Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Marianne Pettersen

Machine Learning based Acoustic Anomaly Detection of Gas Leakages in Industrial Environments

A comparison of Clustering, Gradiant Boosting and Neural Networks

Master's thesis in Engineering and ICT (Robotics and Automation)
Supervisor: Gunleiv Skofteland
Co-supervisor: Christian Holden
June 2022

NTNU
Norwegian University of Science and Technology

Marianne Pettersen

Machine Learning based Acoustic Anomaly Detection of Gas Leakages in Industrial Environments

A comparison of Clustering, Gradiant Boosting and Neural Networks

Master's thesis in Engineering and ICT (Robotics and Automation)
Supervisor: Gunleiv Skofteland
Co-supervisor: Christian Holden
June 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Preface

This master thesis is written as part of a five-year M.Sc. program named Engineering and ICT at the Norwegian University of Science and Technology. The thesis is conducted within the specialization in *Robotics and Automation*, at the department of Mechanical and Industrial Engineering (MTP) during 20 weeks in the spring semester of 2022.

The thesis is a prolongation of the work done in a specialization project from the autumn semester of 2021. The project, named *Research review on Acoustic Anomaly Detection Techniques for Leakage Detection on Industrial Plants*, involved a comprehensive literature review on the state-of-the-art techniques for discovering gas leakages in noisy industrial environments. Some experimentation using a basic digital processing technique called the Root-Mean-Square (RMS) method was conducted on a dataset of sound recordings, and the results indicated the need for more modern and complex techniques such as the use of machine learning. Therefore, the use of machine learning as an alternative technique to solve the problem of detecting gas leakages is the main focus of this master thesis.

The project was initialized by the largest offshore operator of oil and gas in Norway, Equinor ASA. All work during both the project and the subsequent master was conducted in collaboration with representatives from Equinor ASA. I would like to use this opportunity to express my gratitude towards the involved representatives from Equinor for their contribution to an increased understanding of the comprehensive topics related to this project. They have also provided necessary and helpful insight into how the oil industry and today's gas detection methods on oil platforms work on a daily basis, including why there is a need for improvement.

Professor Gunleiv Skofteland and professor Christian Holden has supervised the work in both the project thesis and the master thesis. I would like to thank the professors for their help and engagement during this past year. Their feedback, advice and discussions have been truly valuable and appreciated. The professors' and representatives' prior knowledge and guidance on particularly the topic of digital signal processing has been especially valued as there have been no courses addressing this topic previously during the five-year M.Sc. program. Thus, their explanations and abbreviations have been critical in helping to tighten my knowledge gaps related to acoustic signals and how to process them in order to work on the gas detection problem with more familiar programming techniques.

Abstract

The objective of this master thesis was to provide a proof of concept for an acoustic gas leakage detector based on machine learning, and to develop the software part of such a detection system. Gas leakages can be frustrating for industrial plant owners as they lead to energy consumption, higher costs, increased safety risks for workers, and cause pollution. The gas detection systems of today are based on gas monitors. However, gas monitors do not perform well on detecting small leakages and especially have trouble on outdoor plants. As some gases rise upwards and can be taken away with the wind, the concentration of gas in the air could be too low for tripping the alarms. Therefore, finding a better method for early detection of leakages is of great value to industries dealing with gases. As sound is emitted from the leakages when the gas exits its compressed container, the use of acoustic anomaly detection techniques is possible. A traditional signal processing method called the *Root-Mean-Square (RMS)* method was experimented with during a specialization project conducted during the fall of 2021, and did not produce satisfactory results. During this master thesis project, an experiment was conducted, utilizing the more modern machine learning methods *K-means clustering, Deep Neural Networks* and *Gradient Boosting*. Classification models has been developed for each of the different methods by training and testing the models on a dataset consisting of audio features from signals with and without gas leakages. The signals in the dataset were processed by use of signal processing techniques, and background noises with frequencies below 4000Hz were removed using a *Butterworth* filter. For the deep neural network method, two models were developed using the *PyTorch* an *FastAI* programming libraries. For the gradient boosting method, two models were developed, one with the *XGBoost* variant and one with the *Catboost* variant. Different combinations of audio features were used as input to the models. A performance evaluation of the models indicated that K-means did not separate the leakages from the non-leakages successfully. Both neural network models and both gradient boosting models performed adequately, as evident by accuracy scores of 89.32% and above. The results of the experimental study indicated that a system based on recordings from directional microphones and classification by machine learning methods is feasible for gas leakage detection. Based on the literature review of related work in the field of acoustic detection, as well as the promising results of the experimental study, the thesis has managed to serve as a proof of concept for the use of machine learning based acoustic anomaly detection techniques for gas leakage detection in noisy industrial environments.

Contents

List of Figures

List of Tables

1 Introduction

This chapter covers the background and motivation for the problem at hand in this master thesis. Further, the problem is formulated and described in detail. Then, the objective and approach of the practical work conducted in relation to the problem are presented. During a specialization project conducted in the fall of 2021, an extensive literature review was performed. The literature review, in combination with meetings with representatives from Equinor ASA, has laid the foundation for the defined problem and objective. Some findings from the specialization project on related work and the state-of-the-art equipment and methods for solving similar problems are included. Next, this master thesis' contribution to the field is declared. Finally, an outline of the report structure is given.

1.1 Background

The new hardware and software technologies that have become available in the last decades, along with the increased availability of data, have enabled the industries and society to change, innovate and improve quickly [72]. *Artificial Intelligence, Big Data analytics*, and *Cloud Computing* are some of the newer software technologies that have emerged [72]. *Robotics* and *Internet of Things* are examples of technologies that combine hardware and software to enable communication between physical and digital objects [62]. These emerging technologies have evoked the fourth industrial revolution, often called *Industry 4.0* [22]. The term was initially introduced in Germany to describe the digital transformation of the manufacturing industry but was later expanded to other industries [72]. Industry 4.0 refers to the current trend of digitizing and automating the processes in multiple industries by creating a cyber-physical system [22]. Industry 4.0 systems include intelligent networking of machines and processes with the help of information and communication technology (ICT). The leading countries in Industry 4.0 are countries with a high *Human Development Index* (HDI), such as Germany, the US, Singapore, and Switzerland [62]. The trend is also present in the Scandinavian countries, including Norway [22].

Through the Industry 4.0 paradigm, companies are given the opportunity to use the continuously increasing stack of new and improved technologies to optimize and digitize their processes and workplaces [62]. Industrial companies that work with, for instance, oil, metal, or energy generate a vast amount of data during daily processes that could be useful in combination with the mentioned technologies for gaining insights. Equinor ASA is one of these companies that want to optimize their enterprise by linking their physical systems to cyber-systems by use of ICT, *Robotics* and *Cloud Computing*. They also want to analyze data from their worksites through digitization, *Big Data analytics* and *Artificial Intelligence*. Digitization can be especially valuable for worksites that are mostly unmanned, such as remote oil and gas platforms far from land. When the available information comes solely from human workers, these sites rarely provide information when not visited often. Ideally, the company wants to have all possible information from their oil platforms at all times, but sending workers to do physical checks and measurements is costly. Maintaining production and control of oil and gas plants poses a high cost related to attention, travel, and use of resources, and an unnecessary amount of checks is therefore undesirable. Today Equinor uses many types of sensors on their platforms to measure different things like temperatures and oil flow in real-time in order to reduce physical visits. However, some things are not easily measured by sensors, such as gas leakages. Today, gas monitors are the standard tool for monitoring gas leakages through the concentration of gas in the air. These gas monitors have limited use, especially in outdoor environments, which include

oil platforms. Many gases rise in the air, and on outdoor platforms, the concentration necessary to trip the alarms may therefore take a much longer time to reach as the gas disperse into the sky. A concentration limit entails that a leak must have taken place long enough to trip the gas detector alarm system, costing the company a lot every minute the leakage continues. Therefore, the company has to carry out some monitoring in the form of manual checks to ensure leakages do not go undetected. Monitoring is essential to ensure that potential gas leakages or other flaws or disruptions to the normal processes are quickly discovered and handled. Ensuring a high level of monitoring while simultaneously keeping costs at a minimum is a challenge that the company is currently trying to solve.

1.2 Motivation

Representatives from Equinor ASA have thoroughly explained the motivation behind this thesis project. Although Equinor initiated this project to solve a problem on their oil and gas platforms, it can be helpful for many other industries that deal with other products. The motivation was previously discussed in the specialization project report [53] but will be reproduced here in order for the reader to understand the connection to this master thesis better.

Some procedures of leakage detection are based upon sound emissions from the gas exiting its compressed container. Trained personnel is sometimes able to hear abnormal sounds from a process or machinery, thereby detecting anomalies like gas and fluid leakages [37]. However, human hearing is limited, and small leakages can go undetected by even the most experienced worker [15]. Suppose humans can hear characteristics of abnormal sounds in pipes and machinery after some training. In that case, there should, in theory, be a way to extract the same characteristics with the use of software techniques. Today there exists many types of highly technological equipment for detecting sounds that humans cannot even notice [78] [11] [56]. The range of human hearing is 20Hz to 20kHz, and humans can only separate a difference of frequencies between around 200 Hz, while machines and robots do not have these limitations [15]. For the reasons mentioned above, the topic of detecting leakages in the context of industrial plants and processes has many possibilities.

The reasons why this topic is vital to investigate are numerous. Firstly, an ongoing leakage could have massive physically destructive consequences to machinery and objects in the leakage's surroundings [79] [76], especially if the contained gas that is leaking is toxic or corrosive [75]. Secondly, gas leakages could lead to poisoning if inhaled by workers in the area where an ongoing leakage has released a large amount of toxic gas into the air [75]. Another serious consequence of leakages is pollution to the environment from fluids that could be toxic, flammable, or corrosive [81] [75]. Thirdly, all the gas is contained for a reason and is either supposed to be used in processes on the working site or be transported to a customer [76]. The loss of gas would therefore result in unnecessary energy consumption and related economic impairment [81] [3]. According to Anthony Schenck et al. [3] one-third of the power consumption in compressed air and gas networks is lost due to leakages, which entails that leakages pose a significant threat to the economic side of the business owners of the gas pipelines. Lastly, leakages of highly flammable substances like hydrogen and methane can lead to accidents, fires, and explosions [16]. In many high-risk workplaces like oil- or melting plants, all kinds of equipment and processes can cause sparks and ignite nearby flammable objects if oil or gas leaks from the processes and gets ignited. One example of a recent incident like this is the fire on Equinor's methanol plant at Tjeldbergodden on the 2nd of September in 2020 [16]. Due to a malfunction in a steam turbine on the plant, chunks of metal were ejected from the turbine and created a hole in

a pipe. Lube oil then leaked from the pipe and ignited, causing a massive fire. Luckily there were no injuries, but the subsequent investigation concluded that:

"*... the outcome of the Tjeldbergodden incident could have been very serious, such as leakage and explosion of syngas. The potential included multiple fatalities, as several people were near the building where the fire occurred, at that particular time.*" [16]

Transportation using pipelines has become conventional in the modern industry in most countries, and a vast amount of gas is transported every day [79] [71]. Although energy consumption is the most common consequence of leakages, there can also be other more severe consequences such as the one just mentioned. In other words, detecting leakages as early as possible is of significant importance for companies and workers in any industry with high-risk environments. Any workplace with machinery and pipes will benefit from a system that detects leakages or other anomalies early. The longer the leakage or fault goes undetected, the more considerable impact it makes on the machines themselves, the risk to workers, and the energy consumption of the process, eventually leading to economic consequences due to wasted resources [3]. For all of the reasons mentioned above, it is valuable to investigate newer and better systems for locating and stopping gas leakages and other anomalies early.

1.3 Problem Description

There has already been conducted much scientific research on the topic of detecting acoustic anomalies using different methodologies. Moreover, as more and more companies and industries are being digitized, a vast amount of digital data are becoming available. Using methods designed for big data analysis is therefore highly advantageous in this field.

In a hypothetical finished assembled leakage detection system, sound recordings from around a worksite could be sent to an analysis software continuously. Sending continuous recordings would alert of any hypothetical leakages right away. Using continuous recordings would entail having recording devices or sensors attached to the objects or placed close to them at the worksite at all times. Alternatively, robots or personnel could be deployed to take recordings at specified times. Not sending information in real-time and rather choosing a duration to record before sending a discrete signal to the software would still alert of a potential leak, but the alert will come slightly later. The system designer could define a duration that the robot shall record at each spot on the worksite before sending it to the software. Doing that should not affect the result of the detection method. However, increasing the chosen duration will increase the delay of the system. Keeping the duration low, for instance, 30 seconds, would consequently delay the system by 30 seconds. This would still be a significant improvement to the current system of using gas detectors, which can delay the detection of leakages for hours or days.

An example use case is that the system is designed so that a robot is deployed to record sound for 30 seconds at each relevant spot on an oil and gas plant. After recording for 30 seconds in one room with gas pipes, it sends the recording to the software analysis program, which is stored in a cloud service. The program then searches for leakage characteristics through an acoustic anomaly detection method. If the classification algorithm labels the signal a leakage, an alert is sent to the system owner or personnel. An example architecture of a leakage detection system such as the one proposed is visualized in figure 1.

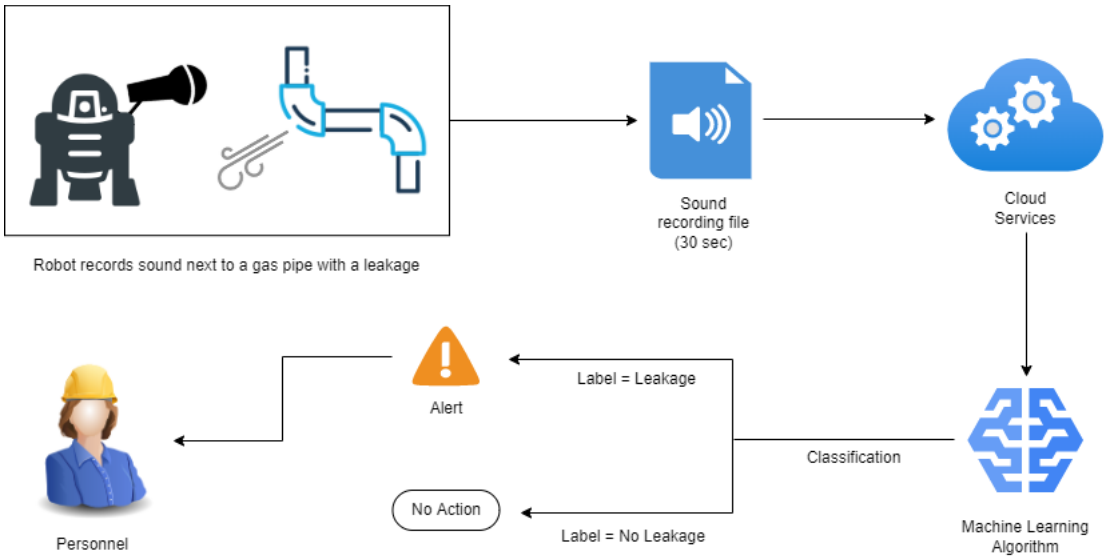


Figure 1: Architecture of a proposed leakage detection system.

Regardless if the system is based on continuous or discrete signals, the focus of this thesis revolves around exploring the use of machine learning for the detection of leakages. The problems or research questions to address in this master thesis are:

1. **How can companies detect leakages at an early stage by using machine learning based acoustic anomaly detection methods, designed for a large amount of data?**
2. **Is a system such as the one proposed in figure 1 feasible in industrial environments?**

1.4 Objective and Approach

The main objective of this master thesis is to provide a proof of concept for an acoustic leakage detector based on machine learning, which should be capable of giving early and correct warnings to the system owners of newly occurred leakages.

During the former specialization project, some experimentation using a basic digital processing technique was conducted on a dataset of audio signals. The technique involved using a single audio feature extracted from the signals, namely the Root Mean Square energy (RMS) feature, and defining a maximum limit for the RMS-value of the signals. This limit would alert the system that a leak was present if it was exceeded. At best, this method classified the signals with only 54.69% accuracy. The project report concluded that using only one feature for classification was not extensive, complex, and accurate enough. However, the RMS-values of some of the signals with a leakage was slightly higher than the ones without a leakage. Therefore, the RMS-value was deemed possible to include in classification methods combined with other features. As a consequence of the poor results from the project report using a basic digital processing technique, the choice of trying more modern machine learning techniques in the further master thesis work was made. [53]

In order to reach the objective of this master thesis, the following tasks were defined:

1. Conduct a literature review on the state-of-the-art machine learning approaches for acoustic anomaly detection, with a focus on gas leakages.
2. Establish a plan for developing one or more machine learning models, and design the framework for training and testing them with a dataset.
3. Construct a balanced, high-quality dataset of audio recordings by sorting and filtering available historical data.
4. Perform signal processing techniques on the dataset signals to reduce noise and prepare them for further use.
5. Extract relevant audio features from the signals to be used as input to machine learning algorithms.
6. Train and test the chosen machine learning models on the dataset and present the performance results.
7. Conduct a feature importance analysis and implement the necessary changes to the feature combinations to improve model performance.

The first task of the above-mentioned ones was the main objective of the previous specialization project and has been performed thoroughly already. Still, some further research has also been conducted in this master thesis period to ensure a good choice of machine learning models to use, as well as correct implementation of the methods. The findings from the previous project implied that the *K-means Clustering Algorithm* and *Convolutional Neural Networks* should be further investigated for classification of leakages. Furthermore, the research conducted during the master thesis period implied that *Gradient Boosting* had potential for this problem and should also be explored. As Convolutional Neural Networks work best with images and not numerical values, a natural choice was to change from CNN with spectrograms as input to using Deep Neural Networks (DNN) with audio features as input. The reason for choosing this approach was to be able to compare the performance of all ML models on the same numerical input and thereby identify the most fitting model. The same dataset that was used in the previous project has also been used in this master thesis work. Five different machine learning models have been developed using *Python* as the programming language with several helpful libraries for feature analysis and machine learning model development.

1.5 Related work

An extensive literature review was conducted during the specialization project, and related work in the field of acoustic anomaly detection was presented in detail in the report. The essence of the discoveries about the state-of-the-art equipment and methods are included in this section.

1.5.1 Detection Equipment

There are several ways to detect sound, and different types of equipment can be used for the purpose of leakage or anomaly detection. *Contact microphones*, also known as *Piezo microphones*, are microphones assembled onto an object to measure the object's vibration [11]. This kind of microphone has a good sensation and is less impacted by noise than a microphone that is not in contact with an object [11]. However, there are some drawbacks to using these as detection devices. One drawback of using contact microphones is that they are greatly affected by the vibrations of the object they are attached to, which could camouflage the sound of a leakage close by the microphone on another object. Since industrial plants consist of many pipes that could stretch kilometers in total length, another drawback of using contact microphones is that the amount necessary to attach to the system could be tremendous and naturally also expensive. The companies also need to evaluate if it is desirable to assemble something onto the existing structure for safety reasons.

An alternative to contact microphones is *Acoustic Cameras*, which is an imaging device used to locate sound sources. The primary use of an acoustic camera is a dynamic visualization of sound propagation, but it has also been used to detect machinery or equipment failures by characterizing abnormal sounds. Pavlikova Ľudmila [84] showed during their research that acoustic cameras are perspective tools in the area of predictive maintenance as it enables early identification of machinery and equipment failures by sound analysis. Another type of camera that can be used for anomaly detection is a camera that captures temperature. As a leakage provides an expansion to the air around it, it will decrease the temperature, thus indicating anomalies at the location. These tiny temperature changes could be detected by *Thermal* and *Infrared cameras*. A research survey on the applications of thermal cameras was conducted by Rikke Gade et al. [56], where the authors concluded that gas leakage detection was a possible application for them. Sandsten et al. [33] used IR-cameras for detecting highly flammable gasses such as methane, ethylene, and ammonia. There is a practical issue related to the use of cameras as they have to be capsuled and ATEX approved. ATEX approval is necessary for being able to operate a high-risk industrial plant. The ATEX directive (2014/34/EU) applies in all workplaces in the EU and EØS, and it describes minimum safety requirements for workplaces containing explosive atmospheres, such as oil and gas plants [17].

As there are some limitations to the mentioned detection equipment, the current state-of-the-art in compressed air leak detection is to use hand-held devices with highly directional ultrasonic microphone arrays, which are called *Directional Microphones*. Even though using a microphone that is standing freely in a room will result in a more significant impact of background noise than using a contact microphone [78], it is more applicable on large systems and allows for the possibility of mobility. Since the directional microphone picks up sound from a specific area or direction, it can find the direction from which a leak originates. The amplitude of the leakage will seem higher if the microphone is directed at it. Thus, it can be helpful in the task of not only detecting a leak in the area but also locating it. Yukinori Nagaya et al. [80] showed during their work on cavitation detection that directional microphones obtained a greater improvement of sensitivity than non-directional microphones.

1.5.2 Methods for Acoustic Anomaly Detection

Scientists have researched several acoustic anomaly detection methods to detect anomalies such as leakages. Basic signal processing methods and machine learning methods have been broadly used for anomaly detection.

Basic signal processing methods are based on mathematically manipulating signals. T. Shindoi et al. [65] used *Adaptive Digital Filtering (ADF)* for diagnosing plant equipment by signal processing. According to the researchers, the noise that the leakages produced was barely perceptible to the human ear, but the filter separated the leakages from the normal signals. The ADF method can compare an input signal to the desired signal (which would be a signal without a leakage) and calculate the mean squared error. If the input signal and the desired signal are exactly the same, the method generates an all-pass filter that passes through the entire signal. A signal with a leakage should, in theory, give a high mean squared error, indicating that an anomaly is present. The ADF method creates a filter that passes normal sounds, and if there are characteristics of an abnormal sound present, the signal will not be passed through.

Another basic signal processing method for stationary and non-linear signals is the *Hilbert Huang Transform (HHT)*. Zhang Shuqing et al. [81] attempted to apply HHT to the problem of detecting leakages in gas pipelines. The researchers simulated artificial leakages in between two dynamic pressure transmitters positioned 12500km from each other. They attempted to use HHT to calculate the location of the leakages based on the point of dynamic pressure signal and time difference. Their results showed that the HHT-method could detect the pressure change accurately and find the location with good precision. Their results are based on sensors inside gas pipelines.

The *Root Mean Square method (RMS)* is a basic signal processing method that tries to take advantage of the increased energy in the signals with a leakage. The method works by taking the rms-energy values of a signal and comparing it to a predefined rms-threshold in order to alert of any anomalies [14]. Dimitrios Kampelopoulos et al. [14] introduced the use of an rms-based approach that relies on monitoring the rms-values of the signal and defining a threshold based on previous rms-values. By the logic of this method, a leakage is only recognizable when the rms-value becomes higher than the mean rms. This method was experimented with in the previous project [53], and gave below 60% accuracy for the dataset it was tested with, which contained signals with and without a leakage.

In recent years, machine learning models have been favored compared to basic signal processing methods. *Support Vector Machines (SVM)* is a supervised machine learning method widely used for pattern recognition and classification problems [58]. Tianshu Xu et al. [71] presented in September 2021 a pipeline leak identification method for a spherical detector based on combining variational mode decomposition and a support vector machine. The Mel frequency cepstral coefficients (MFCCs) were extracted from leakage signals and used to create a characteristic vector for the SVM-based leakage detector. Their experimentation with the SVM resulted in a classification accuracy of 93% [71]. Rui Xiao et al. [58] introduced in 2019 an acoustic method for natural gas leak detection based on Wavelet Transform and SVM. Wavelet transform was used to de-noise the signals, and a time-frequency feature was used to identify characteristics of different leakage severity. The researchers argued that SVM has excellent generalization properties, handles outliers and nonlinear boundaries simultaneously, and finds the optimal global solution. Their arguments were supported by an accuracy of 95.60% for the leakage detection model.

An unsupervised machine learning technique commonly used to find patterns in a dataset with different classes is *clustering* [24]. Cheng-Yu Peng et al. [10] proposed a system for the detection of abnormal vibrating sounds from the milling process of a Computer Numerical Control (CNC) machine by use of the K-means clustering method. Their preprocessing phase consisted of filtering the sound using a band-pass filter and analyzing the resonant frequency with Fast Fourier Transform (FFT). The frequency results were then passed to the K-means clustering algorithm for classification, which yielded satisfactory results.

Another very popular machine learning technique is to use neural networks, which can be structured in different ways. Some drawbacks of neural networks are that in order for them to perform well, they need to be trained on a big dataset with adequate quality [27]. Changes in background noise, which is the essence of the industrial environments, could significantly impact the trained model's performance. However, neural networks are widely used for sound classification tasks and have shown promising results in noisy environments [83]. *Convolutional Neural Networks (CNN)* is a neural network that is structured by three specific types of network layers: a convolutional layer, a pooling layer, and a fully connected layer. By sending image representations of signals, such as spectrograms and MFCCs, as input to a CNN, the network could learn to classify different classes of sounds, like "Leakage" and "No Leakage" sounds [83]. Zhejian Chi et al. [83] published a study on using logarithmic spectrograms as input to a deep Convolutional Neural Network (deep-CNN) for the classification of environmental sounds. One issue with CNNs in relation to dynamic environments, like industrial plants, is that "normal" background noise could differ from what the networks were initially trained on. Manabu Kotani et al. [43] looked into this issue and presented a study in 2019 on the topic of using neural networks for leakage detection in noisy industrial environments. Their study aimed to introduce a modular neural network that handles the dynamic environment by adding modules to the network after a given time.

An *Autoencoder* is a neural network that has a different structure than the CNN and learns to copy its input to its output [61]. The network contains an input layer and an output layer, and the purpose of the algorithm is to reconstruct the inputs by minimizing the distance between the input and output layer[82]. In between these two outer layers, there is an encoder and a decoder that conducts the reconstruction. Taha Berkay Duman et al. [66] used Convolutional Autoencoders (CAE) for feature extraction and as an end-to-end anomaly detector for industrial plants. They did not focus on leakages but instead on other abnormal sounds like explosions, fire, and glass breaking. Their results demonstrated that the CAE is successful when the sounds stem from industrial plants where cutting and welding processes exist.

1.6 Contribution

The contribution of this master thesis to the science field is twofold. One is to accumulate relevant research on the topic of detecting acoustic anomalies. The other is to apply some of the relevant state-of-the-art machine learning methods for acoustic anomaly detection on a dataset that fits specifically to the problem of detecting gas leakages in industrial conditions. A proof of concept is presented for a specific detection system using machine learning classification methods, a mobile robot, and directional ultrasonic microphones.

1.7 Outline

Throughout the entirety of this thesis, the focus will be on the problem of detecting gas leakages in industrial environments. The contents of this thesis firstly consist of the introduction, which includes the motivation for the topic and the problem description. Thereafter, relevant theory regarding how to process audio signals will be explained in section 2, including feature extraction methods and classification methods. Research on the state-of-the-art acoustic anomaly detection (ADD) techniques used for similar problems will be introduced in this section. The three utilized classification methods, K-means, Deep Neural Networks, and Gradient Boosting, will also be thoroughly explained. Furthermore, section 3 will familiarize the reader with the content and structure of the dataset that was used for experimentation. Methodology and software tools used during the experimentation will be discussed closely in section 4. The results produced will be displayed in section 5, including a performance evaluation. Section 6 will consist of a discussion regarding the findings, as well as a comparison of the different classification methods. Lastly, the thesis ends with a short summary in section 7, followed by implications as to the focus of further work on the leakage detection problem.

2 Theory

This section of the thesis will explain the theory related to sound and methods for modifying, analyzing, and visualizing sound. Further, some state-of-the-art approaches for acoustic anomaly detection will be introduced.

2.1 Signal Processing

Sound is usually produced by the vibration of objects. Vibrations cause air molecules to oscillate, thus creating a change in air pressure. Sound can also be produced by air leaking from a compressed air system, such as a tube or a vent. These air leaks also change the air pressure. This change in air pressure produces a mechanical wave [15]. A mechanical wave is energy or oscillation that travels through space. These waves carry information about the characteristics of the sound and its source, such as frequency, intensity, and energy [15]. The sound is captured digitally by taking samples of the air pressure over time. The most common rate at which sound is sampled is 44100 samples per second, or 44.1kHz [74], which is double the frequency of the human hearing limit. The digital representation of a sound will be further referred to in this thesis as a *signal* or *audio signal*. The process of extracting and analyzing features of signals digitally is called signal processing [32]. Signal processing is important in this project as every signal from the dataset needs to be processed properly in order for it to be used for further work with machine learning methods.

2.1.1 Audio Features

Machine learning algorithms need data to train and test on. It is not common to use a whole signal (represented by a vector of numbers) as input to a machine learning algorithm, but rather to extract several audio features from the signal and use a combination of these as input [19]. Different audio features capture different aspects of a sound. Audio features are either extracted through the time-domain or the frequency-domain. An audio feature based on the time-domain displays the changes in a signal over time. An audio feature based on the frequency-domain shows characteristics of a signal within each given frequency band over a range of frequencies and is unrelated to time.

2.1.2 Time-Domain Features

Some time-domain features that are frequently extracted and used for signal processing and classification tasks are the following:

- **Amplitude Envelope**: Measure of the changes in the amplitude of an audio signal over time. The measure is an influential property as it affects our perception of timbre. [21]
- **Zero Crossing Rate (ZCR)**: Measure of the number of times the amplitude of a discrete-time signal in a given time interval changes from a positive to a negative value (and visa versa) [6].
- **Root Mean Square energy (RMS)**: Measure of the average magnitude, or power, of a signal. [14]

$$\text{Zero Crossing rate} = Z_t = \frac{1}{2}\sum_{n=1}^{N} |sign(x[n]) - sign(x[n-1])| \tag{1}$$

$$\text{Root Mean Square} = RMS_t = \sqrt{\frac{1}{T_M}\int_0^{T_M} x^2(t)dt} \cong \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} x^2(n)} \tag{2}$$

In equation 1, the *sign* function is 1 for positive arguments and 0 for negative arguments, while $x[n]$ is the time-domain signal for frame n. N is the total number of frames in the signal [73]. For the RMS equation (see 2), x represents the signal, T_M is the time duration of the signal, and N is the number of samples [44]. T_M is equal to the number of samples N times the sample rate.

As the time-domain features displays changes over time, and time is not a relevant factor for the leakage detection problem, they have not been at focus in this thesis. Of the three time-domain features above, only the RMS feature has been utilized in the experimental study, as the focus has been on the spectral components of the signals regardless of time. As will be further explained later in the thesis, the RMS feature has been calculated based on frequency intervals of 4000Hz for the experimental study. Therefore, the RMS has acted like a spectral feature as it shows the RMS per frequency bin.

2.1.3 Frequency-Domain Features

A frequency-domain analysis can give a better understanding than a time-domain analysis since a signal is tacitly based on the breaking down of intricate sounds into their separate component frequencies. Frequency components and characteristics of a sound are seen as unrelated to time when analyzing features in the frequency-domain. Some frequency-domain features commonly used for signal processing and classification tasks are:

- **Spectral Centroid**: Measure of the shape of the spectrum. The spectral centroid can be viewed as the spectrum's 'center of gravity'. A higher value corresponds to more energy of the signal being concentrated within higher frequencies. [68]
- **Spectral Bandwidth**: Measured of the weighted average of the distances between the spectral components and the spectral centroid. [5]
- **Spectral Rolloff**: Measure of the amount of right-skewness in the power spectrum. It points to the frequency bin in the power spectrum below which around 80-90% of the distribution magnitude is concentrated. [68]
- **Spectral Flatness**: Measure of how much noise-like a sound is, as opposed to being tone-like. [36]
- **Power Spectral Density (PSD)**: Measure of the power distribution of a signal in the frequency domain. PSD shows which frequency bin the biggest power distribution is located. [39]

The spectral features are defined as the following:

$$\text{Spectral Centroid} = \text{SC}_{i,b} = \frac{\sum_{u=l_b}^{u_b} u|f_i(u)|^2}{\sum_{u=l_b}^{u_b} |f_i(u)|^2} \tag{3}$$

$$\text{Spectral Bandwidth} = \text{SB}_{i,b} = \frac{\sum_{u=l_b}^{u_b} (u - \text{SC}_{i,b})^2 . |f_i(u)|^2}{\sum_{u=l_b}^{u_b} |f_i(u)|^2} \tag{4}$$

$$\text{Spectral Rolloff} = \text{SR}_i = \sum_{u=1}^{R_i} M_i(u) = c * \sum_{u=1}^{N} M_i(u) \tag{5}$$

$$\text{Spectral Flatness} = \text{SFM}_{i,b} = \frac{\left[\prod_{u=l_b}^{u_b} |f_i(u)|^2\right]^{\frac{1}{u_b - l_b + 1}}}{\frac{1}{u_b - l_b + 1} \sum_{u=l_b}^{u_b} |f_i(u)|^2} \tag{6}$$

$$\text{Power Spectral Density} = PSD(f) = \frac{|X(T,f)|^2}{T} = \tag{7}$$

In the equations above, $f_i(u)$ is the Fourier transform of the ith frame of the signal, and u is the frequency band ranging from $u \in [0, N]$. N is the highest frequency that can be captured and stored, and it is a bit less than half of the sampling rate of the signal. For a signal with a sampling rate of 48kHz, N would be $\simeq$24kHz. The features are extracted on non-overlapping logarithmically spaced bands, where l_b and u_b are the lower and upper edges of the band b [5]. For equation 5, R_i represents the frequency bin below which the c percent (usually $c = 80 - 90$ %) of the magnitude distribution M of the Fourier transformed signal is concentrated for frame i [73]. For the Power Spectral Density (equation 7), $X(T, f)$ is the Fourier Transform of the signal over the interval $[0, T]$, and f represents the frequency [57]. The Fourier Transform and how it is implemented for the signals used in this project will be further explained in section 4.2.2.

2.2 Machine Learning

Artificial intelligence (AI) is the simulation of human intelligence in machines that are programmed to conduct different human-like operations. Machine Learning has, in recent decades, become the most popular sub-field of AI. One advantage of machine learning is its ability to handle a vast amount of data. Today, many companies generate data continuously through their daily processes. Making such data available and analyzing them could open up possibilities to gain valuable insight into how the companies can optimize their processes and work more efficiently, which is the very vision behind Industry 4.0. Machine learning can make use of this data and even analyze and give answers or solutions in real-time. [29]

The problem of detecting gas leakages from sound can, through machine learning terms, be expressed as an acoustic anomaly detection problem. Further, since there is either a leakage present or no leakage present in a signal, the task for the machine learning model to solve is a binary classification task. The aim for a machine learning model is to classify each signal into the classes *1* or *0*, which respectively means *leakage* or *no leakage*. Therefore, it is necessary to introduce the basic concepts of machine learning and classification. [29]

Machine Learning is categorized into either unsupervised or supervised learning. The difference between the categories is that unsupervised learning methods handle a set of unlabeled data, while supervised learning methods handle labeled data. Supervised learning, therefore, builds on the principle of knowing the answer in advance and training the algorithms to learn the relationship between the input x and the output y. The goal of supervised learning is to approximate the mapping function $f(x) = y$ so well that one can predict the output variables for new input data. The learning algorithm of a supervised learning model trains on historical data, including the output y, in order to configure the mapping function f. When this function f is given a new input value x that it has never seen before, it will predict the output value y. [29]

In contrast, the goal of unsupervised learning is to model the underlying structure or patterns in the data in order to learn more about it. Unsupervised learning methods can be applied to many different data sets originating from several areas of real life. Some use cases where these methods have been successfully applied for many years are market segmentation, identifying crime areas, and detecting insurance fraud. In both unsupervised and supervised approaches, the input x would be a vector consisting of the audio features, and y would be either *1* or *0*. [63]

2.2.1 Classification

Classification tasks involves the definition of k categories or classes. In binary classification tasks k is equal to 2. A software program created for solving the classification problem will have the task of selecting which of the k classes a given input belongs to. The program uses a learning algorithm to classify each input by trying to produce a function $f : \mathbb{R}^N \rightarrow \{1, ..., k\}$. The input data x will be assigned to a class y through $y = f(x)$. Different types of learning algorithms can be used to create the function f, and both supervised and unsupervised learning methods are applicable for classification tasks. Some examples of learning algorithms commonly used for classification are Random Forest, Support Vector Machines, K-means, Gradient Boosting, and Artificial Neural Networks. The latter three will be further explained as they have been experimented with during this thesis. [29]

2.2.2 Clustering

One common method for searching through unlabeled data sets for patterns is *Cluster analysis*, also called *Clustering*. Clustering involves dividing a set of data into subsets called clusters, where each cluster contains data points with more similarity to each other than the data points in the other clusters, as one can see in figure 2. Another way to look at it is that the algorithm tries to separate data points with significant dissimilarity by measuring the divergence of the data. The method is *a priori* because it does not have information about the data or the groups in advance. Clustering is not one specific algorithm but rather a general technique for statistical data analysis. Clustering is at the core of unsupervised learning, and the most commonly used clustering algorithm is *K-means*. K-means is efficient for numerical problems. Since audio features from signals are represented by numerical values, this algorithm is applicable for the leakage detection problem. Ideally, the algorithm would assign all leakages in one cluster and all non-leakages in the other. Alternatively, there could be more than two clusters, such that the leakages would be separated into different clusters based on the severity of the leakage. [4]

The K-Means algorithm's main objective is to assign each data point in a dataset to one of the k clusters, where k is a number chosen by the data analyst. Specifically, the algorithm's input is a training set of data points $X_1, ..., X_n \in \mathbb{R}^d$, and a number k which represents how many clusters one wants the algorithm to produce. [4]

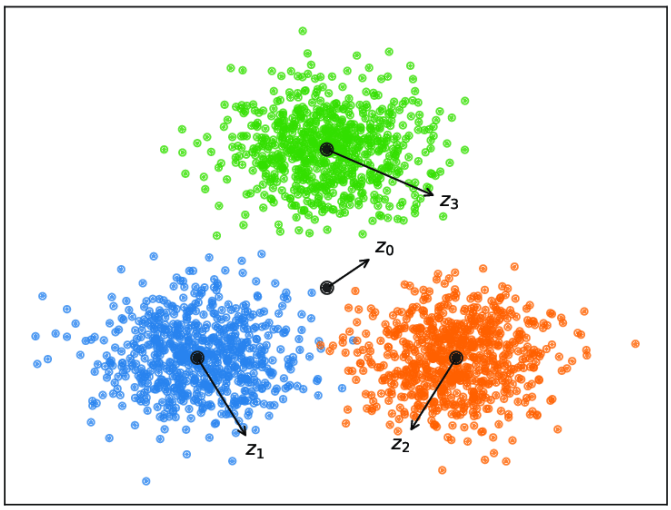


Figure 2: Visualization of clustering with clusters of data points separated by different colors and the related centroids represented by the black middle points (z_1, z_2 and z_3).

Source: Liyan Xiong et al. [41]

The K-means algorithm starts with a randomized selection of centroids $m_1^{(0)}, ..., m_k^{(0)}$. An iterative optimization process then begins. For each iteration i, all data points are assigned to the nearest centroid by use of a distance function, expressed in equation 8, which in turn creates the clusters $C_1^{(i+1)}, ..., C_k^{(i+1)}$. Thereafter new centroids, $m_1^{(i+1)}, ..., m_k^{(i+1)}$, are defined by calculating the mean vector of each cluster, expressed in equation 9. This iterative process runs until it converges, meaning the centroids remains the same and the clusters are completely formed. The K-means algorithm is popular because of its simplicity and generality, as one can see from the pseudo-code in algorithm 1. [4] [24]

Algorithm 1 K-means algorithm

procedure K-MEANS($X_1, ..., X_n$, k) ▷ Input: set of data points, # of desired clusters
  Randomly initialize centroids $m_1^{(0)}, ..., m_k^{(0)}$
  while not converged **do**
    Assign each data point to the closest centroid by use of equation 8:

$$X_s \in C_k^{(i+1)} \iff ||X_s - m_k^{(i)}||^2 \leq ||X_s - m_l^{(i)}||^2, l = 1, ..., K \tag{8}$$

    Compute new centroids, $m_1^{(i+1)}, ..., m_k^{(i+1)}$, by use of equation 9:

$$m_k^{(i+1)} = \frac{1}{|C_k^{(i+1)}|} \sum_{s \in C_k^{(i+1)}} X_s \tag{9}$$

  end while
  return $C_1, ..., C_k$ ▷ Output: k final clusters
end procedure

The K-means clustering algorithm is an applicable method for audio classification and anomaly detection by use of a combination of numerical audio features. Cheng-Yu Peng et al. [10] proposed a system for the detection of abnormal vibrating sounds from the milling process of a Computer Numerical Control (CNC) machine by use of K-means. Their system consisted of recordings of the CNC machine by using an Intopic JAZZ-016 mini-PC desktop microphone and classification using the K-means algorithm. Their preprocessing phase consisted of filtering the sounds using a band-pass filter and analyzing the resonant frequency with Fast Fourier Transform (FFT). The frequency results were then passed to the K-means clustering algorithm for classification, which yielded satisfactory results. The researchers' choice of audio features for input to the K-means algorithm was just the frequencies from the signals. Other audio features can also be used as input, like the ones mentioned in 2.1.1.

The performance of the K-means algorithm is highly dependent on the dataset, or more specifically, the similarities or distance between points numerically in the dataset. Furthermore, different results of the algorithm may also appear depending on the algorithm's initialization of centroids, and in the worst case, the result may fall into the local optimum instead of the global optimum. For the number of clusters k, the dimensions d, and the number of data points n, a problem can be solved by K-means in $O(nkdi)$ time, where i is the iterations needed for convergence. Since the necessary iterations i are approximately the same as the number of data points n, the time complexity becomes $O(n^2)$. The effective complexity is therefore quadratic, and directly proportionate to the number of data points to be clustered. The K-means algorithm may also produce disproportionate cluster sizes, as it is a greedy algorithm. This method could be used to learn more about the patterns in the dataset and also for classification purposes. [4] [42]

2.2.3 Deep Neural Network

Artificial Neural networks are machine learning systems inspired by neuroscience, or more specifically, the human biological brain. Brains consist of neurons that send information between each other, and the neural networks try to model this phenomenon by having layers of nodes called artificial neurons that transmit information. Neurons in one layer receive input from the prior layer, compute an activation value, and transmit the value forward in the network to the next layer of neurons. These networks are often referred to as *feed-forward network* since the information is transmitted forward in the network, from the initial input x, through the layers, and eventually to the output y. [29] [49]

The depth of a network indicates the total number of layers L that are used to map the relationship between the input and the output. *Deep Neural Networks* are neural networks with multiple hidden layers between the input and output layer, and they are therefore considered to be *deep*. The overall structure of a deep feed-forward neural network can be seen in figure 3. For classification tasks, the input x is mapped to a class y by a function f^*. The function f^* is approximated by the mapping $f(x;\theta) = y$, which is defined by the deep feed-forward network. The network tries to learn which value of the parameters θ that provides the best function f. The mapping function f is usually composed of several different functions in the different layers l, which form a chain $f(x;\theta) = f^L(f^{L-1}...(f^2(f^1(x;\theta^1))))$. Each layer's function provides an output or a signal that is sent forward to the next function, typically in the form of vectors. The hidden layers $l = 1, 2, ..., L-1$ transform the data, while the input layer ($l = x$) provides the input data without transformations, and the output layer ($l = L$) converts the last transmitted signal (or vector) to the final output $\hat{y}$. [29] [49]

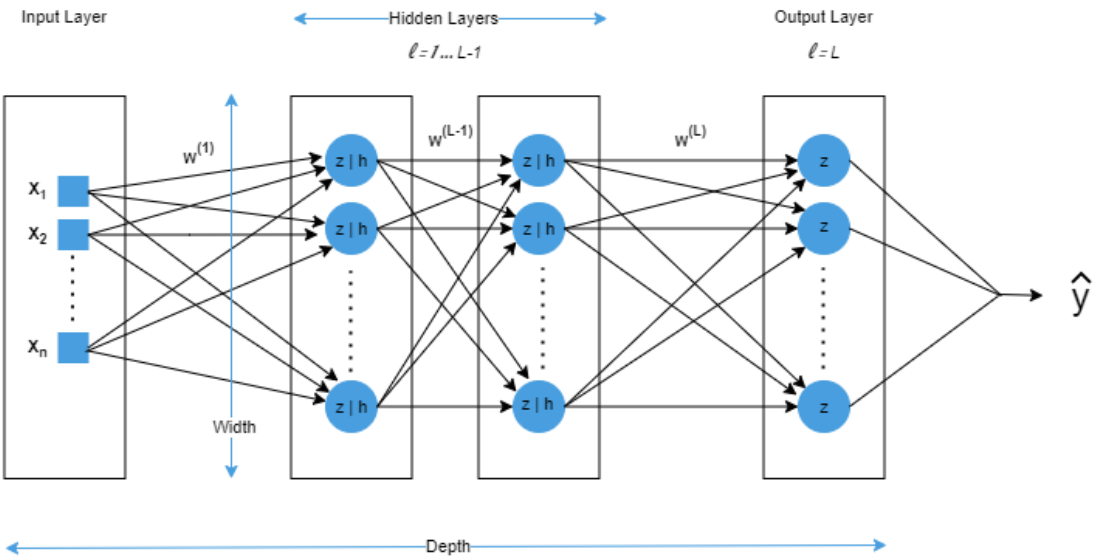


Figure 3: Structure of a deep neural network. Illustration inspired by Nikhil Ketkar [49] and Ian Goodfellow [29].

The artificial neurons are the basic building blocks in the network. One neuron is basically a function that takes in the input vector z_i and produces a scalar h_i by equation 10, where w_i is a weight vector and b is a bias term.

$$h_i = \sigma(\sum_{i=1}^{n} z_i w_i + b) \tag{10}$$

The functions that transform the vectors in each neuron can be of different types, and are called *Activation functions*. There is no state-of-the-art activation function $\sigma(x)$ that works for all problems, and there is no guideline for which to use for a specific machine learning task. The data analyst must simply test out the different activation functions to see which performs best for each problem. The most common activation functions are *ReLU, Sigmoid, Linear* and *Tanh*, and they are defined in equation 11, 12, 13 and 14 [49].

$$\sigma_{ReLU}(x) = max(0, x) \tag{11}$$

$$\sigma_{Sigmoid}(x) = \frac{e^x}{1 + e^x} \tag{12}$$

$$\sigma_{Linear}(x) = ax \tag{13}$$

$$\sigma_{Tanh}(x) = tanh(x) \tag{14}$$

For the Linear activation function above, a denotes a constant. The ReLU function transforms negative values to zeros and has shown promising results in gradient-based learning problems [49].

A visualization of how the activation functions work inside a neuron is shown in figure 4. As seen from the figure, the bias is not sent from the prior neuron calculation but can be considered an additional input to the neuron and is constant. The weights control the strength of the connection between two neurons, and there is one weight for every input-to-neuron connection in the network.

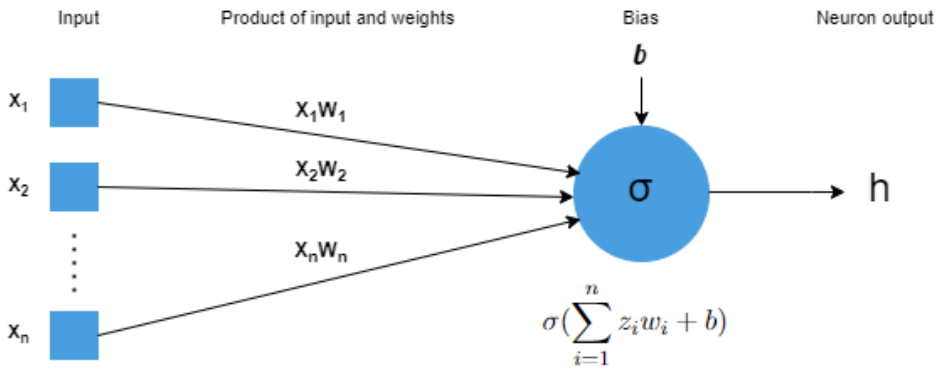


Figure 4: Structure of an artificial neuron with its input and output. Illustration inspired by Nikhil Ketkar [49].

A neural network can also be recurrent, meaning it sends information backward in the network after sending it forward. These types of neural networks are called *Recurrent Neural Networks*. Recurrent networks produce an output in each layer and back-propagates through the network layers, giving the output back to a layer as a new input. The structure is similar to the feed-forward network in figure 3, except it has an additional arrow going backward from one layer to the prior layer, creating a loop.

Some drawbacks of neural networks are that in order for them to perform well, they need to be trained on an extensive dataset with sufficient quality [27]. Changes in background noise, which is the essence of the industrial environment, could significantly impact the trained model's performance. However, neural networks are widely used for sound classification tasks and have shown promising results in noisy environments [83].

2.2.4 Gradient Boosting

Gradient boosting in a supervised machine learning technique used for regression and classification tasks. The technique iteratively builds a predictive model by creating multiple "weak" learners and combining them to get a single "strong" learner. Gradient Boosting builds on the *ensemble technique*, which is a technique that has been a very effective tool for improving the performance of multiple existing models. Ensemble methods mainly rely on randomization and generate many diverse solutions to the problem at hand and combine the different base models to produce one optimal predictive model. [9]

The word *ensemble* means collection, and ensemble methods are based on learning from an ensemble of previous weaker models, which is typically a decision tree. Gradient Boosting builds on the idea of ensemble methods and tries to "boost" weak models to be better. The word *gradient* points to the fact that gradient boosting is an iterative functional gradient algorithm. A gradient algorithm minimizes a loss function by iteratively choosing a function that points towards the negative gradient. The gradient boosting algorithm has three main components [9]:

- **A Loss Function**: The role of the loss function is to estimate how good the model is at making predictions with the given data. The loss function calculates the pseudo residual, which is the difference between the predicted and true output. [70]
- **A Weak Learner**: A model that classifies the data but does so poorly and has a high error rate. This is typically a Decision Tree [9].
- **An Adaptive Model**: The iterative and adaptive approach of adding the weak learners (or trees) one step at a time. After each iteration, the value of the loss function should be reduced, and it should be closer to a final "strong" model. Existing trees in the model are not changed, but new trees are added using the gradient descent procedure in order to minimize the loss. [9]

A visualization of the gradient boosting method's iterative process of adding decision trees is included in figure 5.

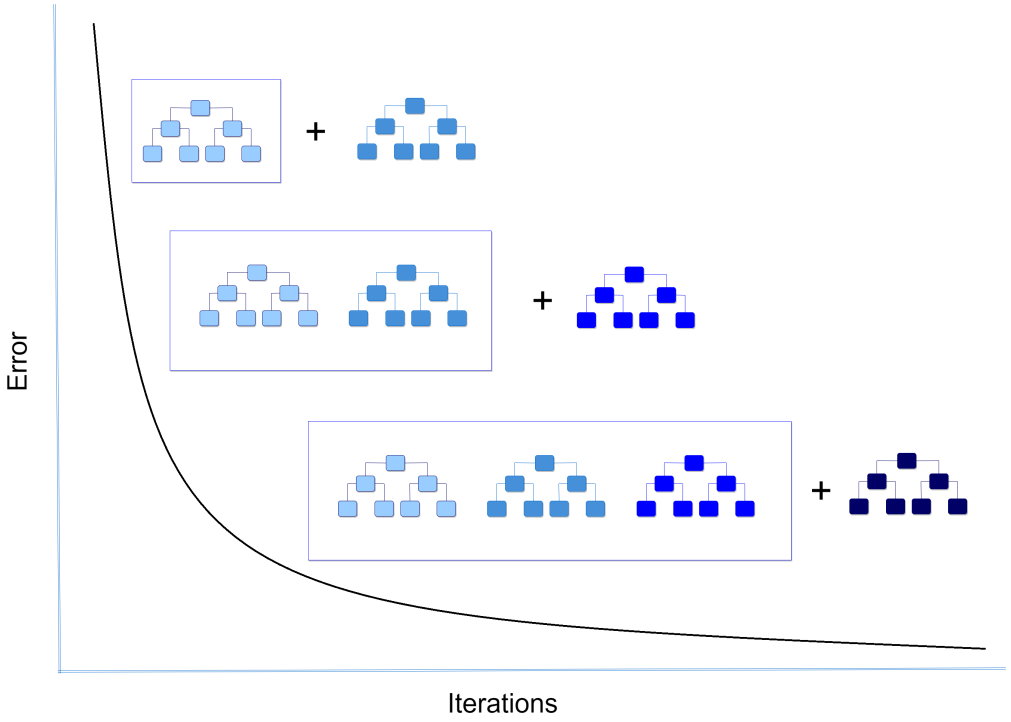


Figure 5: Visualization of the Gradient Boosting Method. The method adapts the model by adding decision trees at each iteration. Illustration inspired by Syahidah Izza Rufaida et al. [64]

Gradient boosting is a greedy algorithm, as it selects the best option available at each iteration, regardless if another option would have provided the overall optimal result [70]. There are several variants of gradient boosting, two variants being *eXtreme Gradient Boosting*, more familiar as *XGBoost*, and the *Categorical Boosting*, also called *Catboost* [9]. *XGBoost* is a more regularized form of Gradient Boosting. XGBoost uses advanced regularization (L1 & L2), which improves model generalization capabilities [30]. Both XGBoost and Catboost tries to approximate the mapping function $f(x) = y$ for the given input x and output y, by minimizing the expected value of a given loss function $L(y, f(x))$ [9]. The loss function in binary classification tasks are usually a logarithmic function defined by equation 15, where p_i indicates the probability of the i-th instance assuming the value y_i [55].

$$L = -\frac{1}{N}\sum_{i}^{N}((y_i \log p_i) + (1 - y_i)\log(1 - p_i)) \tag{15}$$

For the XGBoost, each ensemble model uses the sum of D functions to predict the output. Equation 16 shows this sum, where F is the function space of the classification tree models, f_d is an independent tree structure, n is the number of observations (data samples), $\hat{y}_i$ is the predicted output and x_i is the input for each individual sample i [30].

$$\hat{y}_i = F(x_i) = \sum_{d=1}^{D} f_d(x_i) \qquad f_d \in F, i = 1, ..., n \tag{16}$$

The objective function of the XGBoost algorithm is made up of the logarithmic loss function L and a regularization expression Ω [9]. The regularization expression is essential for avoiding overfitting as it penalizes various parts of the algorithm and controls the complexity. The regularization expression can weigh the contribution of each tree to the sum of trees to slow down the learning. This weighting is called a shrinkage or a learning rate, and it is defined by equation 17 [70]. The shrinkage consists of the sum of the score of the j-th leaf w_j^2 for the total number of leaves T, and the tree pruning parameter Υ which regulates the depth of the tree and the regularization parameter λ which is associated with l2-norm of the scores vector.

$$\Omega = \Upsilon^T \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{17}$$

It is common to constrain the weak learners in specific ways, such as a maximum number of layers, leaf nodes, or by shrinkage. This is to ensure that the learners remain weak but can still be constructed in a greedy manner. The effect of the constraints is that learning is slowed down, in turn requiring more trees to be added to the model. More trees can require more training time, providing a configuration trade-off between the number of trees and the learning rate. XGBoost solves problems with a time complexity of $O(Kd||x||_0 \log n)$, where K is the total number of trees, d is the maximum depth of each tree, and $||x||_0$ is the non-missing entries in the training data. [9]

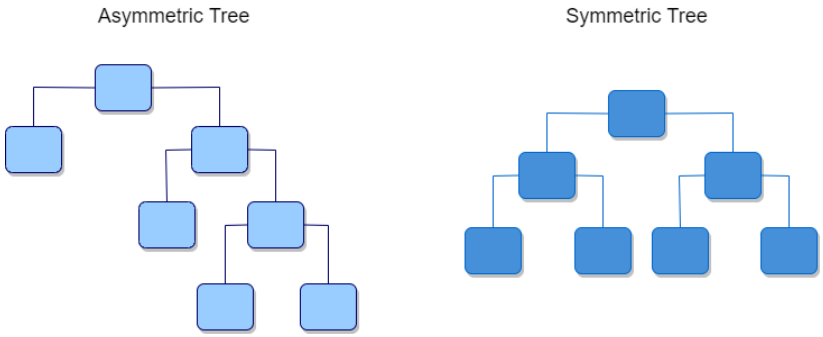


Figure 6: Illustration of an asymmetric and a symmetric decision tree.

Catboost is a gradient boosting method that can be used on categorical data as well as numerical data. The method performs better than XGBoost, but it also takes a longer time for the training phase. Compared to the XGBoost method, Catboost always builds symmetric decision trees, while XGBoost can build asymmetric trees (see figure 6). During each iteration, the leaves from the previous decision tree are split using the same condition, and the feature-split pair that provides the lowest loss is chosen. In addition, the Catboost method uses the concept of ordered boosting, which is a permutation-driven approach to train models on a subset of data while calculating residuals on another subset, thus preventing target leakage and reducing the chance of overfitting. Still, gradient boosting methods will continue to minimize all errors when irregularities in a dataset exist, which overemphasizes outliers and can cause overfitting. The time complexity of Catboost is $O(sn)$. [2] [9] [40]

2.2.5 Training, Validation and Testing sets

In order to create a machine learning model that performs well in classifying new data, three different types of data sets are required. These are the *training, testing* and *validation sets*. The *training set* is the data samples used during the training phase of the model, where the model tries to learn the relationship between the input data and the output. Therefore, the training set includes the output information, which are the class labels that belong to the input. This is the most important data set as the model learns from it and configures the parameters of the models to fit the data in this set. [29] [35]

The *validation set* contains the samples of data used to evaluate the model during the training phase. Machine learning models contain hyperparameters that define the structure of the model and determine how the model is trained. In neural networks, the structure is related to how many hidden layers the model contains, which is considered a hyperparameter. Another important hyperparameter is the learning rate, which defines how quickly a network updates its parameters. A low learning rate slows down the learning process and converges smoothly. In contrast, a high learning rate speeds up the learning but may fluctuate a lot and possibly not converge. Hyperparameters are set before the training phase, and the validation set is used to evaluate the model's performance given a fixed combination of hyperparameters. As the validation set is used during the training phase and can leak information on the output, another set is necessary to evaluate the final model's performance. The validation set is usually constructed of 20% of the samples from the training set, leaving the size of the training set as around 80% of the initial size. [29] [35]

The *testing set* contains the samples of data that do not contain information about the output and is used to provide an unbiased evaluation of the final model. The model tries to predict the output y from the new input data x in the testing set based on the learned relationship or the classification function $f(x) = y$. This set can not contain the same samples as the other sets. The predicted output labels are compared to the actual labels in order to evaluate the model. [29] [35]

2.2.6 Overfitting and generalization

There are several challenges involved in machine learning tasks. One of the biggest challenges is that the finished machine learning model must perform well on new inputs and not only those used to train the model. The ability to perform well on previously unseen inputs is called *generalization* [29]. In order for the model to generalize well, it needs to find the equilibrium between *overfitting* and *underfitting*.

Overfitting occurs when the model has learned the characteristics of the relationship between the input and the out too well [29]. It can happen if the data points in the training set have high similarity. During the training phase, the training error is measured and reduced recursively until it converges. When the trained model is tested on new data with lower similarity to the training set, it performs much worse than the training error indicated. The testing error describes how well the model is able to generalize, and it is found by testing the trained model on a new set of data points called the testing set [29]. The testing error is often referred to as the generalization error. When the gap between the training error and the testing error is large, the phenomenon of overfitting occurs.

During the optimization process of the model, data characteristics stored in the validation set leak into the model due to the fact that the model is optimized to best predict the output variable of the validation set [29]. This is known as information leakage, and the more the model is adjusted based on the performance on the validation set, the more information from the validation set is leaked into the model. For this reason, the problem of overfitting can occur. An illustration of the overfitting and underfitting phenomenon in relation to a clustering example is included in figure 7.

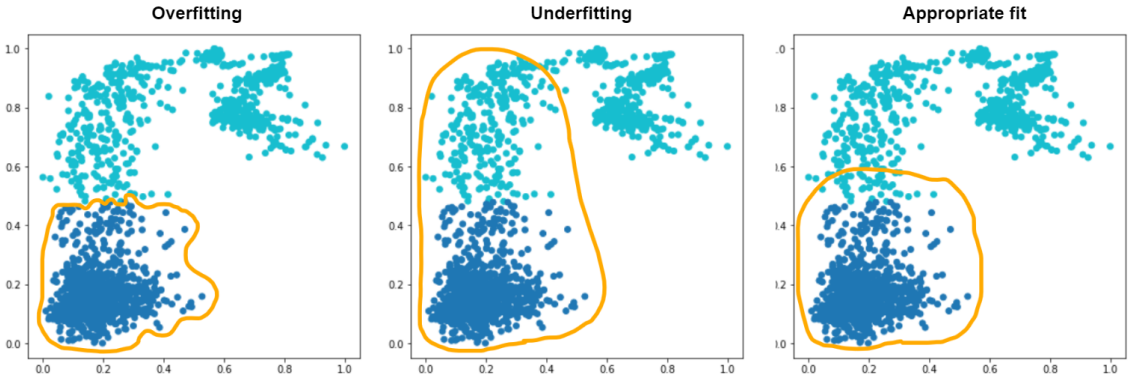


Figure 7: Visualization of overfitting and underfitting of clusters in a classification problem, where the blue colors are the actual class labels and the yellow color draws a hypothetical class for the dark blue data points. Illustration inspired by Aleix Solanes and Joaquim Radua [1].

In contrast, *underfitting* occurs when the model has not learned the characteristics of the relationship between the input and the output well enough. When the training error is high, the model generalizes too much and is not able to produce the correct outputs [29]. Finding the equilibrium between overfitting and underfitting can be tricky. The optimal point is where the training error is slightly lower than the testing error, as depicted in figure 8.

Several measures can be taken in order to reduce the generalization error, thereby avoiding overfitting or underfitting. One way is to use a training set with high dissimilarity or by choosing the data points that go in the training set randomly. Another measure is to add more data points to the training set. The mentioned measures are related to changes in the data set, but adjustments to the machine learning algorithm could also be made. For example, the hyperparameters of the model could be modified, and constraints can be set for how detailed information the model is allowed to learn by adjusting the learning rate. The learning rate controls how much the model should change in response to the estimated error during training [29].

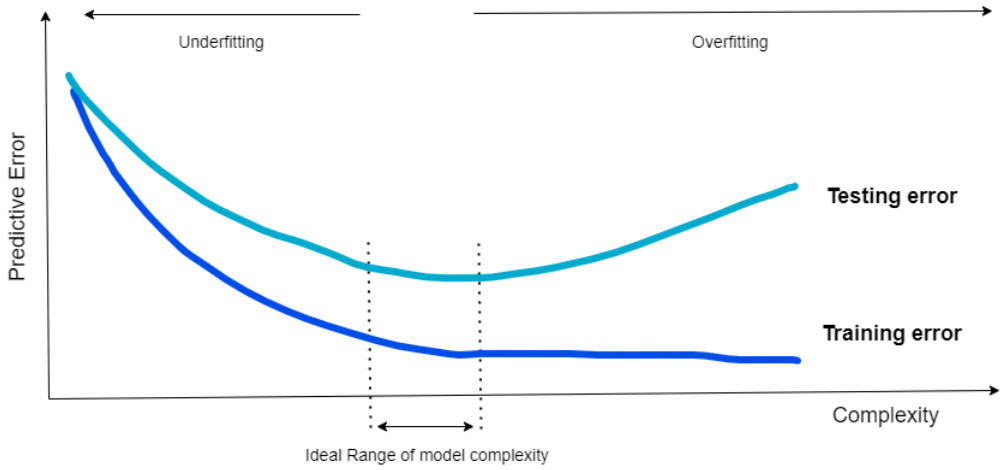


Figure 8: Visualization of the overfitting and underfitting phenomenons seen in relation to testing and training error. Illustration inspired by Hayder Al-Behadili et al. [25]

3 Data

In order to look further into the problem of detecting leakages from sound, some acoustic data was required. A large set of data was necessary for testing out signal processing techniques and classification methods. With almost all classifiers, an adequately balanced data set with good quality is crucial for achieving optimal performance [38].

3.1 Data Collection

In order to have a large and balanced dataset to work with, a public open-access dataset from the *Fraunhofer Institute for Digital Media Technology* was collected [12]. The dataset consists of 3680 files, which contains recordings of different industrial background noises, with and without leakages, and of different types and sizes. Earthworks M30 Omnidirectional Measurement microphones were used as recording devices. The name of the dataset is *IDMT-ISA-COMPRESSED-AIR*, but it will be further referenced in this thesis as the *leakage dataset*.

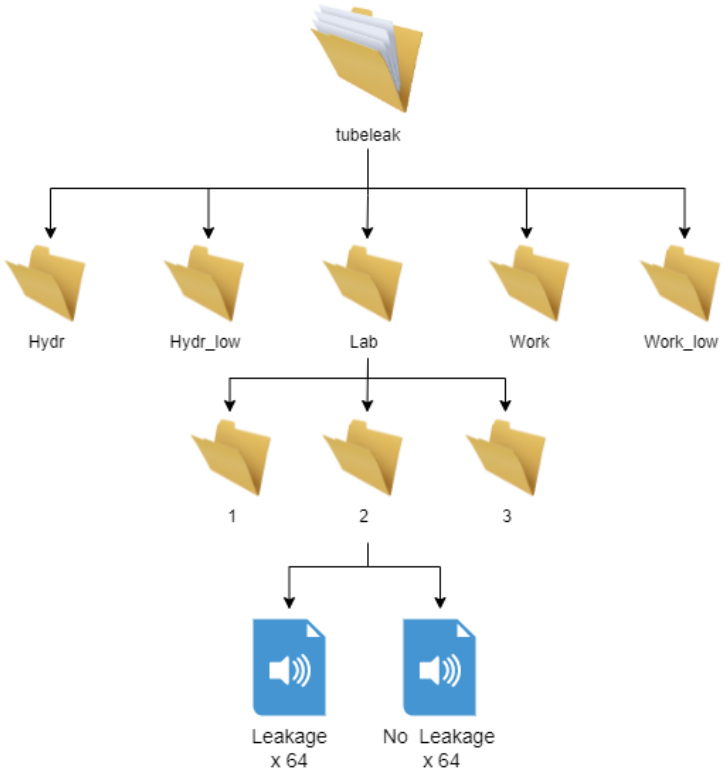


Figure 9: The structure of the leakage dataset.

3.2 Dataset structure and content

In figure 9 a visualization of the leakage dataset's structure is presented. The structure consists of to main folders: *ventleak* and *tubeleak*. These main folders have five sub-folders with different types of leakages and environments at focus. These five types are: *Hydraulic machine noise with high volume*, *Hydraulic machine noise with low volume, Laboratory* (no added background noise), *General factory workshop noise with high volume*, and *General factory workshop noise with low volume*.

In each of the five sub-folders, there have been three recording sessions labeled in their own folder as *1, 2,* or *3*. In all recording sessions, 64 of the audio files are with a leakage, while an additional 64 are without a leakage. Figure 9 presents the content of the tube leakages folder, but the folder with vent leakages looks almost identical. The only difference in the *ventleak*-folder is that it has one less recording session of type *General factory workshop noise with low volume*, and only 96 files in the first recording session of the *Laboratory* type. In the latter mentioned recording session, there are 64 files with a leak and 32 files without.

Each audio file's content is expressed in the filename. The files are named with the format *S_ L_ N_ M_ .wav*. First, the recording session number *1, 2* or *3*, is presented by *S*. Thereafter, *L* describes if there exists a leakage or not, where *niO* means leakage and *iO* means no leakage. The *N* represents the knob rotations on the vent or tube, which is between *0.0* and *9.0*. Lastly, there have been used 4 different microphone configurations, labeled *1-4*, for the recordings, and also two volume settings, *l* for low volume and *m* for maximum volume, which is represented by *M*.

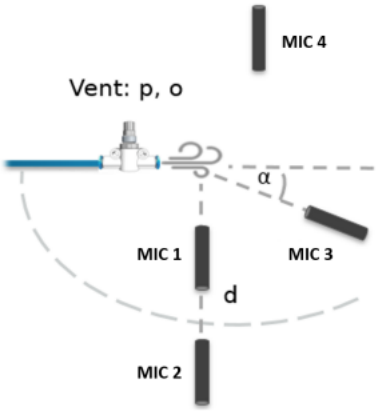


Figure 10: Depiction of the microphone positions relative to the vent outlet.

Mic	**Distance** (d)	**Angle** (α)
1	20 cm	90°
2	2 m	90°
3	20 cm	30°
4	Full Room	-

Table 1: Specifications on the microphone placements.

The creators of the data set used different microphone configurations in order to open the possibility of evaluating the influence of microphone placement on leakage detection. The microphones were positioned in four different locations, demonstrated by figure 10. The fourth microphone is positioned higher than the others, close to the ceiling of the recording room. The rest are oriented towards the leakage source at an angle α and a distance d. Table 1 gives an overview of the locations of all microphones. Unfortunately, the creators have not included information about the size of the recording room, only that the fourth microphone was placed in the center of the room and above the leakage source.

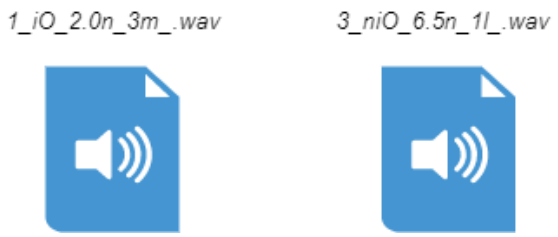


Figure 11: Examples of two audio files with recording information.

Figure 11 shows two examples of audio files and their filenames. The first file contains a signal from recording session nr.1, with no leakage, using 2.0 knob rotations and microphone configuration nr.3 with maximum volume. The second file contains a signal from recording session nr.3, with a leakage, using 6.5 knob rotations and microphone configuration nr.1 with low volume.

The dataset was almost perfectly balanced, to begin with, with 50.43% of the files containing leakages. There was only one recording session, *Vent Laboratory 1* that had double as many files with leakages. A positive note on this dataset is that the researchers had taken recordings both in a laboratory and on a working site. Recordings from the laboratory make it possible to hear a leakage without any noise, thus creating the possibility of analyzing a leakage sound by itself. This has, however, not been investigated much in this project. The recordings on the working site are especially valuable as it contains actual industrial workplace background noises such as welding, cutting, and banging noises. Therefore, the environment in which the recordings have been taken is highly similar to the environment for which the leakage detection methods discussed in this project have their intended usage. Thus, his dataset fits the problem description of this thesis very well.

4 Methodology

This section of the thesis will address the approach and technologies used for experimentation on the gas leakage dataset mentioned in section 3. The aim of the experimentation was to implement some of the state-of-the-art methods for acoustic anomaly detection introduced in section 2, and analyze the performance of the methods.

4.1 Libraries and Tools

The machine learning models presented in this section, along with all associated signal processing and data analysis, were programmed in *Python version 3.8.5*. Python was chosen as the programming language because it contains useful audio analysis libraries. *Jupyter notebook*, which is a web-based interactive computing platform, was used as the integrated development environment (IDE) [69]. Jupyter Notebook is handy for smaller code snippets with the intent to display visualizations and plots, as it combines common developer tools into a single graphical user interface (GUI). It was, therefore, a natural choice as an IDE for this project. In addition to the well known open source libraries *NumPy*, *Matplotlib*, *os* and *csv*, the following libraries were used during programming:

- **Librosa**: And open-source Python library for music and audio analysis. Librosa is often used when the problem to solve revolves around using audio data like in music generation, speech recognition, and sound classification. The library provides the building blocks necessary to create an audio information retrieval system [8]. In this project, the library was used to load the audio signals from the .wav-files in the dataset and thereafter to extract different audio features from the signal.
- **SciPy**: An open-source Python library containing efficient algorithms for linear algebra, sparse matrix representation, basic statistical functions, and other special functions [52]. For this project, the libraries' built-in functions for applying windowing and filters to a signal were utilized.
- **Pandas**: An open-source library for Python which provides data structures and data analysis tools. This library simplifies the data preprocessing step of storing large amounts of data by using *Pandas Dataframes*. Functions for sorting, filtering, visualizing, and describing the content of the Pandas Dataframes are included in this library [77]. For the problem at hand in this project, this library was used to store all signals from the dataset, along with its features in a Dataframe.
- **Scikit-Learn**: An open-source library for Python which provides tools for data analysis. It includes functions for machine learning algorithms such as clustering, in addition to functions for calculating evaluation metrics of ML-models [18]. For this project, the library was used to train and test the K-means clustering algorithm on the dataset. The built-in function for calculating a confusion matrix based on the performance of all the presented classification models was also used.
- **Seaborn**: An open-source Python library for visualizing data. It includes a high-level interface, and statistical graphics [45]. This library was used in this project to present the confusion matrix of the classification models' performance in a more reader-friendly way. Additionally, it was used to create a heatmap that demonstrates the correlations between the features used as input to the models.

- **AutoGluon**: An open-source library for Python which enables the use of automatic machine learning (AutoML). AutoML includes automated stack ensembling, deep learning, and real-world applications spanning image, text, and tabular data. This library enables ML beginners to prototype deep learning and classical ML solutions for raw data. It allows users without expert knowledge to utilize state-of-the-art ML techniques [47].

All the listed coding libraries above are well-known for being up-to-date with new technologies and state-of-the-art methods, and are constantly configured to improve run-time or add new functionality. These have all been around for years and are popular in the coding community, as they are considered to be efficient, reliable and accurate. [28]

4.2 Data Preprocessing

Preprocessing data is a fundamental phase in data analytics, regardless of the model approach. Several steps are involved in data preprocessing, which depend on the dataset and the data type. The steps involved in this project include structuring the dataset, processing the signals, extracting features from the signals, and normalizing the features. All these steps will be thoroughly explained. A visualization of the architecture for the data preprocessing of an example signal, including all the executed steps, can be seen in figure 12.

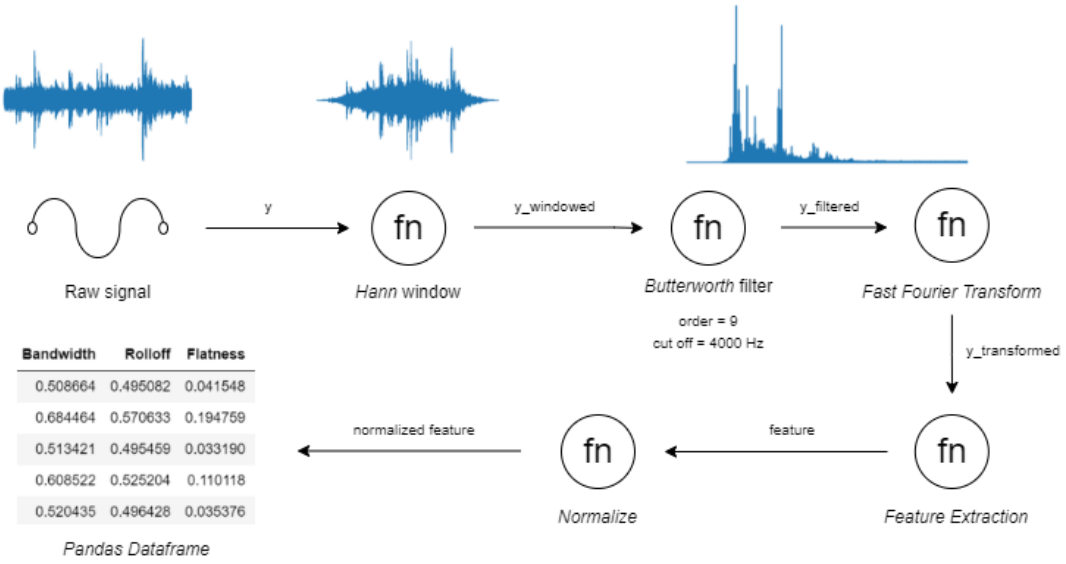


Figure 12: Diagram of the steps in the data preprocessing phase for one signal.

4.2.1 Structuring the data

As described in section 3 the raw data used in this project was initially structured into several folders containing .wav-files of audio recordings. In order for these recordings to be used as an aggregated dataset, they needed to be restructured and sorted. The descriptive filenames of the .wav-files contained useful information and were used to sort the recordings. The following procedure was done for all folders and sub-folders in the dataset:

1. **Saving paths**: Beginning at the main folder, *tubeleak*, all sub-folders were iterated through while saving the local file paths of each signal file in a list (or column). The same process was repeated for the *ventleak*-folder. The reason for saving the file paths into a column was to be able to find each recording much easier in the future by going directly through the path while simultaneously gathering all files into one collection. In addition, having all files together allows for easier reorganization and sorting of all files at once.

2. **Filtering the data**: For the single recording session inside the *ventleak* folder that did not contain the same amount of leakage signals as non-leakage signals, the excess 32 leakage files were removed. These were removed from the dataset in order to have a perfectly balanced dataset containing the same amount of files for both classes.

3. **Labeling the data**: A sorting algorithm was created in order to sort the files by their content. For each path in the list of paths, the algorithm searches for the word '*niO*' and labels the signal as a leakage by the number *1* if it finds it. If the path only contains the word '*iO*', it labels it as a non-leakage by the number *0* instead. This information is saved to a column. The third to last index in the path contains information on the microphone configuration that was used. These labels were also saved to a column.

4. **Defining the data structure**: The columns that were just created were then merged together and stored as a two-dimensional matrix-similar data structure called *Pandas Dataframe*. Each row in the dataframe contains information about one file (or signal) and the column-names are *Path, isLeak* and *Mic*. The dataframe structure can be seen in figure 13. For the *tubeleak*-folder there were 1920 files and consequently 1920 rows in the *Tube*-dataframe. For the *ventleak*-folder, there were 1760 files, and after removing the 32 excess leakage signals, the *Vent*-dataframe contained 1728 rows.

	Path	isLeak	Mic
0	C:/Users/Marianne Pettersen/PycharmProjects/So...	0	1
1	C:/Users/Marianne Pettersen/PycharmProjects/So...	0	1
2	C:/Users/Marianne Pettersen/PycharmProjects/So...	0	2
3	C:/Users/Marianne Pettersen/PycharmProjects/So...	0	2
4	C:/Users/Marianne Pettersen/PycharmProjects/So...	0	3
...	...	...	...

Figure 13: The initial Pandas Dataframe that was created for the *tubeleak*-folder.

4.2.2 Processing signals

The first step of processing a dataset of signals is to ensure all signals are in the same standard format and use the same sample rate. Suppose the audio signal is stereo, meaning that multiple channels have been used to convert the sound to a digital signal. In that case, it should be converted to mono by averaging the right and left channels [5]. The dataset used in this project contained all mono signals, and it was therefore not necessary to do any converting in this case. After checking that the signals are in the correct format, they need to be windowed and filtered to avoid artifacts and to prevent noisy data from impacting the performance of the later constructed model [48]. As all files were appended together in one dataframe for each of the main folders, the signals could be easily loaded from the paths in order to be processed further. All signals had the same sample rate of 48 000 Hz (48 kHz) and lasted approximately 30 seconds. Therefore, the signals did not need to be re-sampled.

The necessary signal processing steps were executed in a specific order for all signals in the dataset in order to ensure consistency. The following procedure was done for all signals:

1. **Windowing using the *Hann Window***
2. **Filtering with the *Butterworth filter***
3. **Transforming to the frequency-domain using *Fast Fourier Transform***

The steps above are done sequentially in the order listed. These steps are common during audio-related tasks such as speech recognition and acoustic anomaly detection [68].

Windowing Signals

Smoothing windows are often used to improve the spectral characteristics of a sampled signal. When performing spectral analysis on finite-length data, smoothing windows can be used to minimize the discontinuities of truncated waveforms, thus reducing spectral leakage and artifacts [48]. A smoothing window operates by multiplying the time waveform by a finite-length window (equal to the number of samples in the signal) with an amplitude that varies smoothly and gradually toward zero at the edges. The window function used in this project is the *Hanning* window, often called the *Hann* window. Equation 18 defines the Hann window $w(n)$, where M is the total number of samples and n represents the current sample. [67]

$$w(n) = 0.5 - 0.5\cos(\frac{2\pi n}{M-1}), \quad 0 \leq n \leq M-1 \tag{18}$$

If the waveform (often referred to as amplitude envelope) of a signal is plotted, one can see how the amplitude changes over time. Figure 14 displays the waveform of a signal from the *tubeleak*-folder. After the *Hann* window (visualized in figure 15) is applied to the signal, the waveform changes at the start and end of the signal's duration. Figure 16 displays the signal's waveform after being windowed. The amplitude envelope of the signal has been smoothed at the edges, and the chance of spectral leakage and artifacts has therefore been reduced.

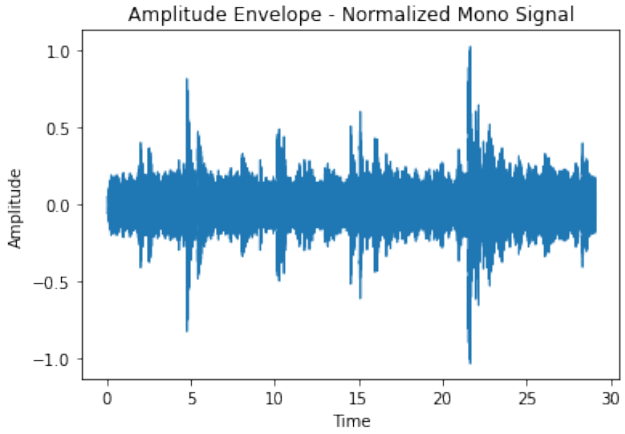


Figure 14: Plot of the Amplitude Envelope for a signal in the dataset. Amplitudes has been normalized in the range [-1, 1].

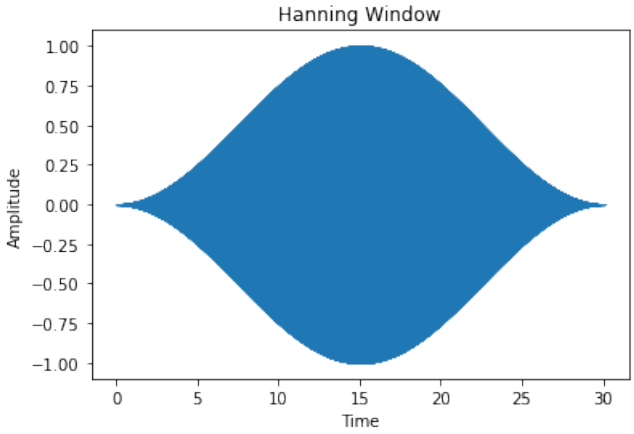


Figure 15: Plot of the Hanning Window for a signal in the dataset.

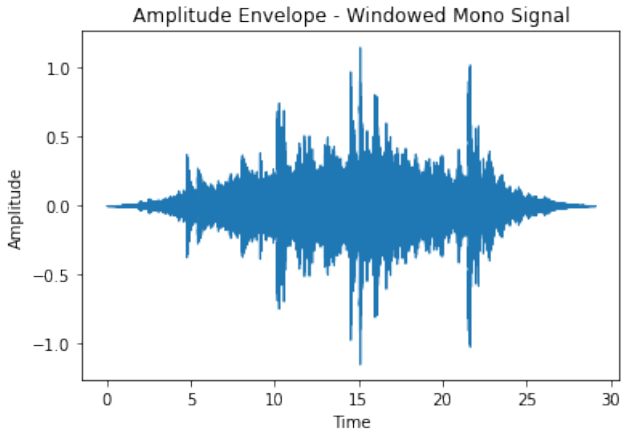


Figure 16: Plot of the Amplitude Envelope for a signal in the dataset after a Hanning window has been applied. Amplitudes has been normalized in the range [-1, 1].

Applying Audio Filters

Audio filters are frequency depended circuits designed to amplify or attenuate a specific range of frequency components from a signal [31]. A high pass filter (HPF) is an equalization tool that removes all frequencies below a set point, and a low pass filter (LPF) removes all frequencies above a set point. In other words, the filters remove unwanted specific frequencies while allowing the remaining frequencies to pass through. High pass filters are commonly used for reducing noise in audio signals. Noise is usually low-frequency and can come from wind, movements during recordings, and other sources. Background noise in signals is undesirable and can interfere with the central message of the signal and mask the information content. In this project, both high pass and low pass filters have been used in order to focus the analysis on different frequency ranges.

Digital filters can be classified into *Finite Impulse Response (FIR)* and *Infinite Impulse Response (IIR) filters*. FIR filters should be used when phase information is required. FIR filters require more processing power and are more work to set up, which is undesirable. IIR filters are suited for applications like this one, which do not require phase information. There are some advantages to using IIR filters compared to FIR filters, which include the following [54]:

- Less number of arithmetic operations
- Shorter time delays
- Less number of side lobes in the stop band
- More susceptible to noises

For the reasons just mentioned, an IIR filter was chosen for this project. A popular IIR filter is the *Butterworth* filter. The Butterworth filter is a signal processing filter that can serve as both a high pass and a low pass filter. It has a maximum flat frequency response, which means there are no ripples in the passband and zero roll-off response in the stopband. The filter requires two parameters: the cut-off frequency f_c and the order n. Based on the order, the filter rolls slowly toward the cut-off frequency in a linear manner instead of cutting out frequencies exactly at the defined cut-off. A visual description of this can be seen in figure 17, which depicts how the slope steepens as the order increases. The Butterworth filter transfer function $H(jw)$ is defined by equation 19, where w is the angular signal frequency, w_c is the angular cut-off frequency, and n is the order. The cut-off frequency is expressed as an angular value w_c, which is equal to $2\pi f_c$.

$$|H(jw)| = \frac{1}{\sqrt{1 + (\frac{w}{w_c})^{2n}}} \tag{19}$$

For all signals filtered with the Butterworth filter, an order of $n = 9$ was used. The reason for using a high order is that high orders provide greater roll-off rates between passband and stopband. At the same time, as different types of audio features were retrieved from the signals, different frequency ranges were in focus. For some of the simple frequency-domain features such as *Spectral Bandwidth*, *Spectral Rolloff* and *Spectral Flatness*, the frequencies below 4000 Hz was removed using the Butterworth high pass filter with $f_c = 4000$. Removing these lower frequencies was a measure done to reduce the impact of background noise.

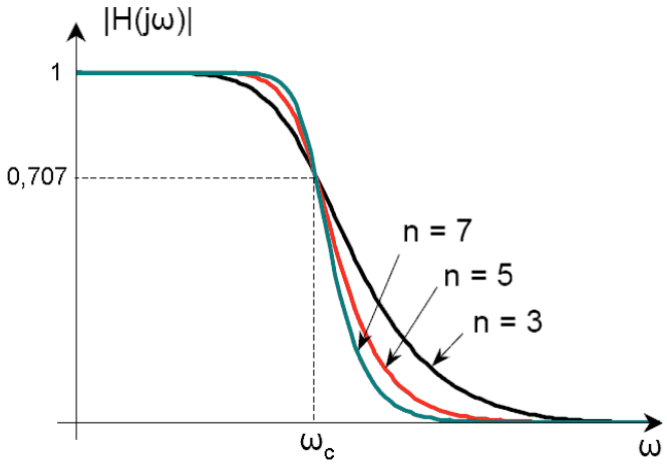


Figure 17: The Butterworth filter frequency response with different orders n.

Source: Bogdan Mihai and Panu Mihai [7]

For the *Spectral Centroid* and the *RMS* feature, the signal was divided into 6 different frequency bins at an interval of 4000 Hz by using both the high pass and low pass Butterworth filter with $f_c = [4000, 8000, 12000, 16000, 20000]$. The result of dividing these two features is that they create six new features for each. The reason for doing this will be further explained in section 4.2.3, which includes a description of how the features were extracted and how the importance of each feature was analyzed.

As the range of human hearing is between 20 Hz and 20 000 Hz, looking at frequencies above 20 kHz (which is defined as ultrasounds) is especially interesting. Human ears will not detect leakages with a frequency above this value, and they are therefore more important for a leakage detection system to detect. Therefore, some audio features were used to focus solely on ultrasonic frequencies. These features included *Power Spectral Density*, *Maximum Amplitude* and *Integrated Power*, which will be thoroughly described in section 4.2.3. In addition to those, the last frequency bin for the *Spectral Centroid* and the *RMS* feature also considered ultrasonic frequencies as the frequency interval in those bins was 20-24 kHz.

Transforming to Frequency-domain

Each signal in the dataset is presented in the time domain and lasts for approximately 30 seconds. The time domain shows how each signal changes with time. Theoretically, the signal's properties could be different at the start of the time interval compared to the end. This could, for instance, happen if a leak were to appear at the 25th second of the signal's duration. Regardless if there were to appear a leakage in the middle of the recording or if it existed already and is present during the whole signal's duration, there should be an alert from the leakage detection system. Assuming that an appeared leakage does not go away on its own and needs to be stopped by personnel, the signal should be labeled as a leakage immediately after the leakage is detected by the system. In other words, if the system were to detect a leak on the second 25th, it is safe to assume that the remainder 5 seconds of the signal also contains a leakage.

Naturally, if robots are used to take recordings, they could be programmed to start at the same place, during the same time of day, and for the same duration each time. However, these recording conditions do not need to be exactly the same each time. Even with a signal that only lasts for one second, there is still important information available for extraction. In other words, a leakage detection system can analyze signals of any duration. For these reasons, time is irrelevant during the analysis. Frequency is, on the other hand, highly relevant for the analysis. To retrieve frequency information from the signals, the signal has to be converted from the time domain to the frequency domain. In order to convert a signal to the frequency domain, the *Fast Fourier Transform* (FFT) algorithm is used.

The FFT algorithm rapidly computes the Discrete Fourier Transform (DFT) of a sequence of values, thereby decomposing a signal into frequency components. By using FFT on the data size N, the complexity is reduced from $O(N^2)$ to $O(NlogN)$ compared to the standard DFT, which saves computing time. The DFT $\overline{\mathrm{X}}_k$ is defined in equation 20, where x_m represents the sequence of values and N is the total number of values [26].

$$\begin{aligned} \overline{\mathrm{X}}_k &= \sum_{m=0}^{N-1} x_m W^{mk} & W &= e^{-\frac{j2\pi}{N}} \\ k &= 0, ..., N-1 & j &= \sqrt{-1} \end{aligned} \tag{20}$$

The Cooley-Tukey algorithm is the most commonly used FFT algorithm [51]. It works by recursively breaking down the DFT in equation 20 of size N into smaller DFTs of composite size $N = N_1 N_2$. The mathematical derivation of the Cooley-Tukey algorithm is out of the scope of this thesis. However, it is applied to the signals by using the embedded FFT-function in the *NumPy*-library [50]. After the signal has been transformed to the frequency domain, it is sent to the feature extraction methods, which will be introduced in section 4.2.3. The same signal that was visualized in the time-domain in figure 16 is visualized in the frequency domain in figure 18. The result of using the FFT algorithm on a signal is visualized by a Frequency-Magnitude Spectrum such as the one shown in figure 18.

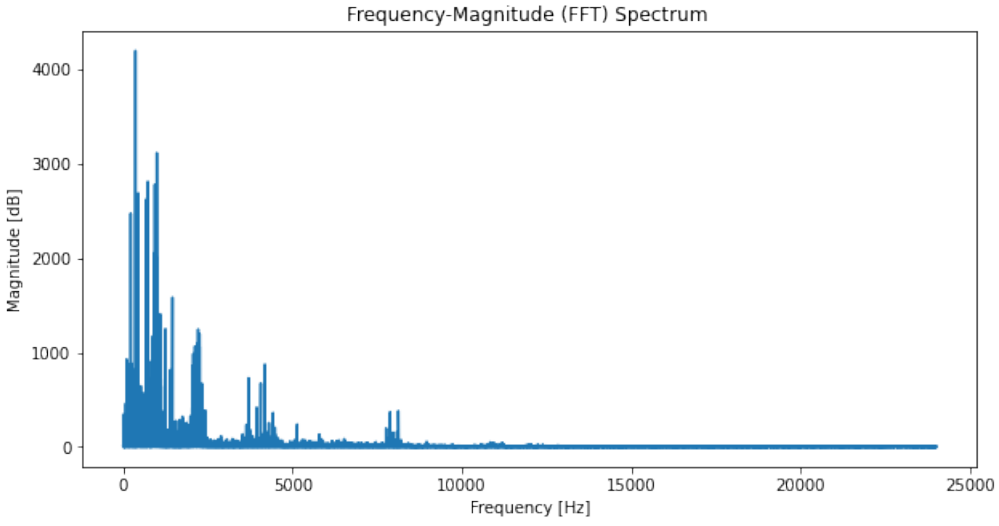


Figure 18: Frequency-Magnitude Spectrum of a signal in the dataset after applying the Fast Fourier Transform (FFT).

4.2.3 Feature Extraction

As mentioned in section 2.1.1, several audio features in both the time-domain and frequency-domain can be examined through signal processing. During the feature extraction step of the data preprocessing phase, some of these were extracted from the signals in the dataset. Recall that the dataset has two main-folders with two different leakage types, *tubeleak* and *ventleak*. These leakage types are kept separate in different dataframes in order to be able to analyze the potential differences in characteristics. The results of using the different leakage types as input to the machine learning models is presented and compared in the Result-section (see section 5). The two dataframes has also been merged together afterwards to form a third dataframe with all signals from the entire dataset.

Simple Audio Features

Some spectral audio features that are easy to extract are: *Spectral Bandwidth, Spectral Rolloff* and *Spectral Flatness*. These were previously defined in equation 4, 5 and 6. As the spectral features are measuring frequency-related characteristics, the signals had to be transformed to the frequency domain by FFT before the features were extracted. Before using the FFT, the signals had to be windowed using the *Hann* window. The audio analysis library *Librosa* contains embedded functions that execute these steps automatically before calculating the features. The following commands retrieve these functions:

- *librosa.feature.spectral_bandwidth(y, sr, n_fft, window)*
- *librosa.feature.spectral_rolloff(y, sr, n_fft, window)*
- *librosa.feature.spectral_flatness(y, sr, n_fft, window)*

The inputs to the functions are the signal array y, the sample rate of the signal sr, the size of the window one wants to run FFT on (n_fft), and the type of windowing function to use ($window$). As the Hann window was used in this project, the latter input was set to $window = "Hann"$. The window size n_fft defines how many samples of the signal the function runs its calculations on at the same time. For this project, the window to run FFT on is the whole signal, so n_fft was equal to the length of the signal. When using these feature-extraction functions, the data analyst can decide if the output should be either a list of feature values per sample of the signal, only one feature value for the whole pack of samples (the whole signal), or something in between. Doing the first alternative will return an array with the same length as the signal's array representation. Saving all these values for each signal and using them as input to a machine learning algorithm is possible. However, it increases the complexity and run-time, especially if the dataset grows larger. Therefore, the entire signal is used as the window for the FFT, resulting in one returned value for each of the feature functions. In figure 19, one can see that the returned feature values from these functions were added to the dataframe for each of the 1920 signals in the *tubeleak*-folder. In addition to these, the microphone configurations were used as a feature, as the sound can be presented differently by being recorded with different settings. The features are represented by the column names **Mic, Bandwidth, Rolloff** and **Flatness**. The same process was done for the *ventleak*-folder. Adding these simple features to one of the dataframes took approximately 2 hours of run-time on a regular computer with an Intel Core i5-7200U CPU processor.

	isLeak	Mic	Bandwidth	Rolloff	Flatness
0	0	1	0.508664	0.495082	0.041548
1	0	1	0.684464	0.570633	0.194759
2	0	2	0.513421	0.495459	0.033190
3	0	2	0.608522	0.525204	0.110118
4	0	3	0.520435	0.496428	0.035376
...	...	...	...	...	...

Figure 19: Pandas Dataframe with the simple features that was extracted for the signals in the *tubeleak*-folder.

Audio Features by Frequency-bins

In addition to the *Librosa*-functions just mentioned, the library contains similar functions for the *Root Mean Square* and the *Spectral Centroid*. These were defined in equation 2 and 3. As mentioned in section 4.2.2, these features were divided into six new features by filtering the signals with different cut-off frequencies. The *Butterworth* filter was applied to the initial signal multiple times with a frequency interval of 4kHz. The newly filtered signals would only contain frequencies inside the defined interval, creating a frequency-bin of size 4kHz. The six new signals would then be sent directly to a function that calculates the RMS-value and also transformed it to the frequency domain. The same signals were also sent to the function that calculates the Centroid. Since the RMS is a time-domain feature, the signals have not been transformed by FFT before being sent to the RMS function. The signals were used as input to the embedded *Librosa*-functions underneath:

- *librosa.feature.rms(y)*
- *librosa.feature.spectral_centroid(y, sr, n_fft, window)*

After the signals with frequencies between 0-4, 4-8, 8-12, 12-16, 16-20, and 20-24 kHz has been sent to the feature extraction functions, 12 new features are returned. As previously mentioned, the frequencies below 4kHz were considered as noise and was filtered away early in the preprocessing steps. However, for only the RMS and Centroid features, the frequencies below 4kHz were included and separated into one bin. The reason for including these frequencies for these features was to be able to analyze if the assumption regarding background noises was appropriate. If there were no difference in feature values of the leakage signals compared to the non-leakage signals, it could be safe to assume that the frequencies below 4kHz were composed of only background noise.

These two features and their associated bins were added to the dataframe with the other audio features under the column names represented by **RMS [interval]** and **Centroid [interval]**, as can be seen from the illustration of the dataframe included in figure 20. The figure only shows the Centroid bin columns, but the same was done for the RMS bins.

Adding these bin features to one of the dataframes took approximately 4 hours of run-time on a regular computer with an Intel Core i5-7200U CPU processor.

	isLeak	Mic	Bandwidth	Rolloff	Flatness	Centroid 0-4kHz	Centroid 4-8kHz	Centroid 8-12kHz	Centroid 12-16kHz	Centroid 16-20kHz	Centroid 20-24kHz
0	0	1	0.078342	0.105466	0.026617	0.567034	0.218660	0.225789	0.268467	0.108733	0.199208
1	0	1	0.408112	0.239315	0.182215	0.653749	0.166904	0.247430	0.523021	0.478968	0.757307
2	0	2	0.087265	0.106133	0.018128	0.593453	0.248196	0.268363	0.281456	0.140578	0.238371
3	0	2	0.265656	0.158831	0.096255	0.617574	0.203525	0.273240	0.384121	0.303932	0.752392
4	0	3	0.100422	0.107850	0.020349	0.542724	0.198065	0.279249	0.259140	0.071398	0.147594
...	...	...	...	...	...	...	...	...	...	...	...
1915	1	2	0.797178	0.716497	0.276873	0.434904	0.290881	0.310613	0.744795	0.716551	0.761191
1916	1	3	0.900463	0.969166	0.476486	0.354952	0.312049	0.767341	0.599795	0.769735	0.979286
1917	1	3	0.963442	0.940346	0.568449	0.365455	0.376918	0.336966	0.813007	0.946215	0.792694
1918	1	4	0.747411	0.650196	0.272780	0.435592	0.311302	0.293273	0.677026	0.638665	0.821555
1919	1	4	0.720462	0.625704	0.300054	0.445608	0.295804	0.278982	0.678423	0.622259	0.797494

Figure 20: Visualization of the part of the Pandas Dataframe containing the frequency-bin features that was extracted for the signals in the *tubeleak*-folder.

Ultrasonic Audio Features

In order to compare the differences between the leakage and non-leakage signals, some features were plotted per frequency bin of 1000Hz from the *Tube*-dataframe. Creating these bins was done by applying the Butterworth filter to the signals, and setting the filter to only let through frequencies between 5000-6000, 6000-7000, etc. Each signal in the dataframe will then get one PSD value per frequency-bin. Thereafter, the mean PSD value was calculated for all leakage signals, and the same for all non-leakage signals. The mean PSD value for each frequency bin is plotted in figure 21. The frequencies below 5000 Hz were ignored, as they are considered background noises. Besides, these bins have much higher PSD values than the rest, which disturbs the scaling of the plot so it is impossible to see the differences in the higher frequencies.

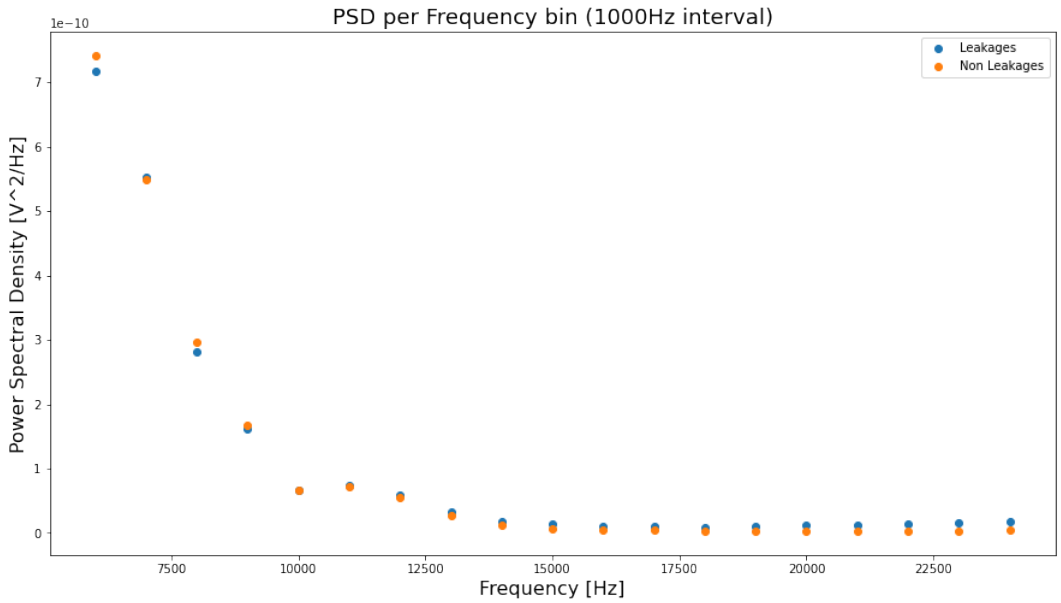


Figure 21: Power Spectral Density plot for the signals in the *tubeleak*-folder spread over frequency bins of 1000Hz intervals.

Zooming in at the plot from figure 21, looking at only the top half of frequencies (see figure 22), it is clear that the difference in PSD from the leakage versus non-leakage signals is increasing as the frequency increases. The frequencies above 20kHz have the largest difference in PSD for the leakage signals, and it could therefore be valuable to examine signals in the ultrasonic frequency range more closely.

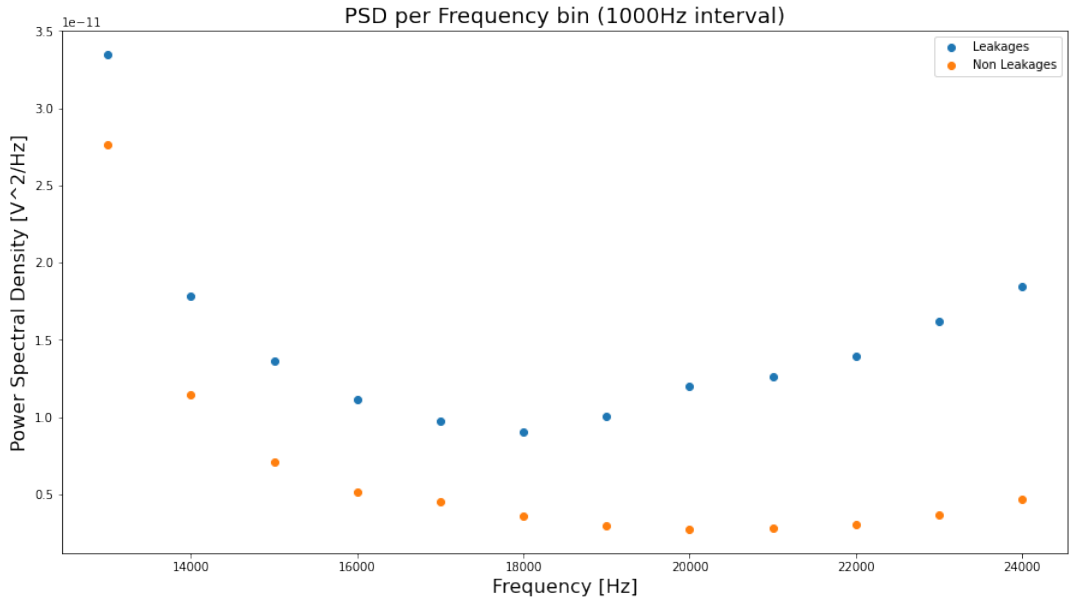


Figure 22: Power Spectral Density plot for the signals in the *tubeleak*-folder spread over frequency bins of 1000Hz interval starting at 12 000Hz.

Ultrasonic frequencies could also be especially interesting to analyze, as humans are unable to hear them. Leakages in the ultrasonic frequency range (20kHz and higher) are more likely to go undetected if dependent on human limitations. Four different features were therefore extracted from the signals in the ultrasonic frequency bin of interval 20-24kHz. One of these features is the *Power Spectral Density* which was defined in equation 7. Another feature is the *Maximum Amplitude*. The two remainder features are the last frequency bin for the *RMS* and *Spectral Centroid*, which contains frequencies between 20-24kHz. These features were added as an experiment to see if they made any positive impact on the machine learning model's performance. In order to detect leakages with frequencies in the 20-24kHz range, the sound needs to be recorded with at sample rate of at least 48kHz.

The two features *PSD* and *Maximum amplitude* were extracted in different ways than the previous features. Firstly, the *PSD* feature was extracted by use of the signal processing package *Signal* in the Python library *Scipy*. The package contains the following function for retrieving the PSD of a signal:

- *scipy.signal.welch(y, window, fs)*

The function above takes in the signal y, the windowing function is set to *'Hann'*, and fs is the sample rate. This function returns a list of frequencies and the associated Power Spectral Density values of those frequencies. Naturally, the signal was windowed and filtered before being used as input for this feature extraction function.

The *Maximum Amplitude* feature was found by simply taking the maximum value of the signal array after windowing the signal and removing frequencies below 20kHz. The well-known package for mathematical operations, *Numpy*, was used for this task, which contains the function:

- *numpy.max(array)*

The function above returns the maximum value in an array. Since the initial signal is represented by a list of amplitude-values, this was a quick and easy feature to extract.

The ultrasonic frequencies were added to the dataframe with the other features, under the column-names **Centroid 20-24kHz, RMS 20-24kHz, PSD >20kHz** and **MAX_AMP >20kHz**, which can be seen from figure 23.

	isLeak	Mic	Bandwidth	Rolloff	Flatness	Centroid 20-24kHz	RMS 20-24kHz	PSD >20kHz	MAX_AMP >20kHz
0	0	1	0.508664	0.495082	0.041548	0.950999	0.001394	0.000005	0.009470
1	0	1	0.684464	0.570633	0.194759	0.985150	0.124609	0.020817	0.317782
2	0	2	0.513421	0.495459	0.033190	0.953396	0.000992	0.000002	0.005738
3	0	2	0.608522	0.525204	0.110118	0.984849	0.033081	0.002667	0.249995
4	0	3	0.520435	0.496428	0.035376	0.947841	0.000948	0.000003	0.007702
...	...	...	...	...	...	...	...	...	...
1915	1	2	0.891876	0.839977	0.287965	0.985387	0.034608	0.001170	0.032210
1916	1	3	0.946937	0.982596	0.484516	0.998733	0.015681	0.000239	0.015081
1917	1	3	0.980511	0.966328	0.575069	0.987315	0.062087	0.003738	0.068396
1918	1	4	0.865345	0.802554	0.283935	0.989081	0.003268	0.000011	0.003032
1919	1	4	0.850978	0.788729	0.310790	0.987609	0.026082	0.000671	0.024128

Figure 23: Pandas Dataframe with the ultrasonic features (the four rows at the end) that was extracted for the signals in the *tubeleak*-folder.

Adding these ultrasonic features to one of the dataframes took approximately 10 minutes of run-time on a regular computer with an Intel Core i5-7200U CPU processor.

4.2.4 Normalization

An overview of all the 18 features that was extracted and stored in dataframes in the previous steps are included in table 2. The last step in the data preprocessing phase is to normalize these features. Each feature measures different aspects of the signal, and their numeric value may therefore vary greatly. For instance, the maximum value for the spectral centroid feature of a signal from the tubeleak-folder was 16 434, while the maximum value for the RMS feature in the highest frequency bin (20-24 kHz) on the same signal was only 0.04. Looking at these values solely numerically, one can see that the spectral centroid is significantly higher. Nonetheless, they measure totally different characteristics and should not be compared numerically. In order to be sure that the machine learning model that takes in these features as input does not favor the higher numerical values as more important, they need to be changed to the same scale. A natural way to scale these features is by normalizing them by compressing the values between 0 and 1. This is done simply by taking the difference between the feature value X and the minimum feature value X_{min}, and diving it by the difference between the maximum X_{max} and the minimum feature value X_{min} [59]. The maximum value for each feature is then 1 and the minimum is 0. The calculation is very straightforward, as can be seen from equation 21.

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{21}$$

Feature Name	**Description**
Mic	Microphone Configuration
Bandwidth	Spectral Bandwidth
Rolloff	Spectral Rolloff
Flatness	Spectral Flatness
Centroid 0-4kHz	Spectral Centroid for a frequency bin of 0-4kHz
Centroid 4-8kHz	Spectral Centroid for a frequency bin of 4-8kHz
Centroid 8-12kHz	Spectral Centroid for a frequency bin of 8-12kHz
Centroid 12-16kHz	Spectral Centroid for a frequency bin of 12-16kHz
Centroid 16-20kHz	Spectral Centroid for a frequency bin of 16-20kHz
Centroid 20-24kHz	Spectral Centroid for a frequency bin of 20-24kHz
RMS 0-4kHz	RMS energy for a frequency bin of 0-4kHz
RMS 4-8kHz	RMS energy for a frequency bin of 4-8kHz
RMS 8-12kHz	RMS energy for a frequency bin of 8-12kHz
RMS 12-16kHz	RMS energy for a frequency bin of 12-16kHz
RMS 16-20kHz	RMS energy for a frequency bin of 16-20kHz
RMS 20-24kHz	RMS energy for a frequency bin of 20-24kHz
PSD >20kHz	Power Spectral Density for frequencies over 20kHz
MAX_AMP >20kHz	Maximum Amplitude for frequencies over 20kHz

Table 2: The initial audio features used in this project.

After the features was normalized, the data preprocessing phase was complete, and the result was two dataframes, *Tube-dataframe* and *Vent-dataframe*, of sizes 18x1920 and 18x1728. Both dataframes contain 18 columns of the same features and one row for each signal.

4.3 Machine Learning Models

Three types of machine learning models were developed for the task of classifying signals into the classes *leakage* and *non-leakage*. All models used the same set of features as input, which are the features that were reviewed in section 4.2.3. After some training and testing, the performance of the models was evaluated. Based on the model evaluation results, some changes were made to the combination of features used as input to the models.

4.3.1 Training, validation and testing sets

A training and validation set is used to try a vast number of preprocessing, architecture, and hyperparameter combinations. These trials result in trained models that are then evaluated on the validation set for quality assurance and to guide the exploration of additional combinations. First, the training set has to be imported from a file. The dataset is generally split in an 80:20 ratio for the training and testing set [34]. The training set used in this project is retrieved from the complete *Tube*-dataframe and *Vent*-dataframe of features. The training set is created with a fraction of 80% and picks rows (or signals) at random. The 20% remaining rows from the original dataframe are then used as the testing set. Then the algorithm automatically splits the training set randomly into a training set of 90% of the original size and a validation set with the remaining 10% of the samples.

Three different dataframes have been used as the original dataset before being split into training, validation, and testing sets. These are: the *Tube*-dataframe on its own, the *Vent*-dataframe on its own, and both frames merged together. The size of each set for the different dataframes can be seen in table 3. The reason for testing the *Tube* and *Vent* dataframes separately is that the signals contains different types of leakages, one being gas leakages from a tube and the other from a vent.

Dataframe	**Size**	**Training**	**Validation**	**Testing**
Tube	1920	1383	153	384
Vent	1728	1244	138	346
Tube + Vent	3648	2627	291	730

Table 3: Overview of the sizes of the original dataframes, and the distribution of samples to the training, validation and testing sets.

4.3.2 K-means clustering model

For the task of developing an *Unsupervised Clustering model*, the *Scikit-learn* (commonly known as *Sklearn*) library was used. The library contains a module for performing clustering on unlabeled data samples called *Cluster*. The module contains different clustering methods, but the *K-means* method was utilized for this project. As mentioned in section 2.2.2, which elaborated on the theory behind the K-means algorithm, the only input the method needs is the set of data samples and the number of classes k the data analyst wants to separate the samples into. For the problem of this thesis, $\mathbf{k=2}$, since there are only two classes in the dataset, namely *Leakage* and *Non-leakage*. The K-means algorithm in the *Sklearn*-library separates the samples in the dataset in the k classes based on equal variance, minimizing a criterion known as the *Sum of Squared Error (SSE)* or the *Inertia*.

The inertia is defined in equation 22, where μ_j is the mean of the samples (also known as the centroid) and N is the total number of samples x [60].

$$Inertia = \sum_{i=0}^{N} \min_{\mu_j \in C} (||x_i - \mu_j||^2) \tag{22}$$

Inertia measures how internally coherent clusters are, but it suffers from some drawbacks. Inertia assumes that clusters are convex, which means it should be possible to draw a line between any two points in the same cluster without leaving the cluster. It also assumes that clusters are isotropic, which means it is identical in all directions. These assumptions essentially mean that the method responds poorly to irregular shaped clusters. Inertia is not normalized either, but the closer to zero the metric gets the better. The K-means algorithm uses Euclidean Distance, but in high-dimensional spaces (using a high number of features), the Euclidean distances between points can become inflated. So there are some challenges in using K-means. However, it can be helpful in finding groups in a dataset that has not been explicitly labeled, and it can also be used as a classification method. As the dataset has been labeled in advance, it is already known which samples (signals) that contain leakages or not, and for this reason it is possible to evaluate the K-means' performance on the leakage detection. By simply removing the "isLeak"-column from the dataset, the algorithm could be fed the unlabeled dataframe of features as input. The method was experimented with using all the features mentioned in section 4.2.3, but also with a combination of just a few of them. The reason for testing with only some of the features as input is that the problem related to inflated Euclidean Distances in high-dimensions is avoided. During the experimentation of the K-means method, different features has been plotted together in order to visualize the relationships and cluster shapes. These plots, as well as the feature combinations and associated performance metrics are included in section 5, which presents the results of all methods.

4.3.3 Deep Recurrent Neural Network models

For the task of developing a *Deep Neural Network model*, the *AutoGluon* library was used. AutoGluon has a package of functions for classifying tabular data, which includes dataframes such as the ones created to store the features. This package, called *Tabular Predictor*, enables the use of automatic testing of many different machine learning models on the same dataset simultaneously. This automatic testing process is called *Automatic Machine learning* or *AutoML* for short. The performance of each model is ranked according to the rest, and the best model is saved for future use. The predictor could train on a wide range of machine learning models, but as the purpose of this project was to investigate the use of Deep Neural Networks and Gradient Boosting methods, these were the only two model types defined for the algorithm to use. One could run both model types simultaneously or separately if the hyperparameter is only set to *Neural Networks* or *Gradient Boosting*. Either way, the results are the same for the individual types.

The neural networks used in the AutoML-run are based on the *FastAI*-library and the *PyTorch*-library, which is recurrent networks. Both libraries are well-known machine learning libraries that offer state-of-the-art classification methods. Since the AutoML-process tests out different models, including different number of layers and learning rates, one does not need to define the hyperparameters in advance.

The neural networks that were developed through AutoML are composed of **4** layers, and uses **ReLU** (see equation 11) as the activation function. The number of epochs (or iterations) the algorithm runs during the training phase is **500**. In one epoch, all the data is used exactly once. The higher number of epochs, the more times the weights are changed in the neural network. A high number can lead to overfitting, and it is, therefore, an important hyperparameter to pay attention to. During the training of the neural network, the **Adam optimizer** is used to optimize the learning rate and update network weights iterative based on the training data [13]. The Adam optimizer is an extension of the stochastic gradient descent method. The regular stochastic gradient descent maintains the same learning rate for all weight updates, while the Adam optimizer computes individual adaptive learning rates for different parameters from estimates of the first and second moments of the gradients [13]. The hidden layer size (or the width) in the network is **128**, meaning each layer is composed of 128 neurons.

The performance of the models is directly linked to the quality of the dataset. Too few samples as input will not be enough for the model to properly learn the relationship between the output and input, resulting in underfitting. There is no well-known number of samples required for a neural network or any type of machine learning method; it depends on the data type and the problem. Basic intuition is applied for this question, as we know; the more observations a human brain makes of the same thing, the more it remembers and learns the characteristics of that specific thing. The same applies to neural networks, as the more samples it gets as input, the higher quality of learning is achieved. Too much of similar samples, however, will result in overfitting.

4.3.4 Gradient Boosting models

The *AutoGluon* library was also used for the task of developing a *Gradient Boosting model*. The two different boosting methods that the AutoML-process tests out is *Catboost* and *XGBoost*. The process that was conducted for the neural networks is also conducted for the gradient boosting models, using the datasets displayed in table 3. The hyperparameters of the two created models consists of **booster:gbtree**, which is a gradient descent decision tree used for boosting, and an objective function based on a logarithmic loss function for binary classification denoted as **binary:logistic** (defined previously in equation 15). The algorithm automatically chooses the hyperparameters since the task is of type binary classification, but they could be changed if the data analyst would like to.

Editing the hyperparameters was unnecessary for this project, as the algorithm automatically picked the appropriate ones. The resulting model was created after **10 000** iterations, which means that 10 000 trees have been boosted, eventually creating the final predictive model. The utilized learning rate was **0.1**.

4.4 Evaluation

The validation set and the testing set is the primary source of evaluation for a machine learning model's performance. There exist several performance metrics that are appropriate for evaluating the performance of machine learning models and their input features. The procedure used to evaluate both the models and the feature combinations is explained in this section, including the utilized performance metrics.

4.4.1 Evaluating models

The two classes in binary classification task are often referred to as *Positive* (P) and *Negative* (N). In the case of the gas leakage problem, the Positive-class consists of audio signals with a leakage present, while the Negative-class consists of audio signals without a leakage. The correctly predicted leakages are referred to as *True Positives* (TP), and the correctly predicted non-leakages are referred to as *True Negatives* (TN). These values are summarized and visualized by a *Confusion Matrix* that is used to calculate the different evaluation metrics. The confusion matrix is a 2x2 table and is visualized in figure 24.

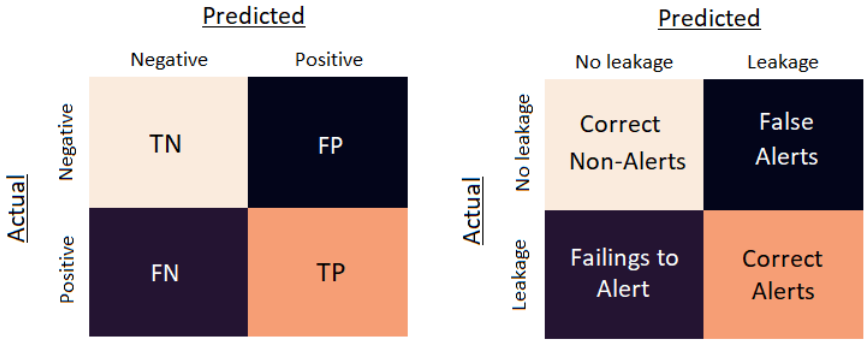


Figure 24: Confusion Matrix for binary classification problems on the left, and the associated confusion matrix for the specific leakage detection problem on the right.

To evaluate a machine learning model's abilities on classification tasks, it is most common to measure the *Accuracy* [29]. Accuracy displays the proportion of samples for which the model produced the correct output (see equation 23). During the training phase of the machine learning model, the learning algorithm uses accuracy to evaluate and modify the model [29]. The model is modified throughout the training phase by updating its weights. The accuracy of the testing set is calculated after the training phase, where the final model will classify the new samples and obtain results for the testing set.

Other metrics, such as *Recall* and *Specificity*, can be used to analyze further the already trained model's behavior related to each class separately [46]. In binary classification tasks, such as the leakage problem, these two metrics provide additional information on how the model performs separately on each of the two classes "leakage" and "non-leakage". Some researchers argue that these metrics are not suitable for selecting the optimal model since they are focused on only one of the classes. However, the importance of the model's performance in only one of the classes can be subjective and dependent on the use case. The system designer could, for instance, be more worried about the risk of undetected leakages (false negatives) than getting false alerts (false positives). Recall displays the proportion of positive samples that produced the correct output, or in other words, the proportion of the actual leakages the model classified as a leakage. The same idea applies to the Negative class for the specificity metric, which measures the proportion of actual non-leakages the model predicted to be non-leakages. Equation 24 displays the recall, while equation 25 displays the specificity [46].

$$Accuracy = \frac{TP + TN}{P + N} \tag{23}$$

$$Recall = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{24}$$

$$Specificity = \frac{TN}{N} = \frac{TN}{TN + FP} \tag{25}$$

Precision is a metric that indicates the classification model's ability to identify the true positives [46]. It displays the proportion of the total predicted positive samples that were actually positive, and is defined by equation 26. In this case, precision shows how many of the predicted leakages were actual leakages. It is looked at as the quality of a positive prediction made by the model.

$$Precision = \frac{TP}{TP + FP} \tag{26}$$

Finally, *F1-Score* is a metric that represents the harmonic mean between recall and precision, as displayed by equation 27.

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{27}$$

One drawback of using the F1-score is that precision and recall are given equal importance. By the assumption that the classification model alone results in alerts or not, the consequences of failing to alert a leakage due to false negatives are greater than the risk of unnecessary checks due to false positives. The recall is, therefore, a more significant metric for this problem. However, both recall and precision in collaboration with specificity and accuracy are included in order to compare the model's results for the different metrics.

For the K-means method it is not common to evaluate the model based on the metrics mentioned above. Instead, the statistical metric *Inertia* is often used. The K-means algorithm already tries to minimize the inertia during the clustering, but it could also be used as a metric to evaluate the model afterwards as it converges on a final value. Since this metric calculates the sum of the squared distance between the centroid and each member of the cluster (see equation 22), it indicates how similar all the members in the created clusters are. A large number for the inertia would indicate that there are large differences or dissimilarities inside the cluster, thereby that they might not be a homogeneous group and does not fit well together.

As mentioned previously, K-means is an unsupervised technique, meaning it does not train on the output labels such as the supervised methods. However, since the optimal clusters (or class labels) are known to the data analyst in advance, one can compare the classes made by K-means algorithm to the actual classes of leakages versus non-leakages. To do this, the accuracy and the other mentioned metrics could be calculated based on one of the clusters being defined as the leakage-class. The leakage class was defined as the cluster with the most actual leakages in it. The true positive samples would then be the actual leakage samples within the defined leakage cluster, and the false positives would be the actual leakage samples within the non-leakage cluster. The false positives and false negatives are

then easily found by the rest of the members in the clusters, and then the evaluation metrics could be calculated. The *Accuracy* will be mostly focused on for evaluating K-means as a classifier.

4.4.2 Evaluating Features

After the initial features were used as input to the machine learning models, the performance of the models gave some pointers as to which features were most important and which were irrelevant. Some of the features did not offer that much "new" information on the characteristics of the signal. Correlations between features can be calculated to find out which of the features are redundant. The *Pearson Correlation Coefficient* is the most widely used correlation measure [23], and it is defined by equation 28.

$$p_{x,y} = \frac{\sum(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum(X_i - \overline{X})^2(Y_i - \overline{Y})^2}} \tag{28}$$

The coefficient $p_{x,y}$ measures the linear association between variables x and y, which means it quantifies to which degree the relationship between two features can be described by a line. The mean of x is denoted by $\overline{X} = \frac{1}{n}\sum_{i=1}^{N} x_i$, and the mean of y is denoted by $\overline{Y} = \frac{1}{n}\sum_{i=1}^{N} y_i$. The coefficient $p_{x,y}$ ranges from -1 to 1, and can be easily found by the embedded correlation function inside the *Pandas*-library. The *Seaborn*-library contains a function for visualizing the correlations of variables or features in a dataframe. These two functions are:

- *correlations = Pandas.dataframe.corr()*
- *correlation_map = Seaborn.heatmap(correlations)*

These functions were used to create the correlation map of the utilized features, which is displayed in figure 25. As there are all of 18 features present in the map, the matrix contains many boxes and could seem overwhelming to analyze, but the color bar helps by indicating the correlation. The closer the boxes seem to a white or black color, the more the features correlate. By examining the values in the map, one can see that the RMS bin features correlated with a coefficient of 0.93-0.99. As this indicate a high similarity, they were removed from the set of features and replaced with one total RMS feature for the whole signal. Similar adjustments was done for other features, and will be explained in detail in section 5. The machine learning models were then trained and tested again with the new combination of features to examine the effect on the models' performance.

The *AutoGluon* library contains a function for examining the importance of each feature in the input dataset on the model's performance. The function returns a table of the features in a dataframe with associated values for the following measures: importance score, the standard deviation of the importance score, p-value for a statistical t-test of the null hypothesis, number of shuffles performed to estimate the importance score, the upper end of the 99% confidence interval for true feature importance score and the lower end of the 99% confidence interval for true feature importance score. These measures are respectively denoted as *importance, stddev, p_value, n, p99_high* and *p99_low*, as can be seen from the table in figure 26. These importance tables have different values depending on the model used as the predictor, and they can be created for all models in the AutoGluon library, but not the K-means model from the Sklearn library.

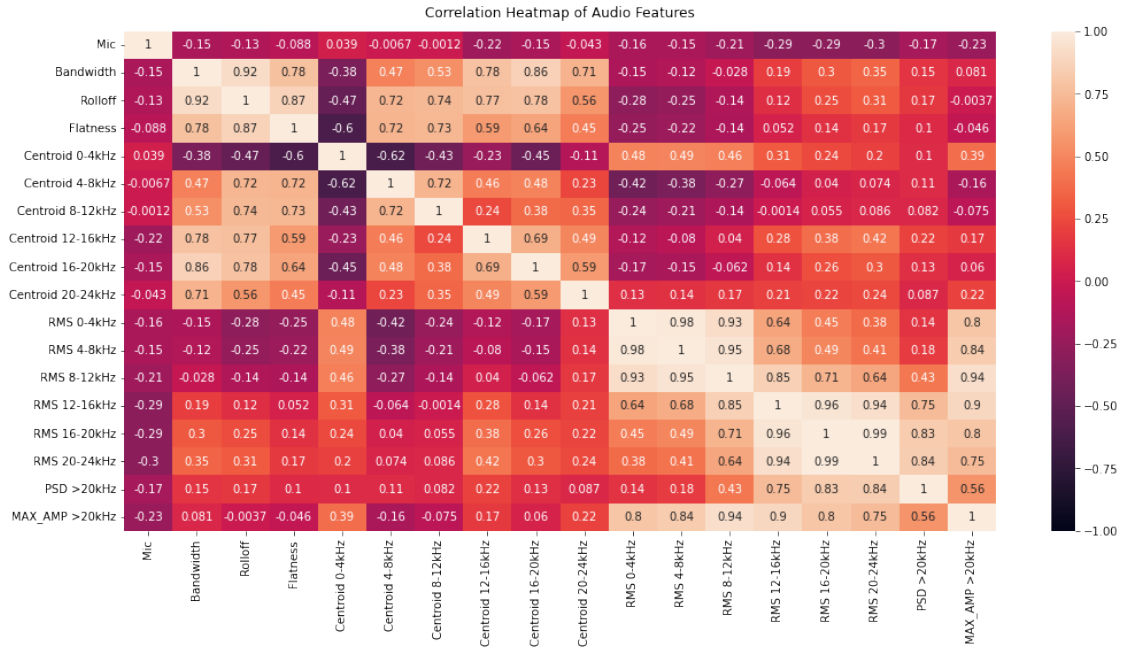


Figure 25: Audio feature correlation map for the features extracted from the signals in the *tubeleak*-folder.

	importance	stddev	p_value	n	p99_high	p99_low
RMS 20-24kHz	0.284667	0.012342	0.000313	3	0.355390	0.213944
Centroid 20-24kHz	0.114333	0.008145	0.000844	3	0.161002	0.067664
RMS 0-4kHz	0.113667	0.009713	0.001212	3	0.169321	0.058013
Centroid 0-4kHz	0.104667	0.008145	0.001006	3	0.151336	0.057998
RMS 8-12kHz	0.057000	0.008718	0.003854	3	0.106954	0.007046
PSD >20kHz	0.054667	0.005033	0.001407	3	0.083508	0.025826
Flatness	0.051667	0.004933	0.001512	3	0.079933	0.023401
RMS 12-16kHz	0.035667	0.010017	0.012649	3	0.093063	-0.021730
Centroid 16-20kHz	0.035000	0.005196	0.003633	3	0.064775	0.005225
Centroid 12-16kHz	0.034333	0.003786	0.002014	3	0.056027	0.012639
RMS 4-8kHz	0.033667	0.008083	0.009339	3	0.079983	-0.012649
MAX_AMP >20kHz	0.033333	0.005033	0.003757	3	0.062174	0.004492
Centroid 8-12kHz	0.026000	0.001732	0.000738	3	0.035925	0.016075
Mic	0.020333	0.001528	0.000938	3	0.029086	0.011580
Rolloff	0.016333	0.000577	0.000208	3	0.019642	0.013025
RMS 16-20kHz	0.014333	0.001528	0.001882	3	0.023086	0.005580
Centroid 4-8kHz	0.014000	0.001732	0.002532	3	0.023925	0.004075
Bandwidth	0.013667	0.005508	0.025051	3	0.045226	-0.017892

Figure 26: Table that displays the importance of each feature. Created by running AutoGluon on the *tubeleak*-features with the Pytorch Neural Network model.

The most important measures to pay attention to in the feature importance table are the importance score and the p-value. The importance score indicates how important the feature is for the good performance of the model. If the score is negative, it indicates that the features are destructive for the final model. The lower the score, the more reason there is to remove a feature from the input dataset.

The p-value is also important to inspect as it indicates the statistical probability of the importance score for the feature being high by random chance. A low p-value can offer assurance regarding the feature choice in relation to the importance score of the feature. It is common to define a threshold for the p-value, which there are different opinions about in the statistical science community, but it is often set to 0.05 or 0.01 [20]. Any feature with a p-value above the threshold of 0.01 would, in that case, be regarded as having an important influence on the model solely due to chance.

4.4.3 Evaluating Generalization

One measure that was taken to avoid overfitting was to pick the files that went into the different datasets at random. By randomly picking 80% instead of using the first 80% samples as the training set, the chance of using very similar signals (for instance, from the same recording sessions) to train on is reduced. An alternative method could have been to deliberately pick out which files should go into the training, validation, and testing sets. However, this is a time-consuming task, and it also entails that the data analyst knows exactly which files to pick.

Based on the feature evaluation results, some measures could be taken to avoid overfitting and high complexity. The less important features that also correlate highly with others could be removed from the dataset that is used as input to the models. Features that do not offer any new and relevant information about a signal should not be included. Therefore, after evaluating the original features, some of them were removed or merged into one, creating a new set of features ready to be used as input to the machine learning models. The models that were trained with the new features were then compared to the performance of the old model and the old features. This will be further looked into in section 5, where results are presented.

Finally, the training and testing error could be examined to watch out for overfitting. The training accuracy is the opposite of training error, where accuracy is the number of correct predictions and error is the number of incorrect predictions. A much higher training accuracy than testing accuracy indicates overfitting, implying that the model does not generalize properly. Therefore, a comparison of the training and testing accuracy has been made for all models during all runs.

5 Results

This section of the thesis presents the results obtained during the experimentation of the leakage dataset. Characteristics of the leakage signals are analyzed and presented both visually and numerically. The developed machine learning models produced output that was analyzed by the performance metrics defined in the previous section, and the values of each model are displayed and compared.

5.1 Feature Analysis

Different audio features were analyzed during the experimentation with the signals from the leakage dataset. The normalized features were plotted in order to obtain a visual perception of leakage characteristics. The feature value of each leakage signal in the *Tube* dataframe was plotted in a scatter plot next to the non-leakage signals to compare them visually. As the features were normalized for all signals, they have values between 0 and 1 and can be compared relative to each other. The signals recorded at the laboratory are not included in the plots as they had very different characteristics than the signals from the other recordings sessions. For this reason, the y-axis on the plots will not always have one as the maximum value.

The phenomenon of the laboratory recordings is visualized by the scatter plot in figure 27, where the signals from the laboratory are seen in the middle of the x-axis from file number 1344 to 1536. On the mentioned plot, the flatness values for the normal non-leakage signals are much higher for the laboratory recordings than for the other recording sessions.

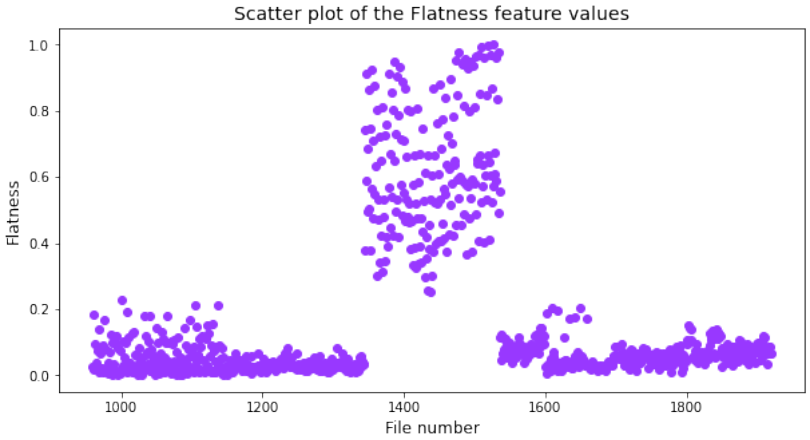


Figure 27: Plot of the flatness value for each of the signals in the *Tube* dataframe.

The laboratory signals disturb the perception of the differences between non-leakage and leakage signals, making it harder to analyze the characteristics. These recordings are considered a "special case", as the characteristics do not match the signals taken in the other environments. Additionally, the laboratory environment is not representative of where the hypothetical leakage detection system shall have its intended usage. Nevertheless, they were included in the input for the machine learning models in order to have more samples to train with. Alternative approaches will be discussed in section 6.

The simple features, *Bandwidth, Rolloff* and *Flatness*, are represented for each of the signals in figures 28, 29 and 30. From the figures, one can see that the leakage signals have on average larger values for all the simple features relative to the non-leakage signals. The average values of each feature for the leakage versus non-leakage signals are presented in table 4 and are discussed at the end of this feature analysis section.

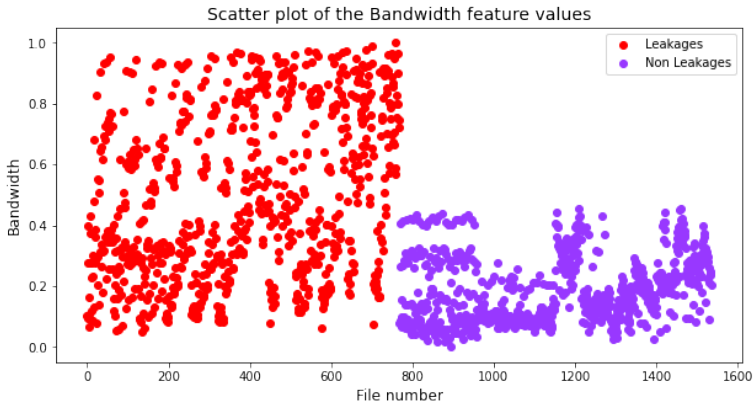


Figure 28: Plot of the bandwidth value for each of the signals in the *Tube* dataframe (except for the signals from the lab-recording).

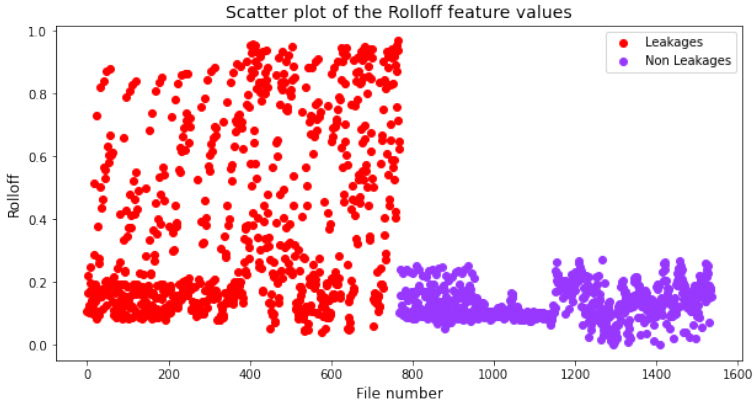


Figure 29: Plot of the rolloff value for each of the signals in the *Tube* dataframe (except for the signals from the lab-recording).

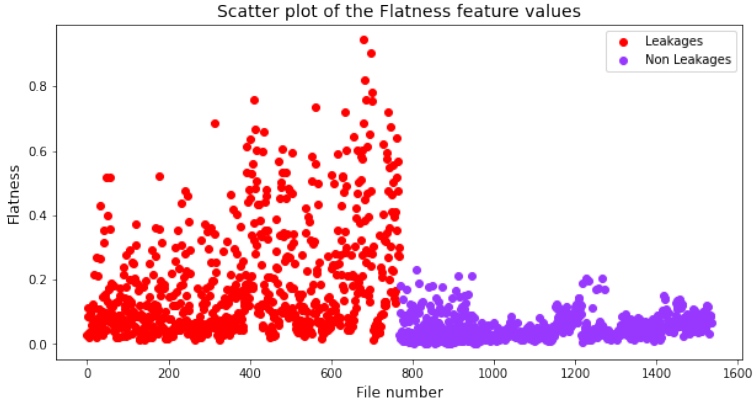


Figure 30: Plot of the flatness value for each of the signals in the *Tube* dataframe (except for the signals from the lab-recording).

The bin features, *RMS* and *Spectral Centroid*, are presented with the lowest and the highest frequency bin, specifically 0-4kHz and 20-24kHz. Figure 31 and 32 displays the Spectral Centroid values of each signal. The first red "grouping" of values in figure 31 are from the recording sessions with *hydraulic machine noise* and the other red "group" represents signals with the *general factory workshop noise*. The same applies to the purple "groups". As is clearly depicted in the plot, the spectral centroid values are very different depending on the background noise (recording session), and the leakage signals are similar to the non-leakage signals for this audio feature. In section 4 it was explained that the frequencies below 4000Hz were assumed to be solely background noise. The plot in figure 31 indicates that it was a fair assumption as the values are very similar regardless if there is a leakage. Adding to that, the average value for the spectral centroid between the frequencies 0-4kHz was **0.5298** for the leakage signals and **0.5297** for the non-leakage signals, which is almost exactly the same.

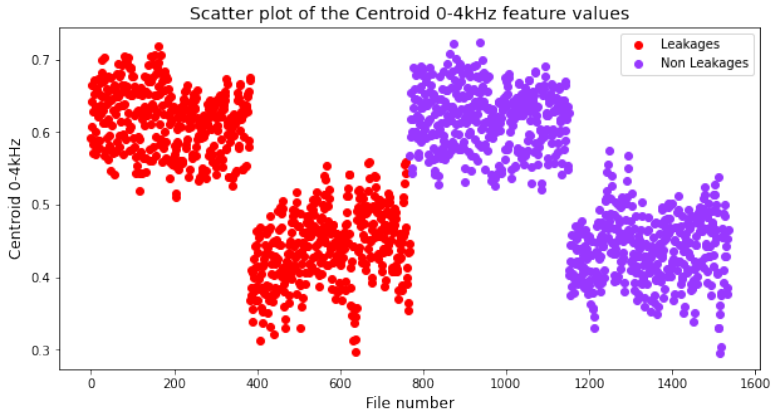


Figure 31: Plot of the spectral centroid value between frequencies of 0-4kHz for each of the signals in the *Tube* dataframe (except for the signals from the lab-recording).

In contrast, the spectral centroid between the frequencies 20-24kHz have more heterogeneity for the leakage and non-leakage signals. Figure 32 shows that most of the non-leakage signals have a value of ≤ 0.65, while the value for most of the leakage signals is ≥ 0.75. Naturally, there are some outliers for both signal types. The average spectral centroid value between 20-24kHz for the leakage signals is **0.7257**, while for the non-leakage signals, the average is **0.4777**.

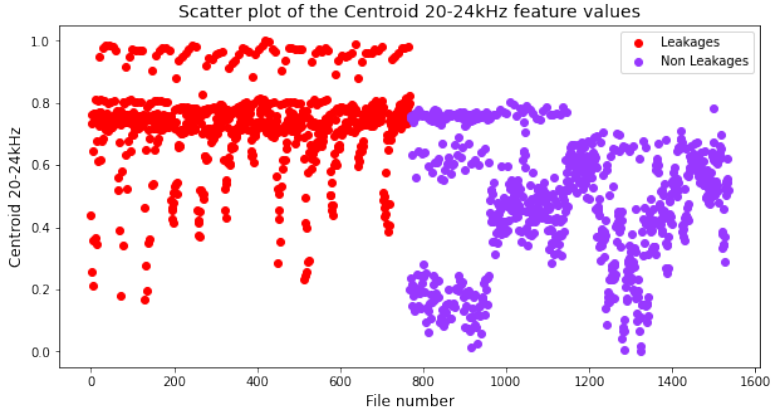


Figure 32: Plot of the spectral centroid value between frequencies of 20-24kHz for each of the signals in the *Tube* dataframe (except for the signals from the lab-recording).

The RMS values for each signal for the lowest and highest frequency bin is displayed in figure 33 and 34. In the lowest frequency bin, containing frequencies between 0-4kHz, the RMS value is highly dependent on the environment (recording session), similarly to the spectral centroid. There are four recording sessions when disregarding the laboratory session. In the plot in figure 33 it is relatively easy to identify the signals from the different recording sessions. The first 172 red values on the x-axis are the leakage signals from the first session; the next 173-345 values are from the next session, and so on. The purple values for the same sessions have approximately the same pattern. The average RMS value for the lowest bin is **0.1907** for the leakage signals and **0.1842** for the non-leakage signals. The averages indicate high similarities for the RMS of the leakage versus non-leakage signals under 4kHz. Although the leakage signals have a slightly higher RMS value, the difference is only 0.0065.

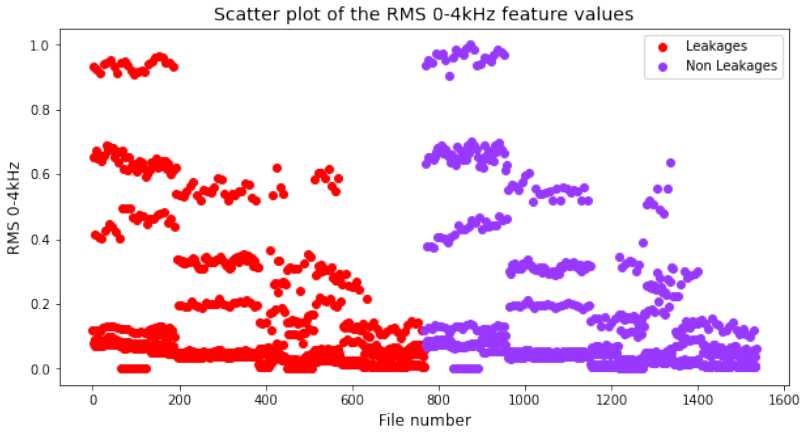


Figure 33: Plot of the RMS value between frequencies of 0-4kHz for each of the signals in the *Tube* dataframe (except for the signals from the lab-recording).

The RMS values for the highest or ultrasonic frequency bin are displayed on the plot in figure 34. The plot shows a more spread in RMS values for the leakage signals than for the non-leakage signals. The non-leakage signals are mostly concentrated at around zero, except the signals from the first recording session, which have some values up to around 0.15. The average RMS value for the leakage signals is **0.0446**, while it is **0.0100** for the non-leakage signals.

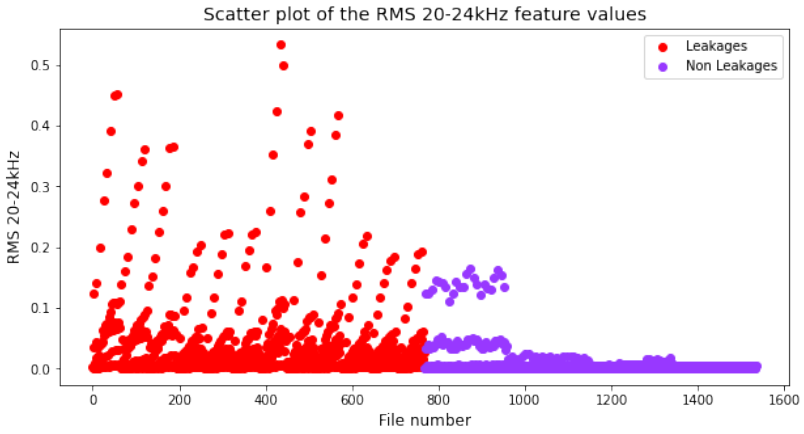


Figure 34: Plot of the RMS value between frequencies of 20-24kHz for each of the signals in the *Tube* dataframe (except for the signals from the lab-recording).

For the ultrasonic features *Maximum Amplitude* and *Power Spectral Density* the values for each signal is visualized in figure 35 and 36. The maximum amplitude values for the leakage signals are somewhat similar to the non-leakage signals but are slightly higher on average. The average value for the leakage signals is **0.0815**, while it is **0.0593** for the non-leakage signals. Notice that a lot more of the non-leakage signals is concentrated around zero compared to the leakage signals, which have some around zero and many samples slightly above.

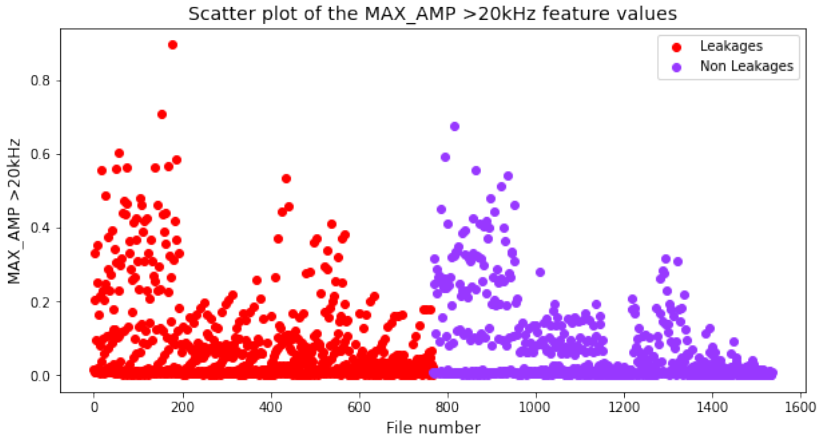


Figure 35: Plot of the maximum amplitude value for each of the signals in the *Tube* dataframe (except for the signals from the lab-recording).

Most signals have low values for the power spectral density feature, but for some of the leakage signals, the values are higher. There are only a few of the signals that have a higher PSD value, which does not elevate the average of the leakage signals to more than **0.0075**, while the average for the non-leakage signals is **0.0010**.

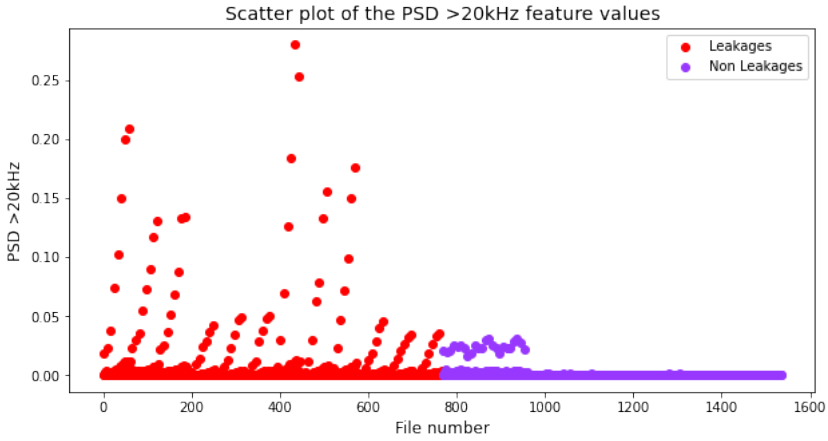


Figure 36: Plot of the power spectral density value for each of the signals in the *Tube* dataframe (except for the signals from the lab-recording).

For the PSD feature, in particular, the laboratory recordings had a high impact on the difference between the leakage and non-leakage signals. By including the laboratory recordings in the analysis, the average PSD value of the leakage signals was raised from **0.0075** to **0.0116**, while for the non-leakage signals, it was slightly reduced from **0.0010** to **0.0008**.

An overview of all the average feature values of the leakage and non-leakage signals is presented in table 4, including the difference between the averages. Overall, the values are generally higher for all features when a leakage is present in a signal.

Feature	**Average Leak**	**Average Non-leak**	**Difference**
Bandwidth	0.4970	0.1840	+0.3130
Rolloff	0.3941	0.1315	+0.2626
Flatness	0.1840	0.0532	+0.1308
Centroid 0-4kHz	0.5298	0.5297	+0.0010
Centroid 20-24kHz	0.7257	0.4777	+0.2480
RMS 0-4kHz	0.1907	0.1842	+0.0065
RMS 20-24kHz	0.0446	0.0100	+0.0346
Max Amplitude	0.0815	0.0593	+0.0222
PSD	0.0075	0.0010	+0.0065

Table 4: Average feature values for the leakage signals versus the non-leakage signals (when not considering the laboratory files).

In order to compare the effect of the laboratory environment on the audio features, table 5 is enclosed. For many of the average feature values, the difference between the leakage and non-leakage signals decreases when the laboratory recordings are considered. However, the differences increase when considering the laboratory recordings for the Centroid 0-4kHz, RMS 20-24kHz, Maximum Amplitude, and PSD features.

Feature	**Average Leak**	**Average Non-leak**	**Difference**
Bandwidth	0.5335	0.2694	+0.2641
Rolloff	0.4992	0.2566	+0.2426
Flatness	0.2401	0.1660	+0.0741
Centroid 0-4kHz	0.5129	0.4694	+0.0435
Centroid 20-24kHz	0.7367	0.5191	+0.2176
RMS 0-4kHz	0.1542	0.1483	+0.0059
RMS 20-24kHz	0.0515	0.0084	+0.0431
Max Amplitude	0.0807	0.0478	+0.0329
PSD	0.0116	0.0008	+0.0108

Table 5: Average feature values for the leakage signals versus the non-leakage signals (when including the laboratory files).

A discussion could be made on whether or not to include the laboratory recordings as input to the machine learning models. Although the laboratory environment does not portray a relevant environment for the proposed system, it could help the machine learning models understand the leakage characteristics without any disturbing factors from background noise. More on the use of the laboratory files and other points of worry will be discussed in section 6.

5.2 Unsupervised Clustering model

The K-means clustering model was developed as an attempt to both learn more about the patterns and characteristics of leakage signals and also as an attempt to classify the signals. The *SSE* and the model evaluation metrics defined in section 4.4.1, were used as the primary basis to evaluate the K-means model for the respective purposes.

5.2.1 Model Performance

The K-means model was tested out on the leakage dataset during three runs with different dataframes of audio features. The dataframes and their content were presented in table 3, all having a balanced distribution of leakage and non-leakage signals. Table 6 displays the SSE and the accuracy metric for the three different dataframes (runs). The other metrics are not presented in the table as they had almost exactly the same value as the accuracy, ranging from 50.01% to 50.06%.

Run	Dataframe	Cluster sizes	SSE	$\overline{SSE}$	Accuracy
1	*Tube*	960	1610.59	0.840	50.05%
2	*Vent*	864	1683.41	0.974	50.06%
3	*Tube + Vent*	1824	3348.38	0.918	50.03%

Table 6: Table of the sizes, sum of squared error (SSE), average SSE, and accuracy for the clusters created from the different dataframes containing the initial audio features.

As mentioned in section 4.3.2, a drawback of clustering models is that the Euclidean distance gets inflated when there are many dimensions (many features). For this reason, the model was tested with combinations of three audio features from the *Tube + Vent* dataframe to see if the performance improved. The combinations are based on different features types, specifically the *Simple, Bin* and *Ultrasonic* types, and the *Top 3* and *Bottom 3* features by importance score from the table in figure 26. Note that table 26 ranked the features according to the importance for only the Neural Network model, and they are not necessarily important for the K-means model. Nevertheless, they can serve as an indicator for overall important features. An overview of the features used for each type is included in table 7.

Comb.	Type	Feature 1	Feature 2	Feature 3
1	Simple	Flatness	Rolloff	Bandwidth
2	RMS Bin	RMS 0-4kHz	RMS 12-16kHz	RMS 20-24kHz
3	Cent Bin	Centroid 0-4kHz	Centroid 12-16kHz	Centroid 20-24kHz
4	Ultrasonic	PSD >20kHz	Centroid 20-24kHz	MAX_AMP >20kHz
5	Bottom	Bandwidth	Centroid 4-8kHz	RMS 16-20kHz
6	Top	RMS 0-4kHz	Centroid 20-24kHz	RMS 20-24kHz

Table 7: The combinations of features used for the K-means model.

As the dataframes used as input to the K-means model contain 1824 signals with leakages and 1824 without leakages, two optimal clusters should be of size 1824, separating these two signal types completely. The results from table 6 express that the clusters created by the K-means model, with all the initial features as input, have managed to create optimal cluster sizes. Nonetheless, the accuracy shows that around half of the samples in each cluster created by K-means are leakage samples. The accuracy metric indicates that the model does not manage to separate the leakage signals from the non-leakage signals.

After changing the input to the K-means model to only three features, the sizes of the clusters were changed. Recall that K-means does not define which cluster is the leakage cluster; instead, the program was instructed to label the cluster with the most leakage samples as the leakage cluster after the clusters were created. The size of the leakage cluster for each feature combination can be seen in table 8, along with the performance of the K-means model represented by the evaluation metrics.

Comb.	**Leak Size**	**Accuracy**	**Recall**	**Specificity**	**Precision**	**F1-Score**
1	1522	72.59%	64.31%	80.87%	77.07%	70.11%
2	3456	44.74%	89.47%	0.00%	47.22%	61.82%
3	1643	84.13%	79.17%	89.09%	87.89%	83.30%
4	2247	81.94%	93.53%	70.34%	75.92%	83.81%
5	2740	43.97%	69.08%	18.86%	45.99%	55.21%
6	2235	82.10%	93.37%	70.83%	76.20%	83.91%

Table 8: Performance of the K-means model on the *Tube + Vent* dataset, represented by different evaluation metrics, including the size of the leakage cluster.

By examining table 8, one can see that some clusters were a great deal larger than the optimal size of 1824. For instance, for the RMS bin features in combination 2, the model produced a leakage cluster of size 3456, which left the non-leakage cluster with merely any samples. The non-leak cluster had only 192 samples, and all of them were actually leakages, resulting in a specificity score of 0%. Both combinations 2 and 5 produced worse results than the K-means model achieved after using the whole set of 18 features. Recall that the accuracy for the whole set of features was 50.03% for the *Tube+Vent* dataset, while combinations 2 and 5 produced only 44.74% and 43.97% respectively. The other combinations of features produced slightly better classification results, but none of them had over 84% for the F1-score or over 85% for the accuracy. Nevertheless, reducing the number of features/dimensions from 18 to 3 seems to have improved the performance of K-means a great deal. Given a proper choice of feature combinations, the model seems to be able to separate the signal types better than using all of the features. However, the results are still not satisfactory.

5.2.2 Cluster Visualization

The clusters that the K-means model created can be visualized by a plot in n dimensions, n depending on the number of features. For each of the combinations presented in table 7, the clusters that the K-means model defined are presented by 3D-plots.

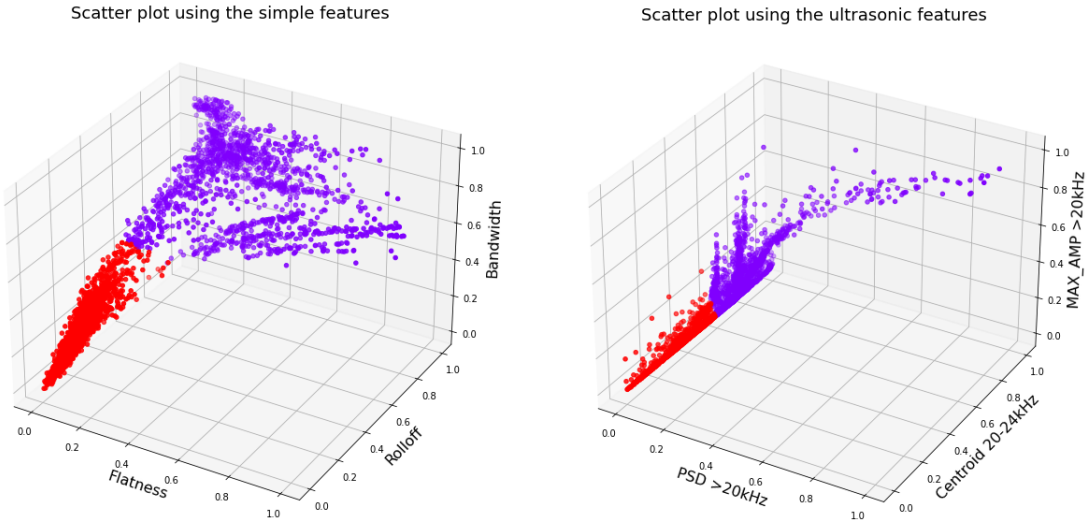


Figure 37: Visualizations of the clusters created by the K-means model on the *Tube+Vent* dataset for the simple features and the ultrasonic features. The defined clusters are represented by the colors.

The 3D-plots for the simple and ultrasonic feature combinations is included in figure 37, which includes a point for each of the samples in the dataset. Keep in mind that the colors are not representing a leakage or non-leakage class but rather the K-means model separation of clusters. For the RMS and centroid bin combinations, the 3D-plot of the cluster distribution is included in figure 38. Lastly, the cluster distribution for the bottom and top 3 feature combinations are plotted in figure 39.

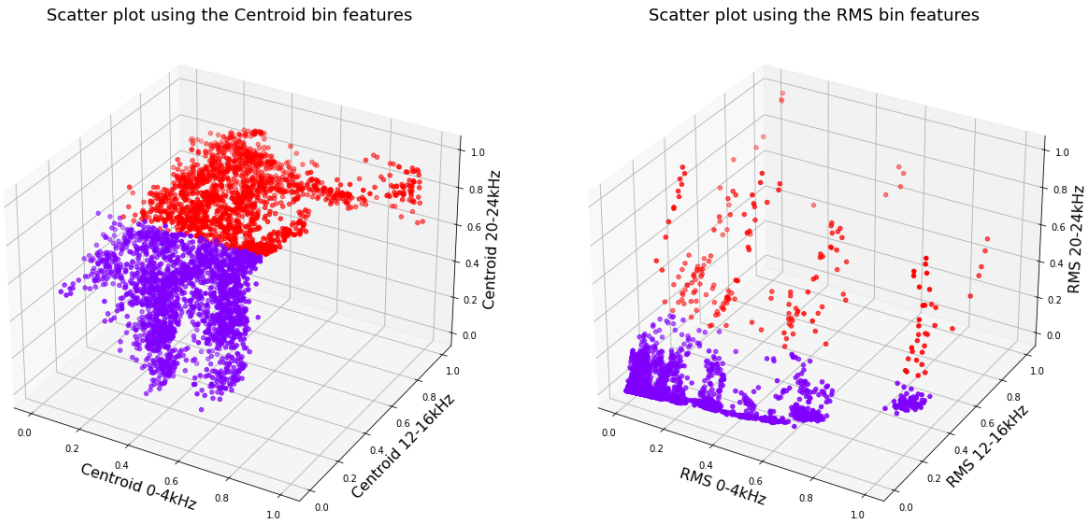


Figure 38: Visualizations of the clusters created by the K-means model on the *Tube+Vent* dataset for the bin features. The defined clusters are represented by the colors.

For the RMS clusters in figure 38, a few samples have very high values for the 20-24kHz bin relative to the rest. These are assigned to one cluster, and the majority are assigned to the other. Numerically, this assignment makes sense for the K-means algorithm, which bases its assignments on the SSE. Nevertheless, as we know, there should be an equal amount of samples in each cluster as there is an equal amount of leakage and non-leakage signals in the dataset.

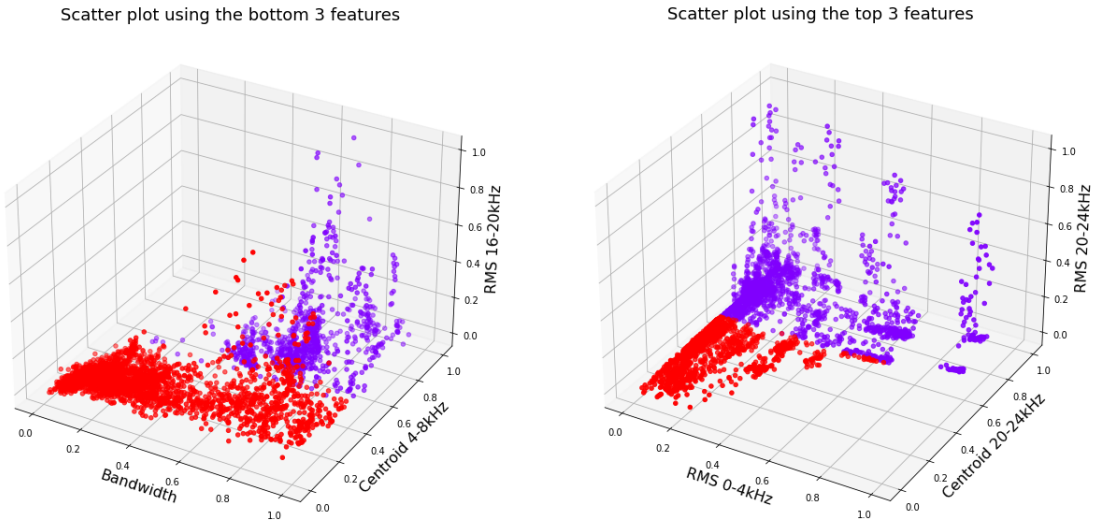


Figure 39: Visualizations of the clusters created by the K-means model on the *Tube+Vent* dataset for the bottom three and top three important features. The defined clusters are represented by the colors.

As can be seen from all the three figures, and by disregarding the colors, the samples do not seem to form two distinct groups. For most of the combinations, all the samples seem to either form one big group or are widely spread across the dimensions. Either way, the K-means algorithm tries to approximate two groups by calculating the least SSE based on the linear values that the samples represent. The shapes of the samples are different depending on the features, but none of the cluster shapes are very clearly separated from each other. Many of the clusters, for instance, the simple and ultrasonic clusters in figure 37, look like they have been formed by simply separating the whole group of samples at a certain point. At the spot where the samples go from being red to purple, there are very small numerical differences, meaning the samples in the different clusters are actually very similar. Since there are no distinct convex and isotropic groups, the K-means algorithm performs as well as it can be based on the data it is given.

5.2.3 Visualizations of the Optimal Clusters

To compare the clusters defined by K-means to the actual class labels that represent the optimal clusters, the following 3D-plots have been created. The red clusters contain the samples which are true members of the leakage class. As can be seen from all the following plots, the red clusters generally contain samples "higher" on the axes in the 3D-plot.

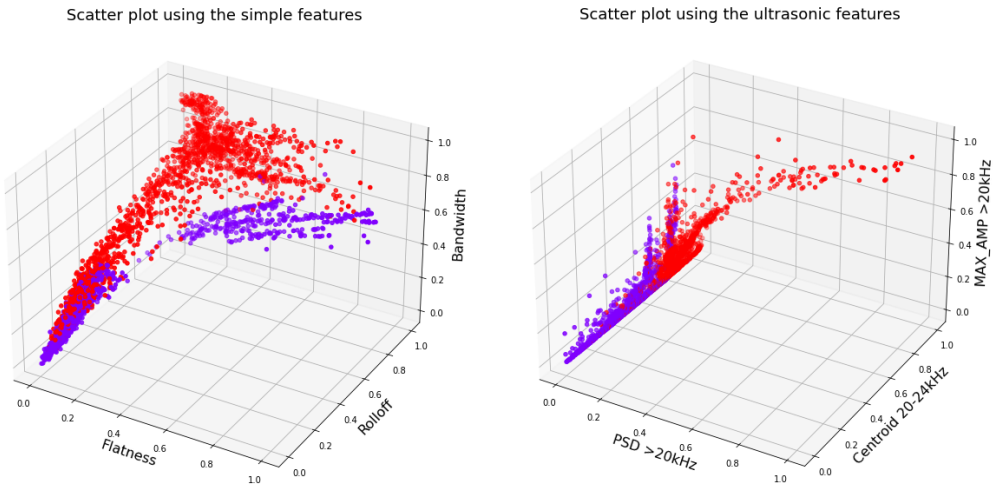


Figure 40: Visualizations of the optimal clusters on the *Tube+Vent* dataset for the simple and ultrasonic features. The true classes are represented by the colors.

The optimal clusters for the simple features are very different from the predicted clusters of the K-means algorithm, as can be seen when comparing figure 37 with 40. It is important to keep in mind that the red color in the predicted clusters does not necessarily represent the leakage class, such as for the optimal cluster plots. The color in the predicted cluster plots represents a cluster, while the ones with the most leakage samples have been labeled as the leakage class later on when evaluating the K-means model. Hence, by disregarding the colors and focusing on the separation of samples, the predicted ultrasonic clusters do come relatively close to the optimal. This is also substantiated by the evaluation metrics scores for the ultrasonic features, which include accuracy of 81.94%.

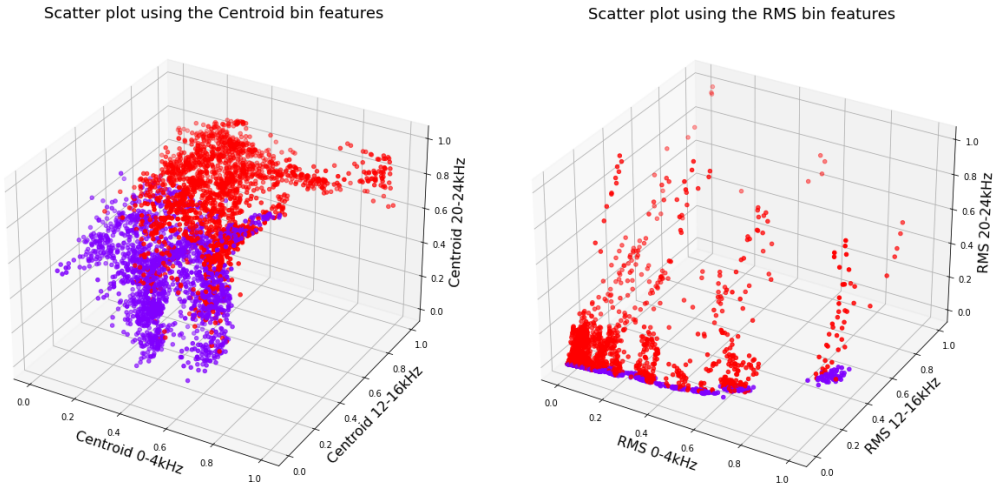


Figure 41: Visualizations of the optimal clusters on the *Tube+Vent* dataset for the bin features. The true classes are represented by the colors.

For the bin features, there is a large contrast in the performance of the Centroid bin clusters and the RMS bin clusters. The predicted Centroid clusters look like they were separated by a straight line close to the middle of the whole group of samples, with 1624 in one cluster and 2005 in the other. On the other hand, the predicted RMS clusters have 3456 samples in one cluster and only 192 in the other. From the 3D-plots in figure 41, one can see the true class separations, otherwise known as the optimal clusters. By comparing the optimal clusters to the predicted clusters in figure 38 one can see some deviations. For the Centroid features, the separation of clusters is not that "straightly cut off" in the middle for optimal clusters as the predicted clusters were. Naturally, the algorithm would have had a higher SSE if it included the red samples that are seen in the middle of the purple optimal cluster as they are farther away from the rest of the red cluster in Euclidean distance. So the algorithm worked as well as could be expected for this feature, based on the fact that the algorithm uses a numerical distance measure as the foundation for classification.

For the predicted RMS bin clusters, the performance is worse by a great deal. The optimal clusters in figure 41 show that the non-leakage cluster shall contain all the 1824 lowest values close to 0 seen by the z-axis. However, the predicted non-leakage cluster contains the 3256 lowest values seen by the z-axis. Likewise, for this feature combination, the poorly predicted separation can be explained by the minimization objective of the SSE by the K-means algorithm.

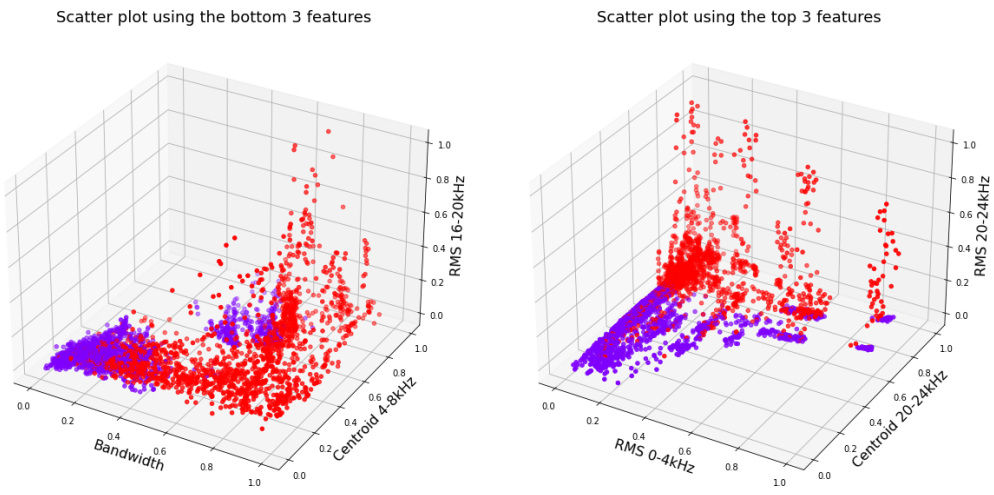


Figure 42: Visualizations of the optimal clusters on the *Tube+Vent* dataset for the bottom three and top three important features. The true classes are represented by the colors.

Finally, the true classes for the bottom three and top three important features were plotted in figure 42. The predicted clusters for the bottom three features have a very non-isotropic shape. For example, the purple cluster is shaped like two clusters, and the red cluster is widely spread across the three dimensions. So it could be considered understandable that the algorithm struggled to predict the clusters for this feature combination.

For the top three features, the optimal clusters are similar to the Centroid bin clusters, not as straightly separated as the predicted clusters in figure 39. The predicted leakage clusters contain a few too many samples and have, at the same time, missed a few leakage samples that are placed inside the purple non-leakage cluster. Here too, the separation is as well as could be expected considering the Euclidean distances between samples.

For all the feature combinations that were tested with the K-means model, the result of the classification process was visualized in confusion matrices. Recall from section 4.4.1

that the upper left box in the confusion matrix displays the number of correctly classified non-leakages, while the bottom right box displays the correctly classified leakages. As the input dataframe had a size of 3648 samples, these two boxes should optimally contain 1824. The other two boxes display false alerts and undetected leakages and should, in a perfect model, contain zero. The confusion matrices for the simple and ultrasonic features can be seen in figure 43. For the bin features, the confusion matrices are included in figure 44, and for the bottom and top 3 features, they are included in figure 45.

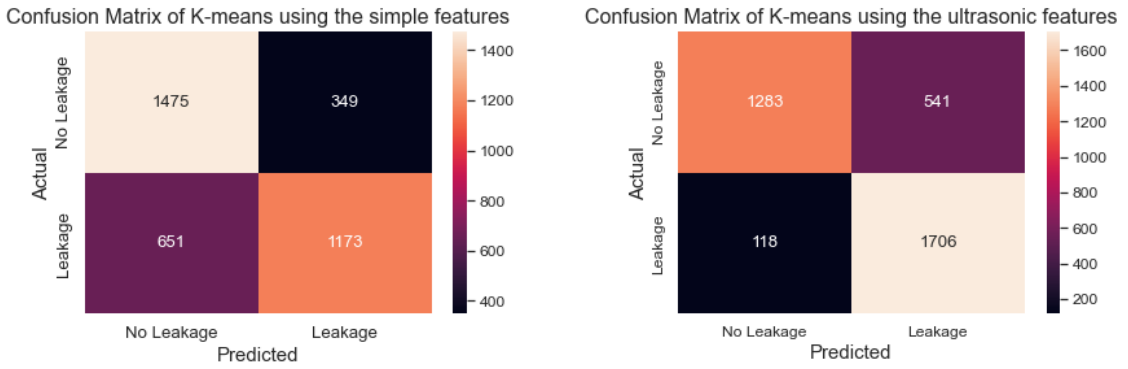


Figure 43: Confusion Matrix for the simple and ultrasonic features using K-means.

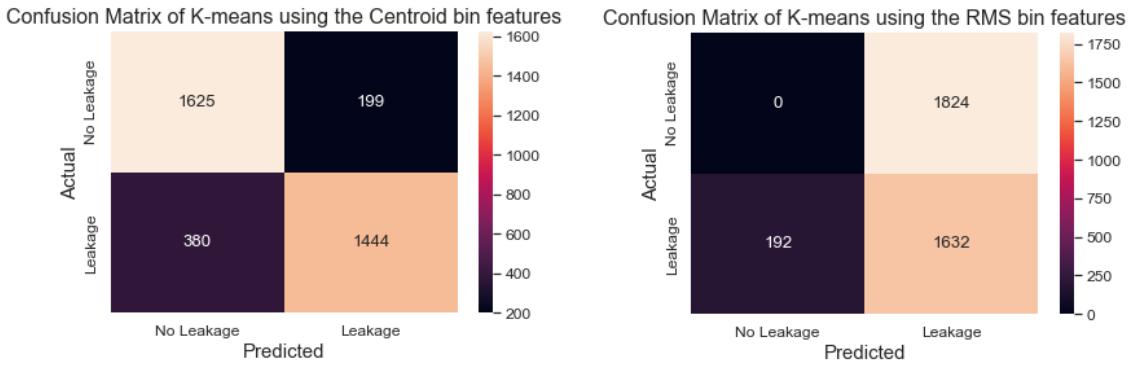


Figure 44: Confusion Matrix for the Centroid and RMS bin features using K-means.

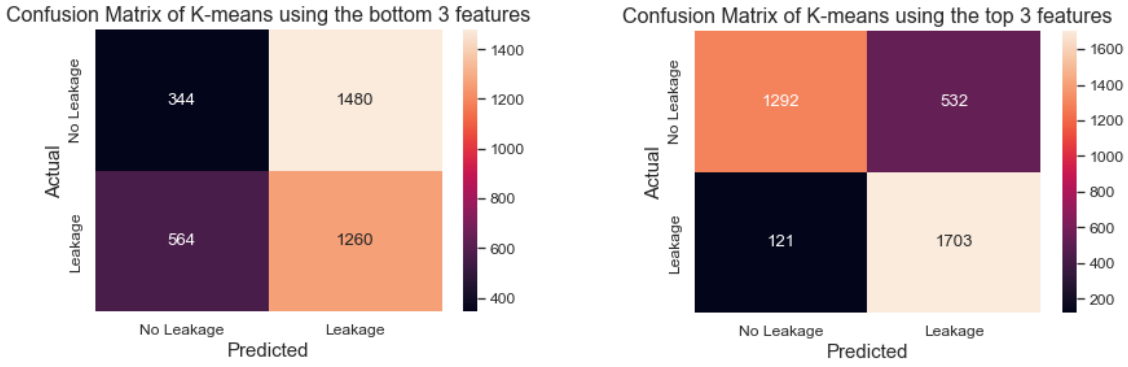


Figure 45: Confusion Matrix for the bottom three and top three important features using K-means.

All confusion matrices, except the ones for the simple features and the Centroid bin features, show a higher number of correctly classified leakages than correctly classified non-leakages. Notice that the RMS bin feature did not manage to classify a single non-leakage correctly and predicted almost all samples to be leakages.

5.3 Supervised classification models

As mentioned previously in the thesis, the supervised classification methods *Neural Networks* and *Gradient Boosting* were tested with the dataset of audio features. There are no visual results of these methods, as their output is only a class label. The methods classify each sample in the dataframes into either class 1 (which represents a leakage) or class 0. Three dataframes were used as input to these models, which are the same ones used for the K-means model, and were presented in table 3. The features used as input to the models are all the 18 initial audio features presented in table 2. Evaluation metrics are calculated by comparing the output of the supervised models to the true class labels, and the result of each metric is included for the *Tube* dataframe in table 9. Similarly, the results for the *Vent* and *Tube+Vent* dataframes are included in table 10 and 11 respectively.

Model	Train Acc	Test Acc	Recall	Specificity	Precision	F1-Score
NN Torch	91.88%	91.40%	84.66%	97.95%	97.56%	90.65%
NN FastAI	91.88%	89.32%	83.07%	95.38%	94.58%	88.45%
XGBoost	95.20%	94.65%	90.48%	98.66%	98.48%	94.31%
Catboost	94.40%	94.25%	89.08%	99.19%	99.07%	93.80%

Table 9: Performance of each model on the *Tube* dataset, represented by different evaluation metrics.

Model	Train Acc	Test Acc	Recall	Specificity	Precision	F1-Score
NN Torch	97.11%	97.11%	93.98%	100%	100%	96.89%
NN FastAI	96.75%	96.82%	93.37%	100%	100%	96.57%
XGBoost	97.11%	96.82%	93.98%	99.44%	99.36%	96.59%
Catboost	96.75%	96.82%	93.37%	100%	100%	96.57%

Table 10: Performance of each model on the *Vent* dataset, represented by different evaluation metrics.

Model	Train Acc	Test Acc	Recall	Specificity	Precision	F1-Score
NN Torch	95.80%	93.42%	88.79%	97.86%	97.54%	92.96%
NN FastAI	94.00%	92.32%	89.08%	95.44%	94.93%	91.91%
XGBoost	95.20%	94.66%	90.48%	98.66%	98.48%	94.31%
Catboost	94.40%	94.25%	89.08%	99.20%	99.07%	93.81%

Table 11: Performance of each model on the *Tube+Vent* dataset, represented by different evaluation metrics.

Of all the runs with the neural network methods, the model from the *PyTorch* library produced the output closest to the true labels and thereby performed best of the two. For the gradient boosting methods the *XGBoost* model performed slightly better than the *Catboost* model based on the evaluation metrics. However, all four models provide adequate and promising results based on the evaluation metrics. Likewise, as for the K-means model, the results of these models are visualized by confusion matrices. For the two neural network models the results for the testing dataset are included in figure 46. For the gradient boosting models, the results are included in figure 47.

The testing set used for these models was of size 730 and contained 357 leakages and 373 non-leakages. Recall that signals were randomly picked for the testing set, so the number of leakages is not necessarily the same as the number of non-leakages. If the testing set was generated again, other signals would be picked.

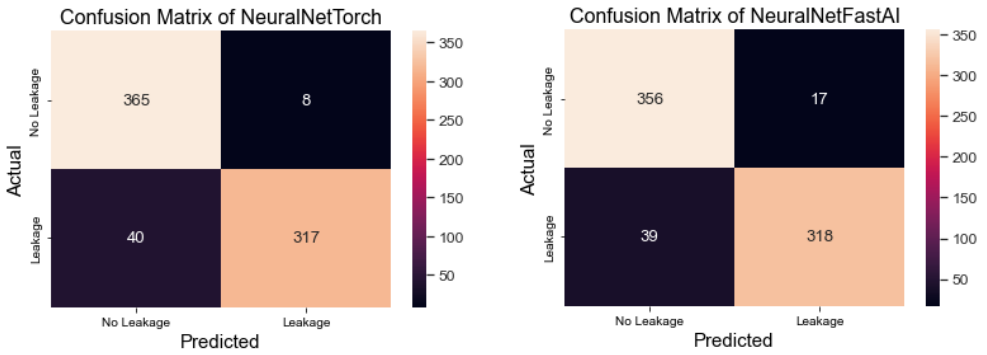


Figure 46: Confusion Matrix for the Neural Network models on the *Tube+Vent* dataframe..

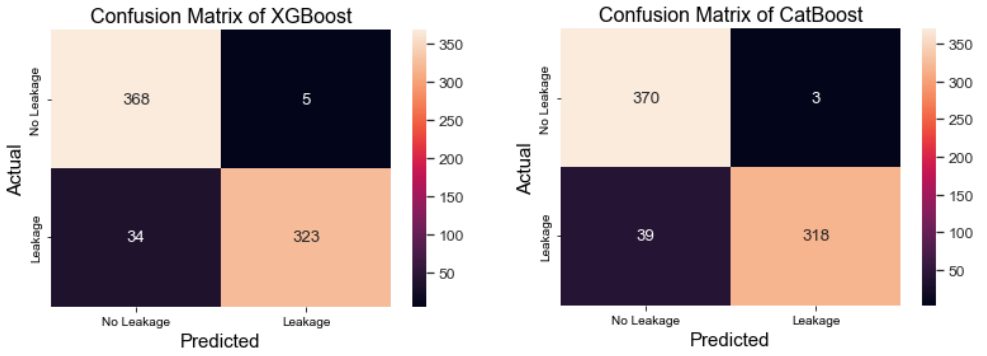


Figure 47: Confusion Matrix for the Gradient Boosting models on the *Tube+Vent* dataframe.

Already by taking a quick look at the confusion matrices in figure 46 and 47, one can get a sense of the models' performance. As the colors of the upper right and bottom left boxes are very dark, they indicate a small amount of wrongly classified samples. For the neural networks, the false alerts are only 8 and 17 out of 373 non-leakage samples for the PyTorch and FastAI networks, respectively. The gradient boosting models provided even fewer false alerts, with only 5 for the XGBoost variant and 3 for the Catboost variant. False alerts are undesirable, and none of these models manage to avoid them completely. However, judging by the few numbers of false positives in the boxes, they come very close.

Although it is a goal to avoid false alerts, it could be even more important to avoid undetected leakages as they have massive consequences for the company. To examine the number of undetected leakages, one must look at the lower-left boxes containing the number of false negatives. For all models, the number of false negatives is close to 40.

The training time of the gradient boosting models was between 2 and 5 seconds for the training dataset used in this project. For the neural networks, the training time was between 9 and 15 seconds. Although these run-times are considered very short, the training time could increase if the models had even more data to train on. Additionally, the training time of the neural networks could increase if the network had a wider or deeper structure. For the gradient boosting models, the limit was automatically set at 10 000 iterations, and the training time could have increased if the limit had been set higher.

5.4 Feature Adjustments

After examining the feature correlation matrix in figure 25 and the feature importance table in figure 26, some adjustments were made to the combination of features used as input to the models. To evaluate if the changes had an impact on the performance of the models, an additional test run on the *Tube+Vent* dataframe was conducted.

From the correlation matrix in figure 25 it seems that the RMS values in each frequency bin are not that different from the others, as the Pearson Correlation Coefficient is over 0.7 for many of them. These features are thereby not providing much new information. The RMS bin features also correlate some with the Centroid bin features. A choice was therefore made to remove the four middle RMS bin features while keeping the highest and lowest bins as they are considered highly important. Two other features that correlated highly were the Bandwidth and Rolloff features, with a Pearson Correlation Coefficient of **0.92**. Looking at the table in figure 26, one can see that the Bandwidth feature scored lowest for the feature importance score and also had a higher p-value than the Rolloff feature. Therefore, the Bandwidth feature was removed. The Centroid 4-8kHz feature also had a low importance score and was removed. The new dataframe used as input to the models is of size 12x3864. The features included in the new feature combination can be seen in table 12, and the result of using them as input to the models can be seen in table 13.

Feature Name	Description
Mic	Microphone Configuration
Rolloff	Spectral Rolloff
Flatness	Spectral Flatness
Centroid 0-4kHz	Spectral Centroid for a frequency bin of 0-4kHz
Centroid 8-12kHz	Spectral Centroid for a frequency bin of 8-12kHz
Centroid 12-16kHz	Spectral Centroid for a frequency bin of 12-16kHz
Centroid 16-20kHz	Spectral Centroid for a frequency bin of 16-20kHz
Centroid 20-24kHz	Spectral Centroid for a frequency bin of 20-24kHz
RMS 0-4kHz	RMS energy for a frequency bin of 0-4kHz
RMS 20-24kHz	RMS energy for a frequency bin of 20-24kHz
PSD >20kHz	Power Spectral Density for frequencies over 20kHz
MAX_AMP >20kHz	Maximum Amplitude for frequencies over 20kHz

Table 12: The new audio features used in this project.

Removing features could impact the performance of the models negatively. Therefore, another test was run with the new features as input to the models to see if that was the case. By comparing the evaluation metrics of the new features to the ones of the initial features, one can see a slight reduction for all models.

Model	Train Acc	Test Acc	Recall	Specificity	Precision	F1-Score
NN Torch	94.20%	92.74%	86.83%	98.39%	98.10%	92.12%
NN FastAI	93.80%	92.47%	87.39%	97.32%	96.89%	91.90%
XGBoost	94.60%	94.52%	89.64%	99.20%	99.07%	94.12%
Catboost	94.00%	94.11%	89.08%	98.93%	98.76%	93.67%

Table 13: Performance of each model on the altered *Tube+Vent* dataset.

The correlation matrix in figure 48 shows the new feature combination and how they correlated. By examining the matrix, one can see directly from the color map that the features are more different than the prior correlation matrix as there are far fewer white boxes. Also, by examining the Pearson Correlation Coefficients, there are only a few features that have above 0.7, which is an improvement from the first feature combination from figure 25.

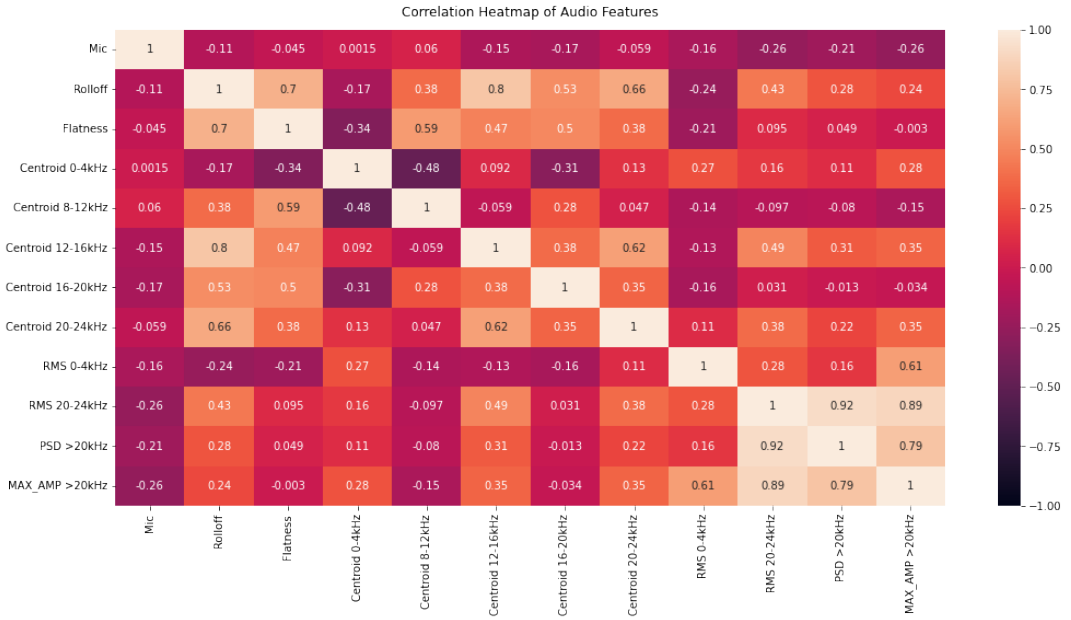


Figure 48: Audio feature correlation for the features extracted from the signals in the *Tubeleak* and *Ventleak* folders.

For the K-means model, the SSE was somewhat reduced with the new feature combination, as can be seen from table 14. However, the model had no improvements as a classifier, as the accuracy in table 14 remained almost exactly the same as the ones in table 6 for all dataframes.

Run	Dataframe	Cluster sizes	SSE	$\overline{SSE}$	Accuracy
1	*Tube*	960	1246.39	0.649	50.05%
2	*Vent*	864	1220.85	0.707	50.06%
3	*Tube + Vent*	1824	2513.44	0.689	50.03%

Table 14: Overview of the sum of squared error (SSE) and the accuracy for the clusters created from the different dataframes containing the alternative audio features.

Recall that section 4.4.3 discussed that the difference in training and testing accuracy could be compared to evaluate the models' ability to generalize. An appropriate model is one where the testing accuracy is slightly lower than the training accuracy. The difference in the testing and training accuracy for the initial *Tube+Vent* was **-2.38** for the Pytorch Neural Network and **-1.68** for the FastAI network. For the XGBoost model, the difference was **-0.54**, and for the Catboost model, it was only **-0.15**. The testing accuracy is thereby slightly lower than the training accuracy for all models. For the gradient boosting models, the difference is very tiny, and it indicates avoidance of overfitting. For the *Tube+Vent* dataframe with the new 12 features, the difference in testing and training accuracy was slightly reduced, and by the order of the placement in table 13 they were respectively **-1.46, -1.33,** +**0.08** and +**0.11**.

6 Discussion

This section of the thesis discusses the results and observations previously presented in section 5. This section also provides a review of the strengths and weaknesses of the utilized machine learning methods. Alternative approaches to the leakage detection problem are presented, and suggestions for improvements are included. The reliability and validity of the thesis work is also discussed, along with the possible sources of error and uncertainties of the methodology. Additionally, limitations to the leakage detection system proposed in section 1 is discussed.

6.1 Evaluation of Performance Results

The performance results that were presented in the previous section gave certain indications as to which audio features and methods are best suited for the problem of detecting gas leakages. Some findings emerged based on the evaluation metrics, the feature analysis, and the cluster visualizations. These findings are included below:

- Frequencies under 4000Hz are considered background noise as the features have very similar values for the leakage versus non-leakage signals.
- Neural networks and Gradient Boosting methods are applicable for the leakage detection problem and can classify gas leakages with appropriate accuracy given that the dataset is of a large enough size. It is not known if the methods are applicable for the detection of other abnormalities than gas leakages, as this has not been explored in this thesis.
- K-means is not applicable for gas leakage detection given the combinations of features used in this project. Characteristics are not solely linearly dependent but could have complex patterns that the K-means is unable to find, in contrast to the Neural Networks and Gradient Boosting methods.
- Most features have larger values for the signals with a gas leakage present, which proves a linear relationship.
- The difference in energy for the leakage and non-leakage signals is greater for the ultrasonic frequencies.
- The superior performance of the Neural Network and Gradient Boosting models as opposed to the K-means model indicates that the linear relationship is not enough to separate the leakage signals from the non-leakage signals adequately.
- There are some apparent differences in the utilized features in the leakage versus non-leakage signals. Thereby, some of the features used in the project are relevant for the intended purpose and should be investigated further.

From the feature analysis, it became clear that there are some specific acoustic characteristics of gas leakages. Audio signals with gas leakages had on average higher values for all audio features that were investigated. Some features had a more significant increase in values than others, which includes the Bandwidth, Rolloff, and Centroid 20-24kHz features. Some of the features that have high values for the leakage signals are also considered the most important features for the neural network model. Figure 26 showed that the Centroid

20-24kHz, RMS 20-24kHz, and RMS 0-4kHz features were the three most important features for the neural network. Comparing the table with average feature values (see table 4) to the importance table in figure 26, one can see that there is not necessarily a correlation between the most important features and the features with higher differences for leakage signals. This discovery indicates that the linear difference between feature values of leakage and non-leakage signals is not the dominating factor on which the neural network model depends its predictions. K-means bases its predictions on linearity, while a neural network can see more complex connections. Since the neural network outperformed K-means for all three input datasets, one could draw the inference that the characteristics of gas leakages are more complex than just the increase in feature values. The K-means algorithm was not able to appropriately separate the classes into different clusters and had at most an accuracy of 84.13%. Neither did K-means manage to provide new information about hidden patterns in the dataset. Nevertheless, some other potential audio features and combinations could be tested with K-means. The method cannot be completely disregarded as a possible leakage detection method based on this thesis study.

From the model evaluation, it became clear that the models generally had a higher score for the specificity metric than the recall metric. This indicates that the models are better at correctly classifying the normal signals than the leakage signals. For the K-means model, there were large variations, and it performed very poorly on both metrics for some combinations, especially the RMS bin features and the bottom three important features, which only provided a specificity score of 0.00% and 18.86% for the respective feature combinations, as could be seen in table 8.

6.1.1 Complexity and Generalization

For each sound recording, the proposed system wants to check for leakages; the software program would need to extract d features. In this study d has been the 18 features in table 2 and the 12 features in table 12. Increasing the number of features d increases the time complexity of the system, as well as the focus area. Fewer features could, therefore, be better for the models' overfitting, run-time, and complexity. By comparison of the results for the initial feature combination in table 11 and the altered feature combination in table 13, the accuracy of all models was only slightly reduced (by at most 1.6%) when the number of features was reduced from 18 to 12. The accuracy for the K-means model remained the same, but the SSE was reduced when the number of features was reduced, as can be seen by comparing table 6 and 14. The importance of increased accuracy versus increased complexity and the chance of overfitting is based on subjective opinions. Therefore, it cannot be deduced from this study which of the two alternatives is superior. Both feature combinations provide adequate results in the opinion of the author, a master's student in ICT. However, it can be deduced from the study that both alternatives are appropriate feature combinations for the problem at hand.

The size of the focus area or the number of properties of sound that the detection system is searching for leakages in will contribute to the system's ability to generalize. With the use of many features, the classification will be impacted by several different audio properties. Some that might not be as relevant to the characteristics of a leakage. Using irrelevant audio features in the system should be avoided as it does not contribute positively to the performance of the models. As previously stated in section 4.4.3, the training and testing accuracy could be used to evaluate the generalization ability of the machine learning models. Since the testing accuracy was only slightly lower than the training accuracy for all models with the initial 18 features, ranging from a difference of -0.15 to -2.38, the models

did not exhibit clear signs of overfitting. For the other 12 features, the difference was even smaller, ranging from -1.33 to -1.46 for the neural network model and only +0.08 and +0.11 for the gradient boosting models. The latter two models even had a slight increase in training to testing accuracy when reducing the features from 18 to 12. Therefore, it could be deduced that the gradient boosting models have a good ability to adapt to new unseen data. The same applies to the neural network models, although they had a marginally higher training accuracy than testing accuracy compared to the gradient boosting models.

Despite the fact that the training and testing accuracy indicated good generalization abilities of all models, there could still be some overfitting. For instance, the models were trained on only two types of leakages and three primary environments, which were the ones included in the dataset presented in section 3. The two leakage types were gas leakages from a vent and gas leakages from a tube. The three environments included a laboratory without background noise and two rooms with the hydraulic machine background noise, and general factory workshop noise. The latter two were also adjusted in volume, which could be considered as additional background noise (and thereby also different environments). However, there are many other possible environments that the models could have trained on, and it is not possible to be sure that the models used in this thesis will perform similarly if deployed to detect leakages in those environments. Even if the models have an appropriate fit for the dataset included in this study, they could prove to be overfitted if applied to different environments.

6.1.2 Strengths and Weaknesses of Methods

Although the machine learning methods used in this project were carefully chosen for the problem at hand through a literature review, they have some weaknesses. The choice of input to the methods is the most important factor for the performance, but the methods also have some procedural weaknesses that affect the performance.

The poor performance of the K-means method on the leakage dataset could be explained by the fact that the algorithm only examines the Euclidean distance between two data points in nD. The problem is likely more complex than what a numerical distance calculation can solve. The K-means method looks for linear relationships. If there are few observations of that kind of relationship, the method naturally performs insufficiently. The type of relationships that existed between the leakage signals was not known in advance and was explored during the feature analysis and the clustering process. The feature analysis showed that there were some linear relationships, but the clustering process showed that these relationships were dependent on the environment and the background noise, which made it hard for the K-means algorithm to generalize. By visualizing the data points of three features from the leakage dataset in 3D, the shape of the two classes could be examined. The classes, namely leakage and non-leakage, had irregular shapes, which were neither isotropic nor convex. As mentioned in section 4.3.2, the K-means algorithm uses inertia to classify points into clusters, and the inertia assumes that clusters are isotropic and convex. Since the optimal clusters, or class separations, did not have the appropriate shapes, the K-means method naturally performed inadequately. Additionally, the K-means method has a complexity that is not suited for large applications, as the problems are solved in $O(n^2)$ time. Although the dataset used in this study was not considered very large, the run-time could be worse if the method was tested on a different larger dataset.

Strengths of neural networks and gradient boosting methods include the capacity to cluster multivariate data and find non-linear relationships between data in a dataset. The influence of outliers on the performance of the K-means method is more significant than for the neural networks. So neural networks are less impacted by irregularities in the dataset than the K-means method. One weakness of neural networks is that the method is highly dependent on a balanced and quality-assured dataset, which also needs to be very large in order for the network to be appropriately trained. Another weakness of neural networks is that the data analyst has little influence on the function of the network and which weights are used. Neural networks are often described as black boxes since studying their structure does not provide any insights into what the function of the network is examining. In other words, the data analyst is not able to know how all the individual neurons in the network work together to provide the final output. Both neural networks and gradient boosting methods are prone to overfitting if not stopped at the appropriate iteration, which can be hard to define. In other words, there are strengths and weaknesses of all the methods utilized in this study, and in order to identify the most optimal fit for the problem at hand, they must be tested.

6.1.3 Significance of Results and Study

The results indicate that machine learning methods for sound classification can detect gas leakages, specifically Neural Networks and Gradient boosting methods. For the company that initiated this project, Equinor, the project showed promising results for detecting leakages through sound recording and anomaly detection techniques on noisy worksites similar to theirs. The hypothetical system introduced in the introduction could be a possible improved detection system for their worksites compared to the standard system based on gas monitors, given that the system is based on NN or GB models. The system would naturally have to be tested carefully and compared with the predecessor before being put to use either as a supplementary or substitute system. The project brings light on the possibility of using acoustic detection systems as a safety measure for leakage detection for not just Equinor but for other companies dealing with gases as well. The project is, however, not conducted with what could be considered a large enough dataset nor general enough data to be sure of the feasibility of such a system. The system is not tested over a long time either, where the utilized recordings are taken in the same environment over a short period of time, so there is not enough diversity in the data to be sure if it will fit with real-world applications such as on an oil and gas platform. In an extensive context, these results do not provide much new information to the science field, as it is already known that sound can be classified with acoustic anomaly detection methods. It has, however, contributed to an indication that Neural Networks and Gradient Boosting methods, in particular, are applicable for the classification of gas leakages in noisy environments. Also, the results indicate that the K-means algorithm does not work sufficiently on gas leakage detection as the characteristics of the leakages are more complex than a linear comparison method can comprehend. This study could be considered as a proof of concept for the proposed leakage detection system presented in figure 1, as the study indicated that the system is feasible and has great potential to replace or support the prior detection system composed of gas concentration monitors.

6.2 Reliability and Validity of Project

It is crucial to ensure reliability and validity when experimenting with the objective of potentially replacing prior solutions for real-world applications with alternative methods. Taking the first demand into account, a project needs to have a high degree of quality and consistency in results to be considered trustworthy and reliable. During the whole project, the experimentation has been conducted with careful and attentive actions, aiming at a high degree of reliability and reproducibility. As state-of-the-art machine learning methods have been used through well-known coding libraries, human alterations have been mainly unnecessary, thus reducing potential human errors. All the utilized coding libraries have a great reputation for being reliable. The libraries contain functions for calculating evaluation metrics automatically, meaning avoidance of potential human calculation errors. The dataset used in this project contains signals recorded with a different setup, at different locations, and with several microphones. This substantiates the consistency of leakage characteristics across time and place to a certain degree. Through the way the code in this project has been developed, the results should be perfectly reproducible by running the different attached files in the specific order that is explained in the attached coding project in appendix B.

Concerning validity, a project needs to contain accurate calculations and results. Additionally, the results should correspond to real and natural properties, characteristics, and variations in the physical world. Reliable results can indicate valid results; however, reliability on its own is not enough to ensure an accurate reflection of the real situation. This demand is harder to assess, but some measures were taken to strive for validity in this project. The methods and dataset used in this project were carefully chosen and are considered appropriate for the problem at hand. Although the dataset could have been even larger, it is relevant to the problem and was the best available alternative. All the chosen methods and techniques are state-of-the-art and of high quality. They are based on thorough research and existing knowledge. The evaluation metrics used in this project are set to measure exactly this kind of problem, namely classification. Already at the early stage of the project, during the data collection, validity was considered. By including different types of gas leakages in the dataset, the results are more generalizable, thereby having a higher chance of fitting the real-world situation.

6.2.1 Sources of Potential Error or Flaws

A requirement that all machine learning methods have is a large and balanced dataset of high quality. The results of machine learning methods depend highly on the input and having an unbalanced dataset could persuade the model towards the class that contains the majority of the data samples. Potential disturbances of classification performance have been avoided by using a large, balanced, and normalized dataset. Recall that the initial leakage dataset was altered before being sent to the ML algorithms. Audio features were extracted from the signals and used as input to the algorithms.

The potential combinations of audio features to use is plenty, and the choice of which to use for this problem could have been sub-optimal. For instance, using ultrasonic features could prevent generalization by focusing the model on high-frequency leakages. This entails that the model could be accustomed to leakages being in the higher frequencies and could therefore miss the lower or medium-sized leakages. Then again, the medium-frequency leakages could be easier for the personnel to hear. However, depending on humans to detect some of the leakages is already a weakness for the present-day detection system. The findings in the feature analysis conducted in this project indicated that most leakages are present in the high-frequency range, close to the ultrasonic frequencies. It could be the objective of a model to become an expert on detecting these high-frequency leakages and not the rest. In that case, a prioritization of which leakages to focus the detection system towards should be up to the system's owners to define.

Another feature that could be a point of worry is the maximum amplitude. Using a maximum amplitude feature could be disadvantageous as the amplitude depends on where the recording device is located relative to the leakage source. The amplitude of a signal is higher the closer the microphone is to the leakage source. If the model train on signals where the microphone has been placed close to the source, it could be taught to believe that the amplitude has to be over a certain size for the signal to be considered as containing a leakage. By way of explanation, there could be a leakage present further away from the microphone that is not detected due to the model having learned a "false relationship" between amplitude and leakage characteristics. Since all the signals that the machine learning methods have trained on are from the same dataset with the same four microphone locations, the model will perhaps not generalize well to leakage sounds from totally different locations.

Additionally, the use of several frequency bins for the same feature, like what was done with the RMS and spectral centroid, could influence the model to pay more attention to one type of characteristics more than others. There are no specific state-of-the-art features to be used for leakage detection. This entails potential sub-optimal choices. The combination of features to use for a specific problem have to be experimented with and studied. A study of which features that are fitting for the problem of detecting gas leakages has been performed to a certain extent in this project but likely needs further examination.

6.3 Improvements and Alternative Methods

Many datasets and approaches could be experimented with in regard to the problem of detecting gas leakages. Other alternative datasets could have been used, including datasets that do not contain leakages at all. Approaches that focus on anomaly detection by training solely on a normal state would then be possible to use. A drawback of only training on the normal state is that any anomalies would be classified as a leakage. An anomaly could just be a special incident such as the personnel dropping heavy equipment to the floor, creating a loud bang. Naturally, it would not be beneficial for the system to start the alarms for an abnormality like the one in the example. Additionally, each worksite might have its own normal state with different characteristics from the normal background noises. Therefore, it might be necessary to train the machine learning models for each specific environment, but this is a time-consuming task. Training on only the normal state of the exact place where the detection system shall be used is a possibility. In that case, the model would have learned the characteristics of the background noises of that exact place. Nevertheless, the background noises could change throughout the weeks, months, or years based on the processes running at the industrial site. It could be hard to define a normal state when the normal state constantly changes. For the model to be able to generalize, it may have to be trained on many different background noises/normal states over time. Alternatively, it could be trained with a vast amount of environments and hope it will learn what a normal (non-leakage) state is through the many examples of normal industrial background noises. It might be easier for companies to gather data on different industrial environments' normal states than to gather data on real leakages, but there could be some challenges with that. For instance, one needs to be completely sure that there are no undetected leakages anywhere at the worksite where the recordings are being taken to avoid mislabeling the data samples. Feeding the machine learning models with wrongly labeled data will teach them a false relationship and disturb the model's perception of a normal state.

Even for each dataset, such as the gas leakage dataset utilized in this project, there could be different approaches for analysis. Some alternative approaches could have been investigated for the specific dataset used in this project. Firstly, the laboratory recordings could have been removed from the input dataset to the models as they might disturb the perception of leakage characteristics if they are not representational of leakages in a noisy environment. All the sound recordings (signals) in the dataset have a duration of approximately 30 seconds. The whole signals were used to extract features, but they could also have been partitioned into several smaller signals in order to have more data points in the dataset. For instance, each 30-second file could have been divided into six files of 5 seconds, which would have increased the size of the dataset with a factor of 6. In order to partition the files, one would need to be sure that the signals are stationary, meaning that the leakage is present during the whole duration of the signal. Alternatively, to scale up the input dataset, one could combine the gas leakage dataset used in this project with another dataset to get leakage files that are taken with other specifications and in other environments. Another possibility is to use recordings from different locations with different background noises and add artificial leakages afterward.

Alternative approaches to the general methodology are, of course, to use other features or other classification methods, such as the ones introduced in related work in section 1. Many alternative machine learning methods could have been experimented with in this thesis; only time has constrained the author to focus on three. As mentioned previously, there is no state-of-the-art audio feature that is defined to work optimally for gas leakage detection. Therefore, many alternative audio feature combinations could have been used to train the machine learning models. The audio features do not necessarily need to be presented by numerical values either. It is, for instance, possible to use visualizations of the MFCCs of signals as input to a machine learning model such as a Convolutional Neural Network. This approach was researched in the project thesis, and related work by other scientists provided promising results.

6.4 Feasibility of the Proposed Detection System

Either directional or omnidirectional microphones are considered feasible for the recording of leakages. These types of microphones have the potential for a more specific location-based detection, as they can indicate to the robot where the leakage source is. It could be made possible through a software program to tell the robot to move towards where the amplitude of the signal is higher, but this has not been considered in this thesis project. The microphones must have a high enough sampling rate to be able to detect ultrasonic frequencies, at least 48kHz. The directional microphones could minimize the unwanted background noise by focusing the sampling of sound to the front of the microphone, thereby reducing the influence of noise from the sides. Either of the alternatives is suitable as recording equipment and can be used in the proposed detection system. Using mobile robots to carry the microphones is both a feasible alternative and could reduce costs by avoiding the deployment of personnel. Mobile robots can be used at different locations on the same worksite or at multiple worksites. Continuous recordings can increase the amount of empirical data that can be used later to retrain and improve the machine learning model in order to classify the leakages on the specific worksites more accurately. This system is made possible by the high quality of the modern robotic systems and the emerged internet of things technologies.

Cloud services could be beneficial for storage in the proposed system. Running the software application in the cloud allows multiple users (personnel) to view data in real-time. Since it is desirable to stop leakages as early as possible, it is preferable to view the information in real-time and get automatic alerts. Furthermore, cloud services provide the possibility of accessing historical data and information from any location. The solution is also less expensive than using one's own servers, as high-quality and costly GPU cards are required. Additionally, the cloud provider handles the complex task of building, managing, and maintaining the servers, so the personnel/users can focus on other tasks. Supervised machine learning models have, through the experimental study in this thesis, proven to be feasible for gas leakage detection. The research and results of the specialization project and thesis project also indicate that machine learning approaches are preferred over basic digital processing techniques for classification as they handle more complex problems and find characteristics and patterns that are less clear and nonlinear. Using numerical audio features as input to the machine learning algorithms is a feasible alternative, but images in the form of spectrograms and visualization of MFCCs could be a possible alternative.

Recall that the training time for the machine learning models was only a few seconds. The testing time for one signal is even less. When the pre-trained model is set to use, it is only the testing time that could create a delay for the alarm system if the model discovers any leakages. However, a testing time of around a second should not be a problem for the system's feasibility, and the information could still be considered as being given in real-time. The results of the experimental study indicate that the proposed system as a whole is feasible and can likely be turned into reality, naturally after some testing and comparison to the prior detection system. A proof of concept is therefore provided through the results in the thesis. A pilot project might provide more definitive answers as to the system's feasibility for the specific worksites.

6.5 Limitations and Uncertainties

There are some limitations to the proposed system. Since the signals are sampled with a sampling rate of 48kHz, any leakages with a frequency above 24kHz will likely go undetected. Potential low-frequency leakages in the range $\leq$ 4000Hz will also go undetected as the frequencies under 4000Hz are ignored. However, it is believed that leakages are likely not present in this low-frequency range. Another limitation is that only gas leakages are at focus in the proposed system, and other abnormalities are ignored. Some companies might find it useful to have a system that also checks for other abnormalities. Training a single machine learning model to detect different types of abnormalities might be too complex as it would need to learn the characteristics of different abnormal sounds at the same time. Perhaps a system that contains several models that have been trained separately on each type of abnormal sound could be an alternative.

There also exists some uncertainties about the methodology of the software part of the system. For instance, the is uncertainty related to the choice of appropriate features, especially concerning the maximum amplitude and the other ultrasonic features. As previously mentioned, the amplitude of a signal is based on where the microphone is placed relative to the leakage source. The amplitude will therefore be higher if it is closer to the source. Training on this feature could be unproductive as it would depend on the microphone and robot to take recordings at the same exact location each time. In that case, the same exact locations should also have been used to train the machine learning model. Otherwise, the model could be accustomed to very high amplitudes being a characteristic or an implication of a leakage. The model could thereby not detect leakages farther away from the microphone. Additionally, all recordings in the utilized dataset are taken with the microphones placed at exactly the same spot relative to the leakage source (with four different placements). There is uncertainty related to the model's ability to generalize to leakages in many locations and if it should have been trained with recordings from more than four placements. There is a larger certainty over the normal state (non-leakages) when the robot is standing at the same place each time. This entails that the system should be designed so that the robot takes new recordings at the same spot each time and is trained with the same specifications. It is not known if the dataset is representative of the real world. The impact of weather on the microphones, which would apply to outside oil platforms, is not considered since the recordings in the utilized dataset were taken inside a laboratory and in designated rooms. Therefore, there exists uncertainty about the effect of outside disturbances such as wind and weather. Finally, there is uncertainty as to whether the limit for background noise is exactly at 4000Hz, or if some important information, therefore, is filtered away.

7 Conclusions

In this final section, a summary of the contents and findings in this master thesis is presented, along with recommendations for further work.

7.1 Summary

Industries that deal with high-risk substances such as oil and gas are especially concerned about safety. Leakages of these substances can lead to massive economic and environmental consequences and increase the safety risks of workers at an industrial plant. The problem of detecting leakages early is therefore of great value to companies in these industries. The fourth industrial revolution, *Industry 4.0*, has evoked a trend of digitizing data from processes and worksites with the help of cyber-physical systems and information and communication technology (ICT). A vast amount of data generated daily at digitized companies allows for big data analysis and the possibility of gaining insights into the processes and how to improve them. Therefore, the main objective of this master thesis was to build a gas leakage detector based on machine learning methods fitting for big data analysis, capable of detecting leakages in noisy industrial environments. The proposed detection system consisted of a mobile robot taking sound recordings at a worksite using a directional microphone and sending the digitally sampled sound in the form of 30-second long signals to a detection model that searches for leakage characteristics. The model would know the characteristics through a prior training phase based on signals with and without leakages. The proposed system and the background for the problem and objective of this thesis were presented in section 1.

In order to reach the objective of this thesis, several approaches for acoustic anomaly detection were reviewed, concluding that supervised classification methods can detect leakages. A literature review on related work in the field of anomaly detection was presented in section 1.5. The state-of-the-art equipment used for the purpose of recording sound was also introduced, where the emphasis on the practical use of directional microphones as supposed to contact microphones or cameras was discussed. Additionally, state-of-the-art acoustic anomaly detection methods based both on basic digital processing and machine learning techniques were presented. Further, section 2 presented relevant theories on signal processing and audio feature extraction, in addition to a deeper explanation of the components of the machine learning methods *Neural Networks, Gradient Boosting Methods* and *Clustering* used for classification tasks. The structure and features of the gas leakage dataset utilized for experimentation with the machine learning methods were gone through in section 3. During section 4 the methodology used during experimentation was explained in detail. The results of utilizing the machine learning methods on the gas leakage dataset were presented in section 5. The evaluation scores were discussed in section 6, along with proposals for improvements and alternative approaches for further experimentation on this problem. Uncertainties and potential sources of error were also discussed in section 6, as well as an assessment of the significance of the findings.

7.2 Findings and Conclusions of the Thesis

The experimental study conducted in spring 2022 provided some results that yielded new information about the utilized leakage dataset and confirmed some known facts about leakage characteristics. The feature analysis indicated that the leakages are mostly present in the higher frequencies and that the values of all audio features are generally higher for leakage signals. Especially the study related to frequency bins showed that the leakage signals contained far more energy than the non-leakage in the higher frequencies close to the ultrasonic range. The power spectral density of the signals increased for the leakage signals as the frequency increased. The largest differences were for the frequencies of around 19 000Hz and higher, as was presented by the plot in figure 22. These findings indicate the importance of focusing on elevated values in the higher frequency range (from around 14kHz and above) when searching for leakages.

The performance of the *K-means Clustering* method was inadequate on the gas leakage dataset and showed neither hidden patterns in the dataset nor worked as a good separator or classifier for the leakage and non-leakage signals. The clusters created by K-means were unevenly sized and, at best, produced an accuracy of 84.13%. Both neural network models performed adequately, as evident by the accuracy scores of at best 97.11% and 96.82% for the *Pytorch* and *FastAI* networks, respectively. At worst, the neural networks classified the signals with an accuracy of 91.40% and 89.32%, which is still considered good results. The *XGBoost* model classified the signals with an accuracy of 96.82% at best and 94.54% at worst. Closely following the XGBoost model was the *Catboost* model, which produced an accuracy of 96.82% at best and 94.11% at worst. The K-means model showed signs of underfitting, while the remaining models showed little to no signs of overfitting. The evaluation scores demonstrated that *Recurrent Neural Networks* and *Gradient Boosting* methods such as *XGBoost* and *Catboost* work sufficiently on detecting gas leakages in the utilized dataset. Given some requirements and limitations, the latter two approaches show great promise for being the basis of a leakage detection system in noisy dynamic environments. These requirements include the use of directional microphones with a sample rate of at least 48kHz. A limitation to the models is that gas leakages above the frequency of 24kHz will likely go undetected.

In conclusion, the proposed system that was introduced in figure 1 is considered to be feasible as the results of the experimental study indicated that machine learning methods can detect gas leakages from sound recordings taken by directional microphones with good performance. Modern cloud services have allowed for fast connection between remote hardware and real-time software applications and can be useful for such a system. A software application should be easy to connect to an alarm system at the relevant worksites. Through internet of things and robotics, the detection system can be made mobile and adaptable to different worksites such as oil and gas platforms. More research and studies on the audio features and machine learning methods, in combination with physical system testing, are necessary before putting such a system to use. Nevertheless, the thesis serves as a proof of concept for the use of machine learning based acoustic anomaly detection techniques for the purpose of gas leakage detection in noisy industrial environments.

7.3 Further Work

This thesis gave positive indications for the use of an acoustic leakage classification system as a substitute or additional system to the gas monitors used today. As there is no state-of-the-art acoustic detection system yet, and the experimental study did not manage to identify the most optimal model to use for such a system, more research should be conducted on this problem. Further work on this problem could include research and testing with the use of other features, methods, and recording equipment. For instance, recording equipment that samples sound with a higher sampling rate than 48kHz could allow for the detection of even higher high-frequency leakages than the ones included in the dataset used in this thesis. Alternative datasets could be explored, which could be accumulated from recordings of different worksites and be of much larger size than the one used in this experimental study. Additionally, the models created in this thesis would have been interesting to test out on sound from real worksites to see if they had adequate generalization abilities.

By expanding the problem to include a more exact localization of leakages, more research questions appear. Such as the feasibility of making a robot check continuously for leakages and move towards a leakage source if it receives an alert of a leakage close by. Using a directional microphone allows for the possibility of locating leakages, as the signal-to-noise ratio will be higher when the microphone is pointed directly at the leakage source. Another similar problem to consider for further research is the detection of abnormal sounds from machinery, thereby determining if machines have begun to decay and are close to reaching their life limit. In this thesis, acoustic anomaly detection (AAD) techniques based on machine learning have been applied to detect only gas leakages. However, the techniques could be of interest for further research on machine decay and the detection of other abnormalities.

To summarize, the use of acoustic data, machine learning, and AAD techniques opens up many possibilities in the sound analysis field, from detecting leakages of gas, locating them, and possibly identifying other abnormalities such as poor machinery conditions. Some of these topics have been broadly investigated in the science field already, and some should be explored even deeper. What should be memorized from this thesis is that the technologies that emerged from Industry 4.0 have opened up the opportunity of improving the gas leakage detection systems of today. Furthermore, robotics, Artificial Intelligence, Big Data Analytics, and Cloud Computing are useful technologies available for companies that want to digitize and optimize their processes to meet the increasing demands of the green transition, focusing on climate action and emission reduction. The applications and possibilities are vast regarding the use of the mentioned technologies, and a lot more interesting research on how they can stimulate innovation will likely and hopefully be seen in the future.

Bibliography

[1] Aleix Solanes and Joaquim Radua. Advances in Using MRI to Estimate the Risk of Future Outcomes in Mental Health - Are We Getting There? *Frontiers in Psychiatry*, 13, 2022. http://dx.doi.org/10.3389/fpsyt.2022.826111.

[2] Alexander Vezhnevets and Olga Barinova. Avoiding Boosting Overfitting by Removing Confusing Samples. In Joost N. Kok, Jacek Koronacki, Raomon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron, editors, *Machine Learning: ECML 2007*, pages 430–441, Springer Berlin Heidelberg, 2007. https://doi.org/10.1007/978-3-540-74958-5_40.

[3] Anthony Schenck, Walter Daems, and Jan Steckel. AirLeakSlam: Automated Air Leak Detection. In Barolli, Leonard., Peter Hellinckx, and Juggapong Natwichai, editors, *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 746–755. Springer International Publishing, 2020. http://dx.doi.org/10.1007/978-3-030-33509-0_70.

[4] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461, 2003. https://doi.org/10.1016/S0031-3203(02)00060-2.

[5] Arunan Ramalingam and Sridhar Krishnan. Gaussian Mixture Modeling of Short-Time Fourier Transform Features for Audio Fingerprinting. *IEEE Transactions on Information Forensics and Security*, 1(4):457–463, 2006. https://doi.org/10.1109/TIFS.2006.885036.

[6] R. Bachu, S. Kopparthi, B. Adapa, and Buket D. Barkana. Voiced/Unvoiced Decision for Speech Signals Based on Zero-Crossing Rate and Energy. In Elleithy, Khaled., editor, *Advanced Techniques in Computing Sciences and Software Engineering*, pages 279–282. Springer, 2010. https://doi.org/10.1007/978-90-481-3660-5_47.

[7] Bogdan Mihai and Panu Mihai. Labview modeling and simulation of the digital filters, 2015. http://dx.doi.org/10.13140/RG.2.1.2567.6641.

[8] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. LibROSA: Audio and music signal analysis in Python. In *Proceedings of the 14th Python in Science Conference*, volume 8, 2015. https://doi.org/10.5281/zenodo.6097378.

[9] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54(3):1937–1967, 2020. https://doi.org/10.1007/s10462-020-09896-5.

[10] Cheng-Yu Peng, Ully Raihany, Shu-Wei Kuo, and Yen-Zuo Chen. Sound Detection Monitoring Tool in CNC Milling Sounds by K-Means Clustering Algorithm. *Sensors*, 21:4288, 2021. http://dx.doi.org/10.3390/s21134288.

[11] Chieh-Feng Cheng, Abbas Rashidi, Mark A. Davenport, and David V. Anderson. Evaluation of Software and Hardware Settings for Audio-Based Analysis of Construction Operations. *International Journal of Civil Engineering*, 17(9):1469–1480, 2019. https://doi.org/10.1007/s40999-019-00409-2.

[12] David Johnson, Jakob Kirner, Sascha Grollmisch, and Judith Liebetrau. IDMT-ISA-Compressed-Air. *Fraunhofer Institute for Digital Media Technology*, 2020. https://www.idmt.fraunhofer.de/en/publications/isa-compressed-air.html (Accessed: 2021-09-15).

[13] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference for Learning Representations*, 2015. https://doi.org/10.48550/arxiv.1412.6980.

[14] Dimitrios Kampelopoulos, Nikolaos Karagiorgos, Georg P. Kousiopoulos, Dimitrios Porlidas, Vasileios Konstantakos, and Spyridon Nikolaidis. An RMS-based Approach for Leak Monitoring in Noisy Industrial Pipelines. In *2021 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6, 2021. https://doi.org/10.1109/I2MTC50364.2021.9460014.

[15] Douglas Cohen. Music - Its History, Language and Culture. *CUNY Brooklyn College*, 2015. https://academicworks.cuny.edu/bc_oers/4/ (Accessed: 2021-11-20).

[16] Equinor ASA. Investigation of the fires at Tjeldbergodden and Hammerfest now concluded. *Equinor Official Website*, 2021. https://www.equinor.com/en/news/20210512-investigation-fires-tjeldbergodden-hammerfest-concluded.html (Accessed: 2021-10-17).

[17] European Commission. Equipment for potentially explosive atmospheres (ATEX), 2014. *Official Website of the European Union*, https://ec.europa.eu/growth/sectors/mechanical-engineering/equipment-potentially-explosive-atmospheres-atex_en.

[18] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreaas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. https://doi.org/10.48550/arXiv.1201.0490.

[19] Feng Rong. Audio Classification Method Based on Machine Learning. In *2016 International Conference on Intelligent Transportation, Big Data Smart City (ICITBS)*, pages 81–84, 2016. https://doi.org/10.1109/ICITBS.2016.98.

[20] Giovanni Di Leo and Francesco Sardanelli. Statistical significance: p value, 0.05 threshold, and applications to radiomics — reasons for a conservative approach. *Eur Radiol Exp*, 4(18):25–36, 2020. https://doi.org/10.1186/s41747-020-0145-y.

[21] Guillaume T. Vallet, David Shore, and Michael Schutz. Exploring the Role of the Amplitude Envelope in Duration Estimation. *Perception Journal*, 43(7):613–630, 2014. http://dx.doi.org/10.1068/p7656.

[22] Hans Torvatn, Pål Kamsvåg, and Birgit Kløve. Industry 4.0 Visions and Reality - Status in Norway. In Farhad Ameri, Kathryn E. Stecke, Gregor von Cieminski, and Dimitris Kiritsis, editors, *International Conference on Advances in Production Management Systems (APMS). Towards Smart Production Management Systems*, pages 347–354. Springer International Publishing, 2019. https://doi.org/10.1007/978-3-030-29996-5_40.

[23] Haomiao Zhou, Zhihong Deng, Yuanqing Xia, and Mengyin Fu. A new sampling method in particle filter based on Pearson correlation coefficient. *Neurocomputing*, 216:208–215, 2016. https://doi.org/10.1016/j.neucom.2016.07.036.

[24] J. Hartigan and M. A. Wong. Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. https://doi.org/10.2307/2346830.

[25] Hayder Al-Behadili, Ku Ku-Mahamud, and Rafid Sagban. Rule pruning techniques in the ant-miner classification algorithm and its variants: A review. In *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 2018. http://dx.doi.org/10.1109/ISCAIE.2018.8405448.

[26] Henri J. Nussbaumer. The Fast Fourier Transform. In *Fast Fourier Transform and Convolution Algorithms*, pages 80–111. Springer Berlin Heidelberg, 1981. https://doi.org/10.1007/978-3-662-00551-4_4.

[27] HyunYong Lee, Nac-Woo Kim, Jun-Gi Lee, and Byung-Tak Lee. A Study on Distance Measure for Effective Anomaly Detection using AutoEncoder. In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1348–1348, 2020. https://doi.org/10.1109/ICTC49870.2020.9289177.

[28] I. Stančin and A. Jović. An overview and comparison of free Python libraries for data mining and big data analysis. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 977–982, 2019. https://doi.org/10.23919/MIPRO.2019.8757088.

[29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[30] Jessica Pesantez-Narvaez, Montserrat Guillen, and Manuela Alcañiz. Predicting Motor Insurance Claims Using Telematics Data—XGBoost versus Logistic Regression. *Risks*, 7(2), 2019. https://doi.org/10.3390/risks7020070.

[31] Jimmy Ludeña-Choez and Ascensión Gallardo-Antolín. Feature extraction based on the high-pass filtering of audio signals for Acoustic Event Classification. *Computer Speech & Language*, 30(1):32–42, 2015. https://doi.org/10.1016/j.csl.2014.04.001.

[32] John Daintith and Edmund Wright. *A Dictionary of Computing*. Oxford University Press, 2008. https://www.oxfordreference.com/view/10.1093/acref/9780199234004.001.0001/acref-9780199234004.

[33] Jonas Sandsten, Petter Weibring, Hans Edner, and Sune Svanberg. Real-time gas-correlation imaging employing thermal background radiation. *Optics Express Journal*, 6(4):92–103, 2000. https://doi.org/10.1364/OE.6.000092.

[34] V. R. Joseph. Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2022. https://doi.org/10.1002/sam.11583.

[35] Joseph V. Roshan. Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2022. https://onlinelibrary.wiley.com/doi/10.1002/sam.11583.

[36] Jürgen Herre, E. Allamanche, and O. Hellmuth. Robust matching of audio signals using spectral flatness features. In *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575)*, pages 127–130, 2001. https://doi.org/10.1109/ASPAA.2001.969559.

[37] Kaori Suefusa, Tomoya Nishida, Harsh Purohit, Ryo Tanabe, Takashi Endo, and Yohei Kawaguchi. Anomalous Sound Detection Based on Interpolation Deep Neural Network. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 271–275, 2020. https://doi.org/10.1109/ICASSP40776.2020.9054344.

[38] Kotsiantis, Sotiris., Dimitris Kanellopoulos, and Panayiotis Pintelas. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30:25–36, 2005. https://www.researchgate.net/publication/228084509_Handling_imbalanced_datasets_A_review.

[39] Laurentius K. P. Saputra, Hanung A. Nugroho, and Meirista Wulandari. Feature extraction and classification of heart sound based on autoregressive power spectral density (AR-PSD). In *The 1st International Conference on Information Technology, Computer, and Electrical Engineering*, pages 139–143, 2014. https://doi.org/10.1109/ICITACEE.2014.7065730.

[40] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. https://doi.org/10.48550/arXiv.1706.09516.

[41] Liyan Xiong, Cheng Wang, Xiaohui Huang, and Hui Zeng. An Entropy Regularization k-Means Algorithm with a New Measure of between-Cluster Distance in Subspace Clustering. *Entropy*, 21(7):683, 2019. http://dx.doi.org/10.3390/e21070683.

[42] Malay K. Pakhira. A Linear Time-Complexity k-Means Algorithm Using Cluster Shifting. In *2014 International Conference on Computational Intelligence and Communication Networks*, pages 1047–1051, 2014. https://doi.org/10.1109/CICN.2014.220.

[43] Manabu Kotani, Masanori Katsura, and Seiichi Ozawa. Detection of gas leakage sound using modular neural networks for unknown environments. *Neurocomputing*, 62:427–440, 2004. https://doi.org/10.1016/j.neucom.2004.06.002.

[44] Martin Novotny and Milos Sedlacek. RMS value measurement based on classical and modified digital signal processing algorithms. *Measurement: Innovative Design and Paradigms in Instrumentation and Measurements*, 41(3):236–250, 2008. https://doi.org/10.1016/j.measurement.2006.11.011.

[45] Michael L. Waskom. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. https://doi.org/10.21105/joss.03021.

[46] Mohammad Hossin and M.N. Sulaiman. A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, 5(2):1–11, 2015. http://dx.doi.org/10.5121/ijdkp.2015.5201.

[47] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *arXiv*, 2020. https://doi.org/10.48550/arXiv.2003.06505.

[48] Nicolas Pielawski and Carolina Wählby. Introducing Hann windows for reducing edge-effects in patch-based image segmentation. *PLOS ONE*, 15(3):1–11, 2020. https://doi.org/10.1371/journal.pone.0229839.

[49] Nikhil S. Ketkar. *Deep Learning with Python: A Hands-on Introduction*. Apress Berkeley, CA, 2017. https://doi.org/10.1007/978-1-4842-2766-4.

[50] NumPy Developers. Discrete Fourier Transform (numpy.fft). *NumPy API Reference Guide*, 2022. https://numpy.org/doc/stable/reference/routines.fft.html.

[51] P. Duhamel and M. Vetterli. Fast Fourier Transforms: A tutorial review and a state of the art. *Signal Processing*, 19(4):259–299, 1990. https://doi.org/10.1016/0165-1684(90)90158-U.

[52] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan Jvan der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C.J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. https://doi.org/10.1038/s41592-019-0686-2.

[53] Pettersen, Marianne. "Research review on Acoustic Anomaly Detection Techniques for Leakage Detection on Industrial Plants". Unpublished thesis. Available at: https://drive.google.com/file/d/18Lg6pjcfyp8P8ytqineNmfJXyEini24Z/view?usp=sharing, 2021.

[54] Prajoy Podder, Md. Mehedi Hasan, Md. Rafiqul Islam, and Mursalin Sayeed. Design and Implementation of Butterworth, Chebyshev-I and Elliptic Filter for Speech Signal Analysis, 2020. https://arxiv.org/ftp/arxiv/papers/2002/2002.03130.pdf.

[55] Preethi Devan and Neelu Khare. An efficient XGBoost–DNN-based classification model for network intrusion detection system. *Neural Computing and Applications*, 32(16):12499–12514, 2020. https://doi.org/10.1007/s00521-020-04708-x.

[56] Rikke Gade and Thomas B. Moeslund. Thermal cameras and applications: a survey. *Machine Vision and Applications*, 25(1):245–262, 2013. https://doi.org/10.1007/s00138-013-0570-5.

[57] Roy M. Howard. The Power Spectral Density. In *Principles of Random Signal Analysis and Low Noise Design: The Power Spectral Density and Its Applications*, chapter 3, pages 59–91. John Wiley & Sons Ltd, Hoboken, NY, 2002. https://doi.org/10.1002/0471439207.ch3.

[58] Rui Xiao, Qunfang Hu, and Jie Li. Leak detection of gas pipelines using acoustic signals based on wavelet transform and Support Vector Machine. *Measurement*, 146:479–489, 2019. https://doi.org/10.1016/j.measurement.2019.06.050.

[59] S. Gopal Krishna Patro and Kishore Kumar Sahu. Normalization: A Preprocessing Stage. *arXiv*, 2015. https://doi.org/10.48550/arxiv.1503.06462.

[60] Scikit-learn Developers. 2.3. Clustering. *Scikit-learn User Guide*, 2022. https://scikit-learn.org/stable/modules/clustering.html.

[61] Shaikh Akib Shahriyar, M. A. H. Akhand, N. Siddique, and T. Shimamura. Speech Enhancement Using Convolutional Denoising Autoencoder. In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pages 1–5, 2019. https://doi.org/10.1109/ECACE.2019.8679106.

[62] Siau, Keng., Yingrui Xi, and Cui Zou. Industry 4.0- Challenges and Opportunities in Different Countries. *Cutter IT Journal*, 32(6):6–14, 2019. https://www.researchgate.net/publication/334919450_Industry_40-_Challenges_and_Opportunities_in_Different_Countries.

[63] Susmita Ray. A Quick Review of Machine Learning Algorithms. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pages 35–39, 2019. https://doi.org/10.1109/COMITCon.2019.8862451.

[64] Syahidah Izza Rufaida, Jenq-Shiou Leu, Kuan-Wu Su, Azril Haniz, and Jun-Ichi Takada. Construction of an indoor radio environment map using gradient boosting decision tree. *Wireless Networks*, 26(8):6215–6236, 2020. https://doi.org/10.1007/s11276-020-02428-7.

[65] T. Shindoi, T. Hirai, K. Takashima, and T. Usami. Plant equipment diagnosis by sound processing. In *IECON'99. Conference Proceedings. 25th Annual Conference of the IEEE Industrial Electronics Society (Cat. No.99CH37029)*, volume 2, pages 1020–1026, 1999. https://doi.org/10.1109/IECON.1999.816552.

[66] Taha Berkay Duman, Baris Bayram, and Gökhan Ince. Acoustic Anomaly Detection Using Convolutional Autoencoders in Industrial Processes. In Francisco Martinez Alvarez, Alicia Troncoso Lora, José Antônio Sáez Muñoz, Héctor Quintián, and Emilio Corchado, editors, *14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019)*, pages 432–442, Cham, 2020. Springer International Publishing. https://doi.org/10.1007/978-3-030-20055-8_41.

[67] The SciPy community. Signal Processing (scipy.signal). *SciPy API Reference Guide*, 2022. https://docs.scipy.org/doc/scipy/reference/signal.html#module-scipy.signal.

[68] Theodoros Giannakopoulos and Aggelos Pikrakis. *Introduction to Audio Analysis - A MATLAB Approach*. Academic Press, 2014. https://doi.org/10.1016/C2012-0-03524-7.

[69] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team. Jupyter Notebooks – a publishing format for reproducible computational workflows. In Fernando Loizides and Birgit Scmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press, 2016. http://dx.doi.org/10.3233/978-1-61499-649-1-87.

[70] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. https://doi.org/10.1145%2F2939672.2939785.

[71] Tianshu Xu, Zhoumo Zeng, Xinjing Huang, Jian Li, and Hao Feng. Pipeline leak detection based on variational mode decomposition and support vector machine using

an interior spherical detector. *Process Safety and Environmental Protection*, 153:167–177, 2021. https://doi.org/10.1016/j.psep.2021.07.024.

[72] Tsekeris, Charalambos. Industry 4.0 and the digitalisation of society: Curse or cure? *Homo Virtualis*, 1(1):4–12, 2018. https://doi.org/10.12681/homvir.18622.

[73] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002. https://doi.org/10.1109/TSA.2002.800560.

[74] R. van der Vleuten and F. Bruekers. Lossless compression of binary audio signals. In *Proceedings DCC '98 Data Compression Conference (Cat. No.98TB100225)*, pages 579–, 1998. https://doi.org/10.1109/DCC.1998.672321.

[75] Vishal Sharma, Taksh, Kritarth Srivastav, Priyam, and Nihal Anwar Siddiqui. A Critical Study on Role of Sensor-Based Electronic System for Toxic Gas Identification in the Mining (Coal) Industry. In Singh, Rajesh., Sushabhan Choudhury, and Anita Gehlot, editors, *Intelligent Communication, Control and Devices. Advances in Intelligent Systems and Computing Journal (AISC) 624*, pages 1511–1521. Springer, Singapore, 2018. https://doi.org/10.1007/978-981-10-5903-2_157.

[76] Wazir Zada. Khan, Mohammed Y. Aalsalem, Wajeb Gharibi, and Quratulain Arshad. Oil and Gas monitoring using Wireless Sensor Networks: Requirements, issues and challenges. In *2016 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*, pages 31–35, 2016. https://doi.org/10.1109/ICRAMET.2016.7849577.

[77] Wes McKinney. Data Structures for Statistical Computing in Python. In van der Walt, Stéfan. and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. http://dx.doi.org/10.25080/Majora-92bf1922-00a.

[78] Xubao Zhang. Benefits and Limitations of Common Directional Microphones in Real-World Sounds. *Clinical Medicine Research*, 7(5):103–118, 2018. http://dx.doi.org/10.11648/j.cmr.20180705.12.

[79] Ying Jia, Bingkun Gao, Chunlei Jiang, and Sihan Chen. Leak Diagnosis of Gas Transport Pipelines based on Hilbert-Huang transform. In *Proceedings of 2012 International Conference on Measurement, Information and Control*, volume 2, pages 614–617, 2012. https://doi.org/10.1109/MIC.2012.6273368.

[80] Yukinori Nagaya and Michio Murase. Detection of cavitation with directional microphones placed outside piping. *Nuclear Engineering and Design*, 249:140–145, 2012. The 8th International Topical Meeting on Nuclear Thermal-Hydraulics, Operation and Safety (NUTHOS-8), https://doi.org/10.1016/j.nucengdes.2011.08.045.

[81] Zhang Shuqing, Gao Tianye, Han Xu, Jia Jian, and Wang Zhongdong. Research on Pipeline Leak Detection Based on Hilbert-Huang Transform. In *2009 International Conference on Energy and Environment Technology 3*, pages 500–503, 2009. https://doi.org/10.1109/ICEET.2009.587.

[82] Zhaomin Chen, Chai Kiat Yeo, Bu Sung Lee, and Chiew Tong Lau. Autoencoder-based network anomaly detection. In *2018 Wireless Telecommunications Symposium (WTS)*, pages 1–5, 2018. https://doi.org/10.1109/WTS.2018.8363930.

[83] Zhejian Chi, Ying Li, and Cheng Chen. Deep Convolutional Neural Network Combined with Concatenated Spectrogram for Environmental Sound Classification. In *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, pages 251–254, 2019. https://doi.org/10.1109/ICCSNT47585.2019.8962462.

[84] Ľudmila Pavlikova, Beata Hricová, and Ervín Lumnitzer. Acoustic Camera as a Tool for Identifying Machinery and Equipment Failures. *Advances in Science and Technology Research Journal*, 12(1):322–328, 2018. http://dx.doi.org/10.12913/22998624/87110.

Appendix

A Acronyms

AI - Artificial Intelligence

ML - Machine Learning

NN - Neural Network

DNN - Deep Neural Network

ADD - Acoustic Anomaly Detection

PSD - Power Spectral Density

RMS - Root Mean Square

MFCC - Mel Frequency Cepstral Coefficients

FFT - Fast Fourier Transform

DFT - Discrete Fourier Transform

GB - Gradient Boosting

FIR - Finite Impulse Response

IIR - Infinite Impulse Response

P - Positive samples

N - Negative samples

TP - True Positive samples

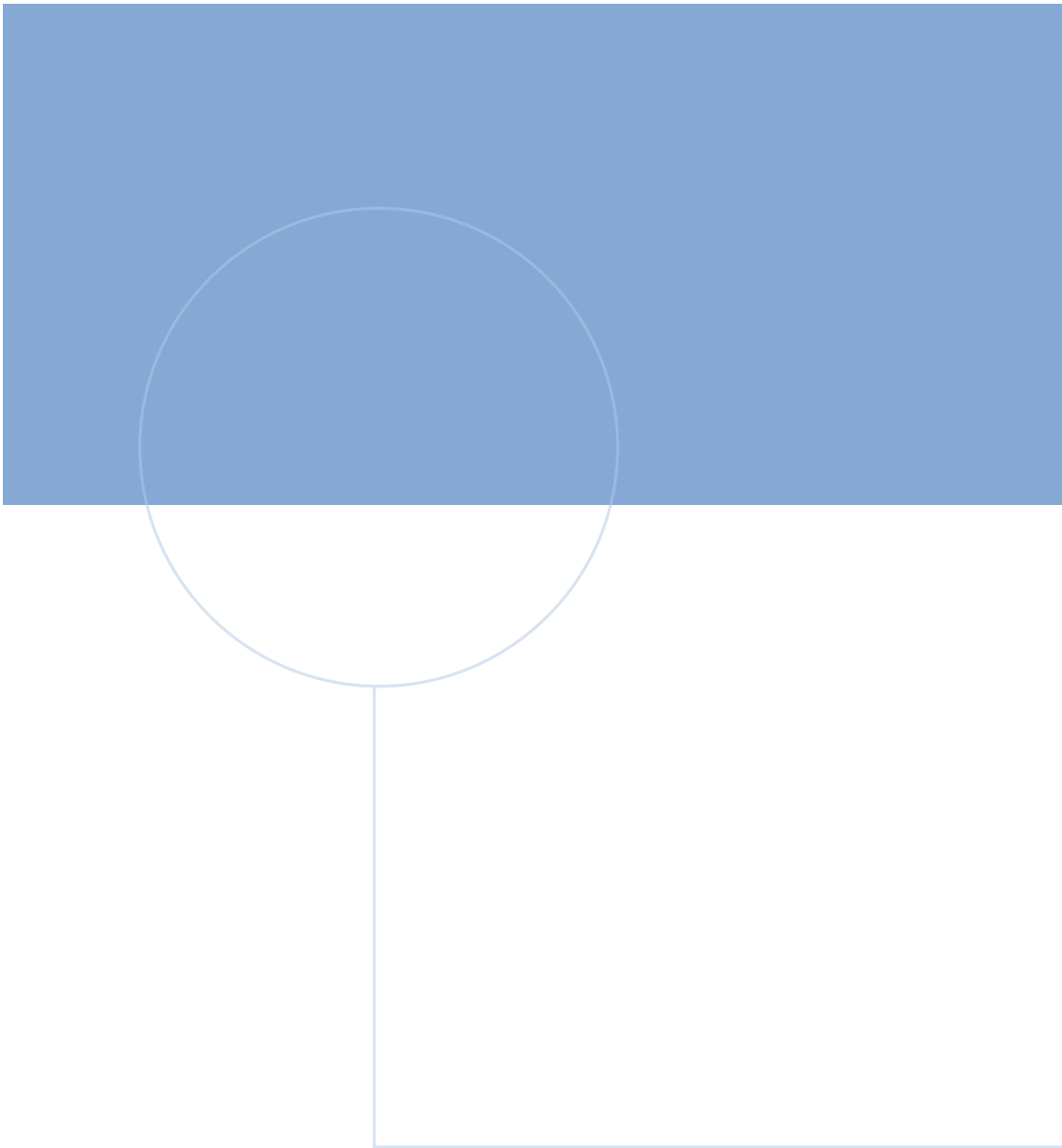
TN - True Negative samples

FP - False Positive samples

FN - False Negative samples

B Programming Project

The experimental study was initialized inside Jupyter Notebook, and the programming tasks were separated into different ipynb-files. All files are available on GitHub through the following link: https://github.com/thewildling/MasterThesis. In order to ensure reproducibility, a Read.me file is included which explains how and in which order to run the codes.



NTNU

Norwegian University of
Science and Technology