Unsupervised Learning for Anomaly Detection in Fleet Based Systems: Application to Air Conditioners

MSc Data Science Dissertation

Candidate Number: LMHB8

Word Count: 10054

Supervisor: Professor Ricardo Silva

Department of Statistical Science

University College London

Contents

List of Figures

List of Tables

1 Introduction

Heating, ventilation, and air-conditioning (HVAC) systems are responsible for a large proportion of total global energy consumption.[1] According to the US Department of Energy, buildings account for 72% of the national electrical energy consumption, 38.9% of the national total energy consumption.[1, 2] Out of the total energy consumed by US commercial and residential buildings, HVAC systems are responsible for 57% of that energy consumption.[1, 2] Unfortunately, HVAC systems often do not meet their performance expectations due to various faults, which are expected to waste over 20% of the machine's energy consumption.[1] According to the National Institute of Standards and Technology (NIST), fault detection and diagnosis methods have the potential to reduce the energy consumption of HVAC systems by 10-40%.[3] For this reason, it is imperative to develop automated monitoring systems to detect faults as early as possible.

In hot countries, air conditioners are responsible for the majority of the HVAC energy consumption, where the compressor consumes the largest amount of electricity.[4] In Singapore, the cooling systems alone account for more than 60% of their buildings' total energy consumption.[5] Furthermore, faults in the cooling system can significantly increase energy consumption. For example, fouling can reduce an air conditioner's coefficient of performance (COP) by 20–30%.[6] Often, soft faults like condenser fouling, high refrigerant charges, and faulty sensors may remain unnoticed for a long time, resulting in higher than needed energy consumption.[4] For example, poorly maintained, degraded, and improperly controlled components in air conditioners are responsible for an estimated 15%–30% energy wastage in commercial buildings.[7] Therefore, automated fault detection systems for air conditioners have the potential to reduce energy consumption and improve the operating and maintenance efficiency of buildings.

Many condition monitoring approaches for air conditioners are limited to analyzing a single machine and involve supervised learning through a large historical dataset.[4] The problem with this approach is that it requires a large dataset annotated with all the possible faulty and healthy conditions under all possible operating conditions. Such a dataset is uncommon and challenging to create in industrial applications due to the complexity of faulty conditions and operating conditions. We have taken an entirely novel approach to air conditioner monitoring through fleet based monitoring. Fleet based monitoring has been shown effective in situations where multiple machines run simultaneously under the same conditions. Some examples are wind turbines [8], aerospace engines (which often operate in pairs) [9], and in train and conveyor belts (e.g., the traction motors, bearings and bogies [10, 11]). Fleet based monitoring assumes that most machines will be in a healthy operating condition at any given time. A novelty detection algorithm is then used to detect if a machine is in a healthy 'normal' state or a faulty 'anomalous' state. This approach was chosen because air conditioners often operate in similar conditions and will run at the same time. This approach also does not require a large historical dataset which is an issue faced by prior researchers.[4] Our fleet based approach for air conditioner monitoring using current and voltage data was shown to be highly effective in fault detection monitoring for air conditioners.

We will begin with a literature review on past approaches taken by researchers for fault and anomaly detection. Additionally, we will cover several alternative distancing metrics to euclidean distance. Next, we will cover the domain knowledge required for understanding the air conditioner data for fault detection. We will do this by first thoroughly covering how an air conditioner works and breaks, and the data collection methodology. Next, we will cover feature engineering and why real power, reactive power, and total harmonic distortion are the features used. Afterward, we will explain our anomaly detection approach and the unsupervised learning algorithms implemented. Subsequently, we will discuss the results of our fleet based fault detection approach and then conclude with future recommendations.

2 Literature Review

2.1 Overview

In recent years, radical improvements made to sensor technology, which has increased research in anomaly detection through sensor data collected during machine operation.[12] Due to the large variety of scenarios and problems, many approaches taken have been implemented by researchers for anomaly detection. One common way these approaches are categorized is the level of supervision required by the algorithm; supervised learning, semi-supervised learning, unsupervised learning.[13] Another way of categorization is based on the methods used, such as probabilistic models, statistical models, linear models, proximity based models, outlier detection, and clustering.[13] Researchers have also applied machine learning and deep learning methods for anomaly detection, such as convolutional neural networks and generative adversarial networks.[4, 14, 15] In this section, we will cover commonly used anomaly detection techniques in time series from recent academic literature. First, we will discuss the traditional engineering approach and the fully supervised approaches for anomaly detection, which both require data annotated by healthy and faulty conditions. Afterward, we will discuss the semi-supervised approach, which requires only healthy condition data for training. Subsequently, we will discuss the unsupervised approaches that require no labeled data. Lastly, we will cover alliterative distancing metrics other than Euclidean distance.

2.2 Traditional Engineering Approach

The traditional anomaly detection approach in engineering is done by handcrafting indicators of interest for a specific type of event.[16] This process requires expertise on a specific machine and a significant amount of testing. During deployment, the machine's health is monitored by comparing the current conditions to the preselected threshold conditions. Kilian Hendrickx, Wannes Meert, and Yves Mollet note that this approach has several drawbacks.[16] The first drawback is that different operation conditions influence machine behavior.

As a result, the traditional approach requires different thresholds, identifying all possible operating conditions, and normalizing the faulty indicator.[16–18] Additionally, rigorous testing is made unfeasible for machines that require a large number of operating conditions or are too complex. For complex systems, researchers are forced to simulate the machine for healthy behavior to choose thresholds.[19] The problem with this is that it is often very challenging to simulate the machine data, and it assumes that every parameter and machine behavior will not change over time, which is rarely the case.

2.3 Supervised Learning

In academic literature, many different methods have been proposed for supervised anomaly detection. Some common methods are statistical models, linear models, random forest models, and gradient boosting models.[13] Some popular machine learning approaches are Convolutional Neural Networks (CNN) [20], Artificial Neural Networks (ANN) [21–23], Deep Learning (DL) [24–26], and Support Vector Machine (SVM) [27–29]. The most notable drawback of these methods is that they require a large labeled dataset that is annotated with all possible healthy conditions and all possible faulty conditions.[30] Correspondingly with the traditional engineering approach, such datasets are uncommon and difficult to create in industrial applications.[31, 32] Supervised learning methods have had good results and will frequently outperform traditional engineering methods. It is common knowledge that these algorithms are powerful in pattern detection. Although, for many production applications, it is questionable if the performance is generalizable and good in practice or just good on the trained dataset due to the data collection issue. Another drawback of supervised learning approaches is that they do not scale easily to different machines. This means that a model trained on data from a specific manufacturer on a specific machine will not generalize to the same machine built by a different manufacturer. Even if the supervised learning model was applied to a single machine and by a single manufacturer, when any component is modified, new data will likely have to be collected.

2.4 Semi-supervised Learning

Classical semi-supervised learning approaches use a small amount of labeled data with a large amount of unlabeled data during training. In fault and anomaly detection, literature semi-supervised methods will use only healthy data in training and do not require any faulty data.[33, 34] This solves one of the major issues with supervised learning. A popular semi-supervised learning method is novelty detection, where you train a model on data that only contains healthy information. When the models are tested, they are able to identify outliers and data that deviates from the trained, healthy data.[35] A drawback of semi-supervised learning is that you still require a large dataset annotated with healthy data. This can be time-consuming to collect if the machine has many healthy states. Furthermore, it may be hard to guarantee that the collected data is indeed healthy and does not include faulty information in practice. This method also still assumes that the healthy state is not affected by time, which is often not the case. One common example of a semi-supervised algorithm is the One Class Support Vector Machine (OCSVM), where the algorithm learns a decision boundary that achieves the maximum separation between the points and the origin. There is an unsupervised variant of OCSVM as well, which does not require any labeled data.[36]

2.5 Unsupervised Learning

2.5.1 Statistical Learning

Unsupervised methods do not require any labeled data and address many of the drawbacks found in supervised and semi-supervised learning [35–38]. Two common unsupervised approaches used for faulty detection are anomaly detection and clustering. Unsupervised anomaly detection operates on the assumption that the majority of the data points are healthy so that the models can classify data points that are less like the majority as faulty. Clustering operates similarly but instead will group the data points into clusters of similar points, then clusters that do not contain the majority of the data points would be considered faulty. One drawback of this method is that if the majority of the data points are faulty, then the models will classify faulty data as healthy. The majority of data points could be faulty when one machine breaking causes overload on the fleet, which then creates a domino effect of damaging the other machines. There are many different unsupervised techniques for anomaly detection and clustering. We will apply Hierarchical Clustering, K Nearest Neighbors (KNN), Local Outlier Factor (LOF), unsupervised One Class Support Vector Machines (OCSVM), Isolation Forests (iForest), and Isolation Nearest Neighbors Ensembles (iNNE). More detailed explanations of these algorithms are given in section 3.

2.5.2 Deep Learning

There are unsupervised deep learning anomaly detection techniques as well. Autoencoders are a type of neural network that is trained to reproduce input data. Autoencoders have two stages: the encoder stage and then the decoder stage. The encoder stage will map the input into code, and then the decoder stage will reconstruct the code back to the input.[11] In recent years, they have been gaining popularity in anomaly and novelty detection problems.[37, 38] They detect anomalies by learning to replicate the most common features in the training data, then the model can reproduce the most common characteristics. Generative adversarial network (GAN) is another unsupervised anomaly detection technique that has been used more recently in anomaly detection.[4] A GAN consists of two neural networks, a generator, and a discriminator, where the generator will try to fool the discriminator by generating data similar to that in the training set. To utilize both of these techniques, you will typically need a considerable amount of data to generate meaningful results.[4, 37, 38] If the data is not large enough or not clean enough, the models will often deliver mixed results. In our case, the data is limited to the small number of ten air conditioners in our fleet running simultaneously, which made this approach unfeasible.

2.6 Distance metrics

We only applied the Euclidean distance metric for all the algorithms that required a distance metric for simplicity and consistency. There are potential alternatives that are worth mentioning that could improve the results. Some common alternatives to the euclidean distance are the Manhattan distance and Minkowski distance. [39] Another popular option for time series data is Dynamic Time Warping (DTW).[40]

Euclidean Distance: The Euclidean distance or L2 norm is the straight line distance between two points.

$$||x-y||_2 = \sqrt{\sum_i |x_i - y_i|^2}$$

Manhattan Distance: Manhattan distance or L1 norm is the distance between two points on a cartesian plane.

$$||x-y||_1 = \sum_i |x_i - y_i|$$

Minkowski Distance: Minkowski Distance or p-norm is a more general distance.

$$||x-y||_p = (\sum_i |x_i - y_i|^p)^{(1/p)}$$

Dynamic Time Warping: Dynamic Time Warping (DTW) is used to measure the similarity between two time series sequences.[40] The algorithm measures similarity for two times series X and Y by [16]:

$$\min_P \sqrt{\sum_{\theta=1}^{N} \left\| X_{P_{\theta,0}}, Y_{P_{\theta,1}} \right\|_2^2}$$

where $P = (P_1, P_2, ..., P_N)$ is the optimal alignment of a time series of length N, and where $P_{\theta,0}$ corresponds to the elements in time series X at position $P_{\theta,0}$ and $P_{\theta,1}$ corresponds to the elements in time series Y at position $P_{\theta,1}$. The optimal alignment satisfies three conditions:

1. $P_1 = (1,1)$ and $P_N = (n,n)$ (boundary condition)
2. $P_{1,i} \leq P_{2,i} \leq ... \leq P_{N,i}$ for $i \in \{1,2\}$ (monotonicity condition)
3. $P_{\theta+1} - P_\theta \in \{(1,0),(0,1),(1,1)\}$ for $1 \leq \theta \leq N-1$ (step a size condition)

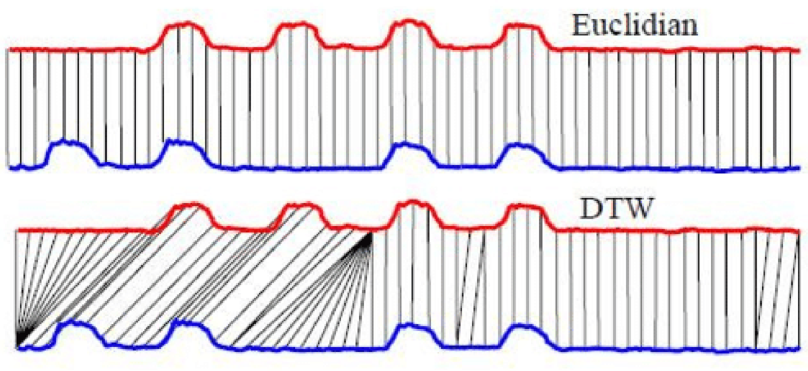

Figure 1 ***Euclidean distance vs. DTW*** [41]

3 Dataset Description

3.1 How an AC Works and Breaks

To understand the data collection process and the feature engineering, we must first go over how the cooling mechanism of an air conditioner works and its key components. There are many different variations of air conditioners, but the majority of them have the same cooling mechanism, which is also found in fridges. We will explain the most common cooling mechanism used by modern air conditioners.[42, 43] This mechanism was also used by the air conditioner from which our data was collected.[44] We will discuss the main components of an air conditioner and some common causes for air conditioner faults. The main components of an air conditioner are also similar between manufacturers, although there are variations in positioning on the device and the manufacturing quality. Generally, most air conditioner faults are shared among all manufacturers, but of course, some air conditioner models may be more or less prone to specific faults. Therefore, any specific components or faults discussed will be done in a general context and not regarding the particular model of the air conditioner used for our data collection.

An air conditioner provides cool air inside an enclosed space by removing heat and humidity from the indoor space and transferring the hot and humid air outdoors.[42–44] An air conditioner has two connected coils that have a specialized chemical called refrigerant continuously flowing inside them. One coil is positioned indoors and is called the evaporator coil. The other coil is positioned outdoors and is called the condenser coil. The cooling process works by keeping the evaporator coil colder than its surroundings and keeping the condenser coil hotter than its surroundings. When this is done, the evaporator coil can absorb the heat and release it through the condenser coil. The compressor is positioned between the evaporator coil and the condenser coil. The compressor will increase the pressure of the refrigerant in its gaseous state so that as the gas is compressed, the temperature rises. The compressor sends the hot gas to the condenser coils, where the heat is ejected, and the gas is converted to a liquid. The heat ejection is assisted by a fan placed on the condenser coils. The refrigerant will then travel back to the evaporator coils in their liquid state, where it will evaporate and cool the indoor area. Another fan positioned around the evaporator coil will blow the cool air out into the room. Finally, the heated evaporated gas is collected and sent back to the compressor. This cycle will continue and is responsible for the cooling process.

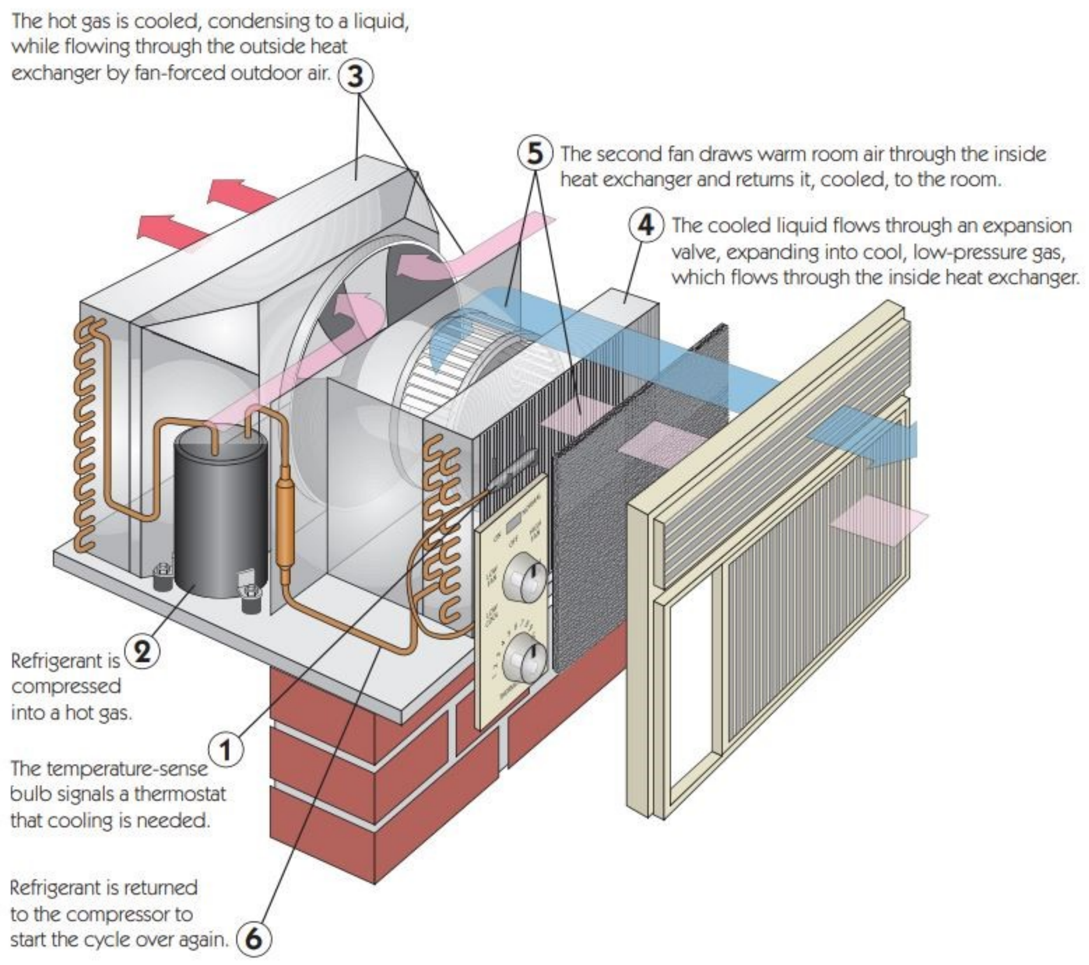
The hot gas is cooled, condensing to a liquid, while flowing through the outside heat exchanger by fan-forced outdoor air. 3
5 The second fan draws warm room air through the inside heat exchanger and returns it, cooled, to the room.
4 The cooled liquid flows through an expansion valve, expanding into cool, low-pressure gas, which flows through the inside heat exchanger.
Refrigerant is 2 compressed into a hot gas.
1 The temperature-sense bulb signals a thermostat that cooling is needed.
Refrigerant is returned to the compressor to start the cycle over again. 6

Figure 2 ***How an Air Conditioner Works*** [43]

In addition to the evaporator coil, condenser coil, and compressor, the air conditioner has several other critical mechanical components responsible for its successful operation.[42, 44] The air conditioner will have fans on the outdoor and indoor units to help disperse the heat and cool air. It will have a thermostat to control the system operation, i.e., how much to cool or not to cool. The air conditioner will have some sort of shell around the air filters. There will be a copper tube that allows the refrigerant to flow throughout the device. The air conditioner will have an expansion valve that is responsible for regulating the amount of refrigerant going into the evaporator coil. There will also be a pipe that collects water condensation from the evaporator coil. The air conditioner will also have an electronic system where the device can be controlled from a remote.

Any components not working correctly can result in a higher power expenditure for the air conditioner or can potentially cause the air conditioner not to work at all. The compressor is the air conditioner component that uses the most energy, although any component in the cooling chain not working sufficiently can create a chain effect on the entire system.[4, 42, 43] For example, an insufficiently operating fan on the condenser coil will lead to poorer dispersion of heat, which can cause the compressor to work harder or potentially break in the long run. This issue of a fan not working correctly can be caused by numerous other failures throughout the machine, and may not be caused by direct damage to the fan, for example, clogged air filters. For this reason, it is important to view the air conditioner cooling system as a chain, where if one link breaks, there is a ripple effect in energy expenditure on the entire process. According to Mitsubishi Electronics [45] a few other common issues that can cause poor cooling are:

- Refrigerant leakages or low refrigerant that can lead to temperature fluctuations or insufficient cooling.
- Clogged up or dirty air filters that can significantly reduce airflow.
- A thermostat that is incorrectly measuring temperature.
- Lose or damaged fans that will insufficiently disperse cold air as well as heat.
- Dirty coils can cause the air conditioner to work much harder.

It is also worth noting that many of these issues are progressive and may not be apparent to the user until they are manifested into much bigger issues, such as noise or inadequate cooling. In the short term, soft faults like condenser fouling, high refrigerant charges, and faulty sensors will often cause higher energy consumption.[4] In the next section, we will explain how and what data was collected from the air conditioner.

3.2 Data Collection

3.2.1 Methodology

The time series data collected from the air conditioner was voltage and current. They were collected by a device that positioned between the electrical outlet and plug. The data was then streamed from this device to an S3 AWS database where it could be accessed. Voltage and current data were collected from the air conditioner at 10,000 Hz, meaning that 10,000 data points are recorded per second. Two hundred hours of data were collected and stored in 24,000 binary files containing every thirty seconds interval. Each binary file had 300,000 data points for voltage and 300,000 data points for current. The data was collected over several months and under different conditions. Feature engineering was performed by converting the current and voltage to real power, reactive power, and total harmonic distortion, which reduce each thirty second interval to three features that contained 1500 data points each rather than the original 600,000.

We only had access to one air conditioner, so we needed to simulate a fleet of ten air conditioners running at the same time. This was done using the previously collected faulty and not faulty air conditioner data. To generate a thirty second instance of a fleet of ten air conditioners, we would randomly sample ten thirty second time intervals from the dataset where the air conditioner was running under the same operating conditions, such as the temperature setting and fan speed. This creates a single sample in our fleet dataset, where each thirty second interval would correspond to one air conditioner in a fleet. This process of fleet sampling was repeated 10,000 times, and then the metrics were computed. Our simulation does not consider the correlation between one machine breaking on the other working machines. In other words, we simulated a situation where the machines operate independently.

Ideally, the data collected should be from ten simultaneously running air conditioners so the effects of one machine not working correctly on the other working machines can be considered. Additionally, having long run data from ten simultaneously running air conditioners would allow us to use prior predictions to improve future ones. A weighted moving average could do this on the past p-values of how faulty an air conditioner is or by a voting system on past fault predictions. Unfortunately, we only had access to data from one air conditioner due to time and cost constraints. When the data was collected day to day, the weather was different, and the baseline room temperature was different as well. On one hand, this was not ideal since we desire as similar conditions as possible within our fleet of air conditioners. On the other hand, this is desirable since, in a real world setting, each room may have variations in baseline temperature due to external factors such as size, shape, and number of people inside the room. During the simulation, we also assume that machines that are of the exact same build, i.e., the same manufacturer and model, should have indistinguishable performance with respect to voltage and current.

3.2.2 Voltage

Voltage is the pressure of electricity from the power source onto the electrical circuit.[46] Voltage from wall sockets varies depending on the country you are in, from 110-240 volts. The air conditioner was plugged into a standard UK wall plug, so the voltage was 230v during our data collection. The type of voltage supplied by our electrical wall sockets is an alternating voltage; this means that the voltage alternates between flowing forward and backward.[47] This is why our volts appear in a sinusoidal waveform, as seen in figure 3. Voltage is not a useful feature on its own since it is not affected by the state of the appliance but rather by the electricity coming through the wall plug. However, voltage data was captured because the voltage sinusoidal time series is necessary for creating other meaningful features combined with the current time series.

Voltage is measured in Volts. According to the International Bureau of Weights and Measures, a volt is defined as "the difference of electric potential between two points of a conducting wire carrying a constant current of 1 ampere, when the power dissipated between these points is equal to 1 watt."[48]

3.2.3 Current

Current is the flow of electrons in an electrical circuit.[46] To use electricity, we need electrons to flow in the same direction across a circuit so that they can do things like power a light bulb or air conditioner. Current is the flow of electrons in a circuit, while voltage is the pushing force of electrons on the circuit. Unlike voltage, the current is affected by the device, although current alone is not sufficient in detecting anomalous events. In the next section, we will explore features created from current and voltage data that can be used to detect anomalous events successfully.

Current is measured in amps. According to the International Bureau of Weights and Measures, an ampere is defined as the "constant current which, if maintained in two straight parallel conductors of infinite length, of negligible circular cross section, and placed 1 metre apart in vacuum, would produce between these conductors a force equal to $2 * 10^{-7}$ MKS unit of force [newton] per metre of length."[48]

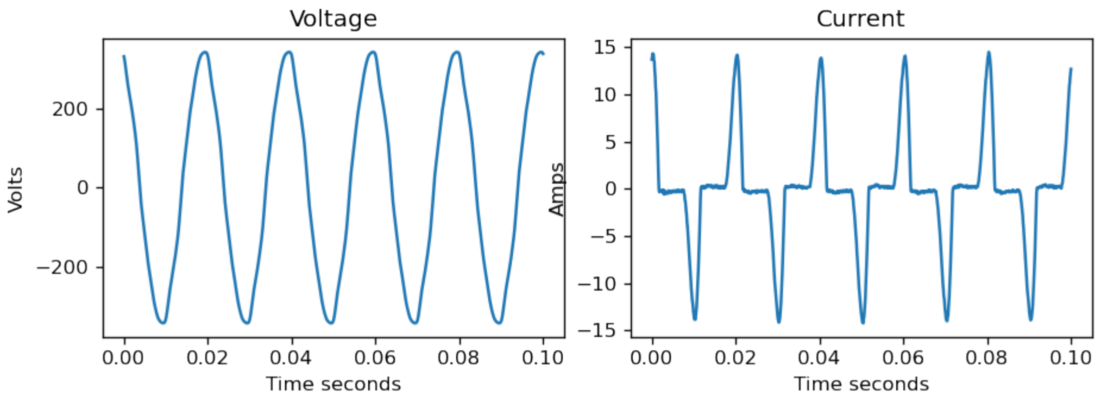

Figure 3 ***Sample Raw Current and Voltage***

3.3 Data Preprocessing

We will make use of a combination of three features in detecting faulty conditions. Those three features are real power, reactive power, and total harmonic distortion. These three features are typical diagnostic metrics electrical engineers use in system monitoring. We collected data at 10,000Hz, and the fundamental frequency of the air conditioner is 50Hz. Therefore, for every 200 data points, our device will go through one sinusoidal period. Thus, for each feature, one data point is created over one period of the fundamental frequency. In other words, all three features will be 1500 in length since our Voltage and Current are 300,000 in length. We used the simple min-max 0-1 linear scaling within the batch of ten air conditioners for pre-processing. This was done so all features are on the same scale and hold the same weight for the euclidean distancing measurements. When we run our models, we will test each of these features on their own, and additionally, we will test all three features combined into one large feature. For the large feature, each thirty second period of an air conditioner running has a vector of length 4500, and for the others, it will be 1500. We explain what real power, reactive power, and total harmonic distortion are in the following sections. We will also explain how they are calculated and why they are important.

3.3.1 Power

In section 3.1, it was mentioned that an air conditioner draws alternating current from an electrical grid, so the voltage alternates between flowing forward and backward. If the air conditioner is a perfect electrical load, the current will be sinusoidal and in phase with the voltage. A perfect electrical load is rarely the case because of non-linear and reactive loads such as motors and rectifiers, which cause the current to become non-sinusoidal and not in phase with the voltage.[49, 50] This means that the alternating current will not be an ideal sinewave and in phase with the alternating voltage.[49, 50] The none ideal nature of electrical loads means that more current must be drawn from the electrical grid to do the same amount of work.[49, 50]

The phase displacement can be represented by the power triangle as seen in figure 4.[49] The three powers in the power triangle are real power, reactive power, and apparent power. Real power is the actual amount of power being consumed in order to produce the cooling effect. Reactive power is a measure of circulating energy in the electrical grid that is an undesirable burden. Apparent power is the measure of total current drawn by the air conditioner that the grid needs to be able to withstand.

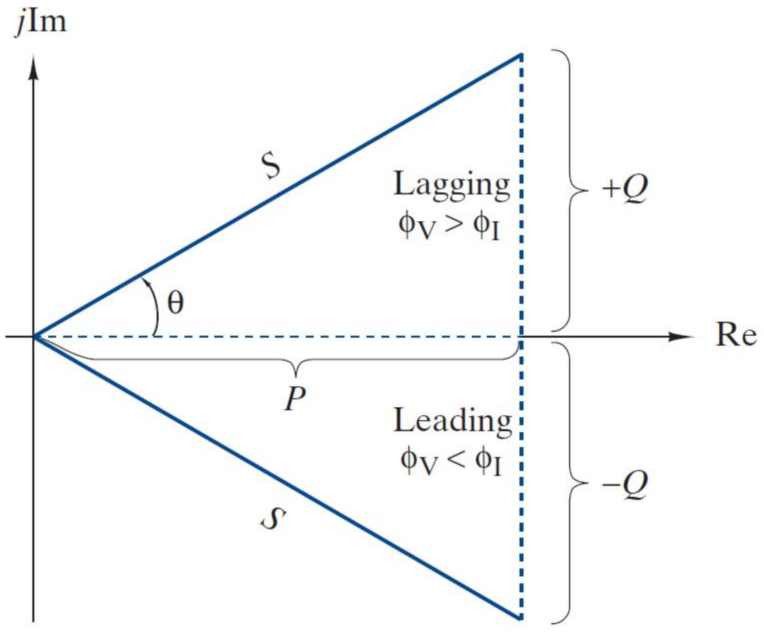

Figure 4 ***Power Triangle*** [49]

- Real power, P: watt (W).
- Reactive power, Q: volt-ampere reactive (var).
- Apparent power, $|S|$: the magnitude of complex power S: volt-ampere (VA).
- Phase of voltage relative to current, θ: the angle difference between current and voltage.

Real power and reactive power are two valuable features in detecting air conditioner health.[49, 50] We can use real power in identifying abnormal intermittency of power (e.g., incorrect powering on and off) and abnormally high or low power consumption, which helps identify component issues like incorrectly measuring thermometers. Reactive power can typically help us identify abnormalities with motors and electronics in the device because an abnormally amount of reactive power is indicative that the machine is not running efficiently. Apparent power is not helpful in detecting system faults.

There are several ways real power, reactive power, and Apparent power can be computed.[49] We computed real power by the $E(\text{Voltage} * \text{Current})$ at each fundamental frequency, where $E(\cdot)$ is the mean. Apparent power was computed by the $\sqrt{E(\text{Voltage}^2)} * \sqrt{E(\text{Current}^2)}$ at each fundamental frequency. Reactive power was computed by taking the $\sqrt{(\text{Apparent Power})^2 - (\text{Real Power})^2}$. Below we display real power and reactive power for healthy and faulty machines. It is worth noting that the in-group variability is due to both randomness and external conditions, such as the day the data was recorded and measurement errors from the device.

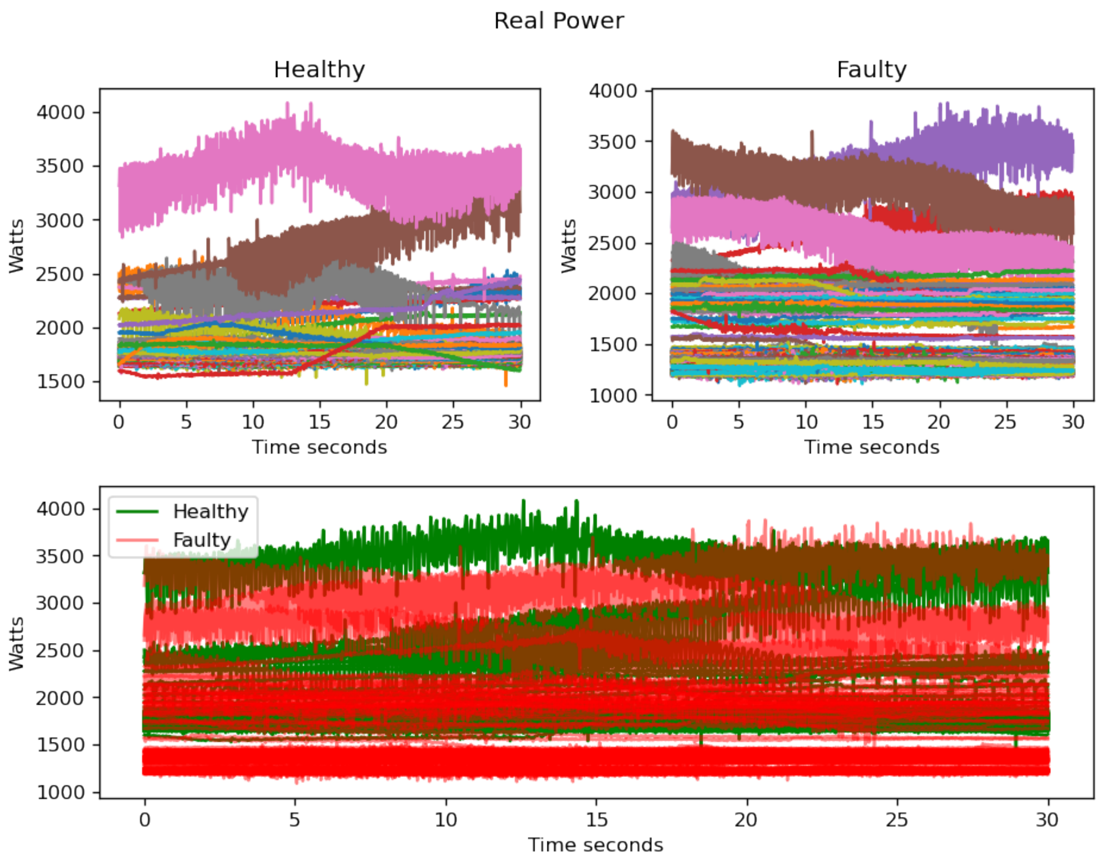

Figure 5 ***Real Power***: In the top two plots, we have thirty second intervals of real power for healthy and faulty air conditioners. Each color represents another thirty second interval. In the bottom plot, we have the top two plots overlapped to contrast similarity and difference between healthy and faulty.

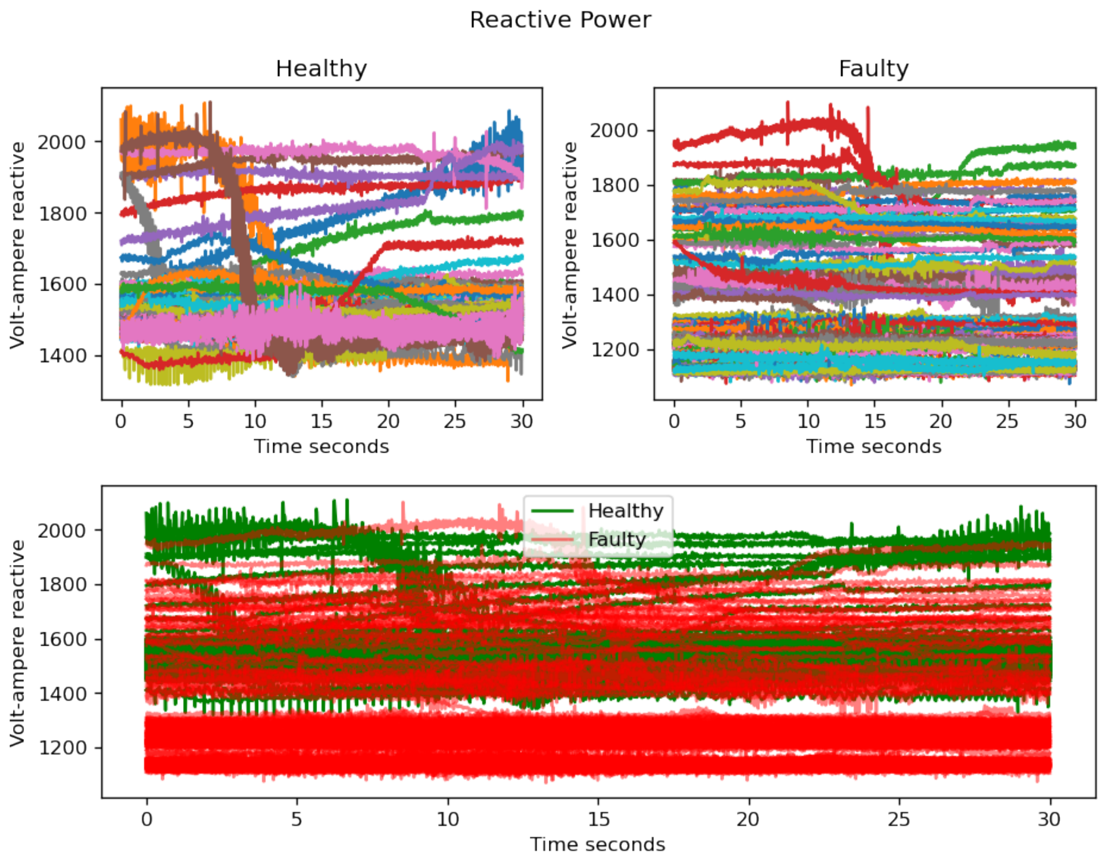

Figure 6 ***Reactive Power***: In the top two plots, we have thirty second intervals of reactive power for healthy and faulty air conditioners. Each color represents another thirty second interval. In the bottom plot, we have the top two plots overlapped to contrast similarity and difference between healthy and faulty.

3.3.2 Total Harmonic Distortion

The total harmonic distortion is a measurement of the distortion present in the electrical current due to harmonics.[50] Total harmonic distortion is defined as the ratio of the sum of the powers of all harmonic components to the power of the fundamental frequency.[50] The components I_i are the coefficients from the fast Fourier transform.[51] The first coefficient is the fundamental frequency, and the rest are the harmonic components.[50] To put it formally, total harmonic distortion is $\sqrt{I_2^2 + I_3^2 + I_4^2 + ...}/I_1$.[50] The total harmonic distortion was computed for each fundamental frequency, which resulted in a vector of 1500 points. Total harmonic distortion is helpful when trying to detect issues in the electric processing train, such as the compressor motor not working correctly.

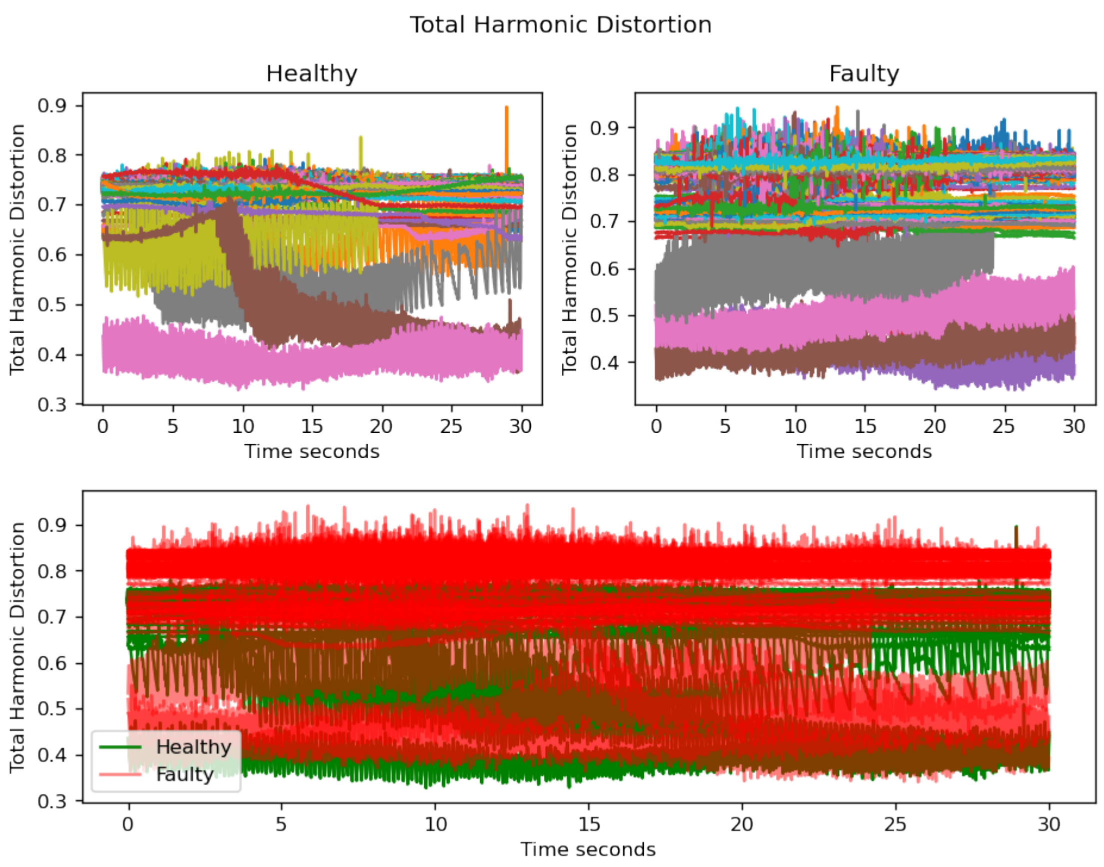

Figure 7 ***Total Harmonic Distortion***: In the top two plots, we have thirty second intervals of total harmonic distortion for healthy and faulty air conditioners. Each color represents another thirty second interval. In the bottom plot, we have the top two plots overlapped to contrast similarity and difference between healthy and faulty.

3.4 Evaluation Criteria

We will consider AUC (the area under the receiver operating characteristic curve) as our evaluation metric when quantifying how well the models can distinguish anomalous events from normal events. It is not enough for us to consider precision since anomalous events are rare. Additionally, the consequences of a false positive and true positive do not hold equal weight. In our case, the consequences are direr if an anomalous event is labeled normal than if a normal event is labeled anomalous. How much importance goes to one or the other depends on the specific user. Furthermore, we are interested in creating probability values for how faulty a machine is. In other words, we are also interested in determining if the machine is very faulty or just a little faulty. How we create probability values from our unsupervised learning algorithms depends on the algorithm and is explained in detail in section 4. Generally, when using a clustering algorithm, the faulty score is the number of machines not assigned to the cluster, and for algorithms that compute a faulty threshold value, linear min-max 0-1 scaling is applied. Therefore, the evaluation criteria we will use is AUC.[52]

3.4.1 AUC

AUC is computed by taking the area under the ROC (receiver operating characteristic) curve. The ROC curve is created by plotting the true positives (TPR), also known as recall, against the false positives (FPR) at all possible thresholds. The TPR is the proportion of positive samples that are known to be positive and are correctly labeled positive; the FPR is the proportion of negative samples that are known to be negative and are incorrectly labeled positive. A perfect AUC score is one, and the worst is a zero. The higher the score is, the better the score. One way to interpret AUC is that it will give the best score for the correct ordering of labeled probability values for the anomalous events. This allows the user to pick a threshold later that meets their tolerance for TPR and FPR.

$$\text{FPR} = \frac{\text{False positive}}{\text{False positive} + \text{True negative}}$$

$$\text{TPR} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}}$$

4 Anomaly Detection Algorithms

4.1 General Approach

Our general approach for detecting faulty air conditioners was inspired by the paper *A general anomaly detection framework for fleet based condition monitoring of machines.*[16] Our general approach has five phases:

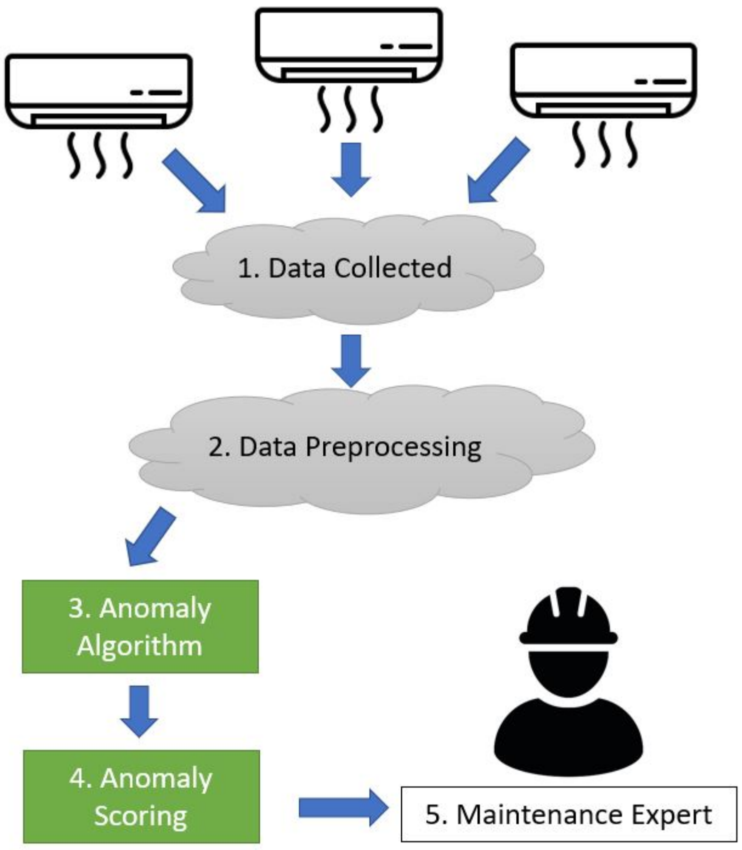

Figure 8 ***Fault detection workflow****: visualize*

1. **Data Collection:** We collect thirty second intervals of data from the fleet of air conditioners running in parallel.
2. **Data Preprocessing:** We apply preprocessing from section 3.3
3. **Anomaly Algorithm:** We apply a anomaly detection algorithm to the fleet.
4. **Anomaly Scoring:** We assign an anomaly score to each air conditioner.
5. **Maintenance Expert:** The anomaly scores are reported to a maintenance expert.

The general approach is the same for all the algorithms applied, only the (3) anomaly detection and (4) anomaly scoring phases change. The Euclidean distance metric was used for all the algorithms that required a distance metric. The following sections will go over the unsupervised algorithms we used for anomaly detection and scoring methods within the fleet of ten air conditioners.

4.2 Agglomerative Hierarchical Clustering

Agglomerative Hierarchical Clustering (AHC) is a clustering algorithm that groups similar objects into clusters from the bottom up.[16] Hierarchical clustering can be done from the top down as well and is called divisive hierarchical clustering. However, the agglomerative approach is more common.

We first use AHC to cluster the fleet into groups, and then we compute the anomaly scores by the number of samples not assigned in that cluster.[16] In AHC, each observation starts in its own subcluster, then pairs of most similar subclusters are iteratively merged until stopping criteria is met. Several functions can be used for linking clusters:[53–55]

- Ward linkage: minimizes the variance of the clusters being merged.
- Average linkage: uses the average of the distances of each observation of the two sets.
- Complete linkage: uses the maximum distances between all observations of the two sets.
- Single linkage: uses the minimum of the distances between all observations of the two sets.

We used average linkage as our linkage method. Additionally, we use the distance threshold D for the stopping criteria of merging clusters. When an object is D distance farther for a cluster, it will not be merged. D can take on any value greater than 0, but a larger D value will make it more likely for samples to be grouped into a smaller number of clusters, while a smaller D will do the opposite. In other words, a D too large will always group all samples into one cluster, and a value too small will not form any clusters. This parameter is crucial in effectively clustering samples. In practice, as the air conditioner ages, the best D value may change. It is also highly dependant on the features used, see figure 11. Furthermore, the underlying cluster structure is not taken into account when using this threshold. The cophenetic correlation threshold would be a better alternative to the D distance threshold since it is independent of distance and instead takes into account the structure of the data.[16] We still used the D distance threshold since a package employing the cophenetic correlation threshold was not available, and we did not have the time and resources to build one from scratch.

The cophenetic correlation threshold which measures how well a clustering partition preserves the original pairwise distances.[16, 56] This parameter can be tuned depending on tolerance for type 1 and type 2 errors.[16] The cophenetic correlation threshold it is formally defined as:

$$\frac{\sum_{(X,Y)\in P}(s(X,Y)-\bar{s})(t(X,Y),\bar{t})}{\sqrt{[\sum_{(X,Y)\in P}(s(X,Y)-\bar{s})^2][\sum_{(X,Y)\in P}(t(X,Y)-\bar{t})^2]}}$$

where P is the set of all sample pairs (X,Y), $s(x,y)$ is the linkage method used, and $t(X,Y)$ is the dendrogrammic distance, which is the height at which samples X and Y are first joined in the dendrogram.

Results from hierarchical clustering can be represented in a tree-like diagram called the dendrogram.[57] In the diagram, the lowest items connected are most similar, where the highest items connected are the least similar. For example, in figure 9 we can see two instances of clustering. In the left figure, we can see how one faulty air conditioner was not grouped into the large cluster of healthy air conditioners, and in the right figure, we can see how all the healthy air conditioners were grouped into one cluster. In the left image, the healthy air conditioners would have an anomaly score of 0.1, and the faulty air conditioner would have an anomaly score of 0.9. In the right image, all the air conditioners would have an anomaly score of 0.

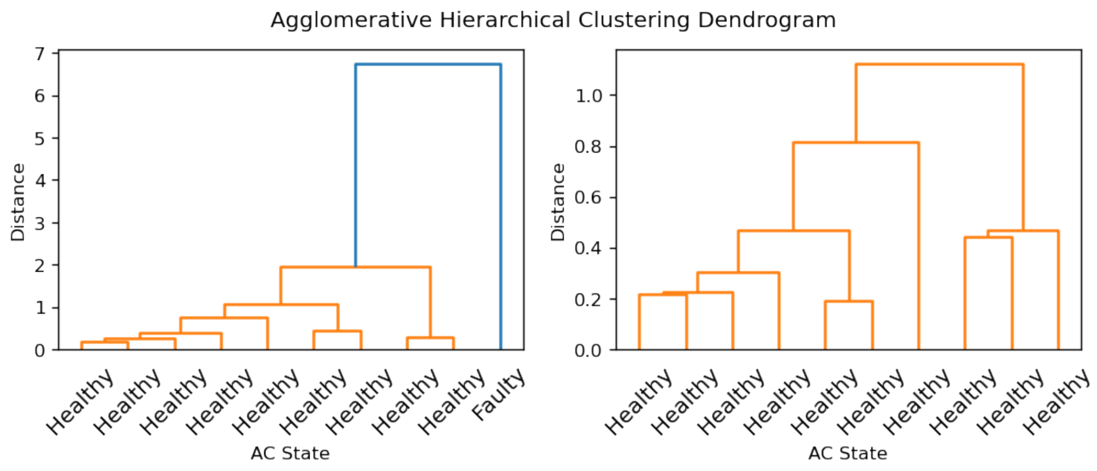

Figure 9 ***Agglomerative Hierarchical Clustering Dendrogram***

4.3 K Nearest Neighbor

K Nearest Neighbor (KNN) is a non-parametric classification algorithm initially used for classification and regression tasks.[58] The algorithm classifies data points by voting on the K nearest neighbors to the data point, and for regression tasks, it will take the average of the k nearest neighbors.[58] With slight modifications to the original method, we can use KNN for unsupervised anomaly detection.[59]

We compute the mean distance of the K nearest neighbors to our desired data point to compute an anomaly score. Therefore, the less anomalous a data point is, the lower the mean of distances, and the more anomalous the data point is, the higher the mean distance should be. A linear scaling on these values is done to generate probability values. The main advantage of KNN is it is a very simple and robust algorithm with few hyperparameters.

4.4 Local Outlier Factor

The local outlier factor (LOF) is an anomaly detection algorithm that works by comparing the local density of a data point with that points K nearest neighbors.[60] By doing this, the algorithm identifies regions that have higher densities and lower densities, where regions with lower densities are classified as outliers. The advantage of considering this density based approach is that data points that are a small distance away from a dense cluster would be classified as outliers, along with data points that may be farther away from sparse clusters will not be classified as outliers.

To understand LOF formally we must first define several functions.[61] Have k-distance(A) be the distance of a point to the k-th nearest neighbor. Then we will define the reachability distance between points A and B by:

$$\text{Reachability-distance}_k(A, B) := \max(\text{k-distance}(B), \text{distance}(A, B))$$

Let $N_k(x)$ be the set of points within the kth nearest neighbors. We define the local reachability density of a points by:

$$\text{lrd}_k(A) = 1/(\frac{\sum_{B \in N_k(A)} \text{Reachability-distance}_k(A, B)}{|N_k(x)|})$$

Finally, the local reachability densities are compared with their neighbor densities using the following local outlier factor $\text{LOF}_k(x)$ formula:

$$\text{LOF}_k(x) := \frac{\sum_{p \in N_k(x)} \text{lrd}_k(x)}{\text{lrd}_k(x) * |N_k(x)|}$$

A LOF value lower than one means the density of the point is higher than its neighbors, and subsequently, the point is not an outlier, while values greater than one indicate that the point is an outlier.

4.5 One Class Support Vector Machines

One Class Support Vector Machines (OCSVM) identify data points as a specific class among other objects using a Support Vector Machines (SVM).[35, 62, 63] Typically, OCSVM have been used for semi-supervised learning by training the algorithm to non-anomalous data, so when a new data point is given, the algorithm can identify the point as anomalous or not. However, this would require known non-anomalous data, which does not meet our problem criteria. In our problem, we know that most air conditioners will be healthy at all times, so we use the unsupervised variant of OCSVM, which does not require any labeled data.[36] We train the OCSVM classification using a gaussian kernel on our entire fleet of air conditioners and score how anomalous an event is by its proximity to the decision boundary. We expect anomalous events to score more anomalous because they would be farther away and less dense than the healthy points.

More formally, OCSVM works by identifying the smallest hypersphere using the data points.[64] The problem is defined by the following constrained optimization formula, where r is the radius, c is the center, $\Phi(.)$ is a nonlinear transformation, and $\kappa(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$ is the kernel.

$$\min_{r,c} \ r^2$$
$$\text{subject to } \|\Phi(x_i) - c\|^2 \leq r^2 \text{ for all } i = 1, 2, ..., n$$

This formula is very sensitive to outliers and too restrictive, so in practice an alternative formula is used that is less sensitive to outliers. ν is a positive tuning parameter on the range (0, 1] that specifies the trade-off between the sphere volume and the number of outliers, and ζ_i are non-negative slack variables.

$$\min_{r,c,\zeta} \ r^2 + \frac{1}{\nu n} \textstyle\sum_{i=1}^{n} \zeta_i$$
$$\text{subject to } \|\Phi(x_i) - c\|^2 \leq r^2 + \zeta_i \text{ for all } i = 1, 2, ..., n$$

By Karush-Kuhn-Tucker (KKT) the optimality conditions are,

$$c = \textstyle\sum_{i=1}^{n} \alpha_i \Phi(x_i)$$

Where α_i is equal to,

$$\max_{\alpha} \textstyle\sum_{i=1}^{n} \alpha_i \kappa(x_i, x_j) - \sum_{i,j=1}^{n} \alpha_i \alpha_j \kappa(x_i, x_j)$$
$$\text{subject to} \textstyle\sum_{i=1}^{n} \alpha_i = 1 \text{ and } 0 \leq \alpha_i \leq \dfrac{1}{\nu n} \text{ for all } i = 1, 2, ..., n$$

4.6 Isolation Forest

Isolation Forest (iForest) is an anomaly detection algorithm that identifies anomalies using isolations.[65, 66] iForests will randomly generate partitions on the data by selecting an attribute and then randomly selecting a split value for that attribute. The algorithm works because anomalous instances tend to be easier to separate compared to normal instances. An example of this split can be seen in figure 10. The algorithm runs in linear time, which is advantageous for datasets with a large number of invariants. Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou also note that the algorithm works particularly well with high dimensional data with irrelevant attributes and can be trained with or without anomalies in the training set.[66]

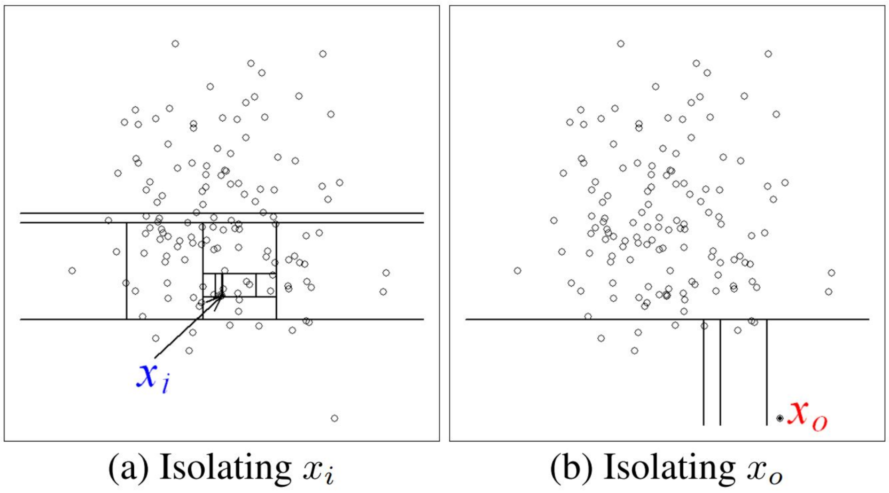

Figure 10 ***iForest Isolation***: *Anomalies are more susceptible to isolation and hence have short path lengths. (a) a normal point x_i requires twelve random partitions to be isolated; (b) an anomaly x_o requires only four partitions to be isolated.* [65]

To understand how iForets works, we must first define the isolation tree (iTree).

> **Isolation Tree:** Let T be a node of an isolation tree. T is either an external-node with no child, or an internal-node with one test and exactly two daughter nodes (T_l, T_r). A test consists of an attribute q and a split value p such that the test $q < p$ divides data points into T_l and T_r.[65]

iForest anomaly detection is a two stage procces. The first is training stage and the second is the evaluation stage. The training stage workflow is [67]:

1. Select ψ sample points randomly as the subsample set.
2. Randomly select a dimension and generate a cutting point p between the minimum value and the maximum value of the dimension.
3. Place the data with the specified dimension smaller than p on the left side of the current node, and the data with the dimension greater than or equal to p on the right side of the current node.
4. Repeat steps (2) and (3) in the subnodes to continuously construct new subnodes, until the data are no longer separable or has reached the maximum depth $\log_2(\psi)$

During the evaluation stage, the anomaly score is calculated by the number of edges a point traverses an iTree from the root to the leaf. The iForest anomaly score of a points x is defined as [65]:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Where for path length $h(x)$ the expected path length is $E(h(x))$, and $c(n)$ is defined as,

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}$$

Where $H(i)$ is the harmonic number and can be estimated by ln(i) + 0.5772156649 (Euler's constant).

4.7 Isolation Nearest Neighbor Ensembles

Isolation Nearest Neighbor Ensembles (iNNE) is an anomaly detection algorithm that identifies anomalies using isolations and nearest neighbor ensembling.[68] Similarly to KNN, iNNE relies on the idea that anomalies are usually farther away from their nearest neighbors. iNNE works by creating random subsamples from the data and then building a hypersphere around each data point to create isolations. The hypersphere is centered around the point that is being isolated, and the radius is determined by the distance between the point and its nearest neighbor.

To understand the algorithm formally we must first define a hypersphere, isolation score, and anomaly score. Let $S \subset D$ be a subsample of size ψ selected randomly without replacement from D, and let η_x denote the nearest neighbor of x. A hypersphere $B(c)$ centered at c with radius $\tau(c) = \|c - \eta_c\|$ is defined to be $\{x : \|x - c\| < \tau(c)\}$, where $x \in \Re^d$ and $c, \eta_c \in S$. The isolation score $I(x)$ for a data point $x \in \Re^d$ based on S is as follows:

$$I(x) = \begin{cases} 1 - \dfrac{\tau(\eta_{cnn(x)})}{\tau(cnn(x))}, & \text{if } x \in \bigcup_{c \in S} B(x) \\ 1 & \text{, otherwise} \end{cases}$$

where $cnn(x) = \min_{c \in S}\{\tau(c) : x \in B(c)\}$. The isolation score takes on a range $[0, 1]$. A score of 1 is given when a point is very far away and 0 when it is very close.

iNNE has t sets of hyperspheres generated for t subsamples S_i. The anomaly score for $x \in \Re^d$ is defined as follows:

$$\bar{I}(x) = \frac{1}{t} \sum_{i=1}^{t} I_i(x)$$

where $I_i(x)$ is the isolation score based on S_i. Like the iForests, iNNE has a training and evaluation stage. In the training stage iNNE t sets of hyperspheres are built from t randomly selected subsamples of size ψ. In the evaluation stage each test instance is evaluated against t sets of hyperspheres in iNNE, and the isolation scores are averaged to produce the anomaly score.

5 Results and Discussion

For AHC, KNN, LOF, OCSVM, iForest, the well established python sklearn libraries were used.[69] iNNE is a newer, less established algorithm and is not available within the sklearn ecosystem. The python iNNE implementation used was written by postdoctoral researcher Vincent Vercruyssen.[70] The calculations were computed using the processor Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz (12 CPUs), 2.2GHz, and with 32 GB of ram. It took several days for all of the calculations to finish computing.

When testing the models, we used four variable instances; real power, reactive power, total harmonic distortion, and a combination of all the other three variables "All Variables". The AUC scores were computed on 10,000 thirty second fleet instances of ten air conditioners for each of the four variables. In total, there were 100,000 anomaly scores computed for each model, variable, parameter pair. For AHC, the parameter D was computed from the range of 0.5 to 10. To save computational resources, the highest D value we tested was 10. This cut-off was chosen because, in test runs, performance would drop drastically after D became greater than 10. For KNN and LOF, the tuning parameter k was computed from the range of 2 to 8 or 20% of the fleet to 80% of the fleet. The range from 2 to 8 was the natural choice for k since 1 and 9 would hold consistently inferior results in test runs. For OCSVM, the parameter ν was computed from 0.2 to 0.9. For iForest and iNNE, the parameter ψ was computed from 3 to 10 since 2 and 10 are the minimum and maximum values. We opted to omit 2 and start at 3 since, in test runs, the performance was comparatively bad at 2, and our computational resources were limited.

Variables	**Models AUC**					
	AHC	**KNN**	**LOF**	**OCSVM**	**iForest**	**iNNE**
All Variables	0.986	0.958	**0.987**	0.973	0.976	0.949
Real Power	**0.987**	0.964	0.984	0.962	0.971	0.937
Reactive Power	0.979	0.969	**0.981**	0.976	0.973	0.918
Total Harmonic Distortion	0.98	0.921	**0.986**	0.955	0.973	0.940

Table 1: The best AUC scores for the six algorithms. The best performers are given in boldface.

Variables	Best parameter					
	AHC	KNN	LOF	OCSVM	iForest	iNNE
	D	*k*	*k*	ν	ψ	ψ
All Variables	5.0	4 (40%)	5 (50%)	0.90	80 (80%)	10 (100%)
Real Power	2.5	3 (30%)	5 (50%)	0.90	9 (90%)	10 (100%)
Reactive Power	3.0	4 (40%)	5 (50%)	0.90	8 (80%)	10 (100%)
Total Harmonic Distortion	1.5	4 (40%)	5 (50%)	0.90	8 (80%)	10 (100%)

Table 2: The best performing parameter values for the six algorithms.

Variables	Run time (seconds)					
	AHC	**KNN**	**LOF**	**OCSVM**	**iForest**	**iNNE**
All Variables	0.51	0.58	0.59	0.52	0.95	0.80
Single Variables	0.40	0.44	0.44	0.40	0.74	0.62

Table 3: Average execution time for the best parameters in six algorithms and variable combinations. The runtime is measured in seconds, and the results are averaged over the 10,000 runs.

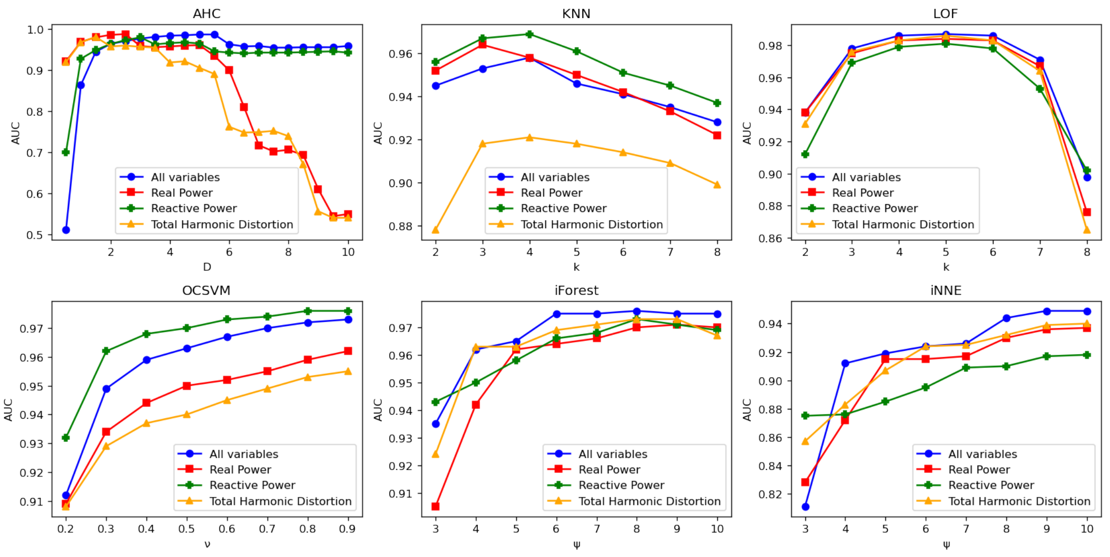

Figure 11 AUC scores by variable and parameters: This plot contains the AUC scores for the six algorithms corresponding to each hyperparameter value.

In table 1 we have the best AUC scores for each model and variable combination. We can see that the majority of the algorithms performed similarly. Generally, LOF performed the best among all the variables and algorithms tested. Although, AHC performed slightly better with real power and similarly with all the other variables. OCSVM and iForests performed well overall but did not score the best for any variable. KNN performed poorly compared to the other algorithms, especially using total harmonic distortion. Overall, iNNE performed the worst of the six algorithms but still outperformed KNN using total harmonic distortion. All the algorithms had high AUC scores, and arguably all of them were successful at detecting anomalies. Just looking at the best performance is not sufficient in deciding the best approach.

In table 2 we can see the parameters corresponding to the best AUC scores for each given algorithm and variable pair. In figure 11 we can see the AUC scores show the parameter sensitivity of the algorithms on the four variables tested. AHC had a good performance, but we can see that the D parameter is rather sensitive. It varies greatly depending on the variable used, and a slight shift can significantly impact the AUC. For this algorithm to be effective, it requires careful tuning of the D parameter, and this may not generalize well in production. Potentially a solution to this could be to use the cophenetic correlation threshold as mentioned in section 4.2. KNN had consistent results where a k value between 3 (30%) to 5 (50%) held good scores. This is expected since the majority of the units in each fleet are healthy. LOF also had consistent results in which a k value between 3 (30%) and 6 (60%) held good results. For LOF, all the optimal k values where 5 (50%), see table 2. Both KNN and LOF were not very sensitive when selecting a k value. Results would be good as long as the value of k was large enough so that enough points are considered and not too large so anomalous events are included. OCSVM favored higher values for ν and was very impactful in the performance of the algorithm. ν was not too sensitive, and any value above the default 0.5 value had good results. For iForest and iNNE, larger values for ψ performed better. For both models, the parameter was not very sensitive. Since the number of points in your fleets is small, even using the default max value for ψ would work comparably well. The most important takeaway from figure 11 and our analysis on the hyperparameters is that there is more variability on performance when considering the best parameters than when considering the best algorithm. In other words, selecting a good hyperparameter is more important than the algorithm itself, and choosing the best hyperparameters can be tricky.

In table 3 we can see the average run for a fleet of air conditioners for each algorithm. Not surprisingly, real power, reactive power, and total harmonic distortion had the same speed since each is an array of 1500 data points, and when all variables were used simultaneously, the algorithms ran slower since this is an array of 4500 data points. AHC and OCSVM were the fastest, and KNN and LOF were the second fastest. In our application, KNN and LOF have a polynomial time complexity since, for every point, all the other points must be compared. Despite this, their performance was faster due to the small number of machines in each fleet. As the size of the fleet increases, we would expect a performance drop on these algorithms. iNNE was the second slowest, and iForest was the slowest. iForest is known to be a fast algorithm since it has linear time complexity.[65, 66] The fleets had only ten data points of 1500 or 4500 rows. The number of rows was not large enough for the algorithm to take advantage of its linear time complexity. When the number of machines in a fleet gets large, we expect iForests to have a faster performance from the other algorithms such as KNN and LOF.

There is no one-size-fits-all for choosing algorithms. With good hyperparameter choices, all the algorithms had satisfactory AUC scores and could detect faulty air conditioners from our dataset. However, one important takeaway is that performance was more dependant on the hyperparameter selected than the algorithm. LOF had the best results overall; it consistently had the best AUC scores, was not very sensitive to parameter tuning of k, and ran only 0.04 seconds slower than the fastest algorithms. Additionally, the parameter k is easily interpretable, and so are the results. In practice, there are drawbacks to LOF. The case we examined is specific to a fleet of only ten machines. Computational performance may drop when fleets are very large. KNN performed consistently worse than LOF. Unless new features are created, it appears the density approach of LOF is better than the distance approach used by KNN. AHC had good AUC scores but was tricky to tune. Nevertheless, the algorithm was fast and could be a good choice if computational speed is important. If this algorithm is used, we suggest trying the cophenetic correlation threshold approach rather than a distance threshold approach since it is more interpretable and would take into account the structure of the data rather than the distance. OCSVM had good AUC scores and was fast to run. The parameter

ν was relatively easy to tune and would make this algorithm a good choice if speed was a top priority. Furthermore, OCSVM AUC could potentially be improved with different kernels. Out of the isolation algorithms, overall, iForest performed better than iNNE. iForest was the slowest algorithm, but the linear time complexity should make it one of the fastest when the fleet size becomes large. LOF, AHC, OCSVM, and iForest have clear advantages, while KNN and iNNE were outclassed by their counterparts. All the algorithms had satisfactory AUC scores, and any of the unsupervised learning approaches covered can detect faulty air conditioners from our dataset. The issue of hyperparameter sensitivity should be considered and not just the best AUC score for generalizable performance in a production environment.

6 Conclusion

6.1 Overview

We covered a literature review on past approaches taken by researchers for fault and anomaly detection. We also discussed several alternative distancing metrics for euclidean distance. Next, we covered how an air conditioner works and breaks and the data collection methodology. After we covered the feature engineering used for anomaly detection and why real power, reactive power, and total harmonic distortion were the features used. Afterward, we explained our anomaly detection approach and the unsupervised learning algorithms implemented. Lastly, we discussed the results and concluded that all the algorithms were relatively good at detecting faulty conditions in air conditioners.

6.2 Future Recommendations

We have four recommendations if our fleet based condition monitored approach for air conditioners is to be used in production. In this section, we will go over each of these recommendations.

The first recommendation is to create a robust fleet based data set for testing filled with as many different faults and operating conditions as possible. As we discussed at the beginning, creating a dataset containing every possible event is very hard and not feasible, but having something as close as possible is best. Furthermore, in our experiment, we only attempted to detect anomalies with a fleet of size ten. Therefore, we recommend testing with smaller fleets of three or more and for fleets much larger than ten. The size of the fleet would have to be taken into consideration if our approach was used in production since in a production application the size of the fleet will vary from building to building.

The second recommendation is to use algorithms that do not have sensitive hyperparameters like LOF or iForests. These two algorithms performed the best and have easily interpretable hyperparameters that performed consistently across all variables. All the algorithms employed in our research had good results; however, performance variability was more dependant on hyperparameters selected and the algorithm chosen. Therefore, we recommend avoiding algorithms with sensitive hyperparameters as this would likely not generalize well with different air conditioner models. Furthermore, the dataset is likely to be incomplete, so it would be best to use algorithms that are more likely to generalize to new cases. Lastly, in section 5, we covered some potential improvements to the algorithms implemented in this research, such as using the cophenetic correlation threshold for AHC rather than a distance threshold.

The third recommendation is to apply past data to the current anomaly detection score rather than just looking at thirty second intervals. One option is a voting system or moving average on past predictions to determine a faulty condition value. This has the potential to give better results since an extended time period will be considered and would be simple to implement. The time range could be over an hour or a day; further research would be required to optimize this. Another option is to use all past data points over a period where operating conditions go unchanged rather than just the current thirty second interval. Using all past data would increase the number of data points available to the anomaly detection algorithm and potentially improve performance even further.

The fourth recommendation is to try different distance metrics covered in section 2.6. Some of these distancing metrics come prebuilt in their libraries and would be easily testable, and some others would require a bit of programming. In particular, it would be interesting to see dynamic time warping implemented since it has had many successful applications in time series data.

7 Acknowledgments

Code for calculations, simulations, models and plots available on github at https://github.com/PhilipPhil/Unsupervised-Learning-for-Anomaly-Detection-in-Fleet-Based-Systems-Application-to-Air-Conditioners

The tools folder contains all the classes and methods programmed for this project. Calculations.py contains the class Calculations, which includes methods for computing real power, reactive power, apparent power, and total harmonic distortion from the binary files. Simulate.py contains the class Simulate which includes methods for simulating the fleet data-sets. Cluster.py is an abstract class that includes methods used by all of the six algorithms we used. The remaining six files contain classes for the algorithms used in this paper, each of these implements the abstract class Cluster.

The jupyter notebook files have examples of the code written in tools. ”AUC Parameter Plots.ipynb“ has the plot for AUC scores by variable and parameters. Calculations.ipynb contains plots for current and voltage and shows how the Calculations class is used. Data.ipynb shows how the Simulate class is used and has plots for real power, reactive power, and total harmonic distortion. Finally, ExampleScores.ipynb shows how our anomaly detection methods created implemented.

The runners folder contains a jupyter notebook for each algorithm implementation with the hyperparameter optimization plots.

References

1. Wu, S. & Sun, J.-Q. Cross-level fault detection and diagnosis of building HVAC systems. eng. *Building and environment* **46,** 1558–1566. ISSN: 0360-1323 (2011).
2. DOE, U. Buildings Energy Data Book, Energy Efficiency and Renewable Energy Department.
3. Schein, J., Bushby, S. T., Castro, N. S. & House, J. M. A rule-based fault detection method for air handling units. eng. *Energy and buildings* **38,** 1485–1492. ISSN: 0378-7788 (2006).
4. Yan Ke Chong Adrian, M. Y. Generative adversarial network for fault detection diagnosis of chillers. *Building and environment* **172,** 106698. ISSN: 0360-1323 (2020).
5. Lapisa, R., Bozonnet, E., Salagnac, P. & Abadie, M. Optimized design of low-rise commercial buildings under various climates – Energy performance and passive cooling strategies. *Building and Environment* **132,** 83–95. ISSN: 0360-1323. `https://www.sciencedirect.com/science/article/pii/S0360132318300416` (2018).
6. Jia, Y. & Reddy, T. A. Characteristic Physical Parameter Approach to Modeling Chillers Suitable for Fault Detection, Diagnosis, and Evaluation. eng. *Journal of solar energy engineering* **125,** 258–265. ISSN: 0199-6231 (2003).
7. Katipamula, S. & Brambley, M. R. Methods for Fault Detection, Diagnostics, and Prognostics for Building Systems-A Review, Part I. eng. *HVAC research* **11,** 3. ISSN: 1078-9669 (2005).

8. Siegel, D. *Prognostics and Health Assessment of a Multi-Regime System using a Residual Clustering Health Monitoring Approach* PhD thesis (2013), 210. ISBN: 978-1-303-74735-9. https://www.proquest.com/dissertations-theses/prognostics-health-assessment-multi-regime-system/docview/1508276649/se-2?accountid=14511.

9. W, J., H, E., V, K. & A., M. Inter-Engine Variation Analysis for Health Monitoring of Aerospace Gas Turbine Engines. *PHM Society European Conference* **4,** 1 (2018).

10. Fumeo, E., Oneto, L. & Anguita, D. Condition Based Maintenance in Railway Transportation Systems Based on Big Data Streaming Analysis. *Procedia Computer Science* **53.** INNS Conference on Big Data 2015 Program San Francisco, CA, USA 8-10 August 2015, 437–446. ISSN: 1877-0509. https://www.sciencedirect.com/science/article/pii/S1877050915018244 (2015).

11. Hodge, V. J., O'Keefe, S., Weeks, M. & Moulds, A. Wireless Sensor Networks for Condition Monitoring in the Railway Industry : a Survey. eng (2015).

12. Park Seyoung Kang Jaewoong, K. J. Unsupervised and non-parametric learning-based anomaly detection system using vibration sensor data. *Multimedia tools and applications* **78,** 4417–4435. ISSN: 1380-7501 (2019).

13. Munir Mohsin Siddiqui Shoaib Ahmed, D. A. DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. eng. *IEEE access* **7,** 1991–2005. ISSN: 2169-3536 (2019).

14. *2018 International Conference on Advances in Computing, Communications and Informatics : 19-22 September 2018, Bangalore, India / Institute of Electrical and Electronics Engineers.* eng. ISBN: 1-5386-5314-1 (2018).

15. Li Zhipeng Qin Zheng, H. K. *Intrusion Detection Using Convolutional Neural Networks for Representation Learning* eng. in *Neural Information Processing* **10638** (Springer International Publishing, Cham, 2017), 858–866. ISBN: 9783319701387.

16. Kilian Hendrickx Wannes Meert, Y. M. A general anomaly detection framework for fleet-based condition monitoring of machines. *Mechanical Systems and Signal Processing* **139,** 106585 (2020).

17. Hodge, V., O'Keefe, S., Weeks, M. & Moulds, A. Wireless sensor networks for condition monitoring in the railway industry: A survey. *IEEE Transactions on Intelligent Transportation Systems* **16.** cited By 264, 1088–1106 (2015).

18. Farrar, C. R. & Worden, K. *Structural Health Monitoring: A Machine Learning Perspective* 1. Aufl. eng. ISBN: 1119994330 (Wiley, New York, 2012).

19. Jia, X. *et al.* A deviation based assessment methodology for multiple machine health patterns classification and fault detection. *Mechanical Systems and Signal Processing* **99,** 244–261. ISSN: 0888-3270. `https://www.sciencedirect.com/science/article/pii/S088832701730328X` (2018).

20. Li, Z., Qin, Z., Huang, K., Yang, X. & Ye, S. *Intrusion Detection Using Convolutional Neural Networks for Representation Learning* eng. in *Neural Information Processing* **10638** (Springer International Publishing, Cham, 2017), 858–866. ISBN: 9783319701387.

21. Yang, B., Han, T. & An, J. ART–KOHONEN neural network for fault diagnosis of rotating machinery. *Mechanical Systems and Signal Processing* **18,** 645–657. ISSN: 0888-3270. `https://www.sciencedirect.com/science/article/pii/S0888327003000736` (2004).

22. Unal, M., Onat, M., Demetgul, M. & Kucuk, H. Fault diagnosis of rolling bearings using a genetic algorithm optimized neural network. *Measurement* **58,** 187–196. ISSN: 0263-2241. `https://www.sciencedirect.com/science/article/pii/S0263224114003601` (2014).

23. Schlechtingen, M. & Ferreira Santos, I. Comparative analysis of neural network and regression based condition monitoring approaches for wind turbine fault detection. *Mechanical Systems and Signal Processing* **25,** 1849–1875. ISSN: 0888-3270. `https :`

//www.sciencedirect.com/science/article/pii/S0888327010004310 (2011).

24. Zhao, R. *et al.* Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing* **115,** 213–237. ISSN: 0888-3270. https://www.sciencedirect.com/science/article/pii/S0888327018303108 (2019).

25. Jia, F., Lei, Y., Lin, J., Zhou, X. & Lu, N. Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. *Mechanical Systems and Signal Processing* **72-73,** 303–315. ISSN: 0888-3270. https://www.sciencedirect.com/science/article/pii/S0888327015004859 (2016).

26. Khan, S. & Yairi, T. A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing* **107,** 241–265. ISSN: 0888-3270. https://www.sciencedirect.com/science/article/pii/S0888327017306064 (2018).

27. Ge, M., Du, R., Zhang, G. & Xu, Y. Fault diagnosis using support vector machine with an application in sheet metal stamping operations. *Mechanical Systems and Signal Processing* **18,** 143–159. ISSN: 0888-3270. https://www.sciencedirect.com/science/article/pii/S0888327003000712 (2004).

28. Widodo, A. & Yang, B.-S. Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical Systems and Signal Processing* **21,** 2560–2574. ISSN: 0888-3270. https://www.sciencedirect.com/science/article/pii/S0888327007000027 (2007).

29. Gryllias, K. & Antoniadis, I. A Support Vector Machine approach based on physical model training for rolling element bearing fault detection in industrial environments. *Engineering Applications of Artificial Intelligence* **25.** Special Section: Local Search Algorithms for Real-World Scheduling

and Planning, 326–344. ISSN: 0952-1976. https://www.sciencedirect.com/science/article/pii/S0952197611001631 (2012).

30. Zenati Houssam Romain Manon, F. C.-S. *Adversarially Learned Anomaly Detection* in *2018 IEEE International Conference on Data Mining (ICDM)* (2018), 727–736.

31. Bull, L., Worden, K., Manson, G. & Dervilis, N. Active learning for semi-supervised structural health monitoring. *Journal of Sound and Vibration* **437,** 373–388. ISSN: 0022-460X. https://www.sciencedirect.com/science/article/pii/S0022460X18305479 (2018).

32. Rogers, T. *et al.* A Bayesian non-parametric clustering approach for semi-supervised Structural Health Monitoring. *Mechanical Systems and Signal Processing* **119,** 100–119. ISSN: 0888-3270. https://www.sciencedirect.com/science/article/pii/S088832701830623X (2019).

33. Pimentel, M. A., Clifton, D. A., Clifton, L. & Tarassenko, L. A review of novelty detection. *Signal Processing* **99,** 215–249. ISSN: 0165-1684. https://www.sciencedirect.com/science/article/pii/S016516841300515X (2014).

34. Gryllias, K., Antoniadis, I. & Yiakopoulos, C. A novel semi-supervised mathematical morphology-based fault detection and classification method for rolling element bearings. *The 22nd International Congress on Sound and Vibration.* cited By 1, 8 (2015).

35. Li Peng Eickmeyer Jens, N. O. *Data Driven Condition Monitoring of Wind Power Plants Using Cluster Analysis* in *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery* (2015), 131–136.

36. Scholkopf Bernhard Platt C. John, S.-T. J. Estimating the Support of a High-Dimensional Distribution. *Neural computation* **13,** 1443–1471. ISSN: 1530-888X (2001).

37. Morales-Forero, A. & Bassetto, S. *Case Study: A Semi-Supervised Methodology for Anomaly Detection and Diagnosis* eng. in *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (IEEE, 2019), 1031–1037. ISBN: 1728138043.

38. Zhou, C. & Paffenroth, R. *Anomaly Detection with Robust Deep Autoencoders* eng. in *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining* (ACM, 2017), 665–674. ISBN: 1450348874.

39. Alpaydin, E. *Introduction to machine learning / Ethem Alpaydin.* Third edition. eng. ISBN: 9780262325745 (2014).

40. Sakoe, H. & Chiba, S. Dynamic programming algorithm optimization for spoken word recognition. eng. *IEEE transactions on acoustics, speech, and signal processing* **26,** 43–49. ISSN: 0096-3518 (1978).

41. Petr, V., David, N. & Pavel, Z. Employing Subsequence Matching in Audio Data Processing. eng (2012).

42. Carrier. *How Do Air Conditioners Work?* `https://www.carrier.com/residential/en/us/products/air-conditioners/how-do-air-conditioners-work/#`. (accessed: 01.07.2021).

43. Wing, C. *How your house works a visual guide to understanding and maintaining your home* 3rd ed. (Hoboken, N.J. : Wiley, 2018).

44. MitsubishiElectric. *Mitsubishi Electric Service Manual R410A* (Mitsubishi Electric, 2015).

45. MitsubishiElectric. *Reasons Your Air Conditioner is Not Cooling* `https://www.mitsubishielectric.co.id/article/read/2020/05/29/24/5-reasons-your-air-conditioner-is-not-cooling`. (accessed: 01.07.2021).

46. Paul Horowitz, W. H. *The art of electronics* 3rd ed. (New York, NY, USA : Cambridge University Press, 2015).

47. Weedy, B. *Electric power systems* 5th ed. (New York : Wiley, 2012).

48. Chester H. Page, P. V. *The International Bureau of Weights and Measures* ISBN: 9780198520115 (U.S. DEPARTMENT OF COMMERCE National Bureau of Standards, 1975).

49. Roland E. Thomas Albert J. Rosa, G. J. T. *The analysis and design of linear circuits* 7th ed. (Hoboken, N.J. : Wiley, 2012).

50. Rashid, M. H. *Power Electronics Handbook* 4th ed. (Saint Louis : Elsevier Science, 2017).

51. Michael T. Heideman Don H. Johnson, C. B. Gauss and the History of the Fast Fourier Transform. *Archive for history of exact sciences* **34,** 265–277 (1985).

52. Bradley, A. P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition* **30,** 1145–1159 (1997).

53. Learn, S. *Agglomerative Clustering* `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html`. (accessed: 01.07.2021).

54. Mihael Ankerst Markus M. Breunig, H.-P. K. OPTICS: Ordering Points to Identify the Clustering Structure. *SIGMOD record* **28,** 49–60 (1999).

55. Ward, J. H. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association* **58,** 236–244 (1963).

56. Lessig, V. P. Comparing Cluster Analyses with Cophenetic Correlation. *Journal of marketing research* **9,** 82–84 (1972).

57. Nielsen, F. *Introduction to HPC with MPI for Data Science* (Springer, 2016).

58. Fix Evelyn, H. L. J. Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties. *International statistical review* **57,** 238–247 (1989).

59. Sridhar Ramaswamy Rajeev Rastogi, K. S. Efficient algorithms for mining outliers from large data sets. *SIGMOD record* **29,** 427–438. ISSN: 0163-5808 (2000).

60. Markus M. Breuniq Hans-Peter Kriegel, R. T. N. Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties. *SIGMOD record* **29,** 93–104. ISSN: 0163-5808 (2000).

61. Schubert, E., Zimek, A. & Kriegel, H.-P. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. **28,** 190–237. ISSN: 1384-5810 (2014).

62. Mary M. Moya, D. R. H. Network constraints and multi-objective optimization for one-class classification. *Neural Networks* **9,** 463–474. ISSN: 0893-6080 (1996).

63. Oliveri, P. Class-modelling in food analytical chemistry: Development, sampling, optimisation and validation issues – A tutorial. *Analytica Chimica Acta* **982,** 9–19. ISSN: 0003-2670 (2017).

64. Noumir, Z., Honeine, P. & Richard, C. *On simple one-class classification methods* in *2012 IEEE International Symposium on Information Theory Proceedings* (IEEE, 2012), 2022–2026. ISBN: 9781467325806.

65. Fei Tony Liu Kai Ming Ting, Z.-H. Z. *Isolation Forest* in *2008 Eighth IEEE International Conference on Data Mining* (IEEE, 2008), 413–422.

66. Fei Tony Liu Kai Ming Ting, Z.-H. Z. Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data* **6,** 1–39 (2012).

67. Wenjie Zhao Yushu Zhang, Y. Z. Anomaly detection of aircraft lead-acid battery. **37,** 1186–1197. ISSN: 0748-8017 (2021).

68. Bandaragoda, T. R. *et al.* Isolation-based anomaly detection using nearest-neighbor ensembles. **34,** 968–998. ISSN: 0824-7935 (2018).

69. Learn, S. *Open source python machine learning library* `https://scikit-learn.org/stable/`. (accessed: 01.07.2021).

70. Vercruyssen, V. *Anomatools* `https://github.com/Vincent-Vercruyssen/anomatoolsg`. (accessed: 01.07.2021).