Machine Learning for Acoustic Anomaly Detection in Electric Motors

ALEX RENAUDIN

Master's Program Thesis OENG1090

RMIT School of Engineering, STEM College

22 OCTOBER 2023

Special thanks to

Dr. Mohammad Fard,

Sitthichart Tohmuang,

Parin Sanpetchnarong, and

JF Renaudin

for their continued support throughout this project.

Table of Contents

List of Figures

List of Tables

1. Executive Summary

This report outlines a year-long investigation into a machine learning (ML) solution to the problem of acoustic anomaly detection (AAD) in an electric saw. An automated solution was successfully developed in the form of a support vector machine (SVM) model trained on a custom dataset of saw audio recordings.

The model proved successful in classifying both satisfactory and anomalous saw cuts with a mean accuracy of 97.98%. This performance was assisted by a custom feature extraction program which characterises the audio features using the Mel frequency cepstrum coefficient (MFCC) algorithm. Deployment of the model onto a micro-controller unit (MCU) was attempted with the aim of performing active AAD in a factory setting. While deployment was ultimately unsuccessful, it remains a compelling avenue for future work.

Included in this report are details of the project design goals, both of which were supported by a thorough review of current ML literature in AAD. A project plan outlines the key stages of work, followed by the inclusion of all project results including the custom feature extraction and SVM programs. Discussion and conclusions follow the results, highlighting trends and analysing the issues encountered.

Ultimately, the proposed solution demonstrates promise as both a safety feature and maintenance tool for drop saws, showcasing the ability to identify anomalous and potentially dangerous saw behaviour through acoustic signals.

2. Introduction

Recent technological advancements in artificial intelligence (AI), specifically machine learning (ML), capabilities have led to an increase in applications across a range of fields (Bianco et al., 2019). A notable development has been in the problem of pattern recognition to perform acoustic anomaly detection (AAD). This area of research focuses on using characterising and processing audio signals using machine learning techniques with the aim of classifying anomalous system behaviour.

Figure 1: Makita LS1219 slide compound saw.

This paper outlines a project to develop a ML model to detect faults in an electric drop saw. It aimed to use the acoustic signal of the saw as the input for the ML model, with several data processing methods being employed to characterise the audio, such as the use of the Mel frequency cepstrum coefficient (MFCC).

It was aimed to develop a compact model to be deployed onto a micro-controller unit (MCU), thus allowing it to be compatible in Industry 4.0 settings. This gives it the ability to smoothly integrate into larger industrial environments by offering real-time feedback on saw behaviour.

This paper is structured as follows: A detailed outline of the problem of anomalous saw use is provided in Section 2 to highlight the need for an automated solution. Section 3 covers a literature review of existing work on ML for AAD, followed by an in-depth discussion of any research gaps in Section 4. These form the foundation for the project goals as outlined in Section 5 and the project plan in Section 6. Finally, results of all project work are provided in Section 7, and a discussion and conclusions are outlined in Section 8 and Section 9 respectively.

2.1. Problem Description

Ensuring operator safety during the operation of power saws is a crucial, yet often difficult, task. Improper use of the equipment can lead to injury or even death in the worst case, or significant damage to the equipment in minor cases. Typically, faults in power tools are first diagnosed aurally by the operator (Abbasi et al., 2021). However, this method can pose significant risk as it relies on operator experience and judgement and does not account for the possibility of human error.

Figure 2: Depiction of the two classes: satisfactory and anomalous.

A solution that offers real-time, reliable diagnosis of faults in electric saw use would be very beneficial, as it offers immediate feedback to the operator if the machinery is damaged or being misused. This acts as an additional safety measure in preventing further improper use or use of faulty equipment, ultimately improving worker safety.

A ML-based system offers a solution to this problem if it can perform real-time, accurate, and automated identification of saw anomalies. There is an audible difference between anomalous and satisfactory saw cuts. Therefore, a custom dataset can be collected and labelled, and ultimately used to train a ML to identify the presence of anomalous behaviour. Such a solution could be deployed at a relatively low cost across factory environments to improve safety and efficiency in manufacturing.

2.2. Key Terms

An alphabetised glossary of terms used throughout this report is detailed in Table 1.

Table 1: Glossary of key terms.

Acronym	Meaning
AAD	Acoustic Anomaly Detection
AE	Auto Encoder
AI	Artificial Intelligence
CAE	Convolutional Auto Encoder
CLAHE	Contrast-Limited Adaptive Histogram Equalisation
CSV	Comma Separated Value
DCAE	Deep Convolutional Auto Encoder
DT	Decision Tree
KNN	K-Nearest Neighbour
LSTM	Long-Short-Term Memory
MCU	Micro-Processor Unit
MFCC	Mel Frequency Cepstrum Coefficient
ML	Machine Learning
NN	Neural Network
OCSVM	One-Class Support Vector Machine
PCA	Principal Component Analysis
RBF	Radial Basis Function
RMS	Root Mean Squared
SRAM	Static Random-Access Memories
STFT	Short-Term Fourier Transform
SVC	Support Vector Classifier
SVM	Support Vector Machine
VMD	Variational Mode Decomposition
WAV	Waveform Audio File Format

3. Literature Review

A review of recent work in the field of ML for AAD was undertaken to develop the goals and inform the planning stage of the project (Renaudin, 2023). There has been a considerable amount of work done on AAD following recent developments in ML algorithms. A wide range of research in this field is due to the breadth of ML models and range of applications of AAD. Consequently, these works can be categorised by their focus on one of the following three goals, and as such the literature review will explore each of these groups individually.

1. Development of a novel ML algorithm in the context of AAD
2. Implementation of improved feature extraction technique for audio signals
3. Novel application of existing AAD techniques.

3.1. ML Algorithms

ML models for AAD encompass both supervised and unsupervised learning, each offering unique advantages and disadvantages. Traditionally, supervised learning models have been a favoured approach for building models, involving a predictive mapping based on labelled input-output pairs (Bianco et al., 2019).

Various types and sub-types of supervised learning exist. In the context of AAD, early works focused on simpler models such as the K-nearest neighbour (KNN) and decision trees (Grandhi & Prakash, 2021). However, there has been a shift towards more sophisticated methods such as support vector machines (SVM) and neural networks (NN) due to their enhanced capabilities.

Noteworthy examples of AAD using supervised learning include the work by Yan et al. (2022), whose use of a long-short-term memory (LSTM) model to detect faults in diesel engines achieved an overall diagnostic accuracy of 97%.

Similarly, Grandhi and Prakash (2021) proposed a smartphone application that performs AAD on electric motors. The author measured the performance of three different supervised learning models: the decision tree (DT), KNN and SVM. The KNN model was found to be the best model with 90% accuracy, however the alternatives performed within 10% of this score.

Nevertheless, there are disadvantages to a supervised learning approach. Son et al. (2022) raises the issue of collecting representative datasets, since anomalies are extremely rare in most cases. For this reason, the author suggests the use of unsupervised algorithms such as the convolutional autoencoder (CAE) or the one-class support vector machine (OCSVM).

Unsupervised models operate by learning the characteristics of one class of acoustic signal, thus being able to identify faulty signals as simply those not belonging to the identified class. This technique allowed the authors to distinguish abnormal signals from normal signals with improved accuracy over conventional methods, despite their dataset containing only 9 out of 2871 abnormal signals.

Further, unsupervised learning models offer the advantage of eliminating the costly process of manual data classification, which is labour-intensive and prone to human error (Coelho et al., 2022). Advancements have been made in NN capabilities, including lower computational requirements, making models such as auto encoders (AE) and its sub-types a popular choice for fast, deployable learning models.

3.2. Feature Extraction Techniques

A key factor of successful ML algorithms for sound recognition is in the processing of the acoustic signal before input into the model. ML models typically work on classifying visual data rather than audio data directly (Cheng & Kuo, 2022), and thus appropriate feature extraction techniques are required to represent the data visually. Further, the feature extraction process aims to eliminate unnecessary features in the data, ultimately resulting in higher accuracy and significantly lower modelling costs (Ramírez & Flores, 2020).

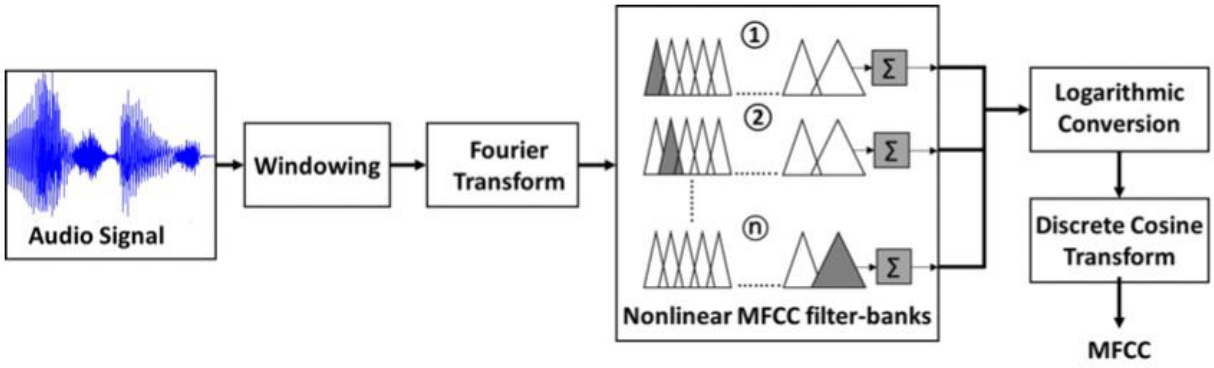


Figure 3: MFCC feature extraction procedure (Abeysinghe et al., 2021).

The most prevalent feature extraction technique is the use of the Mel spectrogram, and the associated Mel frequency cepstrum coefficient (MFCC), which have been used across numerous acoustic applications from speech recognition to noise classification (Son et al., 2022). The MFCC is an information-dense feature that can represent the 'voiceprint' of an acoustic signal (Yan et al., 2022), and the Mel spectrum is the visual representation of this information, showing the time, frequency and intensity features.

The process of obtaining MFCC values from acoustic signals is shown in Figure 3. A key feature of this process is the non-linear filter banks, which are designed to

emulate the human auditory system by emphasising lower frequencies (Abeysinghe et al., 2021).

Novel techniques that combine the use of MFCC with other feature extraction techniques have seen significant success. Son et al. (2022) combined contrast-limited adaptive histogram equalisation (CLAHE) techniques to the visual Mel-spectrum to enhance contrast and reduce noise. Additionally, the author applied principal component analysis (PCA) to reduce the dimension of the MFCC features, ultimately resulting in greatly improved accuracy when used with a OCSVM learning model.

Similarly, the use of a Kalman filter (Suman et al., 2022), and variational mode decomposition (VMD) (Yan et al., 2022) have both been shown to reduce noise from the input signal before MFCC processing. A combination of MFCC techniques with signal enhancement and/or additional feature extraction techniques is increasingly popular due to the improvement in model accuracy.

Alternatively, Glowacz et al. (2021) propose a feature extraction method that does not use MFCC techniques, instead relying on the use of ‘shortened method of frequencies selection nearest frequency components’. This technique shares the MFCC’s use of the FFT spectrum but differs by ultimately computing feature vectors based on the relationship between a signal’s adjacent frequency components. This approach resulted in high accuracy, however, is more complex and less established than its MFCC counterpart.

3.3. AAD Implementation

Consideration to the application and deployment of AAD solutions was noted to be lacking. Most works place focus on maximising model accuracy through improved feature extraction techniques or advancement in ML learning models (Bianco et al., 2019). Limitations in available datasets and costly hardware have discouraged the investigation of novel applications, as they typically require manual data collection.

There has been significant work in AAD on combustion engines in various forms (Abeysinghe et al., 2021; Suman et al., 2022; Wang et al., 2020; Wu et al., 2022), and a small but growing interest in AAD for electric motors (Altinors et al., 2021; Glowacz et al., 2021; Grandhi & Prakash, 2021; Son et al., 2022). However, most of these approaches utilise computers or a smartphone to ultimately run the ML model, which is not feasible as a final system solution.

Figure 4: Intel Core i5-7600K CPU used by Abbasi et al. (2021) (Source: Intel).

A notable exception is the work by Abbasi et al. (2021) in creating a compact model that can perform AAD on industrial fans, valves, sliders and more. The author proposes a series of deep convolutional auto encoder (DCAE) models, which are deployed on an Intel Core i5 and an ARM Cortex A72, both of which are chosen due

to their popularity in industry. Their final models delivered high accuracy, despite having as few as 686 parameters and being as small as 2.7 KB, making them small enough to fit on the static random memory (SRAM) of many microcontrollers.

The construction of these models was facilitated by the novel use of a ‘machine-driven design exploration strategy’: the use of generative synthesis to optimise model architecture to desired design goals. This technique is promising as it eliminates the need for manual optimisation of the ML algorithm.

4. Research Gaps

Following the literature review of current ML approaches to AAD, some key research gaps were identified (Renaudin, 2023). These were treated as areas of opportunity to guide the design goals of this research project.

4.1. AAD in Saw Motor Strain

Most work was noted to focus on combustion engines in automotive applications. Further, work on electric motors, such as that by Grandhi and Prakash (2021) and Glowacz et al. (2021), typically focuses on explicit faults in the machinery, such as damaged bearings, rather than general misuse.

There has been no investigation into identifying sub-optimal performance of a motor, such as the case of motor strain associated with incorrect use. This is notable, as such misuse can ultimately cause damage to the machine, posing safety risks to operators. Further, specific work on drop saws being used on timber has not been observed.

4.2. Open-Source Motor Dataset

The bias towards research on combustion engines can be observed through the availability of datasets. A survey through the internet and literature revealed there is very little open-source acoustic data of electric motor use. Since there is such a range of applications of electric motors, collecting such data is usually a task-specific endeavour. Thus, the necessary dataset to train the ML model is lacking and would need to be acquired experimentally.

4.3. Deployment of Compact Models

Applications of ML for AAD remain focused on developing models with high accuracy and complexity, with less attention being placed on the final form and application. While some models have been developed to be run on a smartphone (Grandhi & Prakash, 2021), there is a gap in the work on purpose-built devices, which are integral to enabling Industry 4.0 advancements. Suman et al. (2022) propose an MCU-based approach to perform AAD on combustion engines, however they do not employ any ML techniques. Notable work is that of Abbasi et al. (2021), who propose a MCU-based model facilitated by the use of their machine-driven design exploration strategy, and suggest future work in further optimisation of such models.

5. Aim & Objectives

This project was guided by a primary and secondary objective:

- Primary objective: Develop a ML model to perform AAD on an electric saw used to cut timber.
- Secondary objective: Propose and build an MCU-based solution that can run the model in real-time.

The primary objective puts focus on identifying motor strain from misuse rather than outright mechanical faults. This stems from the findings of the literature review in which a bias for fault detection rather than misuse was noted. As implied by the primary objective, a novel dataset of saw audio is required. Thus, it was aimed to produce an appropriately large collection of audio recordings of the saw in operation.

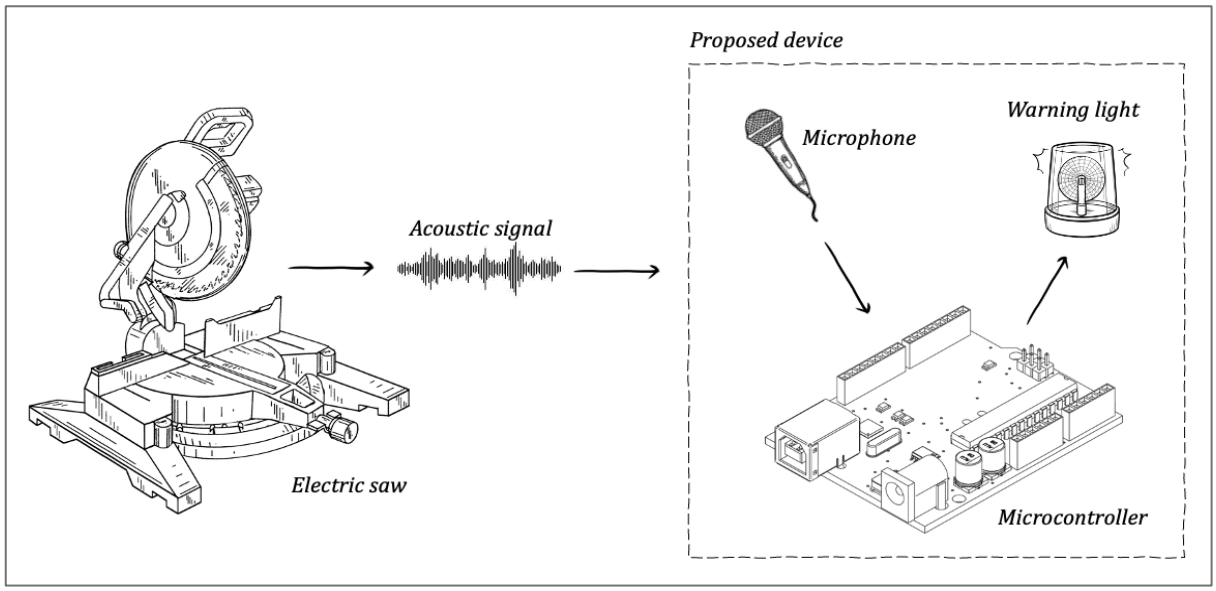


Figure 5: Proposed system with MCU & warning light.

The secondary objective for a deployable device stems from the work by Abbasi et al. (2021), who aimed to design for low-memory and low-latency devices. Where their work was restrained within the limitations of the Intel Core i5 and ARM Cortex A72 processors, this project aimed to investigate alternative solutions for deployment, notably the use of user-friendly MCU devices such as Arduino or Raspberry Pi units.

It was noted that the secondary objective may not be achievable, as deployment onto a physical device may not be possible due to budget or time constraints. Thus, this objective is classed as a stretch goal to be pursued following successful completion of primary objective work.

Regardless, Figure 5 shows a representation of such an implementation. The system includes a microphone to record acoustic signals, a MCU device to run the ML model in real-time, and a warning light to send immediate feedback to the saw operator if the model detects anomalies in the acoustic signal.

5.1. Research Questions

The key research and design questions that this project aimed to answer are:

- What is the most accurate ML model to perform AAD of electric saws?
- What model is the most suitable for in-field deployment?
- How can the acoustic features of the drop saw samples be optimally extracted?
- What is the optimal deployable solution that can perform AAD on electric saws?

5.2. Expected Outcomes

Aligned with the primary project objective, the main expected outcome was to construct a compact and highly accurate ML model for AAD of electric saws. This involved training and testing the model on an experimentally gathered dataset of audio recordings and employing a suitable feature extraction process.

This expectation includes a targeted exploration of ML model architectures with the aim of prioritising a balance between precision and compactness. Additionally, the investigation extends to refining feature extraction techniques, complementing the foundational use of the MFCC method.

The secondary objective had the expected outcome of developing a deployable MCU-based device. Although it was noted to be a stretch goal, expectations included the investigation of a suitable MCU device, appropriate microphone & peripheral components, and necessary techniques to run the deployed, optimised ML model.

6. Project Outline

This section lays out the project plan, detailing key steps, necessary equipment, proposed timeline, and the experimental setup. The plan, derived from the project proposal, charts the course of action essential for achieving the project aims as outlined in Section 5. Seven key stages have been identified for this project:

1. Literature review: An examination of current approaches in ML, feature extraction, and AAD techniques, focusing on literature from the last three years to ensure relevancy and currency of research.

2. Research proposal: A comprehensive paper documenting the literature review, research gaps, and a detailed plan for the project, including timeline, budgeting, risk analysis, and more. This proposal covered the content from Section 2 to Section 4 of this report.

3. Data acquisition: Acoustic data is collected from a local shed factory with the owner's permission. A microphone is placed at a safe, suitable distance from one of the drop saws, recording raw acoustic data during work hours. Review and optimisation of the microphone setup are crucial, as highlighted by Glowacz et al. (2021).

4. Feature extraction: Data from the acquisition phase is prepared for the feature extraction process. This involves developing a feature extraction program based on the MFCC algorithm, leveraging findings from the literature review to create a new algorithm. Testing and iteration of the algorithm can occur concurrently with data collection due to the level of automation in the data collection process.

5. Model development: Following the prototype feature extraction algorithm, investigation into a suitable ML architecture is conducted. The findings of the literature review are explored, leading to the development of a prototype ML model. Development of the ML model and feature extraction algorithm occurs concurrently, including regular iteration for improved performance and efficiency.

6. Model testing: The ML model is tested using a portion of the collected dataset, evaluating its performance against the design goals outlined in Section 5. Further tuning and iteration of both the ML model and feature extraction algorithm may be required, with the process being repeated until satisfactory performance is achieved.

7. Results & reporting: Results from model testing are reported, and conclusions are drawn on the project's success in achieving the design goals. Reporting aligns with RMIT assessment guidelines.

6.1. Resources & Budget – Data Collection

This section details the resources needed to carry out the data collection phase of this project. Table 2 outlines the necessary equipment with details regarding their purpose. Note that all equipment used was already available without any cost.

Table 2: List of equipment required for data collection.

Item	Purpose	Status	Price
MacBook Air (M1, 2020)	Computing/modelling	Owned	-
Sony ICD UX-200F recorder	Data collection	Owned	-
Makita LS1219 slide compound saw	Data collection	Available	-

A Makita LS1219 slide compound saw was used as the audio source in the data collection stage. It is a circular saw with a 305 mm diameter blade, used typically to cut timber for the construction of timber sheds. Testing of the model is also performed using audio from this saw.

Figure 6: Makita LS1219 saw product information close-up.

A 2020 MacBook Air with 500 GB of SSD was used for most computing tasks. It has access to software such as MATLAB, Audacity, Ableton and Python which were used throughout this project.

Figure 7: Sony ICD UX-200F recorder (Source: Sony).

Data collection was done with a Sony ICD UX-200F recorder. This is a conventional audio recording device that was selected for its simplicity. It is a suitable choice as it can be left to record audio for long periods of time, it accepts an external microphone input, and can store up to 22 hours of audio data in standard operation mode. A summary of its specifications is shown in Table 3.

Table 3: Sony ICD UX-200F technical specifications (Source: Sony, 2009)

Feature	Value
File format	MP3, WMA, AAC
Built-in storage	2 GB
Bit-depth	32 kbps – 320 kbps
Frequency range	40 Hz – 20,000 Hz
External microphone input	Yes
Recording time	22 hours (standard mode)

The inline microphone of the Apple EarPods was initially proposed to be used for all audio recording in this project. This is due to it being a cheap and readily available component with good technical performance. However, it was not easily compatible with the Sony ICD UX-200F recorder, and thus was replaced by the recorder's built-in microphone.

The audio profile of the two microphones were compared with an A-B comparison and were found to have very similar characteristics. Thus, it is likely that future applications could substitute the Apple EarPods microphone with minimal loss in model accuracy.

6.2. Resources & Budget – Model Deployment

This section details the resources required to build and deploy the proposed MCU-based solution, illustrated in Figure 5. It is noted that deployment never materialised due to issues in accessing the allocated project funds. Thus, the stretch goal of deployment was not possible, however efforts were still made to propose the components for such a solution. This is done to serve as a guide for future work.

Table 4: Proposed BOM for MCU deployment

Item	Purpose	Quantity	Price
Arduino Nano 33 BLE Sense Rev2	ML deployment	2	$76.56
Extruded Aluminium Enclosure	Protective case	1	$13.73
Silicon Sleeve for Arduino Nano 33	Dust protection	1	$17.60
		Total Cost	$184.45

The proposed materials for the MCU-based solution are outlined in the bill of materials (BOM) in Table 4. This includes an Arduino MCU along with a protective case and dust sleeve. The warning light is not included as it is to be initially prototyped with a small, inexpensive LED, with further research to find a suitable replacement. The total cost of this BOM, including a spare MCU for redundancy purposes, is within the allocated project budget of $300 per student.

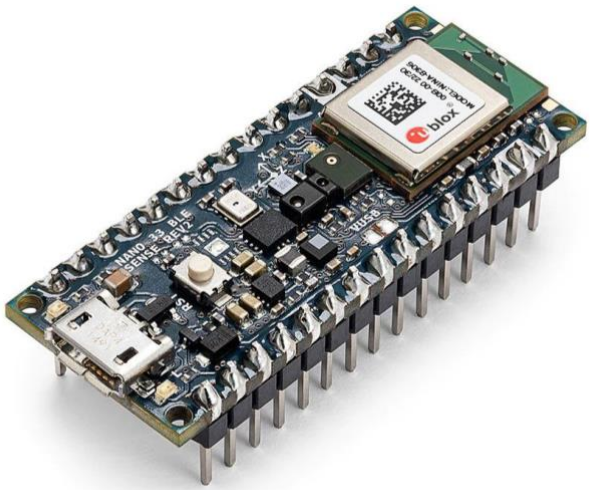


Figure 8: Arduino Nano 33 BLE Sense Rev2 MCU (Source: Arduino).

The Arduino Nano 33 BLE Sense Rev2 was selected for its convenience, low-cost, and built-in array of sensors. It is compatible with Python, as well as additional libraries to support ML models such as TinyML and TensorFlow. It also features a built-in omnidirectional digital microphone and is Bluetooth-enabled.

Additional investigation was done into the use of a Raspberry Pi 4 Model B as an alternative to the Arduino MCU. However, technical difficulties in setting up the device and installing the necessary Python libraries ultimately halted progress. This work is discussed in detail in Section 7.5 on MCU deployment results.

6.3. Timeline

The timeline for the project is shown in Figure 9. The project ran across semester one and semester two of 2023, excluding the midyear break from June 26 to July 16.

Significant overlap between stages is noted, owing to the interconnected and iterative nature of ML model development. For example, the process of data acquisition had a relatively low level of interaction and did not have a strict deadline for completion. Further, results from the model development stage would be fed back to influence changes in the feature extraction stage, and vice versa.

It is noted that the project largely ran on-schedule. The first semester of work resulted in most of the data collection phase complete, and a preliminary feature extraction program was developed. Early model development was attempted, however much of the model building and testing was completed in semester two. Project results are discussed in detail in Section 7.

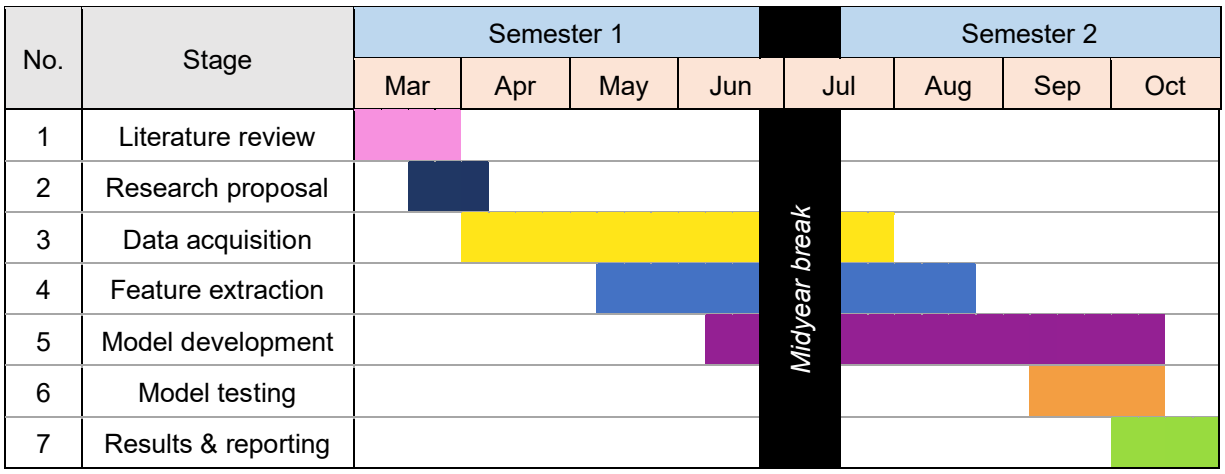


Figure 9: Project Gantt chart.

6.4. Data Collection Procedure

The data collection process was undertaken in a Melbourne shed factory, focusing on a Makita LS1219 saw employed for cutting pine timber. A custom recording rig, shown in Figure 11, was built to capture audio of the saw cuts. Subsequently, each cut undergoes acoustic and mathematical processing during the feature extraction process to shape the final dataset.

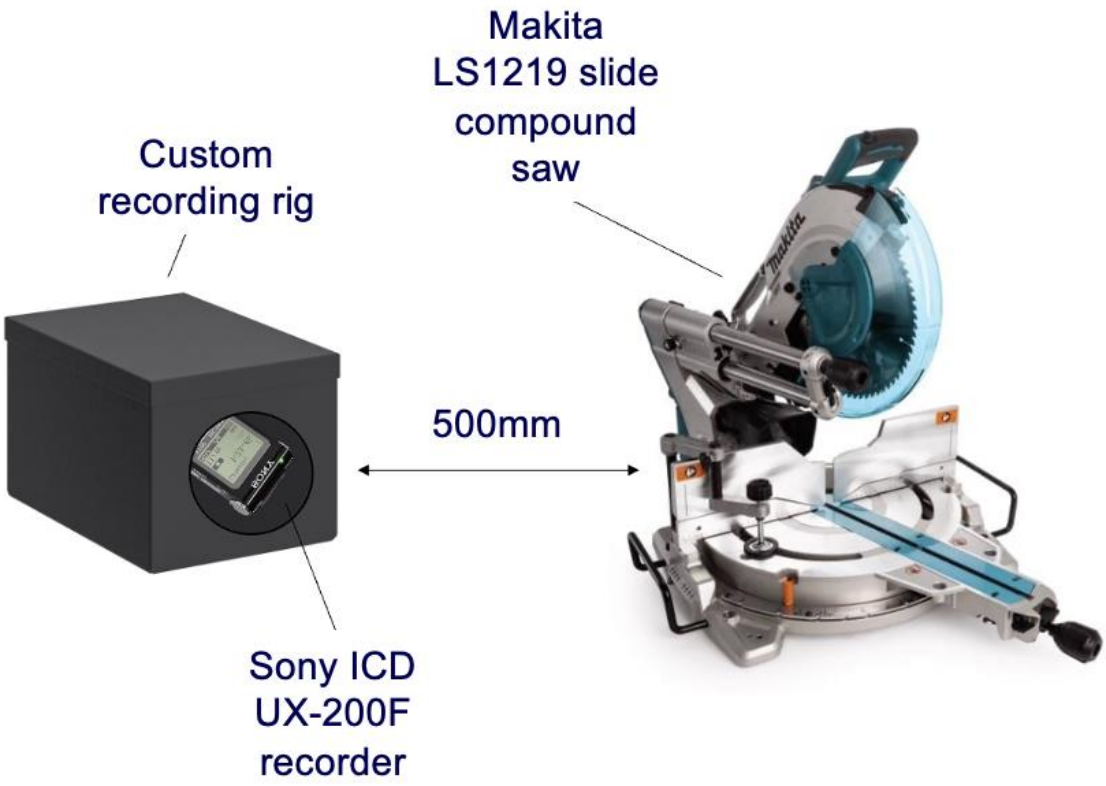


Figure 10: Simplified experimental setup for data collection.

The final experimental recording setup is depicted in Figure 10. Positioned approximately 500 mm away from the saw head, i.e., the primary sound-producing element of the saw, the recording rig captured the acoustic details. The Sony UX-200F, which allows for adjustable mic sensitivity, was set to the lowest sensitivity option to account for the saw's loudness and the recorder's proximity.

The recorder was activated and left to record and entire day of factory work in one continuous session. Audio recordings were regularly downloaded to a computer, ready to be processed in the next stage of the project.

Figure 11: Custom recording rig with and without pop filter.

During the recording process, various background noises, including the operation of other power tools, radio, music, and conversation, were present. However, owing to the low sensitivity of the microphone and the considerable loudness of the saw in operation, the background noises do not have a material impact on the final audio recordings.

The custom recording rig, shown in Figure 11, comprises the Sony ICD UX-200F recorder encased in a protective cardboard box. A cut-out on the front panel allows access to the device, and a pop filter is placed over the opening during operation. Traditionally used in vocal recording to prevent 'popping' distortions caused by high-energy sound waves, the pop filter is well-suited for this application by allowing sound to pass through uninterrupted, as well as preventing dust from entering the device.

Figure 12: Final recording set-up with saw and recording rig.

6.5. Pre-Processing & Feature Extraction Procedure

The process of data collection returns a large collection of audio that captures the saw in use. However, a few stages of pre-processing are required before it can be used as the dataset for a ML model. First, since the recorder captures a full workday's worth of audio recordings, manual work is required to separate the clips of each saw cut and save them as individual files. Next, significant care is taken to appropriately label the clips into the two classes, i.e., as either 'anomalous' or 'satisfactory'. This process is detailed in full Appendix A, as it was imperative to have a strict protocol to produce a high-quality dataset.

Completion of this pre-processing procedure results in a large collection of second-long saw audio recordings saved in a single folder, with appropriate suffixes to the filename to indicate the class. Audio is saved in Waveform Audio File Format (WAV), which is a lossless audio format ensuring no artifacts or data loss occur in the audio. This constitutes the audio dataset for the project; however, it is still not in a suitable format for a ML model. As outlined in the literature review in Section 3.2, the process of feature extraction is required to represent the audio data in a machine-friendly form.

The overarching aim of this process is to represent the audio mathematically, consequently reducing the file-size significantly, while retaining enough of the 'character' of the original audio file for a ML to successfully label its class. This work was planned to be done in the form of a Python program utilising the MFCC algorithm as the primary method of characterisation. Further discussion of this process is detailed in the feature extraction results in Section 7.2.

6.6. ML Model Procedure

Development of the ML model to perform AAD on the experimentally obtained dataset was subject to significant exploration into different model architectures and development platforms. There was no predetermined model architecture, nor was there a set procedure to build the model.

Overarching guidance was provided by findings of the literature review, in which a single-class model was favoured because it does not rely on a large proportion of anomalous samples. This is subject to review following completion of the dataset, for if there is a suitable mix of both classes then the use of two-class classification models can be considered.

7. Results

This section covers the results of the work undertaken for this project, spanning the phases of data acquisition, feature extraction, model development, and testing. Discussion around these results will be placed in Section 8.

7.1. Audio Dataset

The data collection stage of the project resulted in over 50 hours of raw audio from the factory. This was successfully processed using the protocol outlined in Appendix A, resulting in 513 individual audio clips, each one second long and labelled with its class.

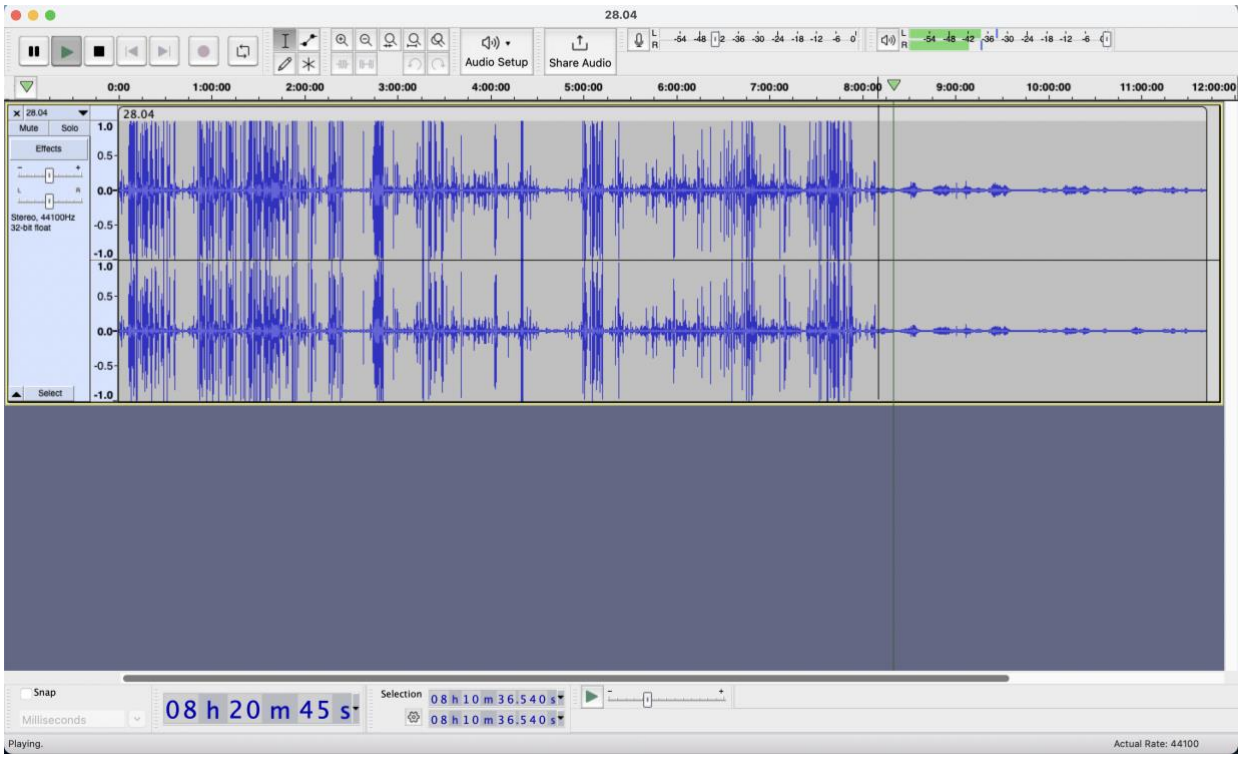


Figure 13: Use of Audacity to trim raw audio into hour-long segments.

In practice, most cuts made on the saw are satisfactory, and anomalous instances are exceptionally rare. Thus, several experimental attempts were undertaken to induce anomalous cuts. A series of bowed or otherwise irregular timber was cut, resulting in recordings of the saw motor straining or the blade catching on the misshapen timber. It is emphasised that these recordings were conducted safely by a professional outside of work hours and in line with the risk framework for this project (Appendix F).

Figure 14: Use of Ableton to isolate & crop audio to 1 second.

Ultimately, the dataset contained a strong representation of anomalous samples at 17.54%. While a more balanced split between the two classes is ideal, this split was deemed suitably large to be used in a two-class classification model as opposed to a single-class model as was explored in the literature review in Section 3.1.

Note that a proportion of the audio dataset was later removed due to being mislabelled. This was done to optimise the final ML performance and is discussed in depth in Section 8.3.

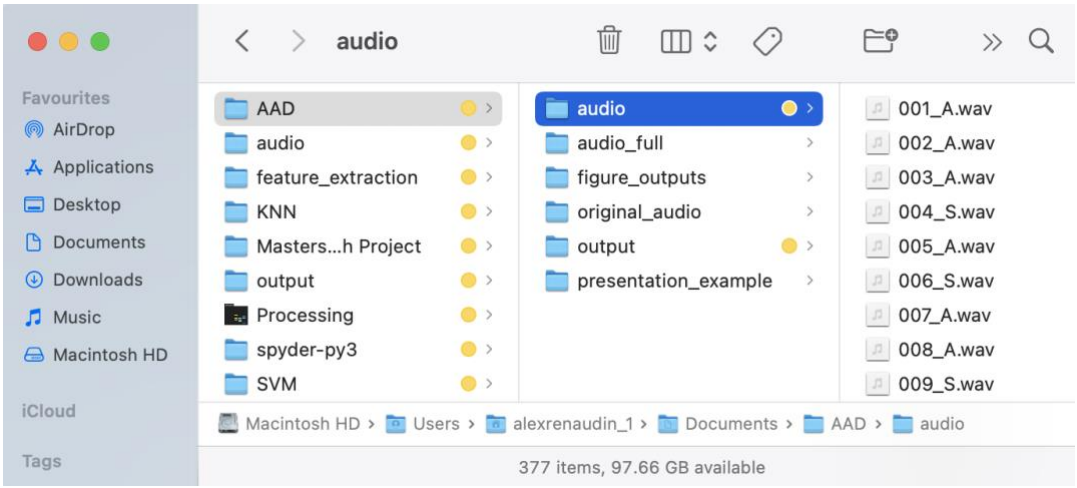


Figure 15: Dataset folder showing audio files after pre-processing.

7.2. Feature Extraction

The feature extraction process is crucial in transforming the audio dataset (after pre-processing) into the finalised dataset for the ML model. Each one-second audio clip is approximately 132 KB in size, which is too large for efficient processing by an ML algorithm. Thus, a feature extraction program was developed to mathematically characterise the audio in a concise form.

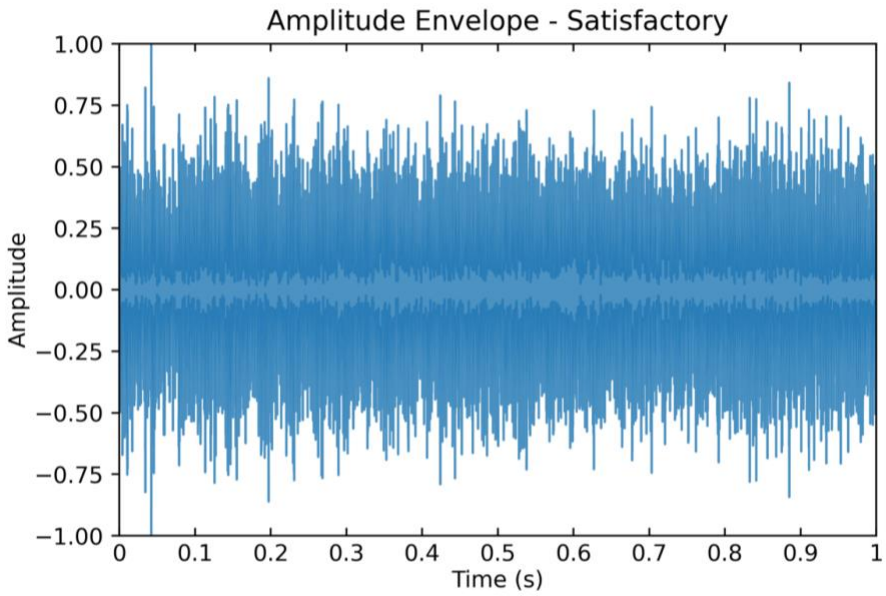


Figure 16: Amplitude waveform of satisfactory sample.

A significant emphasis during the literature review stage of this project was on exploring the most suitable method for feature extraction. It revealed that employing Mel bands and the Mel frequency cepstrum coefficient (MFCC) is highly effective in diminishing dataset size while preserving essential information about the nature of the samples. This preserved information is crucial to the ML model and enable it to perform accurate classification between anomalous and satisfactory saw cuts.

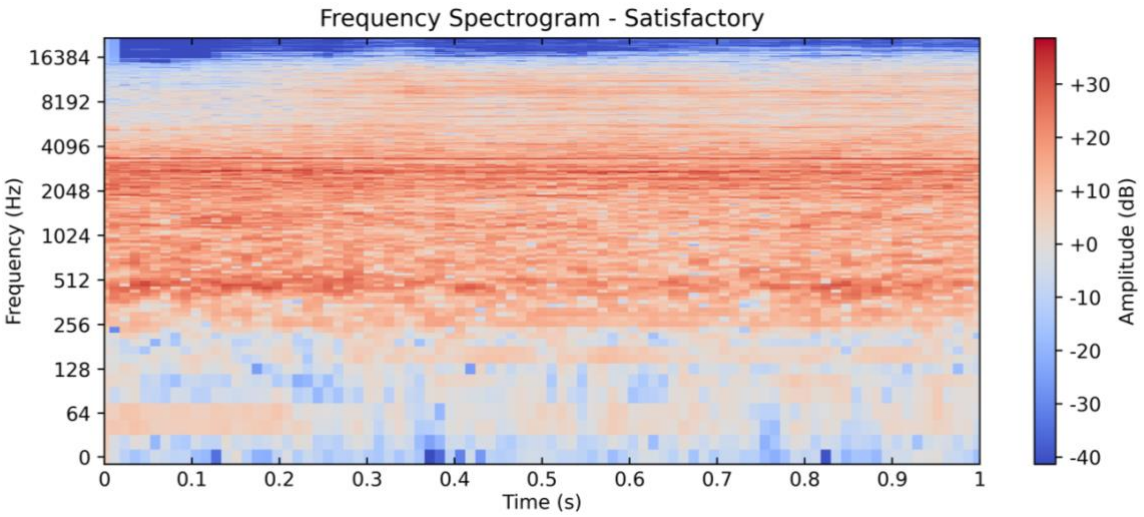


Figure 17: Frequency spectrogram of satisfactory sample.

The MFCC feature extraction process, shown in Figure 3, starts by windowing the time-based audio signal into smaller frames to increases the resolution. Then, a short-term Fourier transform (STFT) converts the signal into the frequency domain. Non-linear filter banks are applied to the signal, where the number of Mel bands (i.e., number of filters) is a key algorithmic parameter. Finally, a logarithmic conversion and discrete cosine transform are implemented, resulting in a range of MFCC values.

A Python program was developed for feature extraction using MFCC values on the audio dataset (see Appendix C for the program code). It iterates through the entire audio dataset, processing each file and generating a single array of all MFCC values

in CSV format. Visualisation of each algorithm stage is presented using the 'librosa' module.

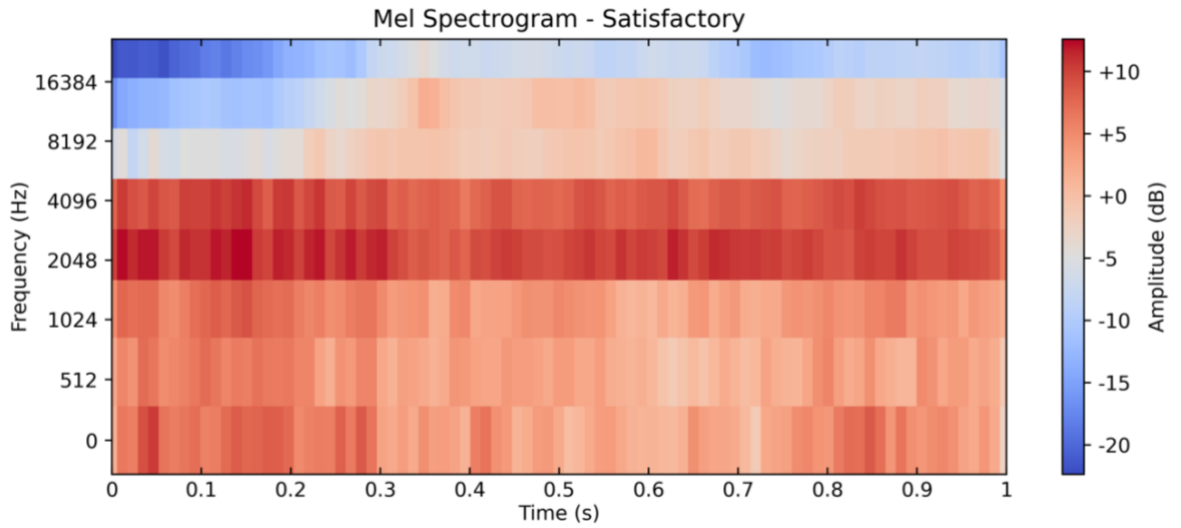


Figure 18: Mel spectrogram of satisfactory sample.

For illustration, a modified program was developed using two samples from both classes: one satisfactory and one anomalous. The results from the satisfactory sample are shown in this section, while the results from the anomalous sample are attached in Appendix B.

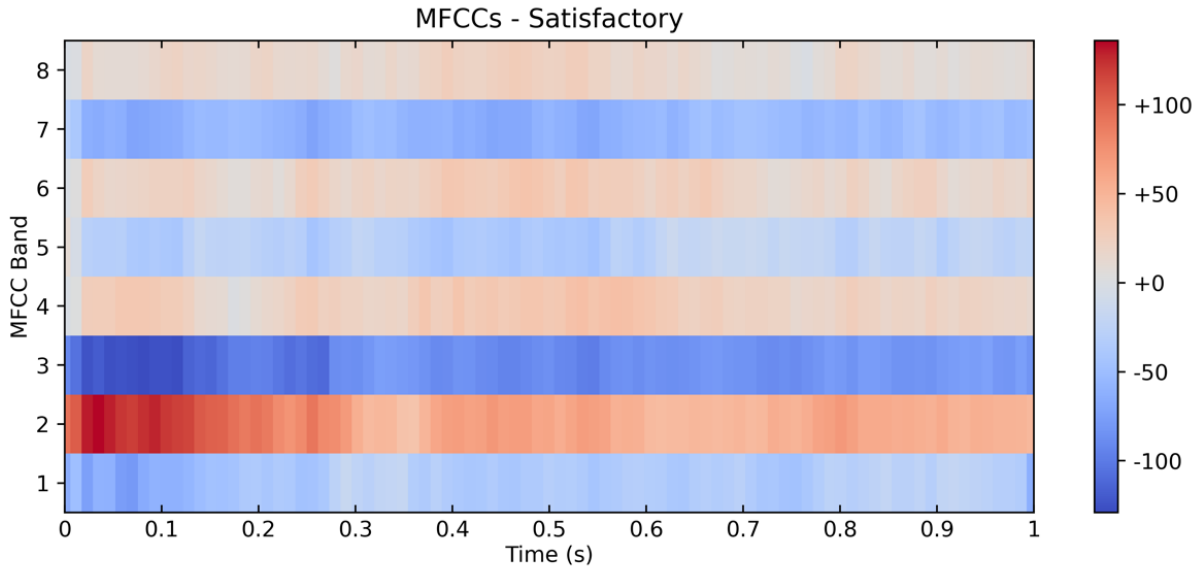


Figure 19: MFCC coefficients of satisfactory sample before statistical integration.

The feature extraction program works by first converting the time-domain audio data into the frequency domain, shown in Figure 17, using the STFT algorithm. It is subsequently processed through Mel bands to achieve the Mel spectrogram as shown in Figure 18.

The number of Mel bands, a critical parameter in the MFCC algorithm, was initially set at eight. More Mel bands allows more complexity to be represented by offering a higher resolution, however this is at the expense of a larger file size. A detailed investigation into optimal number of Mel bands was later undertaken and is detailed in Section 7.4.

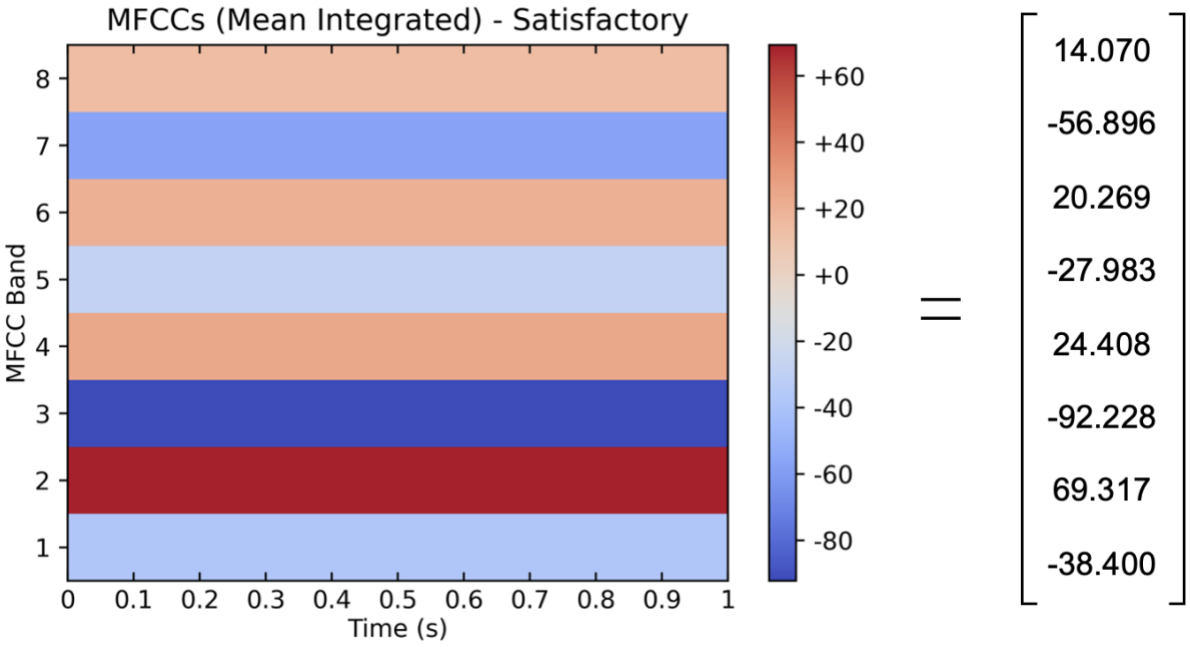


Figure 20: MFCC values of satisfactory sample after mean time integration.

Subsequently, the program extracts the MFCC values from the Mel spectrogram data, resulting in the data shown in Figure 19. This is represented as a time-varying value of intensity across each of the Mel bands. With a sampling rate of $44.1\ kHz$ and a hop

size of 512, the feature set size for the sample is an 8×87 array ($Mel\ bands \times time\ frames$), resulting in 696 total values for each audio sample.

$$no.time\ frames = \frac{sample\ rate\ (Hz)}{hop\ size} = \frac{44100}{512} = 87$$

The resulting feature set size is still too large, so a final step of statistical integration is employed to integrate the range of time-varying values into one single value. From the example above, this reduces the feature set size from an 8×87 array into an 8×1 array, thus eliminating the time domain.

The mean is one of the simplest methods of statistical integration and was thus used in this project. However, there are a range of other methods that can be used, such as the root mean square (RMS), maximum, and minimum. The RMS was also tested in the feature extraction program; however, its results were marginally poorer, and it was consequently disused.

In conclusion, the feature extraction program successfully produces the dataset for the ML model. This takes the form of an array saved in a CSV file. The number of rows in the array is equal to the number of Mel bands plus one (a binary 1 or 0 for the class indicator), and the number of rows is equal to the total number of audio samples plus one (header row).

7.3. Model Development

Following the completion of the finalised dataset, exploration was done into the development of the machine learning model. As with the feature extraction program, work was done in Python. Ultimately, three supervised classification models were built:

1. K-Nearest Neighbours (KNN): A relatively simple algorithm that classifies the data based on the majority class of its 'K' nearest neighbours in the feature space. This was the first model developed, and despite reasonable performance it was later discarded in favour of the customisation and increased performance of the SVM algorithm. Program code can be found in Appendix E.

2. Linear Classifier: Another relatively simple model that separates classes using a linear decision boundary, assigning input data to different classes based on weighted combinations of features. This was the second model built, and production was done in the Colab online environment using the TensorFlow library. Development was done as part of work on MCU deployment using a compact version of TensorFlow. Ultimately, this model was disused since MCU deployment was unsuccessful and performance was sub-optimal.

3. Support Vector Machine (SVM): A powerful algorithm that finds an optimal hyperplane to maximise the margin between classes in a high-dimensional feature space. This was the third and final model built and will remain the focus of further discussion in this report. Program code can be found in Appendix D.

Thus, the SVM algorithm, specifically a support vector classifier (SVC) was used on the dataset. The benefits of an SVC include an increased level of customisation and performance without added complexity. Program code was similar to that of the KNN model, with both leveraging functionality imported from the 'sklearn' module.

Model performance is assessed using the standardised set of metrics outlined below. These operate by assessing the ability of the model to classify positive (satisfactory) and negative (anomalous) samples, and they provide additional context to the total accuracy of the model.

- Precision: The ratio of correctly predicted positive observations to the total predicted positives, emphasising the accuracy of positive predictions. The formula is given as: $precision = \frac{TP}{TP+FP}$.

- Recall: Also known as sensitivity, this is the ratio of correctly predicted positive observations to all the observations in the actual positive class, with an overall focus on the model's ability to capture all positive instances. The formula is given as $recall = \frac{TP}{TP+FN}$.

- F1 score: A harmonic mean of both precision and recall, providing a balance between the two metrics in a single performance measure. The formula is given as $F1 = 2 \times \frac{precision \times recall}{precision + recall}$.

- Support: The number of samples of the class in the given dataset (typically given as the number of samples in the testing dataset), indicating the distribution between classes.

The program operates by first dividing the full dataset into testing and training components. It then builds an SVC and tests it by averaging its performance over numerous iterations (typically 1000). This was done due to slight variation in model performance associated with random allocation of testing and training datasets, and thus ensures reliability in results.

Table 5: SVM model performance metrics.

Class	Precision (%)	Recall (%)	F1 Score (%)	Support (samples)
Satisfactory	98.14	96.90	98.84	66
Anomalous	99.57	87.48	91.52	10

The averaged results of the performance metrics discussed above are detailed in Table 5. This provides feedback on both classes, allowing insight into the difference in performance when classifying satisfactory vs. anomalous samples. A variation of these results is presented in the confusion matrix, shown in Figure 22, which outlines the distribution of positive and negative classifications.

```
Running 1000 SVM models with a 20% split...

Complete! Mean model performance metrics:

   Size of dataset         = 377 samples
   Overall accuracy        = 97.98%
   Standard deviation      = 1.48%

   Satisfactory: Precision = 98.14%
                 Recall    = 96.90%
                 F1 score  = 98.84%
                 Support   = 66 samples

   Anomalous:    Precision = 99.57%
                 Recall    = 87.48%
                 F1 score  = 91.52%
                 Support   = 10 samples
```

Figure 21: SVM program console output.

While a comprehensive discussion of model performance and optimisation is included in Section 8.2, it is noted that ongoing work was done to improve model performance throughout the development process. An example was the development of new model architectures as discussed above, where the SVM algorithm was found to be offer the best mix of performance and simplicity.

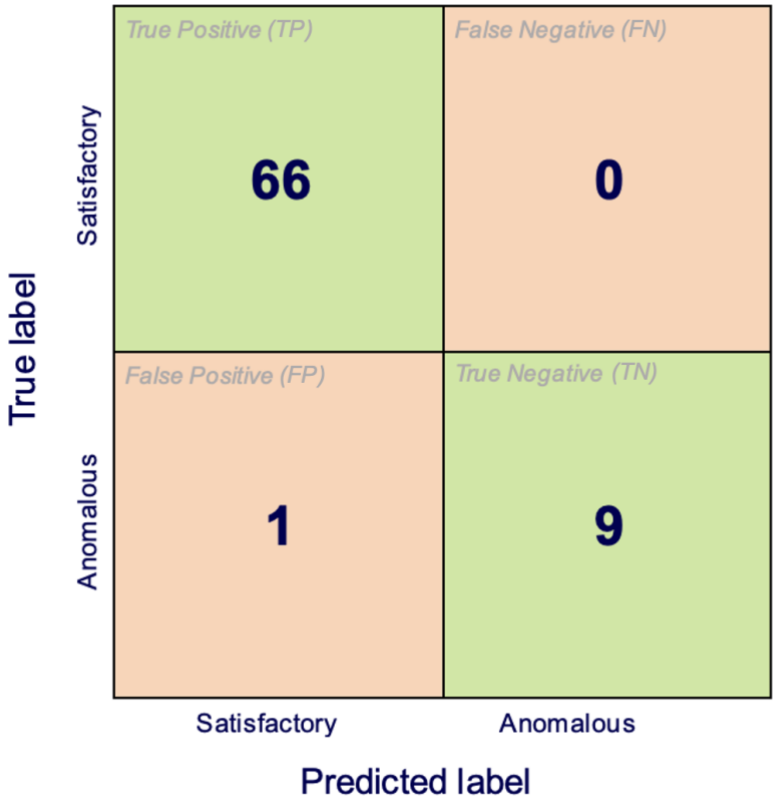


Figure 22: SVM confusion matrix.

The SVM program contains four adjustable parameters which can be varied to influence model performance:

1. Testing split: This is the proportion of the total dataset that is allocated to test the model with. A higher split leads to better accuracy when testing, however it decreases overall model performance as it is trained on a smaller training set.

2. Number of runs: This is the total number of unique models that are built and assessed. The overall performance metrics are computed by averaging the results from each unique run. Modifying this value has no real impact on model performance, however increasing it provides more reliability in the results.

3. C value: This is a hyperparameter of the SVC that affects the regularisation of results. The strength of regularisation is noted to be inversely proportional to the C value. It was found that increasing this value improved model performance to a point, beyond which no change was observed. A value of $100{,}000$ was used for this program.

4. Kernel type: This specifies the type of kernel used within the SVM algorithm, effectively changing the shape of the hyperplane separating the two classes. Common types include linear, polynomial, sigmoid, and radial basis function (RBF). The latter was used in this work.

Ultimately, the SVM model had an overall mean accuracy of 97.98%, with a standard deviation of 1.48%. This is promising performance and indicates a strong ability to classify the type of saw cut based on the audio. A detailed discussion into the performance is covered in Section 8.1.

7.4. Feature Extraction Parameter Tuning

A process of parameter tuning was undertaken to improve the overall model accuracy. It is noted that this investigation was undertaken following the completion of the first ML model (a simple KNN model without tuneable hyperparameters). As such, the focus of this process was placed primarily on the feature extraction program parameters, however it did also explore model parameters such as testing split and dataset size.

Table 6: Details of parameter tuning work.

Parameter	Testing Range	Original Value	Optimal Value
Mel bands	[2, 4, 8, 16, 32, 64, 128]	8	64
Sample rate	[4, 8, 16, 32, 44.1, 48] kHz	44.1kHz	8 kHz
Frame size	[1024, 2048, 4096]	2048	256
Hop size	[256, 512, 1024]	512	128
Test split	[5, 10, 15, 20, 25, 30] %	20%	20%
Statistical integrator	mean, RMS	mean	mean
Dataset size	[20, 50, 200, 335]	335	335

This process was structured by outlining the range of testing values for each parameter as shown in Table 6. Each parameter would be tested in isolation: i.e., all other parameters remained at the original value. Parameters were tested by running numerous (1000) iterations of the KNN model and recording the resulting model accuracy values.

Results from this tuning process were mixed. Some assumptions were confirmed, such as that increasing the dataset size will consistently improve accuracy. Conversely, some assumptions were challenged: for example, decreasing the sample rate saw a moderate increase in overall performance despite decreasing the resolution

of the data provided to the model. Further, the number of Mel bands, one of the key parameters in the MFCC algorithm, was shown to perform best when set to 64.

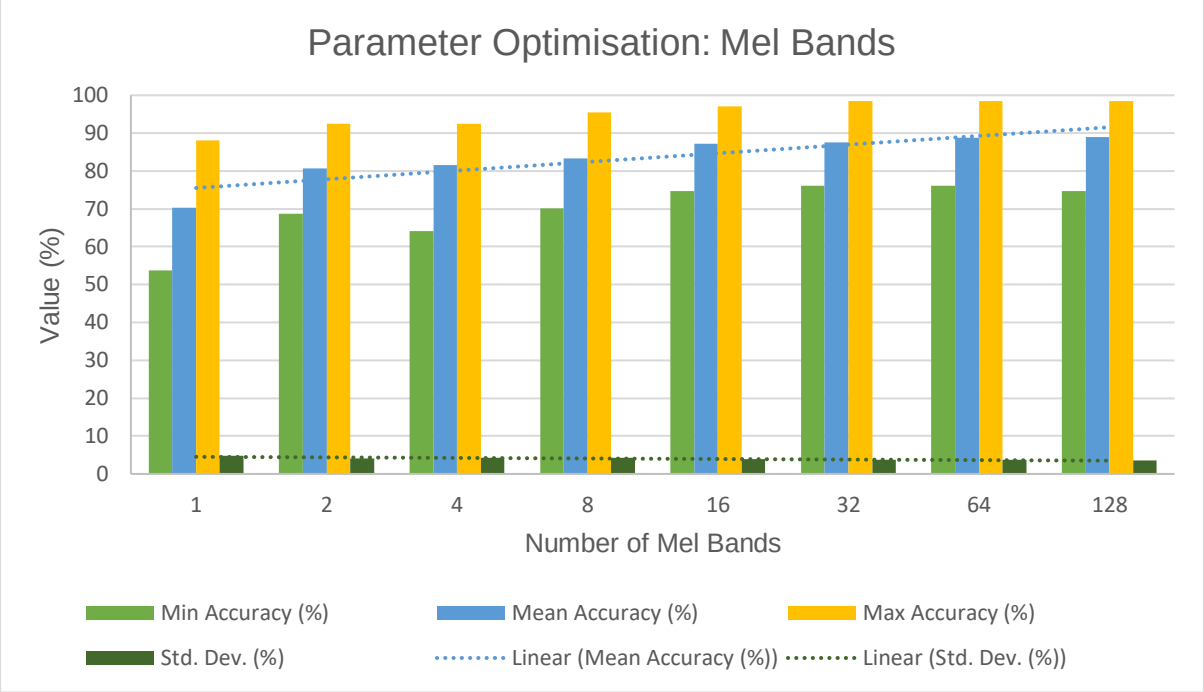


Figure 23: Effect of changing the number of Mel Bands.

The optimisation process resulted in an overall increase in mean model accuracy from 83.39% to 89.91%. Similar increases were observed in the minimum and maximum model accuracies, and the standard deviation decreased slightly from 4.20% to 3.54%. The increase from 8 to 64 Mel bands results in a larger dataset size, however the overall size is still small, and performance is not materially affected.

Table 7: Relationship between sample rate and frame & hop size.

Sample Rate (kHz)	Frame Size		Hop Size	
	Raw	Rounded	Raw	Rounded
4	100	128	40	64
8	200	256	80	128
16	400	512	160	256
32	800.00	1024	320	512
44.1	1102.5	2048	441	512
48	1200	2048	480	512

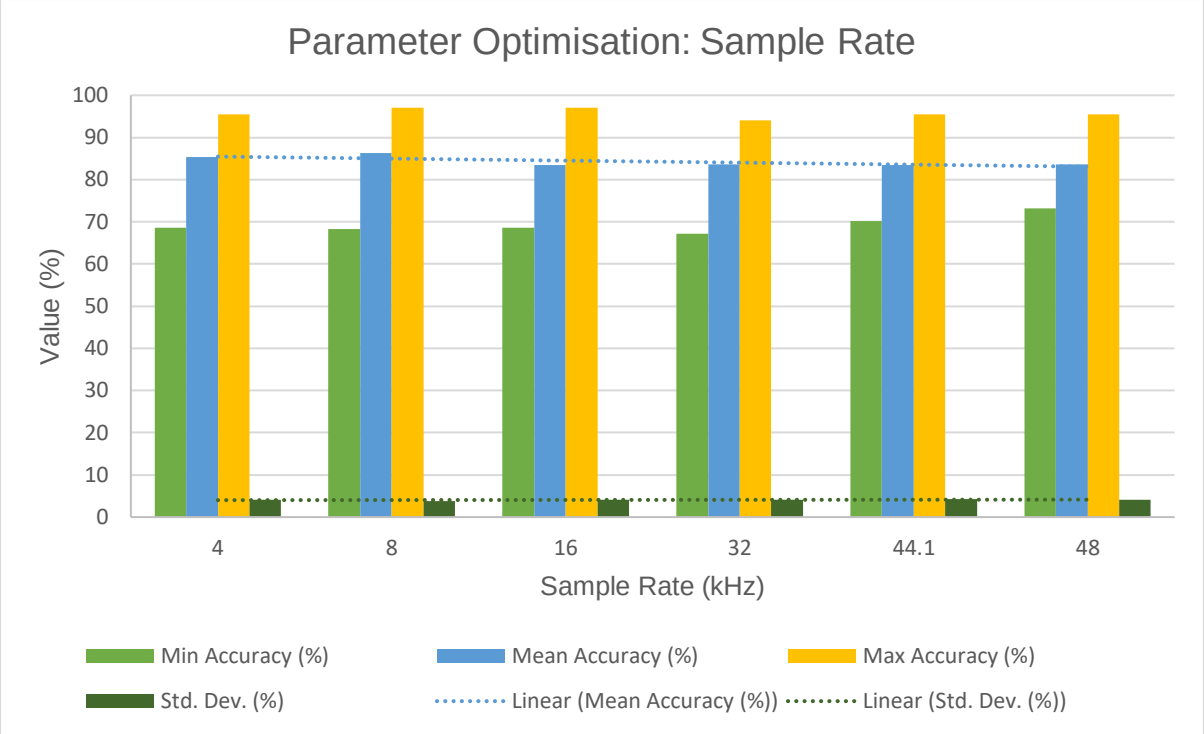


Figure 24: Effect of changing sample rate.

It is important to note the dependent relationship between sample rate and the frame & hop size of the feature extraction program. It is common practice to use a frame size of $25\ ms$ and a hop size of $10\ ms$ (or the nearest suitable value). The ideal number of samples for the hop and frame size are subsequently determined by the formula:

$$no.samples = size\ (ms) \times\ sample\ rate\ (kHz)$$

It is important to round up the final value to the nearest power of 2. For example, for a sample rate of $8\ kHz$, the ideal frame size is calculated as:

$$frame\ size = 25\ ms\ \times\ 8\ kHz = 200\ = 256\ samples$$

Table 8: Full results of parameter testing.

Parameter	Value	Accuracy (%)			
		Minimum	Mean	Maximum	Std. Dev.
Mel bands	1	53.73	70.3	88.06	4.76
	2	68.66	80.68	92.54	4.03
	4	64.18	81.57	92.54	4.26
	8	70.15	83.39	95.52	4.20
	16	74.63	87.24	97.01	3.84
	32	76.12	87.56	98.51	3.71
	64	76.12	88.74	98.51	3.66
	128	74.63	88.91	98.51	3.57
Sample rate (kHz)	4	68.66	85.31	95.52	4.11
	8	68.26	86.26	97.01	3.87
	16	68.66	83.46	97.01	4.15
	32	67.16	83.57	94.03	4.07
	44.1	70.15	83.39	95.52	4.20
	48	73.12	83.67	95.52	4.10
Frame size	256	71.11	83.88	97.30	4.04
	512	70.49	83.59	97.94	4.11
	1024	67.16	83.61	98.51	4.15
	2048	70.15	83.39	95.52	4.20
	4096	70.15	83.57	95.52	4.06
Testing split (%)	5	52.94	84.59	100.00	8.85
	10	64.71	83.73	100.00	5.94
	15	64.71	83.84	98.04	4.94
	20	70.15	83.39	95.52	4.20
	25	70.24	83.32	94.05	3.61
	30	73.27	83.20	93.07	3.32
	35	74.58	83.24	91.53	2.83
	40	76.12	83.03	91.04	2.63
Statistical integrator	mean	70.15	83.39	95.52	4.20
	RMS	70.64	82.97	94.03	3.90
Dataset size	20	0.00	66.57	100.00	24.00
	50	40.00	86.72	100.00	11.01
	200	57.50	76.59	95.00	6.08
	335	70.15	83.39	95.52	4.20
Final comparison	Optimal	76.12	89.91	100.00	3.54
	Original	70.15	83.39	95.52	4.20

7.5. MCU Deployment

The secondary aim of this project was to deploy a ML model on a MCU device, allowing the model to perform active AAD in a factory setting. Significant focus was placed on achieving this goal, and a compatible model was built featuring small size and low complexity. However, physical deployment was ultimately unsuccessful due largely to time and budget constraints.

This follows a proposal for physical deployment, featuring and Arduino MCU device (Section 6.2 for details), was submitted. However, the allocated student budget of $300 was ultimately inaccessible, thus removing this as an option.

Figure 25: Raspberry Pi MCU (Source: Raspberry Pi).

Consequently, an alternative MCU device was selected after a Raspberry Pi MCU was made available free-of-charge. This new approach was similarly supported by the TensorFlow module in Python and would serve as the motivation for the successful development of the second linear classifier ML model (outlined in Section 7.3).

The Raspberry Pi unit was successfully initialised, the latest version of Python installed, and an eight-microphone array successfully connected. However, installation of the TensorFlow module encountered technical issues and was ultimately unsuccessful. Multiple attempts to resolve this issue included restarting and reimaging the MCU device, however all were unsuccessful.

Figure 26: RESPEAKER 4 mic array compatible (Source: RESPEAKER).

Alternatively, it was decided to try the previously developed SVM and KNN models, detailed in Section 7.3, on the MCU. These two models were built using the 'sklearn' module, which is compatible with the Raspberry Pi unit. However, similar installation issues were encountered with a similar lack of resolution.

Following these two issues, an alternative approach to deployment on the Raspberry Pi was not clear, and a resolution was unlikely without significant work. Therefore, it was decided to abandon efforts towards MCU deployment in favour of performing the model optimisation within the project timeline.

8. Discussion

This section aims to provide context to the results of project. This includes an interpretation of the performance metrics of the final SVM model, discussion around work to optimise performance at various stages, followed by details of issues with the dataset's suitability. This is concluded by a summary of the challenges faced throughout work on this project.

8.1. Model Performance

As detailed in the results of ML development in Section 7.3, overall model performance was strong with a high accuracy in classifying saw audio data. The use of the SVM algorithm was found to be a good fit for the problem as it offered strong classification performance without excessive complexity that would have challenged the project timeline.

An analysis of the SVC's performance metrics shows slightly stronger performance in classifying the positive (satisfactory) class as evidenced in Table 5. While precision was strong across both classes, a significant drop in recall is observed for the negative (anomalous) class resulting in a lower F1 score. This is a weakness of the model, as its primary aim is to perform reliable and accurate classification of anomalous behaviour.

An explanation for this weaker performance is the overrepresentation of the positive class within the dataset at 82.46%. This is further illustrated by a relatively low value of support (10 samples) in the testing dataset. This not only provides the model with less negative class data to be trained on, but also amplifies any negative

performance when testing the model. This is evidenced in the confusion matrix (Figure 22) in which a single sample was falsely identified corresponding to approximately 90% accuracy within the negative class.

8.2. Model Optimisation

The final high-accuracy model was ultimately realised through ongoing enhancements in the development process. The primary approach to improving model performance was the optimisation of program parameters for both the feature extraction program and the machine learning models. This process is detailed in Section 7.4 discussing the results of tuning the parameters of the feature extraction program to improve performance.

It is noted that enhancements from Section 7.4 were performed on an early KNN model prototype. The results discuss an average accuracy improvement from 83.39% to 89.91%, highlighting the improvement from this point to the final model accuracy of 97.98%. This was achieved through similar methods of adjusting feature parameters, however a large contributor was the superiority of the SVM algorithm, as well as a revision of the audio dataset which is discussed in Section 8.3.

8.3. Dataset Suitability

One of the most effective improvements to model performance was observed after a comprehensive revision of the audio dataset. This followed the work done on tuning the feature extraction program parameters (Section 7.4) and the development of the SVM model. At this point, the model was able to classify the dataset with an accuracy of approximately 90%, which was promising for the protype KNN model but needed improvement to achieve the project aims.

It was at this stage that the poor performance of negative class classification (as discussed in Section 8.1) was observed. Precision and recall values were approximately 20% weaker compared to the positive class, prompting further investigation into the issue. Since there was little room for improvement through modification of the model architecture, it was decided to investigate for issues with the anomalous class of the audio dataset.

Upon examination numerous samples in the dataset were noted to possess one of the following issues:

- Mislabelled: This was the easiest issue to fix, as it occurred when a sample had been mistaken labelled in the wrong class. Mislabelled samples were corrected and returned to the dataset.

- Excessive background noise: A few samples were removed that contained disruptive amounts of background noise, such as the use of an angle grinder or nail gun. This did not include minor instances of background noises, such as conversations or radio noise.

- Poor pre-processing: This included samples that had been cropped too early or too late, missing the saw cut. Samples were fixed where possible, however a lot of samples needed to be rejected.

- Alternate saw: A few samples recorded audio of a different drop saw. It was decided to remove these samples as the unique sound profile may have been skewing model results.

- Uncertain class: This was the most common and difficult issue to resolve. It involved a significant number of samples which had very little consistency in the class labels. Successful labelling of these samples required a very clear distinction between satisfactory and anomalous saw behaviour, which is a very difficult division to draw. Further, it requires exceptional consistency to label the large dataset, both of which are very difficult to perform manually. Samples were removed for these reasons.

Of the total 513 samples in the audio dataset, 136 samples were removed for one or more of the issues outlined above, resulting in the final dataset size of 377 samples. This is a significant reduction in size, however use of the revised dataset resulted in the high performance of the SVC. This significant increase in model performance supports the theory that poorly labelled data was hindering model performance.

However, removal of the problematic samples is not an ideal solution, as the model should be able to classify ambiguous sounds for it to effectively function as a safety device. A resolution to this issue is proposed in the form of a complete revision of the dataset. However, this would require significant labour, and was thus not considered in the scope of this project.

8.4. Issues Faced

Overall, project progress ran smoothly without major delay or issue. Some minor issues were encountered and were resolved with varying success:

- Lack of anomalous samples: Since the majority of saw cuts are satisfactory (positive), a large bias in the dataset meant there were significantly fewer anomalous (negative) samples. This issue was partially resolved by experimentally inducing anomalous saw; however it was extremely labour intensive and not sustainable given project resources.

- Dataset size: Since no existing dataset was available for this project, a novel dataset was created from over 500 audio samples. This required considerable time and labour, and an increase in dataset size was not possible given the time constraints of the project. A significantly larger dataset would improve model accuracy and allow for the use of a more complex model architecture such as the artificial neural network.

- MCU deployment: As covered in Section 7.5, the deployment of the model onto a MCU device was ultimately unsuccessful. This due to an inability to access project funds to purchase the proposed components. An alternative Raspberry Pi MCU was utilised, however technical issues prevented success in running the ML model on the device. Additional focus on this work promises successful results and would be greatly supported by support and training with MCU devices. However, considering MCU deployment was a secondary project objective, this work was discontinued to maintain project schedule.

- Dataset suitability: As discussed in Section 8.3, a proportion of problematic audio samples were identified as impacting model performance. The samples that were deemed too difficult to resolve were removed from the dataset, providing a quick fix that drastically improved model performance. However, it was noted that this may hinder the model's performance performing AAD in a factory setting due to its inability to characterise ambiguous audio. As with the other issues, an ideal resolution was not possible due to time and labour constraints. Nonetheless, it is affirmed that focused work on relabelling the dataset would provide a solution to this issue.

9. Conclusions

In summary, this report covers the work to develop a ML model to perform AAD on a novel dataset of saw audio recordings. Results were generally successful, and a suitable SVM model was able to classify samples with an average accuracy of 97.98%. This model was trained and tested on an experimentally collected dataset that was characterised by a custom-built feature extraction program.

Detailed in this report is an in-depth review of the current literature in the field of ML for AAD, where trends and gaps in this research influenced the development of the project goals. These aims were leveraged to shape the project plan, outlining the key stages in the project timeline. This was followed by details of the methodology and techniques necessary for each stage of work.

The results of project work are provided in full, covering the collection of the audio dataset, work done on pre-processing, the development of the feature extraction program, and ultimately the various ML models that were trialled, tested, and iterated. A comprehensive discussion follows project results, in which trends were highlighted and issues analysed.

Ultimately, the work of this project shows promise as both a safety feature and maintenance device for drop saws. It demonstrates a strong ability to identify anomalous, and potentially dangerous, behaviour in the saw based on its acoustic signals, thus allowing it to alert the operator of any issues. While testing of this capability was not possible without MCU deployment, this is maintained as a promising scope for future work.

9.1. Recommendations

A few recommendations are made for future work in this space. These are motivated by areas of difficulty and aim to offer guidance for resolution of the issues encountered.

- MCU deployment and testing: While the stretch goal of MCU deployment was ultimately unsuccessful, it acts as the primary recommendation for future work. Additional work would allow the import of the SVM model from this report onto a MCU device. Focus should be placed on performing active AAD in a factory setting, comparing the theoretical suitability of the SVM against real-world performance.

- Data pre-processing automation: One of the most time-consuming stages of this project was the manual processing of the raw audio files into second-long samples of the saw in use. The task was highly repetitive and relatively simple, making it a suitable candidate for automation. Any solution that can eliminate the number of steps in this procedure (outlined in Appendix A) would result in a significantly shorter processing time, ultimately allowing a larger dataset to be produced.

- Data label revision: As discussed in Section 8.3, many data samples were difficult to confidently classify, even with trained human judgement. This is due to a lack of a clear distinction between the two classes. There exists a large middle ground between the two classes, such as the instance where the saw is under minor strain but does not suggest immediate danger to the operator. Therefore, it is recommended to redefine the definition of both classes to improve the reliability of the data labelling. It is suggested that consultation with industry experts would benefit this process greatly.

10. References

1. Abbasi, S., Famouri, M., Shafiee, M. J., & Wong, A. (2021). Outliernets: Highly compact deep autoencoder network architectures for on-device acoustic anomaly detection. *Sensors (Basel, Switzerland)*, *21*(14), 4805. https://doi.org/10.3390/s21144805

2. Abeysinghe, A., Fard, M., Jazar, R., Zambetta, F., & Davy, J. (2021). Mel frequency cepstral coefficient temporal feature integration for classifying squeak and rattle noise. *The Journal of the Acoustical Society of America*, *150*(1), 193-201. https://doi.org/10.1121/10.0005201

3. Altinors, A., Yol, F., & Yaman, O. (2021). A sound based method for fault detection with statistical feature extraction in UAV motors. *Applied acoustics*, *183*, 108325. https://doi.org/10.1016/j.apacoust.2021.108325

4. Bianco, M., Gerstoft, P., Traer, J., Ozanich, E., Roch, M., Gannot, S., Deledalle, C., & Li, W. (2019). *Machine learning in acoustics: a review*.

5. Cheng, Y.-H., & Kuo, C.-N. (2022). Machine Learning for Music Genre Classification Using Visual Mel Spectrum. *Mathematics (Basel)*, *10*(23), 4427. https://doi.org/10.3390/math10234427

6. Coelho, G., Matos, L. M., Pereira, P. J., Ferreira, A., Pilastri, A., & Cortez, P. (2022). Deep autoencoders for acoustic anomaly detection: experiments with

working machine and in-vehicle audio. *Neural computing & applications*, *34*(22), 19485-19499. https://doi.org/10.1007/s00521-022-07375-2

7. Glowacz, A., Tadeusiewicz, R., Legutko, S., Caesarendra, W., Irfan, M., Liu, H., Brumercik, F., Gutten, M., Sulowicz, M., Antonino Daviu, J. A., Sarkodie-Gyan, T., Fracz, P., Kumar, A., & Xiang, J. (2021). Fault diagnosis of angle grinders and electric impact drills using acoustic signals. *Applied acoustics*, *179*, 108070. https://doi.org/10.1016/j.apacoust.2021.108070

8. Grandhi, R. T., & Prakash, N. K. (2021). Machine-Learning Based Fault Diagnosis of Electrical Motors Using Acoustic Signals. In (pp. 663-671). Springer Singapore Pte. Limited. https://doi.org/10.1007/978-981-15-8530-2_52

9. Ramírez, J., & Flores, M. J. (2020). Machine learning for music genre: multifaceted review and experimentation with audioset. *Journal of intelligent information systems*, *55*(3), 469-499. https://doi.org/10.1007/s10844-019-00582-9

10. Renaudin, A. (2023). *Machine Learning for Acoustic Anomaly Detection in Electric Motors* RMIT]. Melbourne.

11. Son, J., Kim, C., & Jeong, M. (2022). Unsupervised Learning for Anomaly Detection of Electric Motors. *International journal of precision engineering and manufacturing*, *23*(4), 421-427. https://doi.org/10.1007/s12541-022-00635-0

12. Suman, A., Kumar, C., & Suman, P. (2022). Early detection of mechanical malfunctions in vehicles using sound signal processing. *Applied acoustics*, *188*, 108578. https://doi.org/10.1016/j.apacoust.2021.108578

13. Wang, Y. S., Liu, N. N., Guo, H., & Wang, X. L. (2020). An engine-fault-diagnosis system based on sound intensity analysis and wavelet packet pre-processing neural network. *Engineering applications of artificial intelligence*, *94*, 103765. https://doi.org/10.1016/j.engappai.2020.103765

14. Wu, Z., Wan, Z., Ge, D., & Pan, L. (2022). Car engine sounds recognition based on deformable feature map residual network. *Scientific reports*, *12*(1), 2744-2744. https://doi.org/10.1038/s41598-022-06818-z

15. Yan, H., Bai, H., Zhan, X., Wu, Z., Wen, L., & Jia, X. (2022). Combination of VMD Mapping MFCC and LSTM: A New Acoustic Fault Diagnosis Method of Diesel Engine [Article]. *Sensors*, *22*(21), Article 8325. https://doi.org/10.3390/s22218325

Appendix A. Data Pre-Processing Procedure

The protocol below was formulated to guide the data acquisition process (Stage 3) of the project. Maintaining a dataset that is consistent, well-labelled, and organised is crucial, as it ensures seamless integration into the subsequent project stages of feature extraction and model development. This protocol offers a strict procedure for the collection and pre-processing of the audio dataset, thus ensuring that the final audio files are prepared for input into a feature extraction program.

1. Record raw audio data:
    - Use Sony ICD UX-200F audio recorder.
    - Record one full workday as a single file in ‘ST’ (standard) mode.
    - Save audio in mp3 format at 192 kbit/s.
    - Export routinely to computer and clear audio recorder memory.

2. Save raw file:
    - Change filename to the date of recording in the form ‘DD.MM’ (i.e., ‘27.04’).
    - Create ‘original’ parent folder and store files in subfolder named ‘raw’.
    - Note each file is approx. 800 MB, and thus too large to import into Ableton.

3. Pre-processing:
    - Import raw file into Audacity.
    - Cut file into one-hour segments and place each in separate tracks.
    - Export each track as a separate file with original format (mp3,192 kbit/s).
    - Name files as the date (as in Step 2) appended by the track number (e.g., ‘27.04.3’ for the third track).

4. Segment hour-long clip into individual samples:
    - Import individual hour-long clip from Step 3 into Ableton.
    - Make slices in the audio track to isolate each cut with the saw.
        - Start when the saw contacts the wood.
        - End when the saw finishes cutting through the wood.
        - Treat multiple cuts as separate samples.
    - Crop each clip to 1 second length (rejecting clips shorter than 1 second).
        - Clips that are longer than 2 seconds can be cut into multiple clips.
    - Note the class (either satisfactory ‘S’ or anomalous ‘A’) of each clip such that it can be assigned to the filename in Step 5 below.
    - Export each clip as a mono-audio, normalised WAV file.
    - Repeat for each segmented track from Step 3

5. Rename each file appropriately:
    - Name as sample number (i.e., ‘001’) with suffix at end to denote class (i.e. used to identify anomalous samples from satisfactory ones)
        - Samples with safe, standard saw use suffix ‘_S’ for satisfactory.
        - Damaged or dangerous samples use suffix ‘_A’ for anomalous.
    - Increase in chronological order, starting from ‘001’ (e.g., ‘001_S’, ‘002_A’, ‘003_S’, ‘004_S’, …).
    - Adjust leading zeros if needed such that all filenames are the same length.

6. Store files appropriately:
    - Create a folder entitled ‘audio’ and add all files from Step 5 inside.
        - This forms the audio dataset to be used later stages.
    - Create an ‘original_audio’ folder with ‘raw’ and ‘segmented’ subfolders.
    - Add the original raw files from Step 2 into the ‘raw’ folder.
    - Add the segmented files from Step 3 into the ‘segmented’ folder.

Appendix B. Feature Extraction Results

The figures below display the results of the feature extraction program example for an anomalous audio sample. These mirror those discussed in Section 7.2 on the feature extraction process and can be used to compare the difference between the two classes of audio.

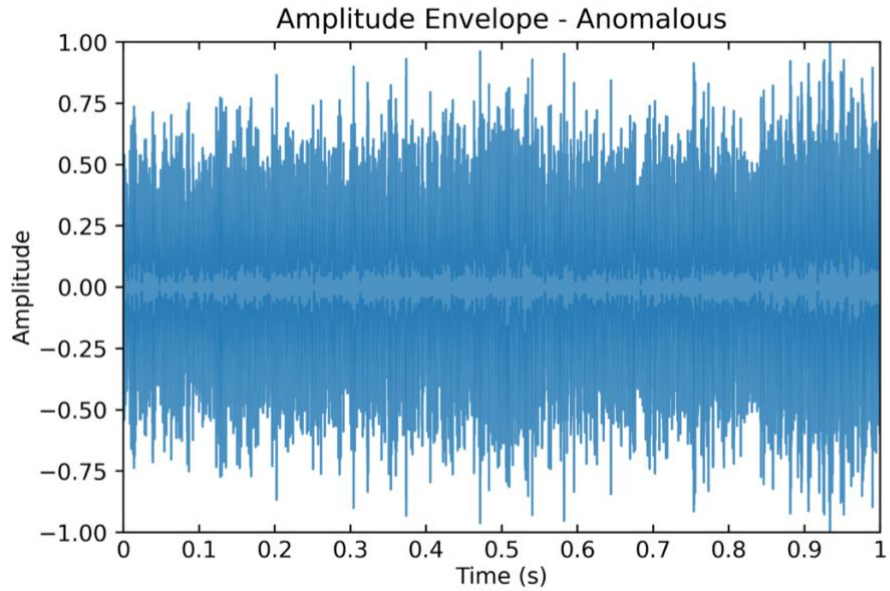


Figure 27: Amplitude envelope of anomalous sample showing audio waveform.

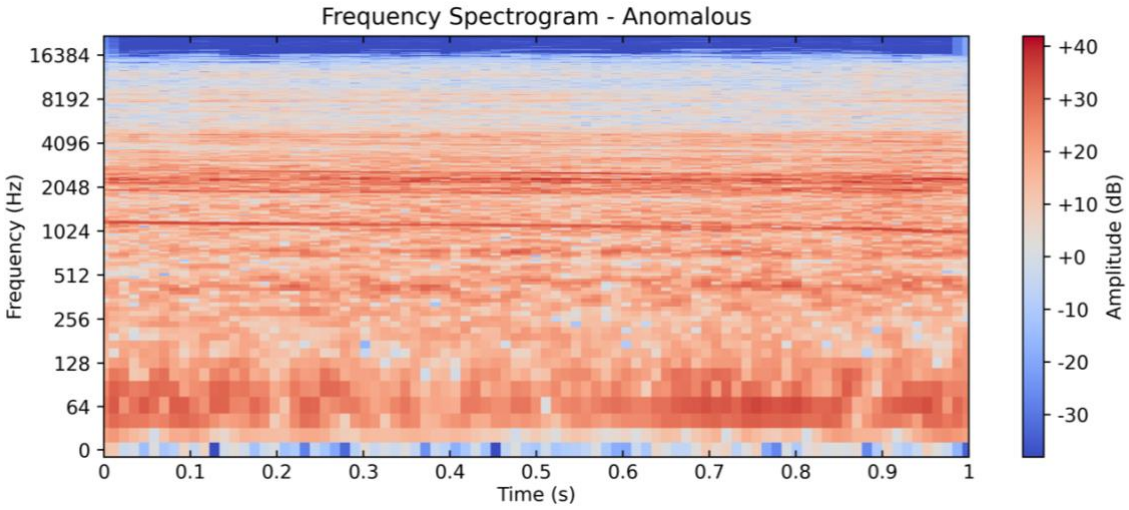


Figure 28: Frequency spectrogram for anomalous sample.

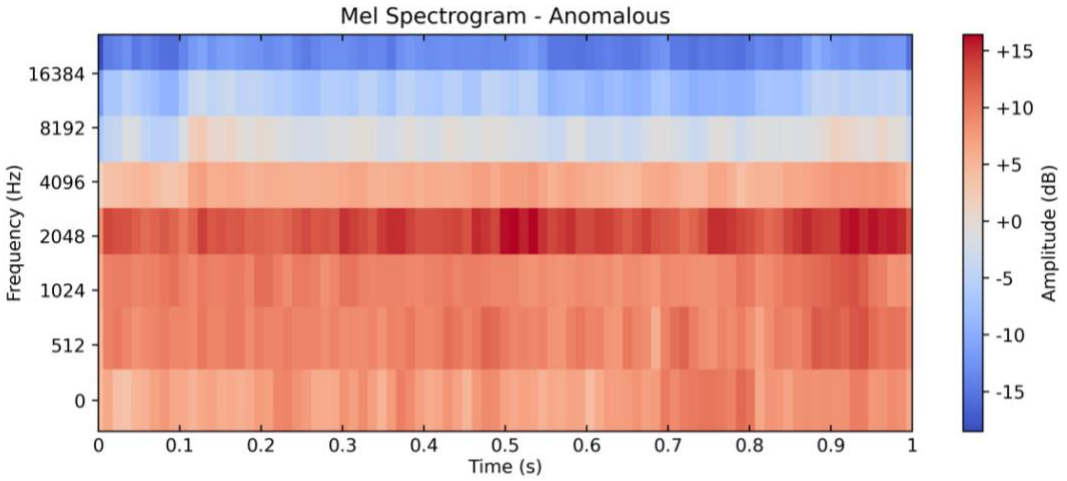


Figure 29: Mel spectrogram for anomalous sample.

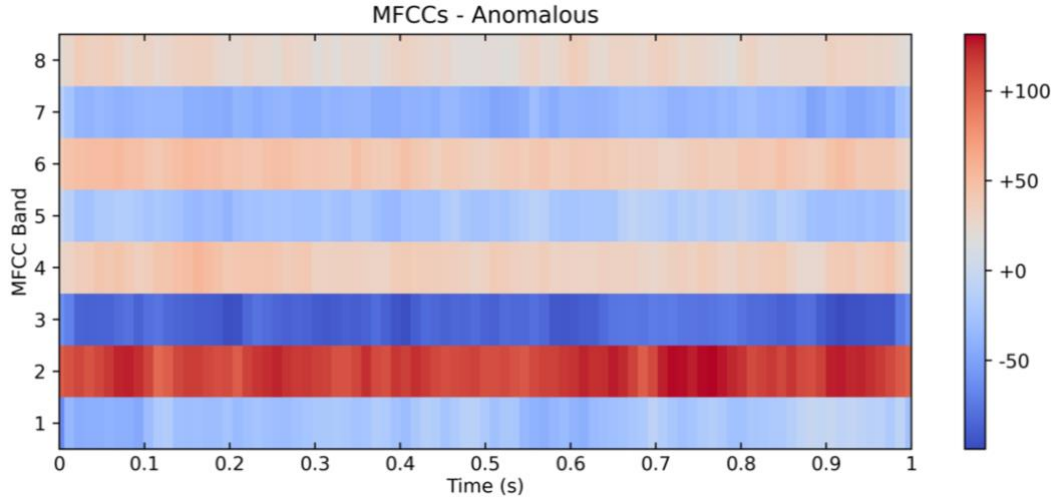


Figure 30: MFCCs for anomalous sample before mean integration.

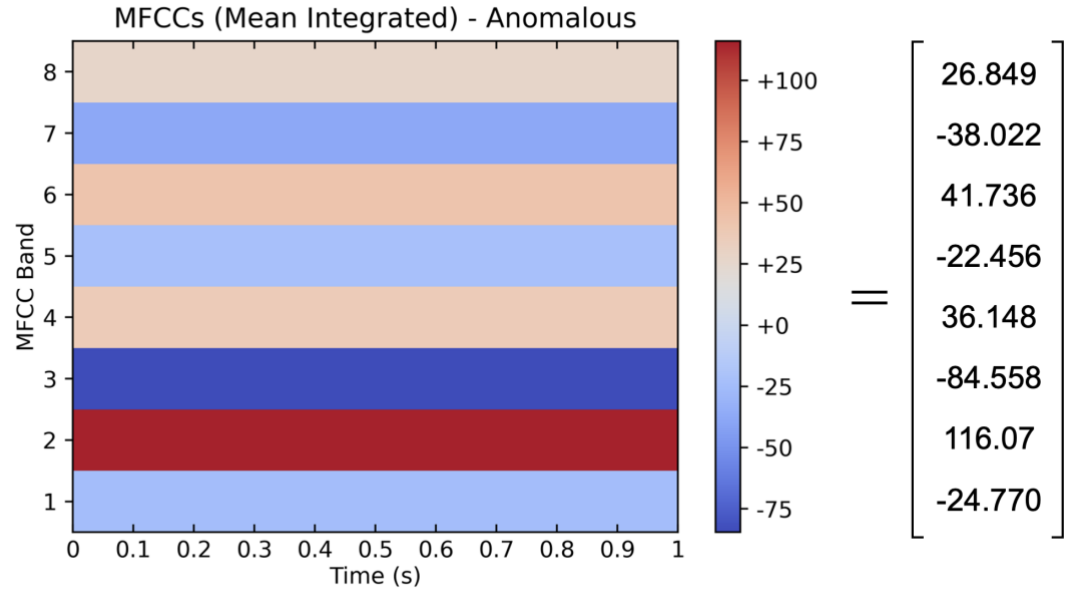


Figure 31: MFCCs for anomalous sample after mean integration.

Appendix C. Feature Extraction Program

```
# ================================================================================
#
# Machine Learning for Acoustic Anomaly Detection
# Feature Extraction (FE) program (version 2.3)
# Alex Renaudin 22 October 2023
#
# Computes the MFCCs of the audio in the 'AAD/audio' folder.
# Audio filenames (e.g., '158_A.WAV') contain a number and a class:
# S = 'satisfactory' (i.e., normal) behaviour, A = 'anomalous' behaviour
# Program returns CSV file containing Mel band information + class indicator.
#
# Tuneable program parameters:
#    - Sample rate
#    - Number of Mel bands
#    - Frame size
#    - Hop size
#
# Uses the mean as statistical integration method. Alternate methods can be
# introduced (e.g., max, RMS) for varied results.
#
# Based off 'Audio Signal Processing for Machine Learning' by Valerio Velardo
# https://www.youtube.com/playlist?list=PL-wATfeyAMNqIee7cH3q1bh4QJFAaeNv0
#
# ================================================================================

import os
import librosa
import numpy as np
import csv
import glob
import random

# Program parameters:
SAMPLE_RATE = 8000      # Sample rate (kHz)
N_MELS = 64             # Number of Mel bands
FRAME_SIZE = 256        # Number of samples in frame
HOP_SIZE = 128          # Number of samples in hop

folder_path = "/Users/alexrenaudin_1/Documents/AAD/audio" # AAD audio folder
file_extension = ".wav" # Define file extension as WAV

num_files = len(os.listdir(folder_path)) - 1 # Number of files in AAD folder

# List of all audio files in the 'AAD/audio' folder using glob module
audio_files = sorted(glob.glob(folder_path + "/*" + file_extension))
random.shuffle(audio_files) # Shuffle list of audio files

# Initalises feature (output) matrices (+1 column for class indicator)
dataset = np.zeros((num_files, N_MELS+1))

# Prints information about folder size and parameter values
print(f"\nTotal files:\t\t{num_files}")
print(f"Samplerate:\t\t{SAMPLE_RATE} Hz")
print(f"Frame size:\t\t{FRAME_SIZE}")
print(f"Hop size:\t\t{HOP_SIZE}")
print(f"Mel bands:\t\t{N_MELS}")

#%% Iterates through each audio file in the 'AAD/audio' folder
```

```
for n in range(num_files):

    # Extract the class from the filename and add to last column of output
    class_label_str = audio_files[n].split("/")[-1].split(".")[0][-1]
    if class_label_str == 'A':
        dataset[n, N_MELS] = int(1)
    else:
        dataset[n, N_MELS] = int(0)

    # Librosa audio extraction
    audio, sr = librosa.load(audio_files[n], sr=SAMPLE_RATE)

    #%% MFCC computation

    # Short-time fourier transform (STFT) from librosa library
    S_scale = librosa.stft(audio,n_fft=FRAME_SIZE,hop_length=HOP_SIZE)

    Y_scale = np.abs(S_scale) ** 2 # Square absolute S_scale value
    Y_log_scale = librosa.power_to_db(Y_scale) # Convert to dB scale

    # Creates Mel filter banks
    filter_banks = librosa.filters.mel(n_fft=FRAME_SIZE,
                                           sr=SAMPLE_RATE, n_mels=N_MELS)

    # Builds Mel spectrogram
    mel_spectrogram = librosa.feature.melspectrogram(y=audio,
                  sr=sr, n_fft=FRAME_SIZE, hop_length=HOP_SIZE, n_mels=N_MELS)

    # Convert to dB scale
    log_mel_spectrogram = librosa.power_to_db(mel_spectrogram)

    # Calls librosa function to extract MFCCs
    mfccs = librosa.feature.mfcc(y=audio, n_mfcc=N_MELS, sr=sr)

    #%% Mean statistical indicator

    # Iterates through Mel bands and integrates using statistical indicators
    for i in range(N_MELS):

        # Temporary values for mean and RMS respectively
        temp_sum = 0

        # Iterates through time of each Mel band and computes sums
        for j in range(mfccs.shape[1]):
            temp_sum += mfccs[i, j]

        dataset[n, i] = temp_sum/mfccs.shape[1] # Final mean value

#%% Export outputs (statistical indicators) as .csv files

header = [f'mel{i+1}' for i in range(N_MELS)] + ['class'] # Define CSV header

# 'test.csv'
with open('/Users/alexrenaudin_1/Documents/AAD/output/dataset.csv', 'w',
                                            encoding='UTF8', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(header)
    writer.writerows(dataset)
print("\nOutput 1:\t\tdataset.csv") # Print after file is saved
```

Appendix D. SVM Program

```
# ================================================================================
#
# Machine Learning for Acoustic Anomaly Detection
# Support Machine Vector (SVM) program (version 1.5)
# Alex Renaudin 22 October 2023
#
# This program builds a Support Vector Classifier (SVC) and records its
# performance by averaging scores across 1000 runs.
#
# Tuneable program parameters:
#    - Testing split proportion
#    - Number of runs
#    - C value
#    - Kernel type
#
# Based off YouTube playlist 'Machine Learning with Python' by Sentdex
# https://www.youtube.com/playlist?list=PLQVvvaa0QuDfKTOs3Keq_kaG2P55YRn5v
#
# ================================================================================

import numpy as np
import sklearn.metrics
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_fscore_support

# Load in dataset
df = pd.read_csv('dataset.csv')

# Assign class features to X and y arrays
X = np.array(df.drop(labels=['class'], axis=1))
y = np.array(df['class'])

# Program parameters:
SPLIT = 0.20        # Proportion of testing set (e.g. 0.200 = 20% testing)
NUM_RUNS = 1000     # Number of iterations of model
C = 100000          # 'Regularisation parameter' for SVC
KERNEL = 'rbf'      # Kernel type for SVC

# Initialise performance metric variables
accuracy = []                           # Total accuracy score
precision_S = []; precision_A = []      # Precision
recall_S = []; recall_A = []            # Recall
F1_S = []; F1_A = []                    # F1 score
support_S = []; support_A = []          # Support
cfm = []                                # Confusion matrix

# Print parameters
print(f'\n\nRunning {NUM_RUNS} SVM models with a {SPLIT:.0%} split...\n')

# Iterate NUM_RUNS times, creating and testing new SVM model each time
for _ in range(NUM_RUNS):

    # Randomly seperate data into training and testing clumps
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=SPLIT)
    model = svm.SVC(kernel = KERNEL, C = C) # Creates the model classifier
    model.fit(X_train, y_train) # Fits the training sets to the classifier

```

```
61        # Compute predicted answers based off testing dataset
62        y_pred = model.predict(X_test)
63
64        # Compute performance metrics for this run
65        report = precision_recall_fscore_support(y_test, y_pred)
66
67        # Append performance metrics values
68        accuracy.append(model.score(X_test, y_test))
69        precision_S.append(report[0][0])
70        recall_S.append(report[0][1])
71        precision_A.append(report[1][0])
72        recall_A.append(report[1][1])
73        F1_S.append(report[2][0])
74        F1_A.append(report[2][1])
75        support_S.append(report[3][0])
76        support_A.append(report[3][1])
77        cfm.append(sklearn.metrics.confusion_matrix(y_test, y_pred))
78
79    # Print results
80    print('Complete! Mean model performance metrics:\n')
81
82    print(f'\tSize of dataset          = {len(df)} samples')
83    print(f'\tOverall accuracy         = {100*np.mean(accuracy):.2f}%')
84    print(f'\tStandard deviation       = {100*np.std(accuracy):.2f}%\n')
85
86    print(f'\tSatisfactory:  Precision = {100*np.mean(precision_S):.2f}%')
87    print(f'\t               Recall    = {100*np.mean(recall_S):.2f}%')
88    print(f'\t               F1 score  = {100*np.mean(F1_S):.2f}%')
89    print(f'\t               Support   = {round(np.mean(support_S))} samples\n')
90
91
92    print(f'\tAnomalous:     Precision = {100*np.mean(precision_A):.2f}%')
93    print(f'\t               Recall    = {100*np.mean(recall_A):.2f}%')
94    print(f'\t               F1 score  = {100*np.mean(F1_A):.2f}%')
95    print(f'\t               Support   = {round(np.mean(support_A))} samples\n')
96
97    # Function to generate confusion matrix plot (CMP)
98    def cmp(cfm, display_labels, title='KNN Confusion Matrix (Averaged)'):
99
100       # Plot confusion matrix using matplotlib
101       d=sklearn.metrics.ConfusionMatrixDisplay(cfm,display_labels=display_labels)
102       d.plot(colorbar=False)
103       plt.title(title)
104       plt.show()
105
106   # Compute confusion matrix averaged & rounded across all runs
107   cfm_averaged = np.round(np.mean(cfm, axis=0)).astype(int)
108
109   # Call confusion matrix function to display results for last run
110   cmp(cfm_averaged, ['Satisfactory', 'Anomalous'])
```

Appendix E. KNN Program

```
# ================================================================================
#
# Machine Learning for Acoustic Anomaly Detection
# K-Nearest Neighbours (KNN) program (version 1.3)
# Alex Renaudin 22 October 2023
#
# This program builds a KNN classifier and records its performance by
# averaging scores across 1000 runs.
#
# Based off YouTube playlist 'Machine Learning with Python' by Sentdex
# https://www.youtube.com/playlist?list=PLQVvvaa0QuDfKTOs3Keq_kaG2P55YRn5v
#
# ================================================================================

import numpy as np
import sklearn.metrics
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import neighbors
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_fscore_support

# Load in dataset
df = pd.read_csv('dataset.csv')

# Assign class features to X and y arrays
X = np.array(df.drop(labels=['class'], axis=1))
y = np.array(df['class'])

# Program parameters:
SPLIT = 0.20        # Proportion of testing set (e.g. 0.20 = 20% testing)
NUM_RUNS = 1000     # Number of iterations of model

# Initialise performance metric variables
accuracy = []                           # Total accuracy score
precision_S = []; precision_A = []      # Precision
recall_S = []; recall_A = []            # Recall
F1_S = []; F1_A = []                    # F1 score
support_S = []; support_A = []          # Support
cfm = []                                # Confusion matrix

# Print parameters
print(f'\n\nRunning numerous KNN models with a {SPLIT:.0%} testing split...\n')

# Iterate NUM_RUNS times, creating and testing new KNN model each time
for _ in range(NUM_RUNS):

    # Randomly separate data into training and testing clumps
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=SPLIT)
    model = neighbors.KNeighborsClassifier() # Creates the model classifier
    model.fit(X_train, y_train) # Fits the training sets to the classifier

    # Compute predicted answers based off testing dataset
    y_pred = model.predict(X_test)

    # Compute performance metrics for this run
    report = precision_recall_fscore_support(y_test, y_pred)

    # Append performance metrics values
    accuracy.append(model.score(X_test, y_test))
```

```
61        precision_S.append(report[0][0])
62        recall_S.append(report[0][1])
63        precision_A.append(report[1][0])
64        recall_A.append(report[1][1])
65        F1_S.append(report[2][0])
66        F1_A.append(report[2][1])
67        support_S.append(report[3][0])
68        support_A.append(report[3][1])
69        cfm.append(sklearn.metrics.confusion_matrix(y_test, y_pred))
70
71    # Print results
72    print('Complete! Mean model performance metrics:\n')
73
74    print(f'\tSize of dataset          = {len(df)} samples')
75    print(f'\tOverall accuracy         = {100*np.mean(accuracy):.2f}%')
76    print(f'\tStandard deviation       = {100*np.std(accuracy):.2f}%\n')
77
78    print(f'\tSatisfactory:  Precision = {100*np.mean(precision_S):.2f}%')
79    print(f'\t               Recall    = {100*np.mean(recall_S):.2f}%')
80    print(f'\t               F1 score  = {100*np.mean(F1_S):.2f}%')
81    print(f'\t               Support   = {round(np.mean(support_S))} samples\n')
82
83
84    print(f'\tAnomalous:     Precision = {100*np.mean(precision_A):.2f}%')
85    print(f'\t               Recall    = {100*np.mean(recall_A):.2f}%')
86    print(f'\t               F1 score  = {100*np.mean(F1_A):.2f}%')
87    print(f'\t               Support   = {round(np.mean(support_A))} samples\n')
88
89    # Function to generate confusion matrix plot (CMP)
90    def cmp(cfm, display_labels, title='KNN Confusion Matrix (Averaged)'):
91
92        # Plot confusion matrix using matplotlib
93        d=sklearn.metrics.ConfusionMatrixDisplay(cfm,display_labels=display_labels)
94        d.plot(colorbar=False)
95        plt.title(title)
96        plt.show()
97
98    # Compute confusion matrix averaged & rounded across all runs
99    cfm_averaged = np.round(np.mean(cfm, axis=0)).astype(int)
100
101   # Call confusion matrix function to display results for last run
102   cmp(cfm_averaged, ['Satisfactory', 'Anomalous'])
```

Appendix F. Risk Analysis

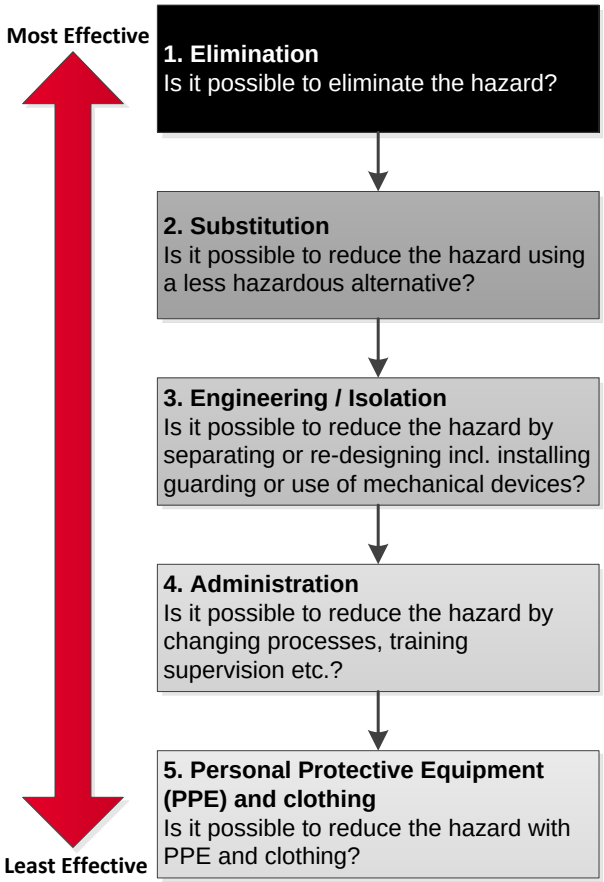
Table 9 below details a risk analysis for the project based off an RMIT template. It qualitatively characterises each risk and includes a control plan. Note that all the risks are associated with the data collection stage of the project (Stage 3), since the remaining stages involve individual computer-based work.

Table 9: Project risk analysis (Source: RMIT).

<table>
<tr><th colspan="5">SECTION 1: GENERAL INFORMATION</th></tr>
<tr><td colspan="2">Lists the main steps in the activity (add more lines as required).
Stage Description
3 Data collection
4 Feature extraction
5 Model development
6 Model testing
7 Results & reporting</td><td colspan="3">Are any of the following licenses / permits required:
☐ Poisons permit covering scheduled drugs / poisons (S4/S7/S8/S9/S10)
☐ Confined space entry permit
☐ Licence for using radiation sources or Genetic Modified Material
☐ Animal Ethics / Human Ethics approval
☐ Other : specify -
If any boxes are checked, specify the applicable permit or licence number:</td></tr>
<tr><td colspan="5">Use the list below to assist in identifying hazards that may be encountered in each step. You must consider all hazards, some of which may not be included in this list. All identified hazards must then be included in Section 4 and controls listed in Section 5.</td></tr>
<tr><td>Plant / equipment</td><td>Animals / Insects</td><td>Biological materials (e.g. bacteria, viruses)</td><td>Confined spaces</td><td>Hazardous chemicals</td></tr>
<tr><td>High pressure / vacuum</td><td>Electrical</td><td>Fumes / vapours / dust</td><td>Manual Handling (lifting, pushing, moving objects)</td><td>Repetitive movements, awkward postures</td></tr>
<tr><td>Mechanical (rotating parts, crush / pinch points)</td><td>Vibration</td><td>Trip / slip hazards / uneven ground or work surface</td><td>Extreme temperatures (High / Low)</td><td>Working at heights or below ground level (e.g. excavations)</td></tr>
<tr><td>Interaction with public (threat to safety)</td><td>Interaction with public (threat to security)</td><td>Overhead power lines</td><td>Working in isolation</td><td>Noise</td></tr>
<tr><td>Radiation (including UV)</td><td>Lasers</td><td>Traffic hazards (vehicles, forklifts, aircraft, drones)</td><td>Gases</td><td>Other</td></tr>
</table>

SECTION 2: RISK SCORE MATRIX							
FACTOR	**CLASSIFICATION**	**VALUE**	**RISK SCORE AND RATING**				
CONSEQUENCES Most probable result of the potential accident.	**Consequence:**	**C**	>500	**Extreme Risk** This activity / process must not be implemented. An alternative must be found.			
	a. Catastrophe; numerous fatalities; major disruption of activities	100					
	b. Disaster; multiple fatalities	50					
	c. Very serious; fatality	25					
	d. Serious; permanent disability	15					
	e. Moderate; serious but non-permanent disability	5	300 – 499	**High Risk** This activity / process must not be implemented without management approval.			
	f. Minor; minor cuts, bruises, burns (first aid treatment required)	1					
EXPOSURE The frequency of exposure to the hazard.	**Exposure:**	**E**					
	a. Continuously (or many times daily)	10					
	b. Frequently (approximately once daily)	6	100 – 299	**Substantial Risk** Unless there is no alternative this activity should not be implemented. Requires management approval.			
	c. Occasionally (from once per week to once per month)	3					
	d. Infrequent (from once per month to once per year)	2					
	e. Rarely (once every 2 - 4 years)	1					
	f. Very rarely (once every 5 years or more)	0.5	50 - 99	**Medium Risk** Hazard must be examined against current standards to determine whether adequately controlled. Requires management oversight.			
PROBABILITY Likelihood that the consequence will occur once the individual is exposed to the hazard.	**Probability:**	**P**					
	a. Almost certain or expected result	10					
	b. Quite possible / not unusual	6					
	c. Would be an unusual sequence or coincidence	3	10 - 49	**Low Risk** Manage by routine procedures. Caution must be observed.			
	d. Would be a remotely possible coincidence	1					
	e. Has never happened after many years of exposure, but is conceivable	0.5					
	f. Practically impossible sequence (has never happened)	0.1					

RISK SCORE CALCULATIONS:	***CONSEQUENCE (C)***	×	***EXPOSURE (E)***	×	***PROBABILITY (P)***	=	***RISK SCORE***

SECTION 3: CONTROLLING THE HAZARDS	
What you should do for each stage of the risk assessment:	**Hierarchy of Control**
For each step in the activity, provide a brief description for each identified hazard and associated risk in **Section 4.** Determine the initial risk rating (i.e. the risk without any controls in place) in Section 4 by referencing the Risk Score Matrix in **Section 2**. Specify the risk **control type** and **control description** for each hazard in **Section 5.** *Notes: Apply the Hierarchy of Controls (opposite) to reduce the level of risk.* *Select the **most effective** controls in preference to **least effective** ones as much as practicable.* *A combination of control measures may be used to reduce risk.* Once controls have been selected, determine the residual risk rating by again referencing the Risk Score Matrix in **Section 2.** Note: Any Residual Risk scores equal or greater than 50 (i.e. medium or higher) **must** be signed off by the Discipline leader before the Activity can be undertaken	

SECTION 4: RISK ASSESSMENT							
Stage # *(from Sect 1)*	**Hazards** *(list the hazards in each step)*	**Associated Risks** *(resulting in damage, injury or illness)*	**Initial Risk Score & Rating (before controls)** *(Refer to risk score matrix in Section 2)*				
			C	**E**	**P**	**Score**	**Rating**
3	Serious misuse of drop saw	Risk of serious injury to operator and bystanders	25	10	0.5	125	Substantial
3	Minor misuse of drop saw	Risk of minor injury	5	10	1	50	Medium
3	Construction noise (incl. drop saw)	Risk of hearing damage	15	6	0.5	45	Low
3	Accident caused by interference of data collection equipment	Risk of serious injury to operator and bystanders	25	10	0.1	25	Low
3	Using saw in isolation	Risk of serious injury without support	25	2	0.5	25	Low
3	Exposure/inhalation of sawdust	Risk of health damage	1	10	1	10	Low

SECTION 5: RISK CONTROL PLAN

Hazard *(from Section 4)*	Risk control measures required to reduce the risk		Residual Risk Score (after controls) *(Refer to risk score matrix in Section 2)*				Residual Risk Rating	Person responsible for implementing controls
	Select risk control(s)	**Details of risk controls** *(provide a brief description)*	**C**	**E**	**P**	**Score**		
Serious misuse of drop saw	☐Elimination ☐Substitution ☐Isolation ☐Engineering ☒Administration ☒PPE	Sufficient training and supervision is required to operate the drop saw or any other power tools. Safety glasses and ear protection and other relevant PPE must be worn at all times.	15	10	0.1	15	Low	Alex Renaudin
Minor misuse of drop saw	☐Elimination ☐Substitution ☐Isolation ☐Engineering ☒Administration ☒PPE	Sufficient training and supervision is required to operate the drop saw or any other power tools. Safety glasses and ear protection and other relevant PPE must be worn at all times.	5	10	0.5	25	Low	Alex Renaudin
Construction noise (incl. drop saw)	☐Elimination ☐Substitution ☐Isolation ☐Engineering ☐Administration ☒PPE	Ear protection will be worn at all times during factory operation.	1	6	0.5	3	Low	Alex Renaudin
Accident caused by interference of data collection equipment	☐Elimination ☐Substitution ☒Isolation ☐Engineering ☒Administration ☐PPE	Recording equipment will be placed safely out of the way of the saw and its operator. Cables will be tied down to prevent entanglement. Workers will be made aware of the equipment.	5	10	0.1	5	Low	Alex Renaudin
Using saw in isolation	☒Elimination ☐Substitution ☐Isolation ☐Engineering ☐Administration ☐PPE	Saws and other power tools will only be used in the company of other people.	25	0.5	0.5	1.25	Low	Alex Renaudin
Exposure/inhalation of sawdust	☐Elimination ☐Substitution ☒Isolation ☐Engineering ☐Administration ☒PPE	Correct use of saws will expel sawdust away from the operator. Additionally, dust masks can be worn to prevent any inhalation	1	1	1	1	Low	Alex Renaudin

Appendix G. Meeting Minutes

Table 10: Legend for meeting minutes.

Full Name	Abbreviation
Dr. Mohamad Fard	MF
Alex Renaudin	AR
Sitthichart (Mark) Tohmuang	ST
Parin Sanpetchnarong	PS
Zezhong Zhang	ZZ
Yuhan Liu	YL

Table 11: Meeting minutes log with attendees, notes and actions.

Date & Attendees	Notes	Actions

15/03/23
 	
AR, ST, PS, ZZ, YL
	• Met on teams for 30 mins to catch-up and discuss initial stages of the project
	• Parin uploaded YouTube playlists to teach the basics
	  ○ Watch these to explain anything that is unclear in the literature
	  ○ Might be useful to watch before the literature review
	• Mark proposed going through content together F2F
	  ○ ~1 hour session
	  ○ Monday in place of meeting?
	• Key terms:
	  ○ Acoustic features
	  ○ Signal processing
	  ○ MFCC (Mel Features Cepstral Coefficient)
	  ○ Noise classification using machine learning
	• 4 papers to read
	• 1 MATLAB example
	• Lab equipment ready to record metal rattle/squeak sounds to form dataset
	• Look for more papers using key terms above
	• Report on Monday w/ findings - 2 pm F2F
		1. 1st goal: collect data. Focus on feature extraction from engine sound recordings
		2. Work on literature review
		3. Ask Craig for dataset ideas

20/03/23 AR, ST, PS, ZZ, YL	• Met up F2F for 2 hours to catch-up and discuss initial stages of the project • Each went through literature and presented our findings + our knowledge up to date • Presented online audio datasets we have found to date. o YL & ZZ found good dataset for engine knocking sounds	1. Take notes on papers we read: type of ML model used, data processing techniques used, etc. 2. Prepare literature review for proposal

28/03/23
 	
AR, ST, PS, ZZ, YL
	• Doesn't matter so much which ML model is used at this stage
	  o Acoustic features is more important, the model is more dependent on this
	• In terms of the proposal, focus more on the literature review, dataset and preliminary on the acoustic features
	  o Can ask Parin for help with the proposal
	• Parin & Mark green-lighted the saw performance diagnosis idea
	• It's okay to change scope down the track
	  o Further, it is okay to 'fail' and record that data
	• 'Two approaches: be the best or be the first, and being the first is easiest
		1. Action: send through datasets from literature

04/04/23
 	
MF, AR, ST, PS, ZZ, YL
	• Dr. Fard joining to discuss our progress to date
	• Presented my proposal to the group (excl. Dr. Fard)
	• Feedback was overall positive
	  o Okay to leave two goals (model building + device design), and leave secondary goal as a stretch goal
	  o Most important factor will be data collection, so focus on how to optimise that process as it will save time down the track
		1. Action: email Neng for the link to the risk assessment that we need to perform
		• CC Dr. Fard and two team members
		• Look on Canvas for the link first

20/04/23
 	
AR, PS, ZZ, YL, MF
	• Meeting with Parin & Dr. Fard to go over progress to date
	  o Plan to go over programming side (Python) to do audio processing basics
	• Presented my sample files so far and told of plans to continue to collect data
	• Best way to organise files is just time for number, sorted between good and bad
	  o Think about best way to organise files
	  o Research for good protocols, etc. online
	• Start working on PowerPoint for end of semester ASAP
	  o Keep updating as time goes on, not at the last minute
	  o Present weekly in meetings
	• Continue taking notes of progress, work and meetings
	  o This will be necessary for end-of-semester submission
		1. Action: look at assignment briefing and prepare template for presentation

05/05/23 AR, ST, PS, ZZ, YL, MF	• Don't collect too much data without visualising it • Work on the Python code first - focus on feature extraction • Make sure every sample is normalised with the same:	1. For next week: • Focus on ~10 samples • Visualise the samples - MFCC spectrogram

	○ Sample rate ○ Volume ○ Length ○ Quality ○ etc. • Whether to keep the start/end of data (i.e. saw accelerating and decelerating but **not** cutting) ○ Mark: try both ways ○ Keep the data separate and try it when training the model ○ Label it with a tag	• Show good vs. bad • Show on PowerPoint & present to the team
16/05/23 AR, ST, PS, ZZ, YL, MF	• Mark showed how to work with basic ML in Python • Reduce to 8(?) features - one for each Mel band ○ Use a **statistical indicator** to 'average' the values ▪ Mean may not be the best function to use ▪ Mark said there will be a paper posted with a good technique ▪ Research and trial different methods • Principal Component Analysis (PCA) to reduce higher-dimension MFCC features ○ For visualisation purposes • Look for YouTube ML Python playlist to cover the code (example) • 'Seaborn' module in Python is an advanced data visualisation module • Aim to include ML model into thesis, but no stress if it's too much ○ Just include basics - don't worry about optimising it just yet • Dr. Fard RE structure: ○ Leng has posted one on the Teams discussion ○ Introduction ○ Background study / literature review ○ Aim (one key aim) ▪ Objectives (sub-aims) ▪ Research questions ▪ 'How can we extract X?' ○ Method ○ Results ○ Conclusion • Figures should be clear (w/ captions - can be long if needed). Should understand without context	1. Send a format skeleton format to Dr. Fard to check the structure 2. For the presentation: Ignore brief to some extent! Show introduction, aim, method, results and discussion
25/05/23 AR, ST, PS, ZZ, YL, MF	• Final presentation pencilled in for Tuesday May 30th 10-11 ○ Reach out to Dr. Fard by Friday arvo / Saturday (27/5) if not confirmed by then ○ Present online via Teams with a PowerPoint to support ○ 5-6 minute live presentation ▪ Pre-recorded presentation (the one uploaded to Canvas) **exactly** 5 minutes	1. Reach out to Dr. Fard RE presentation date if needed 2. Semester 2: investigate the effect of changing:

	• Presentation tips:
	  ○ Use RMIT theme
	  ○ Bigger font
	  ○ Figure captions (why is it there?)
	  ○ Slide numbers
	  ○ ‘Key message’ emphasised on each slide (i.e. on bottom)
	• Upload presentation & thesis w/ a password when uploading to Canvas
	  ○ Not needed for when emailing directly to Dr. Fard as main submission
	• Python feature extraction tips w/ Mark:
	  ○ Calculate mean (& RMS?) directly with call such as ‘mfccs.mean()’
	  ○ Add final column with label to indicate ‘good’ or ‘bad’
	  ○ Can be combined with ‘concat’ call
	• Decrease N_MELS to be lower
	  ○ Dr. Fard suggested ~32, 64
	  ○ Mark suggested lower: 8, 16
	  ○ Leave code to be customisable and test later
	  ○ Error at 256+ may be due to hop size and FFT process
		3. N_MELS value (i.e. in range 8 —> 512)
		4. Statistical indicator (i.e. the 10 used by Abeysinghe et. al)
		5. Keep the model simple and consistent for these tests to focus on **only** the feature extraction process

27/07/23
 	
AR, ST, PS
	• Met with Parin & Mark to discuss scope for semester 2
	• Choice between specialising on a focus (i.e. deep learning) or more general ML knowledge
	  ○ Nominated the latter → more general approach
	  ○ This choice has more of a focus on the application (AWS, MCU)
	  ○ “Output oriented”
	• Mark: learn how to use ML in basic classification tasks
	  ○ Scope for the next few weeks
	  ○ Narrow down to sound context where possible
	  ○ Use WT examples from last semester as a reference
	• Parin: SVM for next week
	  ○ Will send through YouTube playlist as a reference
	  ○ Focus on the model by itself rather than the dataset
	• Dr. Fard away due to family matter - return date TBC
	• Mark mentioned plans to write a comprehensive paper on AAD using ML
	  ○ Extra work on top of subject
	  ○ Will discuss at later date
		1. SVM model to be built for next week
		2. Confirm choice of scope for semester

03/08/23 AR, ST, PS	• Met with Mark, Parin. Dr. Fard away (flying home today or tomorrow) • Email Dr. Fard to explain first assignment and ask what he would like me to present for it • Parin suggested some good YT playlists and will send after meeting • Mark advice: focus on A to B, don’t get too overwhelmed or distracted by the large amount of resources out there	1. Email Dr. Fard to explain first assignment task and request format 2. Develop presentation of work for the week

	○ Focus on achieving the output, otherwise will run out of time • Parin: For next week: develop a presentation ○ Present what the scope of work is ○ Don't worry too much about the model just yet	
17/08/23 AR, ST, PS	• Met w/ Mark & Parin to discuss progress up to week 5 • Fixed issue with integer/float64 data type: ○ Saves automatically as integer in KNN program even though it appears as a float ○ Suggestions if breaks in future include saving as a dictionary or as a pickle (JSON format) • Mark suggested to focus on deployment first, and then come back and look at model development and optimisation ○ He explains that this is a more important part of the ML process to learn as opposed to optimisation / development, as it would be a shame to get stuck and not complete deployment • Suggestions for deployment: ○ Python flask ○ AWS ○ MCU (stretch goal)	1. Investigate options for deployment
24/08/23 AR, ST, PS, MF	• Met with Dr. Fard, Parin, and Mark to discuss MCU deployment • Wanted to get Neng in the call to ask his advice but busy - Parin to call after • Informed them of my idea to use Arduino Nano • Showed Dr. Fard results of parameter testing ○ He suggested consulting literature for similar work, and then applying a similar process to my own work ○ This would be a strong piece of work that could be published ○ Doesn't need to be the first, but is good when it is specific and detailed • Parin reminded to document everything when doing the experimental work (both deployment and feature extraction) • Wait for Neng to get back to Parin regarding purchasing equipment for deployment	1. Progress with purchasing request following resolution w/ Neng
14/09/23 AR, ST, PS	• Caught up with Parin and Mark to discuss RasPi progress • Mark suggested uploading all the code and project I have done on GitHub and documenting it well. This can be added to a CV & LinkedIn as part of a portfolio for work • Parin proposed the following process for implementing the model on the RasPi: 1. Run the ML model on Raspberry Pi 2. Collect data with Raspberry Pi	1. Investigate RasPi implementation as described by Parin 2. Plan out portfolio of work
20/09/23 AR, ST, PS	• Met w/ Mark and Parin • Explained my progress with the RasPi so far:	1. Develop presentation of work for next week

	○ Set up RasPi successfully, however could not update Python or install SKlearn ○ Shared next step plans to try with TensorFlow using their tutorials on YT • Parin emphasised the importance of documenting the process ○ Especially the documentation of setting us the Raspberry Pi ○ Even the failures can be documented - especially useful for the final submission • Parin suggested presented weekly presentation to the team to showcase progress ○ Good motivation for progress ○ Showcase all work - even the failures • Mark suggested using laptop microphone to run program directly on laptop using TensorFlow ○ Treat this is a proof-of-concept for later deployment on the microcontroller	2. Document process of RasPi implementation 3. Investigate use of laptop microphone w/ Python
28/09/23 AR, ST, PS, MF	• Met with Mark, Parin and Dr. Fard • Told them about the errors I encountered trying to install TensorFlow on the RasPi ○ Decided to pause efforts with the RasPi since no progress is being made ○ Instead, I have been focused on developing model in TensorFlow on my laptop using Google Colab • Parin suggests setting up TensorFlow on Spyder and trying to run the model in realtime on my computer with the in-built microphone • Mark and Parin gave a run-through of confusion matrices, ROCs, F1, etc. to characterise the performance of the model ○ They suggested I implement on the linear classifier model and continue to do so with any neural networks • Parin sent through links for performance metrics and one-class SVM	1. Add performance metrics to code (F1 score, recall, etc.) 2. Assess model performance and look for improvements
12/10/23 AR, ST, PS	• Met w/ Mark and Parin • Discussed the final assessments for the semester: poster and thesis • Parin highlighted the need to keep the poster sparse and simple, and ensure it is aimed for the lay-person not versed in Machine Learning etc. ○ Showed his poster presentation from his own Capstone project on wave energy generator ○ Suggested highlighting the project objectives • Parin showed me how to test a range of C values for the SVM hyper-parameters ○ Tests for a customisable range of values to see the effect on overall accuracy of the model ○ Noted that increasing above 1000 has little effect ○ Noted a generally positive correlation between increasing the C value and model performance	1. Produce poster 2. Start work on the thesis 3. Send a draft copy to Dr. Fard ahead of the due date

Appendix H. Project Poster

Machine Learning for Acoustic Anomaly Detection

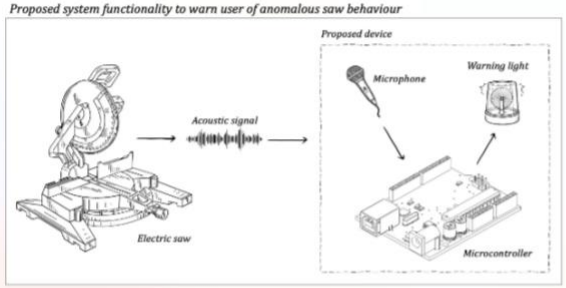
Alex Renaudin

Introduction

Ensuring user safety during power saw use is an important, yet inherently difficult, task. Machinery faults are primarily identified aurally by the trained ear of the human operator. However, this relies on operator experience without room for human error. Thus, the question was posed:

How can machine learning automate acoustic anomaly detection?

Proposed system functionality to warn user of anomalous saw behaviour



Project Description

- Existing work using machine learning for acoustic fault detection focuses largely on automotive applications.
- Very few models are deployed and tested in the field.
- The aim is to develop a machine learning solution to perform acoustic anomaly detection on an electric saw.
- The machine learning model needs to successfully classify the two types (classes) of saw cuts.
- A deployable microcontroller solution is proposed.

Class 1: Satisfactory

Class 2: Anomalous

Gantt chart of project timeline showing key stages

No.	Stage	Semester 1					Semester 2		
		Mar	Apr	May	Jun	Jul	Aug	Sep	Oct
1	Literature review								
2	Research proposal								
3	Data acquisition								
4	Feature extraction								
5	Model development								
6	Model testing								
7	Results & reporting								

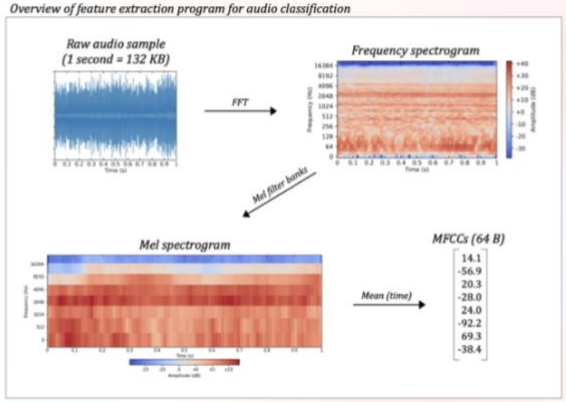
Midyear break

Dataset & Feature Extraction

A novel dataset was needed train the model, which was collected experimentally via field recordings of the saw. A custom program was developed to process the audio and extract key features, thus reducing the size and complexity of the raw data.

The feature extraction program pre-processes the audio into second-long clips and converts the signal into the frequency domain via a fast Fourier transform. Then, the Mel frequency cepstrum coefficient (MFCC) algorithm is employed, which utilises mathematical transforms to represent the audio as a compact numerical array.

Overview of feature extraction program for audio classification



Results

- Collected a high-quality experimental dataset of saw cut audio recordings.
- Dataset includes strong representation of anomalous samples (18%).
- Development of a custom feature extraction program to characterise audio.
- Three types of machine learning architectures explored:
  1. Support Vector Machine (SVM)
  2. K-Nearest Neighbours (KNN)
  3. Linear Classifier
- Final model using SVM algorithm with 20% testing split on dataset.
- Iterative optimisation of both programs to improve overall performance.
- Investigated deployment using Arduino and Raspberry Pi microcontrollers.

Model Performance

Model performance was averaged over 1000 runs and assessed by the following:

- **Accuracy** – percentage of correct predictions.
- **Precision** – accuracy of positive predictions.
- **Recall** – proportion of actual positives identified.
- **F1 score** – measure of both precision and recall.

Average **overall model accuracy was 97.98%** with a standard deviation of 1.48%.

Class	Precision (%)	Recall (%)	F1 score (%)	Support (samples)
Satisfactory	98.14	96.90	98.84	66
Anomalous	99.57	87.48	91.52	10

SVM Python program output

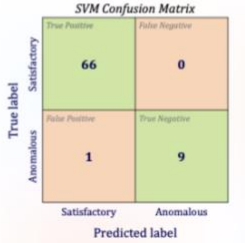
```
Running 1000 SVM models with a 20% split...
Complete! Mean model performance metrics:

  Size of dataset      = 377 samples
  Overall accuracy     = 97.98%
  Standard deviation   = 1.48%

  Satisfactory: Precision = 98.14%
                Recall    = 96.90%
                F1 score  = 98.84%
                Support   = 66 samples

  Anomalous:    Precision = 99.57%
                Recall    = 87.48%
                F1 score  = 91.52%
                Support   = 10 samples
```

SVM Confusion Matrix



Conclusions

Overall, the SVM model demonstrated strong performance in identifying anomalous saw cuts, suggesting high suitability for safety and maintenance applications. Deployment onto a microcontroller was ultimately unsuccessful due to budget and time constraints, however, this remains a promising opportunity for future work.

Some areas of weakness and/or difficulty included:

- Model installation was unsuccessful on Raspberry Pi MCU.
- Slightly inferior performance classifying anomalous samples as compared to satisfactory samples.
- Results may not be extendable to different saw set-ups due to dataset homogeneity.

RMIT UNIVERSITY

Figure 32: Poster submission for research project.