maurya *et al.* [79]. Also, the implementation of a sign function that returns a '+' or '-' from the subtraction of the actual value minus a nominal value to define normal or abnormal behavior was integrated into an AHU system [110].

Similarly, Glass *et al.* used a qualitative model to detect faults in an AHU [37]. In this approach qualitative states that were compared with actual values were considered. In another AHU application the first principle analysis to detect faults was used by Norford et al. [88].

2.2 Quantitative Method

The quantitative approach, which includes physical and rule-based models, has been documented in past literature. For instance, Schein *et al.* implemented a rule-based model to monitor and detect faults in an AHU [103]. Physical models have also been implemented to monitor and assess whole building performance [123, 93, 94]. Further studies by O'Neil *et al.* illustrate a real-time physical model approach [90] for whole building analysis. The faults were then determined based on a statistical review of the deviation between the model and actual sensor values.

2.3 Process History Method

In addition to the threshold, rule-based, and physical model approaches, multiple process history methods were implemented in the present work. Statistical regression models, SVMs, and back-propagation algorithms have been used in the literature to monitor and detect faults in HVAC systems. The ART algorithm has been used in past work as a novelty detection algorithm, however it has not been applied to data sets that originate from sensors in HVAC systems. Similarly, the LAPART algorithm has been used to provide predictions, but has not yet been used to detect faults in the sub-systems similar to the one presented in this paper.

2.3.1 Statistical Regression

Regression techniques can be used to detect novelties within data sets. This was accomplished in the present work by detecting outliers and labeling them as faults. Outliers are observations that are inconsistent with the remainder of the data [78]. The method depends on the absence of outliers in the initial training set so that the model can best fit normal behavior [49]. Worden *et al.* considered this and implemented a regression novelty detection approach to test univariate and multivariate data. The test was based on deviation statistics where the potential outlier would exist at a distance that exceeded the threshold defined by the mean and standard deviation [128].

The statistical regression framework has been applied to fault detection methods in HVAC systems. For example, a model based FDD approach used principal component analysis for fault detection and a regression model as a benchmark or reference model for validation of HVAC system operations during the fault detection process [23, 120]. It has also been directly applied to the identification of faults. Radhakrishnan *et al.* implemented a two step approach for fault detection of HVAC equipment that began with the creation of polynomial and locally weighted regression models to represent actual operations. The second step considered regularities in computed deviations (residuals) from normal behavior in order to detect faults [98].

2.3.2 Support Vector Machines

The SVM is a statistical machine learning method that learns by mapping training data into a high dimensional feature space [22, 113, 114]. The general form of the SVM is a two class classification algorithm that performs supervised learning. This approach has been applied in many instances to detect faults in various HVAC systems [30, 71]. For instance, Dehestani *et al.* applied this type of SVM to an HVAC

system where faults were detected and new SVM classes could be created as more types of faults were found [24]. Other types of SVM algorithms can also be applied to detect faults in HVAC systems, such as SVM for regression and one-class SVM.

The SVM for regression can provide a prediction to model actual system operations instead of classifying new unseen variables. This algorithm was used to model and optimize HVAC operations by Kusiak *et al.* [66]. Similarly, this algorithm has been used to forecast hourly building cooling loads [70]. This approach was also successfully applied to novelty detection on temporal sequences by Ma and Perkins [72, 73] to discover novelties in time-series data. The present work implemented a one-class SVM for fault detection, and a multi-class for fault diagnostics.

The one-class SVM described by Schölkopf *et al.* can perform unsupervised learning on a single class of data. It can then be applied to previously unseen data and flag abnormal behavior [104, 105]. This type of approach is not documented well in existing literature for HVAC systems. However, it provides significant advantages for quick and potentially accurate implementation of a fault detection tool on HVAC equipment. This is especially helpful when only normal behavior is known, and therefore multiple classes cannot be learned before actual faults occur.

The multi-class SVM is a supervised learning algorithm that can assign each observation into one of k classes [129]. In this case, the SVM algorithm was implemented using Python Sklearn [5]. The Sklearn package implement the "one-against-one" approach for multi-class classification [63]. A two-class SVM was used to perform fault diagnosis for HVAC chillers [21]. In another application, a multi-class SVM was used to diagnose faults in induction motors and other machine tools [125]. The "one-against-one" multi-class SVM was also applied to HVAC chiller data sets to perform fault detection and diagnostics [41].

2.3.3 Back-propagation

The BP algorithm is a common form of the multi-layer perceptron. Early development of the algorithm was performed by Werbos and documented in his doctoral work at Harvard University [122]. According to Venkatasubramanian *et al.* it is known as one of the most popular supervised learning ANN [118]. For example, there were many papers published in the early 1990s that used the algorithm for FDD in chemical engineering processing [50, 112]. The algorithm has also been applied to HVAC systems, and often used to forecast building loads [60, 68]. The algorithm has also been used to perform FDD.

The BP algorithm has been used for novelty detection using parametric statistics and residual analysis. For example, Markou *et al.* described studies where parametric statistics were applied to the neural network to perform novelty detection. The basic premise was that the network could discriminate between classes that had different distributions because the error calculation that drives the neural network were significantly different [77]. In the present work, the FDD results were based on a residual analysis.

Residual analysis is the difference between the prediction produced by the algorithm and the actual values. This type of approach was used by Lee *et al.* to identify faults in an AHU. Lee *et al.* implemented an approach that depended on the ability to identify patterns in the residuals between the actual and set-point values [69]. A similar study considered the back-propagation algorithm for an HVAC variable air volume system to detect sensor and coil fouling faults [83].

2.3.4 Fuzzy Adaptive Resonance Theory

The Fuzzy ART algorithm used in the present work is one of the most recent versions of the ART algorithm. Earlier versions include ART1, ART2, ART2A, and ART3. Fuzzy logic was then integrated into the ART algorithm to create the Fuzzy ART algorithm. The new algorithm increased the algorithms ability to perform pattern recognition and also improved the generalization of the algorithm. The algorithm is able to use a match-based learning technique to process input data that creates and stores memories [78]. The learning process is performed in an unsupervised manner, and at each input presentation the network searches for a category that it can join, which is referred to as resonance. If the strength of the response from each category is low, then a new category is created. In this case, the input that does not resonate with an existing category could be considered novel and thus a fault.

The approach has been discussed in past literature for novelty detection. The inventors of the algorithm, Carpenter and Grossberg, discussed the potential for novelty detection in a 1988 paper [11]. Caudell and Newman [19] implemented the ART1 algorithm to define normal and abnormal behavior in time-series data. Similarly, the Fuzzy ART algorithm was used successfully in anomaly detection for simulated time-series data [7]. The same algorithm was also applied to anomaly detection in wireless sensor networks [130].

The Fuzzy ART algorithm has also been applied to machines and power systems. For instance, fault diagnostics on a rotating machine was documented by Yang and An [131]. It was also applied to power systems for fault detection analysis in a 2005 paper [115]. The closest application of the Fuzzy ART algorithm to HVAC equipment was a solar hot water system. The algorithm was used to perform real-time fault detection in solar hot water systems [44, 45]. The existing literature does not contain research that applied the ART algorithm for fault detection in commercial building

HVAC systems.

2.3.5 Lateral Priming ART

The LAPART algorithm was introduced by Healy and Caudell for logical inference and supervised learning [48]. The algorithm can be used as a prediction tool and has been shown to provide accurate weather forecasts [47, 107]. It has also been applied successfully to solar micro-forecasting to predict solar irradiance at small time steps [74]. Additionally, it has the capability to learn and monitor operations in an on-line application without impacting the machine operations [106]. Similar to the case of the single ART algorithm, existing literature does not include investigations of the LAPART algorithm for FDD in commercial building HVAC systems.

Chapter 3

Methodology: Set-Up

The reduction in building energy use through the identification of system failures requires precise and timely decisions. The decisions relate to proper control sequencing and maintenance deployment. However, optimal decision making depends on extensive knowledge of system behavior and performance. Learning to obtain this knowledge at a granular level can be difficult. Albert Einstein once said, "The only thing that interferes with my *learning* is my *education*". This infers that the amount of knowledge gained depends on the efficiency of the learning mechanism to process the education. This is shown in Figure 3.1 where education, which is defined as the transfer of information or data, is an input into a learning mechanism or algorithm. The learning mechanism processes the data and acquires knowledge. Unfortunately, there are losses in the acquisition of knowledge that vary depending on the type of learner. Therefore, the intent of

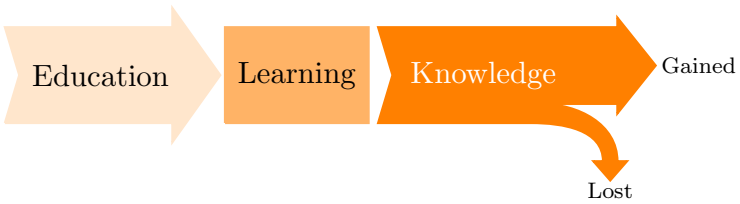

Figure 3.1: Transfer of data into a learning mechanism that acquires knowledge. Not all knowledge can be gained and some is lost due to the inability of specific algorithms.

this study was to evaluate the learning abilities of ART and LAPART in comparison with other methods.

The consideration of other methods is important, because model based FDD is not a new concept and has been discussed widely in literature for the past 30 years. For instance, Rabel *et al.* were concerned about energy use in buildings and described mathematical models to evaluate performance in a 1988 paper [97]. In 1984, Isermann [54] performed fault detection on a centrifugal pump system using parameter estimation methods. ANN were also used for FDD in non-HVAC systems in the 1980s.

ANN were used to diagnose incipient faults within chemical processes and results were documented in a paper by Watanabe *et al.* [121] in 1989. A neural network for FDD was applied to process engineering in the 1980s and was documented by Venkatasubramanian *et al.* [116]. Fan *et al.* [31] used a back-propagation algorithm for fault detection in a 1993 paper that again was focused on chemical processes. It is evident that most of the early work with artificial neural networks analyzed data sets from chemical and oil refining processes, and left HVAC systems untouched. In recent years, ANNs were used in several research projects to analyze HVAC systems. Yet, the capabilities of ART and LAPART to learn system behaviors and detect abnormalities in HVAC systems has not been documented extensively.

The overall approach to test and evaluate the FDD methods in the present work required the development of a platform that is described in Figure 3.2. The platform was designed to transfer data from a building to analysis tools and then to maintenance technicians. A physical test apparatus was defined and instrumented with sensors. This apparatus is an AHU that is described in Section 3.1. The sensor data values of the test apparatus could be sent through an existing building control network system that complied with the Building Automation Control Network (BACnet) protocol. This IT infrastructure also included an XML web-service and

a MySQL database that is described in Section 3.2. Qualitative, quantitative, and process history FDD tools, described in Chapter 4, were used to analyze the AHU data in each of the experiments.

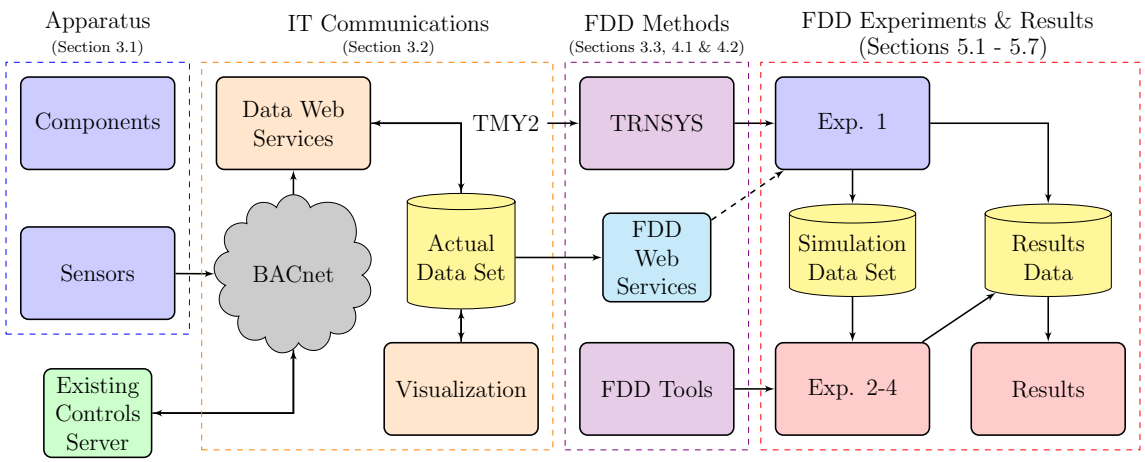

Figure 3.2: The flow diagram of platform for FDD experiments includes the apparatus, IT communications, FDD methods, experiments and results.

The first experiment was the calibration of the rule-based and physical model. The physical model was used to produce training and testing data for experiments 2 through 4. The rule-based model was used for fault detection (FD). In the second experiment, FD methods were evaluated on their abilities to learn with a minimal amount of training data from the TRNSYS physical model that contained no faults. This experiment emulated the situation where the data was acquired from a Retro-Cx effort. The adaptability of the top performing tools from experiment two were evaluated in the third experiment. In this experiment normal changes to non-fault data were presented to the algorithms to identify how well the tools could adapt and still detect faults. Fault diagnostics using LAPART and multi-class SVM were tested in the fourth experiment. The results from the fault detection experiments were evaluated and compared based on the number false alarms, false detections, true detections, and true faults. The fault diagnostic results were based on the F_1 statistic score described in Section 5.7.

3.1 Apparatus

The ability to conduct this research required a physical testbed with two main characteristics. First, it needed to be flexible so that it could simulate a variety of different operations scenarios. Second, it needed to be appropriately instrumented to allow the user to retrieve performance results through standard sensor configurations. The facility that was used to test the fault detection and diagnostics tools discussed here was the University of New Mexico's Building Energy Retrofit Testbed (B.E.R.T.), a facility integrated with the MEBldg.

The building has a total of four floors and which add up to about 6,503m^2 of office, classroom, laboratory, and common area space. In this research a single air handling

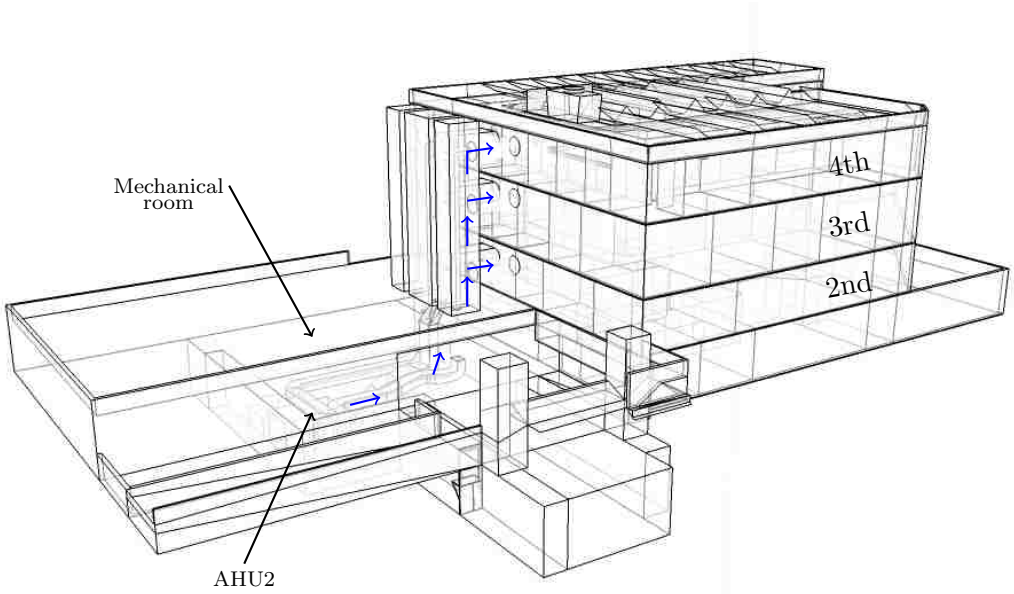

Figure 3.3: The Building Energy Retrofit Testbed (B.E.R.T.) is a four story 6,503m^2 building for lab, classroom, and office space. AHU2 forces cold air from the basement to the 2nd, 3rd, and 4th floors.

unit, **AHU 2**, provides cold air to approximately 46 zones, or about 3,360m^2. These zones are located on the second, third, and fourth floors as shown by the arrows in Figure 3.3.

The building was originally constructed in 1980 as a living laboratory for building energy system research. In the period following commissioning the building performed extremely well, with an energy consumption less than one third of comparable buildings [126]. In the years following its construction, energy prices started to fall, and interest (and funding) in improving energy efficiency in buildings decreased. As a consequence, the building's advanced energy systems could not be maintained properly and quickly fell into disrepair. This highlighted the fragility of complex systems that have demanding control requirements.

Following renewed interest in the field of building energy management, a program to refurbish and modernize the building's energy systems began in 2006 [75, 91]. First, the solar thermal and energy storage systems were upgraded and modernized. Second, the original pneumatic control system at the AHU level was replaced with DDC. All significant pumps and fan motors were upgraded with variable frequency drives. To finalize the upgrade of the controls, zone terminal boxes that AHU 2 and others support were upgraded from pneumatic to DDC controls.

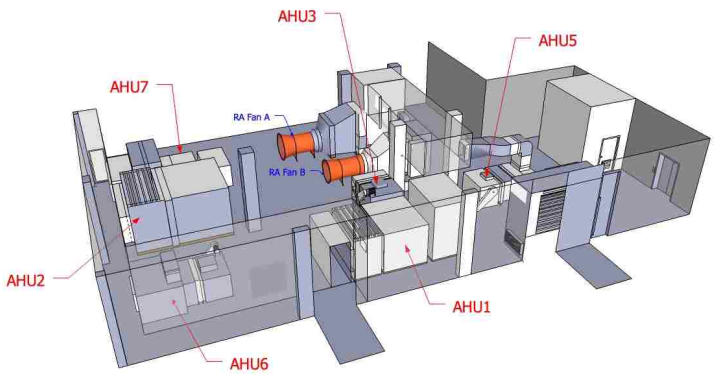

Figure 3.4: Representation of the six AHUs in the basement, including AHU 2, and the two return air fans.

The HVAC equipment is located in the mechanical room of the MEBldg and is shown graphically in Figure 3.4. The chilled and hot water piping, duct work, pumps, heat exchangers, etc. are not shown for clarity. After the most recent upgrades all of the equipment is now controlled with a modern Delta Controls DDC system that uses multiple Delta DSC-1616 control boards.

This system is native BACnet and communicates on twisted pair ethernet 10-Based T through both BACnet over Ethernet and IP. The system can be viewed on a Windows 2008 Server through the Delta ORCAView 3.40 system, and programmers can program equipment controls through the General Control Language Plus (GCL+) language.

The Delta Control system has proven to be scalable and user friendly. Currently there are approximately 400 data points that are being monitored and used in control sequences. These points include fan speeds, zone temperatures, AHU temperatures,

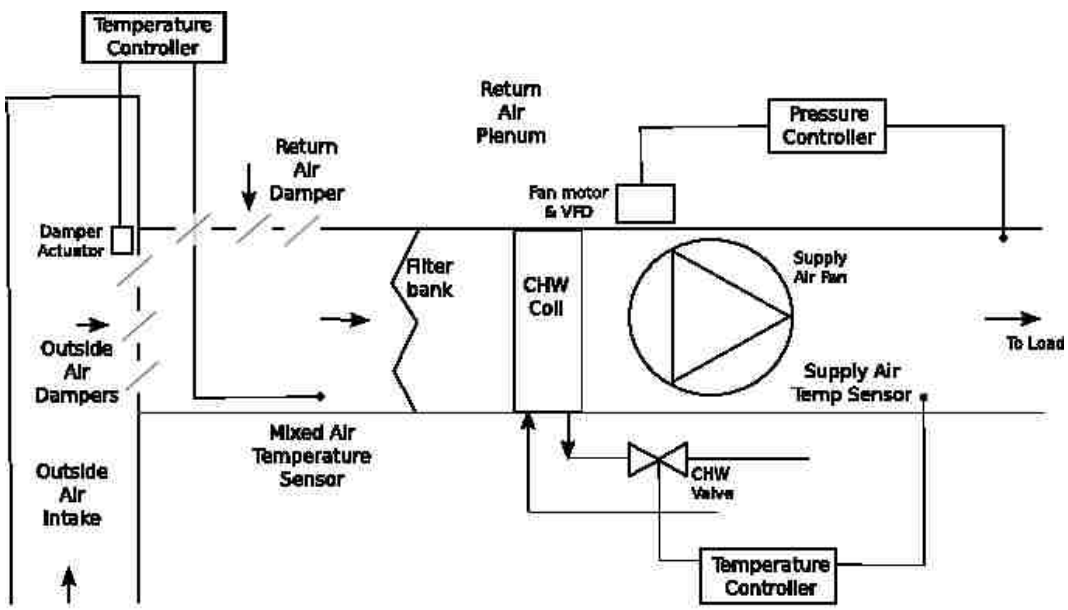

Figure 3.5: AHU2 is comprised of a damper system, chilled water coils, and a supply air fan that are monitored with mixed air temperature sensor, supply air temperature sensor, and a supply air flow sensor respectively.

and air flows. The control system also collects weather data such as solar irradiance, outside air temperature, and relative humidity.

AHU 2 is highly instrumented with sensors and controls. Its general layout is described in Figure 3.5. The system is set up in a draw thru configuration. At the the unit entrance is a damper section where parallel bladed mixing dampers control

the proportion of return and outside air that enters the system. The dampers were designed to have one percent leakage at 746Pa (3.0 In. W.G.). The unit is for cooling only and has two stacked chilled water (CHW) coils. At the exit of the unit is a centrifugal fan with a 91.4 cm fan wheel. The fan draws air from the mixed air section through a filter bank and the CHW coils and then on to the zone boxes in the building. The sheetmetal enclosure that surrounds the coil and fan section includes a 2.54 cm, 0.34 kg density neoprene coated insulation. For the purpose of this work the unit was divided into three sections or sub-systems: (1) damper, (2) heat exchanger (HX), and (3) supply fan.

Damper

Section	Area (m^2)
Outside Air	4.4
Return Air	4.4

Heat Exchanger (HX)

-	Values
Supply Temp	11.7°C
X-Section	6.73m^2
Cool Cap.	276 kW_{th}

Supply Fan

-	Values
Supply Flow	1,030 $\frac{m^3}{min}$
Static Press.	836Pa
Fan Speed	1,131rpm
Motor	21.9 kW

Figure 3.6: AHU 2 specifications for each sub-system from the original construction documents published in 1980.

The sub-systems, delineated by dashed lines shown in Figure 3.5, are defined to simplify the analysis performed in each experiment. These sub-systems each have a single sensor point whose reading was monitored and evaluated by the FD tools. The damper and HX sub-system are each monitored by their own temperature sensor. The supply fan is monitored by a supply air flow sensor. These sensors are typical for most AHU and therefore can be replicated in many HVAC systems across the country and probably the world.

The damper sub-system is comprised of two sets of parallel bladed dampers that

regulate the flow from return and outside air. These dampers, each with an area of 4.4m^2, are controlled with actuators that modulate based on outside and mixed air temperatures. The heat exchanger system has two critical components, namely CHW coils and a modulating valve. The coils act as the heat exchange mechanism to cool the air to a temperature of about 15.5°C. The air is then forced to the various building zones. The valve modulates based on a signal from the controller that considers the supply air temperature and compares it with the predefined set-point. According to the original specification, described in Figure 3.6, the system is capable of providing 276kW$_{th}$ of thermal power. Lastly, the supply air fan is composed of a centrifugal fan and motor. The motor, which powers the fan, is controlled by a Variable Frequency Drive (VFD) that modulates the fan speed based on the static pressure in the duct work. The construction documents state that the fan is powered by a 29.4kW motor and can supply a volumetric flow rate of 1,030m^3/min (17.16m^3/sec).

Figure 3.7: Outside air damper system that has a mechanical actuator controlled by the DDC.

3.1.1 Components

The three sub-systems within AHU 2 that were defined for this work are described in Figure 3.5. Each of these sub-systems components has the potential to fail and disrupt the entire AHU system which can cause it to shut down or work harder than necessary. For instance, the damper sub-system may have an actuator malfunction; the HX modulating valve may be controlled improperly; the fan VFD may fail and

supply a constant or an inadequate volume of air.

Dampers

The damper sub-system, shown in Figure 3.7, consists of a set of dampers for outside and return air. The figure shows the outside air dampers in the foreground, and the filters in the background. The return air dampers are arranged in a similar manner but in a horizontal plane on the top of the unit. The space in between the dampers and the air filter bank is considered the mixed air section. In this section, the outside and return air flows are controlled so that the mixed air temperature are as close as possible to the defined set-point.

Figure 3.8: Example actuator used to modulate the damper blades.

The return and outside air dampers both operate in a similar fashion. A mechanical actuator, similar to the one shown in Figure 3.8, receives a signal from the controls and modulates the dampers by actuating a single bar that is connected to the damper sections through a series of cords and hinges. The controls are currently set up to modulate the two damper systems in an opposing manner. This means that when the damper position for the outside air is x% then the return damper is (100-x)%. Typically, about 10-15% of the total mixed air should be fresh, outside air. Yet, in this system the minimum position of the outside air damper is 0%. This means that the damper completely closes when the outside air temperature is greater than the return air temperature.

Chilled Water Coils & Valve

The heat exchanger sub-system is comprised of two stacked CHW coils, represented in the Figure 3.9 graphic, and a modulating valve that is shown Figure 3.11. The

piping to and from the stacked coils are in direct return arrangement. In this arrangement the supply and return flows for the top coil must each pass through a T fitting. In contrast, the bottom coil supply and return flows must each pass through a L fitting. Typically, this arrangement results in reduced flow at the top coil because the system is unbalanced, and therefore does not provide for an optimal heat exchange [52]. This inefficiency was observed in this system. Temperature measurements were taken at the air flow inlet and outlet of the top and bottom coils. The difference in air temperature across the top coil was lower than the bottom coil by 1.5-2.8°C during normal operating hours.

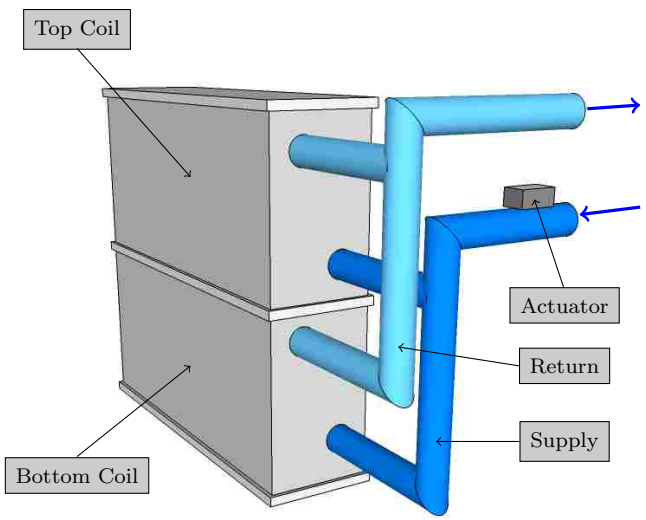

Figure 3.9: AHU2 stacked CHW coils with direct return piping and a modulating valve that can vary the flow rate.

According the original design documents the CHW coils, shown in Figure 3.10, have an area of 6.73m^2 and were specified to provide 273kW$_{th}$ (78 Tons) thermal power at a specified flow rate of 0.355m^3/min (94 gallons/min). The thermal power defined by the construction documents is very large in comparison to actual observed values of between 100 and 130 kW$_{th}$ measured in the air distribution system during the summer months. This could be because the system was originally oversized.

The second major component in the heat exchanger sub-system is the CHW valve. This valve, pictured in Figure 3.11, receives a signal from the control system to regulate the amount of CHW flow. The signal is based on the output of a Pro-

portional Integral Control (PID) control algorithm. This algorithm uses the supply air temperature as the control variable and calculates a signal based on the error between the measured value and the set-point of 15.5°C. The set-point was established by the controls programming and does not match with the specified temperature of 11.7°C stated in the original design documents.

Figure 3.10: CHW cooling coils that transfer heat from the air to the water.

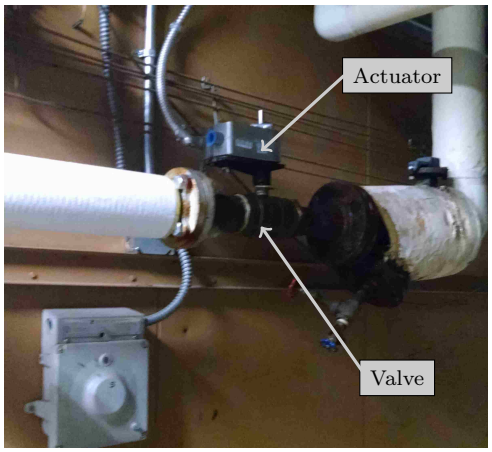

Figure 3.11: CHW valve and actuator that modulates the flow through the coils

Supply Fan

The third sub-system in the AHU is the supply air fan. Its components are described graphically in Figure 3.12. It includes a centrifugal fan, fan motor, fan belt, and VFD. The unit supplies air to about 46 thermal zones in the building and has a maximum flow rate of about 1,030m^3/min (36,520 ft^3/min, 17.17m^3/sec). The entire system is dynamic and is comprised of variable air volume (VAV) boxes at each of the thermal zones. This means that there are multiple dampers throughout the building that make adjustments to the amount of air that is supplied to the space depending on the local thermal loads.

The fan system works to supply the correct amount of air at the temperature

achieved by the CHW coils so that the thermal zone loads are met, and desired zone temperatures are maintained. The load ($\dot{Q}$) supplied by the AHU can be calculated using Equation 3.1:

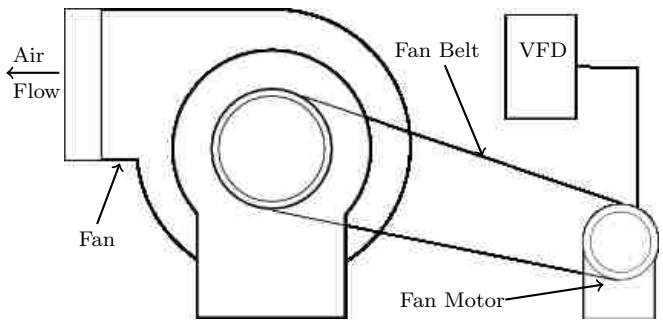

Figure 3.12: Diagram of fan system that includes a centrifugal fan, belt, and a motor.

$$\dot{Q} = \dot{m} C_p \Delta T, \quad (3.1)$$

where $\dot{m}$ (kg/s) is the mass flow rate, C_p (kJ/kg°C) is the specific heat of air, and ΔT ($^\circ C$) is the difference in temperature between the supply and zone air. In order to supply the correct amount of air, the fan requires electrical power that varies proportionally to the cube of the fan speed as shown in Equation 3.2:

$$\frac{P_1}{P_2} = \left(\frac{\Omega_1}{\Omega_2}\right)^3 \quad (3.2)$$

The changes in the VAV terminal unit damper not only alter the amount of air delivered to the zone but also impact the air distribution system resistance curves. The system curve, shown in Figure 3.13, is a second order equation where the static pressure is equal to a constant times the square of the air flow. The constant value, which is a function of the variable air volume positions at each of the zones, represents the overall resistance of the system and determines the steepness of the curve. The two curves shown in the figure intersect at point 1, and is considered the initial operating point.

At operating point 1 the fan demands about 11.2kW of electricity to supply about 13.7m^3/s of air. In this situation the AHU is providing about 125kJ/s (125kW$_{th}$) of thermal power according to Equation 3.3:

$$Q_1 = \dot{m} C_p \Delta T = (16.48\frac{kg}{s})(1.01\frac{kJ}{kg^\circ C})(23 - 15.5)^\circ C = 124.83\frac{kJ}{s} \quad (3.3)$$

Next consider a change in the building's thermal load to about 82kW$_{th}$. The zone dampers would react and restrict the flow. Then, the required flow rate provided by the fan would decrease, and the system resistance and fan curve would adjust to a lower static pressure at point 2 where the flow rate was about 9m^3/sec. In this situation the fan power was estimated using Equation 3.2 and decreased to about 3.2kW.

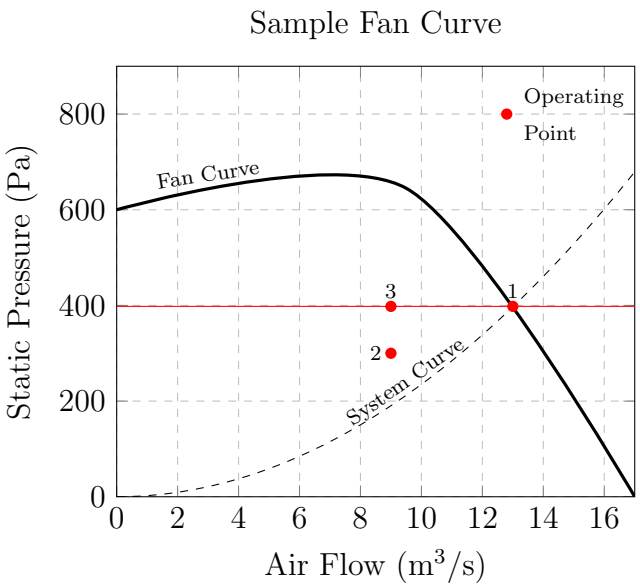

Figure 3.13: Example fan curve shows that the operating points at a constant pressure

However, the current control system does not allow the fan to operate at a static pressure below 398Pa (1.6inH$_2$O). This is because the VFD signal is based on a PID loop that maintains a measured static pressure so that it is equal to the pre-set set-point. Therefore, the VFD would increase the fan speed to force the fan to operating point 3, thus increasing the fan power to 8.1kW. This example highlights the correlation between fan power and zone thermal loads. It also shows that slight variations in the controls and system dynamics have the potential to negatively affect energy consumption.

3.1.2 Sensors

Commercial building HVAC systems require many sensors to monitor and control equipment. Standard sensors include temperature, pressure, air flow, motor status, and fan speed. Unfortunately, there is not a set standard for the installation of

sensors in HVAC equipment such as AHUs. However, AHU are typically equipped with temperature sensors in the mixed and supply air sections. It is also common to have a supply air flow rate sensor to monitor the total volumetric flow provided by the fan.

The present work collected actual data from the mixed air temperature sensor, supply air temperature sensor, and supply flow rate sensor. The collection process was conducted through the extraction of sensor values available in the existing BAS network. This extraction occurred every five minutes. PHP code was used to parse the sensor values and then insert each value according to time in an off-site database. This process is defined in more detail in Section 3.2.

Air Temperature Sensors

The Dwyer Series averaging Platinum resistance temperature detector (RTD) was used in this work to measure the supply and mixed air temperatures. This sensor is able to measure temperature by correlating the resistance of the RTD element with temperature. It is capable of measuring temperatures from -32 to 240°F (-35.5 to 115.5°C). The sensor has an accuracy of ±0.6°F at 32°F (0°C). The sensor is 12 feet long and was placed in a zig-zag fashion at two locations. One at the entrance of the CHW coil to monitor the mixed air temp. The other at the exit of the coil to measure the supply air temperature.

Each of the temperature sensors was subjected to a calibration process. The process involved the comparison the sensor output with a control thermometer that had been calibrated by the manufacturer. In addition, the comparisons were conducted over a temperature range of 5°C. The two sensors provided outputs that matched well with the control and therefore did not require any modifications.

Air Flow Sensor

There are a several different techniques for measuring air flow, including a hot wire anemometer, digital anemometer, and pitot tubes. Pitot tubes are an effective way to measure flow [62], but unfortunately a pitot tube and an associated pressure

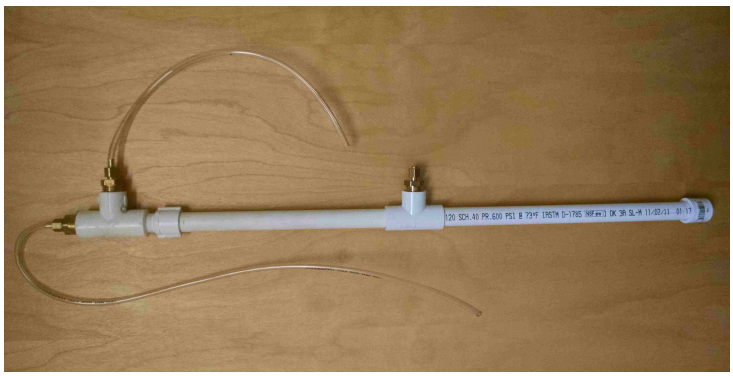

Figure 3.14: Pitot tube air flow sensor that is made of PVC, brass fittings, and poly-tubing

Figure 3.15: Dwyer pressure transducer

transducer can cost between $250 to $1,000. This cost can be prohibitive, so in the present work, a pitot tube system that cost about $100 for both the pitot tube and the pressure transducer was devised.

This pitot tube, made out of PVC fittings and pipes, brass fittings, and poly-tubing is shown in Figure 3.14. The two small tubes at the end are the total and static pressure connections. These tubes connect to the pressure transducer shown in Figure 3.15. The pressure transducer accepts the total and static pressure connections and measures the velocity pressure. It then used the velocity pressure to calculate the velocity of the air. From there, the air speed was mapped to a 0-10 volt signal and sent to the BAS for data collection.

The pitot tube elements are shown in Figure 3.16. The device has a single static pressure port that is labeled in the *Side View*. This static port is connected to an external static pressure connection that is also labeled in *Side View*. This connection is made with an internal tube as shown graphically in the *Side Cut View*. The total

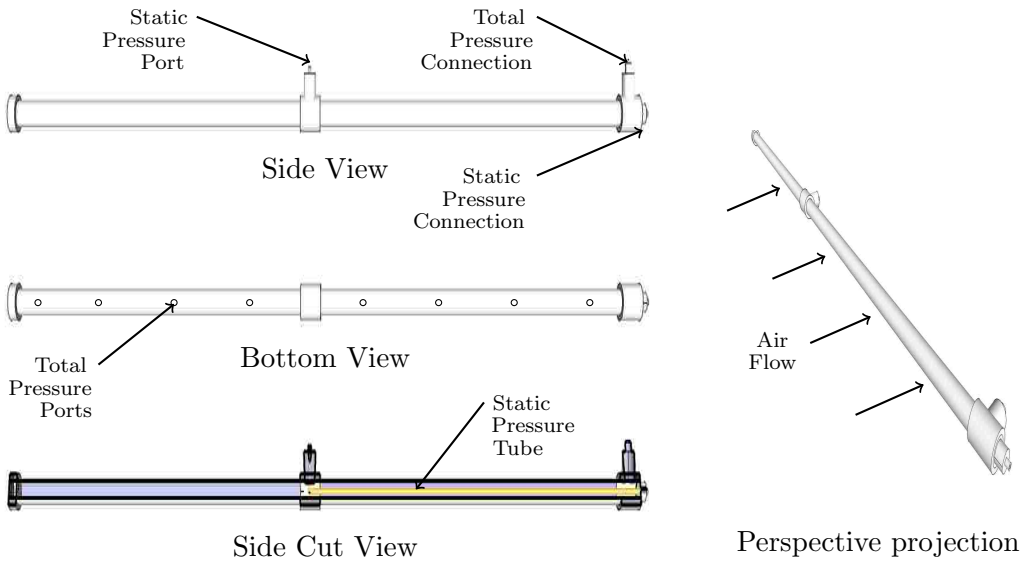

Figure 3.16: Pitot tube design described with side, bottom, and cut views

pressure ports are distributed on the bottom of the device as shown in the *Bottom View*. The ports were distributed across the PVC pipe and were drilled using a 3 mm bit. The connection to the transducer for the total pressure is made at the total pressure connection shown in *Side View*. The diagram on the right side of Figure 3.16 shows the general orientation of the device in relation to the air flow. The ports drilled into the PVC pipe are oriented into the flow stream to measure the total pressure, and the static pressure port is pointing directly away from the flow stream.

The basic principle that defines the pitot tube is based on Bernoulli's equation:

$$p + \frac{1}{2}\rho\nu^2 + \rho g z = C \tag{3.4}$$

In this equation p is pressure, ν is velocity, ρ is specific density, z is the elevation above a reference, and C is a constant. The flow before and after striking the tube can be represented using Equation 3.5:

$$p_1 + \frac{1}{2}\rho\nu_1^2 + \rho g z_1 = p_2 + \frac{1}{2}\rho\nu^2 + \rho g z_2 \tag{3.5}$$

The total pressure ports of the pitot-tube are set perpendicular to the air flow stream. When the air hits the horizontal surface it's speed reduces to zero and ν_2 is set to 0. Additionally, the relative elevation for both sides of the Equation 3.5 are the same and cancel each other out, resulting in Equation 3.6:

$$p_1 + \frac{1}{2}\rho\nu_1^2 = p_2 \tag{3.6}$$

In practice, p_1 is called the static pressure and p_2 is considered the total pressure. The total pressure is defined by the sum of the velocity and static pressure. The static pressure is the pressure that exists inside the duct and is independent of the flow. Furthermore, pressures p_1 and p_2 are measured by the pressure transducer. Once measured these pressures can be used to solve for the velocity of the flow using Equation 3.7:

$$\nu_1 = \sqrt{2\left(\frac{p_2 - p_1}{\rho}\right)} \tag{3.7}$$

The pitot tube was tested against a commercially available flow measurement device. The tests were conducted in a controlled environment and flow was modulated to observe how the pitot tubes performed in relation to the control flow measurement device at different flow rates. The expectation was that the low-cost flow device would remain within the error of the control.

The results from the validation test are shown in Figure 3.17. The graph shows the velocity reading of the low-cost pitot tube versus the results of the control devise. It is evident

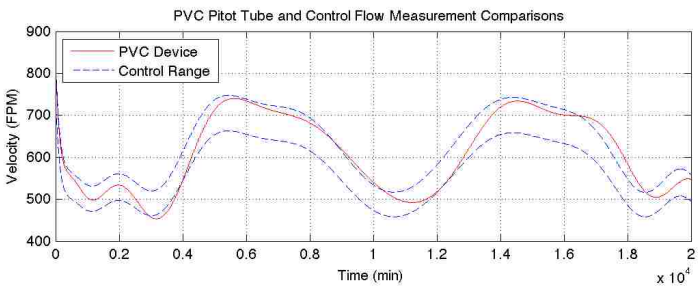

Figure 3.17: PVC Pitot Tube measurements compared to the calibrated control device range

that the pitot tube results remain within the range for the majority (95%) of the time shown. At this confidence level and based on the sensors ability to follow fluctuations in flow the sensor was considered adequate for this research effort and for use in building control in general.

3.2 IT Infrastructure

The research methodology included the development and operations of an IT infrastructure that integrated actual and modeled sensor data with FDD analysis tools. This infrastructure, described in Figure 3.18, included the extraction and insertion of sensor data from the BACnet into a database. The database supported a visualization for a simple user interface and FDD analysis experiments. The transfer of data from BACnet and to and from FDD experiments was accomplished using web services procedures such as Extensible Markup Language (XML) and Structured Query Language (SQL) script embedded in Python and PHP scripts.

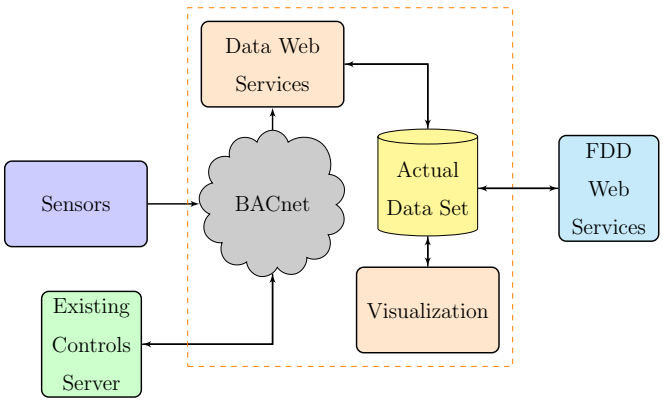

Figure 3.18: Flow diagram of IT infrastructure where the buildings sensors are connected to the BACnet network. The BACnet network can be accessed by the data web services developed in the present work and insert the sensor values into a relational database. The database can be accessed by a web-based visualization and the FDD tools.

The following sections describe BACnet, web services procedures, the database, and finally the web-based visualization.

3.2.1 BACnet Sensor Network

Sensors in buildings are constantly exchanging messages within the BAS that describe system performance and status. These messages are typically in the form of numerical data. Most often, the data are sent to a single server where control programs define activity. This platform accesses the data and routes it to a database that is separated from the proprietary controls. The database can then be accessed by a user friendly visualization interface and most importantly the FDD tools can access and analyze the data. To achieve this configuration, the system connects with the network used by the existing building controls system. In the present case this is **B**uilding **A**utomation and **C**ontrol **net**work (BACnet).

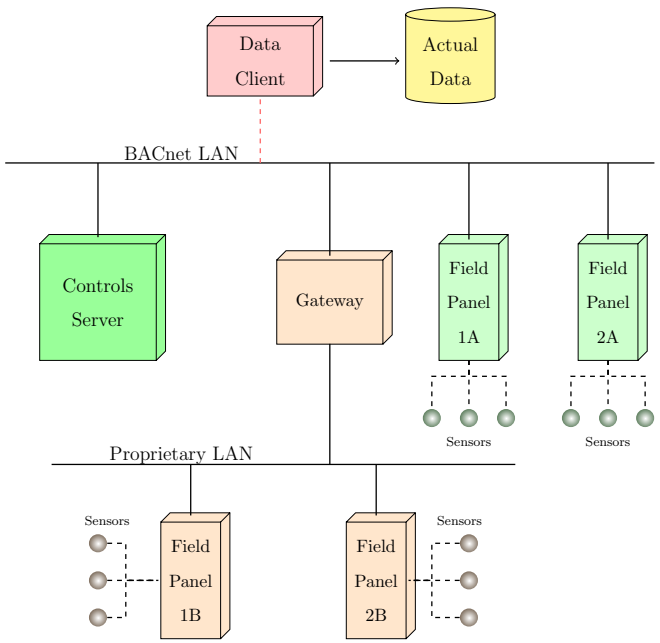

Figure 3.19: The sensor network can be comprised of a proprietary LAN that connects to the BACnet through a gateway. The sensors connect to a field panel that communicates through the LAN to the controls server. Finally, the data client used in the present work extracts the sensor data and inserts it into a database.

The BACnet protocol is a an ASHRAE, ANSI, and ISO [3] standard communication protocol. The protocol provides a reliable network communication and controls for building automation and control of HVAC, lighting, access control, and fire detection systems [86]. The protocol was established in 1987 during an inaugural meeting of ASHRAE

Standard Project Committee (SPC). During the meeting, a consensus was reached that defined BACnet as the central standard for building automation communication. Then in 1995 it became an ASHRAE/ANSI Standard 135, and in 2003 became ISO 16484-5 standard. BACnet is now implemented internationally with over 721 control vendors identified on the BACnet website.

The BACnet protocol defines the communication between building devices and include services such as "Who-Is", "I-Am", "Who-Has", and "I-Have". These services are used for device and object discovery so that data sharing using "Read-Property" and "Write-Property" can be conducted reliably. In addition, the protocol defines objects such as Analog Input (AI), Analog Output (AO), Analog Value (AV), Binary Input (BI), Binary Output (BO), Binary Value (BV), etc. The transfer of data can be conducted through many different types of data links or physical layers. This includes ARCNET, Ethernet, BACnet/IP, Point-to-Point over RS-232, and Master-Slave/Token-Passing over RS-485.

A general benefit of BACnet is that in order to comply with the standard all control vendors must speak the language so that communication can be integrated, available, and centralized. Figure 3.19 describes a simple example set up of the network that includes a BACnet and proprietary Local Area Network (LAN). The proprietary sensors, actuators, and field panels communicate on their own LAN. The proprietary LAN can be con-

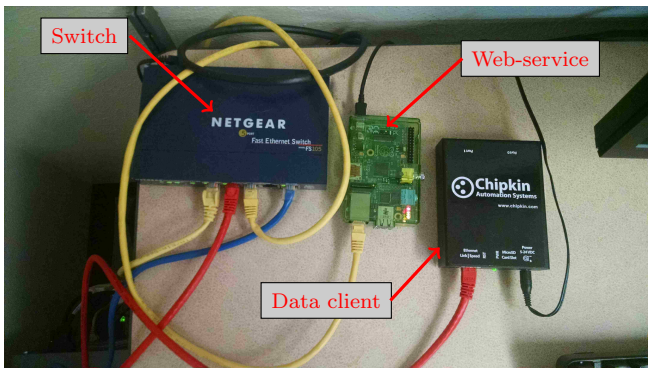

Figure 3.20: The Chipkin BACnet Data Client is connected to the building controls BACnet network and displays data on XML.

nected to the BACnet through a gateway. This set up has allowed a central BACnet workstation to communicate with both BACnet field panels and proprietary devices. For instance, the data acquisition for this project used a data client that extracted sensor data directly from the BACnet. Then, through web-service methods, the data were sent to a database as described in Figure 3.19.

3.2.2 Web Services

In the context of this research, web services are methods used for communication between two devices over a network. This communication can be performed using XML, SOAP, REST, SQL, and others. The present IT framework incorporated web services for two applications. First, web services were used for sensor data extraction from the BACnet and then to transfer the data to a database. Second, stored data was queried by FDD analysis tools and results were stored in the same database.

Data Web Services

The BACnet protocol, as described in the previous section, provides a platform to access the existing sensors independently of the building controls vendor. To read sensor data available on this network a device called a BACnet Data Client, manufactured by Chipkin Automation Systems, was used. The device can connect to the ethernet network through a standard network interface such as a hub, switch, or router. For this research application, because there was a minimal number of hubs, a switch was used to connect to the network (Figure 3.20).

The setup shown in Figure 3.20 allowed sensor data in the BACnet/IP network to be read through the network switch. The values were extracted using the data client and displayed on an XML web page. The web-service, which in this case was a Raspberry Pi Linux based computer, ran PHP scripts that accessed the XML,

parsed it, and finally inserted the data into a MySQL relational database. The scripts were executed every 5 minutes using the crontab application on the linux operating system.

The set-up of the data web services included the separation of networks and the transfer of data through XML. The building controls system, the Data Client, and the Raspberry Pi web-server were located on a private network. The private network, designated by 10.xx Internet Protocol (IP) address prevented access from outside the UNM network. The database, however, was on a public network that could be accessed from outside of the UNM network. The data transfer process began at the Data Client. The Data Client, located in the private network, provided the sensor data in an XML format. The Raspberry Pi parsed the XML to extract the necessary data and then inserted it into the database located in the public network. A firewall was set up in the database server configurations that only allowed a designated IP address to access the database.

FDD Web Services

The operations of FDD analysis for the experiments required web services to transfer data to and from the database. Data were queried by the FDD tool, an analysis tool performed a prediction or classification to review performance status, and the results were inserted into the database. The script was different for each of the analysis tools, but followed the same general approach. Many of the FDD methods, such as ART or LAPART, were performed without the creation and use of text files. Instead of creating, reading, outputting, and then parsing a text files, these FDD tools transfer data to the tool in a much simpler manner. The data was accessed through a single SQL query, and then the results were put into the database with an SQL insert. For example, Algorithm 1 describes Python code to perform these SQL tasks.

Algorithm 1 Python MySQL Query & Insert Example Code

```
# Database connection
$db = MySQLdb.connect(host, user, password, dbname);
cursor = db.cursor

# Database Query
sql = "SELECT * FROM tbname" # select statement
cursor.execute(sql) # execute select
results = cursor.fetchall()
variable = [ ]
for row to len(results)  do
   variable.append(row[0])
end for

# Database Insert
sql = """INSERT INTO %s (col) VALUES(%s)""" %(tb,var) # insert statement
cur.execute(sql) # execute insert
```

3.2.3 Database

Databases have become essential for many business applications. Most major websites across the internet use databases to help service requests and provide information. Database Management System (DBMS) is a tool for managing large amounts of data efficiently over a long period of time [82]. DBMS can be structured with tables that have relationships with one another so that data could be queried across many tables with a single command. The most widely used relational database is Standard Query Language (SQL). The execution of the SQL is supported within many different programming languages such as PHP, Perl, and Python. DBMS provides an effective mechanism for accessing data across internet servers to create an effective means to access building data and store fault detection results.

There are many different database engines such as PostgreSQL, SQLite, Microsoft SQL Server, Oracle, SAP HANA, among others. In this case a MySQL DBMS was used because it is both open source and simple to implement. MySQL uses a

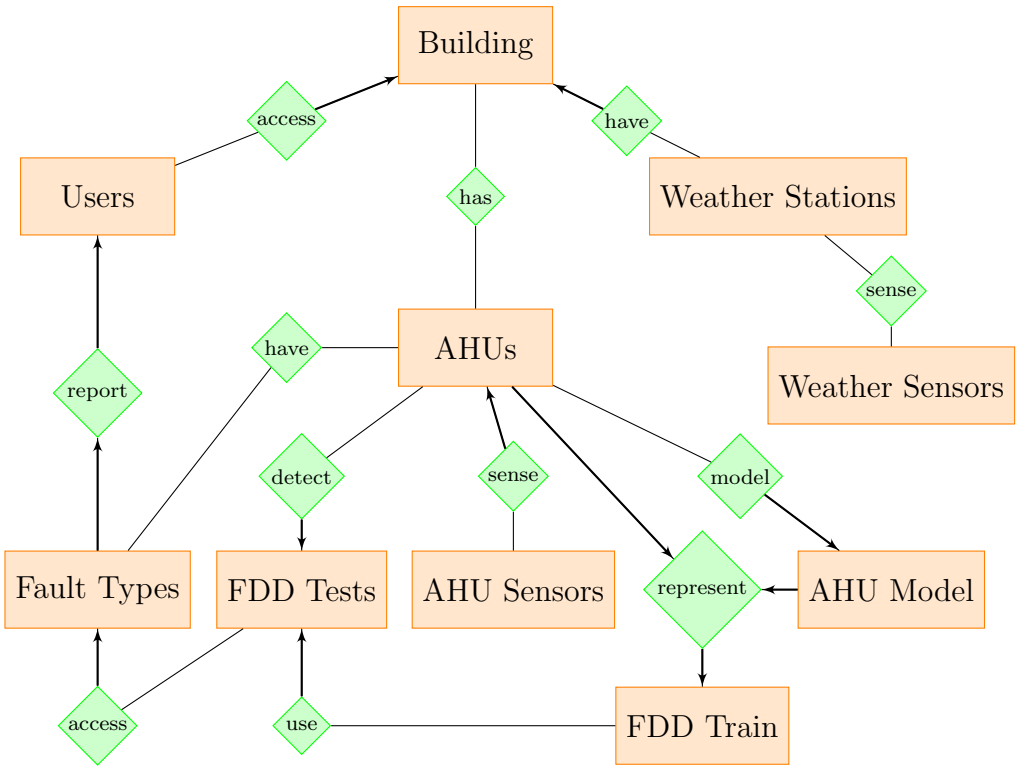

Figure 3.21: Database entity relationship (ER) diagram that describes tables and how they interact with each other in the database created for this application. The boxes and diamonds that have double lines are weak entities. A weak entity is an entity that cannot be uniquely identified by its attributes alone.

relational model for storing data which allowed the user to perform a single query to access data in multiple tables [82]. The MySQL tables, each of which could be considered an entity, contained columns that are referred to as attributes. These attributes describe the meaning of the data entries. Furthermore, the arrangement of the attributes for a table represent the overall schema and are linked through entity relationships. Also, a row of a table is referred to as a tuple which has one component for each attribute of the relation. Lastly, keys, which are composed of one or more attributes, help relate tables with each other. This structure allowed

for data collection, visualization, and FDD analysis to occur in a simple manner, a feature that was crucial for a successful FDD implementation.

The development of a database schema requires the consideration of certain design principles. Molinar *et al.* described these principles as the following: (1) the schema should reflect reality, (2) be simple, and (3) avoid redundancies. These principles were applied to the entity-relational (ER) diagram shown in Figure 3.21. The ER diagram is a model that describes the aspects of the database requirements and the relationships between the tables. In the figure the entities (rectangles) represent the tables and the attributes (ovals) are the columns within the table. The diamonds describe the relationship between the entities.

The database schema for the present work is described in Figure 3.21. The attributes for each entity are not included for in the figure for clarity. The first entity, *Building*, is connected to three other entities which are *Weather*, *AHUs*, and *Users*. The *Weather* entity stores sensor data from environmental conditions associated with a certain geographic location. This entity was considered weak (designated with a extra border) because its key is partially determined by attributes that belong to another entity and therefore, it must use a foreign key in conjunction with its attributes to create a primary key. For example, outside air temperature could have the same designation for different buildings within the same geographic location. Therefore, the location and building identification (id) are required to define the tuple that contains the correct outside air temperature data. The other two entities that are connected to the building entity are *Users* and *AHUs*. The users entity stores password and username information on people such as energy managers, owners, etc. so that the web-site can verify their credentials and allow access to the building data and analysis results. The *AHUs* entity stores general information on individual AHUs. It is connected to other entities such as *Building*, *Fault Types*, and others so that associations can be made in a single query to link data to the AHU. For example,

a single query can consider an AHU, recent sensor data, and fault conditions.

The other entities *AHU Sensors*, *AHU Model*, *Fault Types*, *FDD Tests*, and *FDD Train* are associated to the *AHU* entity. The *AHU Sensors* describes the table that stores all of the data for each AHU. This includes an identification that links each data point to a particular AHU in the *AHU* entity. The sensor data from the *AHU Sensors* entity can be used to calibrate models and the results could be stored in the *AHU Model* entity. Additionally, the algorithm dependent results from the FDD training algorithms are stored in the *FDD Train* entity. Process history FDD tests use the data in the *FDD Training* entity and also actual sensor data to detect faults and then store the results in the *FDD Tests* entity.

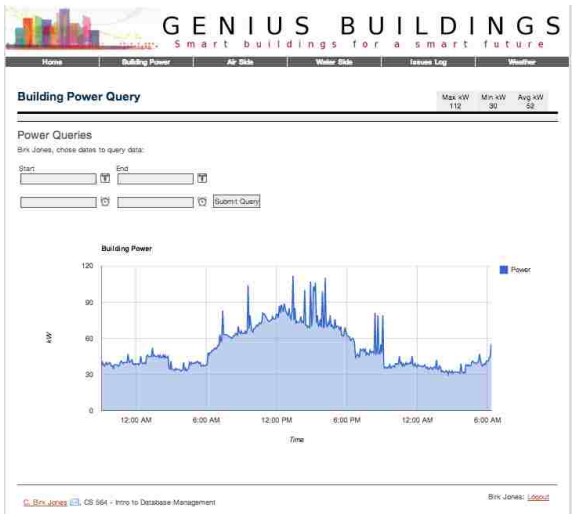

Figure 3.22: Sample screenshot of the web-based visualization for the FDD tools.

The *Fault Types* entity accessed the testing results and expert user input to define fault types. The fault types stored in this entity are associated with particular results in the *FDD Tests* entity and also linked to the particular AHU where the fault occurred.

3.2.4 Visualization

A very important aspect of the FDD tool is the visualization so that users can monitor, review, and understand reported failures. In this case, a web-based visualization was implemented because of its ease of access by both users and the database that is

described in Section 3.2.3. An example screen shot of the web interface is shown in Figure 3.22. To access the interface the user must be registered and use the assigned username and login. This image shows user *Birk Jones* logged in and prompts to query the power usage at different intervals. The user can also exit the interface by clicking "logout" at the bottom right corner. In addition to the graphical representation of the power data there is print out of basic statistics of the graphed data that include max, min, and average.

The web interface includes many different pages that can be used to review building performance. The main page described the intent of building FDD and the general system layout that is being evaluated. The interface also provided a simple list of addressed and existing faults. In addition, it provides an estimation of energy saved after the fault was addressed, or money that was lost due to not addressing a particular fault. Finally, the main page provides links to other pages where users can review identified faults or good behavior in more detail.

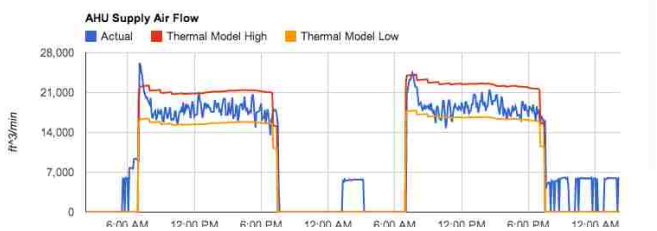

Figure 3.23: Screen shot image of graph found on secondary web page to review actual performance versus modeled results.

The secondary pages, for reviewing system operations in more detail, contain temporal graphs of system performance. For example, a screen shot of the actual air flow sensor data from AHU2 and expected modeled results from a real-time TRNSYS model is shown in Figure 3.23. The actual data were provided in blue and the high and low ranges for the model results are in red and orange respectively. The FDD tools presented in this research paper provide all of the analysis to determine if a fault had occurred or not. However, further questions or concerns can be addressed

by accessing these secondary pages, that provide valuable information for determining corrective action or mitigation procedures to create a more optimal process for training of the FDD tools.

3.3 Physical Model

A physical model was developed to provide a representation that was based on actual physical properties of the building and AHU 2. The critical part of the development was the model calibration. This involved the modification of parameters so that the model outputs best matched reality. The calibration process was conducted as part of Experiment 1. Additionally, the results from the simulations were used as inputs for training and testing of the FDD tools in Experiments 2 through 4.

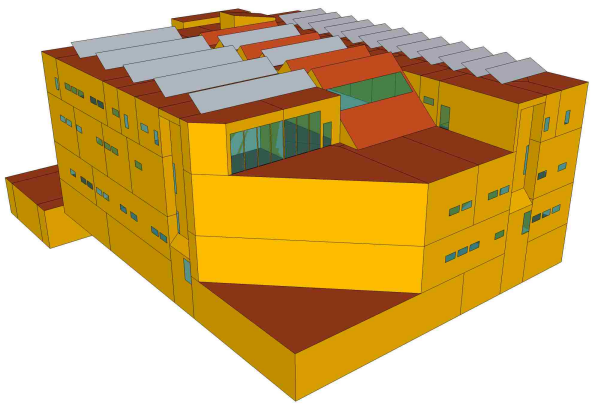

Figure 3.24: TRNBuild 3D thermal model of the Mechanical Engineering Building created in Sketch Up w/ TRNSYS add on

The model was created using a commercially available building energy simulation software called *TRNSYS* (Transient System Simulation Tool). *TRNSYS* is a flexible graphically based software environment used to simulate the behavior of transient systems through algebraic and differential equations. In this case, the tool was used to estimate actual air handling unit air flows and temperatures by running detailed simulations that considered building geometries, external and internal loads, weather,

control parameters, and ventilation rates.

Unlike ANN, the physical model approach requires significant knowledge of the building and the AHU system. The TRNSYS model included a two part approach for model development, that included three dimensional sketch of building geometries and then the set up system components such as fans, dampers, etc. Once the model was complete the simulation of building operations was conducted within a kernel

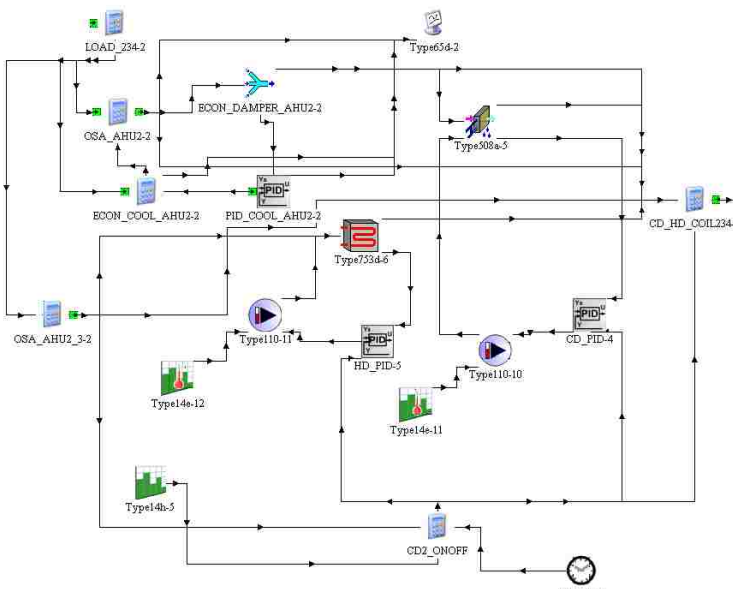

Figure 3.25: TRNSYS Simulation Studio component interface for AHU 2. This graphic does not show the entire model because it was broken out into separate pages to improve the organization of the layout. For example, the building model is on a separate page and is referenced by the component LOAD_234-2.

engine. The engine compiled and processed input files and then iteratively solved the system equation, determined convergence, and finally plotted system variable results. The simulation engine has an extensive library of building components, each of which models the performance of pumps, valves, vans, etc.

The three dimensional building geometries were created in *Trimble Sketch Up*. The building geometries and fenestration locations were replicated to scale and were oriented in the correct cardinal direction as shown in Figure 3.24. This file was

then imported into TRNBld software where internal loads for all thermal zones, and material types and thickness for walls, roofs, ceilings, windows, and other elements were assigned. The second part of the model development was to create AHU 2 in the TRNSYS simulation studio software. The AHU setup is shown in Figure 3.25. The components included a fan, hot and chilled water pumps, hot and chilled water coils, valves, dampers, and controllers. The components were setup in accordance with the physical parameters in actual unit.

After the model was created, inputs were presented to the TRNSYS model at 5 minute intervales. The inputs included actual weather data such as outside air temperature, relative humidity, and solar irradiance. The model then performed its simulation and produced outputs for the AHU that included mass flow rate, supply air temperature, and mixed air temperature.

Chapter 4

Methodology: FDD Tools

Artificial neural networks (ANN) are a form of machine learning that emulate a simplified version of an animal's nervous system for the purposes of acquiring and storing knowledge. The algorithms have the ability to build rules. These rules are built as features from the sensor data are processed through highly interconnected elements, called neurons. The neurons work together to learn and store memory of experiences. ANN provide a high level learning system that can perform complex computations, and calculate many nonlinear problems.

The basic elements of synaptic learning in an animal's nervous system are comprised of dendrites, axon, and synapses. The process begins in the dendrites where sensor data was collected from other neurons. A signal is then transferred through the soma and into the axon. The axon is a cable like structure that passes information to the axon terminal. At the axon terminal there are branches that connect with other dendrites. At this point chemicals, called neurotransmitters, are transmitted to receptors. This transmission is where the learning occurs. These receptors receive the neurotransmitters in a manner that is based on prior learning. This structure of transferring information provides a mechanism for interpretation and

decision making that is highly complex and efficient.

ANN are not nearly as sophisticated as actual neural systems but are never the less capable of performing surprisingly complex learning computations. The basic layout and interconnections of input, hidden, and output layers for a simple single-layer perceptron is shown in Figure 4.1. The interconnections of the neurons and the learning algorithm allow for the tuning of numerical parameters known as weights (w). The interneuron connection strengths, which are the weights, are used to store the acquired knowledge for classification and/or prediction [43].

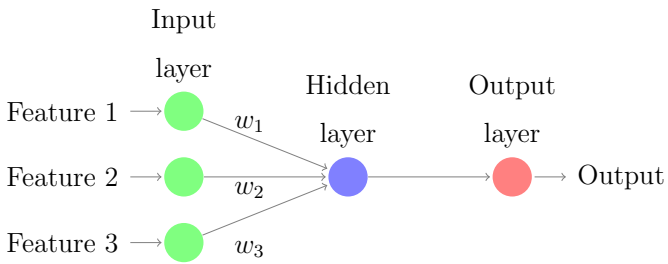

Figure 4.1: A graphical representation of a single-layer perceptron ANN that has an input layer, one hidden layer, & an output layer.

The most common type of learning is called supervised learning where the modification of weights occurs through the presentation of training samples. This includes the input of a unique value and a corresponding desired output. Haykin [43] refers to this as input-output and explains that the weights of the network are modified to minimize the difference between the desired and actual response.

ANNs have the ability to learn during the training mode and then produce predictions while in testing. The predictions can be generalized, which means that the ANN can provide reasonable outputs for inputs not encountered during training. Additionally, the ANN can solve complex problems such as linear and nonlinear systems. These qualities are very important because many HVAC systems have nonlinear properties. HVAC system performance can change from day to day and from month to month. Therefore, an adaptive analysis system can be used to provide

reliable outputs.

4.1 Adaptive Resonance Theory Tools

Some ANN algorithms can adapt their synaptic weights to changes in the surrounding environment [43]. For example, an ANN that was originally trained in a specific environment can be retrained to account for different conditions. One such algorithm is ART. The original ART algorithm was invented by Carpenter and Grossberg [38] and provided a theory for overcoming instability that occurs in competitive learning [39]. Further work in the area incorporated fuzzy set theory into the original ART to create a Fuzzy ART algorithm [13] which can learn from analog input patterns in an unsupervised manner. A separate algorithm, called the LAPART was introduced by Healy and Caudell [48]. This algorithm consists of interconnected Fuzzy ART modules to infer one pattern class from another to create a prediction [40]. The ART and LAPART use self-organizing learning at their cores, and do not use any form of gradient descent. This type of approach allows for rapid learning.

4.1.1 Fuzzy Adaptive Resonance Theory

Typical ANN architectures have the ability to learn. However, they may forget old information or have to relearn the old information in order to gain more knowledge. For example, suppose a typical ANN algorithm is used to recognize the outline of every bicycle. Once the ANN has learned and classified the outlines, which can be a time consuming activity, the training period is over and no further modifications can be made on the final weights. If a new class of bicycle with a different outline is developed the ANN would have to retrain the network with the new pattern plus all of the previous patterns.

Fuzzy ART is a ANN architecture that can learn without forgetting. It is similar to human memory where people can recognize their parents even if they have not seen them in a while and have learned many new faces since. The theory was developed by Grossberg and Carpenter and includes various types such as ART 1, ART 2,

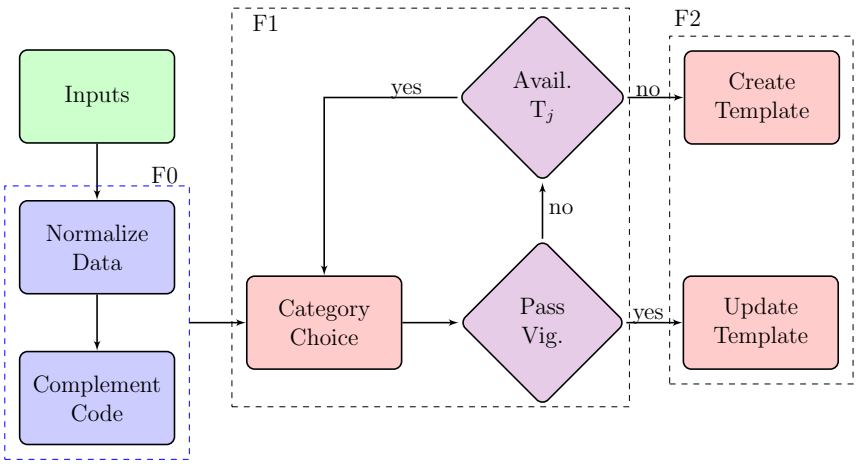

Figure 4.2: Flow diagram of ART training algorithm where inputs are normalized in the F0 layer. Then category choice and vigilance are processed in the F1 layer. The final F2 layer is where the templates are created and stored.

ART 3, and Fuzzy ART. ART 1 is an architecture that can be used for clustering of binary inputs only [15]. ART 2 improved upon the ART 1 architecture to support continuous inputs [14]. Fuzzy ART, used in the present work, incorporates fuzzy set theory into the pattern recognition process.

Unlike the ART 1, the Fuzzy ART approach can provide stable categorization of analog input patterns [13]. The fuzzy logic improves the generalization of the algorithm which increases its ability to perform classification [16]. The ART algorithm incorporates a vigilance parameter (ρ), which can be characterized as a similarity parameter. This parameter is used to judge the similarity between all of the input patterns.

Architecture

The basic structure of the Fuzzy ART architecture is shown in Figure 4.2. This flow chart describes the interconnection of the different layers that are designated as F0, F1, and F2. The F0 layer is for the normalization of the input values, and is consid-

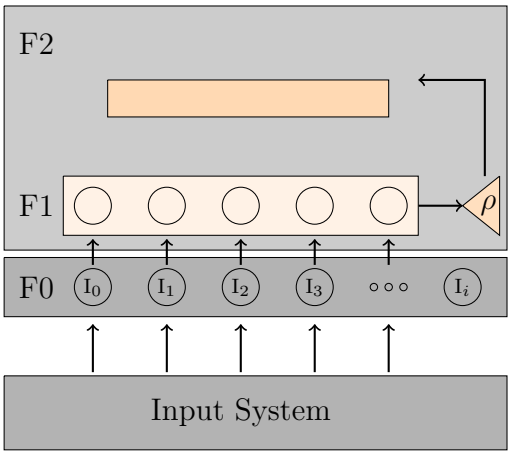

Figure 4.3: The ART algorithm has the F0, F1, and F2 layers that perform data preprocessing, recognition of features, and categorization of inputs

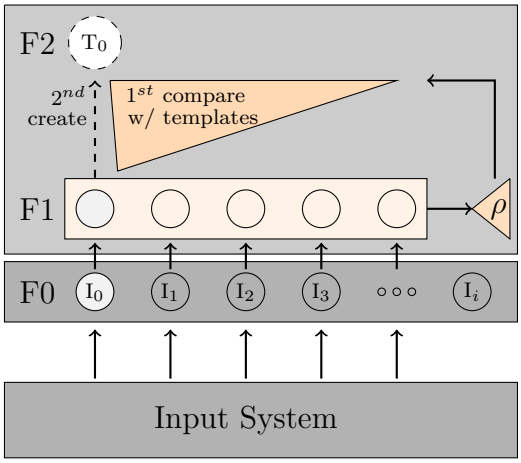

Figure 4.4: The training process began with computations of I_1 in the F1 layer that compared and found no templates in F2 so T_0 is created

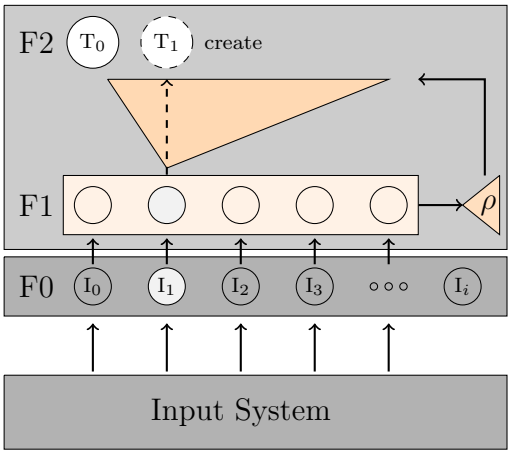

Figure 4.5: I_2 doesn't resonate with template T_0 so T_1 is created

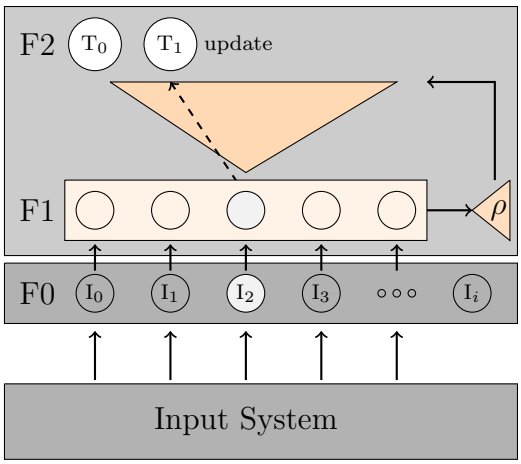

Figure 4.6: I_2 resonated and updated T_1

ered a preprocessing operation. The F1 layer, which includes the choice and vigilance calculations, performs the recognition of features in the data. This process includes

a reset that is based on a vigilance a parameter (ρ), and is considered the orienting subsystem [36]. The final layer, F2, is where the categorization occurs. Similar to other ANN architectures the Fuzzy ART has a weight (or template) matrix. The ART 1 algorithm has a bottom-up and top-down process for determining templates. However, the Fuzzy ART merges the bottom-up and top-down into one process [16]. The training of this template matrix is described in Figures 4.3 to 4.6.

Figure 4.3 illustrates the initial status of the system prior to the presentation of the inputs. Then, in Figure 4.4, the first input, I_0, is presented to the system after first being normalized and complement coded in the F0 layer. The process then moves to the F1 layer where it is compared with any existing template nodes from the F2 layer, as indicated by the orange triangle. In this case, there are no existing templates so I_0 becomes the first template T_0. This process is described with a dashed arrow pointing from the F1 to the F2 layer within Figure 4.4.

The next step is to evaluate the second input, I_1, by passing it from the F0 to the F1 layer and then either create or update the templates in the F2 layer. Figure 4.5 shows that the I_1 was presented to the existing template T_0, and resonance did not occur. Therefore, a new template T_1 is created. The last input presentation for this example is shown in Figure 4.6 where I_2 is compared with both T_0 and T_1, and resonated with T_1. Because of the resonance the T_1 template is updated.

The Python programming language was used to implement this algorithm in the present work. The variables and parameters used in the different layers of the algorithm are described below. In addition, the equations used for preprocessing, feature extraction, and categorization are briefly described in Equations 4.1 to 4.9.

Algorithm Notation

Preprocessing

I - Input matrix

I_{min} - min of Input matrix

I_{max} - max of Input matrix

$a_{i,j}$ - normalized scalar

$a^c_{i,j}$ - complement coded normalized scalar

$\mathbf{A}$ - vector of a

$\mathbf{X}$ - preprocessed input matrix

Category Template

$\mathbf{T}_j$ - Template (weight) vector

$\mathbf{c}$ - Choice function vector

ρ - Vigilance parameter

α - Choice parameter

β - Learning rate

$\wedge$ - fuzzy set theory conjuction or minimum operator

Preprocessing (normalization & complement coding)

The preprocessing of the input patterns occurs prior to their presentation to the Fuzzy ART. This is considered the F0 layer. The first step is to normalize the data. This entails the transformation of the i by j dimensional training data, denoted as D, to values that are between 0 and 1.

$$a_{i,j} = \frac{(I_{i,j} - I_{j,min})}{(I_{j,max} - I_{j,min})} \tag{4.1}$$

Equation 4.1 is used to normalize the data to calculate $a_{i,j}$. The second preprocessing stage is to perform complement coding of $a_{i,j}$. This is done using Equation 4.2 to create $\mathbf{a}^c_{i,j}$. Finally, to complete the preprocessing stage $\mathbf{a}_{i,j}$ is joined with $\mathbf{a}^c_{i,j}$ in Equation 4.27 to get the input matrix ($\mathbf{I}$) that can be presented to the input (F1) layer of the Fuzzy ART algorithm.

$$a^c_{i,j} = 1 - a_{i,j} \tag{4.2}$$

$$\mathbf{X} = (\mathbf{A}, \mathbf{A}^c) \tag{4.3}$$

Template Matrix Computations

After the preprocessing stage in the F0 layer, the template matrix computations occur between the F1 to the F2 layers. First, network parameters α, ρ, and β are set at the beginning of each training phase. The learning rate value, α, can assume values between 0 and infinity, but is typically very small. The vigilance parameter, ρ, must be between 0 and 1, and determines the fineness of the clusters.

The next step is to present the input pattern ($\mathbf{X}$) to the template matrix and find the node(s) that pass vigilance according to Equation 4.4.

$$\frac{|\mathbf{X} \wedge \mathbf{T}_j|}{|\mathbf{T}_j|} \geq \rho \tag{4.4}$$

If none of the nodes pass this vigilance test, then a reset occurs. If there are nodes that pass, then the node with the highest value from Equation 4.5 defined by

$$\mathbf{c} = \frac{|\mathbf{X} \wedge \mathbf{T}_j|}{\alpha + |\mathbf{T}_j|} \tag{4.5}$$

and the fuzzy AND operator $\wedge$ is

$$(\mathbf{x} \wedge \mathbf{y}) = min(x, y) \tag{4.6}$$

Additionally, the norm $|r|$ is defined by Equation 4.7:

$$|\mathbf{r}| = \sum_{i=1}^{M} |r_i|. \tag{4.7}$$

At this point, the particular input pattern either has or has not found a match with a template node. When a does occurs, which is referred to as resonance, then the template node is updated according to Equation 4.8. Otherwise, a new node is created and its template values are calculated using Equation 4.9. At the completion of the algorithm every input

$$\mathbf{T}_j^{update} = \beta(\mathbf{I} \wedge \mathbf{T}_j^{old}) + (1 - \beta)\mathbf{T}_j^{old} \tag{4.8}$$

$$\mathbf{T}_j^{new} = \mathbf{I} \tag{4.9}$$

pattern will belong to a template node. The convergence of the algorithm requires that the template matrix values do not change from one full presentation to the next.

Concept

The Fuzzy ART architecture is very useful for pattern recognition and categorization, because it can adapt easily to significant variations through a normalization of the

network activity. It can also recognize subtle differences in input patterns which is critical for accurate decision making. Additionally, the algorithm can perform detailed recognition of features. The architecture can also remember, and use its memory to make decisions and predictions.

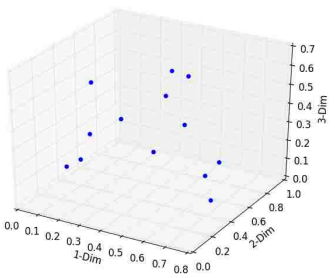

Figure 4.7: Scatter plot of 3 dimensional training data set.

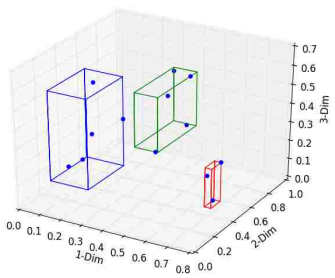

Figure 4.8: Training data that is classified w/ low vig. hyperboxes.

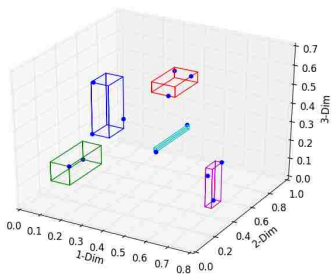

Figure 4.9: Training that is classified w/ high vig. hyperboxes.

The basic implementation of the algorithm on a set of data is considered in Figures 4.7, 4.8 and 4.9. In this example thirteen three-dimensional input values are considered and shown in a scatter plot in Figure 4.7. The Fuzzy ART algorithm can learn these input values by creating a template matrix with nodes that categorize the data. Each template node represents a particular category and is considered a hyperbox in this 3 dimensional space. Figure 4.8 shows three hyperboxes created by Fuzzy ART algorithm with a low vigilance parameter. Each input data point lies inside one of the hyperboxes. If the vigilance parameter value is increased then the amount of hyperboxes will likely increase as well. In this case the generalization, of the system would decrease. This concept is highlighted in Figure 4.9 where the number of hyperboxes increased from 3 to 5 while maintaining full coverage of the input data. The size of the hyperbox depends on the ρ and dimensionality of the input features. In a D-dimensional case the size of the hyperbox is given by Equation 4.10:

$$\text{Size} = D(1 - \rho) \tag{4.10}$$

4.1.2 Lateral Priming Adaptive Resonance Theory

LAPART neural networks couple two Fuzzy ART algorithms to create a mechanism for making predictions based on learned associations. The coupling of the two Fuzzy ARTs has a unique stability that allows the system to converge rapidly towards a clear solution [46]. Additionally, it can perform logical inference and supervised learning similar to fuzzy ARTMAP [12], but has been considered easier to implement [40].

This method has been used in applications for controls, classification, and prediction. Edlund *et al.* [29] implemented LAPART into a virtual test for applications in controls of autonomous vehicles. It has also been implemented within a computer assisted chest radiograph reader system where it acted as a classifier to detect patho-

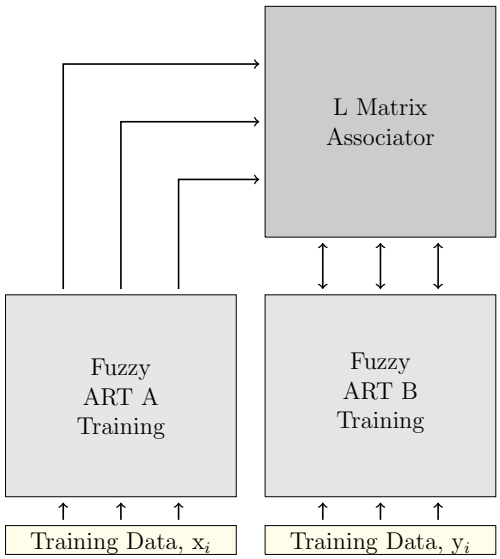

Figure 4.10: LAPART **training** uses two Fuzzy ART (A&B) algorithms connected by an associator matrix (L). During training inputs x_i are applied to the A-side while y_i inputs are presented to the B side. The algorithm then produces templates and an L matrix.

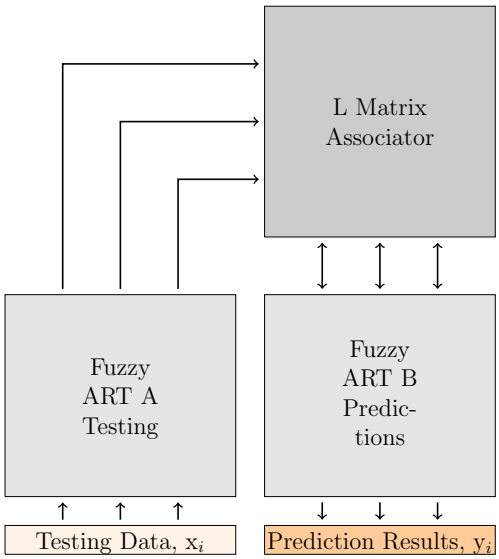

Figure 4.11: LAPART **testing** used the same structure that was uses during training. Yet, in this case, a new, previously unseen testing data set was an input (x_i) for the A side. The algorithm then produced, on the B side, outputs that are prediction results (y_i).

logical features in lungs [108]. Recently, the approach has been applied in image pro-

cessing and predictions for solar irradiance to improve photovoltaic smoothing [74].

Architecture

The general layout of the LAPART algorithm includes two fuzzy ARTs, labeled as A and B, that are connected by an associator matrix referred to as L. These two Fuzzy ARTs are very similar to the algorithms described in Section 4.1.1. Each of them have an input layer, a recognition layer, and a categorization layer. Also, they both have a vigilance parameter ρ_A, ρ_B respectively. The A and B algorithms are connected together by an inference mechanism so that the template connections are established during training and then used to provide predictions during testing. The flow of the algorithm is shown in Figure 4.10. During training, the system is able to learn through the presentation of input pattern pairs ($\mathrm{I}_A\{x_1..x_n\}$ and $\mathrm{I}_B\{y_1..y_n\}$) applied to each Fuzzy ART network [48]. At the same time, interconnections between classes are formed in the L matrix. The interconnections between the A and B Fuzzy ART connect the learned categories and allow for predictions to be made in the testing phase when new data becomes available. During testing, as shown in Figure 4.11, previously unseen data are presented to the A side only. Categorization of the input patterns occurs in the A side which connects, through the L matrix, to a particular category on the B side. The particular B side category for the input pattern is then the prediction for the given A input.

Fuzzy ART A

The training process is initiated with the presentation of an input pattern into the Fuzzy ART A algorithm. Since no templates exist at the onset of the training the initial input becomes the first template on the A-side. Therefore, a new template or class would be created and based on Figure 4.12 decision block, "New A Class", the algorithm implemented Case 1 script. In this situation, the Fuzzy ART B operates as a normal ART algorithm. Since there are no B-Side templates, a new template was

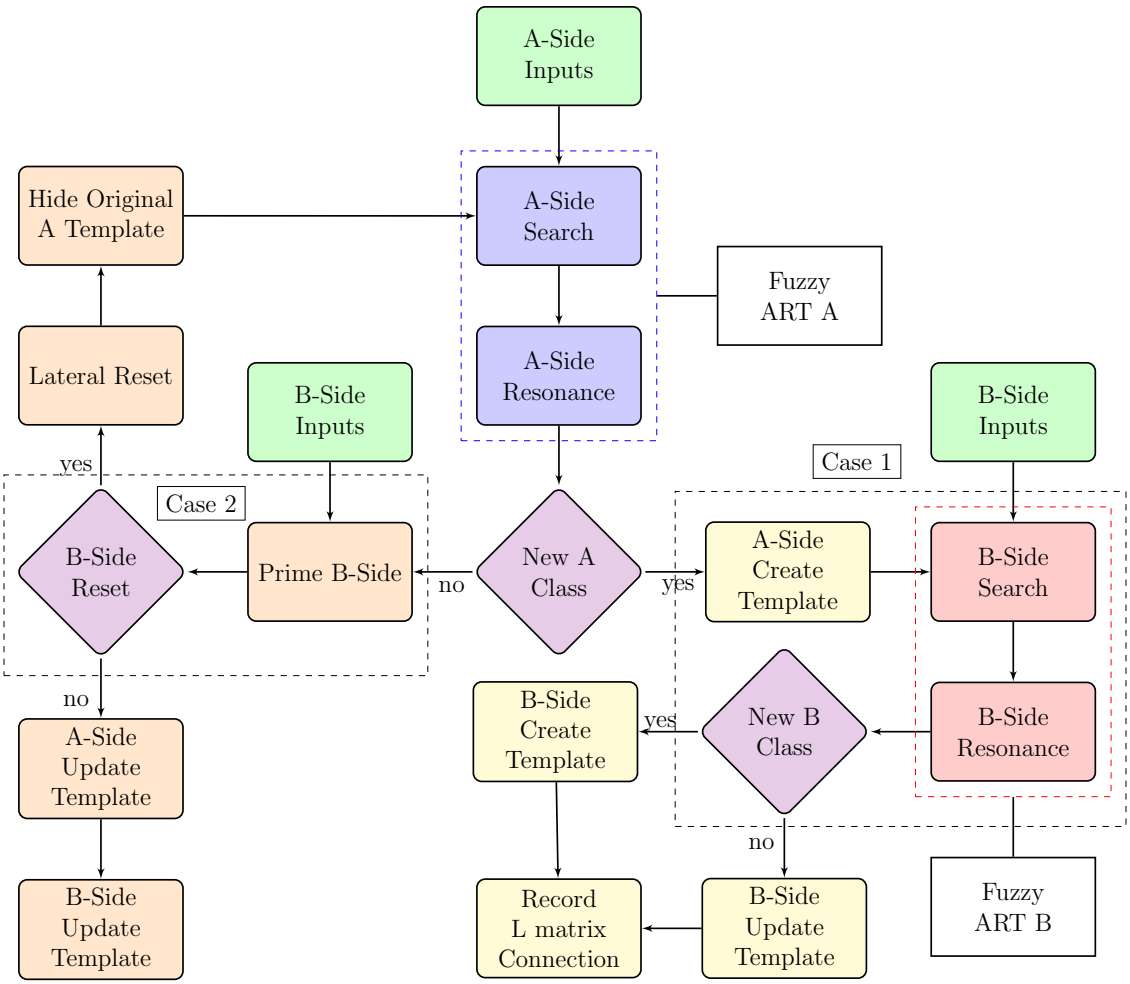

Figure 4.12: LAPART training algorithm flow diagram includes Fuzzy ART A & B and two (2) cases for learning A and B side templates as well as the inference matrix.

automatically created and a link between the A and B Fuzzy ARTs are established in the L matrix.

The algorithm then starts over with a new input, and repeats until all of the inputs are considered. When the new input is presented to all of the existing templates, in the "A-Side Search" block, it searched for the top matches as determined by the choice function (Equation 4.5). Then the vigilance test, defined by Equation 4.4, is computed in the "A-Side Resonance" block to determine if resonance with an existing template occurs or not. If resonance does not occur then the next best template was

considered until no choices were left. If a match is not found, then the script for Case 1 is implemented. If a match does occur, the script for Case 2 is used.

Case 1

The Case 1 code scenario occurs when a new A-side class was created and the Fuzzy ART B is allowed to operate as a normal ART algorithm. First, a new A-Side template is created. Then the B-Side input pattern is presented to the "B-Side Search" block where the choice function is used to discover the best template match. After the best match is found, the system tested the match to see if it met the vigilance parameter criteria in the "B-Side Resonance" block. If it did not, then a new template is created. If it did, then it updates an existing template. After the creation or update of a B-Side template, the inference matrix L was updated to link the newly created A-Side template to the B-Side template.

Case 2

In the event that resonance occurred in the Fuzzy ART A section of the code then Case 2 is implemented. First, the A-side template that resonated with the input

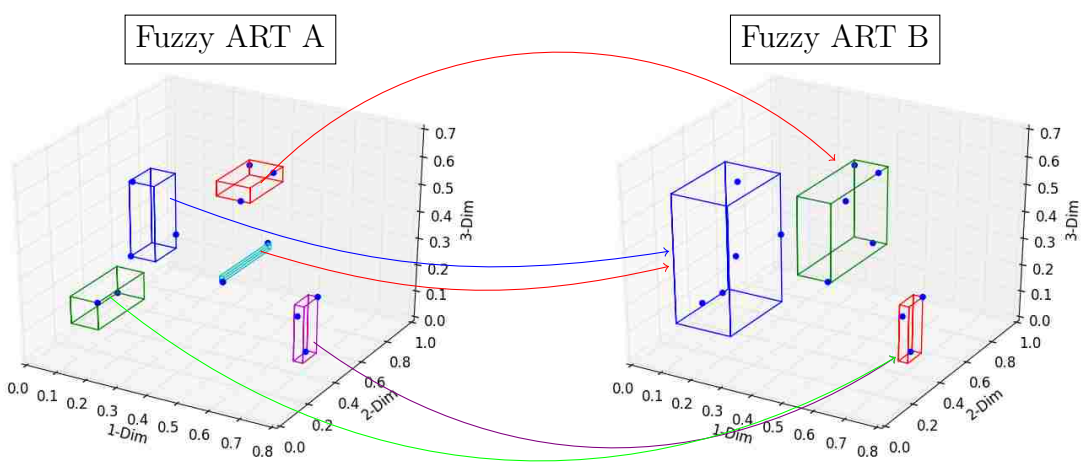

Figure 4.13: This figure presents the end result of a Fuzzy ART A that has learned class representations for an input pattern and at the same time associated with class representations learned in Fuzzy ART B

pattern is not updated, but instead put on hold until further notice. Also, the match function is used to find the template that best matched the given input pattern. Then, if the chosen template passed the vigilance criteria, where the match function was greater than or equal to the B-side vigilance (ρ_B), then the given A and B side templates are updated respectively. But, if it does not pass, then the system experiences a lateral reset and the initial A-side template is hidden and the process is repeated.

Concept

The LAPART system can learn to associate classes of patterns through an adaptive neural inference mechanism [48]. This is significant because LAPART neural networks can be trained to learn input patterns as well as the association between them. Figure 4.13 shows the results from a LAPART training process where an A and B Fuzzy ART has defined categories that are interconnected. The learned categories, for the A and B sides, are represented by the three dimensional boxes that surround the training data. The connections, provided by the inference mechanism, are also shown with the colored arrows. Then during testing, new data is presented to the A side. If resonance occurs with any of the categories a B side category is the output or prediction.

4.2 Non-Adaptive Resonance Theory Tools

Among the many techniques for FDD that were described in the literature review (Chapter 2), this research effort considers a limited but organized sample of methods to compare with the ART algorithms. The selected techniques each belong to a certain diagnostic class, namely qualitative, quantitative, and process history, as de-

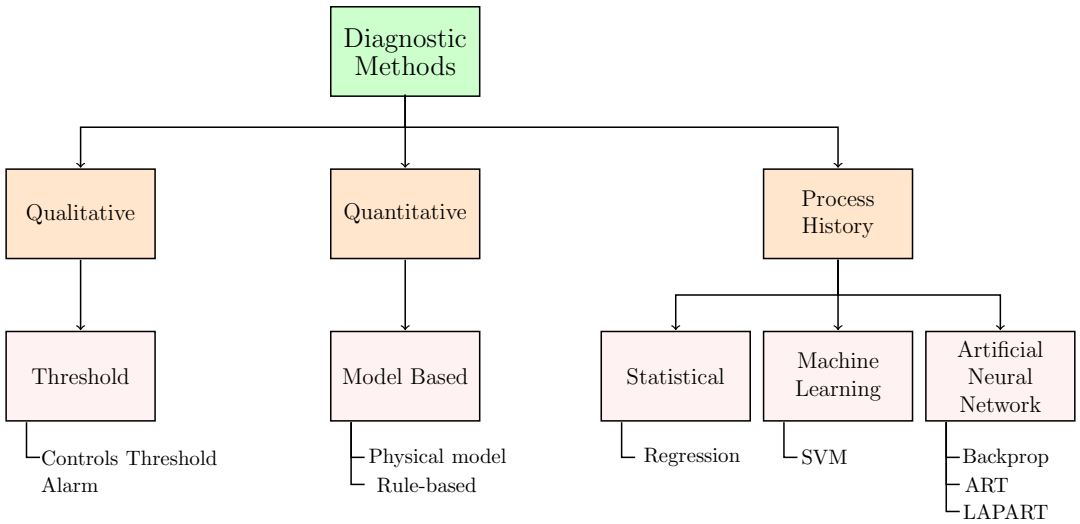

Figure 4.14: The FDD methods include quantitative, qualitative, and process history. Each of the methods were used to detect and/or diagnose faults in the AHU 2 data sets.

scribed in Figure 4.14. The selected techniques include a qualitative method referred to as threshold controls alarm. Rule-based and physical models are categorized as quantitative. Statistical regression, artificial neural networks, and support vector machines are process history methods.

The intent of implementing the different techniques to compare outcomes of each with the ART & LAPART algorithms for the AHU 2 sensor data. The hypothesis is that the two ART algorithms will provide meaningful results that will out perform other techniques. Each of the techniques used for comparison are described in Sections 4.2.1 to 4.2.4.

4.2.1 Controls Threshold Alarm Method

BAS that control HVAC equipment often include an alarm system that provide alerts to users. The triggering of one of these alerts is often based on a simple approach such as: (1) change of state, (2) change of value, (3) command failure, (4) floating limit, or (5) out of range. For example, an analog output (AO) value is a sensor that

monitors equipment performance. A threshold fault detection system can monitor the value, and thus the equipment using the "out of range" approach. In this case, the user sets high and low threshold values and when the AO value exceeds or falls below the pre-set thresholds an alarm is initiated. AHU 2 is currently controlled with a Delta Controls BAS and an out-of-range threshold alarm system has been established to detect faults.

The AHU 2 system has four alarms which are: (1) freeze alarm, (2) filter alarm, (3) supply fan alarm, and (4) smoke alarm. The freeze, filter, and smoke alarms are based on the "change of state" threshold approach. These alarms monitor the equipment for freeze protection, maintenance, and life safety. The supply fan alarm, on the other hand, is based on the "command failure alarm" approach, and monitors the fan's performance. This approach considers the supply fan status and BAS

Algorithm 2 This algorithm describes an AHU supply fan threshold alarm. The fan gets a binary input from the BAS that defines how it should operate. The binary output value describes it's actual status. If the fan status does not match the BAS command then an alarm is triggered.

Require: Binary Input - Supply Fan Command & Binary Output - Supply Fan Status
Ensure: alert users of supply fan fault

```
if Supply Fan Status = Supply Fan Command then
    No Alarm
else
    Alarm
end if
```

control command input to detect a fault as shown in Algorithm 2. The supply fan status is defined by a binary output (BO) value. This BO value should match with the command signal from the controls which is a binary input, and if it does not an alarm is initiated. This simple approach was used to detect fan faults in Experiment 2.

4.2.2 Rule-Based Method

Model-based methods include rule-based expressions and a physical model. In the present work, the rule-based expressions were developed using expert knowledge and statistical review of actual sensor data. The physical model was constructed in a commercially available simulation software. The results from each of the predictions were then compared with the actual sensor values to compute a residual.

Rule-Based Model

Rule-based expressions can simulate actual operations. For example, a tool for assessing performance was developed by Schein *et al.* [103], called the AHU Performance Assessment Rule (APAR). The APAR tool was used to detect faults in an AHU system during different heating and cooling modes. The APAR expressions were developed to define operations during different conditions within the modes. Based on results from fault scenarios that included stuck damper, temperature sensor drift and failure and others, Schein *et al.* concluded that the APAR is an effective tool for detecting HVAC faults.

Similar to Schein et al. approach, a set of rule-based equations were created to define operations of the AHU 2 system for the present work. The expressions were established based on expert knowledge of the system operations, specifications, and parameters. The expression parameters were then put through a calibration process to fine tune the equations to match actual, normal behavior.

These expressions are described in Algorithm 3 and include oversight of the supply air flow, the supply air temperature, and the mixed air temperature. The first step for developing the expressions was to consider the specifications of the equipment. The fan specifications listed in the original construction documents stated a maximum flow rate of 36,520 cubic feet per minute and a design supply air temperature of

Algorithm 3 Rule-based expressions pre-calibration

Require: hour of the day (hr), outside and return air temperature

Ensure: predict low and high supply air flow, supply air temp, and mixed air temp

if hr >= 7 AND hr <= 19.5 **then**

$m_{supply} = 13 m^3/sec$, $T_{supply} = 13.3°C$

if $T_{return} > T_{outside}$ **then**

$T_{mixed} = T_{outside} \pm \sigma_{T_{mixed}}$

else

$T_{mixed} = T_{return} \pm \sigma_{T_{mixed}}$

end if

else

$m_{supply} = 0$

if $T_{return} > T_{outside}$ **then**

$T_{mixed} = T_{return} \pm \sigma_{T_{mixed}}$

$T_{supply} = T_{mixed} \pm \sigma_{T_{supply}}$

else

$T_{supply} = T_{return} \pm \sigma_{T_{supply}}$

$T_{mixed} = T_{return} \pm \sigma_{T_{mixed}}$

end if

end if

53°F dry bulb. During the cooling season the mixed air temperature should be approximately equal to the outside air temperature until it rises above the building temperature. This is because the mixed air temperature should be as low as possible so that the CHW heat exchanger does not have to work as hard.

Based on information provided by the building specifications, and also through the consideration of system losses due to aging, the rule-based expressions were defined. First, the supply air fan has a variable frequency drive and was considered to operate at about 75% most of the time. Therefore, during the operating hours of

7:00 to 19:30 the rule-based expressions simulated an air flow rate that was 75% of the maximum flow rate of 36,520ft^3/min (17.17m^3/sec), or 27,390ft^3/min (12.9m^3/sec). The HX had not been a problem for the maintenance staff, and was considered in good working condition. Therefore, the supply air temperature was set to 56°F (13.33°C). The design specifications provide a temperature of 53°F (11.67°C), but an offset of 3°F (1.67°C) was added to the design specifications in an attempt to account for potential losses over the years. Finally the mixed air temperature was set to be dependent on the return and outside air temperatures by implementing if-then statements.

4.2.3 Machine Learning

Albert Einstein said, "Education is not the learning of facts, but the training of the mind to think". Many believe that the era of big data that we are entering, requires computers to think and perform the task of transfering data into information. Machine learning is a viable candidate for this task, and can perform automated data analysis that can detect patterns, predict future events, perform decision making tasks under uncertainty. In this section, various machine learning methods are described to highlight their potential impact on FDD of building HVAC systems. The techniques include regression, and support vector machines.

Regression Method

Regression is a technique for modeling and analyzing several variables. The statistical process estimates the relationships between one dependent and one or more independent variables. The general computational approach includes the fit of a linear or nonlinear function to a number of points. This is done through a least-squares

approximation where the overall solution minimizes the sum of the squares of the errors made by the function. It is usually accomplished through a gradient descent process that discovers the best function constants, β_0, β_1.

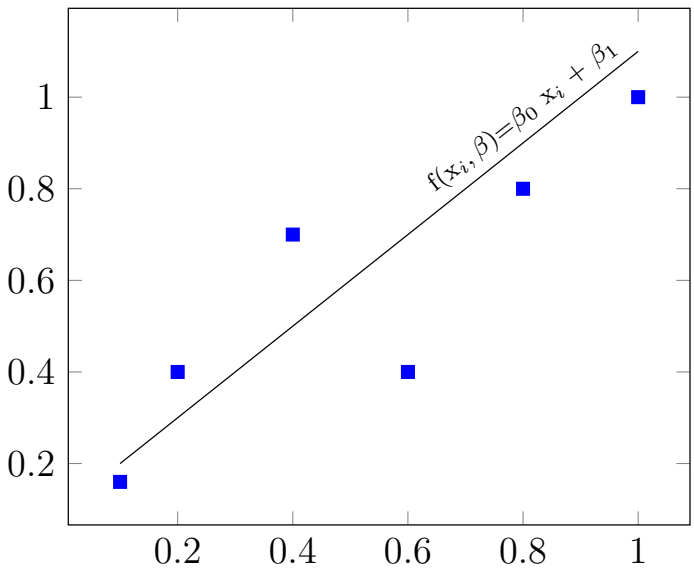

Figure 4.15: Regression linear fit to data conceptual example

Algorithm

Consider a data set that contains N samples, where x_i and y_i represent the input and output variables, respectively. For example, Figure 4.15 describes a set of data with a scatter plot where $N = 6$. The goal is to fit a function, $f(x_i, \beta_0, \beta_1)$, shown in Equation 4.11, to the data by finding the optimal β constants. In this case, a linear fit was accomplished by finding the optimal β_0 and β_1 which represent the slope and intercept, respectively.

$$f(x_i, \beta_0, \beta_1) = \beta_0 x_i + \beta_1 \tag{4.11}$$

The β constants are found through the minimization of the least-squares of the residuals. A python programming package was used that implemented a Gauss-Newton algorithm to iteratively find the minimum sum of squares that is described by Equation 4.12. The optimal constants,

$$S(\beta)_i = \sum_{i=1}^{N} (y_i - f(x_i, \beta_k))^2 \tag{4.12}$$

β, were updated throughout iterations using Equation 4.13. The variable k, that is present in the given equations, represents the number of iterations. The Jacobian

matrix, **J**, in Equation 4.13 was calculated using Equation 4.14 which is the partial derivative of the sum of squares with respect to the constants, β.

$$\beta(k+1) = \beta(k) - \mathbf{J}^{-1} e(\beta(k)) \tag{4.13}$$

$$\mathbf{J}(\beta)_{ij} = \frac{\partial S(\beta(k))_i}{\partial \beta_j} \tag{4.14}$$

Application

The least-squares regression was implemented using Python code. The code used the Python SciPy package that is very useful for efficient numerical routines, including numerical integration and optimization [5]. In this case the curve fit algorithm, called scipy.optimize.curve_fit, from the optimization package was implemented and performed a non-linear, multivariate regression analysis.

Support Vector Machine Method

This algorithm, developed by Cortes and Vapnik [22], can learn using supervised and unsupervised methods. The algorithm learns by separating different classes in a training data set with an optimal hyperplane. The hyperplane is created by maximizing the minimum distance to the training points closest to the plane [78]. This is accomplished by mapping the input vectors into a high dimension feature space. In this space, a linear surface is constructed that separates the data.

For example, the algorithm can perform classification of two classes of data that are labeled as y_i = -1 or 1. The data labeled as -1 and +1 are plotted in Figure 4.16 as ■ and ▲ respectively. Then the algorithm is trained using these data points with their respective labels. The algorithm creates a hyperplane, defined by Equation 4.15:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{4.15}$$

The hyperplane divides the two classes in an orientation where it is as far as possible from the closest members of both classes. This hyperplane is the green line plotted

in Figure 4.16. The gray dashed lines on either side describe the "margin" which represents the distance between the closest members of each class. The margin is defined by Equations 4.16 and 4.17 [10]:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \text{ for } \mathrm{y}_i = +1 \quad (4.16)$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \text{ for } \mathrm{y}_i = \text{-}1 \quad (4.17)$$

The variable w represents the vector that is normal to the hyperplane, and b is the hyperplane intercept.

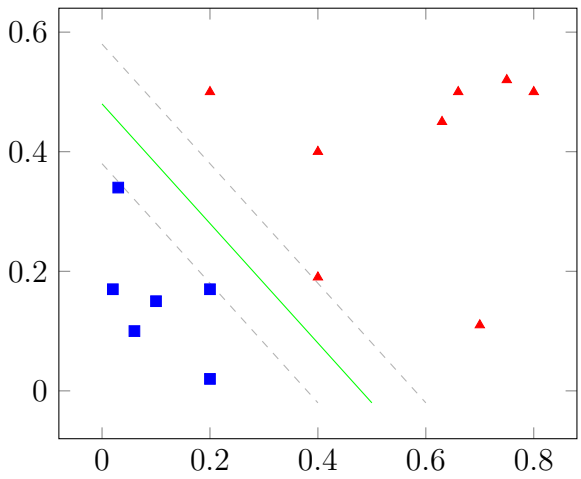

Figure 4.16: Linearly separable data with hyperplane (green line) and margin (gray dashed lines) that separate the data

The present work uses two approaches: The first approach is known as a one-class (OC) SVM that learns in an unsupervised manner. The second approach is the multi-class SVM that learns through supervision. Each appraoch can classify non-linearly separable data. The non-linear case requires a kernel and relevant parameters to map the data into a feature space that is linearly separable. There are many different types of kernels, such as radial basis, polynomial, etc.

The implementation of the algorithm can be performed on a data set that consists of the following:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) | x_i \in R^p, y_i \in \{-1, 1\}\}_{i=1}^{n} \quad (4.18)$$

The first step is to choose a kernal and hence a mapping into a high dimensional features space represented by $x \rightarrow \phi(x)$. This mapping is accomplished by computing a dot product as shown in Equation 4.19

$$Kij = y_i y_j \phi(x_i) \cdot \phi(x_j) \quad (4.19)$$

such as the Gaussian kernel

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2). \tag{4.20}$$

Then the Lagrangian function ($\mathcal{L}$ = (function to be minimized) - λ(constraint)), was implemented to find the Lagrangian multipliers. Equation 4.21 describes the primal form of the Lagrangian function for the soft margin case that was minimized with respect to $\mathbf{w}$, b, and ξ, and maximized with respect to each $\boldsymbol{\alpha}_n \geq 0$ and $\beta_n \geq 0$.

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_n(\mathbf{w}^T x_i + b) - 1 + \xi_i) - \sum_{i=1}^{N}\beta_i\xi_i \tag{4.21}$$

The soft margin approach allows for the data to not be completely seperable. The solution derived from Equation 4.21 is shown the dual form of the Lagrangian that is shown in Equation 4.22

$$\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} \tag{4.22}$$

which was maximized subject to the constraints described in Equation 4.23

$$0 \geq \alpha_i \leq \text{Cand}\sum_{i=1}^{N}\alpha_i y_i = 0 \tag{4.23}$$

to calculate $\boldsymbol{\alpha}$. This calculation required the implementation of a quadratic programming solver. For example, a convex optimization packaged such CVXOPT or the Python Sklearn package could be used to find the optimal solution and the associated Lagrangian multipliers. This calculation was significant because the non-zero α values represented the support vectors for the particular data set.

After the support vectors, $\boldsymbol{\alpha}$, and their indices were found they were then used to calculate $\mathbf{w}$ using Equation 4.24.

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i \tag{4.24}$$

Finally, the training process ends with the calculation of b as given by Equation 4.25.

$$b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{s \in S} \alpha_m y_m \mathbf{K}) \tag{4.25}$$

Then, during testing when each new point x' is presented it is classified by evaluating Equation 4.26. The result produces a -1 or +1 to indicate the particular class designation for the input.

$$y' = sign(\mathbf{w} \cdot \phi(x') + b) \tag{4.26}$$

4.2.4 Artificial Neural Network

ART and LAPART neural networks are the focus of this paper and are described extensively in Section 4.1. However, there are many other types of ANN, including perceptrons, multi-layer perceptrons, radial basis function networks, Hopfield networks, and others. One common supervised learning, multi-layer perceptron ANN is the back-propagation algorithm. This algorithm improves upon the single-layer perceptron and has the ability to perform pattern recognition on non-linear data sets.

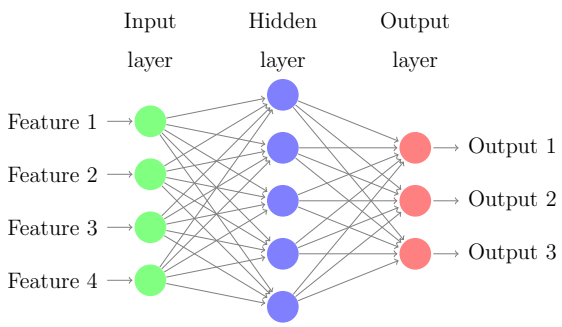

Figure 4.17: Multi-layer perceptron with an input layer, a single hidden layer, and three outputs.

Back-propagation

The back-propagation algorithm is a popular form of the multi-layer perceptron. The algorithm has an input, output, and one or more hidden layers. Figure 4.17 pro-

vides a graphical representation of the layers and associated neurons or nodes. With this set-up the algorithm has the ability to solve complex problems through supervised training that is based on the error-correction rule. The learning involves two passes through the network that are referred to as the forward and backwards passes. During the forward pass an activity input vector is applied to the nodes of the network. In addition, each of the connections contain weights, and during this pass the weights remain constant. The backward pass begins at the output layer and works its way back through the network. Along the way the weights are adjusted in accordance with the delta rule correction ($\Delta\omega_{ji}$(n)) that is applied to the weights (ω_{ji}(n)). The weights are gradually adjusted so that the the error converges to a global minimum.

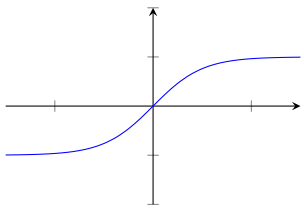

Figure 4.18: Hyperbolic tangent activation function

Forward Pass

In Figure 4.17 the arrows are pointing from left to right signifying the feedforward signal of a non-linear activation function. The activation function used in this experiment is the hyperbolic tangent, which is shown in Figure 4.18. As the signal proceeds forward from the input layer to each node in the hidden and output layers the nodes form an induced field, ν_j(n), that is defined by Equation 4.27. In this equation, ω represents the weights, x is the input vector, and n is the total number of inputs. Each neuron processed the signal by applying the activation function to the induced field to calculate the neuron signal, s_j(n) described in Equation 4.28.

$$\nu_j(n) = \sum_{i=1}^{n} \omega_{ij}(n) x_i(n) \tag{4.27}$$

$$s_j(n) = \varphi(\nu_j(n)) \tag{4.28}$$

Backward Pass

After the signal has been sent through the network it then propagates backwards,

and an error signal is calculated using Equation 4.29. In this equation d_j(n) is the desired outcome and s_j(n)is the signal from Equation 4.28. In addition to the error calculation, a total error energy (ε) for all of the neurons can be calculated using Equation 4.30.

$$e_j = d_j(n) - s_j(n) \tag{4.29}$$

$$\varepsilon(n) = \frac{1}{2}\sum e_j^2 \tag{4.30}$$

Then a similar method to the least mean square (LMS) algorithm is used to apply a correction to the weights. In this case the weight correction is proportional to the partial derivative $\partial\epsilon$(n)/$\partial\omega(n)$, which when applied to the chain rule can be expressed as Equation 4.31. The equation was simplified by applying the differential of Equations 4.30, 4.29, 4.28, and 4.27 and the outcome is expressed on the far right side of the equation.

$$\frac{\partial\varepsilon(n)}{\partial\omega(n)} = \frac{\partial\varepsilon(n)}{\partial e(n)}\frac{\partial e(n)}{\partial s(n)}\frac{\partial s(n)}{\partial\nu(n)}\frac{\partial\nu(n)}{\partial\omega(n)} = -e_j(n)\varphi_j'(\nu_j(n))x_i(n) \tag{4.31}$$

The final steps of the back-propagation algorithm involve local gradient calculations that help define the required change in the weights. The local gradient can be calculated using Equation 4.32, which is the partial derivative of the error energy with respect to the induced local field for each node.

$$\delta_j(n) = -\frac{\partial\varepsilon(n)}{\partial\nu_j(n)} = \frac{\partial\varepsilon(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial s_j(n)}\frac{\partial s_j(n)}{\partial\nu_j(n)} = e_j\varphi_j'(\nu_j(n)) \tag{4.32}$$

Additionally, the weight correction $\Delta\omega_{ij}$(n) is defined by the delta rule and is shown in Equation 4.33. This equation was simplified with substitutions that applied Equations 4.31 and 4.32 to create the result at the far right hand side of the equation.

$$\Delta\omega(n) = -\eta\frac{\partial\varepsilon(n)}{\partial\omega_{ij}(n)} = -\eta e_j(n)\varphi_j'(\nu_j(n))x_i(n) = \eta\delta_j(n)x_i(n) \tag{4.33}$$

The optimal weights can be discovered by matching the minimum error energy with a particular set of weights. To accomplish this task the algorithm implements

the gradient descent method. This approach defined by Equations 4.31 to 4.33 is represented by Figure 4.19. In this figure the error energy function is graphed with respect to the network weights. A blue arrow that originates at the blue dot shows the direction of negative gradient, and theoretically follows that direction until it reaches the global minimum where the optimal weight is located. However, this algorithm is susceptible to getting trapped in a local minimum and as a result convergence to the optimal solution may not occur.

Numerous additions to this algorithm have been implemented to improve convergence and include the learning rate and momentum term. If tuned correctly, the learning rate, which has been implemented into Equation 4.33 determines the speed at which it moves towards the optimal weight. If the learning rate is too large it may skip over the optimal solution; if it is too small it may result in an excessive number of iterations. The momentum term, which was added to the algorithm used in the research, can help the network out of the local minimum. This term adds a fraction of the previous weight update to the current one, and keeps the gradient pointing in the same direction.

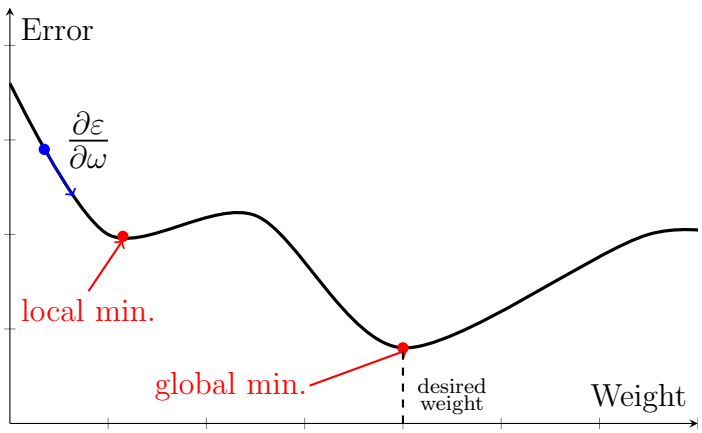

Figure 4.19: The gradient descent of error energy and weights example

The seven fault detection and diagnostice methods including ART, LAPART, threshold method, rule-based expressions, regression, SVM, and BP have been defined in Sections 4.1 and 4.2. The methods were then used in three experiments that tested their abilities to detect faults, adapt to changes in normal data, and diagnose

faults. The experiments are described in detail within Chapter 5. The results from these tests are described in Chapters 7, 8, and 9.

Chapter 5

Methodology: Experiments

There are two primary techniques for the identification of potential HVAC system faults: (1) one-time spot evaluations and (2) real-time FDD. One-time spot evaluations, often referred to as retro-commissioning (Retro-Cx), utilize design documents to evaluate and correct HVAC issues observed at the time of the evaluation. With, real-time FDD, continuous evaluations of the HVAC systems are performed; the real-time FDD assessment can be based on qualitative, quantitative, or process history techniques. The latter approach is more advantageous than a single Retro-Cx effort, because evaluations occur all of the time. Whereas Retro-Cx is implemented every 5 to 10 years [55]. Continuous FDD evaluations will ultimately reduced energy consumption for the entire life of the building, and simultaneously providing reliable occupant comfort.

The achievement of desired energy savings in commercial buildings requires a complete integration of a real-time FDD. The integration of a FDD tool has three major barriers to overcome for success. First, the tool must be simple and inexpensive to implement. Second, it must be accurate and reliable. Third, building owners must trust projected savings results to confirm investment returns. This research effort

evaluates the first and second barriers that address implementation, accuracy, and precision. Further studies may be necessary to evaluate estimated avoided costs due to early detection or mitigation of faults. To address implementation simplicity, and fault detection accuracy three experiments were devised that consider different analysis techniques and implementation strategies.

The four experiments, shown in Table 5.1, involved the development and implementation of tools for building simulation, and the detection and diagnostics of HVAC sub-system faults. Section 5.3 describes the first experiment that involved

Table 5.1: Methodology: Experiment Types

Exp.	Name	Training Data Amount	Testing Data Amount	Description Section
1	Model Calibration	2 weeks	-	5.3
2	Fault Detection	2 months	1 month	5.4
3	Adaptability	3 months	21 days	5.5
4	Fault Diagnostics	3 months	1 months	5.6

the calibration of the physical model created in TRNSYS. The model was developed and calibrated so that the analysis tools described in Chapter 4 could be tested. The second experiment, described in Section 5.4, involved fault detection analysis that was based on a training data set produced by the TRNSYS model. The data contained only normal conditions, and was tested on data that had four different types of faults. The third experiment, described in Section 5.5, represented a step beyond the previous experiment by assessing the ability of different tools to adapt to changing normal conditions. The final experiment, described in Section 5.6, applied LAPART and a SVM for fault diagnostics of the AHU.

The process history FDD tools used in experiments two, three, and four all involved the same three step process. The first step was to optimize the algorithms performance by defining the best free parameter(s). In this case, cross-validation was performed on the training set to generalization performance associated with partic-

ular parameters and define the optimal arrangement. After the parameters were set, the second step was to present the tools with the entire training data set, and each could acquire and store knowledge. In the last step, the tools were used to detect and/or diagnose faults by considering new, previously unseen data.

5.1 Experimental Procedures

The procedure for Experiments 2, 3, and 4 included cross-validation, training, and fault detection and diagnostics. The cross-validation, defined in Section 5.1.1, used the K-Folds approach to find the optimal free parameters. The next step was to train each method through the presentation of data as described in Section 5.1.2. The last step involved the detection and diagnostics of faults on previously unseen data (Sections 5.1.3 and 5.1.4).

5.1.1 Cross-Validation

Each process history method use free parameters to make the algorithm fit a given set of data. However, the free parameters for each method were unknown for the data set. Therefore, a fitting process, commonly referred to as cross-validation, optimizes the parameters to make the model fit the training data as well as possible. In the

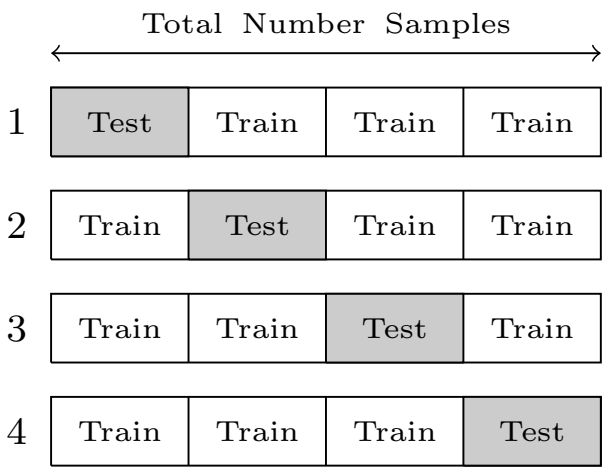

Figure 5.1: K-Fold Cross-validation ($K = 4$)

present work, the K-Folds cross validation method was implemented.

The K-Folds method is similar to random subsampling, but provides the advantage that all of the examples in the data set are eventually used for both training and testing. The method was found to be a very successful means for tuning parameters in an SVM algorithm by Duan *et. al* [27]. The process begins by arranging the entire training set into a random order; the data was then split into K equal parts or folds. This division of the data for $K = 4$ is shown in Figure 5.1. For each fold $k \in \{1,2,..,K\}$ the model was trained on the data that was located in all of the folds except for the k^{th}. Then the algorithm used the data in the kth fold for testing [85]. This process was conducted in a round-robin manner until each of the folds was used for training and testing. The average error or number of novelties for each iteration was computed and the free parameters with the lowest value were used in the proceeding training and fault detection processes.

5.1.2 Training

The term "training" refers to a learning process in which long and short term memory is developed. This process applies only to process history methods that consider past data to influence their memory. In contrast, the physical and rule-based models do not have any memory but instead rely on proven physical and thermodynamic principles. Therefore, these type of models do not require training. However, the process history methods acquire empirical knowledge about a physical phenomenon or environment through training [43]. Knowledge, according to Fischler and Firschein, is the stored information or models used to interpret, predict, and approximate responses to environmental inputs [34].

The development of knowledge during training can be achieved through supervised or unsupervised learning. Supervised learning involves three components: en-

vironment, teacher, and learning machine [43]. The environment consists of the inputs which have a fixed but unknown probability distribution. The teacher provides oversight by defining the desired response for each input in the environment. Input-output mapping functions are performed by a learning machine that develops memory to define the empirical knowledge of the environment. In contrast, unsupervised learning algorithms can gain knowledge of an environment without a teacher to define the desired mapping for a particular input.

5.1.3 Fault Detection

The fault detection process tests the previously encoded knowledge in each of the algorithms on unseen data to make predictions or classify the new input data set. FDD methods that make predictions, such as the rule-based model, and back-propagation, evaluate faults based on a residual error. Classification methods, such as ART and SVM, determine faults based on where the new inputs are located in the classification space. The results for these tests are evaluated based on the review of false alarm and true detection rates.

5.1.4 Fault Diagnostics

Fault diagnostics was performed by the LAPART and multi-class SVM algorithms. This test was conducted in experiment 4 and used training data that contained normal and fault behavior. In addition, each of the points were labeled as one of four faults or as normal. The testing process used the previously unseen data as inputs and each of the algorithms output a prediction of the particular class for each input. The outcomes from each algorithm were evaluated based on the F-scores test [85]. Also, each of the algorithms precision values were compared to identify the optimal method.

5.2 Fault Types

The experiments used five different operations conditions. The first condition was considered normal where no faults existed. The other four were considered abnormal and were labeled as faults. The faults introduced into the different sub-systems of the AHU caused sensor values to deviate from normal. The deviations are shown in Figure 5.2 for a single day of operations. Fault type A experienced an increase in

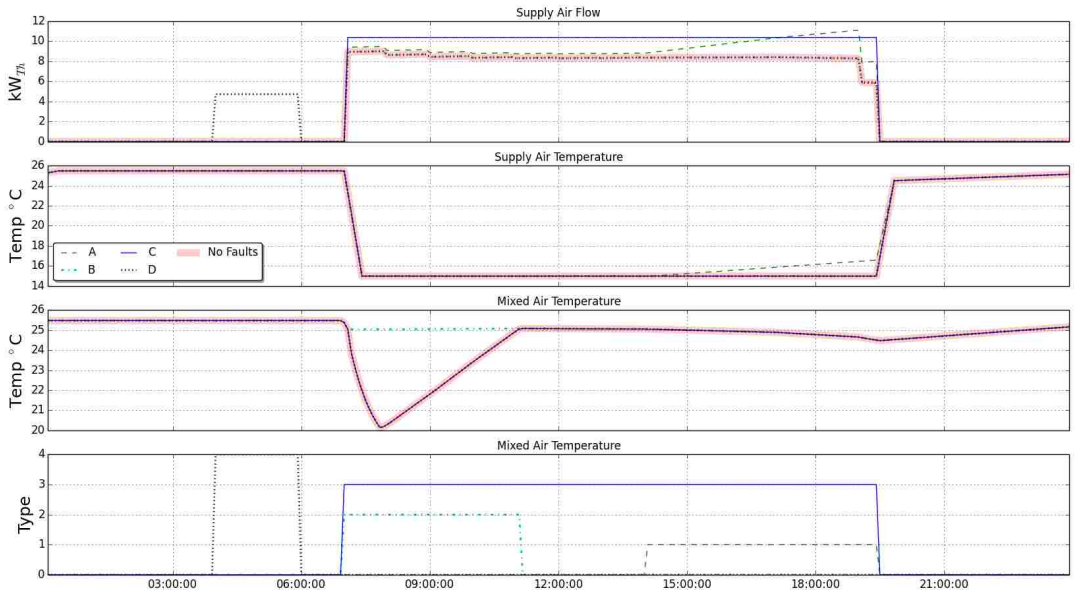

Figure 5.2: The sensor values for the normal and four fault conditions are plotted for a single day. Fault type A experienced an increase in supply air flow rate and temperature. The mixed air temperature remained at about 25°C for fault type B. The supply air flow rate was a constant value throughout the whole day for fault type C. During fault type D an off-schedule air flow rate was measured.

supply air flow rate and temperature. The mixed air temperature remained at about 25°C for fault type B. The supply air flow rate was a constant value throughout the whole day for fault type C. During fault type D an off-schedule air flow rate was measured.

5.2.1 Type A: CHW High Temperature

The type A fault caused deviations from normal supply air flow and temperature sensor values as shown in Figure 5.2. These deviations were caused by high temperatures in the CHW system that supplied the HX sub-system. The computed statistical properties of the sensor outputs for the supply air flow rate and temperature were noticeable different than normal sensor values. For instance, the average for the air flow rate was 9.8m^3/sec in the normal case and 11.2m^3/sec in the fault condition. Similarly, the supply temperature average increased from 15.5°C to 16.8°C. The fault condition, caused by the high CHW temperature, shifted the probability distributions for the supply air flow rate and temperature as shown in Figure 5.3. The mixed air temperature was not affected by the fault and remained the same as the normal sensor value.

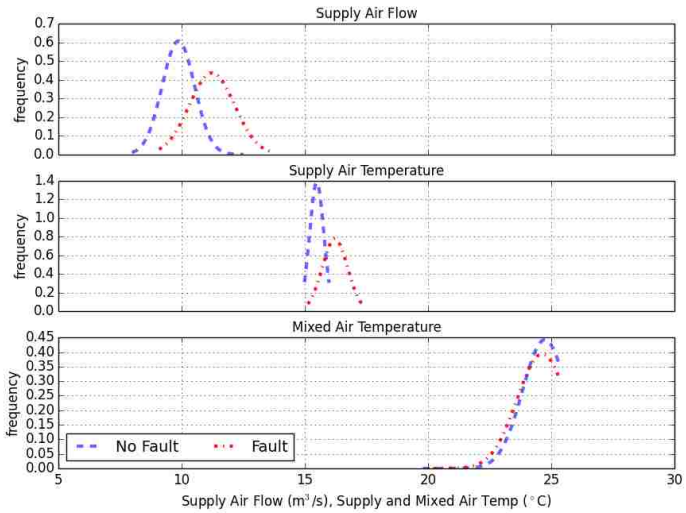

Figure 5.3: The probability distributions for type A fault and normal operation conditions for each of the sensors in the respective sub-systems.

5.2.2 Type B: Mixed Air Damper

The mixed air damper fault, labeled as Type B, caused temperatures in the mixed air sub-system to vary from normal operations. This difference in temperature was

caused by a malfunction in the the mixed air damper actuator and louver system. The malfunction did not impact the supply air flow or temperature and is evident in the probability distribution functions for the respective sensors in Figure 5.4. However, the distributions for normal and the fault condition are different in the mixed air sub-system. The average temperature in this section was recorded to be 24.7°C for the normal case and 25°C for the fault condition. The standard deviation was 0.9 for normal behavior and 0.05 for the fault condition.

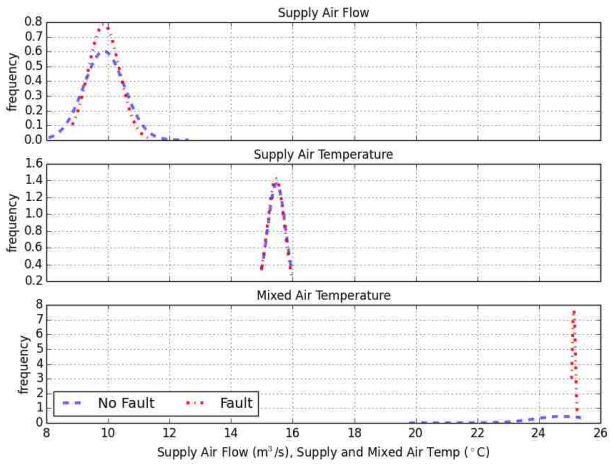

Figure 5.4: The probability distributions for type B fault and normal operation conditions for each of the sensors in the respective sub-systems.

5.2.3 Type C: VFD Fan

The VFD that controls the fan can malfunction or be set to manual by maintenance and not turned back on. In this case the air flow rate remains at a constant throughout the day and then turns off at night as shown in Figure 5.2. While during normal conditions the air flow rate

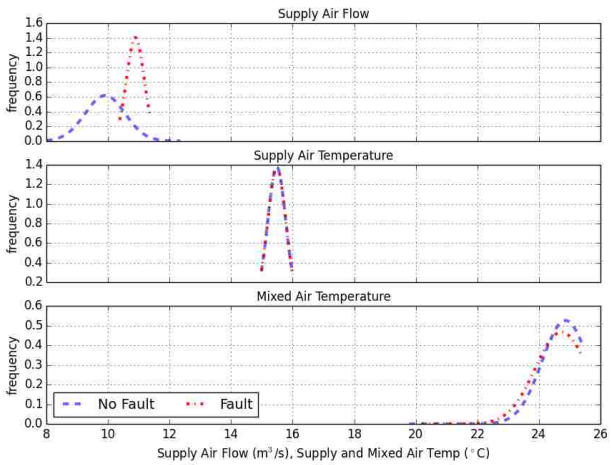

Figure 5.5: The probability distributions for type C fault and normal operation conditions for each of the sensors in the respective sub-systems.

can fluctuate. The different fan behavior between the type C fault and normal operations created different mean and standard deviation values which produced different probability distributions as shown in Figure 5.5. At the same time the other two sensors in the HX and mixed air sub-systems are nearly identical to normal operations because the type C fault did not affect them.

5.2.4 Type D: Off-Schedule Fan

The final fault, type D, occurred when the fan turned on during off-schedule hours. This could be caused by a programming error in the BAS. The air flow rate increases from 0 to about 5m^3/sec as shown in Figure 5.2 at around 4:00 or 5:00 in the morning when the system should be off. This fault caused the supply air flow rate and temperature to deviate from the normal and the distributions are shown in Figure 5.6.

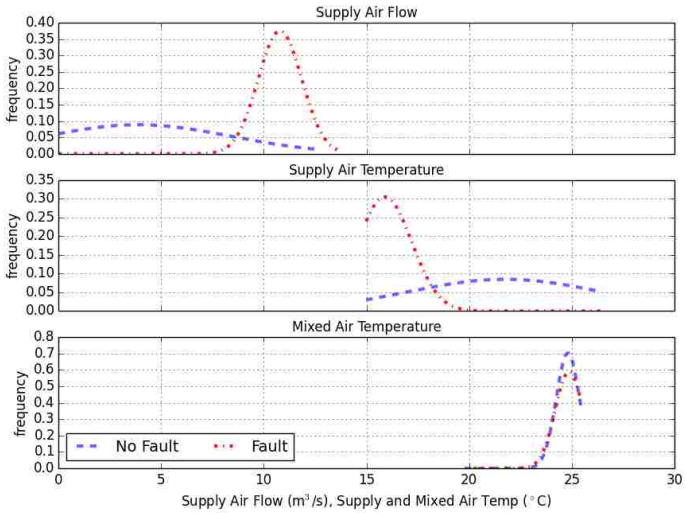

Figure 5.6: The probability distributions for type D fault and normal operation conditions for each of the sensors in the respective sub-systems.

The fault conditions defined as Type A, B, C, and D are used for fault detection and prognosis tests in experiments 2, 3, and 4. The first experiment performs a calibration of the physical and rule-based models. The physical model, created in the TRNSYS simulation studio, provides the training and testing data for the FDD tools in experiments

2 to 4. Also, the rule-based model was calibrated in experiment 1 so that it could perform fault detection tests in experiment 2.

5.3 Exp. 1: Physical & Rule-based Calibration

Both the rule-based and physical model methods required calibration to ensure that each represented actual operations well. Unlike the process history techniques, where actual data was presented to the algorithm so that the system could learn, these qual-

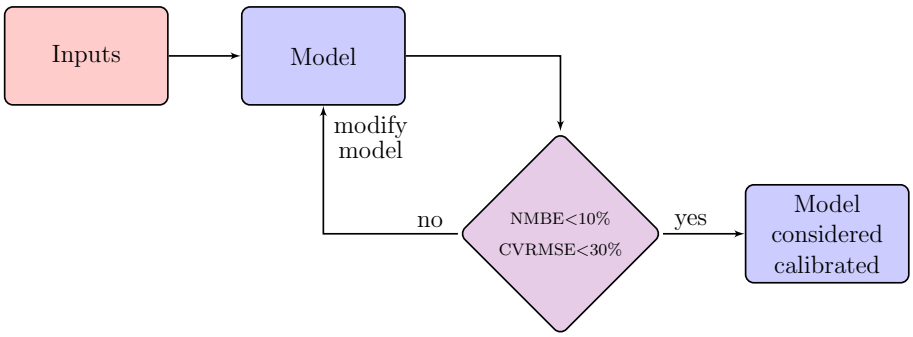

Figure 5.7: Flow diagram rule-based and physical model calibration process

itative methods required a calibration effort to verify that the model could provide accurate results. In addition, accuracy was a requirement for the physical model, because it was a source for training and testing of other FDD methods within experiments 2, 3, and 4.

The calibration process, shown as a flow diagram in Figure 5.7, began by providing the model with weather and occupancy inputs. The models were then executed and outputs were compared with the actual values to calculate the Normalized Mean Bias Error (NMBE) given by Equation 5.1

$$\text{NMBE} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{\sum_{i=1}^{n} y_i}(100\%) \tag{5.1}$$

and the Coefficient of Variation of the Root Mean Squared Error (CVRMSE) shown

in Equation 5.2.

$$\text{CVRMSE} = \frac{\sqrt{\sum_{i=1}^{n}((y_i - \hat{y}_i)^2/n}}{\frac{\sum_{i=1}^{n} y_i}{n}}(100\%) \tag{5.2}$$

The NMBE and CVRMSE are defined by ASHRAE Guidelines 2002 [6] that outlines building model calibration standards. If the NMBE is greater than 10% and the CVRMSE is greater than 30% then modifications to the model are required. However, if the opposite is true, where both NMBE and CVRMSE criteria were meet than the model was considered to represent actual well and the calibration process was concluded.

Existing literature provided examples for model calibration, and each require some degree of actual data for comparison [92, 99, 100]. In one study, performed by Lam *et al.*, an empirical method was used for the calibration of an Energy Plus model that represented a medium sized office building [67]. The process used to calibrate the HVAC system included the calculation of NMBE and CVRMSE. Another paper, by Febres *et al.*, considered the calibration of a model that simulated the performance of heating coils [33].

Similar to Lam *et al.*, the Febres *et al.* paper included the NMBE and CVRMSE calculations to determine calibration acceptance. The ASHRAE guidelines state that the NMBE and CVRMSE should be less than 5% and 15% respectively for model calibration based on monthly data. If hourly data is used for calibration then the NMBE and CVRMSE must be less than 10% and 30% respectively. This experiment evaluated the two models based on hourly data and therefore 10% and 30% for the NMBE and CVRMSE criteria were used.

5.4 Exp. 2: Fault Detection

The intent of the first experiment was to evaluate the benefits for the implementation of the FD tools on a system that had already been Retro-Cx. In this situation the data used to train the FD tools contained no faults. Therefore, the training process

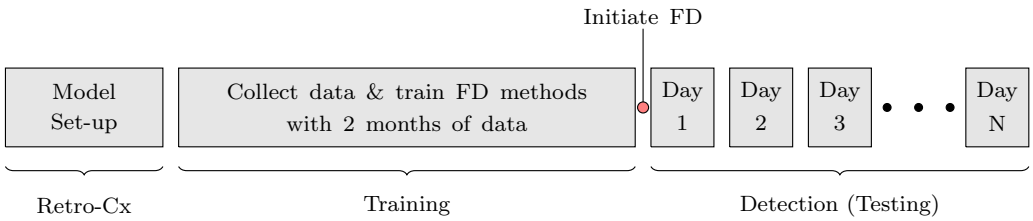

Figure 5.8: Experiment 2 timeline for model set-up, training, and fault detection.

can use unsupervised learning in order to acquire knowledge about good behavior. Then, during testing any abnormal behavior would be recognized as a fault condition. This experiment only considered fault detection and did not take the next step to diagnose the problem.

The experiment followed a specific timeline that is shown graphically in Figure 5.8. It began with the development and calibration of the TRNSYS physical model that is described in Section 5.3. This model represented the actual case where an AHU was Retro-Cx. The TRNSYS model was then run and simulated normal operations for two months. The outputs from the simulation were collected and stored in a MySQL database. Then, the fault detection tools used the data to learn system

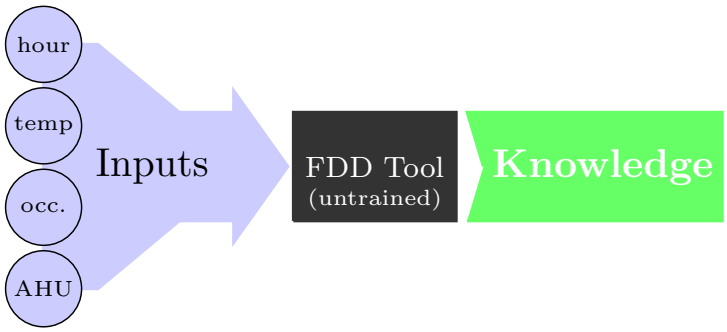

Figure 5.9: The model learned based on presentation of inputs that did not contain any faults.

operations. This was followed by the testing phase where a month of data were presented to and the various FD tools that attempted to distinguish the difference between normal and fault conditions.

In the present work the different tools considered these inputs: hour of the day, outside air temperature, occupancy, and AHU sensor data (mixed air temperature, supply air temperature, and supply air flow rate) for training and fault detection. The learning process was performed differently for each method, but the overall goal was to acquire knowledge of the AHU system behavior. This training process is described in Figure 5.9 where inputs were provided to the model and knowledge was gained and stored.

After the training process during which knowledge was acquired, the models were ready to process previously unseen data and classify normal and fault behavior. This process, shown graphically in Figure 5.10, was conducted on a daily basis as indicated in the timeline shown in Figure 5.8. It began with the presentation of new data to the trained models. The trained models had knowledge of system behavior that was stored in their memory and could perform a binary classification that labeled the new inputs as normal or as a fault.

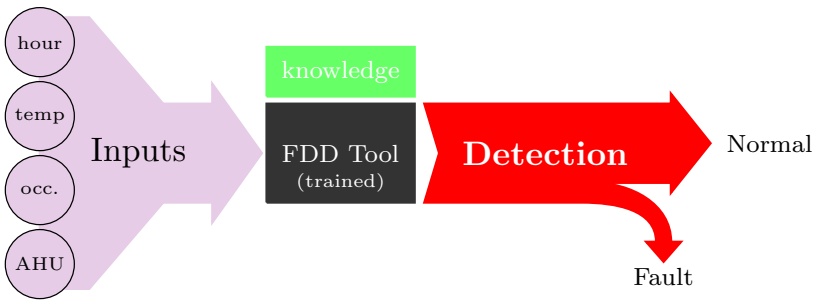

Figure 5.10: Previously unseen data were presented to the model that had acquired knowledge during training and it classified normal and fault conditions in the three sub-systems of the AHU.

5.5 Exp 3: Adaptability

Experiment 3 was the continuation of the previous experiment. In this case FD tools were evaluated on their abilities to adapt to normal changes in the input features, and provide accurate fault detection results. This experiment considered the case where the AHU sub-system outputs changed due to new occupancy schedules caused by the start of the fall semester school schedule. This change in schedule altered the performance of the system to a degree that the FD tools would define the normal changes as faults. Therefore, the FD tools must adapt to the new normal behavior, defined by a human expert, quickly so that the number of false alarms are maintained at a low value.

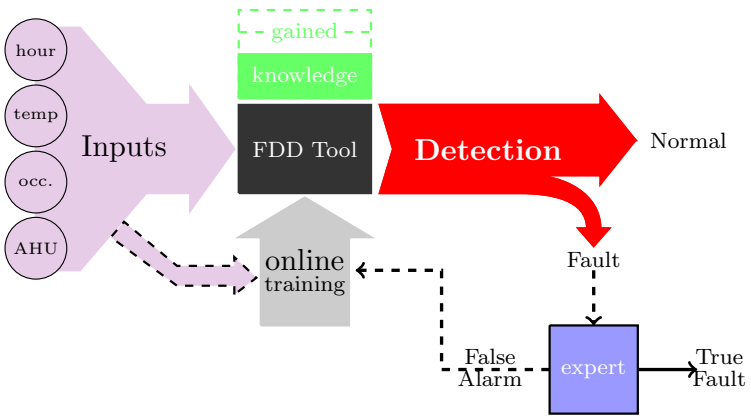

Figure 5.11: The adaptive detection and training process involved the detection of normal, true faults, and false alarms. The false alarm indices were sent to an online training system that found the associated inputs and trained the model with the new inputs.

The training process was the same as described in experiment 2. The detection of faults began with the presentation of the new inputs into the model as shown in Figure 5.11. The algorithm then identified normal and fault conditions. A human expert reviewed the fault conditions and determined if the fault was true or if it was a false alarm. The data indices associated with the false alarm were collected and routed to an online training system. The online system found the associated inputs and provided it to the model for updated training. The model than gained knowledge based on the new training

and the process repeated.

5.6 Experiment 4: Fault Diagnostics

The previous experiments (2 and 3) evaluated the potential of fault detection based on a training data set that contained no faults. The fourth experiment involved both fault detection and diagnostics of the AHU faults. In this case, the diagnostics required some knowledge of the fault type. Therefore, either an expert must label faults within and online training system or the diagnostic tools can be initially trained with data from a TRNSYS simulation that contained faults with labels. In this experiment the former case is implemented and the LAPART and multi-class SVM are trained on TRNSYS simulated data with faults. The general

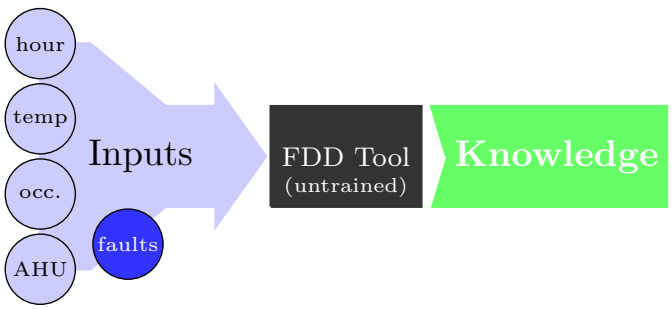

Figure 5.12: The models were trained based on hour of the day, outside air temperature, occupancy, AHU sensors, and type of fault.

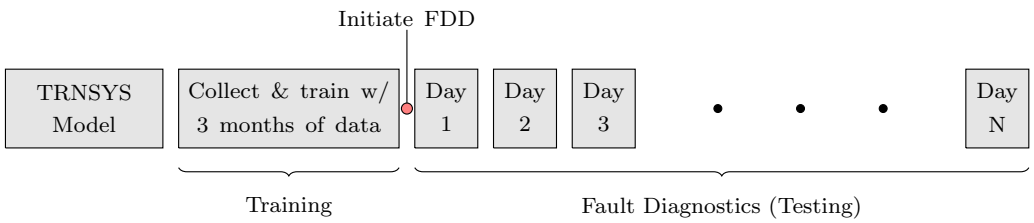

Figure 5.13: Timeline for Experiment 4 that included the model development, training of diagnostic tools, and then testing on previously unseen data.

process for the experiment includes the TRNSYS model development, training of the FDD algorithms, and then the implementation of detection and diagnostics in the testing phase as described in the timeline in Figure 5.13.

The training process included the presentation of inputs to the FDD tools. Unlike the previous experiments, the inputs included faults. In addition the faults and normal data were labeled accordingly to signify the input feature status. The tools learned form the input data and gained knowledge as shown in Figure 5.12. The knowledge was then stored in a database for future use during the testing phase.

During the testing process new inputs, that did not include labels, were presented to the models. Based on the gained knowledge the model was able to determine if the particular input was normal or a specific fault type as shown in Figure 5.14. The diagnostic results for each fault type were recorded and analyzed to discover each methods effectiveness. The analysis method used the F-scores test to compare the LAPART and SVM algorithms on the given data set.

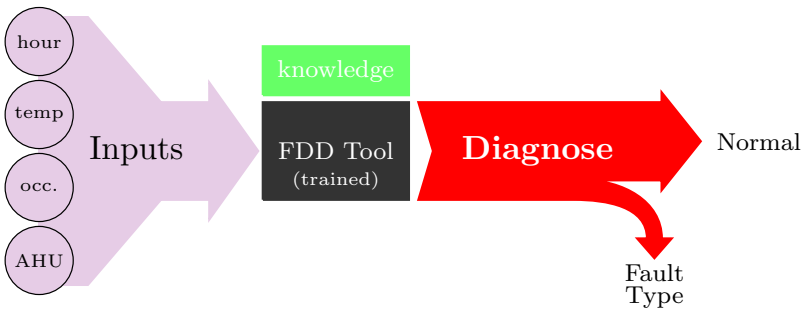

Figure 5.14: The diagnostic process used the trained models to determine the status of the particular input.

5.7 Analysis of Results

The analysis of FDD results for experiments 2, 3, and 4 considered the decision quality of each method. The decision quality was based on accuracy, precision, and true positive and false positive rates. These values are computed based on the following recorded outcomes [85]:

1. True positives (TP) is an observation that was identified by a given algorithm, and is a real instance of a feature.

2. False positives (FP) is when the algorithm identified an observation as a real instance of the feature but in fact, it is not.

3. True negatives (TN) is the case where the observation is not a real instance of the feature and an algorithm identifies it as such.

4. False negatives (FN) is the case where the algorithm does not identify the observation as a real instance of the feature but in fact, it is.

These outcomes can be used to calculate the accuracy which is the fraction of correct cases. The basic calculation is performed by dividing the number of correct decisions by the total number of cases, as shown in Equation 5.3:

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \tag{5.3}$$

However, accuracy cannot provide reasonable interpretation of performance. For instance, two algorithms can produce the same accuracies but provide different correct and incorrect decision results. The results from one algorithm might miss actual faults, and another algorithm may detect a fault that did not occur [81] which is not defined by the accuracy value. Therefore, for the research experiments the performances of the different algorithms were compared using more appropriate criteria that defined the sensitivity and specificity of each algorithm. This comparison criteria is defined by the receiver operating characteristic (ROC) principles for two-class analysis (Section 5.7.1) and the F_1-score statistic for multi-class analysis (Section 5.7.2).

5.7.1 Receiver Operating Characteristics Principles

ROC curves are used to organize, and define the best classification method [32].

Table 5.2: Confusion Matrix Table Example

		Estimate		
		$\hat{y}$=1	$\hat{y}$=0	Σ
Truth	y=1	TP	FN	N_+=TP+FN
	y=0	FP	TN	N_-=FP+TN
	Σ	N_+=TP+FP	N_-=FN+TN	

This approach was first used during World War II for the purposes of correct detection of Japanese aircraft from radar signals [111]. They have also been popular in signal theory [25, 35], medicine [134, 28, 26], and machine learning [109, 9, 65, 8, 53]. In each of these applications, the ROC graphs helped characterize the abilities of a classifier through the statistical review of four possible outcomes TP, FP, TN, and FN.

The positive and negative results for each instance are given by a binary value of 1 or 0 respectively. For the research experiments a positive result translates to a fault, and a negative outcome is considered normal equipment behavior. Therefore, the FDD methods used in experiments 2 and 3 were judged based on a two-class binary problem. A metrics known as the confusion matrix, is shown in Table 5.2. It provides a representation for the dispositions of the set of instances. The correct estimates produced by the given classifier are represented by the values TP and TN that are along the major diagonal, and TP and FP numbers represent the errors or "confusion" between the classes [32].

For example, the FP, which is also referred to as a false alarm, arises when the estimate is $\hat{y}$=1 but the truth is y=0. This can be translated to a rate, known as the false positive rate (FPR), and is shown in Equation 5.4:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \approx P(\hat{y} = 1 | y = 0) \tag{5.4}$$

This type of error is known as the type 1 error and can be considered the probability of false alarm. If the outcome from an estimate is $\hat{y}$=1 and the actual value is also

y=1 then it is a TP. The true positive rate (TPR) is given by Equation 5.5:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \approx \text{P}(\hat{y} = 1 | y = 1) \tag{5.5}$$

The TPR is otherwise known as the probability of detection, the sensitivity, the hit rate, or recall.

FPR and TPR calculated using Equations 5.4 and 5.5 respectively define ROC space (Figure 5.15). Figure 5.15 describes an example where the FPR and TPR results from four sample FD methods were plotted. The best possible FD method would yield a point in the upper left corner as indicated by the "perfect classification" label. The "perfect classification" would occur at 100% sensitivity with no FN, and 100% specificity which means that no FP where encountered. The graph also shows a diagonal line from coordinate (0,0) to (1,1), which represents where the results of a random guess would be plotted. This diagonal divides the ROC space, and if a method produces a result that plots above the line it is considered a good classification method. If a point ends up below the diagonal line then it is considered a poor result, which means that it is worse than random [32].

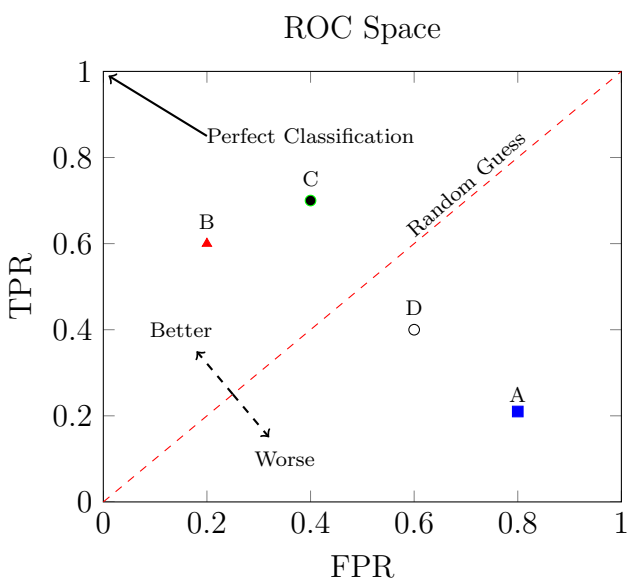

Figure 5.15: ROC space with example plots for four prediction methods

A critical review for the FD methods is the precision and TPR analysis. This analysis method is considered more effective than the accuracy calculation for the current work, because the amount

of normal data is very large in comparison to the quantity of faults. Precision is the fraction of detections that are actually positive, as shown in Equation 5.6:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.6}$$

TPR is a measure of the positive classifications that were detected correctly. For this type of plot the ideal classifier would be positioned at point (1,1) as shown in Figure 5.16. Sample point F, and H represent a well performing classifier, and sample H is the best outcome for the points shown. Sample point E is not as good as F, but is better than the worst case at sample point G.

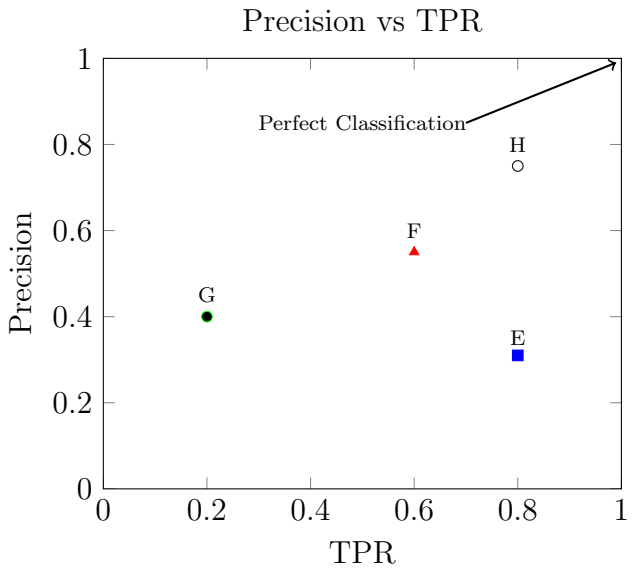

Figure 5.16: Precision and TPR plot example where the ideal case is located at point (1,1).

5.7.2 F_1-Score Statistic

Fault diagnostics performed in Experiment 4 was a multi-class classification task. Therefore, the evaluation of the algorithm's performance was based on the F_1-score statistic. The F_1-score is based on the harmonic mean of the precision (P) and recall (R) as shown in Equation 5.7:

$$\text{F}_1 = \frac{2(\text{PR})}{\text{P} + \text{R}} \tag{5.7}$$

The implementation of the harmonic mean provides for what is called the macro-averaged F_1-score. The macro-averaging process pools across the classes and then

computes an effectiveness measure. In contrast, the less desirable, micro-averaging calculation computes an average over the classes. The macro-averaging test was beneficial and used in the present work because it distributes equal weight to each of the classes [76]. The macro-averaged F_1-score can be calculated using Equation 5.8:

$$F_1 \text{ score} = \frac{\sum_{c=1}^{C} F_1(c)}{C} \tag{5.8}$$

In this equation F_1(c) is the F_1 score obtained on the task of distinguishing class c from all others [85]. The C variable represents the number of classes considered.

The F_1-Score statistic is used to evaluate the performance of the fault diagnostic test in Experiment 4. Results from Experiments 2 and 3 are evaluated using the ROC principles defined in Section 5.7.1. Experiment 1 results were based on model calibration guidelines defined by ASHRAE (Section 5.3). Finally, the results for the four experiments are described in Chapters 6, 7, 8, and 9.

Chapter 6

Experiment 1: Model Calibration Results

Unlike the process history methods that learn system behavior on their own through the presentation of training data, the rule-based and physical model methods require the user to fine tune parameters to create a well functioning representation. The calibration process for the two models considered ASHRAE Guidelines as described in Section 5.3. This required the calculation of NMBE and CVRMSE followed by the modification of model parameters to meet the defined criteria, which where 10% and 30% respectively. The first section in this chapter, Section 6.1, defines the statistical properties of the actual data inputs and outputs for the AHU system. Section 6.2 reviews the results for the **physical model** created in TRNSYS simulation studio. Section 6.3 describes the process and results for the **rule-based** model calibration.

6.1 Actual AHU 2 & Weather Data Statistics

AHU 2 is a dynamic system that has thermodynamic and physical properties which are monitored by temperature and flow sensors. The system has a schedule, controlled by the DDC, that turns "on" at seven in the morning and then "off" at seven thirty at night. Night operations dictate that the dampers should not modulate, the chilled water flow should remain zero, and the fan motor should remain in the off position. The temperature and flow sensors monitor transient and steady state conditions. During the "on" and "off" conditions different statistical properties exist. The calibration process involved the review of actual versus modeled data for nearly 3,146 data points between July 10 and July 31, 2014. Over this 22 day span actual weather conditions where measured by onsite sensors and included outside air temperature, relative humidity, and solar irradiance. Figure 6.1 to 6.3 describe the distribution of data for the three measured values during "on", "off", and "overall" schedule conditions. The boxplots describe the maximum, minimum, upper and lower quartiles, and the median for the different data sets (as shown in Figure 6.1).

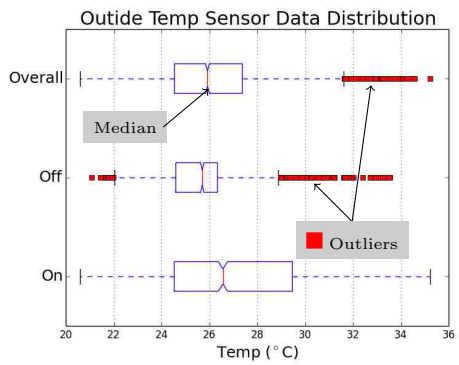

Figure 6.1: Outside temp boxplot

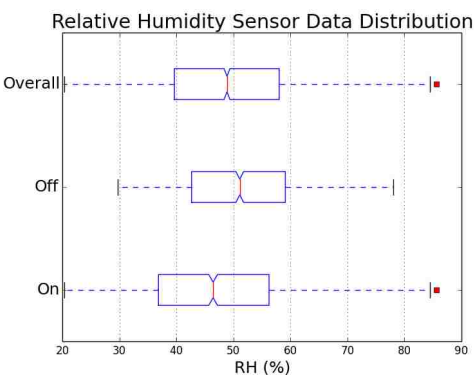

Figure 6.2: Relative humid. boxplot

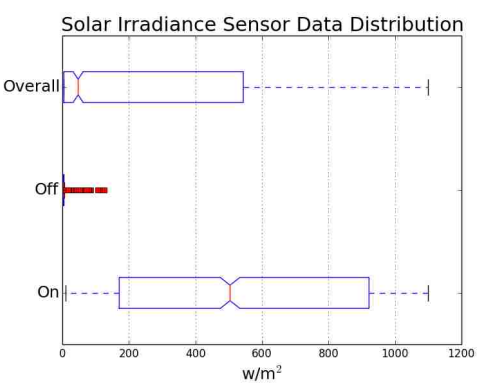

Figure 6.3: Solar Irradiance boxplot

The boxplot for outside air temperature

(Figure 6.1) shows a maximum value of 20.59°C, a minimum of 35.22°C, and a median value of 25.9°C. The relative humidity results, shown in Figure 6.2, have similar distributions for the different times of the day. The median values for relative humidity are 46.4%, 51.1%, and 48.8% for "on", "off", and "overall" respectively.

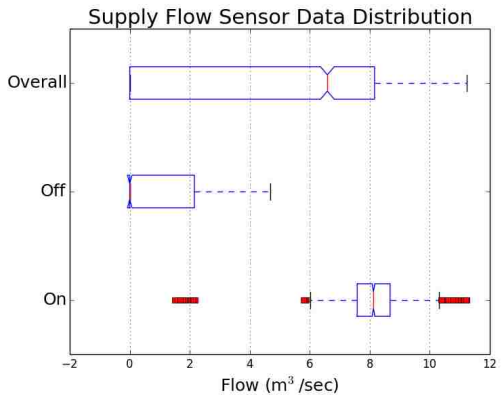

Figure 6.4: Supply flow boxplot

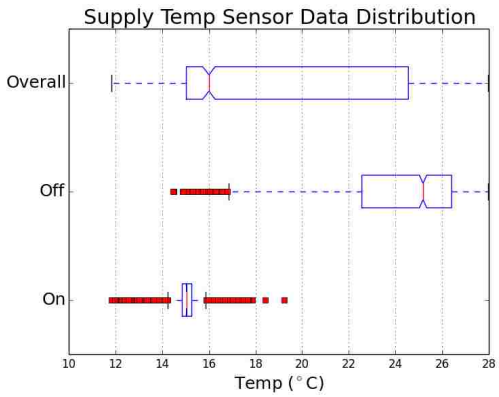

Figure 6.5: Supply temp boxplot

The "on" and "off" median value for solar irradiance is around $550w/m^2$ and $5w/m^2$ respectively as shown in Figure 6.3. The measured value should be zero at night, but the sensor has an offset of about $5w/m^2$.

The weather data were inputs to the actual and model systems. The outputs for the AHU 2 system were supply air flow, supply air temperature, and mixed air temperature. The statistical properties for these actual outputs are shown in Figures 6.4, 6.5, and 6.6.

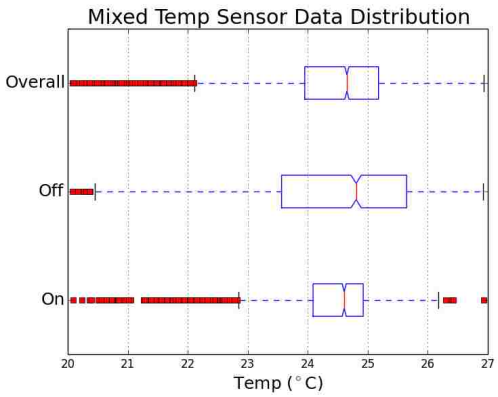

Figure 6.6: Mixed temp boxplot

It is clear that the statistics various greatly from the "overall" data set to the "on" and "off" subsets for the supply air flow data. For instance, Figure 6.4 describes the "on" and "off" data sets to have a median value of 8.12 and 0.0 m^3/sec respectively. Additionally, the maximum and minimum

supply flow rate is 11.2 and 0.0 m^3/sec for the "on" and "off" cases respectively.

The supply air temperature statistics, described in Figure 6.5, show that the maximum and minimum temperatures are 27.9 and 11.8°C respectively. It is evident that the statistics vary from the "on" to "off" schedule conditions. For example, during the "on"condition the supply air temperature has an average value of about 15.1°C with a variance of 0.41, whereas, the average was about 24°C with a variance of 8.93 when the schedule is in the "off" condition. The significant jump in variance indicates that when the system is "on" it remains within a defined value very well and when the system was "off" condition tend to vary.

The last sensor data set, mixed air temperature, was characterized by a statistical distribution that is described by Figure 6.6. The variance and standard deviations were different for the "on" and "off" cases. The variance for the "on" condition was significantly smaller than when the system was "off". This indicates that the system was able to control the temperature when the equipment was running and allowed for temperature to drift when it was "off".

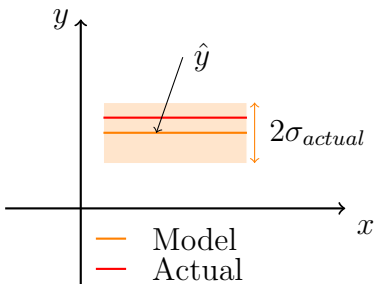

Figure 6.7: Physical model range for calibration review

6.2 Physical Model Calibration

The TRNSYS **physical model** was described in Section 3.3. The intent of the model was to create an accurate representation of AHU 2 system so that the outputs could be used for training and testing of the FDD tools. The model was able to predict supply flow, supply temperature, and mixed air temperature outputs based on outside air temperature, relative humidity, solar irradiance, and occupancy inputs.

The outputs were compared with actual sensor data to determine compliance with ASHRAE guidelines for model calibration. NMBE and CVRMSE were calculated at each calibration iteration as described in Figure 5.7. The final NMBE and CVRMSE values were required to be below the defined thresholds of 10% and 30% respectively. Additionally, the actual and modeled temporal data for each of the sensor values were compared in plots. The model outputs were given a range of plus or minus the standard deviation as shown in Figure 6.7. This range was used as a tool to trouble shoot the calculated NMBE and CVRMSE error values.

The supply air flow rate had an initial NMBE and CVRMSE of -48.8% and 908.3% (Table 6.1) respectively. These values did not comply with the ASHRAE guidelines, and a sample two day plot that shows the discrepancies is shown in the top plot in Figure 6.8. It is evident that the model over predicted the air flow rate. Therefore, modifications to the internal load were made. This included a more detailed review of office computers that was initially too high and thus reduced in the model. This change in the model reduced the supply air flow rate to an acceptable level that represented actual well as shown in the bottom plot in Figure 6.8. In addition, the NMBE and CVRMSE were calculated to be -0.69% and 14.4% respectively. These values now complied with the ASHRAE guidelines.

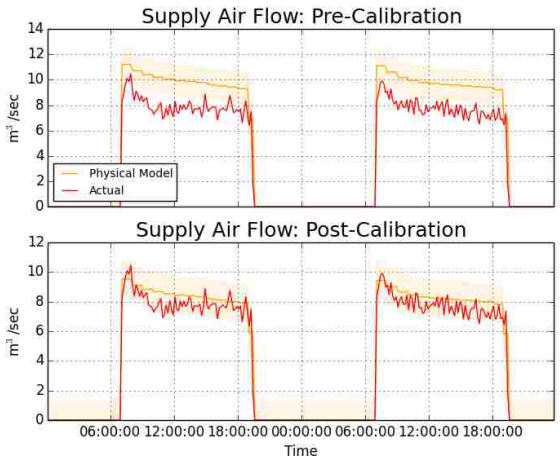

Figure 6.8: Physical model supply air flow results versus actual

The supply air temperature output produced by the initial simulation had an NMBE and CVRMSE of -8.4% and 175.8% respectively (Table 6.1). The NMBE

value complied with the guidelines, but the CVRMSE did not. The simulation output matched with the actual when the AHU system was on, but did not match well when the system was off as shown in the top plot of Figure 6.9. A more thorough review of the model set up revealed the supply air section was not properly linked to the mixed air section. In the initial case, the temperature during the night was based on the building temperature only. Instead, the supply air temperature should match the mixed air value. The fix was made an the model was executed again. A sample of the results is shown in the bottom plot of Figure 6.9 where the actual and model matched well during the day and night. The improvement produced a NMBE and CVRMSE of -0.97% and 20.1% (Table 6.1).

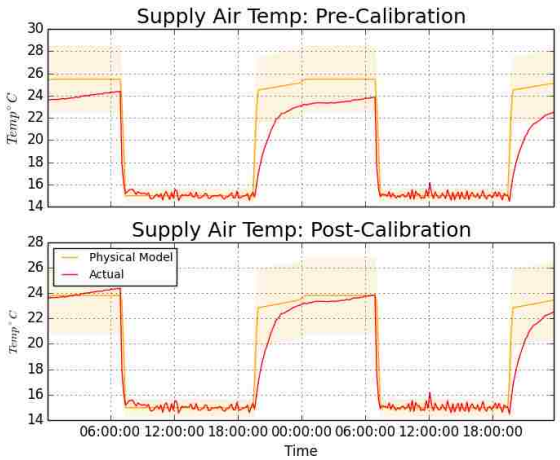

Figure 6.9: Physical model supply temp results versus actual

The mixed air temperature simulation output had an initial NMBE and CVRMSE of -0.58% and 11.9% as shown in Table 6.1. In this case, both the NMBE and CVRMSE values complied with the ASHRAE guidelines after the first iteration. Therefore, no further modifications were required to calibrate the mixed air section model.

Table 6.1: Physical Model: Calibration Results (NMBE & CVRMSE)

	Damper Section		HX Section		Supply Fan Section	
	NMBE	CVRMSE	NMBE	CVRMSE	NMBE	CVRMSE
Pre-Calibration	-0.58%	11.9%	-8.4%	175.8%	-48.8%	908.3%
Post-Calibration	-0.58%	11.9%	-0.97%	20.1%	-0.69%	14.4%
Threshold	10%	30%	10%	30%	10%	30%

The sub-systems within AHU 2 were modeled and calibrated based on the process

defined in Section 5.3 and ASHRAE 2002 guidelines [6]. The NMBE and CVRMSE results for the initial and final simulations are shown in Table 6.1. The supply air flow rate and temperature improved considerably from the initial to the final iterations. The mixed air temperature complied with ASHRAE guidelines immediately, and therefore did not require any modifications. The model was successfully calibrated and ready for use within Experiments 2, 3, and 4.

6.3 Rule-Based Model Calibration

The **rule-based** model, defined in Section 4.2.2, was developed to perform as a FDD tool for AHU 2. The model was based on a set of rule-based expressions. These expressions had initial parameters that were based on the design document specifications. The initial development of the expressions made assumptions that the design documents represented reality. The intent of the calibration process is to review these initial assumptions and modify where necessary. Modification requirements were identified through the calculation of the NMBE and CVRMSE values. Additionally, plots of actual and modeled results were reviewed to help trouble shoot during the calibration modification process that was defined in Section 5.3.

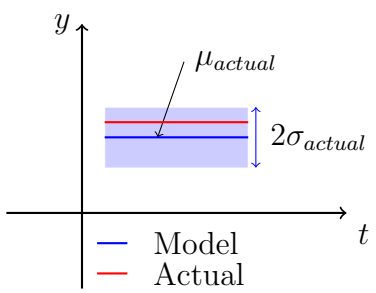

Figure 6.10: Rule-based prediction range used for calibration trouble shooting and also for FD.

The calibration process began by comparing actual data from AHU 2 with results from the rule-based expressions defined by Algorithm 3. These results are labeled as pre-calibration data and the NMBE and CVRMSE were calculated and compared with their respective thresholds of 10% and 30% respectively as shown in Table 6.2. Table 6.2 provides a breakdown of the results for each sensor in the three different

Table 6.2: Rule-Based: Calibration Results (NMBE & CVRMSE)

	Damper Section		HX Section		Supply Fan Section	
	NMBE	CVRMSE	NMBE	CVRMSE	NMBE	CVRMSE
Pre-Calibration	2.72%	97.93%	8.28%	298.11%	-49.96%	1798.8%
Post-Calibration	0.67%	24.33%	0.73%	26.37%	-0.04%	1.64%
Threshold	10%	30%	10%	30%	10%	30%

sub-systems of the AHU. Unfortunately, none of the three outputs initially complied with ASHRAE Guidelines, which required the NMBE and CVRMSE to be below 10% and 30% respectively. Therefore, calibration was required to modify parameters in the model so that an optimal fit with actual values is achieved.

Similar to the physical model approach, a range was placed around the rule-based result of plus or minus the standard deviation of the actual data. In this case, the rule-based expression was updated to use the mean value (μ) of the three sensors for the "on" schedule condition as the basis for the predicted value. Figure 6.10 describes the general form of the model output in comparison with actual. The predicted supply flow rate was initially set to be 75% of the maximum flow rate defined by the design specifications. The supply and mixed air temperatures were dynamic, and were based on return and outside air temperature conditions as shown in Algorithm 4.

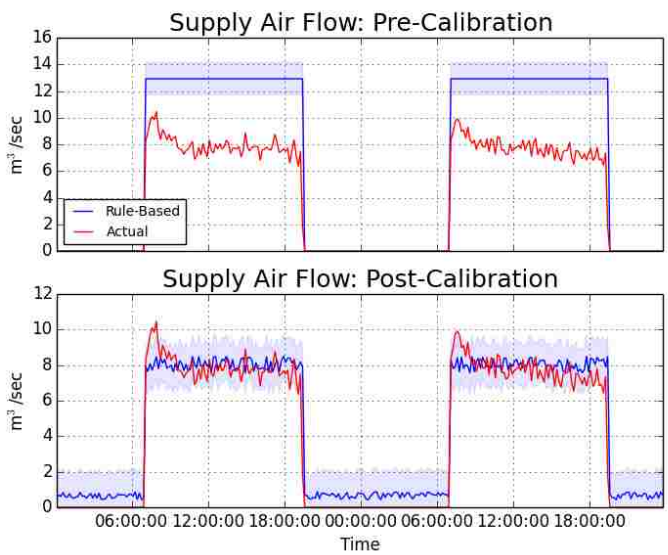

Figure 6.11: Rule-based supply air flow rate pre and post calibration results versus actual sensor values

The supply fan section, which was represented by data from a air flow sensor, had a pre-calibration NMBE value of -49.9% and a CVRMSE of 1798.8%. This CVRMSE was very high, because the initial model assumed a constant flow rate of about 13m^3/sec which was described in Algorithm 3 in Section 4.2.2, while the actual value had an average value of about 8m^3/sec. This deviation from the actual is shown in the top graph of Figure 6.11. Modifications to the model included the reduction in the air flow rate by setting the actual mean value as the supply flow rate.

The supply flow rate expression was changed from 75% of the maximum flow rate (Algorithm 3) to the mean flow rate shown in Algorithm 4. The mean flow rate was calculated to be 8.06m^3/sec and 1.34m^3/sec for "on" and "off" schedule conditions respectively. Therefore, the air flow rate reduced from 13m^3/sec to 8.06m^3/sec. After this single modification, the NMBE and CVRMSE improved to -0.04% and 1.64% respectively, and complied with ASHRAE guidelines. This change provided an improved fit to the actual data as shown in the bottom plot of Figure 6.11.

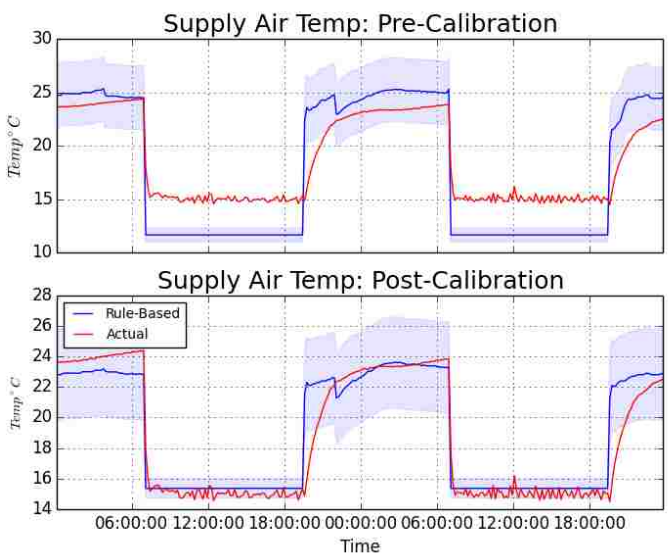

Figure 6.12: Rule-based supply temp pre and post calibration results versus actual sensor values

The supply air temperature model did not match well with the actual results. The pre-calibration NMBE and CVRMSE were calculated to be 8.28% and 298.11% respectively. The NMBE did comply with guidelines, but the CVRMSE was well above the acceptable value of 30%. The top graph in Figure 6.12

provides an example of the initial discrepancies for two days of data. It is clear that a shift in temperature during the day would provide better results. Therefore, the "on" condition supply air temperature was changed from the initial value of 13.3°C, which was set from the design specifications, to the calculated mean of 15.1°C. The "off" schedule conditions were given an offset of +2°C to better match the actual values. The changes improved the NMBE to 0.73% and the CVRMSE to 26.37% and each complied with the guidelines. The modifications changed the goodness of fit of the model as shown in the bottom plot of Figure 6.12.

The mixed air temperature output from the model, initially had a NMBE and CVRMSE of 2.72% and 97.93%. The NMBE met the requirement for the guidelines, but the CVRMSE of 97.93% was well above the threshold of 30%. Modifications to the "on" schedule condition were made by setting the output to be the mean value (24.34°C) of the actual. The "off" schedule expression was changed from the average difference of the outside and return air temperature to the return air temperature minus an offset of about 1.5°C. Figure 6.13 describes two days of the pre-calibration results in the top graph and the post-calibration in the bottom graph. The changes made during the calibration process provided significant improvements to the expressions. The NMBE and CVRMSE for each of the outputs were able to fall below their respective thresholds to 0.67% and 24.33% respectively.

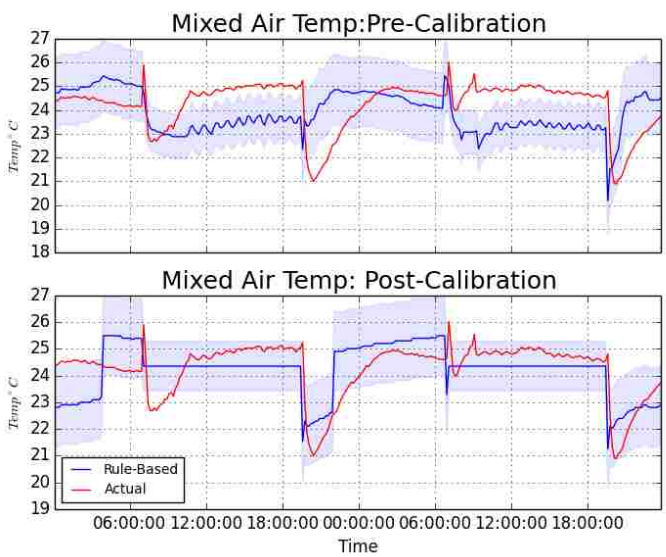

Figure 6.13: Rule-based mixed air temp pre and post calibration results versus actual sensor values

The results from the calibration process were used to update the expressions found in Algorithm 3. The final expressions that include the modifications is described in Algorithm 4. The main changes included the modification of the "on" schedule values so that they equaled the statistical mean of the actual data. In addition, new offsets were applied to the mixed and supply air temperature values for the night time operations.

Algorithm 4 Rule-based expressions post calibration

Require: hour of the day (hr), outside and return air temperature

Ensure: predict low and high supply air flow, supply air temp, and mixed air temp

if hr >= 7 AND hr <= 19.5 **then**

$m_{supply} = \bar{m}_{actualsupply} \pm \sigma_{m_{supply}}$, $T_{supply} = \bar{T}_{actualsupply} \pm \sigma_{T_{supply}}$

if $T_{return} > T_{outside}$ **then**

$T_{mixed} = T_{outside} \pm \sigma_{T_{mixed}}$

else

$T_{mixed} = T_{return} \pm \sigma_{T_{mixed}}$

end if

else

$m_{supply} = 0$

if $T_{return} > T_{outside}$ **then**

$T_{mixed} = T_{return} \pm \sigma_{T_{mixed}}$

$T_{supply} = T_{mixed} \pm \sigma_{T_{supply}}$

else

$T_{supply} = T_{return} \pm \sigma_{T_{supply}}$

$T_{mixed} = T_{return} \pm \sigma_{T_{mixed}}$

end if

end if

This experiment calibrated the physical and rule-based models. The physical model was updated to match with actual values and provide accurate simulation

results for Experiments 2, 3, and 4. The rule-based model was calibrated using the same methods as the physical model. The expressions were updated so that the outputs matched well with the actual values and therefore ready to be used as a fault detection tool in Experiment 1. The development and calibration of the rule-based expressions followed an arbitrary process. The process also required considerably time and effort to develop.

Chapter 7

Experiment 2: Fault Detection Results

Experiment 2 tested the **fault detection** abilities of seven different tools. The tools trained on a data set that contained 17,568 normal data points and 0 faults as shown in Figure 7.1. After the tools had gained knowledge during the training phase, a new set of data that contained 7,242 normal and 1,891 faults was presented to the FD tools. Each tool evaluated the features within each data point and provided a binary prediction. The binary prediction was either normal behavior or a fault.

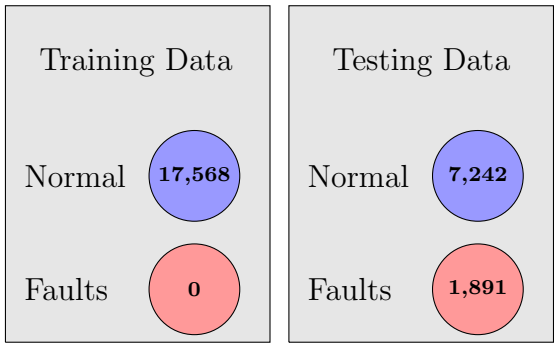

Figure 7.1: The training dataset contains 17,568 normal and 0 fault and the testing set has 7,242 normal and 1,891 faults.

The training process used simulated data for a two month span. It began on June

1, 2014 and ended on July 31, 2014. Over this time period typical meteorological year (TMY) weather data and occupancy levels were collected and used as inputs into the physical model simulation. The outputs from the model represented the sensor values for each of the sub-systems (supply fan, heat exchanger, and mixed air section). The simulation outputs and the TMY weather were then used as inputs into the different models to train and then perform fault detection as described in Section 5.4 (Figure 5.9 and 5.10).

The results from each tool were broken out by fault type (Section 5.2) and were evaluated based on the probability for detection and false alarm, as well as precision as defined in Section 5.7.1. This required the calculation of TP (True Positive), FP (False Positive), FN (False Negative), and TN (True Negative) values. The calculated values were then used to compare the methods on a particular fault and their overall performance.

7.1 Qualitative Method Results

The qualitative method is probably the most commonly used FDD tool for HVAC systems. In this experiment a threshold FD system was implemented. While this method is easy to implement, understand, and maintain, the hypothesis is that it has a performance level that is will below all other methods.

7.1.1 Threshold Method

The threshold FDD method was described in Section 4.2.1. The method can be found in many BAS to monitor HVAC equipment. The basic approach for AHU 2 is described in Algorithm 2, and includes fan status, freeze protection, and filter status. The supply fan status alarm system was the only one that could be applied

to this experiment. However, the supply fan status binary input and supply fan start/stop binary output values were not monitored in the present work. Therefore, to replicate this type of alarm, the supply air flow sensor was used. The sensor value was compared with a schedule that defined optimal system operations. This updated evaluation criteria is described in Algorithm 5.

Algorithm 5 AHU Supply Fan Threshold FDD Algorithm

```
Require: Supply Air Flow
Ensure: alert users of supply fan faults
  if hour > 7 and hour < 19.5 then
    if supply flow > 0 then
      No Alarm
    else
      Alarm
    end if
  else
    if supply flow > 0 then
      Alarm
    else
      No Alarm
    end if

  end if
```

Algorithm 5 did not require training, calibration, or cross validation to determine the expressions. It was applied directly to the testing data and the results were recorded. The testing data included the four fault types described in Section 5.2 as well as normal behavior. The results were broken down by fault type and then the overall results were plotted in ROC space.

Detection: Type A Fault

The first type of fault tested by the threshold method was where the CHW temperature was too high, and as a result the supply fan had to provide larger flow rate than normal. The testing process presented a total of 7,650 inputs and 7,325 of them were normal and the remaining 325 were type A faults. In this case, the threshold method

did not perform well because the fault condition did not affect the schedule of the fan. Therefore, this method could not detect a single fault. As a result the number of true positives and true negatives were both zeros. The probability of detection and the precision were equal to zero, and the the number of faults that were classified as normal was 325. The probability of detection and false alarm are both zero and plotted in Figure 7.2.

Detection: Type B Fault

The second fault was due to a malfunction in the mixed air damper section. This fault had an impact on the mixed air temperature sensor value. A total of 7,475 inputs were presented to threshold algorithm and 150 of them contained faults and the remaining 7,325 were normal. The positive and negative classification results end up with a TPR, FPR, and precision equal to zero as shown in Figures 7.2. Again, the threshold method is not able to detect a single fault because the AHU fan schedule was not compromised.

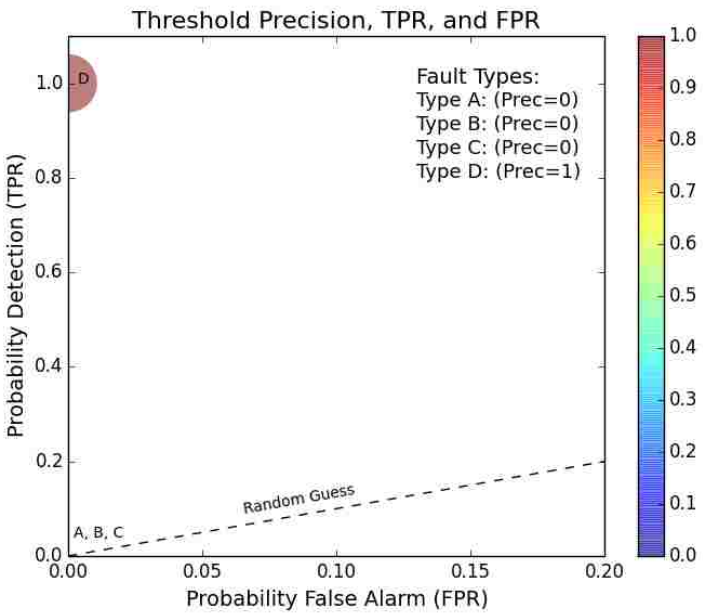

Figure 7.2: Threshold precision (circle size), probability of detection (y-axis), and probability of false alarm (x-axis) results.

Detection: Type C Fault

The type C fault is the situation where the VFD was stuck at a fixed position. The threshold algorithm analyzed 8,525 total inputs that contained a total of 1,200 fault conditions. This method could not recognize any of the faults. Therefore the probability for detection, false alarm, and precision were zero as shown in Figures 7.2.

Detection: Type D Fault

The final fault introduced to the threshold algorithm was the off-schedule fan operations. The threshold method was ideal for identifying this type of fault and detected 100% of the 216 faults. The probability for detection was calculated to be 100%, as shown in Figure 7.2, because the number of false negative classifications was zero. Additionally, the method did not produce any false alarms, and therefore produced a precision of 100%.

Detection: Overall

The results from the individual fault detection tests showed that the threshold method could not detect the type A, B, and C faults. It was able to detect the type D fault well because the fault occurred during off-schedule times which is exactly what the threshold algorithm was set up to recognize. The overall, that combined type A, B, C, and D fault results, are described within a confusion matrix in Table 7.1. It is evident that the method had a low missed detection rate

Table 7.1: Exp 2: Threshold cumulative conf. matrix for type A, B, C, and D faults

		Estimate	
		$\hat{y}$=1	$\hat{y}$=0
Truth	y=1	TP=216	FN=1,675
	y=0	FP=0	TN= 7,242

(TPR = 11%) and a very good false alarm rate (FPR = 0%). The overall accuracy, which is the total correct over the total, was calculated to be a very good 81%. Finally, the precision was 100%, because the method did not produce any false alarms.

7.2 Quantitative Method Results

The quantitative method, which for the experiment included the rule-based model, provided a quantitative assessment of the actual data versus a prediction. The prediction, which represented the "ideal" case, produced values for the mixed air temperature, supply air temperature, and supply air flow for each time-step. A range, set by the standard deviation, surrounded the prediction and was compared with actual values to determine compliance as defined by Figure 6.10. If the actual value fell outside of the range it was classified as a fault.

7.2.1 Rule-Based Results

Section 4.2.2 defines the rule-based expressions that were initially developed based on expert knowledge of the system. The expressions were then calibrated (Section 6.3) and adjusted to match with actual sub-system operations so that they could be applied to the present experiment. In this experiment test data were presented to the expressions that contained four types of faults. The evaluation assessed the methods to detect each fault as well as an overall review of its performance.

Detection: Type A Fault

The rule-based expression considered a total of 7,650 inputs and was able to positively identify 115 faults. However, it falsely identified 389 faults that should have been normal. The probability of detection and false alarm were equal to 35% and 5% respectively. The precision was calculated to be 22% as shown in Figure 7.3.

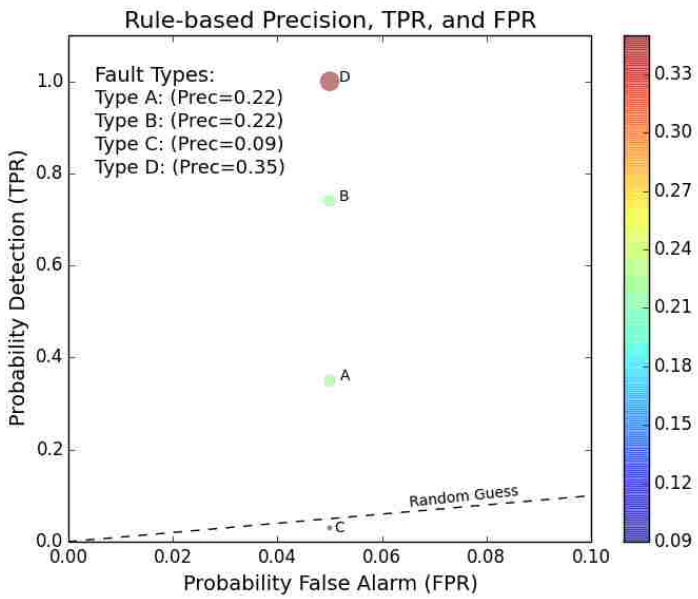

Figure 7.3: Rule-based precision (circle size), probability of detection (y-axis), and probability of false alarm (x-axis) results.

Detection: Type B Fault

The rule based expression was able to identify 112 faults out of the 150. In addition it incorrectly classified 389 false positives. This produced a probability of detection equal to 74% as shown in Figure 7.3. Additionally, the precision value for was calculated to be 22%.

Detection: Type C Fault

The rule-base expression did not perform well at classifying type C faults. It was only able to detect 41 out of the 1,200 total faults. This produced a probability of detection equal to a very low 3% as shown in Figure 7.3. In this case, a random

guess would have been more productive. Also, the precision for this fault type had a very low value of 9%.

Detection: Type D Fault

The rule-based expression was able to detect 100% of the 216 type D faults. This produced a probability of detection equal to 100% as shown in Figure 7.3. However, the expression incorrectly classified 389 normal data points and produced a low precision value of 35%.

Detection: Overall

This fault detection tool was able to predict actual operations fairly well as shown in the actual versus predicted plot in Figure 7.4. Figure 7.4 provides four graphs that describe actual and rule-based results for a two day period. It is evident that the actual air flow results remained within the predicted range except at the end of each day. The supply and mixed temperatures had very similar patterns but tended to not match with the prediction at hour 7 when the system first turned on. These discrepancies indicate that the model cannot represent the ramp rate for the

Figure 7.4: Rule-based experiment 2 time-series example results

temperatures at equipment start up. This trend occurred each day throughout the entire testing data set. To fix this issue an extra expression would need to be added that would affect the rate of increase or decrease. This could be difficult because the rate changes from day to day.

The overall results for the entire experiment are described in Table 7.2. The TP, FN, FP, and TN where 484, 1,407, 389, and 6,936 respectively. Based on these values the overall accuracy, which is the amount correct over the total data points considered, was 80.5% because it was able to classify normal behavior well. Yet, its ability to detect faults was weak and had a precision of 55.4%. This means that if a fault occurs the probability that it will be correctly detected is 25.6%. On the other hand, there is a small (5%) chance that the method will produce a false alarm.

Table 7.2: Exp 2: Rule-Based cumulative conf. matrix for type A, B, C, and D faults

		Estimate	
		$\hat{y}$=1	$\hat{y}$=0
Truth	y=1	TP=484	FN=1,407
	y=0	FP=389	TN=6,936

7.3 Process-History

This experiment evaluated the performance of four process history methods, which were regression, BP, single Fuzzy ART, and LAPART. The regression, BP, and LAPART involved the development of a model that predicted the sub-system sensor values. These predictions were compared with actual values and faults were recognized through the review of the residuals. The ART ANN was developed to classify patterns as either normal or as a fault.

7.3.1 Regression

The regression process entailed training and testing activities. First, training was accomplished by fitting non-linear equations to a data set. Three equations were developed that had the independent variables supply air flow rate, supply air temperature, and mixed air temperature. These independent variables were picked so that during the testing phase predictions could be made and compared with the actual sensor values. After the coefficients for the three equations were found, the test data was presented to each equation to calculate a predicted value for the supply air flow, supply air temperature, and mixed air temperature.

Training

The development of the regression expressions was performed for each of the three sensor points. This meant that the independent variable was either supply air flow, supply air temperature, or mixed air temperature. For each of the expressions the dependent variables, which affect the AHU system performance greatly, were outside air temperature (OSA), and hour of the day. Figures 7.5, 7.6, and 7.7 describe the

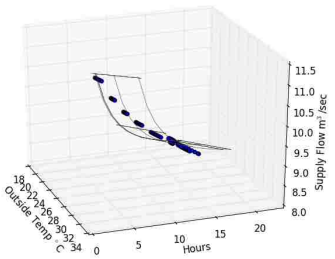

Figure 7.5: Supply air flow regression

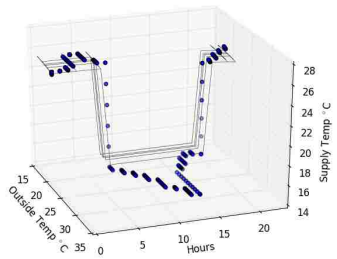

Figure 7.6: Supply air temp regression

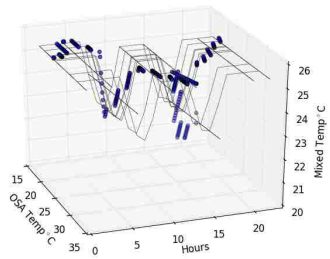

Figure 7.7: Mixed air temp regression

basic surfaces that represent the training data for the three independent variables.

The supply air flow was represented by a quadratic equation described in Equa-

tion 7.1:

$$\text{Supply Flow} = a(\text{OSA}) + b(\text{Hour}^{-3}) + c \tag{7.1}$$

The coefficients for the supply air flow were discovered in the regression optimization to be a = 8.88e-02, b = 7.48e+02, and c = 6.30. The supply air flow regression only considered data points between the hours of 7:00 and 19:30. During the night and early morning the flow should be zero and therefore if flow occurred it would immediately be flagged as a fault.

The supply air temperature was described by Equation 7.2:

$$\text{Supply Temp} = a(\text{OSA}^5) + b(\text{Hour}^2) + c \tag{7.2}$$

The exponents for the outside air and hour variables were determined through trial and error. The coefficients for Equation 7.2 were a = -9.95e-08, b= -1.60e-03, and c = 2.57e+01 were determined using the regression optimization. In this case the regression found the best fit curve for the data when the system was off which was before hour 7:00 and after hour 19:30. The temperature was set to 15.3°C from hour 7:00 to 19:30.

The final sensor value, mixed air temperature, is represented by Equation 7.3:

$$\text{Mixed Temp} = \text{OSA}^{(1/a)} + 2\sin(0.8\text{Hour} + b) + c \tag{7.3}$$

This equation was developed based on user intuition and trial and error. The coefficients a = -1.48e+11, b = 3.96 , and c = 2.39e+01 were determined through the regression optimization process. The surface was constrained during the night when the system was off and also at mid-day when the system was often at a constant temperature that matched with the building return.

Previously unseen data was presented to the regression equations in the testing process. The equations produced outputs that were then compared with actual data

to determine if a fault had occurred or not. This process was conducted for the four different fault types, and the probability of detection and false alarm was calculated.

Detection: Type A Fault

The regression method had an accuracy of about 80% for type A faults. This high accuracy was due to the number of correctly classified data points versus the total number considered. For example, it was able to classify 241 faults and 5,870 normal data points correctly. The probability for detection was calculated to be 74%, but the precision value was only 14% as shown in Figure 7.8. This revealed that the model did not perform as well as the accuracy initially indicated. This was because the number of correctly classified faults was much less than the number of false alarms. The probability for false alarm was a very high 19%.

Detection: Type B Fault

The regression equations had an accuracy of 80% when they attempted to identify type B faults. It was able to correctly identify 110 out of the 150 possible faults. This ratio produced a probability of detection of 73% as shown in Figure 7.8. However, the

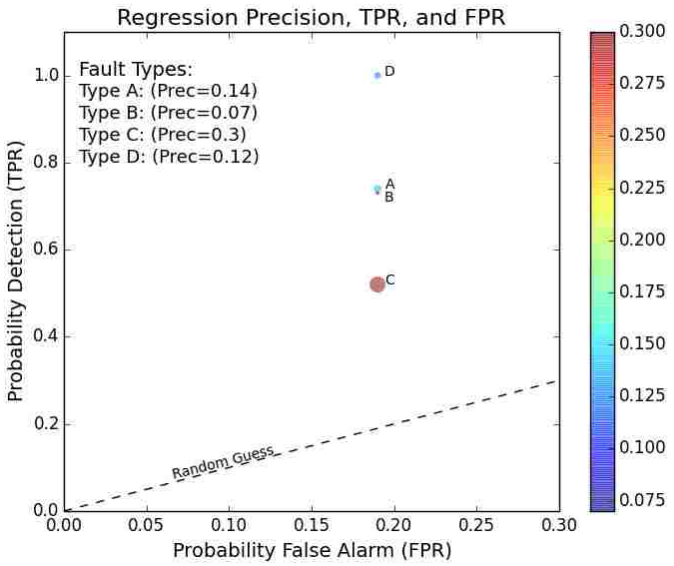

Figure 7.8: Regression precision (circle size), probability of detection (y-axis), and probability of false alarm (x-axis) results.

precision was only 7% because the number of false alarms was very high in comparison to the number of correctly detected faults.

Detection: Type C Fault

The results from the detection of type C faults indicated a low probability of detection (52%) as shown in Figure 7.8. This meant that the equations were able to detect 627 faults out of the total 1,200. The accuracy, which is the amount of true positive and true normals over the total, was calculated to be 70%, however the precision was much less at 30%. This indicated that the number of correctly identified faults was small in comparison to the total number of false positives.

Detection: Type D Fault

The final fault type, that produced off-schedule fan operations, was easily detected by the regression equations, and had a probability of detection equal to 100% as shown in Figure 7.8. This meant that all of the 216 faults were detected and no false classifications were made. However, the equations flagged 1,455 false alarms. This produced an overall precision that was a very low 12%.

Detection: Overall

The regression equations could predict the air flow rate and supply temperature sensor values well. However, the mixed air prediction equation was not as successful at estimating the mixed air temperature values and the discrepancies ended up creating a lot of false alarms. These results are shown in an example plot provided in Figure 7.9, where the actual and modeled results are plotted for a two day period. The regression model results are plotted with a range that is plus and minus

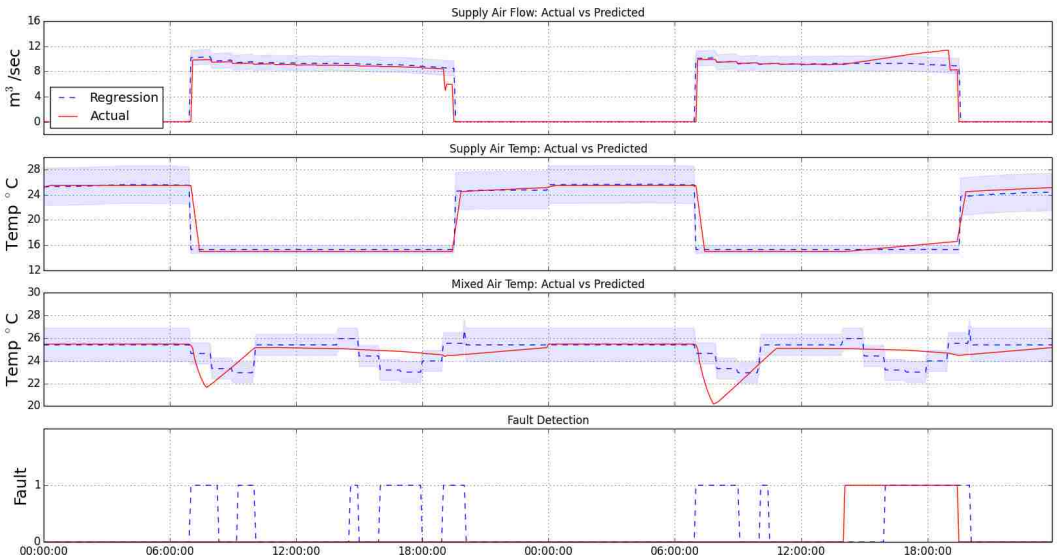

Figure 7.9: Experiment 2: Regression prediction versus actual for fault detection example plot for two days. The plot includes actual data that is normal and type A fault behavior.

the standard deviation. The "on" and "off" schedule times have different standard deviations, and therefore their respective ranges are different sizes. It is evident that the supply flow and temperatures match well but the mixed air model does not fit as well. The mixed air temperature pattern was very difficult to represent using an optimized regression equation.

Table 7.3: Exp 2: Regression cumulative conf. matrix for type A, B, C, and D faults

		Estimate	
		$\hat{y}$=1	$\hat{y}$=0
Truth	y=1	TP=1,194	FN=697
	y=0	FP=1,455	TN=5,870

The overall results for the regression analysis produced the following results: TP = 1,194, FN = 697, FP = 1,455, and TN = 5,870. This results are provided in the confusion matrix given in Table 7.4. The overall accuracy was calculated to be 76%, but the precision was 45%. Additionally, the TPR or probability of detection was equal to 63% and the probability for false alarm was 19%.

7.3.2 Support Vector Machine

The typical SVM, presented in Section 4.2.3, provides a two-class classification evaluation. However, the training data for this experiment only contain one class that does not have any faults. Therefore, a one-class SVM provided in the Python Scikit-Learn Package [95] can be used for novelty detection. Unlike the multiclass approach, this algorithm learns in an unsupervised manner. It can then perform novelty detection by classifying data that is similar to the training data set as normal. If it is not similar than the data point is classified as a fault. The results could then be compared with actual to define the overall accuracy and precision.

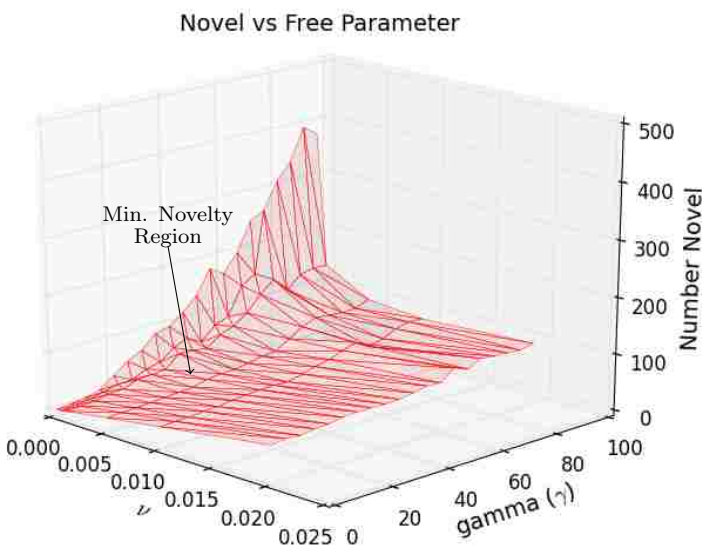

Figure 7.10: Exp. 2 One-class SVM cross validation of free parameters ν and γ

The first step in the implementation of the one-class SVM was to perform a cross validation that defined the best free parameters for the particular data set.

Cross Validation

The one-class SVM incorporates a radial basis non-linear kernel that is shown in Equation 7.4. This approach uses two free parameters ν and γ to define the decision

function which is given by Equation 4.15. The ν parameter defined the upper bound on the fraction of training errors and a lower bound of the fraction of support vectors and is between 0 and 1. The γ parameter represents the kernel coefficient that is shown in Equation 7.4.

$$K(\mathbf{x}, \mathbf{x}') = \exp(\gamma||\mathbf{x} - \mathbf{x}'||^2) \tag{7.4}$$

The K-Fold cross validation described in Section 5.1.1 was used to define the best ν and γ parameters for the training dataset. In this case the data is broken out into 6 folds and the number of novelties for each fold were recorded and averaged for the corresponding free parameter. The final results for the different free parameter iterations are shown in Figure 7.10. For this dataset, the best free parameter scenarios were at ν equal to 0.001 and γ equal to 10.

Training/Testing

The first step in the training process was to set the free parameters defined in the cross validation process. The free parameters were ν equal to 0.001, which set the size of the margin and the number of support vectors, and then $\gamma =$ 10. During training the algorithm builds a model that assigns new examples into one category or the other. The categories are divided by two planes that are at a dis-

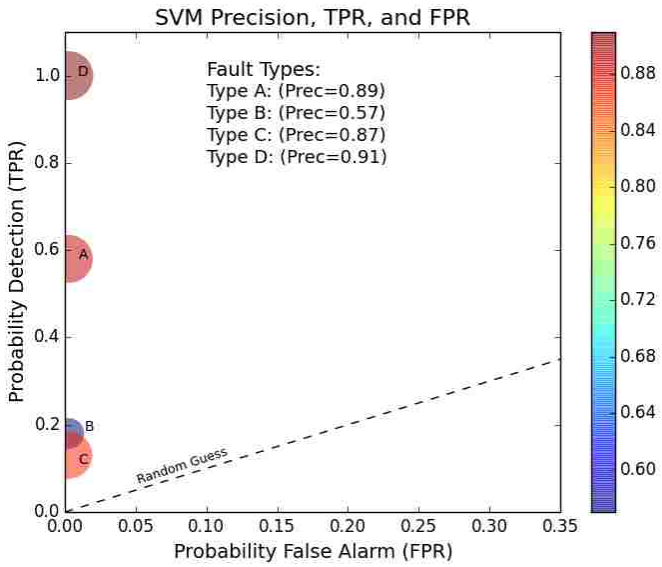

Figure 7.11: OC SVM precision (circle size), probability of detection (y-axis), and probability of false alarm (x-axis) results.

tance from each other that is as wide as possible. After training is complete, new examples are than mapped into the space and a prediction of the particular category is produced based on which side of the planes the new point is located. Throughout the testing of the four faults the TP, FN, FP, TN and other values were calculated and recorded.

Detection: Type A Fault

The first new examples that were mapped into the space that was defined during training where type A faults intermixed with normal data. The result was a detection accuracy of 98%. However, this accuracy is somewhat deceiving because the algorithm was able to identify 189 out of the 325 total faults. This ratio produced a probability of detection equal to 58% that is shown in Figure 7.11. The SVM did not produce many false alarms in comparison to the correctly identified faults and had a precision of 89%.

Detection: Type B Fault

The second set of data presented to the SVM algorithm contained the type B faults. This fault was located in the mixed air sub-system. The SVM algorithm was able to again provide a high accuracy of 98%. However, the accuracy value was very misleading, because the probability of detection was very low (18%) as shown in Figure 7.11. Also, the the SVM tool produced a precision value of 57%. This moderate precision value was due to a similar quantity of correct and false alarm classifications which were 28 and 21 respectively.

Detection: Type C Fault

The type C, which was the VFD malfunction that caused the fan to go to a constant value, was presented to the SVM algorithm for testing. The result was an overall accuracy of 87%. Yet, the probability for correct classification of a fault was a very low 13%. This was because it was only able to correctly classify 154 out fo the total 1,200 faults as described in Figure 7.11. Similar to the algorithm's performance on the type B faults a random guess would have provided better results. It did however avoid false alarms in relation to the number of correctly identified faults. Because of this it produced a precision value of 87%.

Detection: Type D Fault

The final fault presented to the SVM algorithm was the off-schedule fan operations. In this case the SVM achieved a very high accuracy of 100%. It was able to correctly classify all off the 216 total faults. In addition, it had probability of detection equal to 100% as shown in Figure 7.11. The precision value was also high at 91%, which indicated that the ratio of the amount of true fault classifications was much larger than the number of false alarms. The number of true fault classifications was 216 while the number of false alarms was 21.

Table 7.4: Exp 2: OC SVM cumulative conf. matrix for type A, B, C, and D faults

		Estimate	
		$\hat{y}$=1	$\hat{y}$=0
Truth	y=1	TP= 587	FN = 1,304
	y=0	FP= 21	TN = 7,304

Detection: Overall

The SVM was able to accurately (85%) and precisely (95%) detect sub-system faults in the AHU. The overall TP, FN, FP, and TN were found to be 371, 1,304, 21, and 7,304 respectively as shown in the confusion matrix in Table 7.4. This equated to a probability of detection and false alarm equal to 31% and 0.28% respectively. Although the probability of detection was low the SVM algorithm had a precision value that was very high. The precision value provides a good assessment of the methods ability to detect faults. This is because the number of faults is low in comparison to the normal data points, and therefore the precision measures the fraction of detections that are actually faults.

7.3.3 Back-propagation

The back-propagation algorithm, described in Section 4.2.4, is a popular implementation of a multi-layer perceptron. The input layer, in this case, accepted hour of the day, outside air temperature, and occupancy. The input nodes were connected to two hidden layers each with eight nodes. Finally, the output layer had three nodes that correspond to the supply air flow, supply air temperature, and mixed air temperature outputs. The algorithm has two free parameters which were the learning rate and the momentum term. The identification of the best value for the given

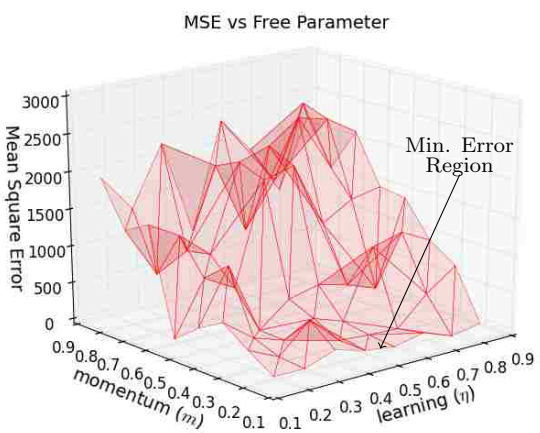

Figure 7.12: Exp. 2 Back-propagation cross validation of free parameters learning rate (η) and Momentum (m)

dataset was accomplished through a K-Folds cross validation process.

Cross Validation

The intent of the cross validation process for the back-propagation algorithm was to identify the learning rate and momentum term that produced the smallest mean square error (MSE) during testing. This process involved a K-folds cross validation procedure that considered 64 different free parameter combinations. The different combinations were tested in four folds and each fold was trained using 600 epochs. The result was a distribution of free parameters and MSE that is shown in Figure 7.12. The surface shown in Figure 7.12 has many peaks and valleys because the each iteration began with different initial weights. The lowest MSE, indicated by the arrow in the figure, was at a learning rate (η) equal to 0.7 and a momentum term (m) of 0.1. For η close to 0 and m close to 1 produces increasing speed of convergence. The opposite, where η is closer to 1 and m is near 0 provides higher learning stability [43]. The learning rate and momentum term were then applied to the training algorithm.

Training

The training algorithm used the learning rate and momentum term found during the cross validation process to produce the weights for the input, hidden, and output layers. The training process described in Section 4.2.4 required 600 epochs (or iterations) of the forward and backward pass process. This process provided the best weights for the various layers of the network. The weights were then sent to a MySQL database for storage. The testing phase would then access the weights and provide a prediction based on the new inputs.

Detection: Type A Fault

The CHW supply temperature fault and normal data were evaluated by the BP method. It was able to detect these faults with an accuracy of 87%. The probability of detection was calculated to be 76% based on the fact that the BP method was able to detect 249 out of the 325 total faults as shown in Figure 7.13. However, there were 916 false alarms detected which created a a probability of false alarm to be 13%, and a low precision value of 21%.

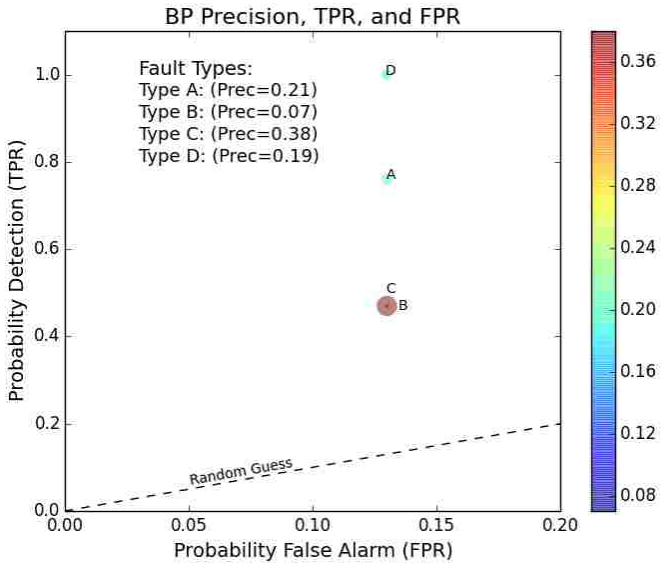

Figure 7.13: BP precision (circle size), probability of detection (y-axis), and probability of false alarm (x-axis) results.

Detection: Type B Fault

The second fault, which was the malfunction of the mixed air damper sub-system, was also evaluated by the BP method. Similar to the type A fault it had an accuracy of 87%. It was able to detect 70 out of the 150 total faults, which produced a 47% probability of detection as shown in Figure 7.13. Additionally, the approach produced very low precision value that was 7%.

Detection: Type C Fault

A total of 8,525 data points containing normal and the type C faults were introduced to the BP method. The outcome provided an accuracy of 81%. However, it was only able to correctly detect 570 out of the 1,200 total faults. Therefore, the probability of detection was 47%, which is shown in Figure 7.13. Additionally, the calculated precision was 38%.

Detection: Type D Fault

The off-schedule supply fan fault was also evaluated, and the accuracy was 88%. It was able to classify 100% of all of the faults, which meant that 0% of the faults were classified as normal. This produced a probability of detection to be 100% as shown in Figure 7.13. However, there were still many false alarms in comparison to the number of correct fault classifications. Therefore, the precision calculation was a very low 19%.

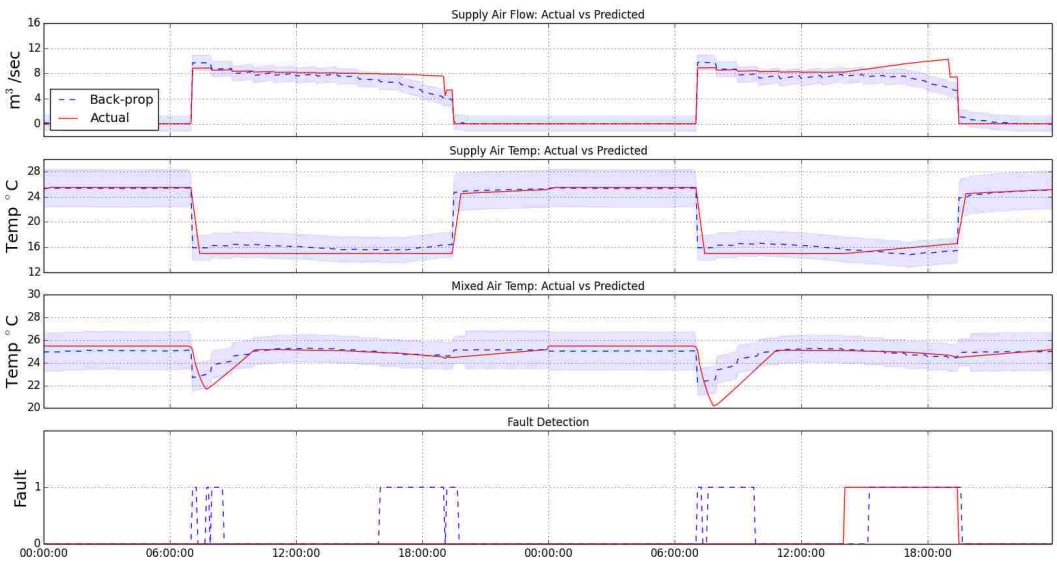

Figure 7.14: Experiment 2: Example output of the back-prop prediction for two days

Detection: Overall

The BP was trained using two months worth of data and then tested on the one month of data that contained normal sensor values and four types of faults. The approach was able to provide an accurate prediction that represented the actual sensor data, and a two day sample is shown in Figure 7.14. However, the prediction could not provide a reliable means for fault detection because the actual often strayed slightly away from the prediction and a false alarm was thus initiated often.

The overall results for the BP method are TP = 1,105, FN = 780, FP = 916, and TN = 6409, and are shown as a confusion matrix in Table 7.5. The accuracy of this method was 81% when applied to all of the faults. Also, the probability for correct detection and false alarm were 58% and 13% respectively. Lastly, the precision of this method for the entire data set was 54%.

Table 7.5: Exp 2: BP cumulative conf. matrix for type A, B, C, and D faults

		Estimate	
		$\hat{y}$=1	$\hat{y}$=0
Truth	y=1	TP= 1,105	FN = 780
	y=0	FP= 916	TN = 6,409

7.3.4 ART

The algorithm for the single Fuzzy ART was described in Section 4.1.1. It was developed in the Python script language. The algorithm has one free parameter that required the cross validation process to define the optimal value. Similar to the previous tests, such as the SVM and BP, the best free parameter was found using the K-Folds cross validation process.

Cross Validation

The K-Folds cross validation process was implemented to discover the best free parameter (ρ). The ρ value is known as the vigilance parameter and defines the complexity of the system. The cross validation was set up in a simple loop within the Python script and ran four different folds or iterations on the training data. For each iteration the ART algorithm trained and tested on data that contained no faults. After each iteration the number of novelties and created categories were recorded for each ρ value. The results from this test are plotted in Figure 7.15. The ρ values less than 0.7 were considered too general and would not provide sufficient complexity to identify faults. The ρ values larger than 0.9 resulted in over 20 categories which hinted that the trained algorithm could be too complex and thus cause false alarms to be prevalent. Therefore a middle ground was found and a ρ value of 0.85 was chosen.

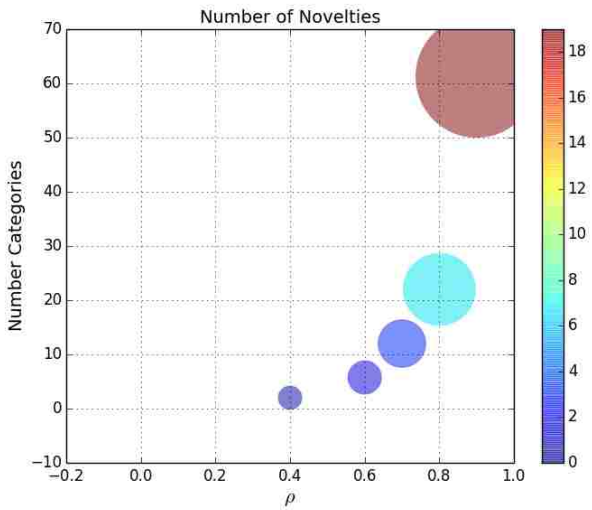

Figure 7.15: Exp. 2: Number of novelties (size of circle) for a give ρ (x axis) and categories (y axis) produced during the cross validation.

Training

The training and testing process used a ρ value of 0.85 found during the cross validation. Then, two months of input data from the TRNSYS simulation were provided to the algorithm and the ART gained knowledge about the sub-systems of the AHU.

The knowledge was stored in the form of templates inside of a MySQL database table. A total of 34 templates were created to represent the training data set. Each of the templates was below the maximum size of 0.9 as defined in Section 4.1.1.

Detection: Type A Fault

A total of 7,649 data points that contained normal and 325 type A faults were presented to the ART algorithm. The algorithm was able to classify 296 faults correctly and 160 incorrectly. The probability of detection was found to be 91% (Figure 7.16) while the false alarm rate was calculated to be 2%. In addition, the overall accuracy of the algorithm for this testing data set was 97%, and the precision was 65%.

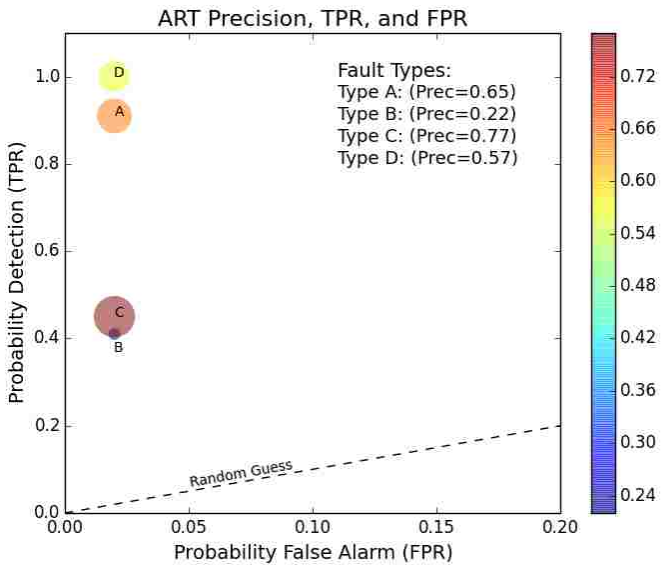

Figure 7.16: ART precision (circle size), probability of detection (y-axis), and probability of false alarm (x-axis) results.

Detection: Type B Fault

The type B fault was difficult to detect for the ART algorithm. Although the overall accuracy was 95% it was only able to correctly classify 62 out of the 150 total faults. The probability for detection was calculated to be 41% as shown in Figure 7.16.

The ratio of correct classified versus false alarms was low and therefore produced a precision value of 22%.

Detection: Type C Fault

The third test considered type C faults. A total of 8,524 data points of which 1,200 of them were faults. The ART algorithm had a overall accuracy of 90% and was able to correctly classify 541 of the 1,200, which equated to a medium probability of detection equal to 45% as shown in Figure 7.16. The precision was 77% which indicated that it could produce a high probability of correct fault classification while maintaining a low false alarm rate.

Detection: Type D Fault

The final test for fault detection using the ART algorithm considered 7,540 total inputs and 216 of them were type D faults. The ART algorithm performed well at an accuracy of 97%. It was able to correctly identify 100% of the faults without classify any faults as normal. In addition, the precision was calculated to be 57% and the probability for false alarm was 2% as shown in Figure 7.16.

Detection: Overall

The ART algorithm had an overall accuracy of 90% for all four of the faults. It was able to correctly classify type A faults very well at a probability of detection that was 91% and a precision of 65%. For instance, the type A fault and normal data are plotted in Figure 7.17. The ART algorithm performed FD by checking each data point versus the algorithm's stored knowledge. If the actual data point fell within the stored knowledge, shown as the green shading in Figure 7.17, then the point

was considered normal. If the stored knowledge could not recognize the actual point then it was considered a fault. The stored knowledge was based on the training process performed by the ART algorithm where multi-dimensional hyperboxes were

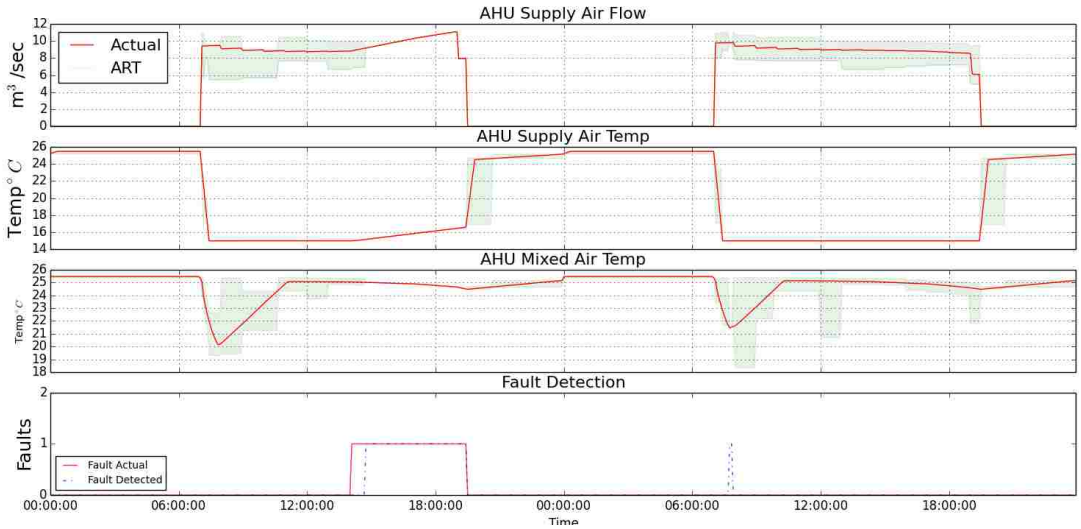

Figure 7.17: Exp. 2: Example output of the Fuzzy ART for two days

created as described in Section 4.1.1. The example shown in Figure 7.17 shows that the algorithm was able to correctly detect a type A fault on day one. During the second day the algorithm detected a fault around hour 8. However, the sensor data represented normal behavior and therefore the algorithm produced a false alarm.

Table 7.6: Exp 2: ART cumulative conf. matrix for type A, B, C, and D faults

		Estimate	
		$\hat{y}$=1	$\hat{y}$=0
Truth	y=1	TP= 1,115	FN = 773
	y=0	FP= 160	TN = 7,164

The TP, FN, FP, and TN values were 1,115, 773, 160, and 7,164 respectively, and are shown as a confusion matrix in Table 7.6. The overall precision, probability for detection, and false alarm were calculated ass 87%, 60%, and 2% respectively.

7.3.5 LAPART

The final method used to evaluate Experiment 2 data was the LAPART algorithm. This algorithm was described in Section 4.1.2 and can provide outputs that predict the AHU sensor values. The prediction values were then compared with actual values and if the actual fell outside of the range determined by the algorithm the particular input was flagged as a fault. Similar to the SVM, BP, and ART algorithms the LAPART was subjected to a cross validation process. The cross validation process evaluated the two free parameters, ρ_A and ρ_B, to find the match that provide the smallest mean square error.

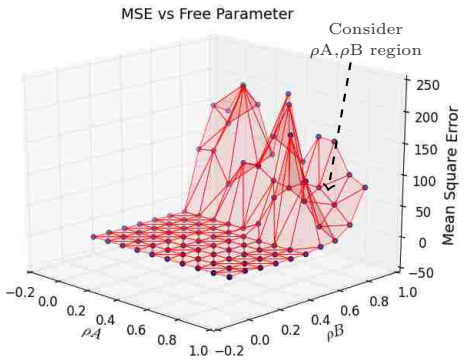

Figure 7.18: MSE was plotted for each free parameter scenario. At low ρ values the MSE was 0. The MSE was high at larger ρ_B and decreased as ρ_A increased.

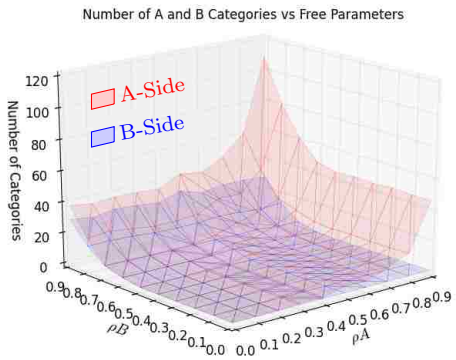

Figure 7.19: The number of categories at the corresponding free parameters. The A categories increase at a larger rate than the B side as the ρ values increase.

Cross Validation

The cross validation process used the K-Fold method with four folds. The cross validation considered the mean square error between the predicted value and the actual. It also took into account the number of categories created for the different free parameter values. The intent was to find a free parameters that had a small error and

medium number of categories. A mean square error of zero was calculated for small ρ_B values as shown Figure 7.18. However, a small ρ_B value would produce results that would be too general. In this case, the probability for detection would likely be zero. Therefore, the free parameters must provide a level of complexity that can detect faults while also providing a generalization that allows for correct detection of data points that have not previously been seen. The best ρ_A and ρ_B were 0.9 and 0.7 respectively as indicated by the arrow shown in Figure 7.18. This arrangement of vigilance values produced an average number of categories that was 61.75 for the A side and 5 for the B side which is shown with the number of category results for the respective free parameter scenarios in Figure 7.19.

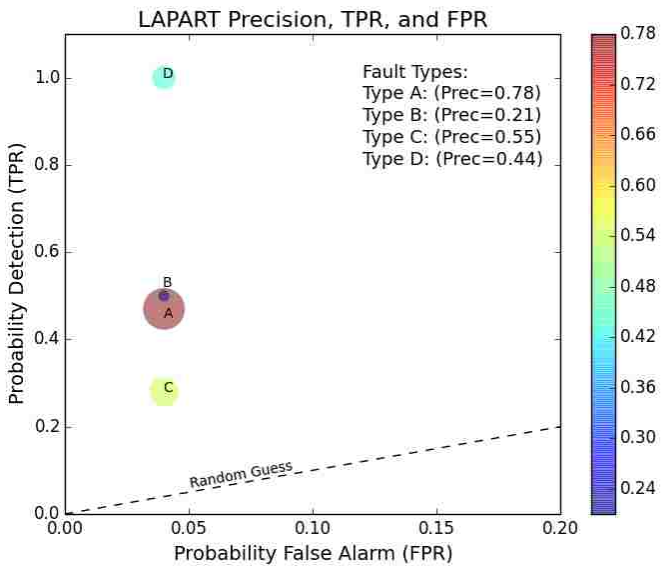

Figure 7.20: LAPART precision (circle size), probability of detection (y-axis), and probability of false alarm (x-axis) results.

Training

The training of the LAPART algorithm used 17,567 inputs that contained no faults. It produced 53 categories on the A-side and 6 on the B-side. Each of the categories on the A-side had euclidean size of 0.3 or less. The categories on the B-side had a maximum size of 0.9. The training process, which presented of all of the inputs twice

to the algorithm, took 467 seconds (7.8 minutes). After the training was complete the algorithm used hour of the day, occupancy, and outside air temperature as an input on the A-side and no inputs on the B-side. Instead the B-side provided a prediction output of the mixed air temperature, supply air temperature, and supply air flow rate. The algorithm considered 9,216 inputs and took 161 seconds (2.7 minutes) to run. The prediction was then compared with actual values to detect if a fault had occurred or not.

Detection: Type A Fault

The LAPART algorithm was able to detect the CHW temperature fault at an accuracy of 95%. It was able to correctly detect 256 out of the 325 total type A faults. This produced a probability of detection equal to 78% as shown in Figure 7.20. However, the method did classify 279 false alarms and therefore had a probability of false alarm equal to 4% and a precision of 47%.

Detection: Type B Fault

The mixed air damper fault (type B) was more difficult to detect than the type A fault. Although the LAPART method had an accuracy of 95% it was only able to detect 75 out of the total 150 faults. This ratio produced a probability of detection equal to 50% as shown in Figure 7.20. Also, the precision was also very low at 21%.

Detection: Type C Fault

The presentation of the normal and type C fault data to the LAPART algorithm produced an accuracy of 87%. However, it was only able to correctly identify 342 out of the 1,200 total faults. Based on these results, the probability for detection was

calculated to be a very low 28%. The precision was moderate at 55%, because the number of false alarms was less than the number of correct detections (Figure 7.20).

Detection: Type D Fault

The LAPART algorithm was able to perform at an accuracy of 96% when it reviewed the normal and type D fault data. Similar to the other fault detection methods it was able to detect 100% of the faults without mis-classifying a fault data point as normal data.

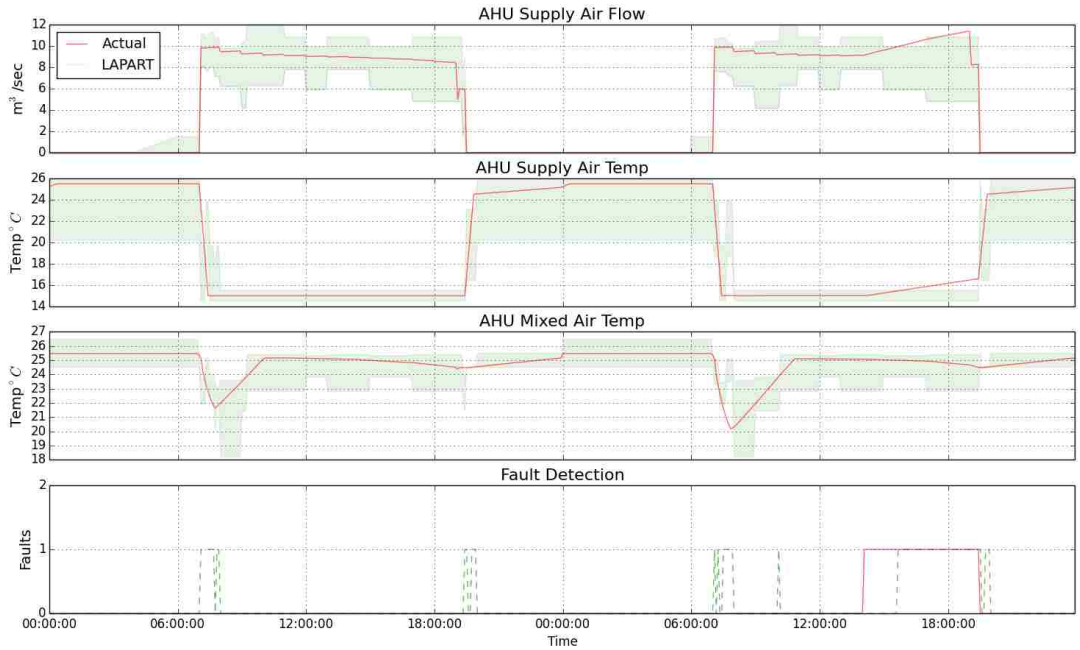

Figure 7.21: Exp. 2: Example output of the LAPART for two days

Detection: Overall

The LAPART algorithm was able to detect faults in the AHU sensor data. The method used all of the input features including hour of the day, occupancy, outside

air temperature, and each of the three AHU sensors to train. Then, only hour of the day, occupancy, and outside temperature were used as inputs and the algorithm produced three predictions that is shown graphically in Figure 7.21. The bottom plot in Figure 7.21 describes the false alarms and correct detections.

Table 7.7: Exp 2: LAPART cumulative conf. matrix for type A, B, C, and D faults

		Estimate	
		$\hat{y}$=1	$\hat{y}$=0
Truth	y=1	TP= 889	FN = 1,002
	y=0	FP= 279	TN = 7,044

The overall results for all of the faults are as follows: TP=889, FN=1,002, FP=279, and TN=7046 as shown in Table 7.7. The overall accuracy and precision were 86% and 76% respectively. The probability for detection and false alarm were 47% and 4% respectively.

7.4 Fault Detection Method Comparison

The criteria used for measuring and comparing the binary classification methods include accuracy, probability for false alarm, probability for detection, and precision (Section 5.7.1). The accuracy, defined by Equation 5.3, describes the quantity of normal and fault cases that were correct in comparison to the total number of data points considered. This measure can be helpful, but in cases such as these, where there are limited occurrences of faults it does not provide a meaningful measure. For example, the rule-based method had a high accuracy of 80%. However, it had a low precision and probability of detection equal to 55% and 26% respectively. Therefore, the present work plots the probability of detection (TPR) and false alarm (FPR) results from each method in ROC space (as described in Section 5.7.1). In addition to ROC space where TPR and FPR are plotted, the present work plots precision versus the probability for detection. This comparison is considered the most critical

because it describes the fraction of classifications that were correct over the total detected.

7.4.1 ROC Space Plots

The TPR versus FPR results for each method's ability to detect type A faults are plotted in Figure 7.22. The ART algorithm was the top performer for the detection of type A faults with a TPR equal to 91% and a FPR of 2%. It plotted very close

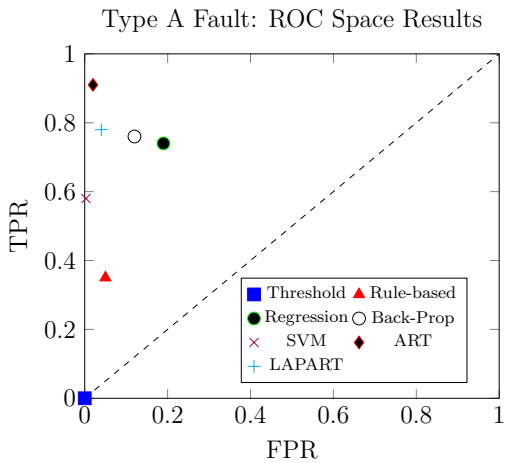

Figure 7.22: Type A fault TPR and FPR results for each detection method

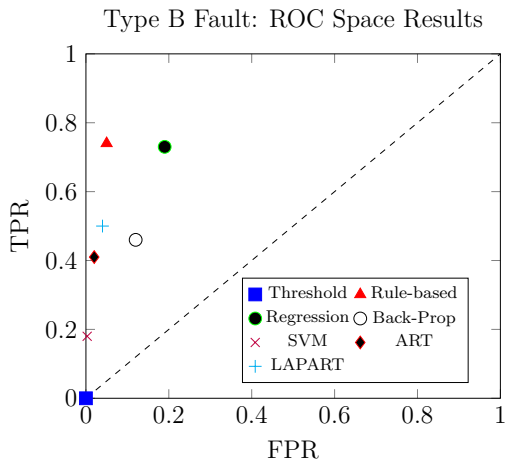

Figure 7.23: Type B fault TPR and FPR results for each detection method

to the optimal point which is (0,1). The next best performer was the LAPART algorithm followed by the BP. The type B TPR and FPR results indicated that the rule-based method performed the best as shown in Figure 7.23. The next best performers were the ART and LAPART who both had smaller FPR, but had smaller TPR in comparison to the rule-based results.

The results from the third fault type (Type C) showed that the ART algorithm performed better than the others (Figure 7.24) with a TPR and FPR equal to 45% and 2% respectively. The next best was the LAPART and then the SVM. The BP and regression methods had high TPR in comparison to the others but had much

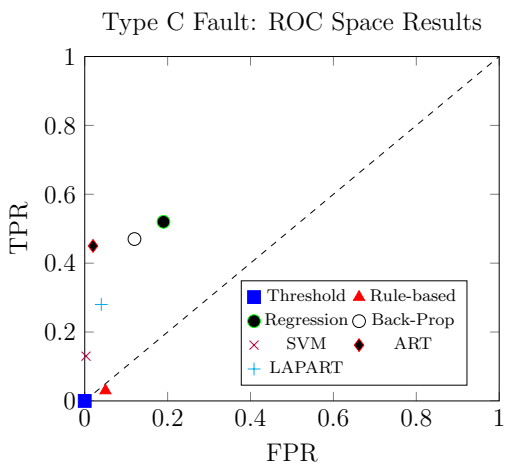

Figure 7.24: Type C fault TPR and FPR results for each detection method

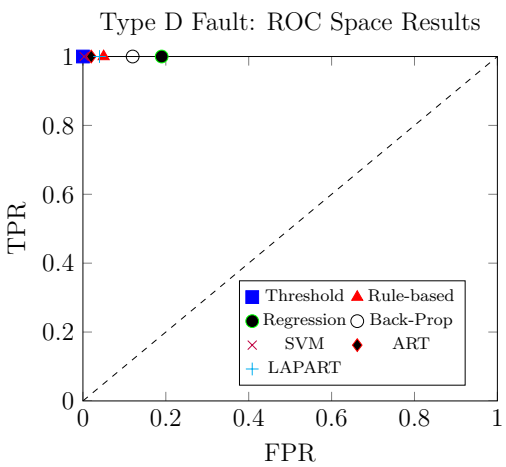

Figure 7.25: Type D fault TPR and FPR results for each detection method

larger FPR. Each of the methods performed well when presented with the fourth fault type (Type D) as shown in Figure 7.25. All seven methods tested had a TPR equal to one and therefore the methods with the smallest FPR, such as SVM and ART, were considered the best for this fault type.

7.4.2 Precision Results

The final review of the fault detection methods was the comparison of the precision. This comparison was performed by plotting the precision value versus the probability of detection (or TPR) for each of the fault types (Figures 7.26 to 7.29). The optimal result would be at point (1,1) on the precision versus TPR graph. Therefore, the results for the type A fault detection indicate that the best methods are ART, and SVM (Figure 7.26). All of the methods did not perform as well on the type B fault. However, the best method was rule-based approach as shown in Figure 7.27.

The results from the type C fault test show that the ART and LAPART algorithms performed the best because they had the highest precision and TPR as shown in Figure 7.29. The final test results for the type D fault indicate that the

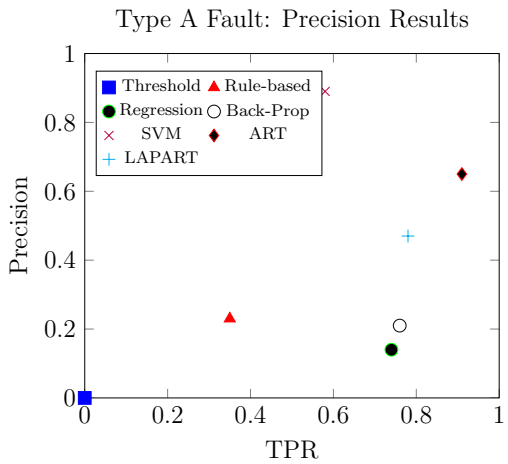

Figure 7.26: Type A fault TPR and Precision results for each detection method

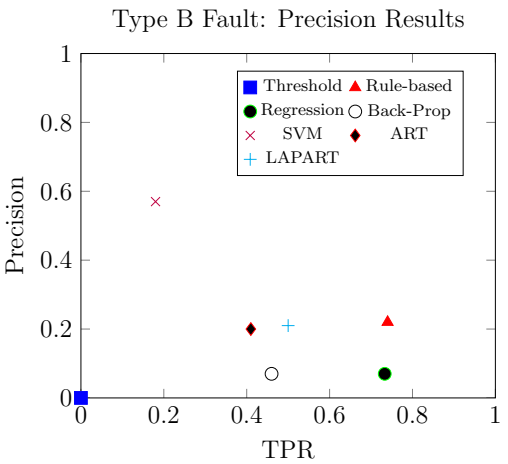

Figure 7.27: Type B fault TPR and Precision results for each detection method

best method was the threshold method because it did not produce any false alarms and therefore had a precision and TPR equal to one. The next best method was the SVM because it had the lowest probability for false alarm.

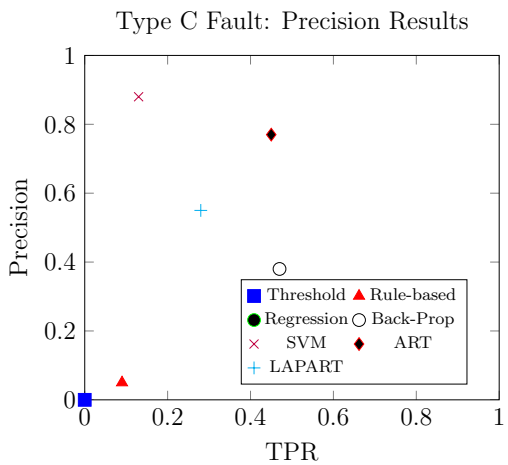

Figure 7.28: Type C fault TPR and Precision results for each detection method

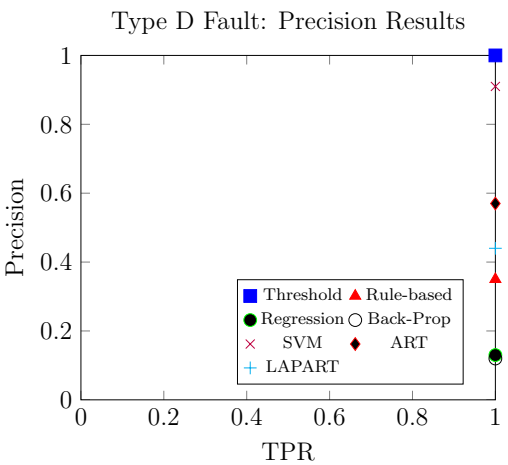

Figure 7.29: Type D fault TPR and Precision results for each detection method

7.4.3 Overall ROC & Precision Results

The overall results showed that the ART algorithm was the best choice for fault detection on an AHU where only training data without faults were available. This conclusion was based on the compilation of data from the four tests that considered the faults: CHW high temperature, mixed air damper malfunction, VFD fan failure,

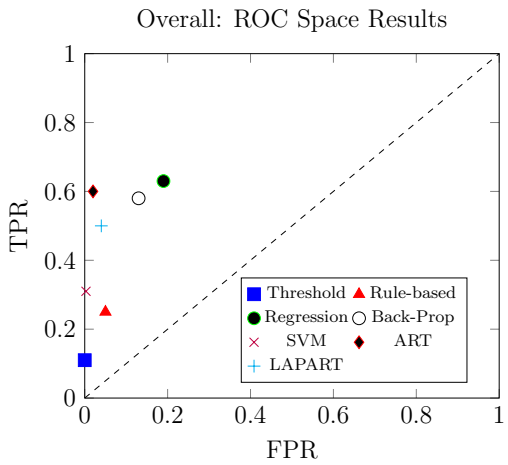

Figure 7.30: Overall TPR and FPR results for each detection method

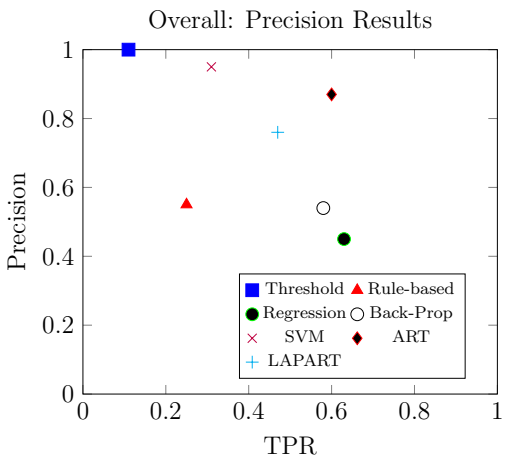

Figure 7.31: Overall TPR and precision results for each detection method

and off-schedule fan operations. The first metric, which was the comparison of TPR versus FPR showed that the ART algorithm plotted the closest to the optimal (Figure 7.30). Additionally, the ART algorithm had the best ratio of precision versus recall as shown in Figure 7.31.

The LAPART algorithm was able to provide a good prediction of the three sensor values, however it fell short at providing the best fault classification results for this test. The LAPART algorithm did not perform as well as ART because of the reset mechanism used to define the correct link between the A and B side templates. The reset could have decreased its ability to provide accurate classifications, and it is hypothesized that the LAPART algorithm would have performed better if it acted as a binary classifier that defined the data as a fault or normal. Instead the experiment

used the algorithm as a predictor and identified faults based on the comparison of the prediction with the actual.

Chapter 8

Experiment 3: Adaptability Results

In the previous chapter, Experiment 2 was used to evaluate the performance of detection methods that considered faults intermixed with normal data. The statistics of the normal data remained the same throughout the experiment, which meant that the average, maximum, minimum, and standard deviations were constant throughout. However, in practice this is unlikely to occur and normal data statistics will change often. For example, the training and testing data in the Chapter 7 were based on occupancy patterns that did not include students because the school year had not started. As opposed to this exper-

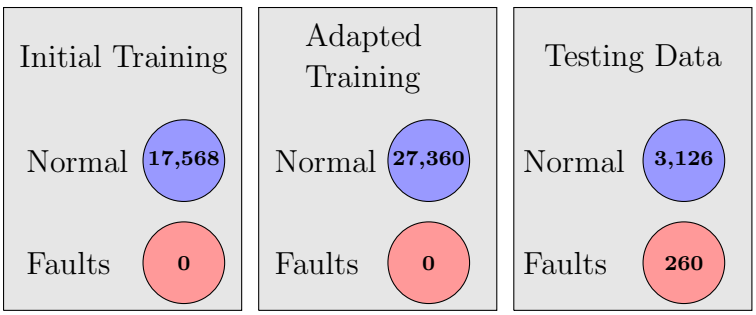

Figure 8.1: The initial training data set contained 17,568 normal and 0 fault, the adapted training set contained 27,360 normal and 0 fault, and the testing set contained 3,126 normal and 260 faults.

iment that applied the top detection tools from Experiment 2 to normal data that experiences significant statistical changes. In this case the fault detection tool must adapt to the changing circumstances and efficiently learn new normal behavior.

This experiment begins with the initial training of 17,568 normal data points and then tested on 3,216 normal data points and 260 faults as shown in Figure 8.1. This initial training and testing process is considered to be the baseline case. The next step was to add to the previous knowledge by training on 9,792 new data. The results from this test were considered adapted and defined how well the method could learn different normal data.

The best fault detectors from Experiment 2 were ART, LAPART, and the one-class SVM. These algorithms were already subjected to a cross validation process for Experiment 2 tests. Therefore, the same free parameters discovered for each method were used for the baseline and adapted tests in the present experiment. The baseline test used the same training data from Experiment 2 and then tested the data on one week of normal and fault data when school was in session. The adapted test applied new normal data that had different statistical properties to the existing training. Then it performed fault detection on the same testing data set.

8.1 Normal Data Statistics

The normal data statistics varied depending on the occupancy levels. This experiment addressed the difference between the summer and fall semesters. During the summer few students were present, while many students attending classes were present in the fall. The operational supply and mixed air temperatures in the AHU were not observed to change. However, the air flow rate went from a median value of 9.4m^3/sec to 10.2m^3/sec. Also, the flow rate standard deviation increased from 0.49 to 1.15. The change in the normal data distribution is shown in Figure 8.2 between

the occupancy levels with and without students. The air flow rate is the only sensor value that changes because it had to increase the amount of thermal power in order to account for the increased cooling load caused by the students. The supply and mixed air temperature are set to a specific set point and therefore would not change due to increased occupancy.

8.2 Baseline Test

The baseline test used data that were produced from the TRNSYS simulation. The simulation took into account the summer reduced occupancy and then the fall semester student occupancy levels. The data from the two occupancy periods had different statistical properties, and the SVM, ART, and LAPART were trained on the summer data and then tested on the fall semester data.

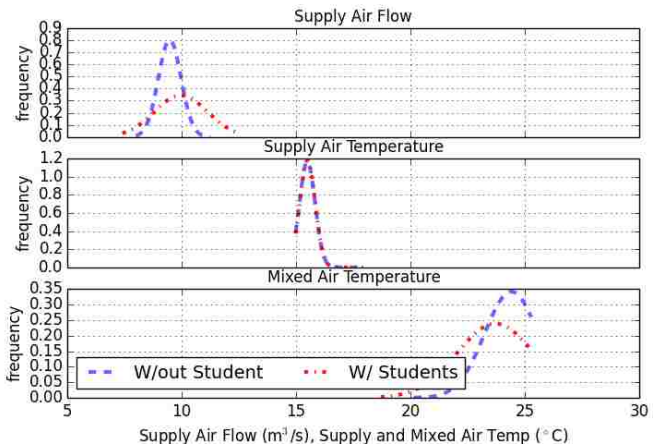

Figure 8.2: Exp. 3: The normal data in the training and testing sets change depending on the occupancy.

8.2.1 One-Class SVM

The one-class SVM correctly detected 293 out of 390 faults in the baseline test. This ratio produced a probability of detection of 75%. This probability was considered

good, but the probability for false alarm was also very high at 30%. Additionally, the precision value, which is the probability of a fault given that a fault has occurred, was a very low 11%.

8.2.2 ART

The ART algorithm considered a total of 4,952 points and flagged 390 of them as faults. It correctly identified 253 out of the 390 and therefore it had a probability of detection that was 65%. However, the algorithm had a total of 1,346 false alarms

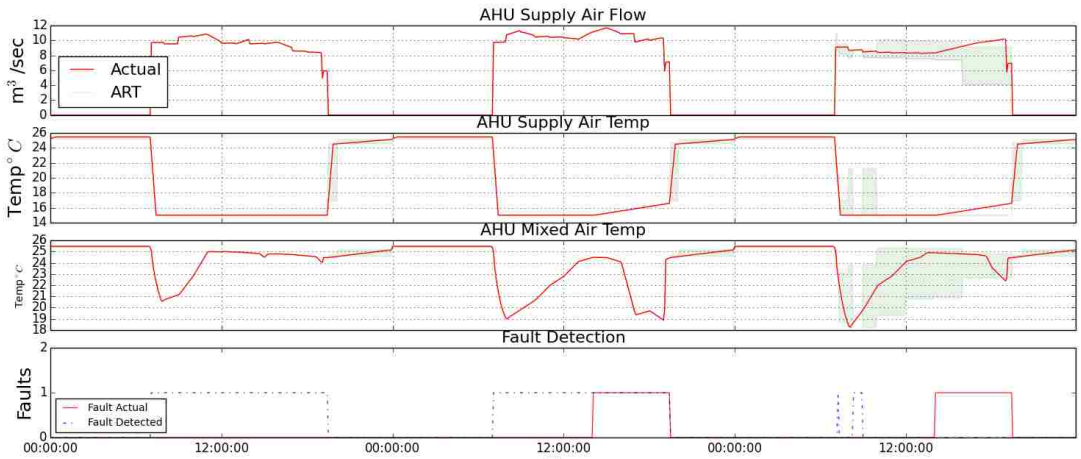

Figure 8.3: Exp. 3: Sample three days of ART algorithm results for the baseline data set

which produced a false alarm rate of 30%. Additionally, its overall precision value was 15%, which is considered a low value. The ART algorithm did not perform well overall for the baseline case.

The fault detection process performed by the ART algorithm considered the three actual AHU sensor values. The actual and the anticipated value produced by ART are plotted in Figure 8.3. The graph at the bottom of the figure shows the resulting actual and detected faults. The graph confirms that there are a high number of false alarms.

8.2.3 LAPART

In the initial application of the LAPART algorithm it was able to detect 348 out of the 390 faults. Based on this the probability for detection was a very high 89%. However, the algorithm flagged a total of 1,427 false alarms which produce a probability of false alarm to be 31%. Additionally, the precision was a relatively poor at 20%.

8.3 Adapted Test

The adapted test used the same data set for testing the fault detection algorithms, but it expanded the training to include more data (27,360) as described by Figure 8.1. The additional data contained no faults, and was more representative of the normal conditions that would be seen in the testing data. Again, the one-class SVM, ART, and LAPART algorithms were trained and tested.

8.3.1 One-Class SVM

The one-class SVM trained on an additionally 9,792 data points. Then it tested on the same data set as the baseline test and the results were recorded. The algorithm improved its probability for producing false alarms by reducing its FPR from 30% to 1.6%. Its probability for detection decreased from 75% to 53%. The probability for detection was high in the baseline case because its precision was very low. After it was retrained on the new data points the probability of detection decreased because the algorithm became more complex. The SVM algorithm still preformed well and its precision increased from a very low 11% to 74% because the number of FP went from 1,352 to 74.

8.3.2 ART

The ART algorithm increased the number of templates, which represented its stored memory, from 33 to 41. This improved its fault detection abilities in all areas from

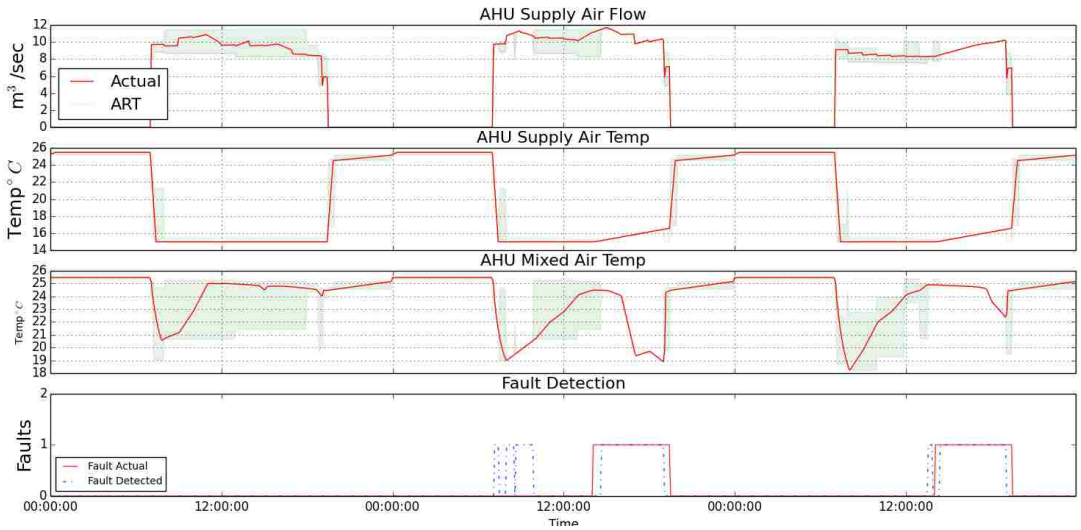

Figure 8.4: Exp. 3: Sample three days of ART algorithm results for the adapted testing data set

the baseline to the adapted test. First, the probability of detection increased from 65% to 85%. Additionally, the probability of false alarm decreased from 30% to 1.6%. This decrease prompted the precision to increase as well from 15% to 81%. This drastic increase in improved fault detection is evident in Figure 8.4 which plots the results for the adapted case over the same time period as the baseline test shown in Figure 8.3. The number of false alarms decreased considerably along with the number of true positives for this three day span.

8.3.3 LAPART

The number of categories increased from 104 to 142 on the A-Side and from 33 to 44 on the B-Side. The adapted LAPART algorithm correctly detected 306 out of the

390 total faults. The probability of detection decreased from 89% to 78% from the baseline to adapted case. The probability of false alarm did improve significantly, decreasing from 31% to 4%. This reduction in the false alarm rate increased the precision from 20% to 62%.

8.4 Overall Results

The ART, LAPART, and SVM algorithm were all able to adapt to the change in normal data. Their probability of false alarm for each of the algorithms decreased to less than 4% as shown in Figure 8.5. The best algorithm according to the TPR versus FPR ROC space was the ART algorithm with a TPR of 84% and an FPR of 1.6%. The ART algorithm also had the best precision value of 81% as shown in the precision versus TPR plot (Figure 8.6).

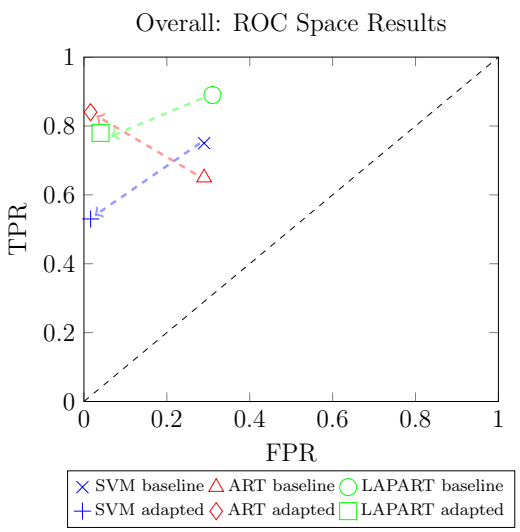

Figure 8.5: Overall TPR and FPR results for each detection method

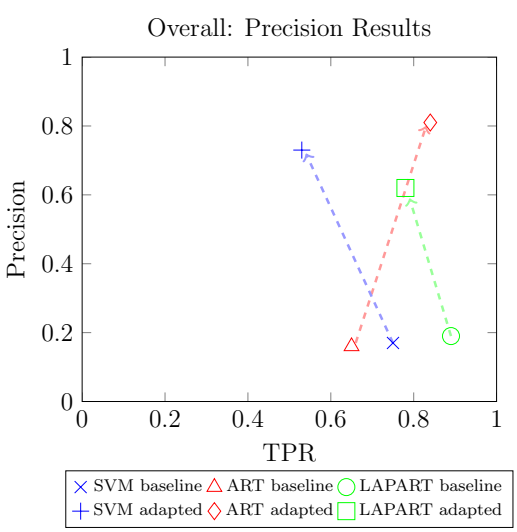

Figure 8.6: Overall TPR and precision results for each detection method

Chapter 9

Exp. 4: Fault Diagnostics Results

This experiment performed the task of distinction or characterization of the particular input features. The experiment implemented the LAPART and multi-class SVM. The two algorithms were trained using data that contained normal and fault behavior. There were 13,672 and 1,496 normal and fault data points respectively as shown in Figure 9.1. Additionally, the particular type of data were labeled as either a zero for normal behavior or 1 through 4 for fault conditions. The testing data set contained 7,372 and 1,844 normal and fault data respectively as shown on the right side of Figure 9.1. The experiment first discovered the best free parameters for the LAPART and SVM algorithm. This was accomplished through a cross validation process. Then the algorithms were trained with the best parameters and tested on previously unseen data. Finally, the results

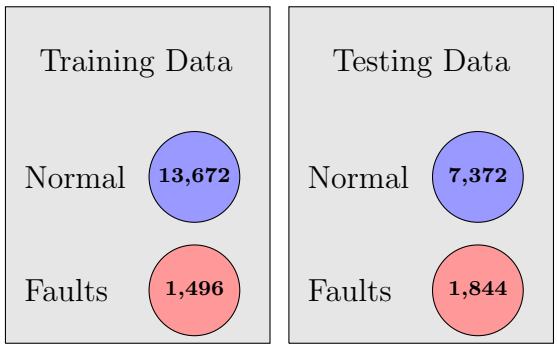

Figure 9.1: The training data set contains 13,672 normal and 1,496 fault while the testing set contains 7,372 normal and 1,844 faults.

from the FDD were analyzed based on the F_1-scores calculation. The F_1 score is a measure of the method's accuracy. It considers both the precision and probability of detection to compute an overall score.

9.1 Cross Validation

Similarly to experiments 2 and 3, the K-Folds cross validation method was used to find the optimal free parameters for the LAPART and SVM algorithms. In this case, the algorithms were subjected to 6 folds, and the average mean squared error was calculated for each iteration. The free parameters corresponding with the smallest error were chosen and used for training and diagnostic testing.

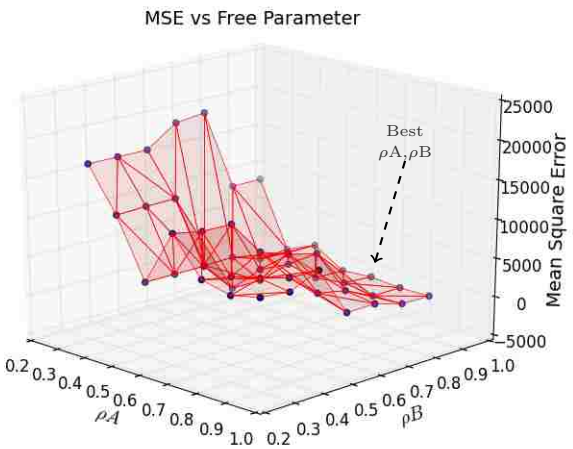

Figure 9.2: MSE as a function of the free parameters. At high ρ A and B values the MSE tended towards zero. The best free parameters were: ρA = 0.8 and ρB = 0.8

9.1.1 LAPART

The free parameters that defined the size of the categories for the LAPART algorithm, were denoted as ρA and ρB. The cross validation process introduced 49 different combinations and the mean square error for each are plotted in Figure 9.2. The lowest mean square error (60.5) was produced by a ρA equal to 0.9 and ρB equal to 0.9. The number of categories created on the A and B side of the LAPART algorithm

are shown in Figure 9.3. The number of categories on the B-Side remained constant at 5 throughout the validation iterations. The categories in the A-Side, however, increased in an exponential manner as ρA increased, and increased slightly in a linear manner as ρB increased. The average number of categories for the optimal free parameters were 92 and 5 for the A and B side respectively.

9.1.2 Multi-Class SVM

The optimal free parameters, C and γ, for the multi-class SVM were found through a 6 fold K-Fold process as well. The different combination of the penalty parameter of the error term, C, and the kernel coefficient, gamma were tested to find the mean square error for diagnostics of the AHU fault conditions. The lowest error was calculated to be 0 and the highest was 645.1. The combination $C = 0.8$ and $\gamma = 10$ had an error of 0 and was used for the training and diagnostic tests.

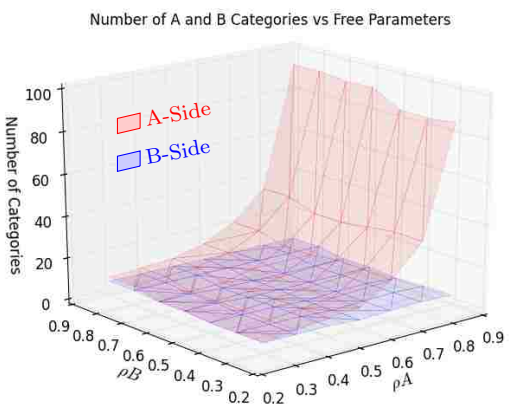

Figure 9.3: Number of categories as a function of the free parameters. The B-Side remained constant at 5, while the A-Side increased in an exponential manner as ρA increased.

9.2 Training

The training process for each of the algorithms introduced two months of data that represented faults and normal behavior. In addition, labels were used that were associated with a specific fault conditions. The algorithms learned the different

classifications and the associated features and stored the knowledge for diagnostic testing.

9.3 Diagnostics

The diagnostic tests for the LAPART and SVM were both presented with one month of data that were previously unseen during the training process. The results from each of the tests produced TP, FN, FP, and TN values for each fault types. The analysis considered these values in the F-score statistic that was defined in Section 5.7.2. Finally, the scores for each algorithm were compared to define the best algorithm for this type of data.

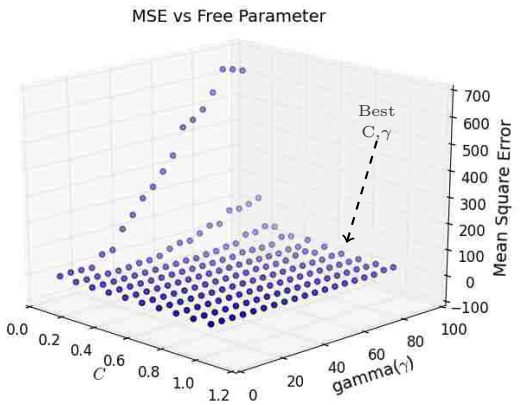

Figure 9.4: MSE as a function of the free parameter. At high C values the MSE was 0. The MSE increased as the gamma value increased at low C.

9.3.1 LAPART

The diagnostic results for the LAPART algorithm for each of the fault types are presented in Table 9.1. The diagnostics of the Type A fault was very successful and did not produce any false alarms or missed detections. This produced a precision and a probability of detection equal to 100%. The second fault, Type B, resulted in 17 false alarms and 153 false detection. As a result the precision and probabil-

Table 9.1: The LAPART results for each of the fault types

	Fault A			Fault B			Fault C			Fault D	
	$\hat{y}$=1	$\hat{y}$=0		$\hat{y}$=1	$\hat{y}$=0		$\hat{y}$=1	$\hat{y}$=0		$\hat{y}$=1	$\hat{y}$=0
y=1	257	0	y=1	150	153	y=1	513	658	y=1	72	0
y=0	0	7,342	y=0	17	7,342	y=0	13	7,342	y=0	0	7,342

ity of detection where 89% and 50% respectively. The diagnostic performance for fault type C was not perfect either, and produced 658 false detections and 13 false alarms. Diagnostics for fault type D was much more successful with a precision and probability of detection of 100%.

The F_1 score statistic (Equation 5.7) was calculated based on the micro-averaging precision provided by Equation 9.1:

$$\text{Micro-Avg Precision} = \frac{\frac{257}{257+0} + \frac{150}{150+17} + \frac{513}{513+13} + \frac{72}{72+0}}{4} = 0.97 \tag{9.1}$$

and the micro-averaging recall defined by Equation 9.2:

$$\text{Micro-Avg Recall} = \frac{\frac{257}{257+0} + \frac{150}{150+153} + \frac{513}{513+658} + \frac{72}{72+0}}{4} = 0.73 \tag{9.2}$$

The micro-averaging precision and recall for the LAPART were 0.97 and 0.73 respectively. These values were substituted into the F_1 score statistic equation and computed a value of 0.83 as shown in Equation 9.3:

$$F_1\text{Score}_{lapart} = \frac{2(0.97)(0.73)}{(0.97 + 0.73)} = 0.83 \tag{9.3}$$

9.3.2 Multi-Class SVM

The multi-class SVM algorithm developed in Python script using the Sklearn package [95] produced successful results as shown in Table 9.2. The TP, FN, FP, and TN results for the four different fault scenarios are defined in this table. The SVM algorithm was able to diagnose the Type A fault with a precision of 100% and a

Table 9.2: The Multi-class SVM results for each of the fault types

	Fault A			Fault B			Fault C			Fault D	
	y=1	y=0		y=1	y=0		y=1	y=0		y=1	y=0
y=1	218	335	y=1	185	115	y=1	1073	127	y=1	72	0
y=0	0	7,266	y=0	0	7,266	y=0	106	7,266	y=0	0	6729

probability of detection of 86%. The algorithm's ability to diagnose the second fault, Type B, was not as effective with a precision and probability of detection equal to 100% and 62% respectively. The Type C fault resulted in a precision of 91% and a probability of detection equal to 89%. The final fault, Type D, was easily detected accurately, and had a precision and probability of detection equal to 100%.

The micro-averaged precision was calculated in Equation 9.4,

$$\text{Micro-Avg Precision} = \frac{\frac{218}{218+0} + \frac{185}{185+0} + \frac{1073}{1073+106} + \frac{72}{72+0}}{4} = 0.99 \tag{9.4}$$

and was found to be 0.99. The micro-averaged recall value was also calculated as shown in Equation 9.5:

$$\text{Micro-Avg Recall} = \frac{\frac{218}{218+335} + \frac{185}{185+115} + \frac{1073}{1073+127} + \frac{72}{72+0}}{4} = 0.73 \tag{9.5}$$

The micro-averaged precision and recall were then used in Equation 9.6 to

$$\text{F1 Score}_{svm} = \frac{2(0.99)(0.73)}{(0.99+0.73)} = 0.84 \tag{9.6}$$

calculate an F_1 score of 84% for the SVM algorithm.

9.3.3 Comparison

The precision and recall results of the LAPART and SVM algorithms for the four faults are shown in Figure 9.5. Both of the algorithms performed very well. For each of them their precision and TPR results were close to one. The LAPART algorithm was able to correctly detect every fault and avoid any false alarms for Type B and

D faults. The SVM algorithm was able to accomplish perfect classification for the Type D fault.

Overall the LAPART and SVM algorithms produced very similar results. For instance, the harmonic mean of the precision and recall for the the LAPART algorithm was 97% and 73% respectively. Similarly, the precision and recall for SVM was 99% and 73% respectively. These values were used as inputs into the F_1 score equation and calculated 83% for the LAPART and 84% for the SVM. Because these results were so similar a further investigation was performed that considered the false positive versus false negative tradeoff. The LAPART algorithm had a total of 30 false positives, whereas the SVM had 106. However, the LAPART only had 811 false negatives versus the SVM, that had 577 false negatives. Definition of the best algorithm depends on the application. If a low number of false alarms is required then the LAPART algorithm should be used. If a low number of false negatives is desired then the multi-class SVM is the best choice.

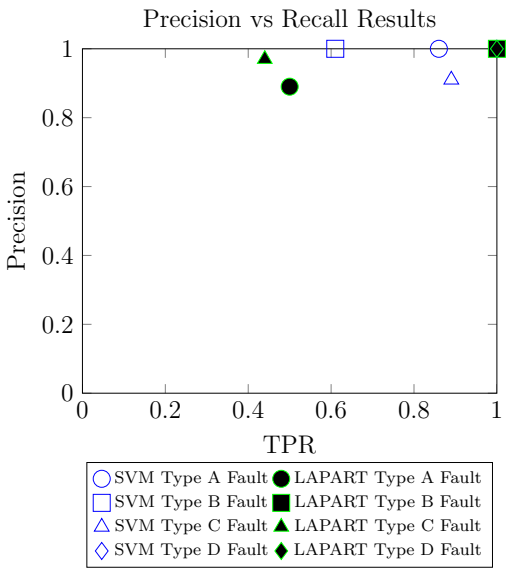

Figure 9.5: Overall TPR and precision results for Type A, B, C and D faults

Chapter 10

Conclusions & Future Work

It is well known that the commercial building sector consumes about 20% of the total energy in the U.S. A significant portion of the energy is wasted due to sub-system HVAC faults. The present work implemented a sophisticated platform that performed detailed fault detection and diagnostics on a single AHU system located in the mechanical room of the UNM MEBldg. The IT framework included sensor network extraction and data storage and visualization. The present work evaluated multiple FDD tools such as threshold method, rule-based expressions, regression, SVM, BP, LAPART, and ART. The intent was to test the FDD tools' abilities to detect faults, adapt to changes in normal behavior, and diagnose faults.

10.1 AHU Sensors

The AHU system was broken out into three sub-systems: mixed air section, HX section, and the centrifugal fan. The mixed air and HX sections were each monitored with a temperature sensor. The fan sub-system was monitored with a air flow measurement device. The device was constructed as a sub-task to the present work

to measure the static and total pressure caused by the fan. It was constructed at a low cost and was installed successfully in many locations throughout the building to monitor air flow rates. The device could potentially be used in many retrofit or new construction applications to monitor air flow reliably at a low cost.

10.2 IT Infrastructure

The IT infrastructure utilized the existing BAS sensor network (BACnet) and extracted sensor data using an off the shelf data web client [56]. The data was stored in a MySQL database where it was accessed by a visualization interface and the seven FDD tools. The transfer of sensor data from the BACnet to the MySQL database was accomplished in an automated fashion, and provided sufficient data for the FDD analysis.

10.3 Fault Detection

The tools were evaluated based on the probability of detection, probability of false alarm, and the precision. However, there is no defined metric for finding the optimal tool. For instance, the best tool depends on the user preference. If energy savings is the goal and maintenance man power is not a concern than a high quality of false alarms can be tolerated and a tool with the highest probability of detection can be chosen regardless of the false alarm rate. The opposite can be true as well, where the maintenance staff may want to avoid false alarms and chose to miss some faults for a more reliable detection tool. For example, the threshold method is a common tool found in BAS that has a very low provability for false alarm (0%), but a very low probability for detection (11%).

The fault detection test (Experiment 2) used 7,242 normal and 1,891 fault data points to evaluate all seven tools. Overall the tool with the lowest probability of false alarm was the ART algorithm at 2%. The ART algorithm also had the second highest overall probability of detection (60%), and the third highest precision (87%). The SVM and LAPART tools also performed well with precision rates of about 95% and 76% respectively. These three tools were then used in the third experiment to evaluate their abilities to adapt.

10.4 Adaptability

In man cases sensor data may change statistically and not because a fault condition has occurred. When this happens the FD tools must adapt. The third experiment evaluated the capabilities of the ART, LAPART, and SVM algorithms to adapt to changes in the AHU performance when the occupancy schedule changed from summer to fall semester. The initial training was the same as Experiment 2 which included 17,568 normal data points and then the algorithms were tested on summer and fall occupancy levels. This initial test provided a baseline assessment of the algorithms abilities to detect faults. Then the training set was updated to 27,360 data points and included data from the new occupancy level. Finally, the trained algorithms were tested on 3,126 and 260 normal and fault data points respectively.

In the baseline case the ART was the worst performer followed by SVM and then LAPART. After the algorithms were updated the ART algorithm performed better than the other two algorithms with a probability of detection and false alarm equal to 84% and 1.6% respectively. The algorithms were able to adjust and provide acceptable results quickly.

10.5 Fault Diagnostics

The final experiment evaluated the abilities of the LAPART and multi-class SVM to diagnose faults. The experiment trained the two algorithms with data that contained normal (13,672) and fault (1,496) behavior that were labeled. The algorithms were then tested on 7,372 and 1,844 normal and fault conditions respectively. The F_1 statistics was used to determine their success and the SVM had a score of 84% and LAPART was 83%. Each tool had a very high precision of 97% and 99% for the LAPART and SVM respectively. The experiment results indicated that the two algorithms could successfully perform fault diagnostics given a training set that contained labeled normal and fault conditions.

10.6 Future Work

The process history methods, which include machine learning and neural network techniques, successfully performed fault detection and diagnostics on the AHU 2 apparatus. This type of approach can be scaled to more AHU as well as other HVAC sub-systems. There is a significant potential for the IT infrastructure and the FDD tools to be integrated into maintenance and/or energy performance contractor operations. Future research work could apply the tools used in the present work to new data sets. Research could also expand the present work to evaluate the transformation of learning. In this case learning could be based on data from a single building, and then the gained knowledge can be applied for fault detection on a different building.

References

[1] *An Office Building Occupants Guide to Indoor Air Quality*, volume Indoor Environments Division (6609J). Washington, DC 20460, October 1997.

[2] Energy Efficiency Trends in Residential and Commercial Buildings. Technical report, U.S. Department of Energy, October 2008.

[3] BACnet, A Data Communication Protocol for Building Automation and Control Networks, 2012.

[4] Buildings Energy Data Book, March 2012.

[5] Python SciPy, 2014.

[6] ASHRAE. ASHRAE Guideline 14-2002 Measurement of Energy and Demand Savings, 2002.

[7] Guilherme A. Barreto and Leonardo Aguayo. Time Series Clustering for Anomaly Detection Using Competitive Neural Networks. In Jos C. Prncipe and Risto Miikkulainen, editors, *Advances in Self-Organizing Maps*, number 5629 in Lecture Notes in Computer Science, pages 28–36. Springer Berlin Heidelberg, 2009.

[8] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June 2004.

[9] Andrew P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, July 1997.

[10] Christopher J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, June 1998.

[11] G.A. Carpenter and S. Grossberg. The ART of adaptive pattern recognition by a self-organizing neural network. *Computer*, 21(3):77–88, March 1988.

[12] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, and D.B. Rosen. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3(5):698–713, September 1992.

[13] G.A. Carpenter, S. Grossberg, and D.B. Rosen. Fuzzy ART: an adaptive resonance algorithm for rapid, stable classification of analog patterns. In *, IJCNN-91-Seattle International Joint Conference on Neural Networks, 1991*, volume ii, pages 411–416 vol.2, July 1991.

[14] Gail A. Carpenter and Stephen Grossberg. ART 2: self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26(23):4919–4930, December 1987.

[15] Gail A. Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37(1):54–115, January 1987.

[16] Gail A. Carpenter, Stephen Grossberg, and David B. Rosen. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4(6):759–771, 1991.

[17] Gail A. Carpenter, Mark A. Rubin, and William W. Streilein. ARTMAP-FD: Familiarity Discrimination Applied to Radar Target Recognition. Technical Report, Boston University Center for Adaptive Systems and Department of Cognitive and Neural Systems, November 1996. Advanced Research Projects Agency; National Science Foundation (IRI-94-01659); Office of Naval Research (N00014-95-1-0657, N00014-95-0409, N00014-96-0659).

[18] Gail A. Carpenter, Mark A. Rubin, and William W. Streilein. Threshold Determination for ARTMAP-FD Familiarity Discrimination. Technical Report, Boston University Center for Adaptive Systems and Department of Cognitive and Neural Systems, May 1997. Advanced Research Projects Agency; Office of Naval Research (N00011-95-1-0657, N00011-95-0109, NOOOB-96-0659); National Science Foundation (IRI-94-01659).

[19] Thomas P. Caudell and David S. Newman. An adaptive resonance architecture to define normality and detect novelties in time series and databases. *In IEEE World Congress on Neural Networks*, pages 166–176, 1993.

[20] S. Chen, C.F.N. Cowan, and P.M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309, March 1991.

[21] K. Choi, S.M. Namburu, M.S. Azam, Jianhui Luo, K.R. Pattipati, and A. Patterson-Hine. Fault diagnosis in HVAC chillers. *IEEE Instrumentation Measurement Magazine*, 8(3):24–32, August 2005.

[22] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[23] Jingtan Cui and Shengwei Wang. A model-based online fault detection and diagnosis strategy for centrifugal chiller systems. *International Journal of Thermal Sciences*, 44(10):986–999, October 2005.

[24] Davood Dehestani, Fahimeh Eftekhari, Ying Guo, Steven Ling, Steven Su, and Hung Nguyen. Online Support Vector Machine Application for Model Based Fault Detection and Isolation of HVAC System. *International Journal of Machine Learning and Computing*, 1(1):66–72, April 2011.

[25] Donald D. Dorfman and Edward Alf Jr. Maximum-likelihood estimation of parameters of signal-detection theory and determination of confidence intervalsRating-method data. *Journal of Mathematical Psychology*, 6(3):487–496, October 1969.

[26] Stephan Dreiseitl, Lucila Ohno-Machado, Harald Kittler, Staal Vinterbo, Holger Billhardt, and Michael Binder. A Comparison of Machine Learning Methods for the Diagnosis of Pigmented Skin Lesions. *Journal of Biomedical Informatics*, 34(1):28–36, February 2001.

[27] Kaibo Duan, S. Sathiya Keerthi, and Aun Neow Poo. Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing*, 51:41–59, April 2003.

[28] R Dybowski, V Gant, P Weller, and R Chang. Prediction of outcome in critically ill patients using artificial neural network synthesised by genetic algorithm. *The Lancet*, 347(9009):1146–1150, April 1996.

[29] K.M. Edlund and T.P. Caudell. Architecture as it controls a simulated autonomous vehicle. In *IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, 2000*, volume 3, pages 41–46 vol.3, 2000.

[30] Hikmet Esen, Mustafa Inalli, Abdulkadir Sengur, and Mehmet Esen. Modeling a ground-coupled heat pump system by a support vector machine. *Renewable Energy*, 33(8):1814–1823, August 2008.

[31] J. Y. Fan, M. Nikolaou, and R. E. White. An approach to fault diagnosis of chemical processes via neural networks. *AIChE Journal*, 39(1):82–88, January 1993.

[32] Tom Fawcett. An Introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[33] Jess Febres, Raymond Sterling, and Marcus Keane. A novel calibration methodology for heating coil models using real data and modelica models. In *2014 ASHRAE/IBPSA Building Simulation Conference*, Atlanta, GA, September 2014.

[34] M. A. Fischler and O. Firschein. Intelligence: The eye, the brain, and the computer. January 1987.

[35] T. Fortmann, Y. Bar-Shalom, M. Scheffe, and Saul Gelfand. Detection thresholds for tracking in clutterA connection between estimation and signal processing. *IEEE Transactions on Automatic Control*, 30(3):221–229, March 1985.

[36] M. Georgiopoulos, I. Dagher, G. L. Heileman, and G. Bebis. Properties of learning of a Fuzzy ART Variant. *Neural Networks*, 12(6):837–850, July 1999.

[37] A.S. Glass, P. Gruber, M. Roos, and J. Todtli. Qualitative model-based fault detection in air-handling units. *IEEE Control Systems*, 15(4):11–22, August 1995.

[38] Stephen Grossberg. Adaptive pattern classification and universal recoding: II. Feedback, expectation, olfaction, illusions. *Biological Cybernetics*, 23(4):187–202, December 1976.

[39] Stephen Grossberg. Competitive Learning: From Interactive Activation to Adaptive Resonance. *Cognitive Science*, 11(1):23–63, January 1987.

[40] Gabsoo Han, F.M. Ham, and L.V. Fausett. Fuzzy LAPART supervised learning through inferencing for stable category recognition. In *, Proceedings of the Third IEEE Conference on Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence*, pages 46–51 vol.1, June 1994.

[41] H. Han, B. Gu, T. Wang, and Z. R. Li. Important sensors for chiller fault detection and diagnosis (FDD) from the perspective of feature selection and

machine learning. *International Journal of Refrigeration*, 34(2):586–599, March 2011.

[42] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Springer Science+Business Media, 2009.

[43] Simon Haykin. *Neural Networks A Comprehensive Foundation*. Tom Robbins, Prentice-Hall, Inc., 2nd edition, 1999.

[44] Hongbo He, Thomas P. Caudell, David F. Menicucci, and Andrea A. Mammoli. Application of Adaptive Resonance Theory neural networks to monitor solar hot water systems and detect existing or developing faults. *Solar Energy*, 86(9):2318–2333, September 2012.

[45] Hongbo He, David Menicucci, Thomas Caudell, and Andrea Mammoli. Real-Time Fault Detection for Solar Hot Water Systems Using Adaptive Resonance Theory Neural Networks. In *ASME 2011 5th International Conference on Energy Sustainability*, volume ES2011, Washington, D.C., August 2011.

[46] M.J. Healy. A logical architecture for supervised learning. In *1991 IEEE International Joint Conference on Neural Networks, 1991*, pages 190–195 vol.1, November 1991.

[47] M.J. Healy and T.P. Caudell. Acquiring rule sets as a product of learning in a logical neural architecture. *IEEE Transactions on Neural Networks*, 8(3):461–474, May 1997.

[48] M.J. Healy, T.P. Caudell, and S.D.G. Smith. A neural architecture for pattern sequence verification through inferencing. *IEEE Transactions on Neural Networks*, 4(1):9–20, 1993.

[49] Victoria J. Hodge and Jim Austin. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22(2):85–126, October 2004.

[50] J.C. Hoskins, K.M. Kaliyur, and D. M. Himmelblau. Fault diagnosis in complex chemical plants using artificial neural networks. *American Institute of Chemical Engineers Journal*, 37(1):137–141, 1991.

[51] J.M House, W.Y. Lee, and D.R. Shin. Classification techniques for fault detection and diagnosis of an air-handling unit. *ASHRAE Transactions*, 105(1):1087–1097, 1999.

[52] Ronald L. Howell, William J. Coad, and Harry J. Sauer, Jr. *Principles of Heating Ventilating and Air Conditioning*. American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., 6 edition, 2009.

[53] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, March 2002.

[54] Rolf Isermann. Process fault detection based on modeling and estimation methodsA survey. *Automatica*, 20(4):387–404, July 1984.

[55] Birk Jones and Susan M Bogus. Decision Process for Energy Efficient Building Retrofits: The Owner's Perspective. *Journal of Green Building*, 5(3):131–146, August 2010.

[56] C. Birk Jones, Andrea Mammoli, Larry Schuster, Hans Barsun, and Richard Burnett. Implementation of IT Infrastructure for Model based Real-time HVAC Diagnostics. *Procedia Computer Science*, 19:654–661, 2013.

[57] Tapas Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, July 2002.

[58] Srinivas Katipamula and Michael R. Brambley. Review Article: Methods for Fault Detection, Diagnostics, and Prognostics for Building SystemsA Review, Part 1. *HVAC&R Research*, 11(1):3–25, 2005.

[59] Srinivas Katipamula and Michael R. Brambley. Review Article: Methods for Fault Detection, Diagnostics, and Prognostics for Building SystemsA Review, Part 2. *HVAC&R Research*, 11(2):169–187, 2005.

[60] A. Khotanzad, R. Afkhami-Rohani, T.-L. Lu, A. Abaye, M. Davis, and D.J. Maratukulam. ANNSTLF-a neural-network-based electric load forecasting system. *IEEE Transactions on Neural Networks*, 8(4):835–846, July 1997.

[61] Sila Kiliccote, Mary Ann Piette, and David Hansen. Advanced Controls and Communications for Demand Response and Energy Efficiency in Commercial Buildings. *Lawrence Berkeley National Laboratory*, January 2006.

[62] Rex Klopfenstein Jr. Air velocity and flow measurement using a Pitot tube. *ISA Transactions*, 37(4):257–263, September 1998.

[63] S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In Franoise Fogelman Souli and Jeanny Hrault, editors, *Neurocomputing*, number 68 in NATO ASI Series, pages 41–50. Springer Berlin Heidelberg, 1990.

[64] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, September 1990.

[65] Miroslav Kubat, Robert C. Holte, and Stan Matwin. Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning*, 30(2-3):195–215, February 1998.

[66] Andrew Kusiak, Mingyang Li, and Fan Tang. Modeling and optimization of HVAC energy consumption. *Applied Energy*, 87(10):3092–3102, October 2010.

[67] Khee Poh Lam, Jie Zhao, Eric B Ydstie, Jason Wirick, Meiwei Qi, and Jihyun Park. An EnergyPlus whole building energy model calibration method for office buildings using occupant behavior data mining and and empirical data. In *2014 ASHRAE/IBPSA Building Simulation Conference*, Atlanta, GA, September 2014.

[68] K.Y. Lee, Y.T. Cha, and J.H. Park. Short-term load forecasting using an artificial neural network. *IEEE Transactions on Power Systems*, 7(1):124–132, February 1992.

[69] Won-Yong Lee, John M. House, Cheol Park, and George E. Kelly. Fault diagnostics of an air-handling unit using artificial neural networks. *ASHRAE Transactions*, 102:540–549, 1996.

[70] Qiong Li, Qinglin Meng, Jiejin Cai, Hiroshi Yoshino, and Akashi Mochida. Applying support vector machine to predict hourly cooling load in the building. *Applied Energy*, 86(10):2249–2256, October 2009.

[71] J. Liang and R. Du. Model-based Fault Detection and Diagnosis of HVAC systems using Support Vector Machine method. *International Journal of Refrigeration*, 30(6):1104–1114, September 2007.

[72] J. Ma and S. Perkins. Time-series novelty detection using one-class support vector machines. In *Proceedings of the International Joint Conference on Neural Networks, 2003*, volume 3, pages 1741–1745 vol.3, July 2003.

[73] Junshui Ma and Simon Perkins. Online Novelty Detection on Temporal Sequences. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 613–618, New York, NY, USA, 2003. ACM.

[74] A. Mammoli, A. Menicucci, T. Caudell, A. Ellis, S. Willard, and J. Simmins. Low-cost solar micro-forecasts for PV smoothing. In *2013 1st IEEE Conference on Technologies for Sustainability (SusTech)*, pages 238–243, August 2013.

[75] Andrea Mammoli, Peter Vorobieff, Hans Barsun, Rick Burnett, and Daniel Fisher. Energetic, economic and environmental performance of a solar-thermal-assisted HVAC system. *Energy and Buildings*, 42(9):1524–1535, September 2010.

[76] C. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, England, April 2009.

[77] Markos Markou and Sameer Singh. Novelty detection: a reviewpart 2:: neural network based approaches. *Signal Processing*, 83(12):2499–2521, December 2003.

[78] Stephen Marsland. Novelty Detection in Learning Systems. *Neural Computing Surveys*, 3:1–39, 2002.

[79] M. R. Maurya, R. Rengaswamy, and V. Venkatasubramanian. A Signed Directed Graph and Qualitative Trend Analysis-Based Framework for Incipient Fault Diagnosis. *Chemical Engineering Research and Design*, 85(10):1407–1422, 2007.

[80] Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. *Elements of Artificial Neural Networks*. MIT, 1997.

[81] Charles E. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 8(4):283–298, October 1978.

[82] Anthony Molinaro. *SQL Cookbook*. O'Reilly Media, Inc, Sebastopol, CA, 1st edition, 2006.

[83] Olivier Morisot and Dominique Marchio. Fault detection and diagnosis on HVAC variable air volume system using artificial neural networks. In *IBPSA Building Simulation*, Kyoto, Japan, 1999.

[84] M.R. Moya, M.W Koch, and L.D. Hostetler. One-class classifier networks for target recognition applications. In *Proceedings on World Congress on Neural Networks*, pages 797–801, Portland, OR, 1993.

[85] Kevin P Murphy. *Machine Learning A Probabilistic Perspective*. MIT Press, Cambridge, MA, 2012.

[86] H. Michael Newman and Ebooks Corporation. *BACnet the global standard for building automation and control networks*. Momentum Press, New York, 2013.

[87] L. K. Norford and R.D. Little. Fault detection and monitoring in ventilation systems. *ASHRAE Transactions*, 99(1):590–602, 1993.

[88] L. K. Norford, J. A. Wright, R. A. Buswell, D. Luo, C. J. Klaassen, and A. Suby. Demonstration of Fault Detection and Diagnosis Methods for Air-Handling Units. *HVAC&R Research*, 8(1):41–71, January 2002.

[89] US Department of Energy. Commercial Buildings Energy Consumption Survey (CBECS), 2012.

[90] Zheng O'Neil, Madhusudana Shashanka, Xiufeng Pang, Prajesh Bhattacharya, Trevor Bailey, and Philip Haves. Real time model-based energy diagnostics in buildings. In *Proceedings of Building Simulation 2011*, pages 474–481, Sydney, NSW, Australia, 2011. International Building Performance Simulation Association, United States.

[91] M. Ortiz, H. Barsun, H. He, P. Vorobieff, and A. Mammoli. Modeling of a solar-assisted HVAC system with thermal storage. *Energy and Buildings*, 42(4):500–509, April 2010.

[92] Yiqun Pan, Zhizhong Huang, and Gang Wu. Calibrated building energy simulation and its application in a high-rise commercial building in Shanghai. *Energy and Buildings*, 39(6):651–657, June 2007.

[93] Xiufeng Pang, Prajesh Bhattacharya, Zheng O'Neil, Philip Haves, Michael Wetter, and Trevor Bailey. Real-time Building Energy Simulation using EnergyPlus and the Building Controls Virtual Test Bed. In *Proceedings of Building Simulation 2011*, Sydney, Aust, November 2011.

[94] Xiufeng Pang, Michael Wetter, Prajesh Bhattacharya, and Philip Haves. A framework for simulation-based real-time whole building performance assessment. *Building and Environment*, 54:100–108, August 2012.

[95] Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and douard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:28252830, October 2011.

[96] H.C Peitsman and L.L Soethout. ARX models and real-time model-based diagnosis. *ASHRAE Transactions*, 103(1):657–671, 1997.

[97] A. Rabl. Parameter Estimation in Buildings: Methods for Dynamic Analysis of Measured Energy Use. *Journal of Solar Energy Engineering*, 110(1):52–66, February 1988.

[98] R. Radhakrishnan, D. Nikovski, K. Peker, and A. Divakaran. A Comparison between Polynomial and Locally Weighted Regression for Fault Detection and Diagnosis of HVAC Equipment. In *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*, pages 3668–3673, November 2006.

[99] Paul Raftery, Marcus Keane, and Andrea Costa. Calibrating whole building energy models: Detailed case study using hourly measured data. *Energy and Buildings*, 43(12):3666–3679, December 2011.

[100] Paul Raftery, Marcus Keane, and James ODonnell. Calibrating whole building energy models: An evidence-based methodology. *Energy and Buildings*, 43(9):2356–2364, September 2011.

[101] Kurt W. Roth, Detlef Westphalen, Patricia Llana, and Michael Feng. The energy impact of faults in u.s. commercial buildings. *International Refrigeration and Air Conditioning Conference at Purdue*, (665), July 2004.

[102] T. I Salsbury and R. C Diamond. Fault detection in HVAC systems using model-based feedforward control. *Energy and Buildings*, 33(4):403–415, April 2001.

[103] Jeffrey Schein, Steven T. Bushby, Natascha S. Castro, and John M. House. A rule-based fault detection method for air handling units. *Energy and Buildings*, 38(12):1485–1492, December 2006.

[104] Bernhard Scholkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support Vector Method for Novelty Detection. *NIPS*, 12:582–588, 1999.

[105] Bernhard Schlkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, 13(7):1443–1471, July 2001.

[106] S.D.G. Smith and R.A. Escobedo. Engineering and manufacturing applications of ART-1 neural networks. In *, 1994 IEEE International Conference on Neural Networks, 1994. IEEE World Congress on Computational Intelligence*, volume 6, pages 3780–3785 vol.6, June 1994.

[107] P. Soliz, T.P. Caudell, and D.R. Hush. The creation of human-friendly rules for long-range weather prediction using the LAPART neural architecture. In *, 1997 IEEE International Conference on Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation*, volume 3, pages 2561–2566 vol.3, October 1997.

[108] Peter Soliz, Marios S. Pattichis, Janakiramanan Ramachandran, and David S. James. Computer-assisted diagnosis of chest radiographs for pneumoconioses. volume 4322, pages 667–675, 2001.

[109] K. A. Spackman. Maximum likelihood training of connectionist models: comparison with least squares back-propagation and logistic regression. *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pages 285–289, 1991.

[110] Raymond Sterling, Gregory Provan, Jess Febres, Dominic O'Sullivan, Peter Struss, and Marcus M. Keane. Model-based Fault Detection and Diagnosis of Air Handling Units: A Comparison of Methodologies. *Energy Procedia*, 62:686–693, 2014.

[111] John A. Swets, Robyn M. Dawes, and John Monahan. Better DECISIONS through SCIENCE. *Scientific American*, pages 82–87, October 2000.

[112] L.H. Ungar, B.A. Powell, and S.N. Kamens. Adaptive networks for fault diagnosis and process control. *Computers and Chemical Engineering*, 14(4-5):561–572, 1990.

[113] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer Science & Business Media, 2000.

[114] V.N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, September 1999.

[115] S. Vasilic and M. Kezunovic. Fuzzy ART neural network algorithm for classifying the power system faults. *IEEE Transactions on Power Delivery*, 20(2):1306–1314, April 2005.

[116] Venkat Venkatasubramanian and King Chan. A neural network methodology for process fault diagnosis. *AIChE Journal*, 35(12):1993–2002, December 1989.

[117] Venkat Venkatasubramanian, Raghunathan Rengaswamy, and Surya N Kavuri. A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies. *Computers & Chemical Engineering*, 27(3):313–326, March 2003.

[118] Venkat Venkatasubramanian, Raghunathan Rengaswamy, Surya N. Kavuri, and Kewen Yin. A review of process fault detection and diagnosis: Part III: Process history based methods. *Computers & Chemical Engineering*, 27(3):327–346, March 2003.

[119] Venkat Venkatasubramanian, Raghunathan Rengaswamy, Kewen Yin, and Surya N. Kavuri. A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. *Computers & Chemical Engineering*, 27(3):293–311, March 2003.

[120] Shengwei Wang, Qiang Zhou, and Fu Xiao. A system-level fault detection and diagnosis strategy for HVAC systems involving sensor faults. *Energy and Buildings*, 42(4):477–490, April 2010.

[121] Kajiro Watanabe, Ichiro Matsuura, Masahiro Abe, Makoto Kubota, and D. M. Himmelblau. Incipient fault diagnosis of chemical processes via artificial neural networks. *AIChE Journal*, 35(11):1803–1812, November 1989.

[122] P Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.

[123] Michael Wetter. Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed. *Journal of Building Performance Simulation*, 4(3):185–203, November 2010.

[124] J. R. Whiteley and J. F. Davis. A similarity-based approach to interpretation of sensor data using adaptive resonance theory. *Computers & Chemical Engineering*, 18(7):637–661, July 1994.

[125] Achmad Widodo and Bo-Suk Yang. Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical Systems and Signal Processing*, 21(6):2560–2574, August 2007.

[126] M.W. Wildin. Results from Use of Thermally Stratified Water Tanks to Heat and Cool the Mechanical Engineering Building at the University of New Mexico. Technical report, Oak Ridge National Laboratory, Oak Ridge, Tennessee, June 1983.

[127] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[128] K. WORDEN, H. SOHN, and C. R. FARRAR. Novelty detection in changing environment: regression and interpolation approaches. *Journal of Sound and Vibration*, 258(4):741–761, December 2002.

[129] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability Estimates for Multi-class Classification by Pairwise Coupling. *J. Mach. Learn. Res.*, 5:975–1005, December 2004.

[130] Markus Wlchli and Torsten Braun. Efficient Signal Processing and Anomaly Detection in Wireless Sensor Networks. In Mario Giacobini, Anthony Brabazon, Stefano Cagnoni, Gianni A. Di Caro, Anik Ekrt, Anna Isabel Esparcia-Alczar, Muddassar Farooq, Andreas Fink, and Penousal Machado, editors, *Applications of Evolutionary Computing*, number 5484 in Lecture Notes in Computer Science, pages 81–86. Springer Berlin Heidelberg, 2009.

[131] Bo-Suk Yang, Tian Han, and Yong-Su Kim. Integration of ART-Kohonen neural network and case-based reasoning for intelligent fault diagnosis. *Expert Systems with Applications*, 26(3):387–395, April 2004.

[132] H. Yoshida, T Iwami, H Yuzawa, and M. Suzuki. Typical faults of air-conditioning systems, and fault detection by ARX Model and extended kalman filter. *ASHRAE Transactions*, 102(1):557–564, 1996.

[133] H. Yoshida and S. Kumar. ARX and AFMM model-based on-line real-time data base diagnosis of sudden fault in AHU of VAV system. *Energy Conservation and Management*, 40:1191–1206, 1999.

[134] M. H. Zweig and G. Campbell. Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine. *Clinical Chemistry*, 39(4):561–577, April 1993.