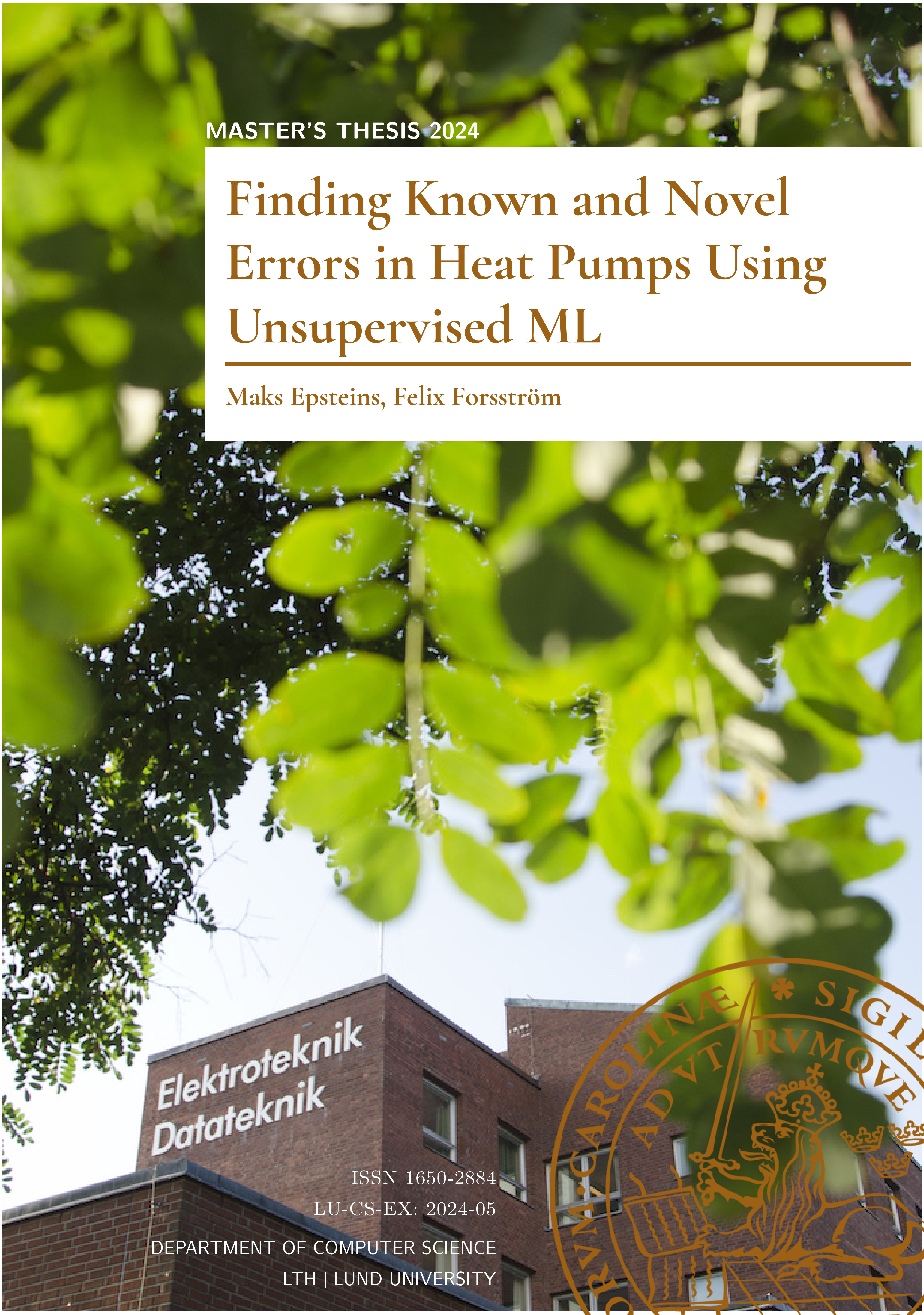


MASTER'S THESIS 2024

Finding Known and Novel Errors in Heat Pumps Using Unsupervised ML

Maks Epsteins, Felix Forsström

ISSN 1650-2884
LU-CS-EX: 2024-05

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2024-05

Finding Known and Novel Errors in Heat Pumps Using Unsupervised ML

Hitta Kända och Nya Fel i Värmepumpar Med Hjälp av Oövervakad Maskininlärning

Maks Epsteins, Felix Forsström

Finding Known and Novel Errors in Heat Pumps Using Unsupervised ML

Maks Epsteins
maksepsteins@outlook.com

Felix Forsström
felix.forsstrom@hotmail.com

January 25, 2024

Master's thesis work carried out at Robert Bosch AB.

Supervisors: Jacek Malec, jacek.malec@cs.lth.se
Vilhelm Åkerström, vilhelm.akerstroem@se.bosch.com

Examiner: Elin A. Topp, elin_a.topp@cs.lth.se

Abstract

Modern heat pumps are complex systems prone to multiple different types of faults. Current IoT technologies allow these machines to continuously report data and error codes corresponding to the aforementioned faults. This thesis explores the possibility of identifying anomalous behaviour corresponding to registered error codes, as well as identifying novel errors, using unsupervised and semisupervised anomaly detection. This research opens up possibilities to analyse IoT systems for health and product analysis without relying on human labeling of operational data. The thesis investigated six different anomaly detection models with different approaches to anomaly detection evaluated against four different preprocessing configurations of the data. The results showed random-guess performance in the general case, but for some specific heat pump and anomaly detection models a >0.9 *Matthews Correlation Coefficient* score was achieved. However, it was deemed that there is no single model and preprocessing configuration that consistently outperformed the others. The discovery of novel errors was evaluated as the ability to automatically cluster registered error points into coherent clusters of the same error code, the idea being that the same technique could then be applied to false positive points, the clusters for which can be evaluated by a domain expert. Using dimensionality reduction techniques the clustering achieved adequate results for a majority of the heat pump systems, with an average adjusted Rand index score of 0.43.

Keywords: Anomaly Detection, Anomaly Clustering, Heat Pumps, Unsupervised Machine Learning, Semisupervised Machine Learning

Acknowledgements

Firstly, we would like to thank our university supervisor Jacek Malec, for his continuous support and guidance throughout the duration of this thesis. Secondly, we would like to thank Markus Klang, for helping guide us in the right direction. Lastly, we would like to thank Bosch AB and our supervisor at Bosch AB, Vilhelm Åkerström as well as our coworker Charles Barbosa for his after work boxing classes.

Contents

Chapter 1
Introduction

Today, IoT-connected devices are more present than ever before. Industry 4.0 and the use of connected devices is driving the need for efficient analysis of the generated data in order to make informed decisions. Heat pump systems are no exception. By providing heat pump systems with internet connectivity and smart sensors manufacturers have access to large quantities of information that can be used to increase system knowledge. This knowledge can then be used to provide services such as predictive maintenance, diagnostics, and to guide development of potential improvements to the machines. The focus of this thesis is to investigate the possibility of using unsupervised machine learning methods to find previously unidentified error causes, by performing anomaly detection and then attempting to cluster the found points to find common patterns. These patterns can then be used to learn more about the heat pump machines, as well as identify causes for system malfunction. This thesis was undertaken collaboratively with Robert Bosch AB, which facilitated access to sensor data from heat pump systems distributed in various European countries.

1.1 Background

The fundamental purpose of a heat pump is to transfer heat (and air) either into an enclosed space (heating), or out of an enclosed space (cooling), thereby changing the temperature of the facility and improving air quality [36]. When heating a facility, the facility becomes the destination of the heat transfer, and to cool a facility the facility instead becomes the source [36]. This makes heat pumps important machines, since they allow facilities to retain a comfortable temperature regardless of the time of year. In addition to being able to heat facilities, heat pumps can also be used to heat water for both heating and domestic use such as showering [36].

In order for a heat pump to be able to transfer heat from a hot source to a cold source the heat pump needs to transfer energy from one medium into another [15] (see 1.1). What defines the type of the heat pump is which two mediums the heat pump uses to transfer heat to and from, in turn requiring different mechanisms to conduct energy transfer [15]. This thesis will focus on AW (air to water) heat pumps, meaning heat pumps that transfer energy from air to water. An abstract model of how a heat pump works may be seen in 1.1 which showcases a simple overview of how this heat transfer occurs.

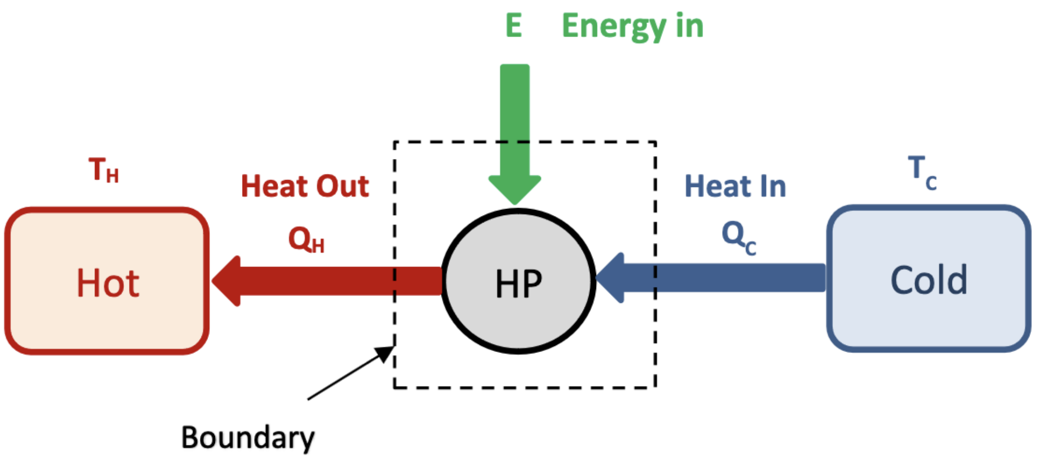


Figure 1.1: Abstract model of a heat pump. Source: *An Introduction Heat Pumps* [30]

As with most complex machines, heat pumps have multiple components and many moving parts. This in turn means that the heat pumps may deteriorate over time, and cause errors that can either be momentary faults without lasting effect or cause total system failure. This makes diagnostics crucial, both for investigating issues, but also to be able to discover areas of improvement for the machines. In order to facilitate diagnostics of the heat pumps, the machines store procedurally generated logs containing a collection of sensor readings and operational information, and how they change over time. These logs can be described collectively as a multivariate time series data set, containing information about the heat pumps and how they are performing. This data also includes preprogrammed error codes, which are triggered when some feature or sensor enters a faulty state.

The aforementioned data sets are continuously growing in size during machine operations and quickly reach a level where manual analysis becomes a difficult and time consuming process. In order to automate this process, a popular approach is to utilise machine learning techniques since they are able to find patterns in data that would otherwise be impractical to gather manually. However, a disadvantage of many machine learning techniques is that they often require labelled data, meaning that the data set has to be manually annotated by domain experts which in itself is an expensive and time consuming process. A way to be able to avoid this is to use so called unsupervised or semi-supervised machine learning techniques, which do not require a pre-labeled data set. A commonly performed task utilising unsupervised/semi-supervised techniques is *anomaly detection*, which is the attempt to identify points that behave differently to normal data points.

One advantage of these techniques over supervised ones is that the points they find are not based on previously known information, and thus they can find new patterns in the data. This in turn can potentially allow finding information, previously unbeknownst to the domain professionals.

The idea of this thesis is to use unsupervised/semi-supervised anomaly detection techniques to predict anomalies in the machine data, which can then be verified using the aforementioned error data set as ground truth. Furthermore, we aim to investigate the possibility of finding novel errors by identifying clusters of common properties among the points that were classified as anomalous that do not have a corresponding error. The purpose of the clustering was to be able to identify whether or not the aforementioned found anomalous points correlate to a previously unknown error cause, which can then be evaluated and labeled by a domain professional.

1.2 Problem Definition

The requirements for simultaneous analysis of temporal patterns and multivariate co-dependent sensor measurements makes multivariate time series anomaly detection (MTSAD) a challenging task [2]. This thesis aims to investigate the applicability of MTSAD on a real-world heat pump machine log dataset. To the best knowledge of the authors, no study has been done on the applicability of unsupervised MTSAD on a real-world dataset of heat pump sensor data to both identify anomalous behaviour corresponding to error codes, as well as identify novel anomalies. Thus, this thesis aims to provide more knowledge on the possibilities of this approach, providing research on data-driven development of heat pumps in particular, and other multi-sensor industrial IoT systems in general. The study compares the performance of the following six anomaly detection methods: Empirical Cumulative Distribution Functions (*ECOD*) [21], Isolation Forest (*IF*) [22], Deep Isolation Forest (*DIF*) [39], Local Outlier Factor (*LOF*) [5], Autoencoder (*AE*) [41] and Long Short-Term Memory Autoencoder (*LAE*) [28].

The research questions are as follows:

> *How efficient are the selected anomaly detection methods at identifying data points corresponding to error codes in heat pump system data?*
>
> *Can we use a clustering algorithm in order to be able to profile/group the predicted anomalies to find new error causes?*

1.3 Contribution Statement

This work provides to Robert Bosch AB an investigation into anomalous behaviour in operational data for their heat pumps. This may be used by Robert Bosch AB to improve internal processes, further product development and potentially enable predictive maintenance operations.

The work contributes a comparative study to the research community on unsupervised anomaly detection methods on a real world dataset of multivariate time series operational data; with specific contributions to the domain of heat pump analysis. The study highlights the possibilities of several anomaly detection methods to be used for identifying erroneous behaviour in multivariate time series and discusses the specific strengths and weaknesses of the methods from a theoretical standpoint and how they translate to a real world dataset.

1.4 Methodology

The aim of this thesis was to investigate the applicability of unsupervised anomaly detection techniques on a real world multivariate dataset. To get a nuanced and up-to-date evaluation, a number of unsupervised anomaly detection techniques were selected with the purpose of including models of fundamentally different approaches to anomaly detection. This resulted in the six models introduced above in section 1.2 repeated here: *ECOD*, *IF*, *DIF*, *LOF*, *AE* and *LAE*. Description for the models as well as motivation for their choice can be found in chapter 3.

Thereafter the provided data set was investigated (see section 4.3, titled *Data Understanding* for more details) to determine what preprocessing steps would need to be performed in order to transform the data into a format conductive for anomaly detection. An example of some of the methods used include *Sequential Forward Selection* (SFS), and *Uniform Manifold Approximation and Projection* (UMAP), for further details, see section 4.4. Finally all models were constructed, trained and evaluated on the heat pump dataset.

The models were in general constructed using the Python based anomaly detection framework named PYOD [43] (Python Outlier Detection) which is built on top of the popular machine learning framework Scikit-learn [31] with the exception of the LSTM autoencoder which was built using the Tensorflow Keras library.

Evaluation was conducted using the metrics discussed in section 3.7 titled *Evaluation Metrics*. Finally the evaluations were used to draw conclusions on the applicability of these models on the data set under investigation; answering the presented research questions.

1.5 Related Work

In a study by Leahy et al. [19] they investigated the possibility to predict faults in a wind turbine using operational data and the support vector machine anomaly detection algorithm. They achieved positive results on identifying some faults, however the generality of the ability to detect faults was low, pointing to the difficulty of finding a general well performing solution in this sort of task.

Beghi et al. [1] investigated fault detection and diagnosis (FDD) on a water chiller. By using domain knowledge, they generated a select number of characteristic features susceptible to faults. Through a semi supervised statistical approach utilising a PCA projection and measuring the prediction error of projections onto the model, they achieved good results at predicting faults. Notable is the use of a Savitzky-Galoy filter (see section 2.4.1) for enhancing the detectability of errors; a method that showed potential in this thesis as well. In contrast to the data-driven approach utilised in this thesis, their work was characterized by the ability to utilize domain knowledge to select/construct features with a high sensitivity to faults.

1.5.1 Outline

The overall structure begins with a general overview of the theory required to understand the utilised process. The theory is separated into two chapters. First, the theory behind the performed data processing techniques is introduced, with motivations on their applicability on the data. Secondly, theory on anomaly detection and clustering is explained together with the utilised evaluation metrics. Subsequently, the employed approach is explained, describing the procedural steps performed throughout the course of this thesis. Lastly, the results are presented together with a discussion on the findings in addition to the final conclusions.

1.5.2 Work Distribution

The work performed in this thesis was evenly distributed and performed in full collaboration between the two authors.

Chapter 2

Data Processing

An important step in any machine learning pipeline is transforming the data into a format that makes it suitable for the given task. This transformation is highly task dependent, but can , for example, include extrapolating new information from the features, or scaling the features to prevent some features from dominating in the predictions [10]. While data processing is important for all machine learning tasks, it can be especially important when performing unsupervised anomaly detection. This is due to the fact that unsupervised methods lack supervision for which specific patterns they are supposed to identify, and are thus more sensitive to noise when separating normal points from anomalies (for more details, see chapter 3).

The data set that the unsupervised anomaly detection was performed on in this thesis can be described as a multivariate time series data set, constructed from data reported by multiple sensors. The reported data can formally be described as a sequence of numerical samples recorded at a discrete time interval. Given K sensors the measurement at time instant n can be defined as [2]

$$x[n] = (x_1[n], x_2[n], ..., x_K[n])^T$$

The time series then becomes a matrix of the form

$$X = (x[1], x[2], x[3], ..., x[N]) = \begin{pmatrix} x_1[1] & x_1[2] & \cdots & x_1[N] \\ x_2[1] & x_2[2] & \cdots & x_2[N] \\ \vdots & \vdots & \ddots & \vdots \\ x_K[1] & x_K[2] & \cdots & x_K[N] \end{pmatrix}$$

where the element at position (n, k) represents the measurement of sensor k at time instance n, and each column $x[n]$ represents a feature vector consisting of all the produced measurements. More details about the data set can be found in section 4.3, titled *Data Understanding*. This chapter introduces the theory for the processing methods used for examining the data and performing anomaly detection/clustering.

2.1 Feature Engineering

Feature engineering in the context of anomaly detection is the practice of extracting, transforming, and manipulating the raw data into features more conductive to detecting anomalous behaviour. The general aim of this feature transformation is to improve the performance of the algorithms, and their ability to discriminate anomalous from normal data. The utilised feature engineering methods include:

- *Normalisation:* The process of scaling features to a standard range (see section 2.2).
- *Dimensionality reduction:* The process of reducing data dimensionality, either by selecting a subset of the features (*feature selection*) or by combining features (*feature extraction*). The aforementioned methods are further described in section 2.3.
- *Interpolation:* The process of filling missing values in a data set in order to create a continuous representation (see section 4.4.1).
- *Seasonality Removal:* The removal of seasonal trends in data to enhance deviations from the norm (see section 2.4).

Feature engineering is an important step in any machine learning application and can have a large impact on the performance of an anomaly detection task [29, 10]

2.2 Normalisation Methods

When performing machine learning, it is important to identify how the used models are affected by differing scales or variance between features in the data [3]. If the features are of differing scales, it can cause features with a larger variance to dominate, which in turn can negatively affect performance [3]. To circumvent this problem, it is common to perform feature normalisation, which scales all features so that they are in approximately the same ranges [3]. Here the employed normalisation methods (as described in section 4.4.3) are introduced, namely Min-Max and Z-score normalisation.

2.2.1 Min-Max Normalisation

Min-Max normalisation is a normalisation method that scales the feature vector so that all values are between two values; typically to either a range of $[0, 1]$ or $[-1, 1]$. The normalisation procedure on the range [low, high] is performed as follows [3]:

$$x_{norm} = \frac{(high - low) * (x - x_{min})}{x_{max} - x_{min}}$$

Where x is a singular value in our list of feature values, x_{norm} is the normalized x value, x_{min}, x_{max} are the smallest and largest value in the list respectively.

2.2.2 Z-Score Normalisation

The other method used is Z-score normalisation, where instead of guaranteeing that all features are on the same range it instead scales the values so that they have a mean of 0 and a standard deviation of 1 [3]. The Z-score normalisation is calculated as follows:

$$x_{norm} = \frac{x - \mu(X)}{\delta(X)}$$

The calculated value x_{norm} corresponds to a single sample after the normalisation procedure. The remaining definitions are as follows: x is a single sample, X is the list of all samples, $\mu(X)$ is the mean, and $\delta(X)$ is the standard deviation.

2.3 Dimensionality Reduction

When performing machine learning on high dimensional data, a common problem that can occur is referred to as *the curse of dimensionality* [20]. This phenomenon is where data tends to be sparser when dimensionality increases, and thus algorithms that are designed to work on low-dimensional space tend to be affected [20]. Furthermore, the more features are introduced the more models have a tendency to overfit, and can thus provide lower performance [20]. A way to circumvent this *curse of dimensionality* is to use a dimensionality reduction method, which can be categorized in two categories, namely *feature selection* and *feature extraction*. Furthermore, dimensionality reduction can be a powerful data analysis tool, often used to reveal simpler underlying patterns in high-dimensional data. This in turn can make dimensionality reduction an important step, assisting with increased understanding and enabling more accurate modeling for any data processing [35]. The dimensionality reduction methods used in this thesis are introduced below, namely *PCA*, *UMAP*, and *Sequential Forward Selection*.

2.3.1 PCA Projection

A commonly employed technique dimensionality reduction technique for machine learning and data analysis is called *Principal Component Analysis* (PCA). PCA is popular due to its ability to reveal underlying structures of complex multivariate data sets [35]. This is achieved by transforming the original high-dimensional dataset into a new coordinate system, consisting of principal components that are linear combinations of the original dimensions. It can be briefly summarized in the following two steps: (1) Given a $m \times n$ matrix of m types of samples and n samples, subtract the mean of each m column from each row n; (2) calculate the eigenvectors of the covariance matrix, which correspond to the principal components of the matrix [35]. The derived principal components capture the variance of the original dataset in decreasing order, where the first principal component captures the most variance and subsequent ones capture less.

In this thesis these properties of PCA were used for data understanding (see section 4.3), in order to analyse the seasonal behaviour of the data (see section 2.4) and to evaluate the clustering and novelty detection capabilities of the predicted anomalies (see section 3.6 for theory and section 6.3 for discussion).

2.3.2 UMAP Projection

UMAP stands for *Uniform Manifold Approximation and Projection* and is a dimensionality reduction method that prioritises the preservation of local distances over global ones. It is used as both a visualization and preprocessing technique for machine learning, typically outperforming other popular algorithms in terms of computational complexity and preservation of global structures [25]. UMAP is in the family of algorithms called k-neighbour graph based algorithms which can be described in a two step process. Firstly, the algorithm constructs a weighted k-neighbour graph [25]. Secondly, the k-neighbour graph is projected into a lower dimensional space [25]. Where the k-neighbour graph based algorithms differ is how these two steps are performed, and thus we will continue to explain how UMAP performs these steps.

UMAP Graph Construction

The first step in constructing a UMAP projection is to construct a k-neighbour graph [25]. Let the input data set be $X = \{x_1, \ldots, x_N\}$. Additionally a dissimilarity measure function is chosen $d : X \times X \to \mathcal{R}_{\geq 0}$ which determines the dissimilarity between two points x_1, x_2 [25]. With given hyper-parameter k, for each $x_i \in X$, the set $\{x_{i1}, \ldots, x_{ik}\}$ of k-closest neighbors to x_i are calculated using the d dissimilarity function [25]. Next, for each $x_i \in X$ two values ρ_i and σ_i are defined. ρ_i is defined as follows [25]:

$$\rho_i = min\{d(x_i, x_{ij}) | 1 \leq j \leq k, d(x_i, x_{ij}) > 0\}$$

While σ_i is defined as the value such that [25]:

$$\sum_{j=1}^{k} exp\left(\frac{-max(0, d(x_i, x_j) - \rho_i)}{\sigma_i}\right) = log_2(k)$$

Using these definitions we can define the weighted graph $\mathcal{G} = (V, E, w)$ where the vertex set V is the same as X and the edge set E is defined as $E = \{(x_i, x_{ij}) | 1 \leq j \leq k, 1 \leq i \leq N\}$ [25]. Lastly, the weight function w is defined as [25]:

$$w(x_i, x_{ij}) = exp\left(\frac{-max(0, d(x_i, x_j) - \rho_i)}{\sigma_i}\right)$$

Let A be a weighted adjacency matrix of the graph $\mathcal{G}$, a symmetric matrix B can be constructed such that [25]:

$$B = A + A^T - A \circ A^T$$

Here $\circ$ is the Hadamard product, which corresponds to an element wise multiplication of the matrices.

The A and B adjacency sets may be interpreted as (a) A_{ij} is the probability that there exists an edge from x_i to x_j while (b) B_{ij} is the probability that either the edge x_i to x_j exists or the edge x_j to x_i [25]. The adjacency matrix of the UMAP graph $\mathcal{G}$ is the matrix B [25].

UMAP Graph Layout

UMAP uses a force directed layout algorithm which defines repulsive forces between vertices and attractive forces between edges [25]. The graph layout step applies the repulsive and attractive forces for each edge and vertex iteratively, continuously updating the position of the projected points. This iterative process makes UMAP a non-convex optimization problem, where the convergence instead occurs due to the diminishing of a scaling factor that after a set amount of iterations diminishes to 0.

UMAP projection was used in multiple ways throughout this thesis. It was used for data understanding (see section 4.3) and feature extraction (see 4.4). The reason *UMAP* was used for feature extraction instead of *PCA*, is that *UMAP* generally performs better at separating observations that are non-linear in nature [25].

2.3.3 Feature Selection

Feature selection is a dimensionality reduction method which improves algorithmic efficiency while at the same time improving learning performance by finding a best performing subset of features for the given task [20]. Feature selection works by selecting a subset of the initial features, in contrast to feature extraction which works by returning a combination of features [20]. Since feature selection methods retain the original features, feature selection has improved explainability and is thus often the preferred method for dimensionality reduction [20].

High-dimensional real-world data tends to contain features that might add unnecessary complexity and not contribute to the task, and those features can thus be categorised as redundant features. Some features may also be noisy, which might have negative effects on the resulting model. By conducting feature selection, these features can be removed to enhance performance while reducing storage and computational costs [20]. The feature selection method used in this thesis is called Sequential forward selection (*SFS*). *SFS* is a supervised feature selection algorithm which starts from an empty set $\emptyset$ of features and then iteratively trains models to detect up to p best performing features for identifying a target [32].

If we define the input as being a matrix of feature vectors of d features $Y = \{y_1, y_2, \dots, y_d\}$ and $X_0 = \emptyset, k = 0$, then the algorithm for determining the p best features is done by iteratively calculating X_k as $X_k = X_k \cup x^+$, where $x^+ = \mathit{argmax}\ J(X_k + x)$. Here, J refers to the criterion function (the classifier) and x^+ may be interpreted as the feature that maximises the result of J [32]. In this thesis *SFS* was utilised to investigate the effect of feature reduction on the high dimensionality data as is discussed in section 6.2.3. Do note that this is a supervised method, and thus requires labeled data which is typically not available when performing unsupervised machine learning tasks. The reasoning for using feature selection in spite of this fact, is that the feature selection acts as a replacement for domain expertise in order to select the most appropriate features for finding anomalies.

2.4 Seasonality in Time Series

When observing systems producing time-series data, the inherent stationarity of the data set is important to take into account when making assumptions and predictions. Stationary systems are systems wherein the statistical properties, namely mean, variance, and auto-correlation are static, and do not vary over time [7]. A time-series can exhibit non-stationary behaviour in different ways.

Change point is when some permanent change in the system is introduced, altering its normal behaviour. This change is often drastic and noticeable in the data. It can be an expected change, for example changing a component in the system, or a unexpected change, for example increased system usage due to outside influence.

Concept drift is when the statistical distribution of time series data shifts over time [7]. One specific instance of concept drift is called *seasonality*. This is drift over time in a seasonally repeating pattern, specifically a cyclical pattern with a significantly longer period than the sample rate [7].

A time series that exhibits non-stationary behaviours makes many anomaly detection techniques difficult. The goal of anomaly detection can be defined as *the ability to detect behaviour deviating from the norm* [7]. It is then clear that when the norm shifts over time the same identifiers cannot always be used to determine if points are anomalous. To counteract this, one can normalise the data using knowledge about the non-stationary behaviour. For example a change points impact can be measured and subtracted from the data. A seasonal time series can subtract the cyclical repeating pattern in the data, thus normalising the mean and variance. This requires at least two periods of the data in order to predict what the seasonal behaviour is.

Due to the inherent limitations of the data we have been provided (see chapter 4.3 for details) we are not able to perform traditional methods for removing seasonality. The data we have been provided is limited to around 450 days, and thus we do not have two seasonal periods (i.e two years). Instead we will focus on signal processing techniques to estimate and visualize the general trend in the data.

2.4.1 Savitzky-Golay Filter

The Savitzky-Golay filter (henceforth referred to as either SAVGOL or SG filter), is a signal smoothing method utilising a least-squares fit convolution [23]. A SG filter is essentially a weighted moving average filter, to which one can provide weight coefficients in the form of a polynomial. In order to perform SG filtering, two parameters must be provided, a window size and the degree of the polynomial [23].

The SG smoothing process can be described as follows:

$$s_j^* = \frac{\sum_{i=-m}^{i=m} c_i s_{j+i}}{N}$$

Here s represents the original signal and s^* is the signal after the smoothing procedure. The coefficient for the i-th smoothing is c_i, N equates to the number of points within the smoothing window and is equal to $2m + 1$, where m is half the width of the window [23].

This method is used to remove seasonality with the idea that the seasonality trend of the data should equate to the smoothed signal. This is not a perfect method for removing seasonality, since SG filters are sensitive to outliers during the smoothing process [23]. However, due to the limitations of the data (more specifically referring to the aforementioned lack of two full seasons) an SG filter was determined to be a suitable option.

Chapter 3
Anomaly Detection

Anomaly detection is the practice of identifying outliers, i.e., entities differing from the norm of the data in a significant way. In this chapter we will first introduce definitions of anomalies, then discuss important considerations when choosing an anomaly detection technique, followed by descriptions of the utilised multivariate time series anomaly detection (MTSAD) methods. Finally, we will discuss the motivation for the choice of techniques for performing anomaly detection on heat pump-sensor time-series data. Note that henceforth, the words *point*, *observation*, and *sample* are used interchangeably, all referring to an element at a unique time instance of a multivariate time series dataset.

3.1 Different Types of Anomalies

Anomalies may be defined in different ways depending on the subject matter. In the case of multivariate systems, an anomaly may be defined as [7]: *the measurable consequences of an unexpected change in state of a system which is outside of its local or global norm.* This definition highlights the challenges and complex nature of multivariate data sets. The definition points to the fact that the normal state of the machine can shift over time, and that the data observations provide a limited perspective on the true operation of the system. In order to give further understanding of the inherent behaviours of anomalies and the behaviours that the anomaly detection models are attempting to predict, we will in this section present the different types of anomalies that may occur in multivariate data.

In figure 3.1 three types of anomalies in univariate data are visualised. They are *point anomalies*, *contextual anomalies*, and *collective anomalies* [2]. Point anomalies are clear deviations from the global norm in a single point. Contextual anomalies are point anomalies but only if regarded in a specific local context. Lastly, collective anomalies are a series of consecutive samples collectively deviating from the norm of other series of consecutive samples. All of these anomalies are relatively clear to see and visualise in a univariate dataset, however when regarding multivariate time series data these definitions become more challenging to apply.

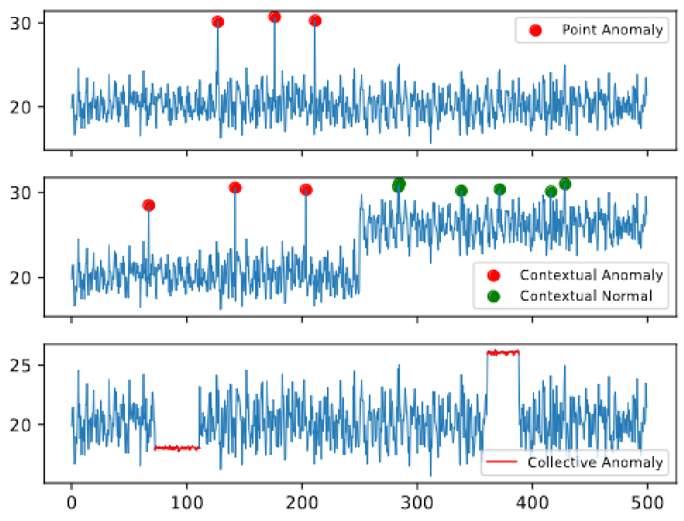


Figure 3.1: Plot showing different types of anomalies. The top plot demonstrates point anomalies marked in red. The middle plot shows contextual anomalies. The last plot shows collective anomalies. Source: *Unsupervised anomaly detection for iot-based multivariate time series* [2]

A sample may be within normal operations given its univariate measurements but when seen in association with other variables it may be regarded as an anomaly [2]. Industrial machines also tend to have degrading performances over time, so called *over time anomalies* [27]. Due to limited domain knowledge over the degrading properties of the machines these anomalies are outside the scope of this thesis.

3.2 Choice of Anomaly Detection Technique

There exist many choices for anomaly detection techniques with different advantages and disadvantages. As stated by the *no free lunch theorem* no technique will be universally best for every problem [38]. When performing MTSAD, the best choice of technique is highly dependent on the structure of the data [16]. It is common that labeling data to acquire a ground truth is prohibitively expensive in real-world scenarios, therefore this study focuses on unsupervised and semi-supervised anomaly detection methods, i.e., anomaly detection methods that do not require a ground-truth for training. When choosing an appropriate anomaly detection method key factors to consider are:

- **Suitability:** How well suited is the algorithm to identify the specific types of anomalies present in the data set?
- **Assumption:** What base assumption does the anomaly detection method make to define an anomaly?
- **Data-complexity:** How well does the algorithm handle high dimensional data and non-linear relationships between features?

- **Time-complexity:** How well does the algorithm scale with the size and structure of the data set?
- **Explainability:** How well can a mapping between predicted anomaly and main contributing features be constructed?

Below we will introduce the different types of MTSAD techniques that are studied in this thesis including the strengths and weaknesses of the different approaches.

3.3 Different Types of Anomaly Detection Methods

When performing anomaly detection, an appropriate choice of model is essential to achieve positive results. This is due to the fact that what is an anomaly is highly dependent on the structure of the data, and different models use different approaches to determine whether or not a point is an anomaly. Here we will present a collection of different categories of anomaly detection methods, together with their strengths and weaknesses, and why they were considered appropriate for the task.

3.3.1 Density-Based Methods

Density-based anomaly detection methods are based on the intuition that anomalies will appear in regions of lower densities in comparison to other normal points [33]. As can be seen in figure 3.2 there are instances where an anomaly does not appear to be an anomaly when observed globally, but is instead anomalous in relationship to its locality. The point O in 3.2 would most likely be classified as a normal point if looking at the data globally, however, when looking at the point O in relation to its locality, it may be observed that the point is in fact anomalous in relation to its local cluster C1. The points in C2, even though they are more spread out, form a distinct cluster in their own right and it is not as clear which of the points in C2 are anomalous.

Some of the advantages that these methods tend to provide is that they are intuitive, and thus their usage promotes explainability. However, they are also computationally expensive since they require calculations of pairwise distances between points; making them typically not suitable for large and complex datasets [33].

3.3.2 Tree-based Methods

Like many other anomaly detection methods the density-based ones work on the basic premise of profiling normal samples and then identifying anomalies as deviations from that norm. In contrast there is a family of methods called tree-based methods based on the premise of profiling anomalies instead. They rely on the idea that *anomalies are few and far between* meaning that anomalies should be inherently isolated from normal data points [33].

These methods work by splitting up the data feature-wise into sub-trees, determining the anomaly score by the amount of sub-tree partitions needed to be performed in order to isolate a sample.

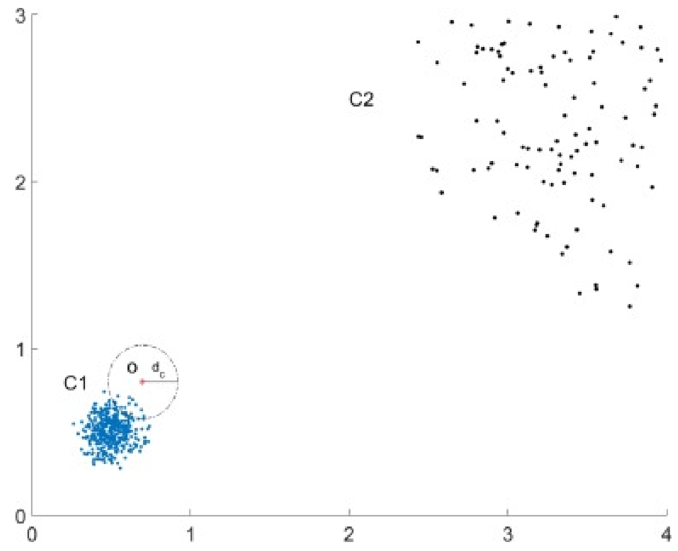


Figure 3.2: Example plot demonstrating the advantage of density-based anomaly detection methods. The image shows two clusters C1 and C2, and the outlier point O which is an outlier in relation to cluster C2 but not globally. Source: *Adaptive kernel density-based anomaly detection for nonlinear systems* [42]

As can be seen in 3.3 the sub-tree partitions can be visualised as "cuts" in the data-space, where the fewer the amount of cuts required to isolated a sample, the higher the anomaly score. The red point in figure 3.3 is marked as an anomaly since the data-set only needed to be partitioned once in order to isolate that point. The main differentiating factor that separates the different tree-based methods is how the partition cuts are created, which we will explain when describing the *Isolation Forest* (see section 3.4.3) and *Deep Isolation Forest* (see section 3.4.4) tree-based methods.

Some advantages of the tree-based anomaly detection approaches is that they tend to have low run time complexity and are scalable even for large datasets. However, tree-based methods, of which Isolation Forest is most widely used, tend to not perform well when anomalies are local in nature in contrast to density-based methods [33]. Isolation Forest also suffers from difficulty of recognising non-linear relationships between different features which can be a problem when performing tree-based methods on complex data sets [2]. This is a problem that Deep Isolation Forest attempts to address by making the sub-tree partitions non-linear in nature, at the expense of added time-complexity (see section 3.4.4)

3.3.3 Statistics-based Methods

Statistical models for anomaly detection are based on the idea that samples in a data set can be modelled using probability distributions, which are then used to identify anomalies as deviations in the distribution [21]. These methods are typically divided into two categories, *parametric* and *non-parametric models*, where non-parametric models tend to be more expensive but have the advantage of being easier to use [21]. The assumption that these models generally make is that if a data set is modelled as either a multivariate statistical distribution or a collection of univariate statistical distributions, then anomalies are samples that occur

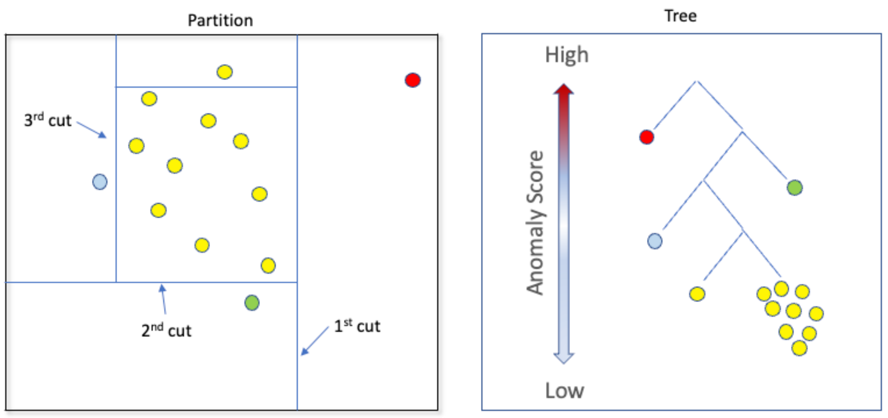


Figure 3.3: An example of isolation forest partitioning of a dataset. Samples are isolated by "cuts" in the dataset. The number of cuts required to isolate a sample represents how anomalous it is. [22]. Source: *Handbook of Anomaly Detection with Python Outlier Detection* [8]

in the tail-end of the distributions [21]. The chosen method *ECOD* (see section 3.4.2) is based on this premise.

3.3.4 Reconstruction-based Methods

Reconstruction-based anomaly detection methods can be described as methods that attempt to deconstruct and then reconstruct the inherent patterns in the data [26]. These methods aim to discover the underlying representation of the input data in a two step process. First the data is deconstructed, reducing its dimensionality, and then reconstructed again by increasing its dimensionality to the dimensionality of the original input. How much the reconstructed signal differs from the initial signal is called the *reconstruction loss*. This loss is the measurement used to determine whether or not a point is an anomaly, with the idea that a reconstruction model trained on normal data will have a high reconstruction loss when faced with anomalous data points as inputs, since it has not been trained to reconstruct them [26].

This behaviour can be seen in 3.4 which shows the original signal, the reconstructed signal, and the reconstruction error. It may be observed that the original and the reconstructed signal appear similar, but when plotting the reconstruction error (i.e., the difference between the original and reconstructed signal), there is deviating behaviour in the area marked red in 3.4, i.e., we have an anomaly. The reconstruction loss is commonly calculated either with MSE (mean square error) or MAE (mean absolute error) [26]. In this thesis, the reconstruction-based methods used were the *Autoencoder* (see 3.4.5) and the *LSTM-Autoencoder* (see 3.4.6 methods. The LSTM-Autoencoder however also takes into account the temporal aspect of the data, as described in the following section.

3.3.5 Time Series-Based Methods

Time-series based anomaly detection methods are inherently different to the previously mentioned methods. The main inherent difference is that time series-based methods are designed

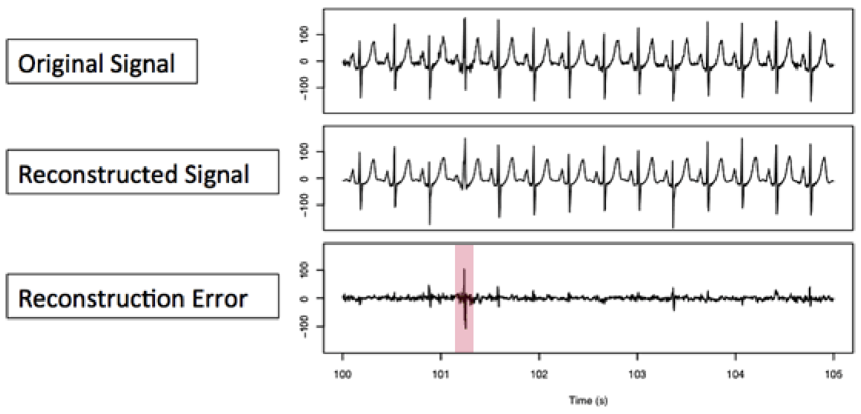


Figure 3.4: Image illustrating how reconstruction-based methods determine an anomaly. The original signal is deconstructed and reconstructed measuring the reconstruction loss. The anomaly is revealed in the area marked in red due to its high reconstruction loss. Source: *New approaches to anomaly detection* [14]

to be able to detect collective and contextual anomalies, meaning anomalies that are anomalous relative to their time locality. The previously mentioned methods are instead mainly for the detection of point anomalies, since they do not consider the time ordering of points during training or classification.

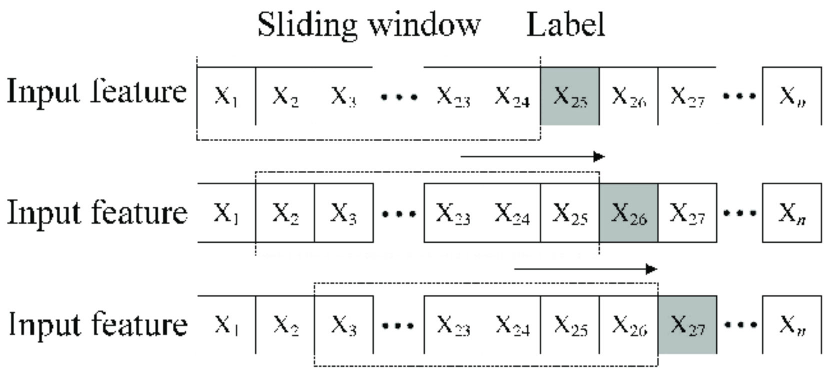

Sliding window
Label
Input feature
X1 X2 X3 ••• X23 X24 X25 X26 X27 ••• Xn
Input feature
X1 X2 X3 ••• X23 X24 X25 X26 X27 ••• Xn
Input feature
X1 X2 X3 ••• X23 X24 X25 X26 X27 ••• Xn

Figure 3.5: A visualisation of the sliding window principle of an LSTM. Source: *Abnormal detection of electricity consumption of user based on particle swarm optimization and long short term memory with the attention mechanism* [4]

The principle of using temporal data to perform classification can be seen in 3.5 where each feature in the data is split into a window of size N and the label for prediction is set to the point occurring after the window. Time series-based methods have the advantage of accounting for the time series behaviour of the data, however, these methods also tend to be more computationally expensive. These methods that utilise the aforementioned sliding window approach can be described as *semi-supervised* since for each window a label is applied that the model then uses as its ground-truth, however, this ground-truth is not pre-labelled but is instead derived from the patterns in the input data [28]. The time-series based method that was utilised in this thesis is called *LSTM-Autoencoder* (see 3.4.6), which partly is a reconstruction based method similarly to the Autoencoder. The LSTM autoencoder is separated into a different category due to the fact that it takes the temporal aspect of the data into account when making its predictions.

3.4 Chosen Anomaly Detection Models

In this section, the chosen anomaly detection methods are introduced, together with the reasoning behind their applicability.

3.4.1 Local Outlier Factor

Local Outlier Factor (LOF) is a type of density-based anomaly detection method which works by calculating a point's local deviation in relation to other close data points [5]. Whether or not a point is determined to be an anomaly is based on the immediate surrounding density of the area around a point in relation to other points in its locality. The lower the density, the higher the anomaly score, and thus the higher the likelihood that a point is classified as an anomaly [5]. In order to define how a LOF for a point is determined we first need to define what the *k-distance of object p* is. The *k-distance of object p* is defined as the distance $d(p, o)$ between object p to point o, point o being the point that is the furthest away from object p while being part of the set consisting of the k closest points to point p [5]. Furthermore we also need to define the *reachability distance of object p to object o* which can be defined as follows [5]:

$$\text{reach-dist}_k(p, o) = max(k\text{-distance}(o), d(p, o))$$

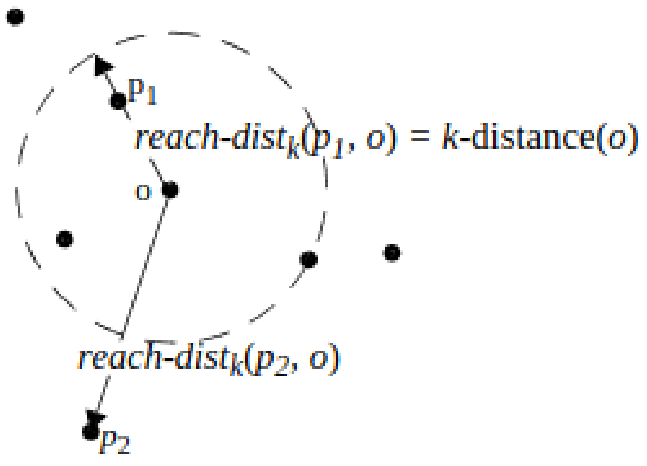


Figure 3.6: Reachability distance of centroid point o to points p_1 and p_2 for $k = 4$. The dashed circle represents a circle of radius 4-distance(o). Source: *Lof: Identifying density-based local outliers* [5]

An illustration of the *reachability distance of object p to object o* can be seen in figure 3.6. Here the reachability distance between point o and point p_1 becomes the k-distance(o) since p_1 is within the k-distance(o) radius of point o, while the reachability distance between point o and point p_2 becomes the distance $d(o, p_2)$ since $d(o, p_2) >$ k-distance(o).
In order to define how LOF determines an anomaly detection score for a point, we now need to define the *local reachability density (lrd) of point p to point o*, which is defined as [5]:

$$lrd_{MinPts}(p) = 1/\left(\frac{\sum_{o \in N_{MinPts(p)}} reach\text{-}dist_{MinPts}(p, o)}{|N_{MinPts(p)}|}\right)$$

This means that the *local reachability density* of object p is the inverse of the mean reachability distance of the *MinPts* nearest points to p. $N_{\text{k-distance}(p)}(p)$ here is called the *k-distance neighbourhood of object p*, which can be explained as that $N_{k\text{-}distance(p)}(p)$ contains all points, such that their distance from $p \leq$ *k-distance*. Above, *k-distance* $= MinPts(p)$, where $MinPts(p)$ is a parameter which specifies the minimum number of points when determining the local density of a specific point.

Using these definitions we can now define the *local outlier factor* (LOF) as:

$$LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts(p)}} \frac{lrd_{MinPts}(o)}{lrd_{MinPts(p)}}}{|N_{MinPts}(p)|}$$

This is how the anomaly score of a point is defined in LOF and it can be described as the mean of the local reachability densities of the point p and p's *MinPts* closest points.

The strength of this method is the fact that it is able to determine whether or not a point is anomalous based on points in its local region.

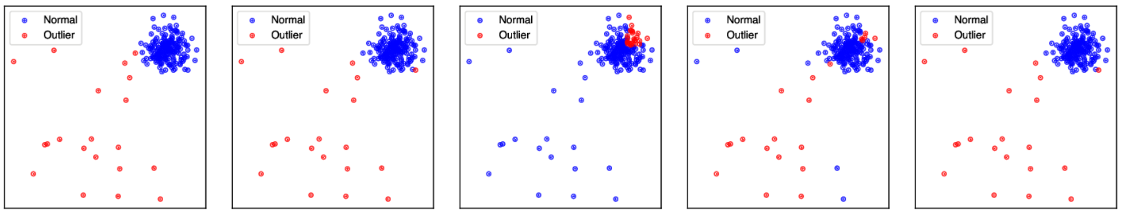
3.4.2 ECOD

ECOD, which stands for *Empirical-Cumulative-Distribution-based Outlier Detection* is a statistics-based method which uses the *empirical cumulative distribution function* (*ECDF*) to approximate the *cumulative distribution function* (*CDF*). This is done with the assumption that the samples of the data that are on the tail-end of the distribution are assumed to be anomalies [21].

A problem with calculating a single *ECDF* for a multivariate dataset is that the more dimensions there are in the data, the slower the *ECDF* converges to the true *CDF* as a direct consequence of the previously mentioned *curse of dimensionality*.

ECOD avoids this problem by making the assumption that the dimensions in the data are independent, which means that it is possible to multiply the separately calculated tail-probabilities with the univariate *ECDF*s [21].

Additionally, in order to determine which tail of the distribution to use for a feature when assuming whether a point is an anomaly or not, *ECOD* uses the *skewness* of the distribution. This means that if one of the tails is more skewed, i.e., one of the tails is longer, then it is assumed that the anomalies occur on that tail of the distribution. An example of this can be seen in figure 3.7 which showcases how the choice of tails affects the result of the anomaly detection. In the distribution shown in plot (a) in figure 3.7 we can see the ground-truth while plots (b) and (c) showcase the results when using the left and right tails respectively. We can see that the distribution is skewed in nature, where the anomalies tend to occur on the left tail of the distribution. When the tails are selected based on the skewness, as can be seen in plot (e) we can see that *ECOD* is more accurately able to discover the existing anomalies.



(a) Ground truth (b) Left tail (c) Right tail (d) Avg of two tails (e) Skewness corrected

Figure 3.7: Plots showcasing how skewness affects the results of *ECOD*. Plot (a) showcases the ground truth, while plot (b) and (c) shows the effect of using the left and right tails respectively. The last two plots (d) and (e) shows the average of the two tails and the skewness corrected result respectively. Source: *Ecod: Unsupervised outlier detection using empirical cumulative distribution functions* [21].

Some advantages of *ECOD* is that it has a low time complexity scaling linearly with the size and dimensions of the dataset, it has interpretable results, while being effective at finding anomalies [21].

3.4.3 Isolation Forest

Isolation forest (IF) is a tree-based unsupervised anomaly detection technique popular for its linear time-complexity, low memory usage and high performance with low hyperparameter configuration [22].

In Isolation Forest, the sub-trees (as described in section 3.3.2) are called Isolation Trees (iTrees). Each iTree is constructed by creating a sub-sample of the data through random selection without replacement. The samples in the iTree are then isolated by recursive random partitions on one randomly selected feature at a time. The anomaly score of each sample is then the path length to the root of the iTree. An ensemble over all isolation trees is what makes up the Isolation Forest (IF). The average path length over all iTrees determines the final anomaly score of each sample. Those samples isolated near the root of multiple iTrees are regarded as anomalies.

The algorithm can be formally explained as follows: given a sub-sample $X = \{x_1, x_2, ..., x_n\}$ from some multivariate distribution; each iTree is constructed through partitioning of X by randomly selecting an attribute q and a split value p. The X sub-sample is then partitioned into a left and right sub-trees (X_l, X_r) using the test $q < p$. This partitioning can be seen as linear axis-parallel cuts in data space, as seen in figure 3.3. The partitioning is recursively performed until either (1) the tree reaches a height limit, (2) all samples have been isolated or (3) all remaining samples are identical. The tree height limit is used to preemptively stop the partitioning with the reasoning that after a certain tree depth all points are regarded as normal and thus not of interest [22].

The limited tree depth is one-key factor to the linear time-complexity of the isolation forest, the second is that the size of the iTrees is bounded by the sub-sampling size. It is shown that keeping the size of the iTrees low not only lowers memory-requirements and increases

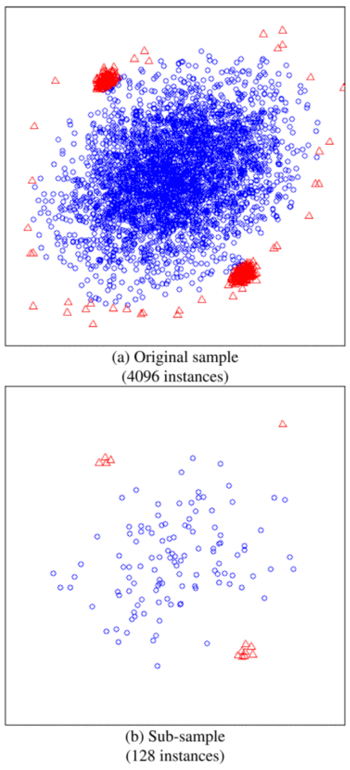


Figure 3.8: An example on generated data of how sub-sampling enhances the possibility to identify deviating samples in the Isolation Forest algorithm. Anomalies are marked by a red triangle and normal samples by a blue circle. By randomly selecting a sub-sample of the data the anomalies are more clearly separated from the normal samples [22]. Image source: *Isolation forest* [22]

efficiency, but it also improves anomaly detection performance due to smaller sample sizes making deviating characteristics more clearly noticeable [22]. This concept is demonstrated in figure 3.8. The two hyperparameters for isolation forest are the sub-sampling size ϵ and the number of iTrees, both of which were shown to converge quickly at low numbers [22].

3.4.4 Deep Isolation Forest

Deep Isolation Forest (DIF) is a modern hybrid method, based on the aforementioned Isolation Forest algorithm [39]. The major difference in DIF is that the cuts in data space are non-linear instead of axis-parallel, aiming to increase its effectiveness at isolating hard to isolate samples in non-linear data. DIF utilises casually induced neural networks to project the selected input features for an iTree (see section 3.4.3) to a randomised non-linear data space as is shown in figure 3.9. Subsequent axis-parallel cuts in this space are equivalent to non-linear cuts in the original data space. If a sample is isolated early in multiple randomly induced sub-spaces it indicates that the sample is a non-linear anomaly. Similarly to the original Isolation Forest algorithm, the anomaly score is given by the average path length over an ensemble of these so called deep iTrees into a Deep Isolation Forest. This means that it is sufficient for an anomalous sample to be isolated in a subset of randomised non-linear data spaces for it to be recognised as an anomaly.

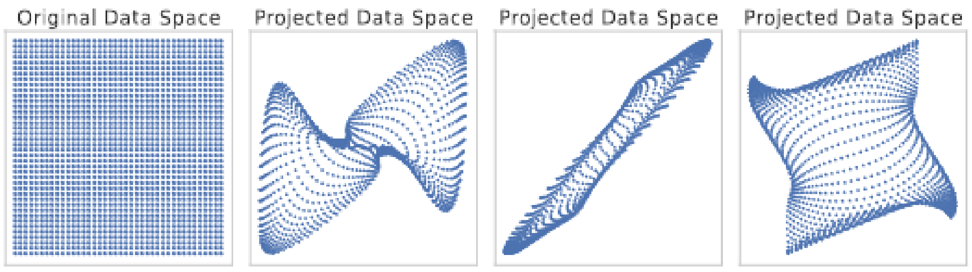


Figure 3.9: Showcase of the data space projection of deep isolation forest from original data space to three randomly induced data spaces. Axis-parallel cuts in the transferred data spaces are equivalent to non-linear cuts in the original data space. Image source: *Deep isolation forest for anomaly detection* [39]

The DIF algorithm effectively attempts to reduce the curse of dimensionality that the original algorithm suffers from. This comes at the price of increased computational complexity due to having to train the neural networks [39].

3.4.5 Autoencoder

An Autoencoder is a type of reconstruction-based method that typically relies on neural networks to find the patterns in the data during deconstruction and reconstruction [41]. An Autoencoder can essentially be thought of as two functions. The encoder function $z = f(x)$ which maps data from data space to the latent space, and then the decoder function $r = g(z)$ which reconstructs its inputs from the latent space back to data space [41]. Generally, the encoder/decoder functions are stochastic in nature and are then defined as $p_{encoder}(z|x)$ and $p_{decoder}(r|z)$ [41].

In figure 3.10 we can see the general structure of a *traditional* Autoencoder which may be described as a densely connected neural network [41]. While it does not necessarily have to be, it is often implemented as a "mirror" architecture, meaning that the decoder layers are a mirror image of the encoder [41]. What separates Autoencoders from other neural network architectures is the fact that the input and output layers have to have the same dimension since we are trying to reconstruct the input signal and the central layer has to consist of less neurons than the outer layers in order to transform the data to latent space [41].

3.4.6 LSTM Autoencoder

The LSTM-Autoencoder falls into the category of reconstruction-based methods since its purpose is to reconstruct a signal. However, its general assumptions and how it makes predictions differs significantly from the pure Autoencoder (see section 3.4.5). The Autoencoder attempts to reconstruct the whole signal at once, while in contrast the LSTM-Autoencoder uses a sliding window transformation (see 3.5) and Long Short-Term Memory units to make

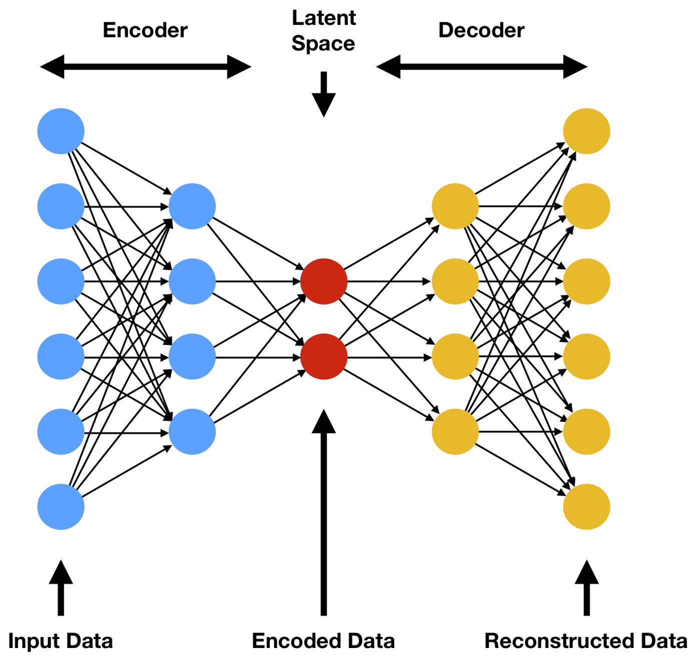


Figure 3.10: Illustration of a densely connected autoencoder architecture. Image source: *Variational autoencoders are beautiful* [13]

predictions for every point based on local patterns in the data [28]. This makes LSTM-Autoencoders especially suited for detecting *contextual* and *collective* anomalies, since they are able to capture temporal dependencies. We believe this property makes LSTM-Autoencoders advantageous for anomaly detection on heat pump data, since the heat pump's behaviour is connected to its temporal properties as discussed in sections 4.3.2 and 4.3.3.

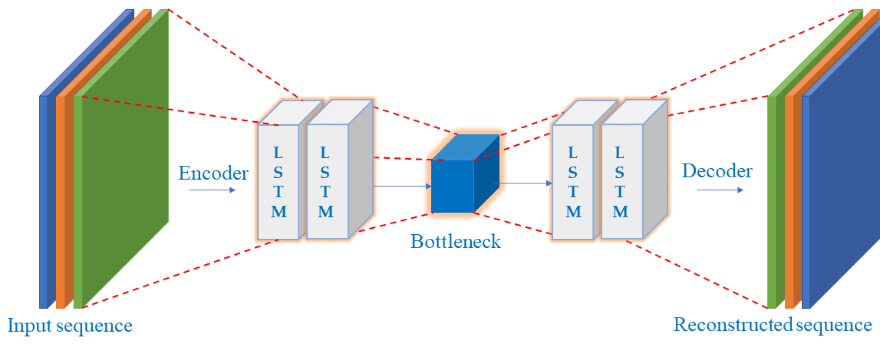


Figure 3.11: Image source: *Lstm-autoencoder for vibration anomaly detection in vertical carousel storage and retrieval system (vcsrs)* [9]

As can be seen in the figure illustrating an LSTM-Autoencoder architecture 3.11, and figure 3.10 showing a *classical* Autoencoder architecture, the two networks appear similar. Where the two architectures differ is that the neurons in a classical Autoencoder are replaced with LSTM units, which are described below.

Long Short-Term Memory (LSTM)

A *Long Short-Term Memory* (LSTM) is a type of improved *Recurrent Neural Network* (RNN) that removes an inherent problem with RNNs, the so called *vanishing/exploding* gradient problem [28]. RNNs work by processing arbitrary length input sequences through a recursive process by activating a transition function on a hidden state vector h_t. The inputs to the h_t for each time-step t are the outputs of the previous layer h_{t-1} and the feature vector x_t. The inputs are then multiplied by two weight matrices, W_x and W_h according to the following function [28]:

$$h_t = tanh(W_x X_t + W_h h_{t-1} + b)$$

In the training of a RNN the W_x and W_h weights are applied at each time-step, therefore the resulting outputs run a risk of having exponential growth/decay over time [28]. LSTMs solve this vanishing/exploding gradient problem by introducing a memory cell which preserves cell states by changing in response to inputs which results in preservation of cell state over long periods [28].

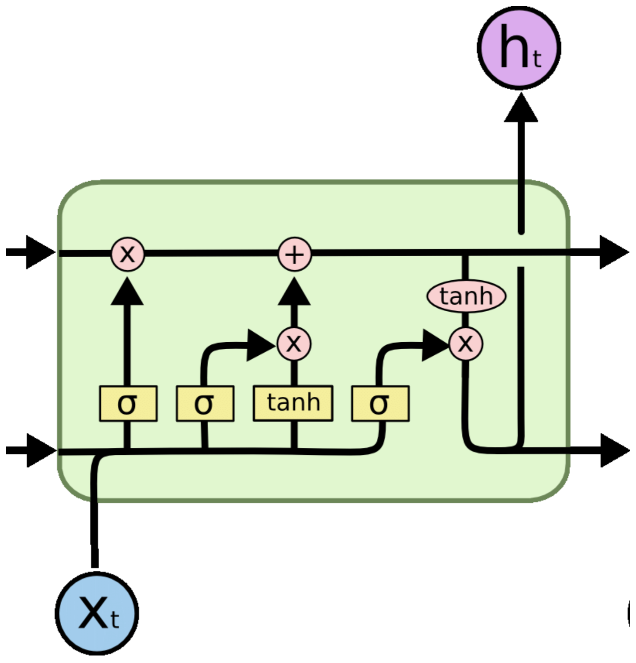


Figure 3.12: Example of an LSTM Unit. Source: *End-to-end radio traffic sequence recognition with recurrent neural networks* [28]

An LSTM-unit is illustrated in figure 3.12. The unit consists of an input gate i_t, an output gate o_t, a forget gate f_t, a memory cell state c_t and lastly a hidden state h_t. The memory cell c_t and the hidden state h_t are the parts of the unit responsible for the long-term dependencies, while the other ones are responsible for the short-term dependencies which is why the unit is called long-short term memory [28]. An LSTM network manages the vanishing/exploding gradient problem by either blocking or passing on the input. Each unit also has its own weights, which are applied to the input.

3.5 Motivation for Choice of Methods

As mentioned in section 3.2, according to the *no free lunch theorem* there is no universal method that can solve every anomaly detection problem. When exploring the data set (see chapter 4.3) where/how the registered errors for the heat pumps tend to occur, the errors that do occur do not follow a consistent behaviour. Some errors seem to be point anomalies that are registered as errors due to spikes in the sensor data, while others appear to be contextual anomalies and are only classified as errors due to local spikes.

Heat pumps are complex machines in the sense that they do not necessarily have just one *normal* state, but rather multiple ones depending on how they are being used. For example if a hot shower is turned on, the heat pump will start consuming more energy since heating water is a costly operation. Thus many of the features can spike during this process, but this does not necessarily mean that the heat pump is malfunctioning.

This is the main motivation for the utilisation of the density-based methods (see 3.3.1). This is because the density-based methods define anomalies as deviating from clusters of high density, and thus these methods may be able to find these *normal* behaviours as distinct clusters while the anomalies may just be anomalies relative to the state of the machine.

However, the existing errors may in some cases not be anomalous relative to a particular state, but rather to the whole system. These errors may be described as *global* errors. These point-wise errors are most likely more easily detected using a tree-based approach like Isolation Forest or a statistical approach like ECOD, since global anomalies are by definition well isolated from normal data points.

While these methods tend to perform well for linear systems, they tend to miss more complex relations between different features [2]. This is where reconstruction-based methods tend to perform better because the encoding and decoding procedure is inherently non-linear (at least in an Autoencoder), and thus they are able to capture more complex patterns. This also speaks for the use of Deep Isolation Forest which attempts to capture the non-linear relationships in the data.

One disadvantage with the *normal* reconstruction-based methods is that they do not consider temporal patterns in the data. These patterns such as yearly weather fluctuations exhibit patterns in the data that the *normal* reconstruction-based methods are not able to catch. Time series-based methods such as LSTM-Autoencoders (see 3.3.5), take previous points into account when making predictions. This fact makes it so that time series-based methods have an advantage when attempting to detect contextual and collective anomalies that are local in nature.

3.6 Anomaly Clustering

When performing anomaly detection, the task is typically categorised as a one-class classification problem, namely classifying normal and anomalous points. This means that the result is binary, with normal points classified as a 0 and and anomalous points as a 1. However, in complex machines such as heat pumps, binary classification does not capture the variation in different issues that may cause the machines to express faulty behaviour. A way to circumvent this problem and be able to categorise the caught anomalies is to use clustering methods, i.e., methods that group similar data points based on their inherent characteristics.

The method that was chosen to perform the anomaly clustering is called Density Based Spatial Clustering of Applications with Noise (DBSCAN), which is an unsupervised density-based clustering technique used to classify data by identifying clusters of arbitrary shape [12]. It clusters samples based on their proximity to other points in a high-density area, with the assumption that high-density areas are separated by low-density areas [2].

DBSCAN uses two hyperparameters; the distance to other samples for clusters ϵ, and the minimum number of points in the cluster M. The ϵ-neighbourhood of a sample x_i are all the samples within ϵ distance of x_i. Formally that is: $N_\epsilon(x_i) = \{x \in D \mid \text{dist}(x_i, x) \leq \epsilon\}$ [2], where *dist(a,b)* is a distance function between samples *(a,b)*. The size of the clusters is decided by varying ϵ and M. Using ϵ and M, DBSCAN clusters the samples by marking each sample as either a *core point*, *boundary point* or *outlier* in the following steps: (1) for each sample identify all points in the ϵ-neighbourhood of that sample and mark the ones with minimum M neighbours within ϵ radius as core points; (2) create a new cluster for every core point if it is not already associated with a cluster; (3) recursively iterate through each cluster and assign all point within ϵ distance to the cluster, those points with less than M neighbours within ϵ are marked as boundary points; (4) iterate through all remaining points [2]. Outliers are those points not belonging to any cluster after the process has finished. A visual example of a DBSCAN clustering can be seen in figure 3.13. In the right plot of the figure the large coloured circles are core points, the smaller coloured circles are boundary points and the small black circles outside the clusters are anomalies.

DBSCAN is popular for its ability to handle relatively large sets of non-linear data while requiring little domain knowledge [2].

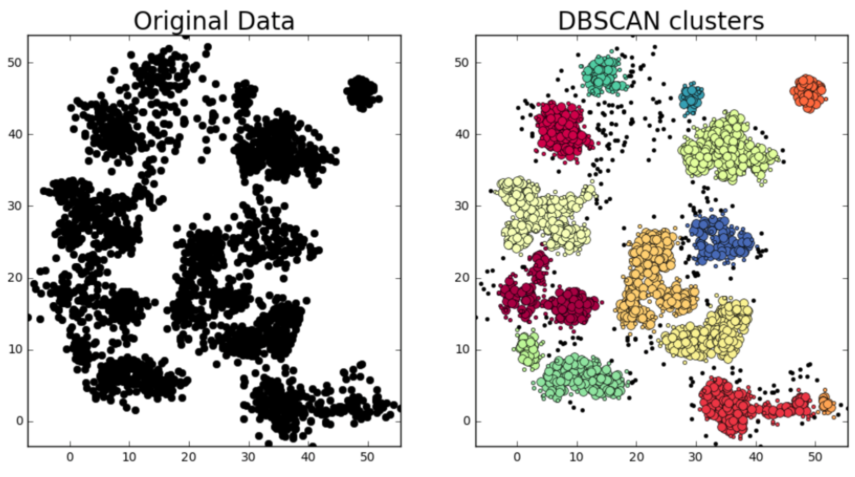


Figure 3.13: An example of DBSCAN clustering. Identified clusters are separated by colour. Large coloured circles represent core points, small coloured circles represent boundary points and small black circles are outliers/anomalies. Image source: *DBSCAN in Python* [11]

3.7 Evaluation Metrics

In this section, we will present the different metrics that we utilised to evaluate the results of our models. Different evaluation metrics have different strengths and weaknesses and thus it is important to evaluate the results of a classifier using different methods in order to give a nuanced perspective on the performance of the trained models. Thus we will in this section also present the strengths and weaknesses of different evaluation methods. The importance of a specific metric is also heavily influenced by the subject matter. In the instance of unsupervised MTSAD algorithms, metrics are mainly valued on their ability to detect complex anomalies in multi-dimensional, co-dependent data while being resistant to noise [2].

3.7.1 Classification Terminology

When evaluating anomaly detection predictions, the predictions can be thought of as a binary classification problem where anomalies are represented by a 1 or true, and normal points are represented by a 0 or false. In order to appropriately evaluate the results it is important to not only look at the hit-rate (how many predicted anomalies that correspond to actual anomalies) but also how many points were incorrectly classified. A resulting prediction in a binary classification problem can be split into four categories; true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN). TP corresponds to the points predicted as anomalies that correspond to the ground truth, while TN are points that were classified as normal points and are labeled as normal points. The TP and TN can be interpreted as the correctly classified samples. Respectively, FP are points that were classified as anomalous but are labeled as normal points and FN are points that were not classified as anomalous but are labeled as such in the ground truth. The main diagonal in the confusion matrix (see 3.14) are thus the correctly predicted points. We will now proceed to explain different methods which utilise the TP, TN, FP, FN values to evaluate the performance of a binary classification model.

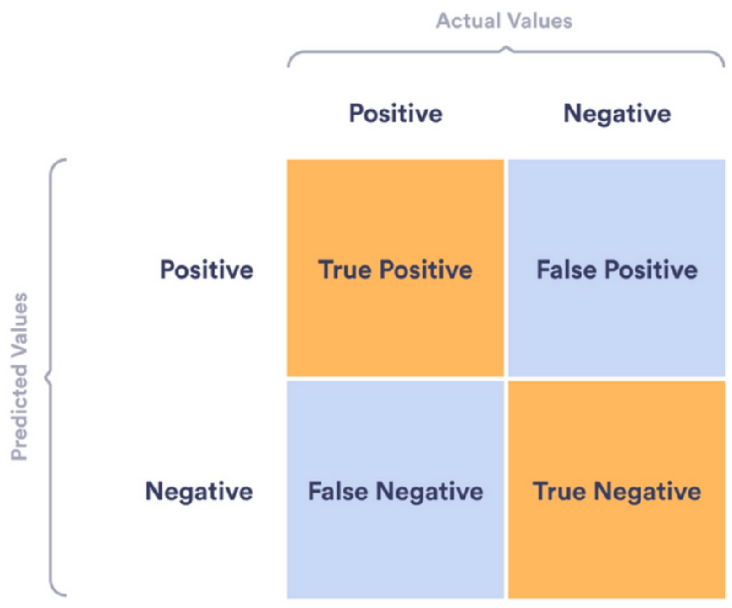


Figure 3.14: Confusion Matrix. Image source: *What is a confusion matrix in machine learning?* [34]

3.7.2 Evaluating Anomaly Detection

In this section, the metrics that were used to evaluate the performance of the anomaly detection models are presented.

Accuracy

The accuracy of a models performance, or ACC, is a metric of the rate of correct classifications, which is calculated by dividing the correct predictions by the total predictions. The accuracy of a model is defined as follows:

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

Precision

Precision can be described as how accurately a machine learning model was able to classify the positive instances (TP and FP) in the data set, meaning that it is a metric of how many predicted positive instances are relevant. The precision of a model is defined as follows:

$$Precision = \frac{TP}{TP + FP}$$

Recall

Recall is a metric which measures the ability of the model to find all of the instances of the positive class. The recall is calculated as follows:

$$Recall = \frac{TP}{TP + FN}$$

F1-Score

The F1-score is a combination of the precision and recall metrics, by taking the harmonic mean of the two. F1-scores are valuable when evaluating classifier performance because the assessment takes both false positives and false negatives into account. There are two distinct types of F1-scores which have different strengths called micro F1 and macro F1. Micro F1 is defined as follows:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

For binary classification problems such as anomaly detection micro F1 becomes a similar metric to *accuracy* and thus we will focus on the macro F1 score. Macro F1-scores can be calculated as follows:

$$\text{Macro F1} = \frac{1}{N}\sum_{i=1}^{N} F1_i$$

MCC-Score

MCC-Score (Matthews Correlation Coefficient-Score) is another metric used for evaluating the performance of machine learning models. The main strength of the MCC-score metric is that it only produces a high score if the model has a high accuracy for all classes, namely normal and anomaly points respectively in a binary anomaly detection problem [6]. A common criticism of F1-scores, which is the most commonly used evaluation metric, is that the rate of correctly classified negative samples are not taken into account which is a problem that mcc-score solves [6]. The MCC-score of a model is calculated as follows:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

3.7.3 Evaluating Anomaly Clustering

In this section, the metrics that were used to evaluate the performance of the anomaly clustering are presented.

Adjusted Rand Index Score

Adjusted Rand Index (*ARI*) is a commonly employed evaluation metric used in clustering analysis to determine agreement between partitions of data [40]. Given the partitions u and v, let n_{ij} correspond to the number of objects that overlap between class u_i and cluster v_j. Furthermore, let n_i and n_j correspond to the number of objects that are in class u_i and cluster v_j. These definitions are illustrated in table 3.1.

Class \Cluster	v_1	v_2	$\dots$	v_C	Sums
u_1	n_{11}	n_{12}	$\dots$	n_{1C}	$n_{1\cdot}$
u_2	n_{21}	n_{22}	$\dots$	n_{2C}	$n_{2\cdot}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
u_R	n_{R1}	n_{R2}	$\dots$	n_{RC}	$n_{m\cdot}$
Sums	$n_{\cdot 1}$	$n_{\cdot 2}$	$\dots$	$n_{\cdot C}$	$n_{\cdot\cdot} = n$

Table 3.1: Contingency table for comparing R classes with C clusters

Using these definitions the *ARI* score can then be defined as follows:

$$ARI = \frac{\sum_{i,j}\binom{n_{ij}}{2} - \left[\sum_i \binom{n_i}{2} \sum_j \binom{n_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_i}{2} + \sum_j \binom{n_j}{2}\right] - \left[\sum_i \binom{n_i}{2} \sum_j \binom{n_j}{2}\right]/\binom{n}{2}}$$

The *ARI* score scales between -1 and 1, where -1 means no agreement between the clusters and the ground-truth, while 1 means that the two partitions are in perfect agreement [40]. The *ARI* score measures the statistical similarity between the partitions. A strength of the *ARI* score, is that the score is adjusted based on the possibility that there is a match due to random chance [40].

Normalized Mutual Information Score

Another popular metric for evaluating the agreement between data partitions is the so called *Normalized Mutual Information* (*NMI*) score. The *NMI* score is a measure of how much mutual information there is between the partitions, normalized by the label entropy. Using the aforementioned definitions, *NMI* can be defined as follows:

$$\mathrm{NMI}(u, v) = \frac{-2 \sum_{i=1}^{R} \sum_{j=1}^{S} n_{ij} log(n_{ij} n / n_{i\cdot} n_{j\cdot})}{\sum_{i=1}^{R} n_{i\cdot} log(n_{i\cdot}/n) + \sum_{j=1}^{s} n_{\cdot j} log(n_{\cdot j}/n)}$$

Chapter 4
Approach

This chapter introduces the methodology applied throughout the duration of this project, spanning from the initial phases to the final outcomes. The first section presents an overview of the steps performed in the anomaly detection process. The following section outlines the application of the CRISP-DM (Cross-Industry Standard Process for Data Mining) framework, elaborating on the execution of each phase within the framework. Furthermore, an overview is provided of the external libraries utilised throughout the duration of the development process.

4.1 Method Overview

Here we present a process diagram of every step performed, starting with loading the data and ending with the final evaluation process. The step-by-step process can be seen below. For a process diagram of the anomaly detection procedure, see figure 4.1. A more detailed description of each performed step can be seen in the following section 4.2.

1. *Data Interpolation & Aggregation*: The process starts with the initial data loading and the preprocessing steps. The preprocessing included forward filling the missing values and then aggregating over a set time span.

2. *Data Projection (optional)*: Dimensionality reduction is done by performing a UMAP projection to a lower-dimensional feature space.

3. *Feature Selection (optional)*: Dimensionality reduction is done by performing a feature selection using sequential forward selection.

4. *Choice of Anomaly Detection Model:* Here, a choice of anomaly detection model is done, choosing between Isolation Forest, Deep Isolation Forest, LOF, DBSCAN, Autoencoder, and LSTM-Autoencoder.

5. *Choice of Normalisation Method:* Here, depending on whether the anomaly detection method requires it, a normalisation method is chosen. The choice is between z-score and min-max normalisation.

6. *Execution and Evaluation*: Lastly the anomaly detection method is trained on the data and then the results are evaluated.

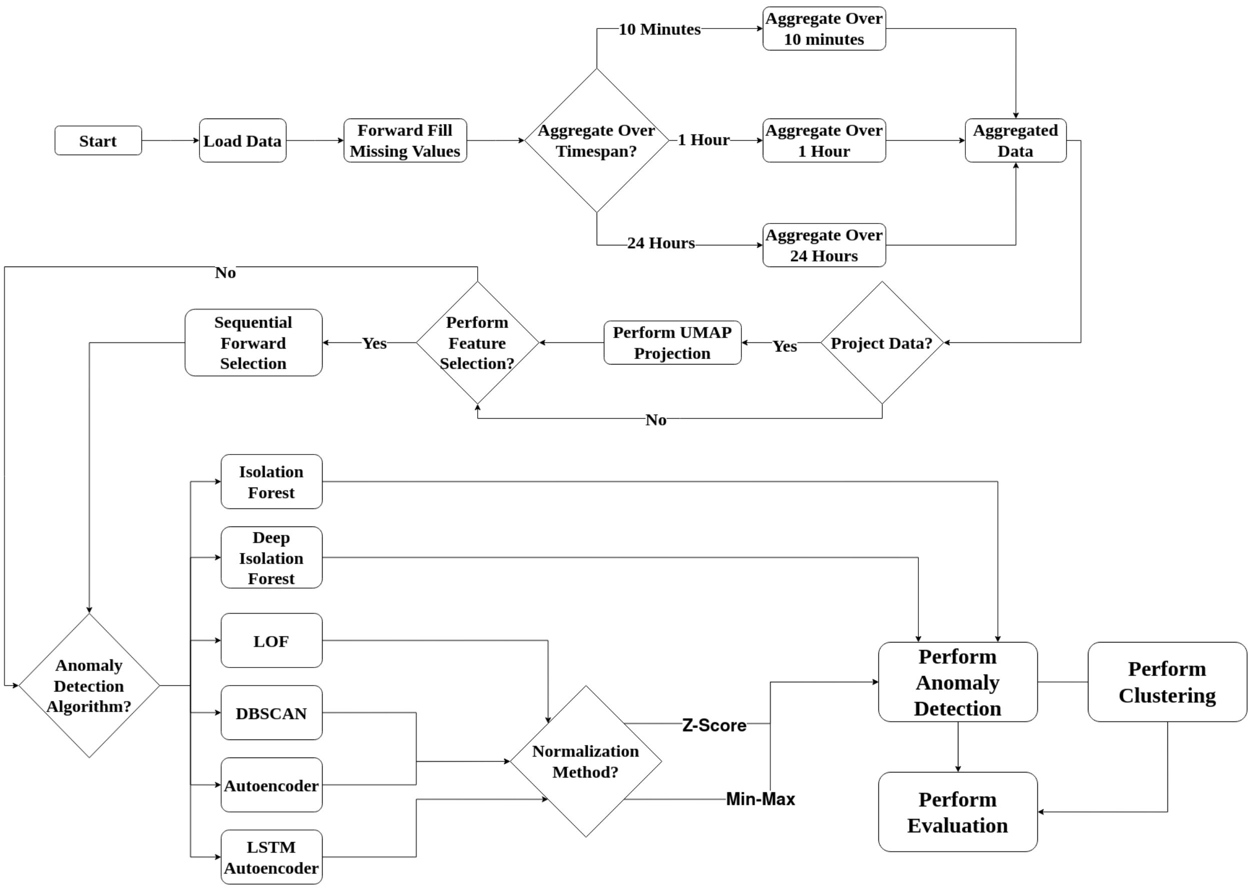


Figure 4.1: Process diagram showcasing the anomaly detection process, from the initial data loading to the final evaluation step.

4.2 The CRISP-DM Process

This thesis was conducted using the industry standard *Cross-Industry Standard Process for Data Mining* (CRISP-DM) process, which is a commonly employed framework for approaching data mining projects [37]. An overview of the CRISP-DM process can be seen in figure 4.2. This process is most commonly split into 6 steps, which are are as follows:

1. *Business Understanding*

   The initial step in the CRISP-DM process is focused on understanding the business use case, defining the requirements and objectives of the business in order to create a project plan.

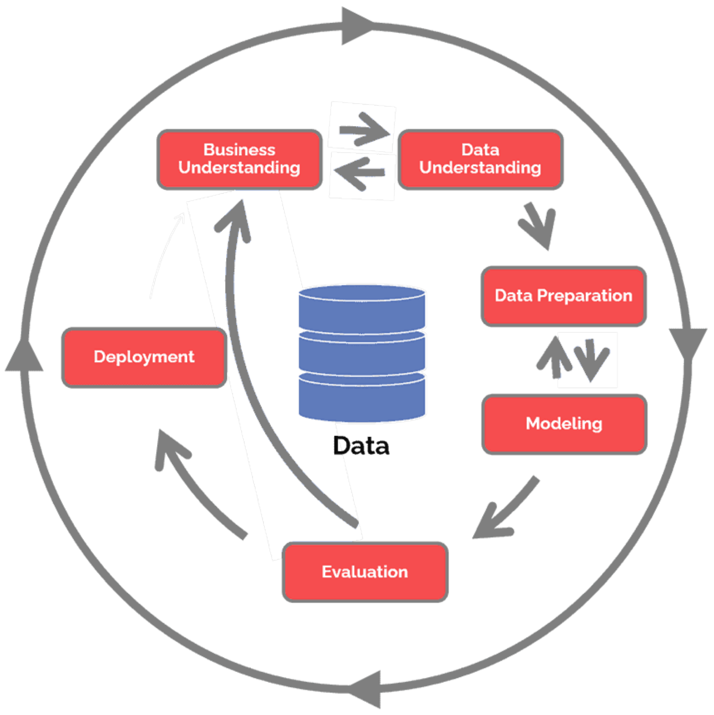


Figure 4.2: Model of the CRISP-DM process, showcasing the phases and life cycle of a data-mining project with arrows indicating phase dependencies. Source: *What is CRISP DM?* [17]

2. *Data Understanding*

   The next phase is exploring the data in order to understand the limitations and possibilities that are inherent to the data set. This is done to gain an understanding of the data, and which steps are required in order to achieve the goal of the project.

3. *Data Preparation*

   Data preparation is where the data is prepared for the forthcoming phases, which can include multiple techniques (e.g., feature engineering, feature selection, and dimensionality reduction).

4. *Modeling*

   In this phase, multiple models and parameters of the models are tested in order to find optimal configurations. In reference to this thesis, the models in question refer to the machine learning models utilised in the anomaly detection process.

5. *Evaluation*

   Systematic evaluation of the performance of the aforementioned models, in order to determine whether or not the model(s) satisfy the project requirements.

6. *Deployment*

   Lastly, the model(s) are deployed. This final phase includes monitoring and maintenance of the model(s).

This project follows steps 2-5 in the CRISP-DM model, omitting the *business understanding* and *deployment* steps as they are outside the scope.

4.3 Data Understanding

In this section, we will cover the second phase of the CRISP-DM process, labeled *data understanding*. The utilised data sets are introduced and explored, with a focus on the general dimensionality, frequency and inherent characteristics of the features. These insights guide the next phase of the CRISP-DM process, namely *data preparation*.

4.3.1 Heat Pump Industrial Metrics

The dataset designated for the anomaly detection process was provided by Robert Bosch AB. It encompasses time-series industrial metrics for heat pumps of multiple models/manufacturers that are distributed over multiple European countries. In this chapter and onwards the terms *column* and *feature* will be used interchangeably depending on the context to describe the columns in the data.

The data consists of roughly 200 - 300 different columns per heat pump system. New observations are continuously reported at a frequency of ~6 times per minute with the changes that have occurred in the system. If no changes have occurred for a feature then it is reported as an empty value. The reported observation may then be interpreted as a sparse vector mostly consisting of empty values, which by extension means that the majority of the data in the dataset is empty.

The quantity of data varies between each heat pump, with a spread of between ~100 and ~450 days of reported data. Additionally, the heat models do not have the exact same features, having only about 50% of the features overlapping between all models.

The quantity of data varies between each heat pump. The heat pumps have a variance in different features, with about 50% of the features overlapping between all models. Furthermore, the length of gathered data for every heat pump varies between ~100-450 days. This means that the data for each heat pump consists of between ~700 thousand and ~3 million observations.

An overview of the different types of features that exist in the data is presented below:

1. *Categorical:*
   (a) *Binary Values:* Signal a state shift such as On/Off or Yes/No or 0/1
   (b) *Tertiary Values:* Signal a state shift when there are 3 machine states.
   (c) *Context:* Static information about the heat pump that "never" change, such as which components are part of the heat pump.
2. *Numerical*
   (a) *Momentary Value:* Signals a recorded momentary value from for example a sensor in the system.
   (b) *Accumulated Values:* Records accumulated values over time, for example total energy consumption of the system or a component.

Since anomaly detection requires that the anomalies are a minority of the existing data points (otherwise the anomalous behaviour becomes the normal behaviour), we chose to focus on heat pumps that have a contamination rate lower than 50%. The contamination rate, here and henceforth, refers to the proportion of total points that are registered as errors by the machines. We also chose to focus on heat pumps that have at least 400 days of data, which left 21 heat pumps systems to be used for the study. An overview of the chosen heat pumps and their respective contamination rates can be seen in table 4.1. Do note that the contamination is split into a left column signifying the total contamination for that system over the entire registered timespan, while the right column shows the contamination of the final 20% of the timespan used as validation data set for the semisupervised models.

The reason for choosing the final 20% of the data as validation rather than a random subset is twofold. Firstly, the first 80% of the data corresponds to approximately a full year (∼ 360 days), meaning that the semisupervised models can learn from a full season of data. Secondly, this results in the fact that the time-series is continuous in both the training and validation sets. This came at the expense of that a minority of the heat pumps lacked contamination in the validation sets, and thus could not be evaluated.

The data from each heat pump was then aggregated to discrete 1 hour time intervals, which reduced the number of samples to ∼10000 per heat pump system. A sample was defined as a ground-truth error if it contained any reported error codes within the 1 hour timespan, and multiple errors could be reported during each discrete time interval.

4.3.2 Seasonal Behaviour in the Heat Pumps

In order to be able to understand the inherent seasonality of the heat pumps, here we will showcase 1-dimensional PCA projections plotted against the registered timestamp for each datapoint. As can be seen in figure 4.3 the plots **(a)** and **(b)** seem to exhibit a sinusoidal cyclic behaviour and this behaviour becomes less prominent as the contamination rate increases. In plot **(c)** in figure 4.3 it can be seen that the seasonality in the data seems to deteriorate with large contamination rates. Note: seasonality here implies regression seasonality [18], as in periodic behavior in the data.

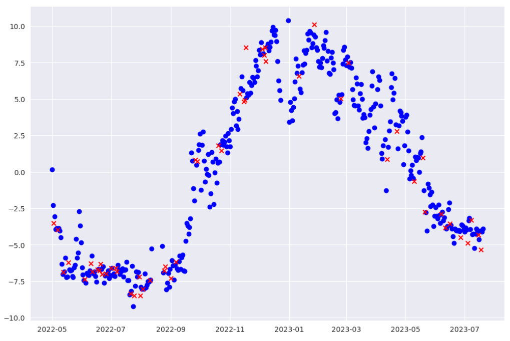
This yearly seasonality can be explained by the fact that heat pumps are inherently contextual machines meaning that their behaviour is correlated with weather conditions. When it is cold outside, the machines will need to produce more heat and vice versa and this behaviour can be seen in figure 4.3. This implies that in order to minimise the effect that the seasonality has on the anomaly prediction this variance needs to be removed. However, as previously mentioned, the data set only consists of up to ∼450 days, i.e., only one full season, which is not enough to use traditional seasonality removal techniques (see 2.4.1 for more information).

However, heat pumps do not only exhibit a yearly seasonality, but also a daily seasonality. Depending on the time of day, the heating needs differ since the temperature changes in a 24 hour time span are not static, with it being usually colder during the night and warmer during the day.

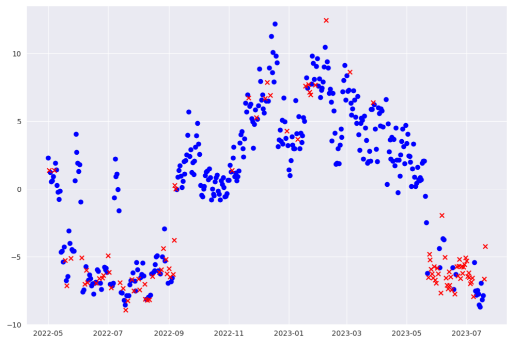
In figure 4.4 this daily seasonality behaviour can be seen, and that it differs highly between different seasons. During January, we can see that the frequency of the PCA plot is higher, having around 4 peaks per day, while during June the projection has around 2-3 peaks. We can also see that in January, the PCA vector tends to have lower values than during June.

	Contamination	
Id	*Total*	*Validation*
H1	2.1%	1.0%
H2	1.4%	0.0%
H3	2.2%	0.0%
H4	1.4%	0.1%
H5	3.0%	7.5%
H6	1.0%	4.8%
H7	1.9%	2.7%
H8	2.7%	0.1%
H9	2.6%	0.0%
H10	4.8%	1.3%
H11	5.4%	0.0%
H12	6.2%	0.3%
H13	4.7%	1.0%
H14	9.0%	15.5%
H15	9.0%	38.2%
H16	6.3%	10.7%
H17	6.6%	0.0%
H18	10.1%	9.7%
H19	15.3%	0.0%
H20	25.9%	96.4%
H23	37.5%	34.2%
Average	7.6%	10.6%

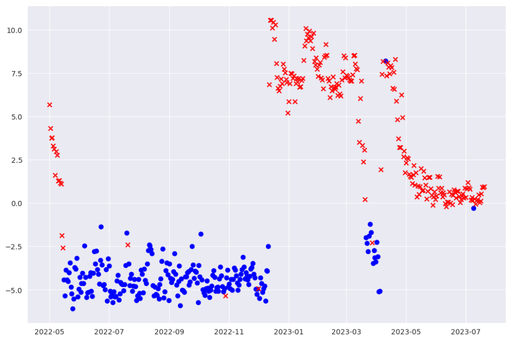
Table 4.1: Contamination rates of the data from each heat pump in the study, based on 1H window aggregates. Left column shows contamination for the entire dataset. Right column shows the contamination of the last 20% of the dataset. The validation set is used to evaluate the semi-supervised anomaly detection algorithms.



(a) Time-series PCA plot of heat pump with 10% contamination rate



(b) Time-series PCA plot of heat pump with 24% contamination rate



(c) Time-series PCA plot of heat pump with 52% contamination rate

Figure 4.3: Time-series PCA plot of three heat pumps with different contamination rates. Blue dots are normal points while the red ones are reported errors.

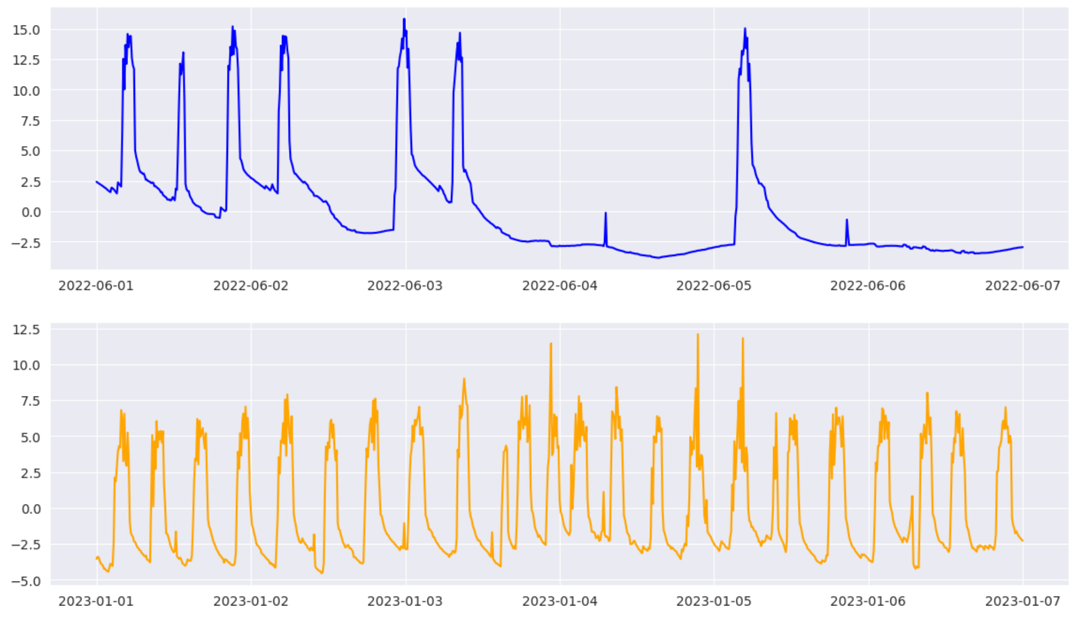


Figure 4.4: PCA plot for one heat pump for two separate weeks during summer and winter.

This provides further complexity when analysing the seasonality and performing anomaly detection on the heat pump data since the seasonality seems to be highly dependent on the context, i.e the temperature and the heating needs. We can also see that during June, this specific heat pump seems to have close to stationary behaviour during some days. This is reasonable considering that most likely the heat pump does not need to heat a facility during the summer months.

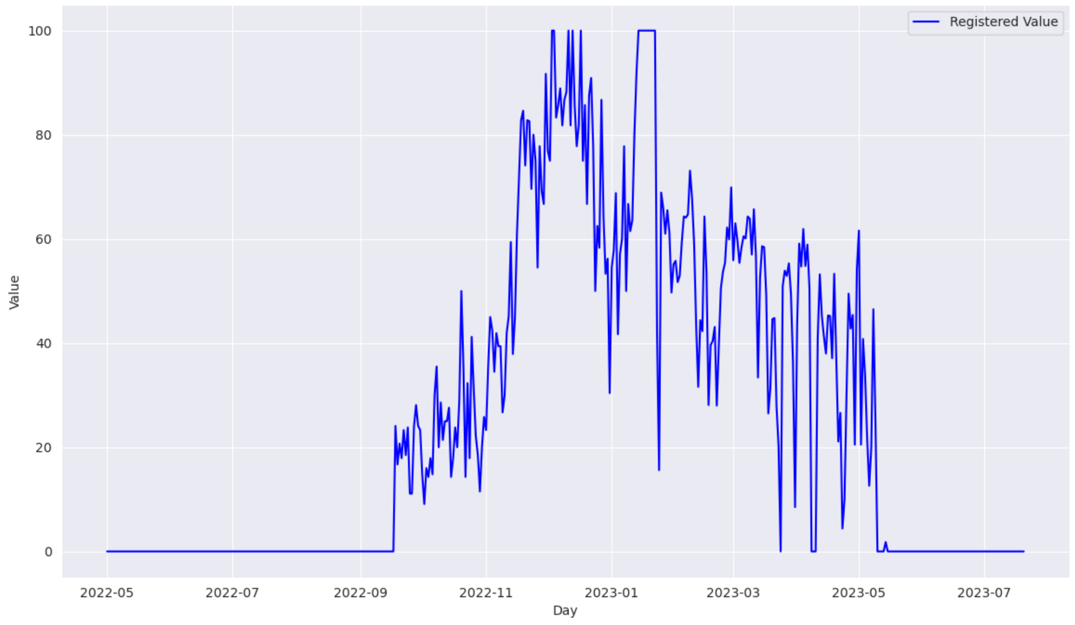


Figure 4.5: Plot showing the registered values for a single feature that shows semi-seasonal behaviour.

Another layer of added complexity is the fact that we do not only have seasonal behaviours in the data, but also *semi-seasonal* features. In the context of heat pump data *semi-seasonal* refers to features that seem to correlate with the outside temperature when it is either hot or cold outside, but not generally across the whole yearly season. This behaviour may be seen in 4.5, where we can see a feature that is stationary up until September, and then spikes up until May with registered peak values around December to January. This behaviour is reasonable considering that the heating needs are higher during the winter months and lower during the summer months. The semi-seasonal features make anomaly detection more complex since, not only, do we have seasonal behaviours in some features but also inconsistent seasonality between different features.

4.3.3 Stationarity of Features

Here we will explore a subset of the heat pump features in order to show that the stationarity is not consistent, i.e., some features exhibit stationary behaviour, while other features tend to follow seasonal trends and exhibit change-point behaviours.

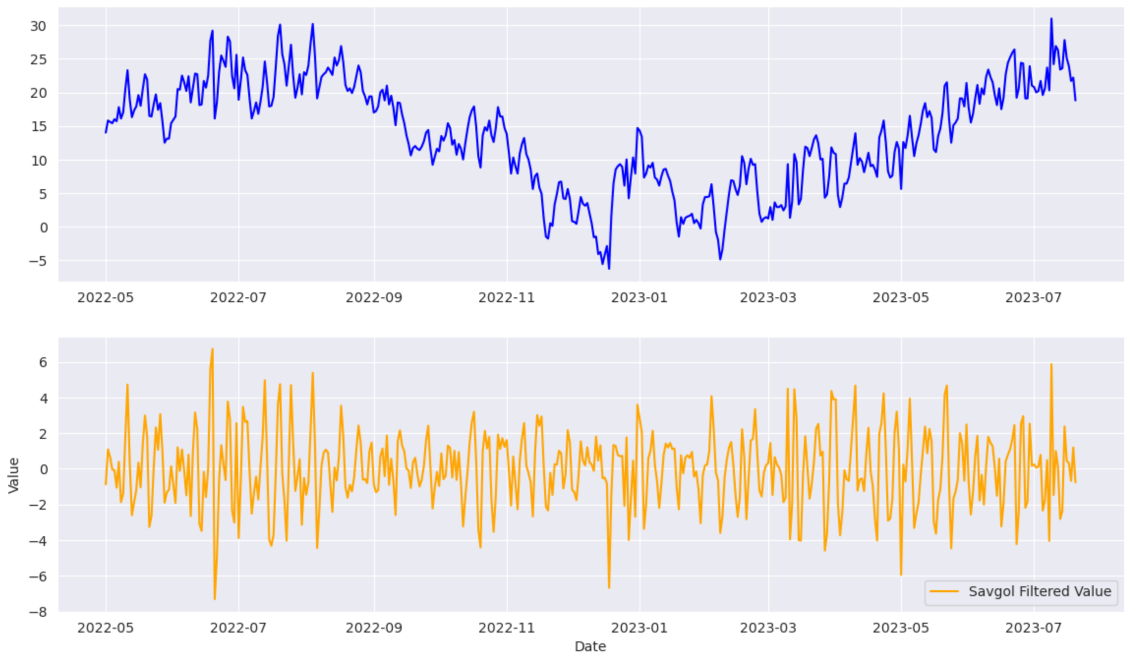


Figure 4.6: Figure showing two plots for data in a singular heat pump. On the top is the data as registered by the heat pump, while on the bottom is the same plot with the seasonal trend removed by subtracting the sav-gol filtered signal.

As we can see in figure 4.6, for this singular feature the data has a clear seasonal trend. The feature generally seems to register high values during the summer months which seem to peak around August, and registers lower values during the winter months with the lowest values registered around December.

In figure 4.7 we can however see the plot for another feature, where no clear seasonality can be observed at least visually. In figure 6.2 we can observe two different features for the

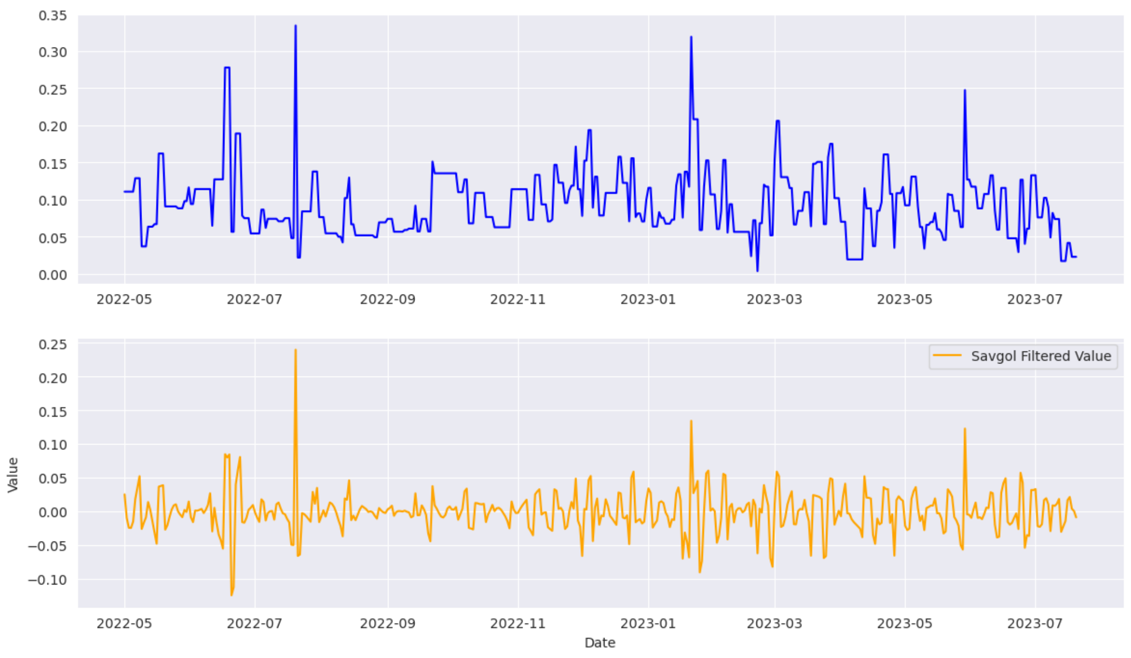


Figure 4.7: Figure showing two plots for a single feature from a heat pump that does not seem to exhibit seasonal behaviour. The top plot showcases the signal as registered by the heat pump, while the bottom plot showcases the same signal where the Savitzky-Golay filtered trend has been subtracted.

same heat pump system both exhibiting a clear *change point* in the final two months of the plot. This change point instance also directly correlated to registered errors as can be seen in figure 6.1.

These observations showcase that we can not regard all features in the same way since they inherently have different behaviours which in turn leads to added complexity to the feature engineering stage of the anomaly detection pipeline.

4.3.4 Identifying Error Behaviours

In this section, we will attempt to demonstrate the patterns that were observed about the error occurrences, in order to be able to explain the patterns that were discovered and that were then used in an attempt to enhance the models. In figure 4.8 it is possible to see a plot showcasing a single feature with the Savitzky-Golay smoothed curve subtracted. For this specific feature, we were able to observe that there seems to be a correlation between noise in the signal and where the error tends to occur. Important to note is that it is important to be mindful that correlation does not necessarily imply causation, especially since we have no way to verify if this is the feature that causes the error.

A source of added complexity when performing anomaly detection specifically on this heat pump dataset and other similarly complex systems, is that the errors may exhibit different behaviours for different features. This can be seen in figure 4.9. Here we have a heat pump with a high contamination rate of 52% and plots showing how this affects 3 different features. In plot **(a)** in figure 4.9 we can see that the errors are collective in nature, this means

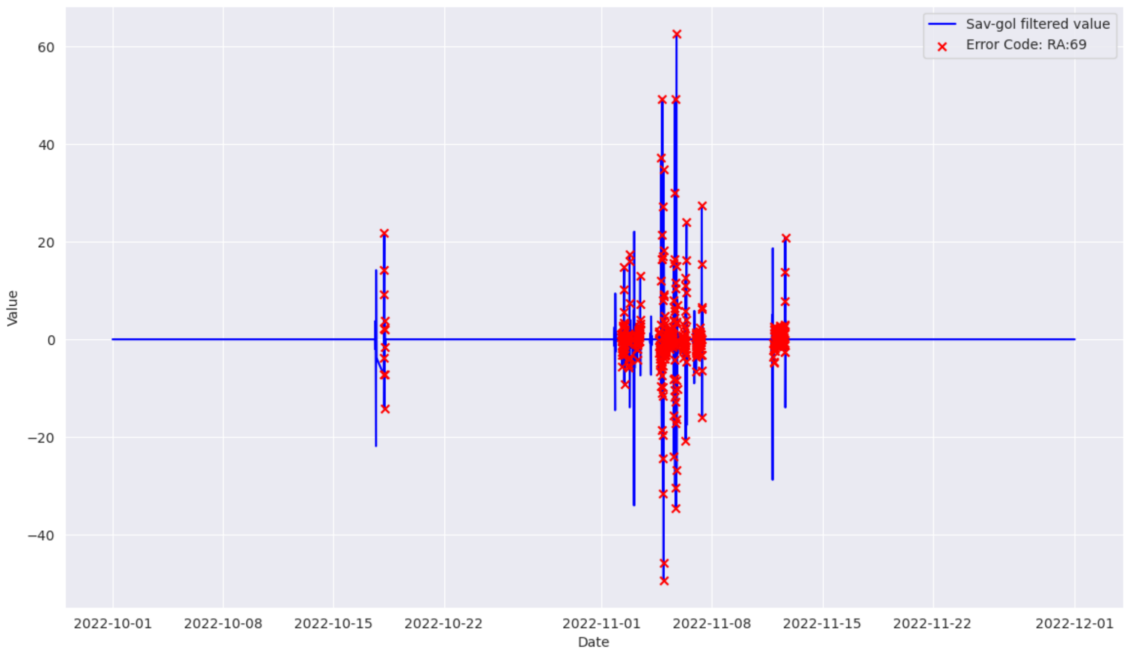


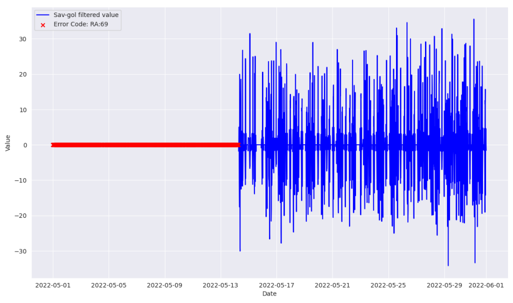
Figure 4.8: Plot of single feature with Savitzky-Golay filtered seasonal trend subtracted, plotted in blue. All the occurrences of errors are marked with red. Plot covers a time period of 2 months.

that the errors occur in a part of the signal that deviates in its behaviour from other parts of the signal. Meanwhile in plot **(b)** in figure 4.9, we can see that for the same time span, the feature signal instead seems to exhibit contextual anomalies.

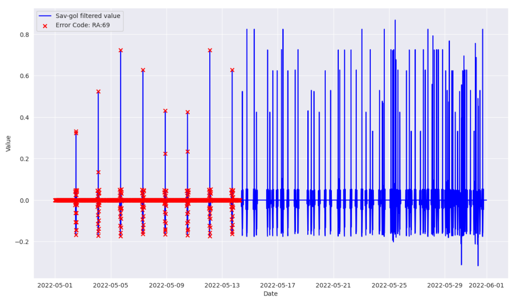
This means that while the signal appears to behave similarly in the whole showcased time span, the part where the errors occur has a lower frequency of peaks than the part of the signal where no errors are registered. Lastly we have another case of point anomalies as can be seen in plot **(c)**, where the errors are registered in points that may be regarded as noise, while the normal behavior for the signal is a flat curve (for a more in depth description of the three aforementioned anomaly types see section 3.1). Again it is important to note that we do not know if the errors are connected to these particular features, but we can still see a correlation since a deviation from the normal behaviour (i.e part of the signal that does not have registered errors) can be observed.

4.3.5 System Wide Patterns

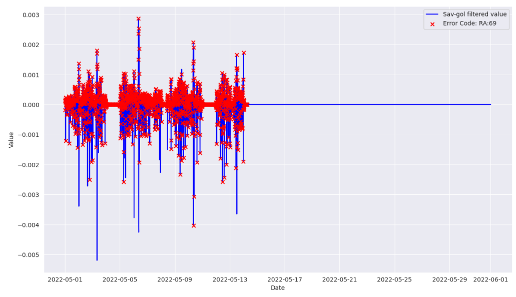
Here we will explore UMAP projections (see 2.3.2) for three different heat pumps, of which the data has varying contamination rates in order to show how/if the errors form distinct clusters. In figure 4.10 we can see three such plots. It seems that when the contamination rate is low as in plot **(a)** in figure 4.10, UMAP is not able to distinctly separate the error points into any clear clusters, and the errors are relatively evenly distributed across all groups. When observing plot **(b)** in figure 4.10 however, we are starting to see a more distinct pattern where a majority of the error points seem to cluster together on the left side of the plot. Lastly, when looking at plot **(c)** in figure 4.10, we can see 6 distinct clusters forming, each with a clear separation between error and normal points. This seems to imply that the higher the con-



(a) Showcases feature exhibiting errors as collective anomalies.



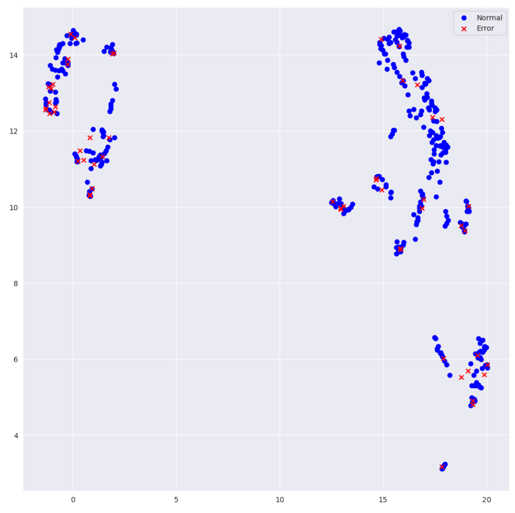
(b) Showcases feature exhibiting errors as context anomalies.



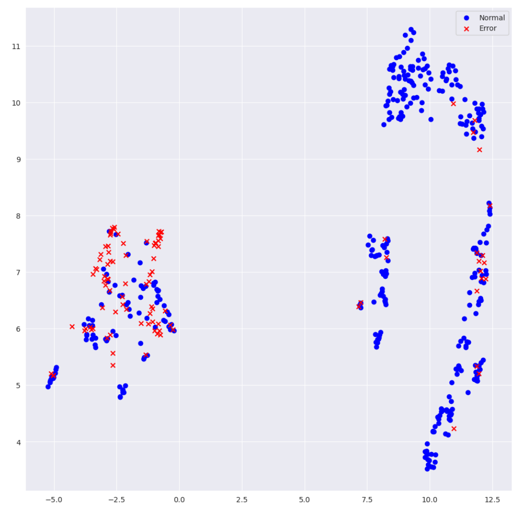
(c) Showcases feature exhibiting errors as point anomalies.

Figure 4.9: 3 plots showcasing how the same error can exhibit different behaviours for different features over a time period of 1 month. The signal is in blue while the red points are registered errors.

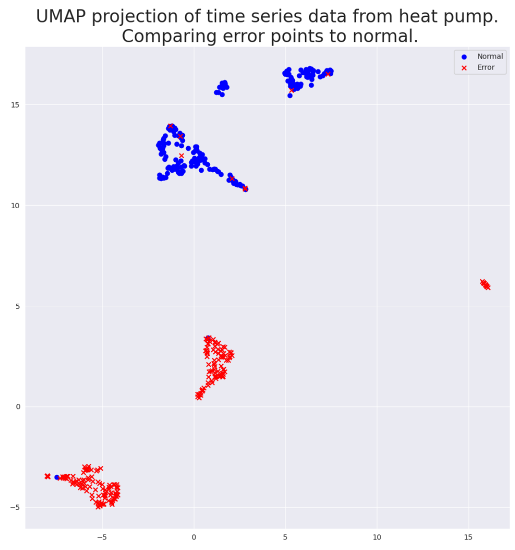
tamination rate, the more accurately the UMAP projection is able to separate normal from anomalous points. It is worthy to note however, that it is possible that the UMAP projection is not able to separate the points for the heat-pump datasets with lower contamination rates due to the fact that the UMAP projection is only two-dimensional.



(a) UMAP projection of heat pump with 10% contamination rate



(b) UMAP projection of heat pump with 24% contamination rate



(c) UMAP projection of heat pump with 52% contamination rate

Figure 4.10: 2-dimensional UMAP Projection of normal and errors points for three heat pumps of varying contamination rates. Normal points are shown in blue while error points are shown in red.

4.4 Data Preparation

Here we will present how we performed the third stage of the CRISP-DM process, i.e., data preparation. This includes the feature engineering steps performed, as well as the chosen dimensionality reduction methods.

4.4.1 Data Interpolation & Aggregation

As mentioned in section 4.3.1 when describing the heat pump dataset, each feature is reported as an empty value until a change has occurred, at which point the updated value is reported. This in turn means that the dataset is inherently sparse, as a vast majority of the values are empty. As a value in a column is only updated when a change has occurred, it was deemed suitable to interpolate the data using a forward fill operation. This is done by filling the missing values for each feature with the most recently occurring value. If the first value in the time series for a column was missing it was instead back filled, and set to the next occurring value. Here, multiple different methods for interpolation were attempted, including rolling window, cubic spline, second degree polynomial, and moving average interpolation but the best result results were achieved using forward fill interpolation.

After the data was interpolated an aggregation procedure was performed, where the data was aggregated over the time spans of 24 hours, 1 hour, and 10 minutes. The 24 hour and 10 minute aggregations were mainly performed for data exploration purposes, while the 1 hour aggregated data was used for the anomaly detection. The data aggregation was done by calculating the means over all features for the aforementioned time spans. The main reason for aggregating the data was to both reduce momentary noise but also to increase the computational efficiency.

4.4.2 Dimensionality Reduction

Dimensionality reduction is an important technique in machine learning and anomaly detection techniques since it reduces a major problem in machine learning, namely *the curse of dimensionality* (see section 2.3). Having many features can often lead to added noise, which adds difficulty when attempting to both explore and understand the data. Additionally, a high dimensionality can also lower the ability of anomaly detection models to discriminate anomalous from normal points, since the discriminatory ability of individual features may diminish as the dimensionality increases. Features can also be highly correlated, meaning that multiple features can convey the same information.

In order to perform dimensionality reduction multiple techniques were used. The techniques in question are sequential feature selection (see 2.3.3) as well as feature extraction techniques, namely UMAP (see 2.3.2).

4.4.3 Data Normalisation

Data normalisation is another important step when performing anomaly detection and many other machine learning tasks. The purpose of normalisation, is to guarantee scale consistency, so that all features are of the same scale. This is important, especially for anomaly detection methods that use distance to determine whether or not a point is an anomaly, since features with larger magnitudes can dominate even though the feature in question is not the cause of the anomalous behaviour. This process was not performed on all anomaly detection methods, since especially the tree-based methods build their trees feature-wise, meaning that the scale of one feature has no effect on other features and their contribution to the total anomaly score. The data normalisation methods used here are z-score normalisation as well as min-max normalisation, depending on which is the most appropriate for the used anomaly detection technique. Fur further information see section 2.2 on normalisation methods.

4.4.4 Removing Seasonality

Due to the inherent behaviours in the data as is presented in section 4.3.2, it was important to take seasonality into account when transforming the data. Seasonality can become a nuisance when performing unsupervised anomaly detection since the seasonality changes the inherent behaviour of the data and by extension the machine learning methods. In order to minimize the effect of seasonality on the utilised models the signal processing technique Savitzky–Golay filtering was used with a time window of 10% of the data length and a polynomial degree of three. For further information on the Savitzky-Golay filter and its applicability for this data set see 2.4.1.

4.5 Modeling

Here we will present how the modeling step of the CRISP-DM process was performed. This step consisted of attempting multiple different types and configurations of anomaly detection methods. The different types of anomaly detection methods investigated were tree, density, reconstruction, statistics, as well as time-series based methods. The contamination rate to be predicted was set to the same contamination rate as the ground-truth for each heat pump system. While this is not possible in a real world scenario due to the fact that the contamination rate can not be known beforehand, we determined that this is a reasonable approach when evaluating the performance of the models. We chose to focus on 4 different preprocessing configurations for the anomaly detection, in order to validate how different manipulations of the data affect the results of different models. The configurations are the following:

1. *Baseline*: Anomaly detection performed on the default data for each model, meaning after interpolation and aggregation. This will act as the baseline for comparison with other results.

2. *Feature Selected*: Anomaly detection performed on features chosen using Sequential Forward Selection (SFS) for each heat pump system.

3. *UMAP Projected*: Anomaly detection performed on data projected to 10 UMAP dimensions, in order to see if UMAP (unsupervised feature extraction) can outperform a supervised feature selection.

4. *Savitzky-Golay Filtered*: In this configuration, the approximate seasonal residual is subtracted from each feature that exhibits seasonal behaviour. The approximate seasonal

residual is calculated per seasonal feature using a Savitzky-Golay filter. The window size of the filter was set to 1000, with a third degree polynomial interpolation function.

The best performing configuration for each heat pump is then used to perform the anomaly clustering using DBSCAN to evaluate whether successful predictions could be used to find novel errors. For further description and motivation for the choice of anomaly detection methods see chapter 3. Below we will present the chosen architectures for the deep learning approaches, namely deep isolation forest, autoencoder, and LSTM autoencoder.

4.5.1 Deep Isolation Forest Architecture

The chosen architecture for transforming the data space for the tree cuts in Deep Isolation Forest is a simple densely connected neural network. The network consists of two layers, one with 500 neurons, connecting to a layer of 100 neurons. A schematic of the architecture can be seen in figure 4.11.

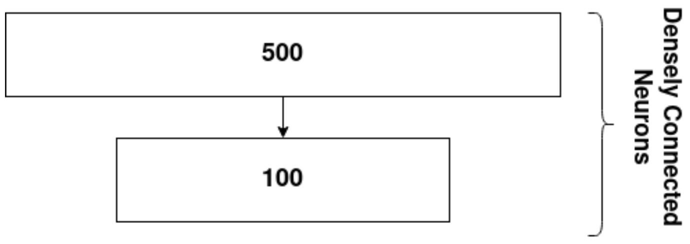


Figure 4.11: Schematic of the used deep isolation architecture that is used to transform the data space for the tree cuts. The architecture consists of two densely connected layers, of 500 and 100 neurons respectively.

4.5.2 Autoencoder Architecture

The chosen architecture for the Autoencoder consisted of five densely connected neural network layers, with the encoder and the decoder layers consisting of two layers while the latent space layer consisted of one dense layer of neurons. The hidden layer activation used was ReLU while the output activation was sigmoid. A schematic of the used architecture can be seen in figure 4.12.

4.5.3 LSTM Autoencoder Architecture

The chosen architecture for the LSTM autoencoder consisted of four densely connected LSTM unit layers with a repeat vector in the middle. The output then passes through a time distributed layer. The chosen activation for the LSTM layers was ReLU.

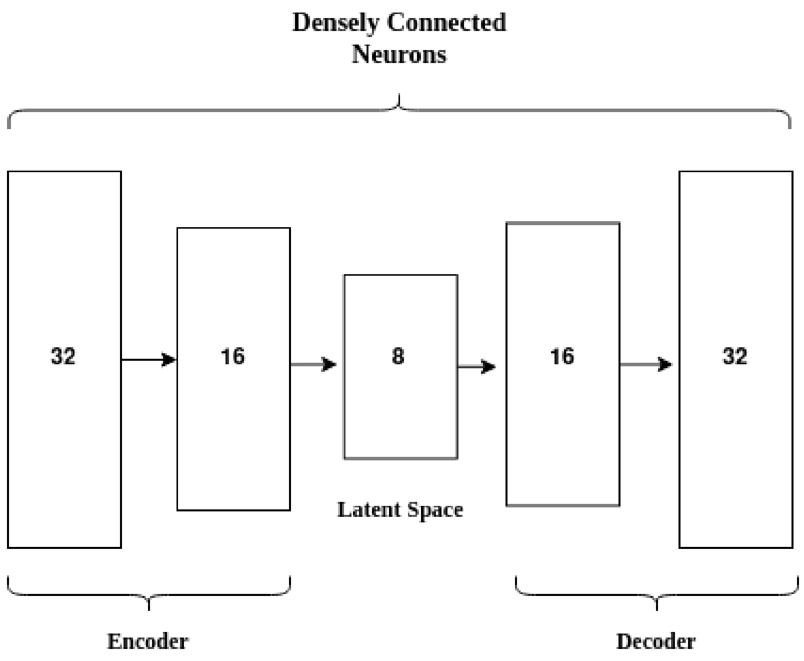


Figure 4.12: Schematic of the used autoencoder architecture. Each block represents a densely connected neural network layer, where the number represents the number of neurons in the layer.

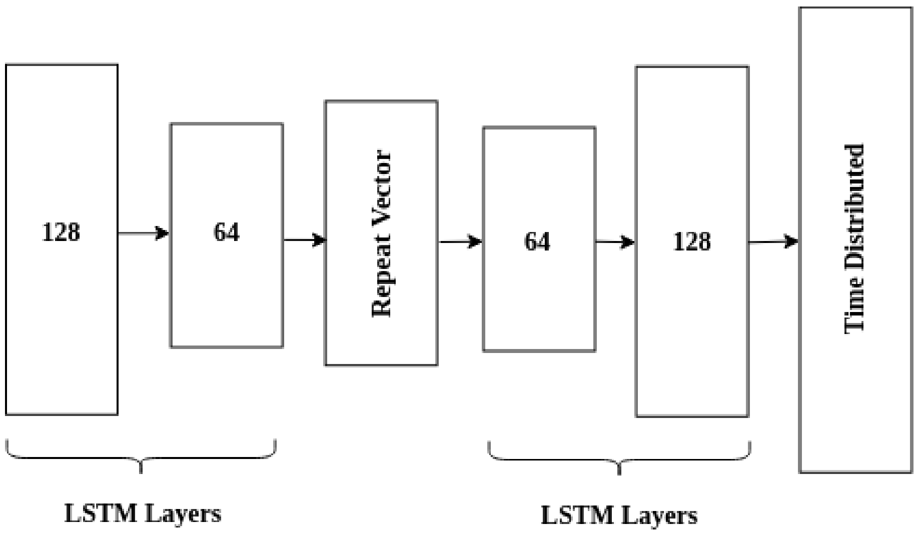


Figure 4.13: Schematic of the used LSTM autoencoder architecture. The blocks represent the different layers, with LSTM, repeat vector, and time distributed layers. The numbers represent the number of LSTM units in the layer.

4.6 Evaluation

This section covers the final phase of the CRISP-DM process, namely the evaluation phase. In this phase, the performance of the models was examined in order to determine their viability for the given task, using a combination of evaluation methods. The evaluation methods used included visual tools like plotting the results with UMAP projections, as well as numerical evaluations using different metrics. The metrics in question included precision, recall, macro F1 and MCC. For more in-depth description of these evaluation methods refer to section 3.7.

4.7 External Tools

Here we will present the different external libraries used for the different parts of the thesis as well as how they were utilised.

4.7.1 Scikit-learn

Scikit-learn is a popular Python-based machine learning library, providing modules for all steps of the machine learning process including pre-processing, running machine learning models, evaluation methods and more[31]. The utilised modules from the scikit-learn library included *sklearn.metrics* for evaluations, as well as normalisation methods from the *sklearn.preprocessing* module.

4.7.2 PyOD

PyOD (Python Outlier Detection) is a Python based library built for performing outlier/anomaly detection, offering a wide-range of different methods and algorithms [43]. This library was mainly used to implement the different used anomaly detection methods with the LSTM autoencoder being the only exception. The methods that were used were all part of the *pyod.models* module, and included the *IF, AutoEncoder, DIF, LOF, DBSCAN* submodules.

4.7.3 MLXtend

MLXtend (machine learning extensions) is a library including modules useful for many different data science tasks and is often used as a complement to other machine learning libraries such as scikit-learn [32]. The module that was utilised from the MLXtend library was the *mlxtend.feature_selection* module which was utilised to perform the sequential forward selection procedure.

4.7.4 TensorFlow Keras

Keras is a high-level API built on top of the popular TensorFlow machine learning library. It is a commonly used library for machine learning with a focus on deep learning tasks [24]. Keras was mainly used for implementing the LSTM autoencoder for which the *keras.layers, keras.optimizers, keras.Sequential, keras.callbacks* submodules were used.

Chapter 5

Results

In this chapter we present the results of the anomaly detection and clustering in accordance with the evaluation metrics presented in section 3.7. All approaches were validated using a fixed contamination rate corresponding to the contamination rate in the ground-truth for validation purposes. The results are mainly focused on the MCC score (see 3.7.2) since this metric was deemed to give the most nuanced view of model performance out of the presented metrics. The results are split into unsupervised (isolation forest labeled IF, deep isolation forest labeled DIF, local outlier factor labeled LOF, empirical-cumulative distribution function based outlier detection labeled ECOD) and semisupervised models (autoencoder labeled AE, LSTM autoencoder labeled LAE) since the semisupervised models should be trained on "clean" data and were thus split into a training (with removed data points corresponding to errors) and validation set, while the unsupervised models were trained on the whole data set. For the heat pumps that have lower than 1% contamination in the validation set the results are marked with an x.

5.1 Feature Selection Results

Here the results of the feature selection is shown, showing the mean cross-validation score of the classifier results as well as the selected number of features for each heat pump. What can be seen in table 5.1 is that the feature selection seems to be able to accurately identify which features are relevant for each heat pump since a mean cross-validation accuracy of >0.97 can be observed for all heat pumps. Important to note here is that the feature selection was trained using a supervised model, and thus these scores do not reflect how well the chosen feature selection will perform in an unsupervised scenario.

Id	*Cross-Validation Score*	*|Features|*
H1	0.997	5
H2	0.994	4
H3	0.993	5
H4	0.997	5
H5	0.986	5
H6	0.999	5
H7	0.995	5
H8	0.996	5
H9	0.997	5
H10	0.983	5
H11	0.994	5
H12	0.991	5
H13	0.996	5
H14	0.971	5
H15	0.986	5
H16	0.986	5
H17	0.999	5
H18	0.948	5
H19	0.984	5
H20	0.991	4
H23	0.997	5

Table 5.1: Results of the feature selection. Shows cross-validation score as well as the number of selected features.

5.2 Prediction Results

Here we will present the prediction results for the 4 different configurations, namely the predictions performed on the baseline, UMAP projected, feature selected and Savitzky-Golay filtered data. The results are presented with tables consisting of macro f1 score and MCC scores, as well as confusion matrices for the best and worst performing models.

5.2.1 Baseline

In this section, the results for the baseline are presented split into the results for the unsupervised and supervised models. Here we can see based on the MCC score that in the general case, the performance of the models amounts to random guess as can be seen in the fact that the best case average MCC score for a model is 0.09.

Unsupervised Models

Here the results for the unsupervised models for the baseline configuration are presented. Here we can see that generally speaking, the prediction is next to random since the average MCC score for the best performing model is 0.05. We can see that for the last two heat pumps in particular (H20, H23), all of the unsupervised models seem to hallucinate and have a worse than random guess performance. For some specific cases, namely heat pump H11 with the IF model, and H6 with the DIF model, the performance is slightly better, with an MCC score of 0.43 and 0.38 respectively. When looking at the macro f1 scores, a similar trend is seen, where average performance for the best performing model is 0.52, in principle random guess. When looking at the confusion matrices for the best and worst performing models (see table 5.1) we can see that for the best performing model a majority of the points were classified correctly, but with a lot of noise with 600 points classified incorrectly in total. The worst performing model was not able to identify any significant patterns.

Semisupervised Models

Here we can see the results for the semisupervised models, namely the autoencoder and LSTM autoencoder models. It can be observed that for heat pump H23 for the LSTM autoencoder, the achieved performance was an MCC score of 0.87, indicating a strong agreement between the predicted and labeled anomalies. We see that the autoencoder (AE) has poor overall performance with an average MCC score of 0.03, resulting in marginally-above-random-guess performance. The exception is for heat pump H4 where the autoencoder outperformed the LSTM autoencoder with an MCC score of 0.41 against a performance of 0. The LSTM autoencoder seems to perform slightly better in the general case with an average case MCC score of 0.12. While this is an improvement over the autoencoder, the general results of the LSTM autoencoder can still be considered next to random. The macro F1 scores also corroborate these findings, where the LAE model has an average f1 score of 0.58 and the AE model has an average f1 score of 0.5.

	Macro F1 Scores				*MCC Scores*			
	IF	**DIF**	**ECOD**	**LOF**	**IF**	**DIF**	**ECOD**	**LOF**
H1	0.51	0.50	0.49	0.53	0.03	0.01	-0.01	0.05
H2	0.49	0.50	0.49	0.50	-0.01	-0.01	-0.01	-0.01
H3	0.69	0.49	0.60	0.51	0.38	-0.01	0.20	0.03
H4	0.50	0.50	0.50	0.50	0.00	0.01	0.01	-0.01
H5	0.53	0.50	0.53	**0.56**	0.06	0.01	0.07	**0.12**
H6	0.50	**0.69**	0.50	0.49	0.00	**0.38**	0.00	-0.01
H7	0.52	0.51	0.50	0.54	0.04	0.01	0.00	0.08
H8	0.51	0.51	0.52	0.52	0.02	0.03	0.03	0.05
H9	0.53	0.54	0.54	0.54	0.06	0.09	0.07	0.08
H10	0.53	0.49	0.51	0.52	0.05	-0.01	0.02	0.04
H11	**0.71**	0.48	**0.66**	0.53	**0.43**	-0.04	**0.32**	0.06
H12	0.49	0.49	0.48	0.53	-0.01	-0.02	-0.05	0.07
H13	0.51	0.59	0.51	0.54	0.02	0.19	0.03	0.07
H14	0.57	0.53	0.53	0.54	0.14	0.06	0.05	0.08
H15	0.54	0.48	0.50	0.52	0.08	-0.05	0.01	0.05
H16	0.55	0.48	0.57	0.51	0.10	-0.04	0.14	0.02
H17	0.47	0.47	0.46	0.49	-0.06	-0.06	-0.07	-0.03
H18	0.48	0.48	0.47	0.52	-0.03	-0.03	-0.06	0.05
H19	0.56	0.45	0.55	0.49	0.11	-0.10	0.10	-0.02
H20	0.40	0.37	0.39	0.48	-0.20	-0.27	-0.23	-0.05
H23	0.40	0.49	0.30	0.49	-0.19	-0.02	-0.40	-0.02
μ	0.52	0.50	0.50	0.52	0.05	0.01	0.01	0.03

Table 5.2: Macro F1 and MCC score for the unsupervised models for the baseline configuration.

		prediction		
		p_p	n_p	total
ground truth	p_g	256	300	556
	n_g	300	9462	9762
	total	556	9762	

(a) Confusion matrix for best performing model (IF) with MCC score of 0.43 for heat pump H11.

		prediction		
		p_p	n_p	total
ground truth	p_g	494	3417	3911
	n_g	3422	3109	6531
	total	3916	6526	

(b) Confusion matrix for worst performing model (ECOD) with MCC score of -0.40 for heat pump H11.

Figure 5.1: Confusion matrix for best and worst performing unsupervised models for the baseline configuration.

		prediction		
		p_p	n_p	total
ground truth	p_g	656	59	715
	n_g	59	1291	1350
	total	715	1350	

(a) Confusion matrix for best performing model (LAE) with MCC score of 0.87 for heat pump H23.

		prediction		
		p_p	n_p	total
ground truth	p_g	222	582	804
	n_g	583	716	1299
	total	805	1298	

(b) Confusion matrix for worst performing model (AE) with MCC score of -0.17 for heat pump H15.

Figure 5.2: Confusion matrix for best and worst performing semisupervised models for the baseline configuration.

	Macro F1 Scores		*MCC Scores*	
	AE	**LAE**	**AE**	**LAE**
H1	0.49	0.60	-0.01	0.19
H2	x	x	x	x
H3	x	x	x	x
H4	x	x	x	x
H5	**0.57**	0.48	**0.15**	-0.03
H6	0.47	0.47	-0.05	-0.05
H7	0.49	0.52	-0.03	0.05
H8	x	x	x	x
H9	x	x	x	x
H10	0.51	0.51	0.02	0.02
H11	x	x	x	x
H12	x	x	x	x
H13	0.52	0.50	0.04	-0.01
H14	0.49	0.58	-0.02	0.15
H15	0.41	0.65	-0.17	0.29
H16	0.56	0.65	0.12	0.30
H17	x	x	x	x
H18	0.55	0.49	0.10	-0.02
H19	x	x	x	x
H20	0.52	0.53	0.05	0.05
H23	0.44	**0.94**	-0.11	**0.87**
μ	0.50	0.58	0.01	0.15

Table 5.3: Macro F1 and MCC score for the semisupervised models for the baseline configuration.

5.2.2 UMAP Projected

Here the results for the UMAP projected data set is presented. The data has first been normalized using MinMax normalization before being projected to a 10 dimensional UMAP feature space.

Unsupervised Models

Yet again it may be observed that the general results indicate that the performance essentially amounts to random guess across all of the evaluated unsupervised models. The results show a next to 0 average MCC and a close to 0.5 macro f1 score across all models. A notable observation here is that for heat pump H20, the IF and DIF models seem to hallucinate, having an MCC score of -0.33 and -0.34 respectively. The best performing model here is the DIF model, which has a best case performance of an MCC score of 0.25 and a macro f1 score of 0.63 for heat pump H7. When looking at the worst case confusion matrix (see figure 5.3), we can see that the model was barely able to identify any errors with only 12 identified true positives.

	Macro F1 Scores				*MCC Scores*			
	IF	**DIF**	**ECOD**	**LOF**	**IF**	**DIF**	**ECOD**	**LOF**
H1	0.49	0.49	0.49	0.52	-0.02	-0.02	-0.02	0.03
H2	0.49	0.49	0.61	0.50	-0.01	-0.01	0.22	0.00
H3	0.49	0.49	0.49	0.51	-0.02	-0.02	-0.02	0.03
H4	0.49	0.49	0.50	0.51	-0.01	-0.01	-0.01	0.01
H5	0.50	0.49	0.51	0.50	-0.01	-0.03	0.01	0.00
H6	0.50	0.49	0.49	0.50	-0.01	-0.01	-0.01	0.01
H7	**0.62**	0.53	0.49	0.50	**0.25**	0.06	-0.02	-0.00
H8	0.49	0.49	0.50	0.50	-0.02	-0.03	0.00	0.00
H9	0.49	0.49	0.53	0.53	-0.03	-0.03	0.05	0.06
H10	0.48	0.48	0.48	0.50	-0.03	-0.04	-0.04	0.01
H11	0.54	**0.63**	0.51	0.53	0.09	**0.25**	0.02	0.06
H12	0.55	0.55	0.47	0.51	0.09	0.10	-0.06	0.01
H13	0.60	0.48	0.49	**0.55**	0.20	-0.03	-0.02	**0.09**
H14	0.54	0.54	**0.62**	0.54	0.07	0.07	0.24	0.08
H15	0.51	0.47	0.49	0.53	0.02	-0.06	-0.03	0.05
H16	0.49	0.49	0.49	0.51	-0.02	-0.02	-0.02	0.01
H17	0.48	0.48	0.47	0.51	-0.04	-0.05	-0.05	0.02
H18	0.47	0.44	0.47	0.51	-0.06	-0.11	-0.05	0.01
H19	0.46	0.54	0.42	0.49	-0.07	0.08	-0.16	-0.03
H20	0.34	0.33	**0.62**	0.48	-0.33	-0.34	**0.25**	-0.04
H23	0.48	0.49	0.53	0.50	-0.04	-0.03	0.06	0.01
μ	0.50	0.49	0.51	0.51	-0.00	-0.01	0.02	0.02

Table 5.4: Macro F1 and MCC score for the unsupervised models for the UMAP configuration.

		prediction		
		p_p	n_p	total
ground truth	p_g	163	391	554
	n_g	391	9371	9762
	total	554	9762	

(a) Confusion matrix for best performing model (DIF) with MCC score of 0.25 for heat pump H11.

		prediction		
		p_p	n_p	total
ground truth	p_g	12	2575	2587
	n_g	2578	4828	7406
	total	2590	7403	

(b) Confusion matrix for worst performing model (DIF) with MCC score of -0.34 for heat pump H20.

Figure 5.3: Confusion matrix for best and worst performing UMAP projected unsupervised models

Semisupervised Models

For the semisupervised models we can see that we get a best case performance of an MCC score of 0.9 and an macro f1 score of 0.95 for heat pump H23 with the LSTM autoencoder while we have the worst case performance for the same heat pump for the autoencoder model with an MCC score of -0.11 and a macro f1 score of 0.44.

		prediction		
		p_p	n_p	total
ground truth	p_g	669	46	715
	n_g	46	1304	1350
	total	715	1350	

(a) Confusion matrix for best performing model (LAE) with MCC score of 0.9 for heat pump H23.

		prediction		
		p_p	n_p	total
ground truth	p_g	191	524	715
	n_g	525	849	1374
	total	716	1373	

(b) Confusion matrix for worst performing model (AE) with MCC score of -0.11 for heat pump H23.

Figure 5.4: Confusion matrix for best and worst performing semisupervised models using UMAP projection

	Macro F1 Scores		*MCC Scores*	
	AE	**LAE**	**AE**	**LAE**
H1	0.49	0.49	-0.01	-0.01
H2	x	x	x	x
H3	x	x	x	x
H4	x	x	x	x
H5	0.47	0.46	-0.06	-0.08
H6	0.47	0.47	-0.05	-0.05
H7	0.49	0.49	-0.03	-0.03
H8	x	x	x	x
H9	x	x	x	x
H10	0.49	0.51	-0.01	0.02
H11	x	x	x	x
H12	x	x	x	x
H13	0.50	0.50	-0.01	-0.01
H14	0.46	0.50	-0.08	0.00
H15	**0.55**	0.64	**0.1**	0.28
H16	0.45	0.46	-0.11	-0.08
H17	x	x	x	x
H18	0.53	0.45	0.06	-0.11
H19	x	x	x	x
H20	0.54	0.50	0.08	0.01
H23	0.44	0.95	-0.11	**0.9**
μ	0.49	0.54	-0.02	0.07

Table 5.5: Macro F1 and MCC score for the semisupervised models for the UMAP configuration.

5.2.3 Feature Selected

Here the results of the anomaly detection performed on feature selected data are presented. The amount of features utilised varied between 4 and 5, depending on which features were deemed most important by the sequential forward selection process.

Unsupervised Models

The feature selection results shows a marginal average increase from the baseline. The MCC score increases from an average of around $\mathbf{0.03}$ to $\mathbf{0.11}$ when combining all models. Similarly the increase for F1 is from $\mathbf{0.51}$ to $\mathbf{0.55}$. Once again the performance varies largely dependent on the heat pump and anomaly detection model. We observe the best case MCC score to be $\mathbf{0.92}$ using DIF for heat pump H8. The worst case MCC score is $\mathbf{-0.39}$ for heat pump H23. In figure 5.8 the confusion matrix for the best and worst performing models can be seen. What may be observed that in the best case the DIF model showed promising performance, only misclassifying 39 points out of a total of 9466 with a precision of 0.93.

	Macro F1 Scores				*MCC Scores*			
	IF	**DIF**	**ECOD**	**LOF**	**IF**	**DIF**	**ECOD**	**LOF**
H1	0.59	0.54	0.50	0.50	0.18	0.08	0.01	-0.01
H2	0.62	0.55	0.57	0.49	0.24	0.09	0.15	-0.01
H3	0.77	0.53	0.65	0.56	0.55	0.06	0.30	0.12
H4	0.50	0.49	0.50	0.52	-0.00	-0.01	-0.01	0.05
H5	0.53	0.49	0.53	0.52	0.06	-0.03	0.07	0.05
H6	**0.78**	0.49	**0.83**	0.53	**0.55**	-0.01	**0.67**	0.06
H7	0.51	0.51	0.50	0.51	0.02	0.02	0.01	0.03
H8	0.74	**0.96**	0.57	0.50	0.48	**0.92**	0.13	0.00
H9	0.50	0.53	0.49	0.53	0.01	0.07	-0.01	0.06
H10	0.52	0.51	0.51	0.50	0.04	0.03	0.02	0.00
H11	0.72	0.47	0.72	0.53	0.44	-0.06	0.44	0.05
H12	0.77	0.82	0.67	0.55	0.53	0.65	0.35	0.09
H13	0.49	0.49	0.48	0.51	-0.02	-0.01	-0.03	0.02
H14	0.73	0.86	0.62	**0.57**	0.46	0.72	0.24	**0.13**
H15	0.61	0.47	0.60	0.49	0.21	-0.05	0.20	-0.01
H16	0.56	0.56	0.60	0.53	0.13	0.12	0.20	0.06
H17	0.48	0.50	0.48	0.48	-0.03	-0.00	-0.05	-0.04
H18	0.49	0.63	0.49	0.51	-0.01	0.26	-0.03	0.01
H19	0.55	0.53	0.46	0.51	0.09	0.05	-0.08	0.01
H20	0.51	0.48	0.45	0.41	0.02	-0.04	-0.10	-0.17
H23	0.43	0.42	0.31	0.46	-0.15	-0.16	-0.39	-0.08
μ	0.59	0.56	0.55	0.51	0.18	0.13	0.10	0.02

Table 5.6: Macro F1 and MCC score for the unsupervised models using the feature selected configuration.

		prediction		
		p_p	n_p	total
ground truth	p_g	235	18	253
	n_g	21	9192	9213
	total	256	9210	

(a) Confusion matrix for best performing model (DIF) with MCC score of 0.92 for heat pump H8.

		prediction		
		p_p	n_p	total
ground truth	p_g	517	3394	3911
	n_g	3382	3149	6531
	total	3899	6543	

(b) Confusion matrix for worst performing model (ECOD) with MCC score of -0.39 for heat pump H23.

Figure 5.5: Confusion matrix for best and worst performing unsupervised models using the feature selected configuration.

Semisupervised Models

The feature selection results in general shows promising results for the semisupervised models, at least for the autoencoder model with an average MCC score of 0.17 and a best case MCC score of 0.71. The LSTM autoencoder did not perform as well here, with an MCC score of 0.04 and a best case performance of an 0.47 MCC score.

		prediction		
		p_p	n_p	total
ground truth	p_g	255	70	325
	n_g	93	1683	1776
	total	348	1753	

(a) Confusion matrix for best performing model (AE) with MCC score of 0.71 for heat pump H14.

		prediction		
		p_p	n_p	total
ground truth	p_g	0	715	715
	n_g	715	635	1350
	total	715	1350	

(b) Confusion matrix for worst performing model (LAE) with MCC score of -0.53 for heat pump H23.

Figure 5.6: Confusion matrix for best and worst performing unsupervised models using feature selected features

	Macro F1 Scores		*MCC Scores*	
	AE	**LAE**	**AE**	**LAE**
H1	0.49	0.55	-0.01	0.09
H2	x	x	x	x
H3	x	x	x	x
H4	x	x	x	x
H5	0.47	0.52	-0.05	0.04
H6	0.49	0.50	-0.02	-0.01
H7	0.50	0.52	0.01	0.05
H8	x	x	x	x
H9	x	x	x	x
H10	0.49	0.49	-0.01	-0.01
H11	x	x	x	x
H12	x	x	x	x
H13	0.50	0.60	-0.01	0.19
H14	**0.86**	**0.74**	**0.71**	**0.47**
H15	0.54	0.49	0.07	-0.02
H16	0.69	0.62	0.39	0.23
H17	x	x	x	x
H18	0.67	0.49	0.34	-0.02
H19	x	x	x	x
H20	0.50	0.50	0.01	-0.01
H23	0.78	0.24	0.64	-0.53
μ	0.66	0.52	0.17	0.04

Table 5.7: Feature Selected semisupervised models

5.2.4 Savitzky-Golay Filtered

Here we will present the results of the anomaly detection for the heat pump data where the seasonal residual is removed by using a Savitzky-Golay filter with a window size of 1000 and a 3rd degree polynomial.

Unsupervised Models

For the unsupervised models we can see that the best result was for the ECOD model for heat pump H11 which has an MCC score of 0.49 and a macro f1 score of 0.74. We can also see that for the same heat pump, the IF model showcased comparable but slightly worse results with an MCC score of 0.46 and a macro f1 score of 0.73. Also notable is the DIF model which scored an MCC score of 0.35 for heat pump H6 where all the other models were not able to detect the anomalies with all having an MCC score of close to 0. In general the IF model performed the best, but again with close-to-random-guess-performance with a mean MCC score of 0.06 and a mean macro f1 score of 0.53.

	Macro F1 Scores				*MCC Scores*			
Id	**IF**	**DIF**	**ECOD**	**LOF**	**IF**	**DIF**	**ECOD**	**LOF**
H1	0.52	0.51	0.51	0.53	0.03	0.01	0.01	0.05
H2	0.49	0.50	0.50	0.50	-0.01	0.00	-0.01	0.00
H3	0.69	0.49	0.61	0.53	0.39	-0.01	0.22	0.05
H4	0.50	0.50	0.50	0.51	0.01	-0.00	-0.00	0.01
H5	0.54	0.50	0.52	**0.56**	0.09	-0.01	0.05	**0.13**
H6	0.50	**0.67**	0.50	0.49	-0.00	**0.35**	-0.00	-0.01
H7	0.52	0.51	0.52	0.55	0.04	0.03	0.03	0.10
H8	0.50	0.51	0.51	0.51	0.01	0.02	0.02	0.02
H9	0.54	0.60	0.55	0.54	0.09	0.20	0.11	0.07
H10	0.53	0.49	0.50	0.53	0.07	-0.02	-0.00	0.06
H11	**0.73**	0.48	**0.74**	0.54	**0.46**	-0.03	**0.49**	0.08
H12	0.50	0.51	0.50	0.52	0.00	0.02	0.00	0.04
H13	0.52	0.59	0.51	0.52	0.04	0.19	0.03	0.05
H14	0.56	0.53	0.58	0.54	0.13	0.06	0.16	0.07
H15	0.56	0.48	0.53	0.51	0.12	-0.04	0.06	0.02
H16	0.53	0.48	0.57	0.50	0.06	-0.04	0.14	0.01
H17	0.47	0.47	0.47	0.49	-0.06	-0.06	-0.07	-0.02
H18	0.50	0.48	0.48	0.52	-0.01	-0.03	-0.04	0.03
H19	0.53	0.47	0.51	0.48	0.05	-0.05	0.01	-0.03
H20	0.40	0.40	0.38	0.48	-0.19	-0.21	-0.25	-0.04
H23	0.50	0.40	0.48	0.49	-0.00	-0.20	-0.05	-0.01
Average	0.53	0.50	0.52	0.52	0.06	0.01	0.04	0.03

Table 5.8: Savitzky-Golay Filtered unsupervised models

		prediction		
		p_p	n_p	total
ground truth	p_g	285	269	554
	n_g	269	9493	9762
	total	554	9762	

(a) Confusion matrix for best performing model (ECOD) with MCC score of 0.49 for heat pump H11.

		prediction		
		p_p	n_p	total
ground truth	p_g	194	2393	2587
	n_g	2396	5010	7406
	total	2590	7403	

(b) Confusion matrix for worst performing model (ECOD) with MCC score of -0.25 for heat pump H20.

Figure 5.7: Confusion matrix for best and worst performing unsupervised models using feature selected features

Semisupervised Models

For the semisupervised models the LSTM autoencoder shows both the best and worst result for two different heat pumps. The best performing model has an MCC score of 0.47 and a macro f1 score of 0.74 while the worst performing model had an MCC score of -0.53 and a macro f1 score of 0.24. We note that the anomalies seem to be captured in an inverse fashion in the worst performing model.

		prediction		
		p_p	n_p	total
ground truth	p_g	172	139	311
	n_g	139	1627	1766
	total	311	1766	

(a) Confusion matrix for best performing model (LAE) with MCC score of 0.47 for heat pump H14.

		prediction		
		p_p	n_p	total
ground truth	p_g	0	715	715
	n_g	715	635	1350
	total	715	1350	

(b) Confusion matrix for worst performing model (LAE) with MCC score of -0.53 for heat pump H23.

Figure 5.8: Confusion matrix for best and worst performing unsupervised models using feature selected features

	Macro F1 Scores		*MCC Scores*	
	AE	**LAE**	**AE**	**LAE**
H1	0.49	0.55	-0.01	0.09
H2	x	x	x	x
H3	x	x	x	x
H4	x	x	x	x
H5	0.57	0.52	0.13	0.04
H6	0.48	0.50	-0.04	-0.01
H7	0.51	0.52	0.03	0.05
H8	x	x	x	x
H9	x	x	x	x
H10	0.49	0.49	-0.01	-0.01
H11	x	x	x	x
H12	x	x	x	x
H13	0.52	0.60	0.04	0.19
H14	0.56	0.74	0.12	**0.47**
H15	0.54	0.49	0.08	-0.02
H16	**0.59**	0.62	**0.18**	0.23
H17	x	x	x	x
H18	0.53	0.49	0.06	-0.02
H19	x	x	x	x
H20	0.50	0.50	-0.01	-0.01
H23	0.24	0.24	-0.52	-0.53
μ	0.50	0.52	0.00	0.04

Table 5.9: Savitzky-Golay Filtered semisupervised models

5.2.5 Best Performing Models and Configurations Per Heat Pump

Here the summarized results are presented (see table 5.10), showing which model and configuration performed the best for every heat pump. What can generally be noted is that there is no single model or configuration that performs well across all heat pumps. Furthermore, it may be observed that for 8 out of the 23 heat pumps we successfully managed to find configurations that had at least a moderate performance with an MCC score of at least 0.4, but with lacking results for the remaining heat pumps. The models that managed to achieve a high performance of an MCC score of >0.7 were the DIF and LSTM autoencoder models. ECOD was also not far behind in the best case, with an MCC score of 0.67 in the best case. It is worthy to note that for some of the heat pumps a close to random guess performance was the best achieved result. This was the case for H17 and H10 which both have an MCC score of <0.1. For the heat pumps H23 and H8 we were able to achieve a close to perfect result of above 0.9 MCC score.

Id	**Model**	**Configuration**	**MCC Score**
H1	*LAE*	Baseline	0.19
H2	*ECOD*	UMAP	0.22
H3	*IF*	Feature Selected	0.55
H4	*AE*	Baseline	0.41
H5	*LOF/AE*	Savitzky-Golay Filtered	0.13
H6	*ECOD*	Feature Selected	0.67
H7	*IF*	UMAP	0.25
H8	*DIF*	Feature Selected	0.92
H9	*DIF*	Savitzky-Golay Filtered	0.2
H10	*IF*	Savitzky-Golay Filtered	0.07
H11	*ECOD*	Savitzky-Golay Filtered	0.49
H12	*DIF*	Feature Selected	0.65
H13	*DIF/LAE*	Baseline/Feature Selected/Savitzky-Golay Filtered	0.19
H14	*DIF*	Feature Selected	0.72
H15	*LAE*	Baseline	0.29
H17	*LOF*	UMAP	0.02
H18	*AE*	Feature Selected	0.34
H19	*IF*	Baseline	0.11
H20	*ECOD*	UMAP	0.25
H23	*LAE*	UMAP	0.9

Table 5.10: Best Performing model and configuration for each tested heat pump.

5.2.6 True positive rates comparison

Following are results comparing the true positive hit rate of predicted anomalies to actual error codes. Comparisons are presented as baseline true positive rate in the left column and difference after applying a configuration in the right. The configurations are feature selection 5.11, UMAP 5.14 and Savitsky-Golay filter 5.13. In table 5.12 the difference in true positive rate for the most frequently appearing error, when that specific error is targeted in the feature selection, is presented. In the baseline case the best performing models are the semisupervised ones, namely the Autoencoder and the LSTM-Autoencoder, with average true positive rates of 0.20 and 0.30 respectively, notably due to the high true positive rates for heat pump H20 where both have a true positive rate of 0.97. For all other models the average true positive rates are around 0.1.

Observing the configurations one by one we see that for the feature selection in table 5.11 the average case is an improvement of $+0.065$. Feature selection has the highest improvement on the Auto Encoder with an increase of $+0.15$. The other models also show general improvement after feature selection with the exception of the LSTM Auto Encoder which has a -0.06 change, mainly due to the true positive rate of heat pump H23 decreasing by 0.92.

For the UMAP configuration in table 5.14 we observe reductions in the the true positive rate for all models. The average reduction is -0.03 with the most impacted model being the LSTM-Autoencoder with a reduction of -0.07. This mainly due to the -0.34 decrease in true positive rate for heat pump H16.

For the Savitsky-Golay filter configuration in table 5.13 we observe minor changes for all models with the average change approaching zero. In general, changes are minor with the exception for a decrease of 0.92 for the LSTM-Autoencoder for heat pump H23.

For the configuration comparing feature selection for the most frequently appearing error code in table 5.12 we observe a baseline true positive rate of around 0.1 for all models. These are improved for most models with an average improvement of DIF, IF and ECOD being 0.07 whereas LOF observes a minor decrease of -0.01. Largest increase is for DIF with an average of $+0.09$ and an increase of $+0.74$ and $+0.69$ for heat pumps H12 and H14 respectively. General best performance after targeted feature selection is by IF with a true positive rate of 0.18. Note that the semisupervised models, Autoencoder and LSTM-Autoencoder, were omitted since we were unable to guarantee the availability of the targeted error in the validation dataset.

5.3 Anomaly Clustering Results

Here the results of the anomaly clustering using DBSCAN (see 3.6 for details) will be presented. The anomaly clustering was performed using the true positives that were found using the best performing combination of anomaly detection model and configuration as shown in table 5.10. The results will first consist of the baseline case, where we extracted the corresponding baseline values (the aggregated heat pump values without any feature manipulation) for each feature corresponding to the found true positive points. The next two cases that are presented were ones where the baseline values are projected to 2, 5, and 10 dimensions using PCA and UMAP respectively. Lastly, we present the result of performing the DBSCAN directly on the best performing anomaly detection configuration, i.e., if the Savitzky-Golay filtered data had the best result for H1 then the DBSCAN clustering was performed on that data. Important to note here is that it was decided that the results will be limited to cases where there are at minimum two unique error codes among the true positive points, but also

	Baseline						Difference					
Id	AE	DIF	ECOD	IF	LOF	LAE	AE	DIF	ECOD	IF	LOF	LAE
H1	0.00	0.03	0.01	0.05	0.07	0.20	+0.0	+0.07	+0.02	+0.15	-0.06	-0.1
H2	x	0.01	0.00	0.00	0.01	x	x	+0.09	+0.16	+0.25	-0.01	x
H3	x	0.01	0.21	0.40	0.05	x	x	+0.08	+0.1	+0.16	+0.09	x
H4	x	0.02	0.02	0.01	0.01	x	x	-0.02	-0.01	+0.0	+0.05	x
H5	0.21	0.04	0.10	0.08	0.14	0.04	-0.19	-0.04	+0.0	+0.01	-0.06	+0.07
H6	0.00	**0.38**	0.01	0.01	0.00	0.00	+0.03	-0.38	**+0.66**	**+0.55**	+0.07	+0.04
H7	0.00	0.03	0.02	0.06	0.10	0.07	+0.04	+0.01	+0.0	-0.02	-0.06	+0.0
H8	x	0.05	0.06	0.05	0.07	x	x	**+0.87**	+0.1	+0.44	-0.04	x
H9	x	0.11	0.10	0.08	0.11	x	x	-0.02	-0.09	-0.05	-0.02	x
H10	0.04	0.04	0.06	0.10	0.09	0.04	-0.04	+0.03	+0.0	-0.02	-0.04	-0.04
H11	x	0.02	**0.36**	**0.46**	0.11	x	x	-0.02	+0.11	+0.01	-0.01	x
H12	x	0.04	0.02	0.05	0.13	x	x	+0.63	+0.37	+0.51	+0.02	x
H13	0.05	0.23	0.07	0.06	0.12	0.00	-0.05	-0.19	-0.06	-0.03	-0.06	+0.2
H14	0.14	0.15	0.14	0.22	0.16	0.28	+0.64	+0.59	+0.17	+0.29	+0.05	+0.27
H15	0.28	0.05	0.10	0.16	0.13	0.56	+0.15	-0.01	+0.17	+0.12	-0.05	-0.18
H16	0.22	0.03	0.19	0.16	0.08	0.38	+0.24	+0.14	+0.06	+0.02	+0.04	-0.06
H17	x	0.01	0.00	0.01	0.04	x	x	+0.05	+0.02	+0.02	-0.01	x
H18	0.19	0.07	0.05	0.07	0.14	0.08	+0.22	+0.26	+0.03	+0.02	-0.03	+0.0
H19	x	0.07	0.24	0.25	0.13	x	x	+0.13	-0.15	-0.02	+0.03	x
H20	**0.97**	0.06	0.09	0.11	0.22	**0.97**	-0.01	+0.17	+0.09	+0.17	**-0.09**	-0.01
H23	0.27	0.37	0.13	0.25	**0.36**	0.92	**+0.73**	-0.1	+0.0	+0.03	-0.04	**-0.92**
μ	0.20	0.09	0.09	0.13	0.11	0.30	+0.15	+0.11	+0.08	+0.12	-0.01	-0.06

Table 5.11: Average hit rates per pump difference before and after feature selection.

	Baseline				Difference			
Id	DIF	ECOD	IF	LOF	DIF	ECOD	IF	LOF
H1	0.03	0.01	0.05	0.07	+0.05	+0.02	+0.05	-0.06
H2	0.01	0.00	0.00	0.01	+0.0	+0.0	+0.0	+0.0
H3	0.01	0.19	0.41	0.06	+0.0	-0.13	-0.24	+0.08
H4	0.00	0.01	0.00	0.00	+0.0	+0.0	+0.0	+0.0
H5	0.01	0.05	0.08	0.28	+0.02	**+0.35**	+0.28	**-0.17**
H6	**0.4**	0.01	0.00	0.00	+0.0	+0.25	+0.0	+0.0
H7	0.02	0.02	0.02	0.06	-0.01	+0.0	-0.01	-0.05
H8	0.06	0.06	0.05	0.08	+0.0	+0.03	+0.41	+0.01
H9	0.11	0.10	0.08	0.10	-0.04	-0.08	+0.02	+0.01
H10	0.00	0.05	0.02	0.04	+0.0	+0.0	+0.0	-0.03
H11	0.02	**0.35**	**0.46**	0.11	-0.02	-0.17	-0.26	-0.01
H12	0.04	0.02	0.05	0.12	+0.69	+0.26	+0.58	-0.03
H13	0.23	0.08	0.07	0.13	-0.21	-0.05	-0.03	-0.06
H14	0.16	0.09	0.14	0.16	**+0.74**	+0.24	**+0.69**	+0.04
H15	0.04	0.07	0.11	0.11	+0.02	+0.19	+0.0	-0.03
H16	0.01	0.17	0.13	0.08	+0.0	+0.0	-0.09	-0.04
H17	0.01	0.00	0.01	0.04	+0.0	+0.0	+0.01	+0.03
H18	0.07	0.05	0.07	0.14	+0.33	+0.03	+0.05	-0.03
H19	0.07	0.22	0.26	0.12	+0.05	-0.21	-0.06	+0.05
H20	0.03	0.08	0.08	0.21	+0.27	+0.22	+0.13	+0.04
H23	0.26	0.15	0.31	**0.39**	-0.09	+0.03	-0.04	+0.02
μ	0.08	0.08	0.11	0.11	+0.09	+0.05	+0.07	-0.01

Table 5.12: Target hit rates per pump for most frequently appearing error code for that pump, comparing difference before and after feature selection for the four unsupervised algorithms IF, DIF, LOF, ECOD. The two semi supervised models were omitted due to not being able to guarantee the presence of the target error in the validation data set.

			Baseline						Difference			
Id	AE	DIF	ECOD	IF	LOF	LAE	AE	DIF	ECOD	IF	LOF	LAE
H1	0.00	0.03	0.01	0.05	0.07	0.20	+0.0	+0.0	+0.02	+0.0	+0.0	-0.1
H2	x	0.01	0.00	0.00	0.01	x	x	+0.0	+0.01	+0.0	+0.0	x
H3	x	0.01	0.21	0.40	0.05	x	x	+0.0	+0.03	+0.0	+0.02	x
H4	x	0.02	0.02	0.01	0.01	x	x	-0.01	-0.01	+0.01	+0.02	x
H5	0.21	0.04	0.10	0.08	0.14	0.04	-0.01	-0.02	-0.02	+0.03	+0.01	+0.07
H6	0.00	**0.38**	0.01	0.01	0.00	0.00	+0.01	-0.02	+0.0	+0.0	+0.0	+0.04
H7	0.00	0.03	0.02	0.06	0.10	0.07	+0.05	+0.01	+0.03	+0.0	+0.02	+0.0
H8	x	0.05	0.06	0.05	0.07	x	x	+0.0	-0.01	-0.01	**-0.03**	x
H9	x	0.11	0.10	0.08	0.11	x	x	+0.11	+0.03	+0.03	-0.02	x
H10	0.04	0.04	0.06	0.10	0.09	0.04	-0.04	-0.01	-0.01	+0.01	+0.02	-0.04
H11	x	0.02	**0.36**	**0.46**	0.11	x	x	+0.01	+0.15	+0.03	+0.02	x
H12	x	0.04	0.02	0.05	0.13	x	x	+0.04	+0.04	+0.01	**-0.03**	x
H13	0.05	0.23	0.07	0.06	0.12	0.00	+0.0	+0.0	+0.0	+0.03	**-0.03**	+0.2
H14	0.14	0.15	0.14	0.22	0.16	0.28	+0.12	+0.0	+0.09	-0.02	+0.0	+0.27
H15	0.28	0.05	0.10	0.16	0.13	0.56	+0.15	+0.0	+0.05	+0.04	**-0.03**	-0.18
H16	0.22	0.03	0.19	0.16	0.08	0.38	+0.05	+0.0	+0.0	-0.04	-0.01	-0.06
H17	x	0.01	0.00	0.01	0.04	x	x	+0.0	+0.0	+0.0	+0.01	x
H18	0.19	0.07	0.05	0.07	0.14	0.08	-0.03	+0.0	+0.02	+0.02	-0.01	+0.0
H19	x	0.07	0.24	0.25	0.13	x	x	+0.04	**-0.08**	**-0.05**	+0.0	x
H20	**0.97**	0.06	0.09	0.11	0.22	**0.97**	-0.01	+0.05	-0.02	+0.01	+0.01	-0.01
H23	0.27	0.37	0.13	0.25	**0.36**	0.92	-0.27	**-0.12**	+0.21	+0.12	+0.01	**-0.92**
μ	0.20	0.09	0.09	0.13	0.11	0.30	0.00	+0.0	+0.03	+0.01	+-0.0	-0.06

Table 5.13: Average hit rates per pump for Savitsky-Golay filtered vs baseline results.

	Baseline						Difference					
Id	AE	DIF	ECOD	IF	LOF	LAE	AE	DIF	ECOD	IF	LOF	LAE
H1	0.00	0.03	0.01	0.05	0.07	0.20	+0.0	-0.03	-0.01	-0.05	-0.02	-0.2
H2	x	0.01	0.00	0.00	0.01	x	x	-0.01	+0.23	+0.0	+0.0	x
H3	x	0.01	0.21	0.40	0.05	x	x	-0.01	-0.21	**-0.4**	+0.0	x
H4	x	0.02	0.02	0.01	0.01	x	x	-0.02	-0.01	-0.01	+0.02	x
H5	0.21	0.04	0.10	0.08	0.14	0.04	-0.19	-0.04	-0.06	-0.06	**-0.11**	-0.03
H6	0.00	**0.38**	0.01	0.01	0.00	0.00	+0.0	**-0.38**	-0.01	-0.01	+0.02	+0.0
H7	0.00	0.03	0.02	0.06	0.10	0.07	+0.0	+0.05	-0.02	+0.2	-0.08	-0.07
H8	x	0.05	0.06	0.05	0.07	x	x	-0.05	-0.03	-0.04	-0.04	x
H9	x	0.11	0.10	0.08	0.11	x	x	-0.11	-0.02	-0.08	-0.03	x
H10	0.04	0.04	0.06	0.10	0.09	0.04	-0.04	-0.03	-0.05	-0.08	-0.03	+0.0
H11	x	0.02	**0.36**	**0.46**	0.11	x	x	+0.27	**-0.29**	-0.32	+0.0	x
H12	x	0.04	0.02	0.05	0.13	x	x	+0.12	-0.02	+0.1	-0.06	x
H13	0.05	0.23	0.07	0.06	0.12	0.00	-0.05	-0.21	-0.04	+0.18	+0.01	+0.0
H14	0.14	0.15	0.14	0.22	0.16	0.28	-0.05	+0.01	+0.17	-0.06	+0.0	-0.13
H15	0.28	0.05	0.10	0.16	0.13	0.56	+0.16	-0.01	-0.04	-0.06	+0.01	+0.0
H16	0.22	0.03	0.19	0.16	0.08	0.38	-0.21	+0.01	-0.15	-0.12	-0.01	**-0.34**
H17	x	0.01	0.00	0.01	0.04	x	x	+0.01	+0.02	+0.02	+0.05	x
H18	0.19	0.07	0.05	0.07	0.14	0.08	-0.04	-0.07	+0.0	-0.02	-0.03	-0.08
H19	x	0.07	0.24	0.25	0.13	x	x	+0.15	-0.23	-0.16	+0.0	x
H20	**0.97**	0.06	0.09	0.11	0.22	**0.97**	+0.0	-0.06	+0.35	-0.1	+0.01	-0.01
H23	0.27	0.37	0.13	0.25	**0.36**	0.92	+0.0	-0.01	+0.28	+0.1	+0.02	+0.01
Average	0.20	0.09	0.09	0.13	0.11	0.30	-0.04	-0.02	-0.01	-0.05	-0.01	-0.07

Table 5.14: Average hit rates per pump for UMAP dimensionality reduced vs baseline results.

that there are a minimum of 10 true positive points in total.

5.3.1 Baseline

In the baseline case as can be seen in table 5.15, we can generally see that the clustering did not adequately capture any patterns reliably. We can see that both the ARI and NMI score are around 0, indicating a random guess performance for the clustering.

Id	|Error Codes|	ARI	NMI
H2	2	-0.09	0.03
H3	3	-0.03	0.01
H5	6	0	0
H7	3	-0.18	0.1
H8	2	-0.01	0
H10	7	-0.23	0.18
H11	7	-0.06	0.03
H12	2	-0	0
H14	4	-0.01	0
H15	4	0.12	0.14
H19	6	0.04	0.17
H20	4	0.01	0.05
μ	4.2	-0.04	0.06

Table 5.15: Clustering results for the baseline case, showcasing the heat pump id of the heat pump that the clustering was performed on, the number of unique error codes among the detected true positive points, as well as ARI and NMI scores for the detected clusters.

5.3.2 PCA

In the PCA projected case, it can be noted that the clustering overall seemed to be able to more accurately detect relevant clusters than the baseline case (see table 5.16). This can be seen in the fact that for all of the projected dimensions both the ARI and NMI scores exceeded the performance of the baseline. What can be observed here is that the results seem to deteriorate with more dimensions, performing the best when the data is projected to two dimensions. This seems to indicate that two principal components seem to be sufficient to capture the variance in the data, at least when clustering. For the 2-dimensional PCA projection, the result shows an average ARI score of 0.43 and an NMI score of 0.42. Interesting to note here is that there are three heat pumps that get a result of 0, namely heat pumps H8, H12 and H14 which lower the overall average. If these heat pumps are not taken into account, the average ARI increases to 0.58, indicating overall reasonable, while not perfect, agreement between the labels and the predicted clusters.

		PCA=2		PCA=5		PCA=10	
Id	|Error Codes|	ARI	NMI	ARI	NMI	ARI	NMI
H2	2	0.59	0.48	-0.03	0.07	-0.09	0.04
H3	3	0.08	0.11	-0.01	0.02	-0.02	0.01
H5	6	0.57	0.7	0.08	0.44	0	0
H7	3	0.41	0.27	0	0.16	-0.1	0.13
H8	2	-0.01	0	-0.01	0	0	0
H10	7	0.96	0.85	0.36	0.46	-0.07	0.25
H11	7	0.73	0.62	0.43	0.31	0.14	0.13
H12	2	-0	0	0.01	0.01	0	0
H14	4	0	0	0.02	0.02	0	0.01
H15	4	0.68	0.7	0.2	0.33	0.02	0.2
H19	6	0.22	0.44	0.28	0.37	0.05	0.27
H20	4	0.95	0.89	0.95	0.87	0.14	0.16
μ	4.2	0.43	0.42	0.19	0.26	0	0.2

Table 5.16: Clustering results for the PCA projected case, which shows the clustering results when projecting the data to 2,5 and 10 dimensions. Showcased are the heat pump id of the heat pump that the clustering was performed on, the number of unique error codes among the detected true positive points, as well as ARI and NMI scores for the detected clusters.

5.3.3 UMAP

In the UMAP projected case, two general trends can be seen (see table 5.17). Firstly, the UMAP projection does not seem able to accurately separate the different error codes; having close to random guess performance for both ARI and NMI scores for all projections. We can see that the average results are slightly better than the baseline, albeit not significantly. Secondly, we can see that the performance does not seem to change with increasing dimensions, indicating that the UMAP projection seemed to already achieve a distinct separation between different clusters with two dimensions but not in a successful manner.

5.3.4 Best Performing Configuration

When performing clustering on the best performing configuration, a general improvement may be observed over the baseline case, but it did not outperform the 2-dimensional PCA projection (see table 5.18). Here an average ARI and NMI score of 0.21 and 0.24, respectively, was achieved.

5.4 Error codes

Error code distribution and average true positive rate over all models and configurations is displayed in the two histograms in figure 5.9. Histogram (a) shows more errors due to the

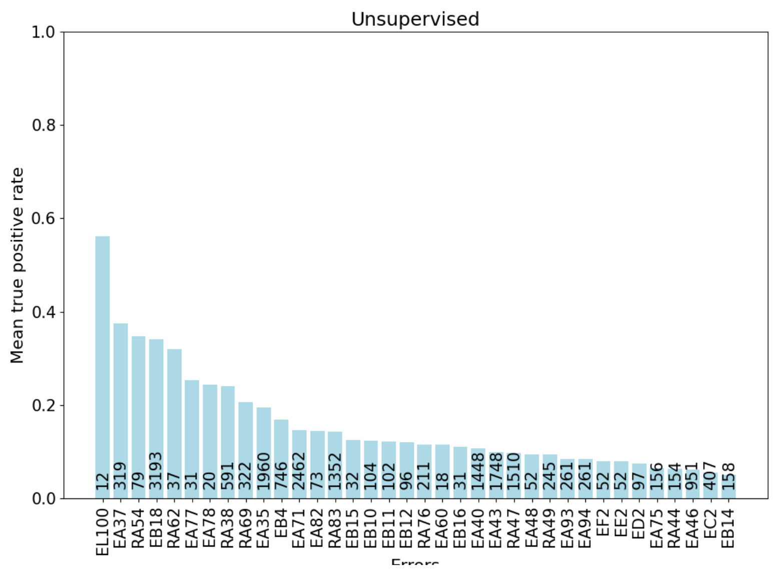
		UMAP=2		UMAP=5		UMAP=10	
Id	|Error Codes|	ARI	NMI	ARI	NMI	ARI	NMI
H2	2	0	0	0	0	0	0
H3	3	0	0	0	0	0	0
H5	6	0	0	0	0	0	0
H7	3	0	0	0	0	0	0
H8	2	0	0	-0.01	0	-0.01	0
H10	7	0	0	0	0	0	0
H11	7	0	0	0	0	0	0
H12	2	-0	0	0	0	0	0
H14	4	-0.01	0	0	0.01	0	0.01
H15	4	0.69	0.71	0.69	0.71	0.69	0.71
H19	6	0.29	0.38	0.29	0.38	0.29	0.38
H20	4	0.02	0.08	0.02	0.08	0.02	0.08
μ	4.2	0.08	0.10	0.08	0.10	0.08	0.10

Table 5.17: Clustering results for the UMAP projected case, which shows the clustering results when projected the data to 2,5 and 10 dimensions. Showcased are the heat pump id of the heat pump that the clustering was performed on, the number of unique error codes among the detected true positive points, as well as ARI and NMI scores for the detected clusters.

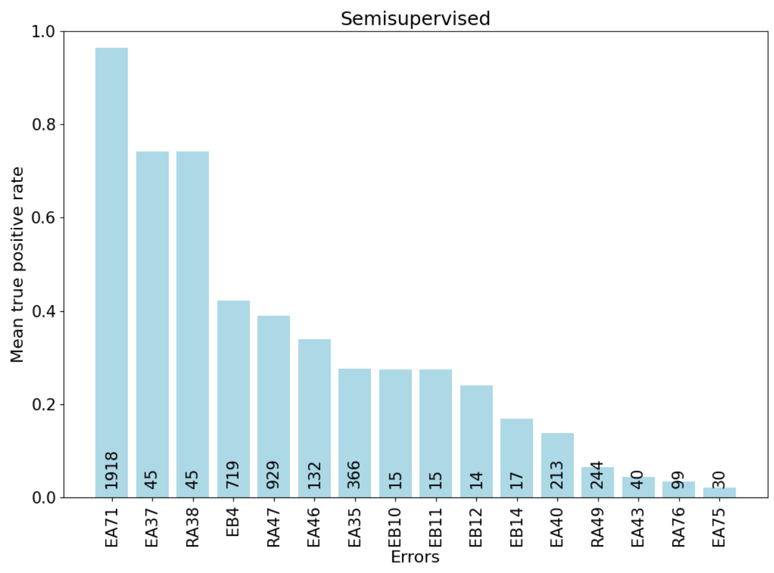
Id	|Error Codes|	ARI	NMI
H2	2	0.43	0.23
H3	3	0.09	0.12
H5	6	0.17	0.35
H7	3	0.3	0.34
H8	2	-0.01	0
H10	7	0.65	0.79
H11	7	0.67	0.52
H12	2	-0	0
H14	4	0.06	0.13
H15	4	0.12	0.14
H19	6	0.04	0.17
H20	4	0.02	0.1
μ	4.2	0.21	0.24

Table 5.18: Clustering results for the best performing configuration for each heat pump. Showcased are the heat pump id of the heat pump that the clustering was performed on, the number of unique error codes among the detected true positive points, as well as ARI and NMI scores for the detected clusters.

unsupervised models being evaluated on the entire dataset and inversely histogram (b) shows fewer due to the semisupervised models being evaluated on a validation set consisting of the final 20% of the data.



(a) Unsupervised anomaly detection models mean true positive rates of predicting error codes.



(b) Semisupervised anomaly detection models mean true positive rates of predicting error codes.

Figure 5.9: Histograms over the mean true positive rates of predictions matching error codes for all error codes appearing in the data, with number of appearances listed on each bar. Split into unsupervised and semisupervised due to the semisupervised case validating on the final 20% of the data.

Chapter 6
Discussion

Following is a discussion of the results presented in chapter 5 with the aim of answering the research questions, which have been formulated as follows:

How efficient are the selected anomaly detection methods at identifying data points corresponding to error codes in heat pump system data?

Can we use a clustering algorithm in order to be able to profile/group the predicted anomalies to find new error codes?

The first question is discussed in section 6.2, analysing the results of applying the six anomaly detection models chosen to our dataset and analysing the effects of different processing configurations on the input data. The second question is discussed in section 6.3, evaluating how well we were able to cluster confirmed error points in the dataset to determine the possibility to do so for false positive points in our data as well given access domain knowledge for determining the validity of the false positive points as indicators of actual erroneous behaviour. Results of particular interest are highlighted, by showing plots and reasoning for why the model and configuration worked particularly well or badly for that specific instance.

6.1 Error Code Distribution

When inspecting the distribution of error codes in the data we can clearly see in the histograms in section 5.9 that there is a large distribution of different error codes and a large variance in the frequency of the error codes. Some error codes, for example EB18, appear thousands of times while others appear very seldom, for example error code EA60 which only appears 18 times.

For both the unsupervised and semi-supervised models, we can see that the models fail to predict more than 20% of the appearing error codes in a majority of cases. It is notable however, that the semi-supervised models are able to successfully predict the error code EA71 with a mean true positive rate close to 100% and the error codes EA37 and RA38 with a mean true positive rate close to 80%. Further analysis shows that these error codes are mostly appearing for a single heat pump, and thus it would be important to gather more data to observe if this behaviour holds in the general case or if it is an isolated incident.

On the other hand, it can be observed that some error codes are poorly predicted, with true positive rates approaching zero for error codes EA43, RA76 and EA75 for the semi-supervised case.

In general, the wide variance of appearing errors shows the complexity of the system and consequently the difficulty in predicting the errors using a singular anomaly detection model. The positive cases indicate that there are possibilities for the anomaly detection models to perform well in a prediction task for some faults, and a more in depth research on those specific instances could be a potential further course of study. However, in the general case, the unsupervised anomaly detection algorithms showed lackluster performance.

6.2 Prediction Results

The general results of the prediction can be summarised with the aforementioned *no free lunch* theorem, which says that there is no single approach in unsupervised machine learning that will consistently show the best results. The errors simply express themselves in vastly different ways across different heat pumps, and there are too many factors that affect the heat pump and their behaviours. This in turn means that all of the tested anomaly detection method categories, namely the density-based, tree-based, reconstruction-based, statistics-based, and time-series-based methods, all had heat pumps and configurations for which they performed the best out of all of the models. Here we will attempt to explain why some of the models and configuration combinations performed well in some cases, while they show subpar performance in others.

6.2.1 Baseline

In the baseline case there are two specific cases of interest that we will focus on. The first consists of the results for heat pump H23, where we can see that the unsupervised models hallucinated across all models, performing worse than random guess (see table 5.2 for reference). What makes this heat pump particulary interesting is that the LSTM autoencoder here excelled with an MCC score of 0.87 (see table 5.2). The second case of interest is heat pump H6, where only deep isolation forest seemed to be able to detect anything at all with a MCC score of 0.38 while all other models had an MCC score of less than or equal to 0.

Starting with the first case which can be observed in figure 6.1 where a 1-dimensional PCA plot shows the results of the LSTM autoencoder on heat pump H23. Here we can see that the errors that are registered on the heat pump occur in the right-most segment of the plot, starting around the 20th of June and continuing for the remainder of the time-span. Here we can see that the registered values seem to be static, barely having any variation. While we can see a similar pattern between the 5th of June and the 20th of June, the pattern in that segment seems to have a higher variance.

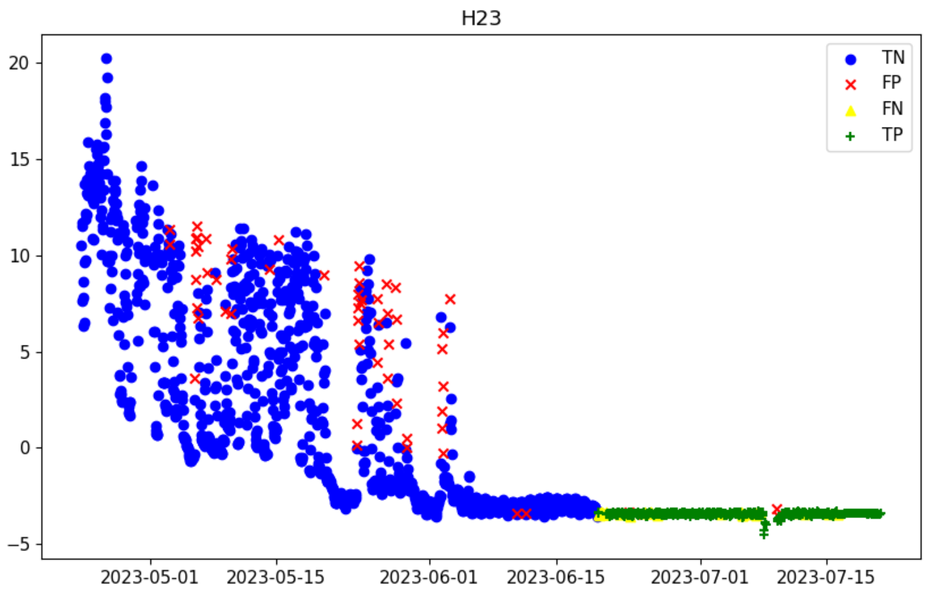
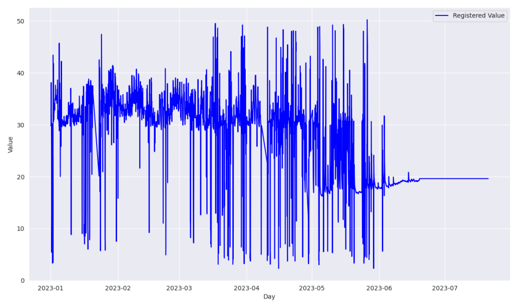
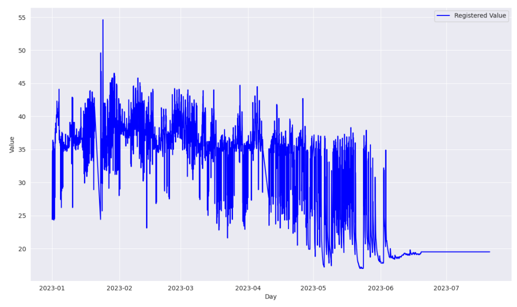


Figure 6.1: 1-dimensional PCA plot for heat pump H23



(a) Time-series plot of registered values for single feature



(b) Time-series plot of registered values for single feature

Figure 6.2: Plot showcasing two separate features which show stationarity

In figure 6.2 this behaviour can be observed, which shows two feature plots. Here, it can be seen that in the time-span starting on the 20th of June, the reported feature values stop changing and the remainder of the curve appears stationary. We can also see that where the stationarity occurs, whether being in a high or low region, seems to be inconsistent. While this is not the case for all features, this trend can be observed over the majority of them. This is a case where the heat pumps seem to exhibit a contextual anomaly, since the relative value of where the feature starts to become stationary is less important than the fact that the feature is itself stationary.

This is where we can see a clear advantage of the LSTM Autoencoder model for this specific case; the points themselves are only anomalous in relation to the time-frame that they appear in and the fact that they do not follow the pattern of the remainder of the data. This is a clear example of a collective anomaly, for which the LSTM Autoencoder seemed to outperform the other models.

Here it also becomes interesting to discuss the performance of the unsupervised models, which all have worse than random guess results, and here ECOD in particular sticks out with an MCC score of -0.4 and showcases a particular weakness of that model. The fact that ECOD classifies anomalies by calculating whether or not a point is on the tail-end of distributions makes this model particularly weak at detecting collective anomalies, since the features do not appear to be in statistically abnormal ranges.

While the remainder of the unsupervised models are not based on the same assumption as ECOD they also seem to suffer from hallucinations. This can be explained similarly, the points simply are not anomalous in a global context for this particular case, and thus all of the unsupervised models show subpar performance.

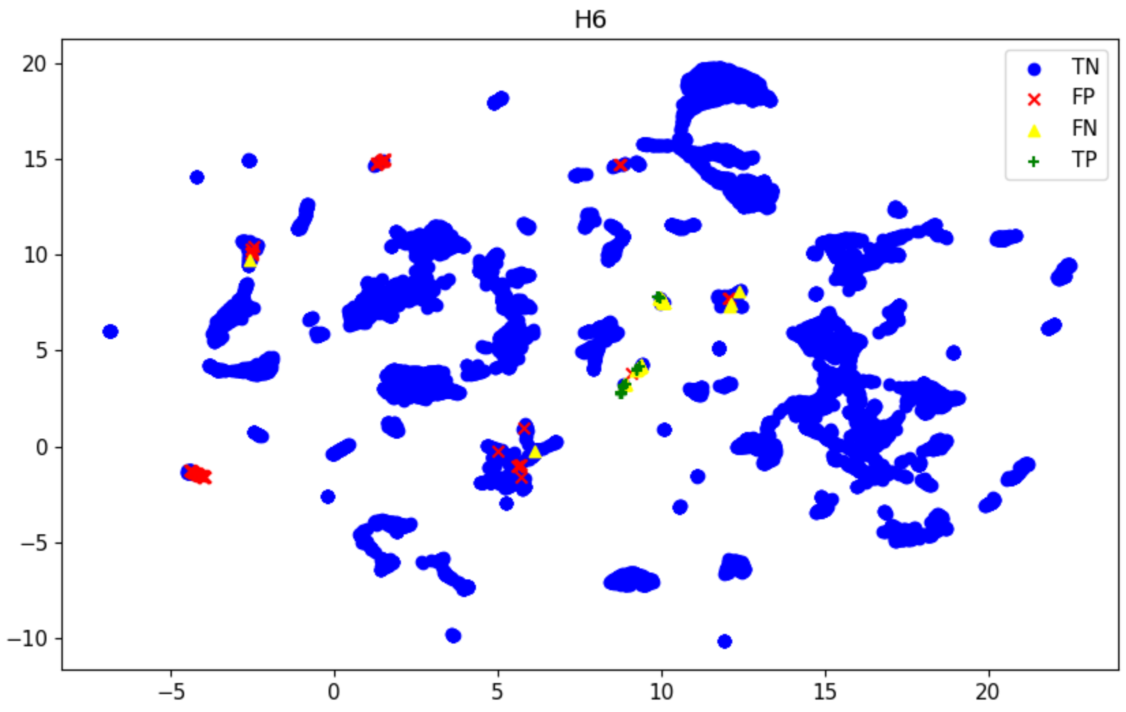


Figure 6.3: Results of the baseline configuration on the H6 heat pump for DIF model. The results are shown using a 2-dimensional UMAP projection.

Now for the second case, where we present the classification results projected to two dimensions using UMAP in figure 6.3. What may be observed here is that the data forms many different clusters, a vast majority of which consist of normal points. Furthermore, it may be observed that a majority of the errors (labeled TP and FN for true positive and false negative respectively), seem to appear in the two central clusters. While it is difficult to explain exactly why deep isolation forest outperforms the remainder of the models since it is not possible to observe more than two projected dimensions, we can still make assumptions based on the general advantages of deep isolation forest as outlined in section 3.4.4. The main advantage of the DIF model is the fact that it uses non-linear feature space transformations in order to isolate samples, which means that it generally can outperform other models which struggle to isolate points in the original feature space.

6.2.2 UMAP-Projected

The next configuration to be evaluated is the anomaly detection performed using a 10-dimensional UMAP projection. Here we can see when looking at table 5.14 that in the general case, the UMAP projection reduces the capability of the anomaly detection models to detect errors when comparing to the baseline. It can be seen that the average capability of the model to detect errors in the worst case decreases by 7 percentage-points, while in the best case it decreases by 1 percentage-point. Based on these results, the conclusion may be drawn that in the general case for this heat pump data set, a UMAP projection at least to 10 dimensions is not a viable method for feature extraction. A problem with using UMAP as a general method for feature extraction is that the quality of the projection is dependent on how well the parameters are chosen, as outlined in section 2.3.2. Thus it is possible that the performance of the models can be improved, but this would require fine tuning of parameters for each specific heat pump which is a time-consuming task.

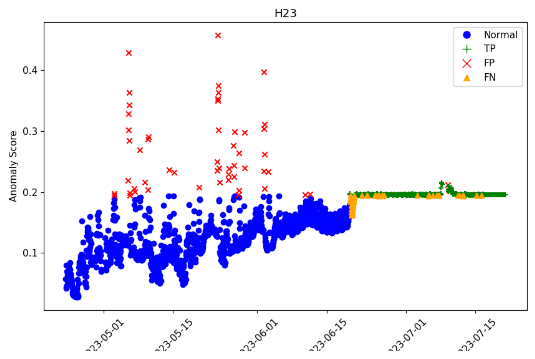
With that in mind, while in general the UMAP projection showed worse results than the baseline, it did manage to improve the MCC score slightly in relation to the basecase for the best performing model, from an MCC score of 0.87 to an MCC score of 0.9. Why this is the case can be seen in figure 6.4 which shows the assigned anomaly score for the baseline and UMAP projected data, respectively. Here it can be seen that a majority of the points that are labeled as false positives in the basecase are no longer classified as such. Additionally, it may be observed that the results for the UMAP projected data more clearly distinguished between normal and anomalous points. This can most likely be explained by the fact that the UMAP projection removed some of the occurring noise in the data, making the contextual anomalies at the end of the time-span more easily distinguishable.

Another interesting observation here is for heat pump H20, where the ECOD model outperformed the remaining unsupervised anomaly detection methods. The performance of the ECOD model was an MCC score of 0.25, while the remainder of the unsupervised models hallucinated with an average MCC score of -0.24. Why this is the case is difficult to explain due to the complexity of the data, as analysing where the point occurs on the Gaussian distribution for each feature is a time consuming task. That being said, a hypothesis for why the ECOD model outperformed the remaining models is that the UMAP projection made the points appear in the tail-end of the distributions for at least some of the UMAP features. If these points deviated statistically in a subset of the features while not appearing isolated globally, then ECOD might have been able to more accurately predict those points. However, more investigation is required to be able to determine if this is the case.

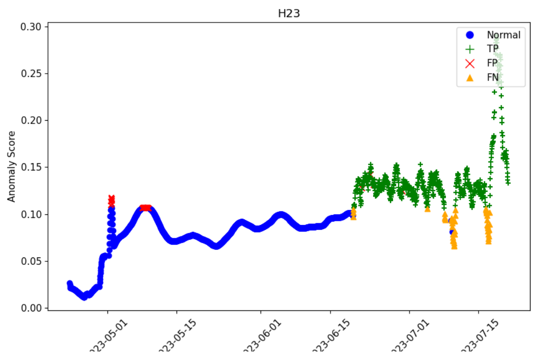
6.2.3 Feature-Selected

When performing feature selection the results indicate that the tree-based models are the best performing. As can be seen in table 5.6 Isolation Forest and Deep Isolation Forest have the best average MCC scores of 0.18 and 0.13, respectively. This is an increase of 0.13 for Isolation Forest and 0.12 for Deep Isolation Forest.

We see that the LSTM-Autoencoder seems to lose its ability to identify the contextual anomaly in H23 (see section 6.2.1 for further discussion of the anomaly) seeing that the MCC score decreases from 0.87 in the baseline(see table 5.3) to -0.52 in the feature selected configuration (see table 5.7). There may be many reasons for this, but the main hypothesis is



(a) Scatter plot showing assigned anomaly score for every point in the baseline configuration



(b) Scatter plot showing assigned anomaly score for every point in the umap configuration

Figure 6.4: Scatter plots showing the assigned anomaly score for each data point for heat pump H23 for the LSTM autoencoder model

that the feature selection may have been too restrictive, and thus removed crucial features affected by this specific erroneous behaviour.

One explanation for the improved performance of the tree-based methods is that they are susceptible to the curse of dimensionality and thus a reduction of the number of features is beneficial for their performance.

The general results of the feature selection indicate that proper feature selection might be an avenue worth exploring further due to the seen improvement in the average MCC score across most models.

6.2.4 Savitzky-Golay-Filtered

The Savitzky-Golay filtering showed varied results and only a negligible improvement over the basecase. The unsupervised model that improved the most was ECOD with just a 0.03 MCC score improvement (compare table 5.3 and table 5.8). When comparing the semisupervised models we can see that the Savitzky-Golay-filtered result was worse in the general case, where the average LSTM autoencoder MCC score dropped by 0.11, and the autoencoder result was essentially the same with a difference of just 0.01. What greatly affects the average for both the semisupervised models but especially the LSTM autoencoder in particular, is the performance on heat pump H23 which dropped from an MCC score of 0.87 to an MCC score of -0.53. A demonstration of the features of this heat pump were shown in figure 6.2, where it can be seen that the errors seem to be caused by the fact that the features start to become stationary where they should not. This means that in this case the stationarity itself is the deviating behaviour that the anomaly detection method should predict. However, since in this case the seasonal residual is removed, the whole feature space will appear almost stationary which makes this error region of the data more difficult to isolate.

That being said, if we look at table 5.13 we can see some instances of improvement. In particular we can see an improved true positive rate of 15 percentage-points for heat pump H11 for the ECOD model. This can be seen in figure 6.5 which shows a 1-dimensional PCA plot of the Savitzky-Golay filtered data for that heat pump. In this case we can see that the

Savitzky-Golay filtering seemed to have managed to capture that the error points tend to occur in an area that deviates from the norm, namely when observing the time-span between January 2023 and March the same year. We can however also see that ECOD seemed to have captured points on the other tail-end that does not seem to correspond to actual registered errors, with a high rate of false positives around February the same year. Generally speaking, it seems that what is determined to be an error in a heat pump is typically more complex than just being a statistical deviation.

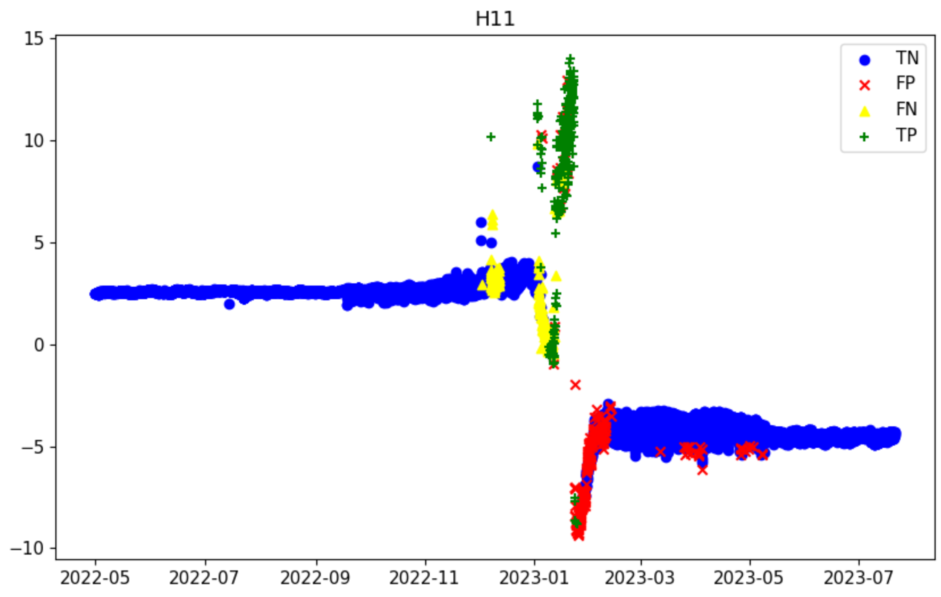


Figure 6.5: 1-dimensional PCA projection showcasing results for heat pump H11 for the ECOD model. The projection was performed on data with removed Savitzky-Golay residual.

6.3 Anomaly Clustering Results

The results for the anomaly clustering showcases that there is a possibility that utilising DBSCAN to cluster anomalies may be a viable option for attempting to discover novel error codes. This is corroborated by the fact that with the right data transformation, in this case a 2 dimensional PCA projection, it was possible to at least with an adequate performance be able to separate the errors into distinct clusters.

For the 2-dimensional PCA projection, as previously mentioned, we were able to achieve an average ARI score of 0.43, with a best case score of 0.95 and a worst case of -0.01. Here it is important to note that a perfect result is not required, since manual identification of the detected points would be required regardless. This is due to the fact that in order to classify these clustered anomalies as a novel error code, it would require careful analysis of what causes these errors to occur in the first place in order to provide a correct label and description to the error. These clusters may then be used as a guide, that can be used to identify and label similarly behaving outliers. These results by themselves may provide sufficient guidance to the domain experts to be able to identify new error codes.

In figure 6.6, we can see the visualisation of the best performing clustering, namely for H20, which was performed on a 2-dimensional PCA projection. Here we can see that the PCA was able to effectively separate the data points, which in turn means that the clustering was able to correctly group the detected errors.

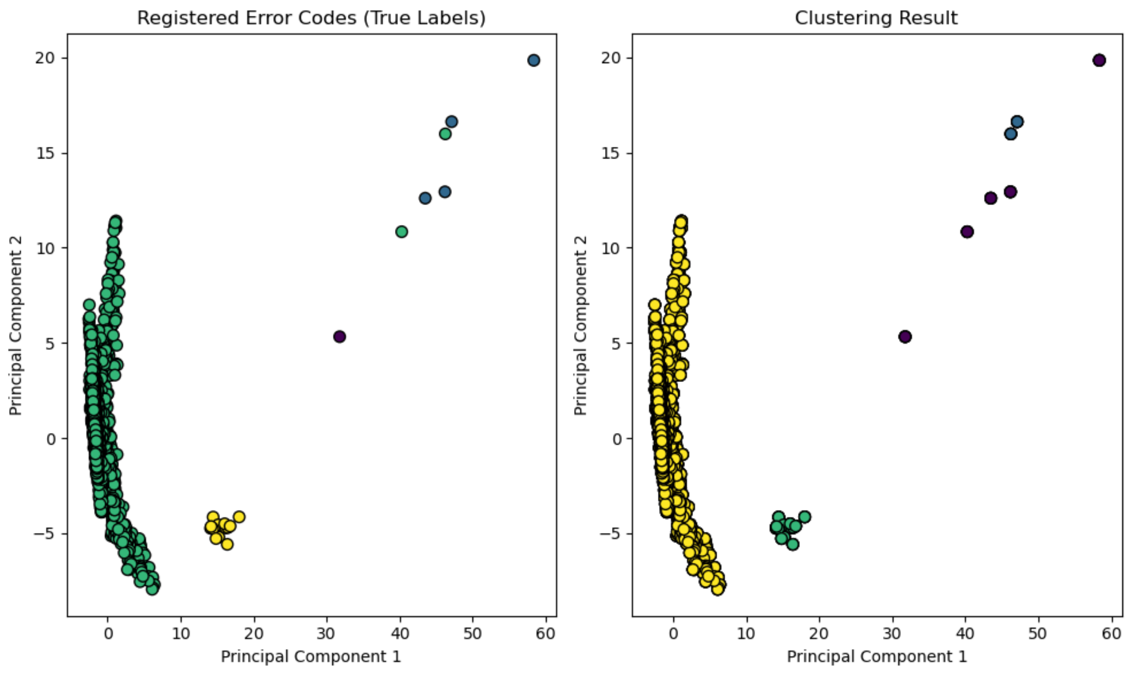


Figure 6.6: 2-dimensional PCA plot of the best performing clustering done on heat pump H20. Same color on a point indicates in the left plot that the points are of the same error code, while in the right plot it indicates that the points are placed in the same cluster.

In contrast, in figure 6.7 we can see the visualisation for the worst performing clustering. What may be observed here is that a majority of the errors correspond to a single error code, while the remaining points are not sufficiently separated. We hypothesise that the reason for this may be that the most occurring error code dominates when deriving the principal components for the PCA, meaning that which features are deemed to be important by the PCA projection is biased towards the most occurring error. This shows that projecting the data is not a perfect approach when clustering, but it did overall manage to identify meaningful clusters.

6.4 Limiting factors

Though some general observations and case by case observations can be made, the study is impacted by the relatively small sample size of data available, both in terms of data per heat pump, but also data of heat pumps of the same model. Would more data have been available, there is a chance that we would achieve more efficient mapping of the nominal state, and

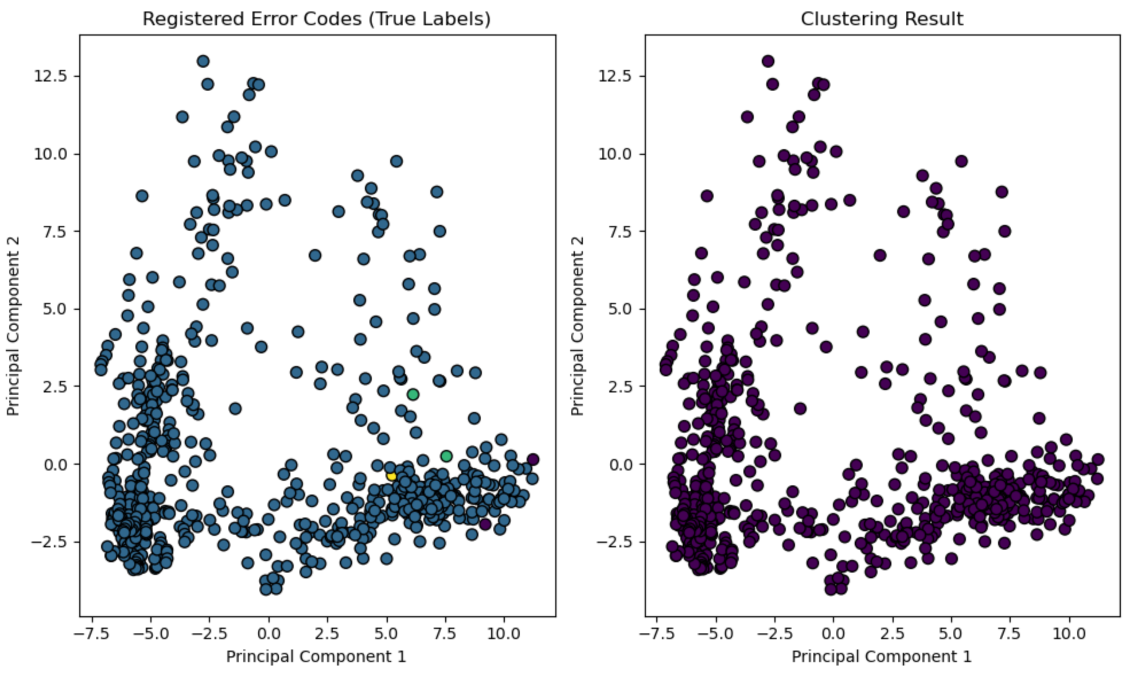


Figure 6.7: 2-dimensional PCA plot of one of the worst performing clustering done on heat pump H14. Same color on a point indicates in the left plot that the points are of the same error code, while in the right plot it indicates that the points are placed in the same cluster.

thus also better be able to capture possible erroneous behaviours. This is highlighted by the oftentimes wide discrepancies between results of different heat pumps, as described earlier in the discussion.

Furthermore, this thesis shows the limitations of a purely data-driven approach to unsupervised anomaly detection in multivariate time-series data. If more domain knowledge was available, the selection of appropriate features in feature selection and more accurate processing of the features would likely be possible. For example, knowing more about which error codes are caused by which components and features would open up the possibility to perform a more targeted study on detection of specific anomalies in the heat pumps. Additionally, without having access to domain knowledge, it is not possible to verify if points that were determined to be false positives are in fact false positives, or could instead be new potential non-labeled errors.

Lastly, with more computing power it would be feasible to predict anomalies on a lower time frame aggregate, which might have provided better performance. It would also open up the ability to analyse how early anomalies are detected before they trigger an error, which could be useful for predictive maintenance scenarios. More computing power would also allow for more hyperparameter testing of the anomaly detection methods. This would have been particularly interesting when investigating the neural network based models (DIF, AE, LAE), since their performance can vary highly depending on the chosen architecture. This thesis limited itself to a base configuration for each method and focused on exploring how processing of the input might affect the outcome.

Chapter 7
Conclusions & Further Work

In this chapter, the conclusions are presented which are based on the results shown in chapter 5 and the discussion in chapter 6. Lastly, we present potential for further work that can be done to improve the anomaly detection procedures based on findings discovered throughout the study.

7.1 Conclusions

This study corroborates the fact that proper preprocessing is a key factor in highly performing anomaly detection. The configurations studied in this thesis had a large impact on model performance and domain knowledge is likely required to further improve preprocessing. The general consensus for the anomaly detection is that there is no single classifier or preprocessing step that consistently produces accurate results. Essentially, it was concluded that even though the different heat pumps technically are the same type of system, when studying them purely as a data set, they can be regarded as completely different machines. The motivation for this conclusion is twofold. Firstly, the behaviour of the heat pumps is highly dependent on uncontrollable outside factors. This means that the exhibited behaviours in the data for the heat pump varies highly, both between different heat pumps but also for the specific heat pump depending on machine state and both daily and yearly seasonality. Secondly, the errors that the heat pumps exhibit are highly varied. They are varied in multiple ways, mainly in the fact that they can have many different causes since heat pumps are complex machines, but also in the way they show themselves in the data with both contextual and point-wise expressions.

That being said, there were cases where we managed to identify errors with a moderate to high degree of accuracy. The general observation was that which model and configuration could accurately identify errors varied between different heat pumps and what errors they experienced. For each tested anomaly detection model and configuration we managed to find a scenario where their combination outperformed the rest. Some observations about specific models was that the LSTM autoencoder was especially capable at identifying collective anomalies, but was also the most sensitive to how the data was transformed. We could also see that generally, the tree-based methods (i.e isolation forest and deep isolation forest) performed the best on average, with particular cases where deep isolation forest was able to identify errors where the remaining methods performance could be amounted to random guess. Out of the tested methods local outlier factor seemed to perform the worst on average, with its best case performance being close to random guess which shows that perhaps at least this density-based method is not a suitable anomaly detection algorithm for the task.

While we managed to find examples of results where each configuration outperformed the others, generally speaking we could see that the best average performance was on the feature selected data. This indicates that it may be possible to perform purely data driven anomaly detection, but having access to domain expertise and thus knowing which features are important can be crucial if the goal is to build a consistently reliable anomaly detector.

It is also interesting to note both when and why the feature extraction (UMAP) and feature selection methods seemed to perform the best. The general observation here is that feature extraction seemed to perform well when the errors expressed themselves as collective anomalies in the data. We believe that this is due to the fact that the feature extraction captures the general trend in the data, allowing especially the LSTM-autoencoder to highly effectively capture collective anomalies. This may also be why the feature extraction did not perform well in the general case, probably due to the fact that the projection may have lessened noise in the data, which made it harder for the models to find point anomalies. Generally speaking, feature selection seemed to perform slightly better, most likely due to the fact that having less features reduces the curse of dimensionality, allowing the methods to more easily identify anomalous points.

To answer the first research question, namely *How efficient are the selected anomaly detection methods at identifying data points corresponding to error codes in heat pump system data?* is that the performance of the methods is highly varied and depends on both the heat pump itself but also on which errors that heat pump exhibits. The general conclusion is that there is no single method or preprocessing technique out of the ones that were tested that consistently show positive results. Thus if we did not have access to ground-truth, which typically is the case when performing unsupervised anomaly detection, we most likely would not be able to find a suitable configuration which shows the general difficulty with performing unsupervised machine learning on complex machines.

For the second question, namely *Can we use a clustering algorithm in order to be able to profile/group the predicted anomalies to find new error codes?* the answer is both yes and no. While we could not identify new error codes due to lack of access to domain experts, we could see that we managed to get at least an adequate average clustering performance when using the ground-truth for verification. This indicates that there is a possibility that if the clustering was done on the points that were identified as false positives that the found clusters could correspond to new potential error codes. However, since we were unable to verify this by actually finding new error codes this answer remains inconclusive.

7.2 Further Work

Firstly, this work was limited by a purely data-driven approach to a real world dataset and the poor performance shows the difficulty of the task. Further work would likely benefit from including human expertise in the loop. In particular in the feature selection and pre-processing steps where more knowledge about the system in question likely would allow for better transformation of the input features to make them more conductive to showing clear deviations when faults appear. Secondly, with the help of domain experts it would be feasible to get a domain expert evaluation of the detected novel anomalies and verify whether they actually correspond to a faulty behaviour in the heat pump and thus can be labeled as a new error code. Thirdly, data for both a longer timespan and more heat pumps and coupled with domain expertise opens up an interesting possibility to more accurately model the nominal state of the heat pumps. This would likely aid the performance of the LSTM-Autoencoder which showed some promise, but was limited by computing power and continuous data. Lastly it would be interesting to investigate how much better performance in the prediction task can be achieved when performing hyperparameter tuning on the models, something that was beyond the time scope of this thesis.

References

[1] A. Beghi, R. Brignoli, L. Cecchinato, G. Menegazzo, M. Rampazzo, and F. Simmini. Data-driven fault detection and diagnosis for hvac water chillers. *Control Engineering Practice*, 53:79–91, 2016.

[2] Mohammed Ayalew Belay, Sindre Stenen Blakseth, Adil Rasheed, and Pierluigi Salvo Rossi. Unsupervised anomaly detection for iot-based multivariate time series: Existing solutions, performance analysis and future directions. *Sensors*, 23(5), 2023.

[3] Samit Bhanja and Abhishek Das. Impact of data normalization on deep neural network for time series forecasting. https://arxiv.org/abs/1812.05519, 2019.

[4] Jiahao Bian, Rafal Scherer, Marcin Wozniak, Pengchao Zhang, and Wei Wei. Abnormal detection of electricity consumption of user based on particle swarm optimization and long short term memory with the attention mechanism. *IEEE Access*, 9:47252–47265, 01 2021.

[5] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, may 2000.

[6] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1), 2020.

[7] Andrew A. Cook, Göksel Mısırlı, and Zhong Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, 2020.

[8] Chris Kuo/Dr Dataman. Handbook of Anomaly Detection with Python Outlier Detection — (4) Isolated Forest. https://towardsdatascience.com/use-the-isolated-forest-with-pyod-3818eea68f08, February 2023. (Accessed on: 2023-12-01).

[9] Jae Seok Do, Akeem Bayo Kareem, and Jang-Wook Hur. Lstm-autoencoder for vibration anomaly detection in vertical carousel storage and retrieval system (vcsrs). *Sensors*, 23(2):1009, January 2023.

[10] Pablo Duboue. *The art of feature engineering: essentials for machine learning*. Cambridge University Press, 2020.

[11] Chris Ernst. chriswernst/dbscan-python. https://github.com/chriswernst/dbscan-python, February 2022. (Accessed on: 2023-12-04).

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining*, 1996.

[13] Steven Flores. Variational autoencoders are beautiful. https://www.compthree.com/blog/autoencoder/. (Accessed on: 2023-12-03).

[14] Ellen Friedman. New approaches to anomaly detection. http://radar.oreilly.com/2014/07/new-approaches-to-anomaly-detection.html. (Accessed on: 2023-12-03).

[15] Walter Grassi. *Heat Pumps: Fundamentals and Applications*. Springer International Publishing, 2018.

[16] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. ADBench: Anomaly detection benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[17] Nick Hotz. What is crisp dm? https://www.datascience-pm.com/crisp-dm-2/, Jan 2023. (Accessed on: 2023-12-05).

[18] Svend Hylleberg. *Seasonality in regression*. Academic Press, 2014.

[19] Kevin Leahy, R. Lily Hu, Ioannis C. Konstantakopoulos, Costas J. Spanos, and Alice M. Agogino. Diagnosing wind turbine faults using machine learning techniques applied to operational data. *2016 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–8, 2016.

[20] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Comput. Surv.*, 50(6), dec 2017.

[21] Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu, and George Chen. Ecod: Unsupervised outlier detection using empirical cumulative distribution functions. https://doi.org/10.48550/arXiv.2201.00382, January 2022.

[22] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

[23] Yanping Liu, Bo Dang, Yue Li, Hongbo Lin, and Haitao Ma. Applications of savitzky-golay filter for seismic random noise reduction. *Acta Geophysica*, 64(1):101–124, February 2016.

[24] Paul Barham Eugene Brevdo Zhifeng Chen Craig Citro Greg S. Corrado Andy Davis Jeffrey Dean Matthieu Devin Sanjay Ghemawat Ian Goodfellow Andrew Harp Geoffrey Irving Michael Isard Yangqing Jia Rafal Jozefowicz Lukasz Kaiser Manjunath Kudlur Josh Levenberg Dandelion Mané Rajat Monga Sherry Moore Derek Murray Chris Olah Mike Schuster Jonathon Shlens Benoit Steiner Ilya Sutskever Kunal Talwar Paul Tucker Vincent Vanhoucke Vijay Vasudevan Fern a Viégas Oriol Vinyals Pete Warden Martin Wattenberg Martin Wicke Yuan Yu Xiaoqiang Zheng Martín Abadi, Ashish Agarwal. TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/, 2015. (Accessed on: 2023-12-05).

[25] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. https://arxiv.org/abs/1802.03426, 2020.

[26] Jiawei Miao, Haicheng Tao, Haoran Xie, Jianshan Sun, and Jie Cao. Reconstruction-based anomaly detection for multivariate time series using contrastive generative adversarial networks. *Information Processing Management*, 61(1):103569, 2024.

[27] Mohsin Munir, Steffen Erkel, Andreas Dengel, and Sheraz Ahmed. Pattern-Based Contextual Anomaly Detection in HVAC Systems. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 1066–1073, New Orleans, LA, November 2017. IEEE.

[28] Timothy J. O'Shea, Seth Hitefield, and Johnathan Corgan. End-to-end radio traffic sequence recognition with recurrent neural networks. In *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 277–281, 2016.

[29] Isaac Ritharson. P, Santhosh S J, Avinash V, and Magesh C Achari. Feature engineering and machine learning for iot-based applications: An overview of algorithms. In *2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, pages 663–668, 2023.

[30] Jeffrey Pattavina. *An Introduction Heat Pumps*. June 2023. https://www.researchgate.net/publication/363255409_An_Introduction_Heat_Pumps. (Accessed on: 2023-11-18).

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[32] Sebastian Raschka. Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software*, 3(24), April 2018.

[33] Durgesh Samariya and Amit Thakkar. A comprehensive survey of anomaly detection algorithms. *Annals of Data Science*, 10(3):829–850, November 2021.

[34] Lory Seraydarian. What is a confusion matrix in machine learning? https://plat.ai/blog/confusion-matrix-in-machine-learning/, March 2023. (Accessed on: 2023-12-09).

[35] Jonathon Shlens. A tutorial on principal component analysis. *CoRR*, abs/1404.1100, 2014.

[36] Samuel C. Sugarman. *HVAC fundamentals*. Fairmont Press, 2005.

[37] Rüdiger Wirth and Jochen Hipp. Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, volume 1, pages 29–39. Manchester, 2000.

[38] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[39] Hongzuo Xu, Guansong Pang, Yijie Wang, and Yongjun Wang. Deep isolation forest for anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12591–12604, 2023.

[40] Ka Yee Yeung and Walter Ruzzo. Details of the adjusted rand index and clustering algorithms supplement to the paper "an empirical study on principal component analysis for clustering gene expression data" (to appear in bioinformatics). *Science*, 17, January 2001.

[41] Junhai Zhai, Sufang Zhang, Junfen Chen, and Qiang He. Autoencoder and its various variants. In *2018 IEEE international conference on systems, man, and cybernetics (SMC)*, pages 415–419. IEEE, 2018.

[42] Liangwei Zhang, Jing Lin, and Ramin Karim. Adaptive kernel density-based anomaly detection for nonlinear systems. *Knowledge-Based Systems*, 139:50–63, 2018.

[43] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2024-01-18

EXAMENSARBETE Finding Known and Novel Errors in Heat Pumps Using Unsupervised ML
STUDENTER Maks Epsteins, Felix Forsström
HANDLEDARE Jacek Malec (LTH), Vilhelm Åkerström (Bosch)
EXAMINATOR Elin A. Topp (LTH)

Usch vad kallt det är?! Kan data avslöja varför värmepumpen inte fungerar?

POPULÄRVETENSKAPLIG SAMMANFATTNING **Maks Epsteins, Felix Forsström**

Värmepumpar är komplexa system vars fel kan uttrycka sig på flertalet sett i värmepumpens data. Detta arbete undersökte möjligheterna att identifiera dessa fel med hjälp av maskininlärning, samt möjligheterna att identifiera nya fel genom att gruppera datapunkter med liknande egenskaper.

Värmepumpar har blivit en viktig del av vår tillvaro. Världen över används värmepumpar för att öka bekvämligheten i hem och andra byggnader, genom att reglera temperatur och luftkvalité. Värmepumpar fungerar genom en samverkan av flera olika komponenter, med både rörliga och statiska delar, vilka över tid kan ge upphov till fel.

Att undersöka dessa fel är både dyrt och tidskrävande. För att underlätta denna process har flertalet tillverkare implementerat loggningsfunktionalitet som kontinuerligt sparar mätvärden från maskinens IoT sensorer.

Detta möjliggör användning av den registrerade datan för att upptäcka fel med hjälp av datadrivna maskininlärnings metoder. Detta examensarbete undersökte först möjligheten att upptäcka fel genom maskininlärningsmetoder och sedan huruvida det gick att gruppera fel genom en klusteringalgoritm. Då syftet var att upptäcka befintliga och nya fel, så användes oövervakade anomalidetektion och klustringsmetoder, det vill säga metoder som använder omärkt data.

För att utföra detta testades 4 olika dataförbehandlingsmetoder, och 6 olika anomalidetektionsmetoder som alla har olika styrkor och svagheter. Det visade sig att eftersom

olika fel uttrycker sig annorlunda mellan olika värmepumpssystem så varierade även metodernas förmåga att upptäcka fel samt vilka förbehandlingsmetoder som gav positiva resultat.

I det bästa fallet uppnåddes en klassifieringsprestation med ett MCC värde på 0.92 (nästan perfekt klassifiering), och i det värsta ett värde på -0.52 (mycket sämre än en slumpmässig gissning) vilket visar på svårigheterna med oövervakad maskininlärning. I det generella fallet hittades ingen metod som lyckades identifiera en majoritet av felen. Klustringen visade bättre resultat överlag, där vi för majoriteten av värmepumparna lyckades få positiva resultat.