UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

University of Padua

School of Engineering

Department of Information Engineering

Master's Degree in Computer Engineering

Curriculum: Artificial Intelligence and Robotics

Master Thesis

Computational Approaches for Anomaly Detection in Heat Pump Units

Supervisor:
Fabio Vandin

Candidate:
Alberto Ursino

Academic year: 2023/2024

Graduation date: July 3rd, 2024

Abstract

The thesis investigates the feasibility of introducing anomaly detection in the context of heat pump units. This research work has been proposed by Swegon Operations Srl. with the ultimate goal of creating an anomaly detection system that helps experts predict anomalies in heat pump units before they occur. The effect of this new system would be fewer machines breaking down due to unreported anomalies, resulting in increased overall reliability. Consequently, the benefits would include reduced maintenance costs and higher customer satisfaction.

To evaluate the feasibility of using anomaly detection in heat pump units, we followed a data-centric approach: rather than focusing on finding the most suitable anomaly detection technique, we concentrated on creating a high-quality dataset. We then experimented with two machine learning techniques — principal component analysis (PCA) and clustering — to find useful insights about the data and to design an algorithm for detecting outliers.

To test the performance of the models, we needed to manually introduce an anomaly into the machine. We reduced the gas in the refrigeration circuit, thus simulating a gas leak. We called this experiment the "Gas leakage experiment".

Using PCA, which is a dimensionality reduction technique, we transformed the dataset from 8 to 2 dimensions. The 2D plots did not reveal any well-defined patterns that could aid in the anomaly detection task. On the other hand, PCA allowed us to derive useful information about the dataset and refine the data preprocessing step. Furthermore, we noticed that already in 2 dimensions, the points related to the gas leakage anomaly were significantly separated from the points associated to the normal operating conditions of the machine.

The results obtained with clustering showed that applying machine learning techniques that work on the original dimensional space, thus considering all the features present in the dataset, is an effective approach to this use case. Based on clustering, we created the Outlier Detection Algorithm, which is designed to classify safe and anomaly points in the dataset. With an F_1 *score* of 0.97, the latter methodology

strongly suggests that anomaly detection has the potential to be used in heat pump units.

Abstract (Italian)

La tesi studia la fattibilità di introdurre il rilevamento delle anomalie nel contesto delle unità a pompa di calore. Questo lavoro di ricerca è stato proposto da Swegon Operations Srl. con l'obiettivo finale di creare un sistema di rilevamento delle anomalie che aiuti gli esperti a prevedere le anomalie nelle unità a pompa di calore prima che si verifichino effettivamente. L'effetto di questo nuovo sistema sarebbe una riduzione delle macchine che si guastano a causa di anomalie non segnalate, con un conseguente aumento della affidabilità complessiva. I vantaggi includerebbero una riduzione dei costi di manutenzione e una maggiore soddisfazione dei clienti.

Per valutare la fattibilità dell'uso del rilevamento delle anomalie nelle unità a pompa di calore abbiamo seguito un approccio incentrato sui dati: piuttosto che concentrarci sulla ricerca della tecnica di rilevamento delle anomalie più adatta, ci siamo concentrati sulla creazione di un dataset di alta qualità. Abbiamo poi sperimentato l'utilizzo di due tecniche di apprendimento automatico — l'analisi delle componenti principali (PCA) e il clustering — per trovare informazioni utili sui dati e per progettare un algoritmo per il rilevamento degli outlier.

Per testare le prestazioni dei modelli, è stato necessario introdurre manualmente un'anomalia nella macchina, riducendo il gas nel circuito frigorifero e simulando così una perdita di gas. Abbiamo chiamato questo esperimento "spillamento del gas".

Utilizzando la PCA, che è una tecnica di riduzione della dimensionalità, abbiamo trasformato il dataset da 8 a 2 dimensioni. Dai grafici 2D non abbiamo rivelato pattern ben definiti che potessero aiutare nel processo di rilevamento delle anomalie. D'altra parte, la PCA ci ha permesso di derivare informazioni utili sul dataset e di affinare lo step di preprocessamento dei dati. Inoltre, abbiamo notato che già in 2 dimensioni i punti relativi allo spillamento del gas si separavano significativamente dai punti relativi al corretto funzionamento della macchina.

I risultati ottenuti con il clustering hanno mostrato che l'applicazione di tecniche di apprendimento automatico che lavorano sullo spazio dimensionale originale, con-

siderando quindi tutte le variabili presenti nel dataset, è un approccio efficace per questo caso d'uso. Basandoci sul clustering, abbiamo creato l'Outlier Detection Algorithm, che è progettato per classificare i dati nel dataset come anomalie o dati "safe". Con un F_1 *score* di 0.97, quest'ultima metodologia suggerisce fortemente che il rilevamento delle anomalie ha il potenziale per essere usato nelle unità a pompa di calore.

List of Figures

List of Tables

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior [1]. Anomaly detection is critical in the smart industry for preventing equipment failure, reducing downtime, and improving safety [2]. In this thesis, "anomaly detection" is intended as an "intelligent" anomaly detection, which uses artificial intelligence techniques to identify complex patterns or outliers in data that traditional approaches cannot detect.

Heat pump units not equipped with an intelligent anomaly detection system usually rely solely on traditional alarms, which only take into account a few sensors, and they typically operate with rule-based methods. For this reason, they may struggle to detect complex anomalies that are indeed associated with complex patterns in the data.

Equipping heat pump units with an intelligent anomaly detection system would lead to many advantages:

- Powered by artificial intelligence, the system can identify anomalies that traditional methods may miss. This leads to more accurate and consistent detection of anomalies;
- By analyzing data in real-time, the system can predict potential equipment failures before they actually happen. This prevents minor issues from escalating into major problems, thereby reducing the risk of costly incidents;
- The system can scale with the growth of industrial operations: as the volume and complexity of data increase, it can adapt and continue to provide effective capabilities;

- AI systems continuously learn and adapt from new data, improving their detection accuracy over time. This results in a system that is easily updatable and increasingly powerful as time passes;

- Implementing cutting-edge technology can provide a competitive advantage by optimizing operations, reducing costs, and improving product quality.

1.2 Research scope

The main objective of this study is to investigate the feasibility of using anomaly detection in heat pump units. The research first focuses on collecting and analyzing data generated by the sensors installed on a heat pump machine, and then on experimenting with two machine learning techniques.

1.3 State of the art

Industrial units suffer damage due to continuous usage and normal wear and tear. Such damage needs to be detected early to prevent further escalation and losses. The data in this domain is usually referred to as sensor data because it is recorded using different sensors and collected for analysis. Anomaly detection techniques have been extensively applied in this domain to detect such damage [3].

However, this thesis does not aim to research the most suitable anomaly detection technique for our task or to optimize existing methodologies. Instead, it addresses a gap in the current literature by examining the feasibility of applying anomaly detection specifically to industrial machines which implement the heat pump technology.

1.4 Thesis overview

The thesis is organized as follows:

Chapter 2. Provides essential theoretical background. It begins with an introduction to heat pumps, followed by a theoretical description of the anomaly detection task. It further covers the two machine learning techniques used in the experiments (principal component analysis and clustering) and defines the performance metrics we used for model evaluation.

Chapter 3. Details the methodologies used in this research work. First, it presents the core strategy we followed. Then, it describes each step of the dataset preparation stage and the designed training and validation sets. In the end, it provides the specifications of the machine learning models employed.

Chapter 4. Shows and analyzes the results obtained from the experiments with PCA and clustering. It also presents the Outlier Detection Algorithm, which we developed in order to perform the binary classification task.

Chapter 5. Concludes the thesis with a summary of the research results, and with considerations on the use of anomaly detection in the world of heat pumps.

The realization of this thesis was possible thanks to a three-month internship at Swegon Operations Srl. Their involvement was pivotal in acquiring the essential knowledge about heat pumps, needed for conducting this research. Having access to on-site machinery data was nevertheless of highly importance; it enabled us to prepare the datasets, validate the models, and consequently yield tangible results.

Chapter 2

Background and Fundamental Concepts

2.1 Heat pumps

Heat pumps are devices designed to transfer thermal energy from one location to another thanks to thermodynamic processes. What we call "heat pump units" are HVAC (Heating, Ventilation, and Air Conditioning) systems that implement the heat pump technology. They are commonly used for space heating and cooling in residential, commercial, and industrial applications. Heat pump units are generally costlier to install than other heating systems such as furnaces working with coal, natural gas, fuel oil, and so on and electrical heaters. However, when the long-term use period is considered, they can provide a huge amount of savings. They can also play a critical role in the electrification of heating purposes. When the electricity is met by renewables, the heating demand can be met in a more sustainable way [4].

They operate by utilizing mechanical work to move heat against its natural flow, extracting it from a source (surrounding air, the ground, or nearby water sources) and delivering it to a target area. The thermodynamic cycle of heat pumps can be reversed, therefore they can be used for both cooling and heating purposes.

Heat pump technologies are widely used for upgrading ambient heat from sustainable sources, such as air, water, the ground, and waste heat, to heating temperatures [5]. Due to this variety of heat sources, we can find many typologies of heat pump units. Among the most common ones are Air Source Heat Pumps (ASHP), Ground Source Heat Pumps (GSHP) and Water Source Heat Pumps (WSHP).

The unit employed for experiments in this work is an ASHP, more precisely an

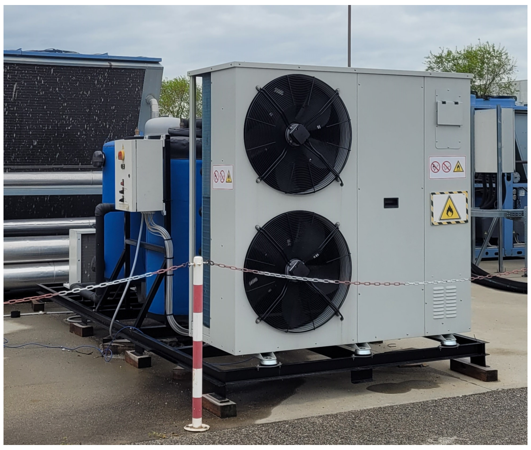


Figure 2.1: Heat pump unit by Swegon Operations Srl. used for experiments

Air-to-Water Heat Pump (Figure 2.1). This typology of machine, in heating mode, extracts heat from the outside air and uses it to heats water, which then circulates within the indoor environment. In chiller (or cooling) mode, it does the opposite: the unit absorbs heat from the indoor water-system and releases it to the outside as heated air.

2.2 Anomaly detection

2.2.1 Definition

An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism [6]. The goal of the anomaly detection task is to find unusual patterns or outliers in data. By establishing a baseline of normal behavior through training on historical data, anomaly detection algorithms can then be used for identifying patterns or outliers that do not conform to the expected behavior.

Anomaly detection is a crucial area engaging the attention of many researchers. It is useful in many real time applications such as industry damage detection, detection of fraudulent usage of credit card, detection of failures in sensor nodes, detection of abnormal health and network intrusion detection [7]. It plays a crucial role in enhancing system reliability, security, and efficiency by identifying irregularities that may go unnoticed through traditional monitoring methods.

2.2.2 Challenges

At an abstract level, an anomaly is defined as a pattern that does not conform to expected normal behavior. A straightforward anomaly detection approach, therefore, is to define a region representing normal behavior and declare any observation in the data that does not belong to this normal region as an anomaly [3]. But several factors make this apparently simple approach very challenging:

- Defining a normal region that encompasses every possible normal behavior is very difficult. In addition, the boundary between normal and anomalous behavior is often not precise. Thus, an anomalous observation that lies close to the boundary can actually be normal, and vice versa;
- In many domains, normal behavior keeps evolving and a current notion of normal behavior might not be sufficiently representative in the future;
- Availability of labeled data for training/validation of models used by anomaly detection techniques is usually a major issue;
- Often the data contains noise that tends to be similar to the actual anomalies and hence is difficult to distinguish and remove.

Due to these challenges, the anomaly detection problem, in its most general form, is not easy to solve. In fact, most of the existing anomaly detection techniques solve a specific formulation of the problem. The formulation is induced by various factors such as the nature of the data, availability of labeled data, type of anomalies to be detected, and so on. Often, these factors are determined by the application domain in which the anomalies need to be detected. Researchers have adopted concepts from diverse disciplines such as statistics, machine learning, data mining, information theory, spectral theory, and have applied them to specific problem formulations.

2.2.3 Example: anomaly detection in 2 dimensions

This example is intended just to give a visual understanding of what is meant by an anomaly in the data. Let's imagine to have a (dummy) machine with two sensors, *sensor_1* and *sensor_2*, which respectively produce periodically and synchronously temperature and pressure values. So, for each instant in time in which the machine is operating, we have a pair of temperature-pressure values. When the machine is running properly, *sensor_1* generates values in the range of -10 to 10 C° while *sensor_2* generates values in the range of -5 to 5 bar.

Experiment 1. *Let's get the machine running and in the meantime collect some data. After a while, the machine is switched off and all the data is plotted. What is obtained is the plot in Figure 2.2. As can be seen, the machine worked as expected. All points are within the nominal ranges. Some are slightly outside, but in this case we decide that this is acceptable. It is possible to interpret this "cloud" of points as a safe zone: every point that lies inside this cloud is a "safe" point.*

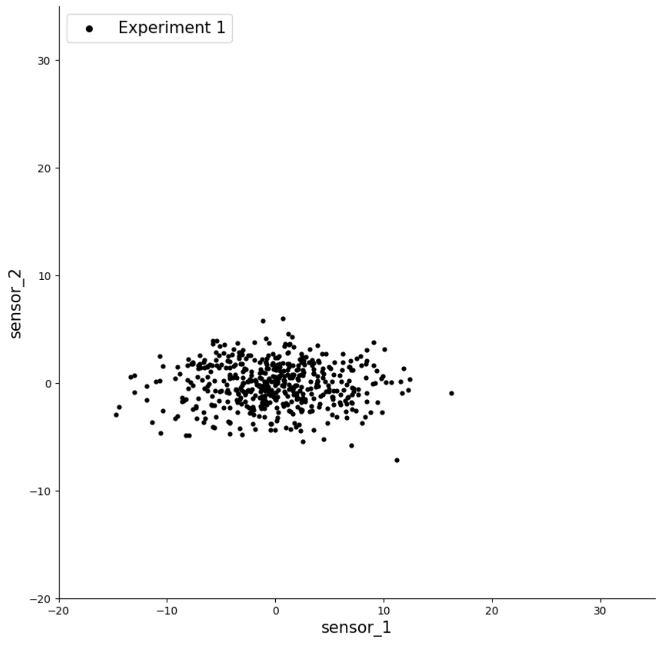


Figure 2.2: Example: safe data points from experiment 1

Experiment 2. *Now let's repeat the experiment, but before starting the machine, let's introduce an anomaly in it. Suppose the sensors now produce higher temperature and pressure values because of this anomaly. Let's get the machine running for a while and collect data. Finally, we obtain the plot in Figure 2.3. The new points (in red) are all outside the safe zone. These new points are anomaly points because they are far away from the safe cloud.*

The anomaly points in this example can be identified using a simple rule-based method. However, our goal was to emphasize that anomalies can also be detected visually. This is because they may exhibit patterns different from expected ones, or data points could be situated far outside the normal operational zone. Moreover, this example is intended to give a hint of how anomaly detection works in a high-dimensional spaces, where there are more safe clouds and outliers are harder to detect.

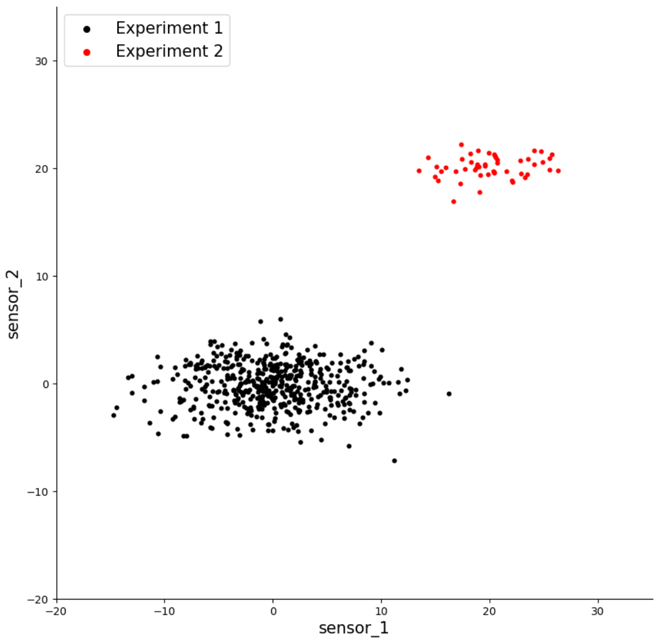


Figure 2.3: Example: safe and anomaly data points respectively from experiment 1 and experiment 2

We will resume this example in the subsequent chapters. Specifically, we will apply the same machine learning techniques we used for the real experiments to this example.

2.3 Machine learning methods

2.3.1 Principal component analysis

Introduction

Principal Component Analysis (PCA) [8] is a dimensionality reduction technique commonly used to transform a high-dimensional dataset into a lower-dimensional representation while retaining as much of the original variability as possible. The fundamental idea behind PCA is to identify the principal components of the data, which are a new set of uncorrelated variables that result from a linear transformation of the original variables. These components are found in such a way as to capture the maximum variance in the original data.

There are several reasons to reduce the dimensionality of data. One of the main reasons for using PCA is that high-dimensional data imposes computational challenges. Moreover, PCA can enhance data interpretability, reveal meaningful structures, and aid in illustration. Given our high-dimensional datasets, we decided to apply this technique to better understand the data.

In the context of anomaly detection, PCA is also used along with spectral anomaly detection techniques, which try to find an approximation of the data using a combination of attributes that capture the bulk of the variability in the data. The general approach adopted by spectral anomaly detection techniques is to determine such subspaces (embeddings, projections, etc.) in which the anomalous instances can be easily identified [3].

Formal definition

The following definition of PCA is taken from the book "Understanding Machine Learning" by Di Shai Shalev-Shwartz, Shai Ben-David (2014) [9]. Let $\mathbf{x}_1, ..., \mathbf{x}_m$ be m vectors in $\mathbb{R}^d$. We would like to reduce the dimensionality of these vectors using a linear transformation. A matrix $W \in \mathbb{R}^{n,d}$, where $n < d$, induces a mapping $\mathbf{x} \mapsto W\mathbf{x}$, where $W\mathbf{x} \in \mathbb{R}^n$ is the lower dimensionality representation of $\mathbf{x}$. Then, a second matrix $U \in \mathbb{R}^{d,n}$ can be used to (approximately) recover each original vector $\mathbf{x}$ from its compressed version. That is, for a compressed vector $\mathbf{y} = W\mathbf{x}$, where $\mathbf{y}$ is in the low dimensional space $\mathbb{R}^n$, we can construct $\tilde{\mathbf{x}} = Uy$, so that $\tilde{\mathbf{x}}$ is the recovered version of $\mathbf{x}$ and resides in the original high dimensional space $\mathbb{R}^d$. In PCA, we find the compression matrix W and the recovering matrix U so that the total squared distance between the original and recovered vectors is minimal; namely, we aim at solving the problem

$$\operatorname*{arg\,min}_{W \in \mathbb{R}^{n,d}, U \in \mathbb{R}^{d,n}} \sum_{i=1}^{m} \|\mathbf{x}_i - UW\mathbf{x}_i\|_2^2 \tag{2.3.1}$$

Back to the example of Section 2.2.3

To better visualize what PCA consists of and how it can be used for anomaly detection, the example presented in the Section 2.2.3 is resumed. PCA is applied to the whole dataset to reduce its dimensionality from 2 to 1 dimension. The Python library *scikit-learn* [10] is used, which provides a powerful API to perform various machine learning tasks.

After applying PCA and plotting the transformed data points, what is obtained is the plot in Figure 2.4. The blue data points represent the original dataset projected into one dimension. In other words, they represent the principal component obtained by linearly combining the two original variables. The most interesting observation is that, even in one dimension, we have two separate dense regions of points: the one on the left is the safe zone and the one on the right is associated to the anomaly event.

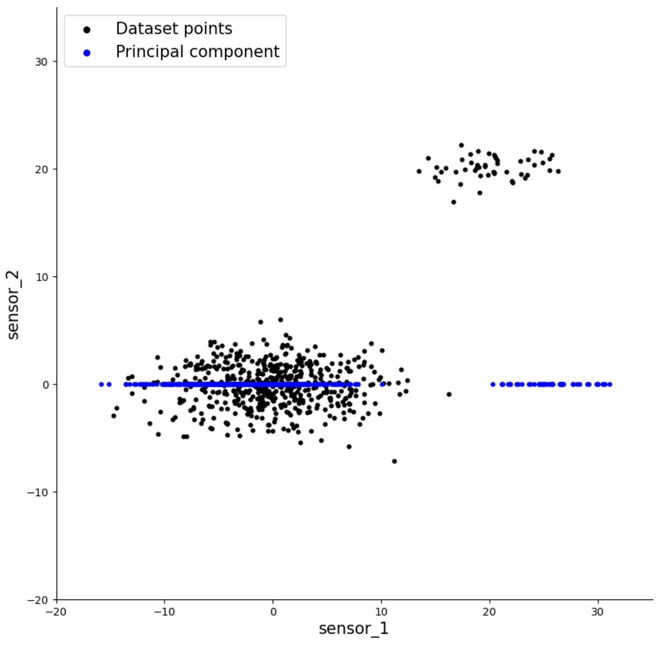


Figure 2.4: Example: safe and anomaly data points, and their corresponding PCA-transformed representations

2.3.2 Clustering

Introduction

Clustering [11] is a machine learning technique that aims to organize data points into groups, where each group contains data points that are more similar to each other than to those in other groups. It is commonly used in unsupervised learning frameworks because it aims to find patterns and structure in data without predefined labels.

Clustering for anomaly detection is widely used in various fields, including network intrusion detection [12], credit card fraud detection [13], image processing domain [14], and anomalous topic detection in text data [15].

Some clustering algorithms used for anomaly detection, like DBSCAN [16], are based on the assumption that normal data instances belong to a cluster in the data, while anomalies do not belong to any cluster. The latter approach is the one we adopted in the experiments with clustering presented in Chapter 4.2, although we did not use DBSCAN but the K-means algorithm.

Formal definition

As for the PCA definition, the following formal definition of clustering is taken from the same book [9]. Clustering tasks can vary in terms of both the type of input they have and the type of outcome they are expected to compute. For concreteness, we shall focus on the following common setup:

Input - a set of elements, X, and a distance function over it. That is, a function $d : X \times X \mapsto \mathbb{R}_+$ that is symmetric, satisfies $d(x, x) = 0$ for all $x \in X$ and often also satisfies the triangle inequality. Alternatively, the function could be a similarity function $s : X \times X \mapsto [0, 1]$ that is symmetric and satisfies $s(x, x) = 1$ for all $x \in X$. Additionally, some clustering algorithms also require an input parameter k (determining the number of required clusters).

Output - a partition of the domain set X into subsets. That is, $C = (C_1, ..., C_k)$ where $\cup_{i=1}^{k} C_i = X$ and for all $i \neq j, C_i \cap C_j = \varnothing$. In some situations the clustering is "soft", namely, the partition of X into the different clusters is probabilistic where the output is a function assigning to each domain point, $x \in X$, a vector $(p_1(x), ..., p_k(x))$, where $p_i(x) = \mathbb{P}[x \in C_i]$ is the probability that x belongs to the cluster C_i. Another possible output is a clustering dendrogram, which is a hierarchical tree of domain subsets, having the singleton sets as its leaves, and the full domain as its root.

Back to the example of Section 2.2.3

To conclude the example, we apply clustering using the K-means algorithm. Again, we use *scikit-learn* to fit a model on the whole dataset, which includes both safe and anomaly points. We decide to run the algorithm with $k = 2$ in order to try to separate the safe from the anomaly points in two distinct clusters. What we get is the plot in Figure 2.5 and, as can be seen, the safe and anomaly points are indeed grouped separately.

2.3.3 Performance metrics

In the context of machine learning, performance metrics serve as quantitative measures that allow the assessment of the quality and accuracy of machine learning models. The evaluation of model performance is essential for understanding the model effectiveness in solving a particular task. Performance metrics play a pivotal role in guiding the development and selection of models based on their ability to

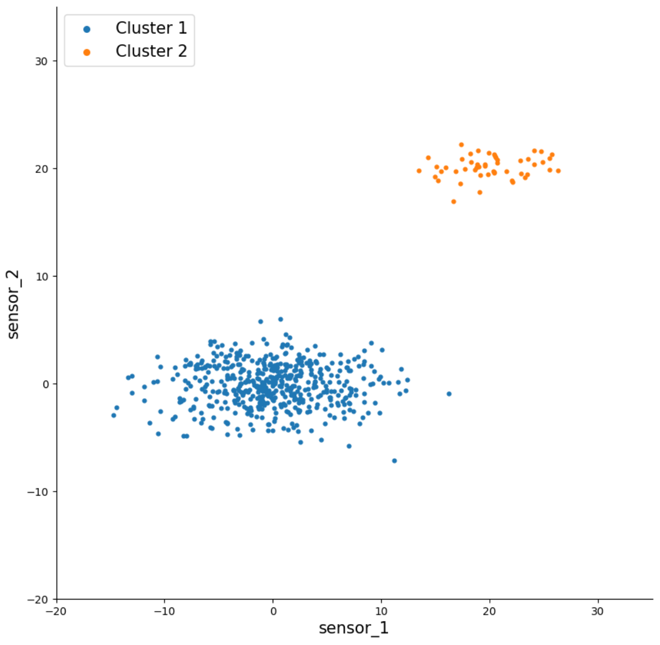


Figure 2.5: Example: safe and anomaly data points split in two groups with clustering

make accurate predictions. Since we used them in the experiments, we will provide their definitions to make them clear to the reader.

In this project, we are working with models that perform a binary classification task: a sample can be considered a safe point or an anomaly point. For this reason, the definitions of the performance metrics are given in the context of binary classification. Note that, in our case, we chose the anomaly points as to be the positive instances, and the safe points the negative instances. We followed this reasoning: since the model goal is to identify anomaly points, whenever it identifies one, it is a positive instance.

Accuracy - proportion of correctly classified instances out of the total instances.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3.2}$$

where TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative. It answers the question: "Of all the instances the model attempted to predict, how many did it get right?". In an intuitive sense, accuracy gives a quick overall score of how well the model is doing. However, it is essential to note that accuracy might not be the best metric in all situations, especially when dealing with imbalanced datasets (where one class is significantly more prevalent than the other). Suppose the anomaly event is rare and therefore not as often seen in the dataset: if the model always classifies a point as negative, it will achieve high accuracy even if it cannot detect the anomaly event at all.

Precision - ratio of true positive predictions to the total positive predictions.

$$precision = \frac{TP}{TP + FP} \tag{2.3.3}$$

It answers the question: "Of all the instances that the model predicted as positive, how many were actually positive?". Precision focuses on the accuracy of the positive predictions.

Recall - ratio of true positive predictions to the total actual positive instances.

$$recall = \frac{TP}{TP + FN} \tag{2.3.4}$$

It answers the question: "Of all the instances that were actually positive, how many were identified by the model?". Recall focuses on the model's ability to capture and correctly predict all the positive instances in the dataset. If the model always classifies a point as positive and there is a rare anomaly event in the dataset, the recall will be 100% but the precision will be very low, that is why, with unbalanced datasets, we should consider bot recall and precision.

F_1 score - harmonic mean of precision and recall.

$$F_1\,score = \frac{2 * precision * recall}{precision + recall} \tag{2.3.5}$$

It ranges from 0 to 1, with higher values indicating better performance. $F_1\,score$ is a useful metric for measuring the performance of classification models in the presence of imbalanced datasets because it takes into account both the recall and the precision. A high $F_1\,score$ generally indicates a well-balanced performance, demonstrating that the model can concurrently attain high precision and high recall.

2.4 Thermodynamics concepts

2.4.1 Refrigeration cycle

Thermodynamic cycles are thermodynamic transformations in which the initial and final states of the system are coincident. They are used in thermodynamic machines to transform heat into work or vice versa. If during the cycle the machine produces work, $W > 0$, the machine is called a thermal machine; if, on the other hand, the net work of the cycle is negative, $W < 0$, the machine is called a refrigeration machine. A thermodynamic machine consists of a substance, that is the thermodynamic system that performs the transformations, and the environment with which the substance exchanges energy in the form of heat and work [17].

Heat pumps operate as refrigeration machines. A visual representation illustrating a refrigeration machine and its heat exchanges with the environment is depicted in Figure 2.6. It operates by extracting heat Q_C from a cold source (e.g., the inside of a refrigerator) by cooling it further, using external work W, and releasing heat Q_H into the environment (e.g., the outside of the refrigerator). In order to do so, heat pumps come with a refrigeration cycle, which enables the exchange of heat by changing the thermodynamic states of a substance. The refrigeration cycle implemented by heat pumps generally involves the use of a refrigerant, that is, the substance that undergoes phase changes (from liquid to vapor and back) to permit the absorption and release of the heat.

It is worth mentioning that the most efficient type of refrigeration cycle when operating between two temperature reservoirs is the Carnot refrigeration cycle. Essentially, it represents the most efficient way a machine can work between two different temperatures. However, since this cycle sets a theoretical benchmark for maximum efficiency, real-world refrigeration systems employ practical cycles based on different thermodynamic principles. Among these, the most commonly used is the vapor compression cycle, which includes an evaporator, compressor, condenser, and expansion valve.

2.4.2 Mechanical vapor compression refrigeration circuit

The majority of cooling systems are based on the vapor compression refrigeration cycle [18]. Indeed, the heat pump unit studied in this work implements the mechanical vapor compression refrigeration circuit. A general representation of this circuit is shown in Figure 2.7. It serves as the backbone of traditional cooling systems, controlling the process of thermodynamic reactions in order to regulate

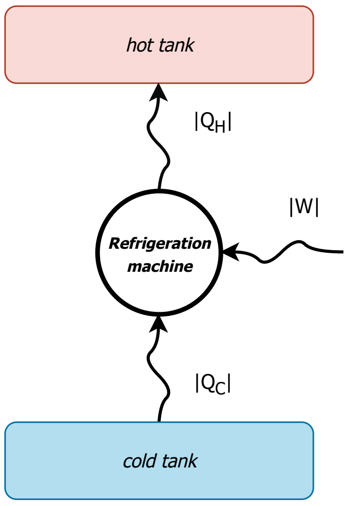


Figure 2.6: Symbolic representation of a refrigeration machine and its thermodynamic exchanges

temperatures effectively. Through the interplay of key components, including the evaporator, compressor, condenser, and expansion valve, the system achieves the extraction and transfer of heat, facilitating the creation of controlled and comfortable environments. At its core, the circuit employs a refrigerant which undergo controlled phase changes, altering its thermodynamic state between liquid and vapor. The refrigerant has suitable thermodynamic properties that enable it to circulate through the pipelines that connect the system components [19].

Explained in the following is the “journey” the refrigerant makes in the circuit to allow the heat exchange. The refrigerant entering the evaporator is a cold, partial liquid-vapor mixture. It is brought into indirect contact with the fluid to be cooled, it absorbs heat Q_L and gradually changes to a saturated vapor state. In the pressure-hentalphy diagram (Figure 2.8) the refrigerant moved from point A to point B. The vapor then enters the compressor which compresses it using work W, bringing it to a high temperature and pressure (B $\rightarrow$ C). After the compression, the refrigerant has absorbed heat Q_C. Note that it is because of the work W that the refrigerant is driven through all the various components and thus change state continuously. The super-heated vapor is then sent to the condenser, where it is first cooled and then the change of state to liquid takes place. This process results in the transfer of the heat $Q_H = Q_L + Q_C$ to a colder external fluid. This is followed by a subcooling of the saturated liquid, in which the refrigerant further reduces its temperature (C $\rightarrow$ D). The liquid, still at high pressure, passes through the expansion valve, dropping the pressure to a value close to that of evaporation (D $\rightarrow$ A). In this way, it changes from a subcooled liquid to a moist vapor with a high

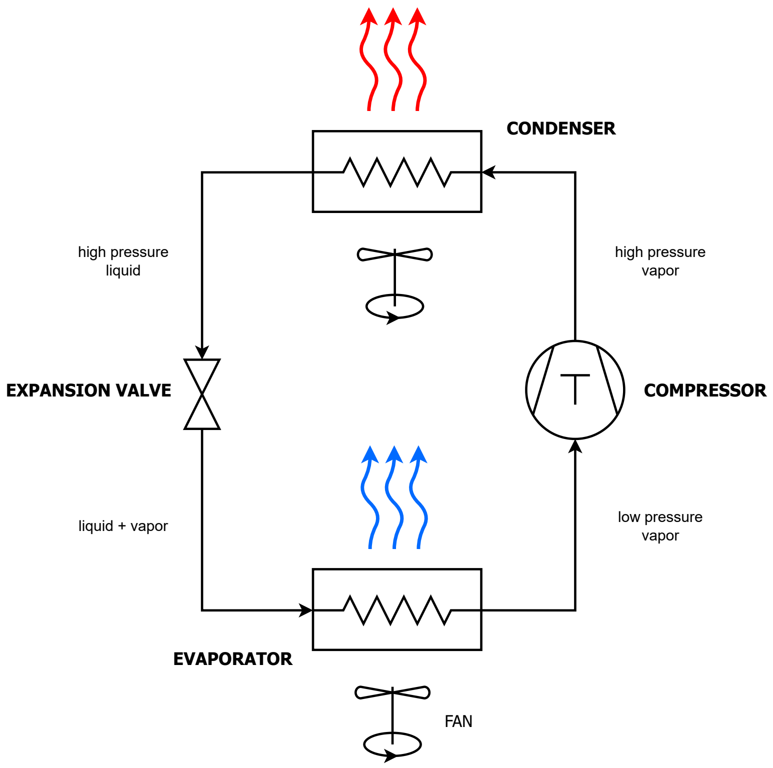


Figure 2.7: Symbolic representation of a mechanical vapor compression refrigeration circuit

liquid density. The refrigerant is ready to re-enter the evaporator, continuing the refrigeration cycle.

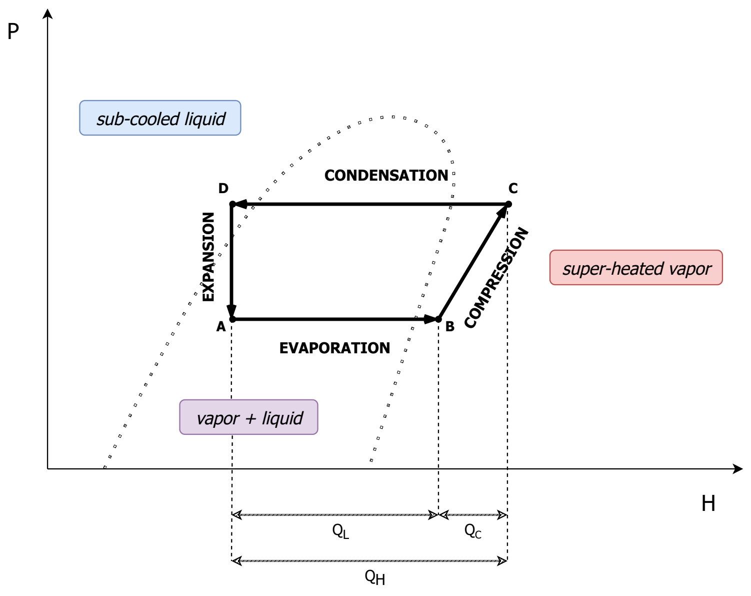


Figure 2.8: Vapor compression refrigeration cycle on the p-h diagram

Chapter 3

Methodology

3.1 Core strategy

The raw data available to us is not labelled, in the sense that we do not know at prior if a certain data point corresponds to an anomaly. However, we are confident that all the data collected refer to instances where the heat pump unit was working properly. Therefore, the strategy we followed was to train a model exclusively on the normal operating conditions of the machine, so that it raises an alarm whenever the sensor measurements deviate from its learned standards.

The strategy explained above is a semi-supervised approach: it assumes that the training data has labeled instances only for the normal class. Since semi-supervised techniques do not require labels for the anomaly class, they are more widely applicable than supervised techniques. For example, in spacecraft fault detection [20], an anomaly scenario would signify an accident, which is not easy to model. The typical approach used in such techniques is to build a model for the class corresponding to normal behavior, and use the model to identify anomalies in the test data [3].

In order to obtain samples of sensor measurements related to anomalies and thereby evaluate the models, we manually introduced an anomaly event into the heat pump unit. We referred to this as the "Gas leakage experiment" which we will discuss in Chapter 3.2.4. Data labeling was performed based on our confidence that the machine was malfunctioning exclusively during the time of the introduced anomaly.

3.2 Dataset preparation

All the datasets have the data stored in chronological order. Samples are stored following the ascending order. Hence, in the first row we have the oldest data point in time and in the last row the most recent one. It is worth mentioning that the datasets contain time series data, and we exploited this property in some experiments, in particular the ones presented in Chapter 4.2.2 and Chapter 4.2.3.

3.2.1 Data collection

In any machine learning research project, the process of data collection stands as a fundamental task, especially when the objective is to detect anomalies from the data itself. Anomalies, as already stated in Chapter 2.2.1, represent deviations from expected behaviors; thus, in order to increase the chances of identifying them, we need to have a good-quality dataset representing the correct state of operation of the heat pump unit. The machine learning models are trained on this dataset to understand the functioning of the heat pump unit and so to identify unusual patterns or outliers. In order to create a good-quality dataset, we decided to focus on three main aspects: the quantity, reliability, and variability of the data.

Quantity, reliability, and variability

We are going to see that the training dataset contains samples from 8 different variables. Therefore, due to its high-dimensional shape, we need to deal with the curse of dimensionality problem. In data analysis, the term refers to the difficulty of finding hidden structure when the number of variables is large [21]. The key point to be aware of is that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse. Hence, in our case, in order to obtain reliable results it is essential to have a sufficiently large amount of data available.

By the term “reliability” of the data, we refer to the fact that we want to have a clean and consistent dataset. For example, we want to discard all those samples that correspond to incorrect sensor readings, or measurements that we are not interested in and that would just “confuse” the model. Thus, filters and other operations are applied to prepare a dataset that is as much as possible representative of the correct state of operation of the heat pump unit. In Chapter 3.2.2 we will discuss in detail all the preprocessing steps we applied.

Through an initial analysis with domain experts, we found that the heat pump unit under study can vary significantly its working state. For instance, when the external air temperature changes, the machine adapts, changing significantly the sensor measurements. Therefore, a good model should be trained using samples collected across various configurations of the external environment to ensure its effectiveness and adaptability to different real-world conditions. Due to the limited amount of data available, we decided to focus the research on studying the heat pump unit during a specific season of the year. We will explain this in more detail in Chapter 3.2.2.

Data source

The data used for this research comes from the different sensors installed on the heat pump unit by Swegon Operations Srl. (Figure 2.1). The unit has several sensors: the exact number is not given, but it is possible to define more than 40 variables. Some of them represents a single sensor reading, others represent a combination of different sensor readings. All sensors generate numeric values that represent different types of measures, comprising pressure, temperature, percentage, and binary values.

Data is pulled from a cloud storage through the use of an API. Each sensor automatically registers its measurements on the cloud every time the new measured value differs from the previous one by a specific threshold. As a result, *raw_dataset*, that is the initial version of the dataset (see Table 3.1), is characterized by a significant presence of missing values, accounting for 82.07% of the entire dataset. For instance, it is common that, at a certain time t, we can extract measurements from just few sensors, because the other sensor measurements are almost identical to the ones registered at time $t-1$, $t-2$, and so on. However, given this mechanism of how data is registered on the cloud, we can easily estimate the missing values. In Chapter 3.2.2 we will discuss more in detail the filling method we exploited.

After pulling the data, we proceeded to store it in both CSV and Parquet file formats. We extensively used the Python library *Pandas* to manage the datasets [22].

At this point, *raw_dataset* contains time-series data obtained from measurements of 12 variables, which are called "features" in the context of machine learning. It contains data from an entire year, for a total of 4 369 673 samples. We only have one year of data because of how the cloud storage is set up. At this stage, the dataset is composed by purely raw data, which means we still need to process it before using it to train models.

3.2.2 Data preprocessing

Data preprocessing is a common step that involves preparing and cleaning data before it is fed into a machine learning model for training. The quality of the data and how well it is preprocessed can significantly impact the performance and accuracy of the model.

The data preprocessing step can comprise many methods. Those that we found necessary to apply are the data filling, data cleaning, feature selection and feature scaling methods. So now, given *raw_dataset*, we are going to apply them to build a reliable dataset.

Data filling

Data filling is an important task to perform when dealing with missing values. Models trained on datasets with lots of missing values will most likely exhibit reduced performance than expected due to the incomplete nature of the input data. Undefined values may be misinterpreted by the model, because it is unable to estimate the missing data in the best way possible, or more simply, because it does not have the ability to estimate them.

Knowing how the data is registered on the cloud, the best strategy to follow is to fill a missing value, for a given feature, with the most recent previous value available. We also imposed the condition to avoid having missing values in the dataset. Therefore, in order to properly fill *raw_dataset*, it is necessary at first to identify the row from which to start the filling method. We cannot start to fill the dataset from the first row, unless it does not contain any missing values. Suppose that, in the first row, we have one feature that has a missing value: it is easy to conclude that is impossible to fill consistently that value if there are no previous values (in time) available for that feature. Thus, we decided to cut *raw_dataset*, removing all the rows that cannot be filled, starting from the first row. Then, the filling method is applied. In order to practically accomplish this: for the cutting part we created an algorithm, and for filling the missing values we used the function *fillna* by *Pandas* with the method “ffill”.

After the filling process step, we obtained *dataset_v1* (see Table 3.1) that contains 4 369 317 rows of values. It has 356 fewer samples than *raw_dataset*, because we cut some not fillable rows. After implementing the filling strategy described earlier, the dataset now contains no missing values. The subsequent step involves cleaning the dataset.

Data cleaning

Data cleaning refers to the process of identifying and correcting errors, inconsistencies, and inaccuracies in the dataset. In our case, we decided to remove from the dataset all sensor measurements that we assumed referred to moments in which the heat pump unit was stabilizing. For example, when the heat pump unit is starting up, we get completely different measurements to those obtained a few minutes later; this is because, before reaching its state of mechanical equilibrium, the machine goes through a start-up phase. Together with the domain experts, we identified and reported all these type of events and subsequently removed all the related points from the dataset.

We defined a set of rules, each corresponding to such an event, and applied them to filter the values in the dataset that match these rules. We created three main rules: the 1^{st} rule is used to remove points related to the starting phase of the heat pump unit, in particular all the data points n seconds after the compressor is switched on. The 2^{nd} rule is related to shutdowns of the machine: all the data points n seconds before a shutdown happen (compressor is switched off) are removed. The 3^{rd} rule is related to the change of the status of the heat pump unit. The unit used for experiments can be configured on several statuses. The most common ones are the "chiller" and the "heater" status. When the setting is on "chiller", the machine cools indoor spaces, while "heater" keeps them warm. Every time the unit changes its status, it undergoes a stabilization phase, that can lead to the production of noisy measurements. Accordingly, the 3^{rd} rule removes all points n seconds after a status change.

In this preprocessing step, we also removed two features from the dataset, named "Compressor 1/1" and "Status". The feature "Compressor 1/1" indicates whether the heat pump unit has the compressor switched to on or off. This variable basically tells us whether the machine is running or not. We decided to keep only the dataset rows with "Compressor 1/1" equals to 1 (compressor on) because when it is inactive, the machine does not produce any relevant measurement. The feature "Status" was removed because we decided to focus the training of the model on one specific status. The chiller and heater statuses greatly change the behavior of the machine, and therefore also the sensor measurements. In order to decrease the complexity, we decided to focus the research exclusively on the chiller status. Thus, we filtered the dataset in such a way as to keep only the points associated with it.

In the Table 3.1 the resulting dataset after this preprocessing step is *dataset_v2*.

There is a greatly reduced number of samples ($4\,369\,317 \rightarrow 1\,485\,720$) due to both the filtering rules and features removal. Additionally, now we have just 10 features; a number that we tried to reduce even more with the next step.

Feature selection

Feature selection is a common task in machine learning applications that eliminates irrelevant and redundant features and improves learning performance [23]. As we said previously, we have 40 or more variables available from which to extract data. This high number of variables certainly poses the need to perform a feature selection analysis. Many features may not be relevant for the anomaly detection task, and others may be redundant, resulting in very inefficient model training if used all of them. In order to carry out the feature selection task, we first consulted the domain expert knowledge and subsequently applied an algorithm for finding highly correlated features.

To get to the initial 12 features included in *raw_dataset*, the domain experts were assessed to exclude from the set of available variables those they thought did not provide useful information for the identification of anomalies. After all the preprocessing steps described above, a feature selection algorithm from *Pandas*, named *corr*, was applied to identify and discard highly-correlated features. This method computes the pairwise correlation of columns. We specifically chose to utilize the Pearson correlation coefficient, which measures the degree of the linear dependency between two features.

The resulting dataset, that is *dataset_v3* (see Table 3.1), comprises 8 features, on which the machine learning experiments detailed in Chapter 4 were conducted. But before using this dataset for training the models, we need to perform one last preprocessing step, that is feature scaling.

Feature scaling

Feature scaling is a preprocessing step in machine learning where the features of the dataset are scaled or normalized to a specific range. This is done to ensure that the features contribute equally to the learning process and that no particular feature dominates because of its larger magnitude. In the experiments, we used the standard scaler and the min-max scaler from *scikit-learn*. Thus, we created two different datasets: *dataset_v4_norm* and *dataset_v4_stand* obtained respectively by applying the standard scaler and the min-max scaler with interval [0, 1] on *dataset_v3*. We conducted the experiments on both datasets.

The standard scaler operates independently on each feature in the dataset. It removes the mean and scales to unit variance. The "standardized" value z of x is calculated as:

$$z = \frac{x - u}{s} \tag{3.2.1}$$

where u is the mean of the values in the dataset, and s is the standard deviation.

The min-max scaler scales each feature in the dataset individually such that all its values are transformed to a specified range, e.g., between zero and one. The transformation of a given value x is calculated as:

$$x_{norm} = \frac{x - X_{min}}{X_{max} - X_{min}} \tag{3.2.2}$$

where min and max represent the scaling range (e.g., $[0, 1]$) and X_{min} and X_{max} are respectively the smallest and the largest values in the dataset, for a specific feature.

This is the last step of the data preprocessing task. At this point, we can consider the dataset cleaned and prepared to be used for training machine learning models. As can be seen from Table 3.1, only the file size changed with respect to the previous preprocessing step. It is bigger because we have lots more numbers inside the dataset. The exploited *scikit-learn* scalers transform the dataset values into values with a precision up to 16 digits, whereas before, each feature had a maximum precision of 2.

Name	**# Features**	**# Samples**	**% NaN**	**MBs**
raw_dataset	12	4 369 673	82.07	212.26
dataset_v1	12	4 369 317	0.0	378.43
dataset_v2	10	1 485 720	0.0	116.96
dataset_v3	8	1 485 720	0.0	101.95
dataset_v4_norm	8	1 485 720	0.0	260.15
dataset_v4_stand	8	1 485 720	0.0	271.79

Table 3.1: Datasets specifications

3.2.3 Data pipeline

In this research work, data play a key part. Moreover, there are several stages of preprocessing in which it is easy to have to make minor changes. That is why we thought it was necessary to define a framework in which we could manage data efficiently. Thus, we decided to take advantage of *DVC* (Data Version Control) [24], which is an open-source version control system specifically designed for handling large files and datasets alongside traditional version control systems like Git [25]. It provides a way to manage and version control data files, guaranteeing full automation and reproducibility.

Thanks to *DVC*, we were able to define the stages shown in Figure 3.1 and described in the previous sections, each with a set of parameters associated. With just one command, that is *dvc repro*, the data pipeline is executed and *DVC* will take care of creating the datasets and tracking their versions.

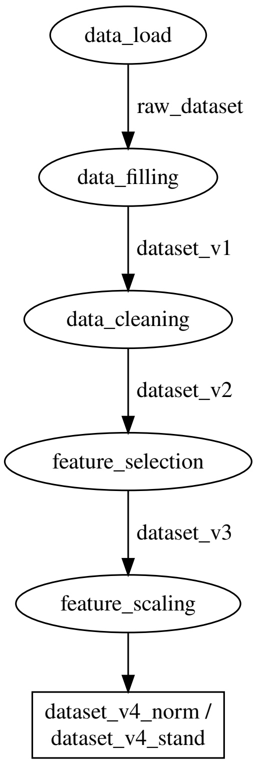


Figure 3.1: Data pipeline stages

3.2.4 Training and validation sets

Introduction

Data splitting is a technique used in machine learning to partition a dataset into multiple subsets, with the main goal of training a model and evaluating its performance. A typical simple workflow, that is the one we followed, is that a model is trained on one subset (typically the training set) and then is assessed on another subset (validation or test set) to estimate its performance.

After the data preparation step, we split the last version of the dataset it in two subsets (training and validation set) in such a way as to follow the strategy explained in Chapter 3.1. The training set contains data generated when the heat pump unit was working properly. It is used to train the machine learning model in recognizing the normal operating conditions of the heat pump unit. On the other hand, the validation set contains both safe and anomaly points. The validation set is used to evaluate the trained model by assessing how accurately it can recognize the anomalies in the data.

Gas leakage experiment

As already stated in Chapter 3.1, in order to evaluate the models' performance, it was necessary to have a well known anomaly event alongside with the associated data samples in the dataset. That is why we introduced an anomaly in the dataset by manually tampering with the heat pump unit. What we did was to reduce the gas (that is, the refrigerant) in the refrigeration circuit of the machine.

We let the machine run for some days with this lack of gas, and we collected the data generated by sensors. Basically, we simulated a gas leak in the heat pump unit. The anomaly data points of this time period were all included into the validation set.

Training and validation sets

The training datasets *dataset_v4_norm* and *dataset_v4_stand* contains a total of 1 485 720 samples. More specifically: 116 880 (7.87% of the dataset) samples correspond to the gas leakage anomaly, and the remaining 1 368 840 (92.13%) samples are all safe points. It is important to point out that, most likely, there is a small subset of points among the 1 368 840 safe points that are not safe points, but we are pretty confident that this subset is irrelevant. Hence, we are also pretty confident that the model is going to effectively learn the correct state of operation of the heat pump unit.

As already stated in Chapter 3.2.2, we filtered the dataset in order to have only samples relative to the chiller status. Thus, since the heat pump unit is used in chiller mode during the warm season, the dataset comprises values from the 23rd of May 2023 until the 2nd of November 2023. The anomaly was introduced in the machine on the 20th of June and taken out on the 26th, six days later. The "timeline" represented in Figure 3.2 shows the proportions of the dataset that are related to safe and anomaly points. The number of safe points before the introduction of the anomaly is 157 206 (10.58%) while the number of safe points after is 1 211 634 (81.55%). These statistics can also be seen in Table 3.3.

Before creating the split types, given the strategy explained in Chapter 3.1, we asked ourselves what a model would see after being sent into production. It would most likely see data it has never seen before, but ideally, the new safe points would follow the same distribution as the one on which the model was trained, while the eventual anomaly points would be expected to be considered outliers. Thus, in order to simulate this scenario and test the models, we created a training set that contains only safe points, and a validation set that contains all the anomaly points plus some safe points.

We decided to use 80% of the dataset to build the training set, and the remaining 20% to build the validation set. We then came up with two split types:

Split type 1. After putting all the anomaly points in the validation set, we filled it with samples taken in the moments just before and after the period of time relative to the anomaly, until reaching a size of 20% of the whole dataset. The training set contains all the remaining samples, that are all safe points.

Split type 2. After putting all the anomaly points in the validation set, we filled it with random safe points until reaching a size of 20% of the whole dataset. The training set contains all the remaining samples, that are all safe points.

Set	**# Samples**	**% Safe points**	**% Anomaly points**
Training set	1 186 762	100.00	0.00
Validation set	296 690	60.60	39.40

Table 3.2: Specifications of training and validation sets of split type 1

It would be ideal to insert in the validation set safe points taken in the exact same conditions (like external air temperature, target indoor temperature, etc.) as the one when the gas leakage anomaly event was introduced, because in this way we would see if the model effectively can discern between the anomaly and the safe points. Since accomplishing this is not simple, especially with just one year of data, we indeed supposed that the samples taken in the moments right before and after the anomaly event could meet this condition, that is how we created the 1st split type.

With the 2nd split type, we wanted to observe the outcome of filling the validation set with safe points taken from a very similar distribution as the one on which the model was trained. We expect to have better performance metric scores, but results which are not reliable. Populating the validation set with samples closely resembling those in the training set would be inappropriate. The model might perform well simply because it recognizes the safe points as to be familiar, rather than using its generalization ability.

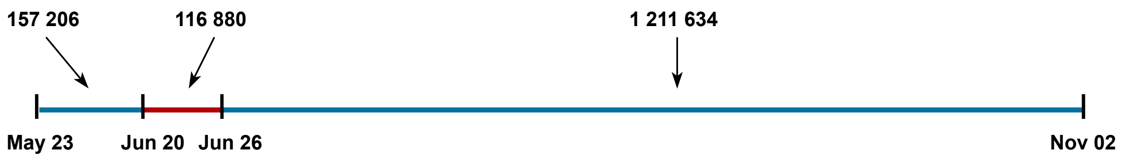


Figure 3.2: Proportions of safe points (in blue) and anomaly points (in red) in the dataset

Data type	# Samples	% Dataset
Safe points	1 368 840	92.13
Anomaly points	116 880	7.87
Safe points before anomaly	157 206	10.58
Safe points after anomaly	1 211 634	81.55

Table 3.3: Occurrences of safe and anomaly points in the dataset

3.3 Models specifications

3.3.1 PCA model parameters

We exploited the probabilistic PCA model [26] provided by *scikit-learn*. The parameters we used are listed in Table 3.4.

Parameter	**Value**
n_components	2
copy	True
whiten	False
svd_solver	auto
tol	0.0
iterated_power	auto
n_oversamples	10
power_iteration_normalizer	auto
random_state	1

Table 3.4: Parameters of the PCA model (by *scikit-learn*) used for experiments

3.3.2 Clustering model parameters

We exploited the clustering model provided by *scikit-learn* which implements the K-means clustering technique. This technique is based on the Lloyd's algorithm [27]. The parameters of the model we used are listed in Table 3.5.

Parameter	**Value**
n_clusters	k
init	k-means++
n_init	20
max_iter	300
tol	$1e-4$
random_state	1
copy_x	True
algorithm	lloyd

Table 3.5: Parameters of the K-means model (by *scikit-learn*) used for experiments

We did experiments with several values of k, that is, the number of clusters as well as the number of centroids that the K-means clustering generates. We chose to select initial cluster centroids using the *k-means++* algorithm.

In the K-means clustering, each cluster is represented by its center, called "centroid", which corresponds to the arithmetic mean of data points assigned to the cluster. The *k-means++* algorithm selects the initial cluster centroids using sampling based on an empirical probability distribution of the points' contribution to the overall inertia.

Chapter 4

Experiments and Results

4.1 PCA experiments

In this chapter, we will describe all the experiments we conducted with PCA. For each experiment, we followed the same workflow: first, we trained a PCA model on the training set, then, we used it to transform the data points of both the training and validation sets, and finally, we analyzed the plots to extrapolate useful information about the data.

What differs from the single experiments is the split type used to train and evaluate the models (see Chapter 3.2.4) and the feature scaling technique (see Chapter 3.2.2) applied to the datasets.

A positive outcome of these experiments would be to see, in the validation set 2D plot, that the samples corresponding to the gas leakage anomaly are visibly separated from the points we consider safe. If we are able to achieve this with the PCA already, it means that with more complex AI methods we could easily obtain promising results. Besides, a positive outcome would also be to discover any useful information about the datasets.

Three figures will be shown in each experiment: Figure (a) and Figure (b) shows respectively the samples of the training and the validation set transformed in 2D with the trained PCA model, and Figure (c) highlights the safe and anomaly points on the same plot of (b).

4.1.1 Split type 1 and standardization

With the split type 1 and using standardization as the feature scaling technique, we obtain the results in Figure 4.1.

Figure 4.1a. Some points deviate far from the densest area on the left side, causing the densest area of data points to be squeezed. This behavior may be because we are using standardization, which does not bring all features to the same scale. It is clear that these few data points are related to an unusual state of the machine: they most likely represent sensor measurements errors that we were not able to filter out in the data cleaning stage.

Figure 4.1b. Similar points to the unusual ones noted in (a) are not present in this plot; most likely is because the validation set comprises samples from different time intervals than those in the training set. This time, we can see in more detail the densest area of the points, which looks similar to the one that was squeezed to the left in (a). It is peculiarly composed by two well-separated "clouds". At first, one can imagine that one of the two clouds corresponds to the gas leakage anomaly, but this is expected to be false since they seem to be present also in (a).

Figure 4.1c. The anomaly points (in red) do not deviate much from the safe zone (in green). There are in fact many overlaps between the anomaly and safe points. The anomaly points create a denser zone above the safe points, but they do not form a distinct group.

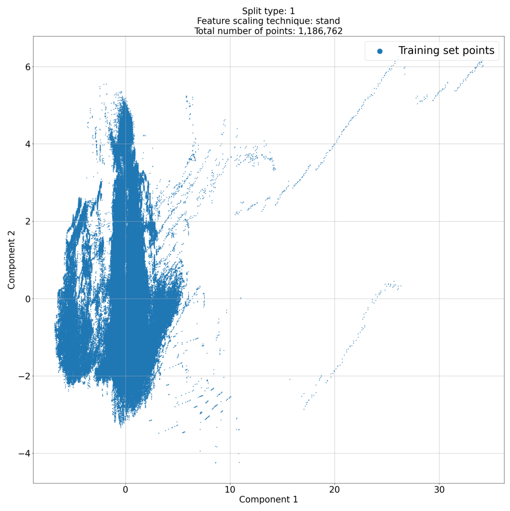
4.1.2 Split type 2 and standardization

With the split type 2 and using standardization as the feature scaling technique, we obtain the results in Figure 4.2.

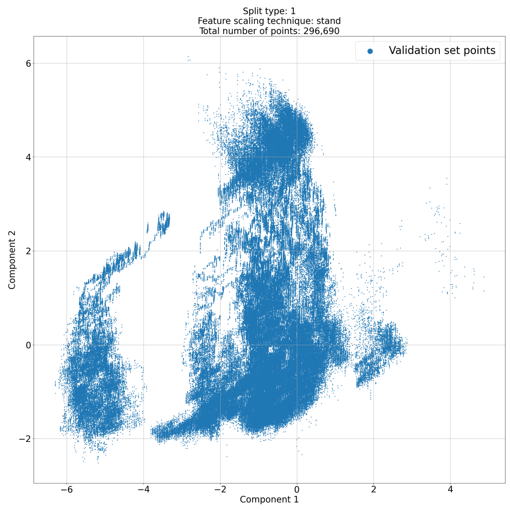
Figure 4.2b. This plot is almost identical to 4.1a, as the training sets of the two split types share nearly identical distributions. Thus, the same reasoning can be applied here.

Figure 4.2b. The unusual points noted in Figure 4.1a are also present in this PCA plot. This is because the validation set is composed by random safe points.

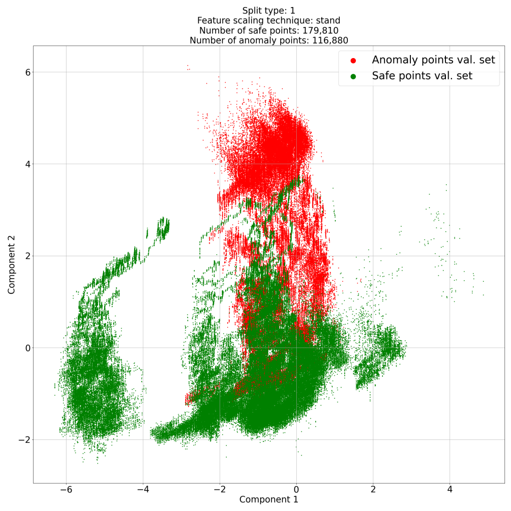
Figure 4.2c. Given the presence of these outliers, the cloud of anomaly points is significantly even more hidden. We can say that the outliers we did not manage to filter out, significantly disturb the model. At this point, we could assume that for our case study, standardization is not a suitable feature scaling technique.



(a) Training set PCA plot



(b) Validation set PCA plot



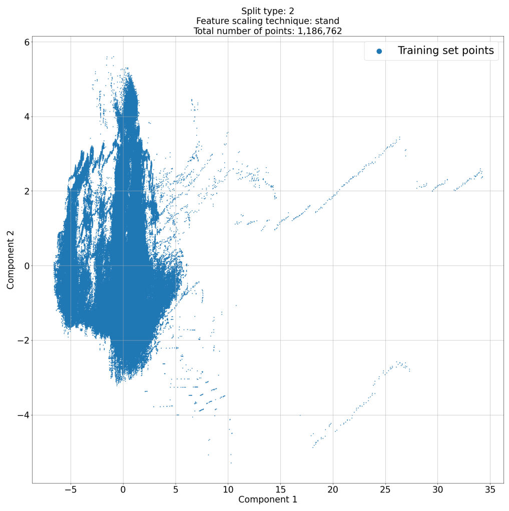
(c) Safe and anomaly points of the validation set PCA plot

Figure 4.1: PCA experiment with split type 1 and standardization

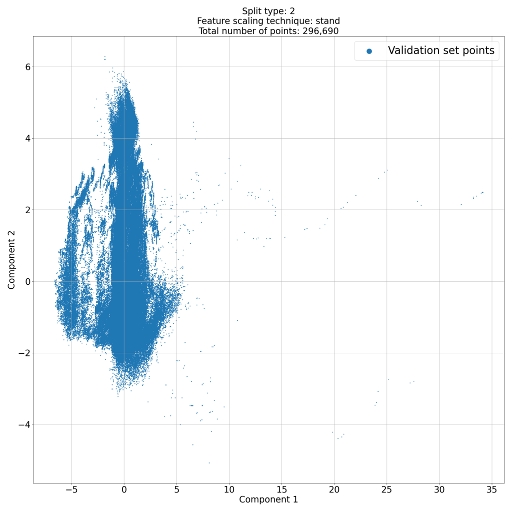
4.1.3 Split type 1 and normalization

With the split type 1 and using normalization as the feature scaling technique, we obtain the results in Figure 4.3.

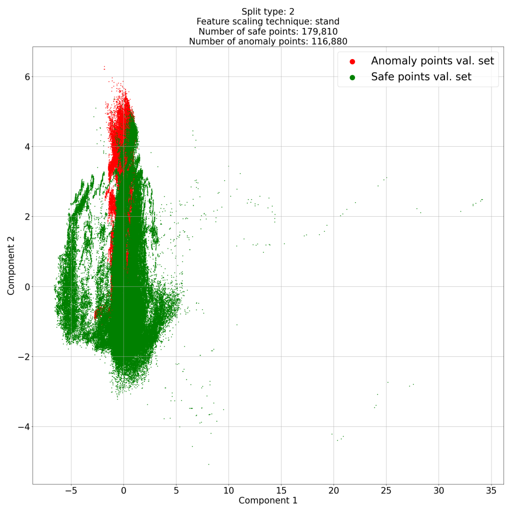
Figure 4.3a. After scaling the features with the min-max scaler, we no longer see the outliers that are present in Figure 4.1a and in Figures 4.2. They are indeed scaled between 0 and 1, and apparently now they get mixed up among the safe points. At first glance, this may be a negative effect, because we lose outliers that should be identified by the model. But since there are few of these outliers, we prefer to mix them with the safe points to minimize their impact on the PCA, rather than having them disturb the PCA as much as



(a) Training set PCA plot



(b) Validation set PCA plot



(c) Safe and anomaly points in the validation set PCA plot

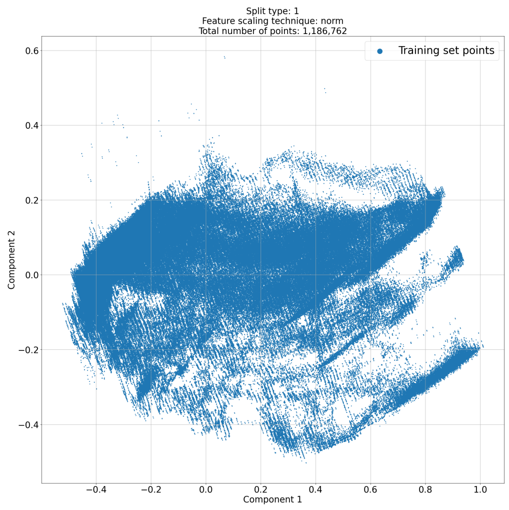
Figure 4.2: PCA experiment with split type 2 and standardization

they do with standardization.

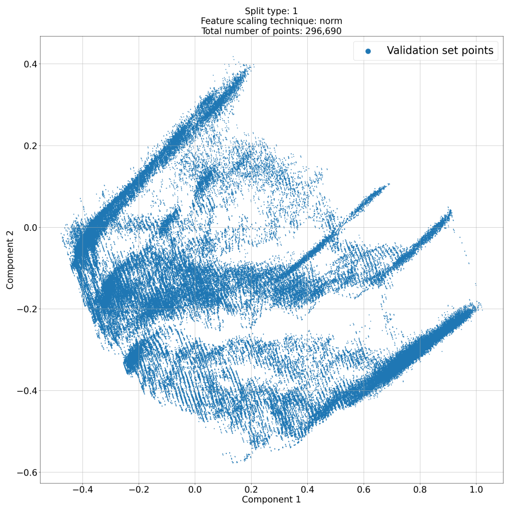
Figure 4.3b. The PCA plot of the validation set (b) is significantly different from (a). This is because with the split type 1, the validation set is composed by samples taken from a different distribution than the one of the training set. An interesting pattern resembling a “spike” can be seen in four different areas of this plot. It would be interesting to investigate (but maybe it is not worth it) what changes in the working state of the machine cause having this pattern to shift in the 2D plot.

Figure 4.3c. The anomaly points are grouped in a distinct area in the plot. They are significantly far away from the safe points. It is interesting to note that

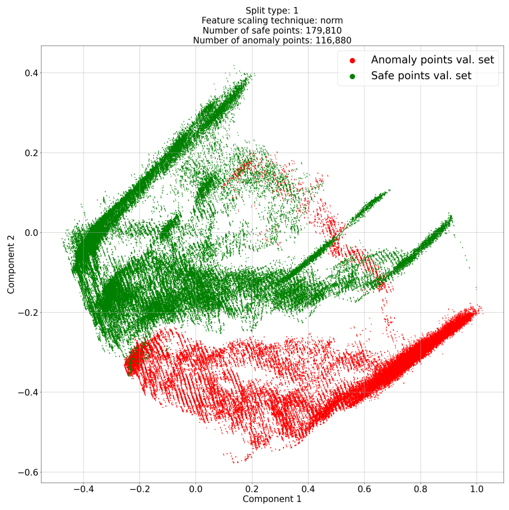
one of the four spikes is completely composed by anomaly points. This result is very promising, also because we got it with split type 1, which we consider to be the best for testing the models.



(a) Training set PCA plot



(b) Validation set PCA plot



(c) Safe and anomaly points in the validation set PCA plot

Figure 4.3: PCA experiment with split type 1 and normalization

4.1.4 Split type 2 and normalization

With the split type 2 and using normalization as the feature scaling technique, we obtain the results in Figure 4.4.

Figure 4.4a. This plot is almost identical to 4.3a, as the training sets of the two split types share nearly identical distributions. Thus, the same reasoning can be applied here.

Figure 4.4b. The PCA plot of the validation set (b) is now similar to (a). This is because the safe points of the validation set were taken randomly, and so they share the same distribution of the training set data points.

Figure 4.4c. In the cloud of anomaly points, there are also some safe points. This means that in some cases, during the correct state of operation of the heat pump unit, the machine behaves as there is some gas missing in the circuit. It could also be that what we considered safe points might actually be anomalies. To be certain, one would have to investigate those instances to determine, for example, if there was indeed a lack of gas in the circuit. It could also be the opposite, in the sense that not all the gas leakage anomaly points are not actually anomalies.

4.2 Clustering Experiments

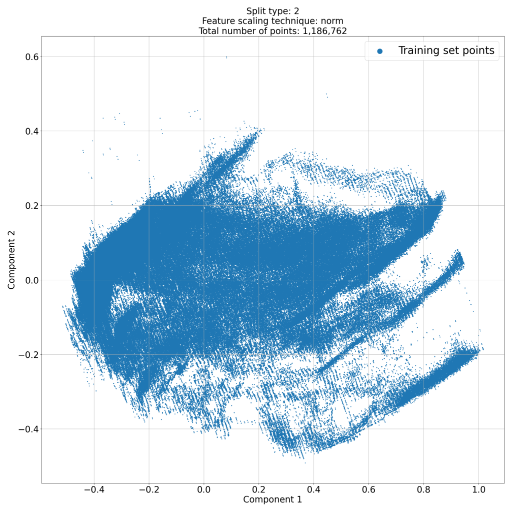
In this chapter, we will report all the experiments conducted using the clustering technique. For each experiment, we followed the same workflow: first, we trained a model on the training set. Then, we ran the Outlier Detection Algorithm, that will be presented in Chapter 4.2.1, which is designed to binary classify the samples in the validation set, and in the end, we computed the performance metric scores (see Chapter 2.3.3).

What differs from the single experiments is the split type we used to train and evaluate the models and the feature scaling technique applied to the datasets.

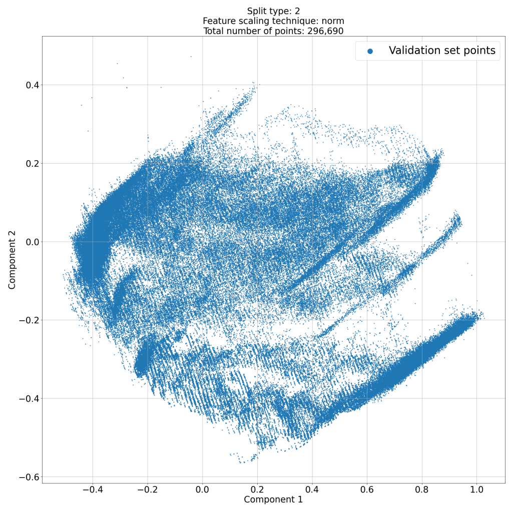
A positive outcome of these experiments would be to achieve high performance metric scores. This would indicate that the Outlier Detection Algorithm is highly effective in precisely classifying safe points and points associated to the gas leakage anomaly.

In each experiment, we will show and analyze the performance metric scores obtained by running the Outlier Detection Algorithm on the validation set. For each experiment, the scores will be reported within a table, together with the algorithm parameters and the number of clusters used.

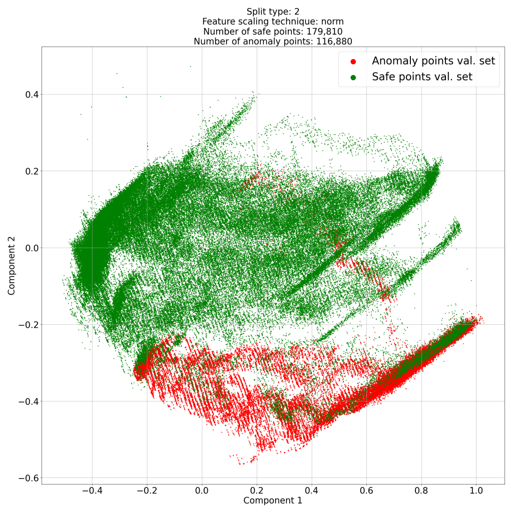
Only the top 10 best scores are shown, because we tried many combinations of parameters. We consider a score better than another if it has a higher F_1 *score*. The chose of using the F_1 *score* as the baseline metric is because the validation set is unbalanced (39.4% of anomaly points and 60.6% of safe points, see Table 3.2), hence, as already explained in Chapter 2.3.3, we need to take into account both



(a) Training set PCA plot



(b) Validation set PCA plot



(c) Safe and anomaly points in the validation set PCA plot

Figure 4.4: PCA experiment with split type 2 and normalization

precision and recall. The accuracy is not a reliable metric in our case, but still we decided to show it.

4.2.1 The Outlier Detection Algorithm

Underlying idea

The clustering performed on the training set generates k centroids. Given the fact that the training set is entirely composed by safe points, the computed centroids correspond to safe clusters. This means that every point close enough to a centroid, it is a safe point, otherwise it is an anomaly point. The latter statement is the key idea on which the Outlier Detection Algorithm is based to perform the classification task.

Our objective is to use the generated centroids to determine which points in the validation set are close enough to be classified as safe, and which are not. If a point is too far from every centroid, it cannot be considered included in any safe cluster and so, it is classified as an anomaly point.

In order to accomplish this, we created Algorithm 1 and also a simpler version of it, which is Algorithm 2. The general idea that both algorithms follow is to take the centroids computed in the training phase and compute, for each point in the validation set, its closest distance to a centroid. If this distance is too high, the point is classified as an anomaly point, otherwise as a safe point.

Since the datasets have the data sorted in chronological order, with Algorithm 1 we decided to take into account the time: if a point cannot be considered safe, but neither is too far from every centroid, we consider it a "warning" point. If the algorithm continues to detect warning points, after some time, it considers them anomaly points, because the heat pump unit is outside its normal state of operation for too long. This approach is used only with the 1^{st} split type, because the 2^{nd} split type takes random safe points from the dataset to build the validation set, hence the dataset loses its property of being composed by time-series data. The simple version of the algorithm is used with the 2^{nd} split type.

Outlier Detection Algorithm description

Input - V is the validation set, C is the set of centroids computed in the training phase, *min_dist* is the distance (represented as a float number) under which a point is classified as a safe point, *warn_time* is the time (expressed in seconds) after which the warning state is turned into an anomaly state, and *anomaly_dist* is the distance after which a point is classified as an anomaly. *anomaly_dist* must be greater than *min_dist*.

Output - *safe_samples*, *warning_samples*, *anomaly_samples* are three lists that contain respectively the safe points, the warning points (if any), and the anomaly points identified in the validation set. The list *warning_samples* is not empty if the algorithm is reading warning points and returns without being able to classify them as safe or anomaly points.

Rows 2:4 - before the main for loop, the algorithm initializes some variables: the three output lists, two flag variables (*w_trigger* and *a_trigger*) set to False

Algorithm 1 The Outlier Detection Algorithm

```
function OutlierDetection($V$, $C$, $min\_dist$, $warn\_time$, $anomaly\_dist$)
    $safe\_samples$, $warning\_samples$, $anomaly\_samples$ ← empty list
    $w\_trigger$, $a\_trigger$ ← False
    $start\_time$ ← None
    for $x_i$ in $V$ do
        $closest\_dist_{x_i}$ ← ComputeClosestDistance($x_i, C$)
        if $closest\_dist_{x_i} \geq anomaly\_dist$ then
            add $x_i$ to $anomaly\_samples$
            $a\_trigger$ ← True
            move $warning\_samples$ to $anomaly\_samples$
        else if $closest\_dist_{x_i} < min\_dist$ then
            add $x_i$ to $safe\_samples$
            $a\_trigger$ ← False
            move $warning\_samples$ to $safe\_samples$
            $w\_trigger$ ← False
        else if $min\_dist \leq closest\_dist_{x_i} < anomaly\_dist$ then
            if $a\_trigger$ then
                add $x_i$ to $anomaly\_samples$
            else if $w\_trigger$ is False then
                add $x_i$ to $warning\_samples$
                $start\_time$ ← GetCurrentTime()
                $w\_trigger$ ← True
            else
                add $x_i$ to $warning\_samples$
                if GetCurrentTime() $-$ $start\_time$ $>$ $warning\_time$ then
                    $a\_trigger$ ← True
                    move $warning\_samples$ to $anomaly\_samples$
    return $safe\_samples$, $warning\_samples$, $anomaly\_samples$
```

and used for triggering respectively the warning and the anomaly state, and *start_time* set to None used for saving the timestamp of the first reading of a warning point in a sequence of warning points.

Rows 5:6 - the for loop iterates through the validation set. Given the sample x_i, the algorithm computes its the closest distance to a centroid, given the set of centroids C. The distance used is the Euclidean distance.

Rows 7:10 - if the closest distance ($closest\_dist_{x_i}$) is greater than the anomaly distance (*anomaly_dist*), the point is inserted into the anomaly point list (*anomaly_samples*). Here, after seeing an anomaly point, the algorithm set the flag *a_trigger* to True because we are in an anomaly situation: if the next point is a warning point, it is classified as an anomaly point. If we were in a warning situation, so the list *warning_samples* contains some points, all those points are then moved in the list *anomaly_samples*.

Rows 11:15 - if $closest\_dist_{x_i}$ is smaller than *min_dist*, the point is safe, so it is inserted in the list $safe\_samples$. Here, we set the flags *w_trigger* and *a_trigger* to False because we are no more in any warning or anomaly situation. If we were in a warning situation, so the list *warning_samples* contains some points, all those points are then moved in the list $safe\_samples$.

Rows 16:27 - if $closest\_dist_{x_i}$ is not smaller than *min_dist* but not bigger than *anomaly_dist*, the point is a warning point. Three cases can happen here: 1) if we are in an anomaly situation, because either we encountered an anomaly point before and the streak of anomaly/warning points is continuing, either we are reading warning points for too long, x_i is classified as an anomaly point so it is inserted in *anomaly_samples*. 2) If the previous point was a safe point, and now we are reading a warning point, we insert this point into *warning_samples*, we trigger the warning situation, and we set the variable *start_time* to the current time. 3) If the previous value was not a safe point, and we are not in an anomaly situation, we insert x_i into *warning_samples* and we evaluate if the starting time of this warning points streak is long enough to consider all the warning points of this streak an anomaly event; if this is the case, then we move all the warning points in the list *anomaly_samples*, otherwise we insert x_i into *warning_samples*.

Algorithm 2 The Outlier Detection Algorithm - Simple version

```
1: function SimpleOutlierDetection($V$, $C$, $safe\_dist$)
2:     $safe\_samples$, $anomaly\_samples \leftarrow$ empty list
3:     for $x_i$ in $V$ do
4:         $closest\_dist_{x_i} \leftarrow$ ComputeClosestDistance($x_i, C$)
5:         if $closest\_dist_{x_i} \geq safe\_dist$ then
6:             add $x_i$ to $anomaly\_samples$
7:         else
8:             add $x_i$ to $safe\_samples$
9:     return $safe\_samples$, $anomaly\_samples$
```

Outlier Detection Algorithm (simple version) description

Input - V is the validation set, C is the set of centroids computed in the training phase, $safe\_dist$ is the distance (represented as a float number) under which a point is classified as a safe point and above which it is classified as an anomaly point.

Output - $safe\_samples$, $anomaly\_samples$ are two lists that contain respectively the safe and the anomaly points identified in the validation set.

Row 2 - before the main for loop, the algorithm initializes only the two output lists.

Rows 3:4 - the for loop iterates through the validation set. Given the point x_i, the algorithm computes its the closest distance to a centroid, given the set of centroids C.

Rows 5:6 - if $closest\_dist_{x_i}$ is greater than $safe\_dist$, the point is inserted into $anomaly\_samples$.

Rows 7:8 - if $closest\_dist_{x_i}$ is smaller than $safe\_dist$, the point is safe, so it is inserted in the list $safe\_samples$.

4.2.2 Split type 1 and normalization

In this experiment, we run the Outlier Detection Algorithm with all the combinations of the following parameter values: $k = \{3, 4, 10, 50\}$, $min\_dist = \{0.01, 0.05, 0.1, 0.2, 0.3, 0.4\}$, $warn\_time = \{10, 60, 300, 900, 3600\}$, and $anomaly\_dist = \{0.1, 0.2, 0.3, 0.4, 0.5\}$. The combinations where $anomaly\_dist$ is greater than min_dist are discarded. The parameter values were selected based on initial test experiments and following a heuristic approach. For this reason, the scores obtained are not the highest possible the Outlier Detection Algorithm can obtain. With a proper parameters tuning process, it is most likely possible to achieve better scores. The results of this experiment are reported in Table 4.1.

The F_1 *score* is surprisingly high. A score of 0.970 means that the algorithm can reliably distinguish a safe point from a point corresponding to the gas leakage anomaly. A recall of 95.93% means that the algorithm can correctly identify almost all the 116 880 anomaly points present in the validation set. A precision of 98.09% means that the algorithm is almost never confused with identifying a safe point as an anomaly. We decided to use these results as the baseline on which to compare the results of the other experiments.

Parameters				**Scores**			
k	**min_dist**	**warn_time**	**anomaly_dist**	**Accuracy**	**Precision**	**Recall**	**F_1 score**
4	0.3	3600	0.5	97.66	98.09	95.93	0.970
4	0.3	3600	0.4	97.59	97.90	95.93	0.969
4	0.3	900	0.5	96.57	95.40	95.93	0.956
4	0.3	900	0.4	96.55	95.34	95.93	0.956
4	0.3	300	0.5	96.01	93.74	96.32	0.950
4	0.3	300	0.4	96.00	93.71	96.32	0.950
3	0.3	3600	0.5	95.81	92.97	96.66	0.947
3	0.3	3600	0.4	95.77	92.89	96.67	0.947
4	0.3	60	0.4	95.71	92.86	96.53	0.946
4	0.3	60	0.5	95.71	92.86	96.53	0.946

Table 4.1: Outlier Detection Algorithm results with split type 1 and normalization

4.2.3 Split type 1 and standardization

In this experiment, we run the Outlier Detection Algorithm with all the combinations of the following parameter values: $k = \{3, 4, 10, 50\}$, $min\_dist = \{0.01, 0.05, 1, 2, 4\}$, $warn\_time = \{10, 60, 300, 900, 3600\}$, and $anomaly\_dist = \{5, 10, 15, 20, 40\}$.

The F_1 *score* is not that high in this case, at least comparing it with the 0.97 obtained with the previous experiment. A score of 0.717 means that the algorithm cannot reliably distinguish a safe point from an anomaly point. We could increase this score by improving the parameter values combinations, but since we followed the same heuristic, we are confident that Algorithm 1 cannot reach the performance metric scores as the one obtained in the baseline experiment.

With these results, we can strengthen the hypothesis that standardization is not a suitable feature scaling technique for our case study, as PCA was already suggesting us with Experiment 4.1.1 and Experiment 4.1.2.

Parameters				**Scores**			
k	**min_dist**	**warn_time**	**anomaly_dist**	**Accuracy**	**Precision**	**Recall**	**F_1 score**
4	2	10	5	76.89	69.28	74.29	0.717
4	2	10	10	76.89	69.28	74.29	0.717
4	2	10	15	76.89	69.28	74.29	0.717
4	2	10	20	76.89	69.28	74.29	0.717
4	2	10	40	76.89	69.28	74.29	0.717
4	2	60	5	76.72	70.39	70.60	0.705
4	2	60	10	76.68	70.36	70.49	0.704
4	2	60	15	76.68	70.36	70.49	0.704
4	2	60	20	76.68	70.36	70.49	0.704
4	2	60	40	76.68	70.36	70.49	0.704

Table 4.2: Outlier Detection Algorithm results with split type 1 and standardization

4.2.4 Split type 2 and normalization

In this experiment, we run the simple version of the Outlier Detection Algorithm with all the combinations of the following parameter values: $k = \{3, 4, 5, 7, 10, 20, 50, 100\}$ and $safe\_dist = \{0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4\}$.

We obtained a high F_1 *score* by running the simple version of the Outlier Detection Algorithm. A score of 0.928 means that the algorithm can reliably distinguish a safe point from an anomaly point. A recall of 96.62%, that is higher than the 95.93% of the baseline, means that the algorithm performs even better in identifying the 116 880 anomaly points present in the validation set. But with a lower precision of 89.28%, compared to the 98.09% of the baseline, we can see that the algorithm, in order to try to catch all the anomaly samples, misclassify more safe points. This loss in precision is most likely due to the fact that Algorithm 2, since does not take into account the time, struggles in identifying complex anomaly and safe patterns.

Parameters		**Scores**			
k	**safe_dist**	**Accuracy**	**Precision**	**Recall**	F_1 **score**
4	0.3	94.10	89.28	96.62	0.928
3	0.3	92.95	86.71	96.97	0.915
4	0.2	84.77	72.55	98.67	0.836
5	0.3	87.21	90.76	75.19	0.822
5	0.2	82.94	74.85	85.37	0.797
3	0.2	76.58	63.05	97.97	0.767
3	0.4	81.36	92.48	57.35	0.708
20	0.1	71.89	60.40	83.21	0.699
10	0.1	61.23	50.45	88.93	0.643
100	0.05	63.32	52.14	83.82	0.642

Table 4.3: Outlier Detection Algorithm (simple version) results with split type 2 and normalization

4.2.5 Split type 2 and standardization

In this experiment, we run the simple version of the Outlier Detection Algorithm with all the combinations of the following parameter values: $k = \{3, 4, 5, 7, 10, 20, 50, 100\}$ and $safe\_dist = \{0.05, 0.01, 0.05, 1, 2, 5\}$.

We obtain an F_1 *score* of 0.741, which is slightly lower than the one obtained running Algorithm 1 with split 1 (using standardization), that is 0.717. This means that the original version of the algorithm does not perform better than the trivial approach in the case of standardization.

Parameters		**Scores**			
k	**safe_dist**	**Accuracy**	**Precision**	**Recall**	**F_1 score**
4	2	80.02	75.63	72.73	0.741
3	2	77.53	69.35	76.97	0.729
5	2	77.68	75.80	63.65	0.692
20	1	71.17	59.92	81.00	0.688
10	1	64.69	53.04	90.39	0.668
5	1	57.16	47.90	99.95	0.647
4	1	54.33	46.28	99.26	0.631
7	1	57.30	47.78	90.18	0.624
50	1	73.04	69.8	55.66	0.619

Table 4.4: Outlier Detection Algorithm (simple version) results with split type 2 and standardization

Chapter 5

Conclusions

5.1 Dataset and models

In this research, we adopted a data-centric approach, with the dataset preparation step being a crucial component. We developed a versatile data pipeline that can be easily adapted and reused for future developments. By following the steps outlined in Chapter 3.2, similar datasets can be constructed, allowing others to work under the same conditions as we did.

The data preprocessing step is the main component of the pipeline. It is composed by several stages that can be adapted based on the typology of the heat pump unit and the strategy for anomaly detection. For instance, the feature selection stage highly depends on the machine under consideration. On the other hand, the structure of the training and validation sets is highly based on the strategy used to train the machine learning models.

Since this is a feasibility study, the machine learning techniques used for the experiments (PCA and clustering) were chosen for their easy-to-use. According to [3], several well known techniques are used when dealing with anomaly detection in mechanical units, which comprises neural networks and also statistical approaches.

In future phases of this work, it would be essential to do a research work to define which anomaly detection technique suits the best for the domain of heat pump units. Regarding the dataset, the one used in this work contains only one year of data from a single heat pump unit; in future developments, it will be essential to expand it with samples collected over several years and from multiple machines.

5.2 Experiments summary and feasibility considerations

PCA shows that the heat pump unit under study exhibits complex behavior, indeed we were not able to find any well-defined pattern after scaling down the dataset to 2 dimensions. However, the analysis was useful for gaining a better understanding of the data and refining the data preparation step. For example, it helped extrapolate clues about the best type of feature scaling technique to use, and highlighted the challenges involved in the data cleaning stage.

Clustering, in particular the Outlier Detection Algorithm, shows that with a proper anomaly detection technique it is possible to obtain promising results. The algorithm is able to classify with high precision the safe and anomaly points given the strategy presented in Chapter 3.1 and the training and validation sets defined in Chapter 3.2.4. This is most likely because clustering does not perform any type of approximation to the dataset, indeed it operates in the original dimensional space. In all the experiments, the best scores are obtained with a low number of clusters; this may indicate that the context under study is not that complex, and highlights the feasibility of creating a reliable intelligent anomaly detection system.

Although an F_1 *score* of 0.97 obtained with Experiment 4.2.2 seems to be an outstanding result, it is important to consider that the Outlier Detection Algorithm has been tested on just one possible anomaly event (manually introduced with the gas leakage experiment). Moreover, the models have been trained to recognize a tiny portion of the normal operating conditions of the heat pump unit, since the training set is composed by samples taken only from the 23^{rd} of May 2023 until the 2^{nd} of November 2023.

Despite this confined environment under which the research was carried out, we can still state with high confidence that it is feasible to use anomaly detection in heat pump units. With a more consistent dataset, strategy and anomaly detection technique, it is likely possible to engineer a reliable system capable of detecting anomaly events before they actually occur, hence building a solution that can be deployed into production.

Acknowledgments

The writing of this thesis represents the final chapter of my journey as a Master's student. Achieving this goal fills me with pride and prepares me for the future. The moments I experienced during this Master's degree will always hold a special place in my memories as times of profound personal growth. I would like to express my deepest gratitude to certain special individuals with whom I shared these moments, and who helped shape them into what they are.

First and foremost, I would like to extend my gratitude to the University of Padua, as well as all the professors I met during my academic journey. Special thanks go to my supervisor, Professor Fabio Vandin, who provided me with the opportunity to work on this project and supported me throughout the evolution of this research.

Many thanks to Swegon Operations Srl. for hosting the development of this project. My most sincere appreciation goes to my company tutor, Francesco Artusi, and also to Mose' Prandin. Their exceptional support and mentoring were invaluable for my personal and professional growth during my time there.

My warmest thanks go to my family, without whose support it would surely have been more difficult to achieve this goal. In particular, I thank my parents, Rosa and Antonio, for giving me a continuous push toward the goal, my brother Mario, for always being a precious guide, and Sara, for being not only a family member but also a true and valuable friend.

Last but not least, I want to express my deepest gratitude to all my friends and loved ones who enrich my life with fundamental emotions and meaning. Special thanks to Mauro, Giacomo, Andrea, Pietro, Lorenzo, Diego, and Cristiano to be inestimable friends. I am also grateful to my dear university former colleagues Luca, Fabio, and Marco with whom I shared my student life. Lastly, but certainly not for importance, I extend my warmest thanks to Anna for her love and tireless support.

References

[1] Sérgio F. Chevtchenko et al. “Anomaly Detection in Industrial Machinery Using IoT Devices and Machine Learning: A Systematic Mapping”. In: *IEEE Access* 11 (2023), pp. 128288–128305. DOI: 10.1109/ACCESS.2023.3333242.

[2] Ali Bou Nassif et al. “Machine Learning for Anomaly Detection: A Systematic Review”. In: *IEEE Access* 9 (2021), pp. 78658–78700. DOI: 10.1109/ACCESS.2021.3083060.

[3] Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly detection: A survey”. In: *ACM Comput. Surv.* 41.3 (2009). ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: https://doi.org/10.1145/1541880.1541882.

[4] Ibrahim Dincer and Dogan Erdemir. “Chapter 3 - Energy Management in Buildings”. In: *Heat Storage Systems for Buildings*. Ed. by Ibrahim Dincer and Dogan Erdemir. Elsevier, 2021, pp. 91–113. ISBN: 978-0-12-823572-0. DOI: https://doi.org/10.1016/B978-0-12-823572-0.00004-7. URL: https://www.sciencedirect.com/science/article/pii/B9780128235720000047.

[5] K.J. Chua, S.K. Chou, and W.M. Yang. “Advances in heat pump systems: A review”. In: *Applied Energy* 87.12 (2010), pp. 3611–3624. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2010.06.014. URL: https://www.sciencedirect.com/science/article/pii/S030626191000228X.

[6] D. M. Hawkins. “Introduction”. In: *Identification of Outliers*. Dordrecht: Springer Netherlands, 1980, pp. 1–12. DOI: 10.1007/978-94-015-3994-4_1. URL: https://doi.org/10.1007/978-94-015-3994-4_1.

[7] Anitha Ramchandran and Arun Kumar Sangaiah. “Chapter 11 - Unsupervised Anomaly Detection for High Dimensional Data—an Exploratory Analysis”. In: *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*. Ed. by Arun Kumar Sangaiah, Michael Sheng, and Zhiyong Zhang. Intelligent Data-Centric Systems. Academic Press, 2018,

pp. 233–251. ISBN: 978-0-12-813314-9. DOI: https://doi.org/10.1016/B978-0-12-813314-9.00011-6. URL: https://www.sciencedirect.com/science/article/pii/B9780128133149000116.

[8] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2002. ISBN: 9780387954424. URL: https://books.google.it/books?id=_olByCrhjwIC.

[9] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. ISBN: 9781107057135. URL: https://books.google.it/books?id=ttJkAwAAQBAJ.

[10] Lars Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[11] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall advanced reference series. Prentice Hall, 1988. ISBN: 9780130222787. URL: https://books.google.it/books?id=7eBQAAAAMAAJ.

[12] Karlton Sequeira and Mohammed Zaki. "ADMIT: Anomaly-based data mining for intrusions". In: July 2002, pp. 386–395. DOI: 10.1145/775047.775103.

[13] Richard J. Bolton and David J. Hand. "Unsupervised Profiling Methods for Fraud Detection". In: 2002. URL: https://api.semanticscholar.org/CorpusID:14365948.

[14] Daniela Balslev et al. "Cluster analysis of activity-time series in motor learning". In: *Human brain mapping* 15 (Apr. 2002), pp. 135–45. DOI: 10.1002/hbm.10015.

[15] A.N. Srivastava and B. Zane-Ulman. "Discovering recurring anomalies in text reports regarding complex space systems". In: *2005 IEEE Aerospace Conference*. 2005, pp. 3853–3862. DOI: 10.1109/AERO.2005.1559692.

[16] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[17] P. Zotto, S.L. Russo, and P. Sartori. *Fisica Generale. Meccanica e Termodinamica*. Società Editrice Esculapio, 2022. ISBN: 8893853299. URL: https://books.google.it/books?id=hp6cEAAAQBAJ.

[18] Craig B. Smith and Kelly E. Parmenter. “Chapter 8 - Management of Heating and Cooling”. In: *Energy Management Principles (Second Edition)*. Ed. by Craig B. Smith and Kelly E. Parmenter. Second Edition. Oxford: Elsevier, 2016, pp. 125–187. ISBN: 978-0-12-802506-2. DOI: https://doi.org/10.1016/B978-0-12-802506-2.00008-2. URL: https://www.sciencedirect.com/science/article/pii/B9780128025062000082.

[19] Bukola Bolaji, M. Akintunde, and T. Falade. “Comparative Analysis of Performance of Three Ozone-Friendly HFC Refrigerants in a Vapour Compression Refrigerator”. In: *Journal of Sustainable Energy and Environment* 2 (Jan. 2011), pp. 61–64.

[20] Ryohei Fujimaki, Takehisa Yairi, and Kazuo Machida. “An approach to spacecraft anomaly detection problem using kernel feature space”. In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. KDD ’05. Chicago, Illinois, USA: Association for Computing Machinery, 2005, pp. 401–410. ISBN: 159593135X. DOI: 10.1145/1081870.1081917. URL: https://doi.org/10.1145/1081870.1081917.

[21] David L. Banks and Stephen E. Fienberg. “Data Mining, Statistics”. In: *Encyclopedia of Physical Science and Technology (Third Edition)*. Ed. by Robert A. Meyers. Third Edition. New York: Academic Press, 2003, pp. 247–261. ISBN: 978-0-12-227410-7. DOI: https://doi.org/10.1016/B0-12-227410-5/00164-2. URL: https://www.sciencedirect.com/science/article/pii/B0122274105001642.

[22] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[23] Razieh Sheikhpour et al. “A Survey on semi-supervised feature selection methods”. In: *Pattern Recognition* 64 (2017), pp. 141–158. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2016.11.003. URL: https://www.sciencedirect.com/science/article/pii/S0031320316303545.

[24] Ruslan Kuprieiev et al. *DVC: Data Version Control - Git for Data & Models*. Version 3.51.2. June 2024. DOI: 10.5281/zenodo.11445998. URL: https://doi.org/10.5281/zenodo.11445998.

[25] Scott Chacon and Ben Straub. *Pro git*. Apress, 2014.

[26] Michael E. Tipping and Christopher M. Bishop. “Mixtures of Probabilistic Principal Component Analyzers”. In: *Neural Computation* 11.2 (Feb. 1999), pp. 443–482. ISSN: 0899-7667. DOI: 10.1162/089976699300016728. eprint: https://direct.mit.edu/neco/article-pdf/11/2/443/814064/089976699300016728.pdf. URL: https://doi.org/10.1162/089976699300016728.

[27] Stuart Lloyd. “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.