
American University of Sharjah

Department of Mathematics and Statistics

STA 401: Introduction to Data Mining

Spring 2025

# Acoustic-Based Fault Detection in Electric Engines Using Machine Learning

Submitted to: Dr. Ayman Alzaatreh

| | |
|---|---|
| Abdullah Salmeh | b00093434 |
| Mohamad Chehab | b00090578 |
| Yousef Irshaid | b00093447 |

Date of Submission: 04/05/2025

## Table of Contents

## List of Figures

## List of Tables

## Abstract

This project aims to explore a predictive maintenance approach that uses audio recordings to classify the condition of electric engines as either good, broken, or under heavy load. Using the IDMT-ISA-Electric-Engine dataset, we implement four distinct feature extraction techniques: a classical Amplitude Spectrum method and three popular pre-trained neural network models (VGGish, OpenL3, and Wav2Vec 2.0) to convert raw audio into information-rich numerical representations. After applying Principal Component Analysis (PCA) for dimensionality reduction, we trained and evaluated ten different machine learning classifiers, including logistic regression, discriminant analysis, ensemble methods, SVMs, and decision trees. Model performance is assessed using accuracy, precision, recall, F1-score, and confusion matrices. Results show that OpenL3 consistently delivers the highest performance, while amplitude spectrum features remain competitive when used with QDA. This work demonstrates a practical and non-invasive diagnostic tool for engine monitoring and contributes to advancing audio-based fault detection for predictive maintenance in industrial settings.

# Introduction

## Background

Predictive maintenance is an approach to industrial machine maintenance that uses sensor data and analytical algorithms to monitor the health of the equipment to help anticipate potential failures, minimize downtime, and ensure safety [1]. Other strategies include reactive maintenance and preventive maintenance. Reactive maintenance involves repairing the equipment only after failure has occurred, whereas preventive maintenance relies on fixed time or usage intervals for servicing regardless of the actual condition of the equipment. These traditional approaches can be inefficient and costly, as they often either result in unexpected downtimes or unnecessary part replacements [2]. For instance, scheduled preventive maintenance is typically based on conservative worst-case lifetime estimates and does not consider factors such as load variations, improper usage, mounting configurations, or bearing wear. As a result, components may be serviced too early or fail prematurely due to unmonitored degradation. Poor communication of user errors, such as crashes or incorrect handling, further exacerbates the problem by causing unplanned outages that can significantly hinder production [2]. Unlike these methods, predictive maintenance continuously assesses the condition of the machines to perform maintenance only when it is needed. [1] added that machine condition monitoring plays a crucial role in maintenance since it requires monitoring the state of the machines during the operation and detecting early signs of faults or any performance degradation. Real-time anomaly detection allows operators to intervene before minor issues escalate into major failures.

## Motivation

An emerging aspect in condition monitoring is the use of sound and acoustic signals for non-invasive diagnostics [3]. Studies have shown that mechanical faults can be detected by vibration or sound analysis. For example, worn bearings can produce a grinding noise and overloaded motors can run at a different frequency or pitch. The author in [3] emphasizes that acoustic signals can be critical for the detection of faults in rotating motors since they contain a rich amount of diagnostic information. An advantage of acoustic monitoring is that it is non-contact since microphones can detect the machine sounds from a safe distance without having physical contact with the equipment [3]. This can help reduce costs and simplify deployment in environments where attaching physical sensors is difficult or unsafe.

## Problem Statement

Using audio data in industrial settings provides a non-invasive, real-time view of machine condition, and complements traditional methods like vibration analysis or thermal imaging. Acoustic diagnostics can be performed continuously and inexpensively with simple microphones, potentially covering multiple machines with one sensor. The main challenge to consider is that industrial environments are generally noisy since background sounds from other machines can interfere with the target engine's sound. However, advanced signal processing techniques can help

isolate the machine's signature. Overall, the ability to listen to machines offers a convenient way of detecting anomalies without interrupting the machine's operation. In this project, we leverage this approach by analyzing sound recordings of electric motors to classify their operational state. This aligns with the Industry 4.0 trend of IoT-based monitoring, where audio sensors combined with AI enable automated fault detection on the factory floor [1].

## Literature Review

There has recently been an increase of research on using acoustic signals for machine fault detection and conditional monitoring. Previous studies have shown that sound-based analysis can be effectively implemented to detect industrial equipment faults including engines. This was noted in [3] where the author investigated acoustic fault detection of electric motor commutators where audio signals were used to reliably indicate broken coils and gear damage. The study used a pattern recognition approach where audio recordings were first preprocessed through normalization and filtering, then transformed to the frequency domain using the Fast neural nFourier Transform (FFT), and finally the features are extracted for classification using algorithms such as the Nearest Neighbor (NN) classifier and Backpropagation Neural Networks (BNN). Results of this study demonstrate that certain motor faults will produce distinguishable audio patterns.

Researchers have used established audio features from speech and music processing to analyze and classify mechanical sounds. Mel-Frequency Cepstral Coefficients (MFCCs) are widely used in speech recognition and have been successfully applied for mechanical sound classification [4]. MFCCs provide a compact representation of an audio spectrum by taking complex audio signals and summarizing them using fewer numbers. Instead of analyzing every frequency equally, MFCCs emphasize the ones most important to human hearing, making the data smaller and more meaningful. Additionally, MFCCs are consistently effective in various audio-related tasks like speech recognition, music classification, and machine fault detection [4]. The study in [4] used MFCC features with a Gradient Boosted Decision Tree classifier to detect conveyor belt idler falter from sound. They achieved an accuracy of 94.5% which suggests that the MFCC-based feature sets were effective for their classification of machine fault detection. Another recent study done on vehicle engine fault detection compared MFCCs with wavelet-based features and found that MFCCs provided superior accuracy in identifying engine anomalies [5]. These studies have shown that MFCCs and similar features are good at identifying the unique sound patterns that tell us whether a machine is working normally, overloaded, or broken.

Other time-frequency representations beyond MFCCs, such as spectrograms, are also widely used in deep learning models. The log-Mel spectrogram, which maps the frequency axis to the perceptual Mel scale and expresses power in a logarithmic scale, is a popular input for convolutional neural networks (CNNs) in audio classification tasks [1]. Like MFCCs, log-Mel spectrograms are extensively used in modern deep learning frameworks for applications such as speech recognition, music classification, and environmental sound detection. In the context of machine condition monitoring, researchers have successfully used spectrogram-based CNNs to

automatically learn patterns that distinguish normal from faulty machine sounds. For example, a relevant study in [1] implemented a convolutional recurrent neural network (CRNN) on a microcontroller to classify the operational state of a DC motor using audio data. The audio recordings were transformed into 32×30 log-Mel spectrogram slices, which were continuously processed by the CNN to output the motor's operational state in real time. This approach, referred to as edge AI, enables real-time inference directly on low-power embedded devices without relying on a cloud connection. The results demonstrate that this method achieved both timely and accurate state recognition, highlighting the power of Mel-scale representations in precise audio-based classification tasks.

The techniques used in this project aim to draw on broader audio classification research beyond industrial case studies. Research in speech recognition has proposed MFCCs and Hidden Markov Models as standard approaches, where many of these principles can be adapted to classifying engine noises. Similarly, environmental sound classification such as urban sound recognition often uses supervised learning on audio features like MFCCs, zero-crossing rate, and spectral roll-off. Classical machine learning models including SVM, random forest, etc. have shown strong performance on datasets like UrbanSound8K by using those features. For instance, a baseline algorithm in urban sound tasks achieved improved accuracy on low-noise clips, leveraging the robustness of MFCC features [6]. The common thread in these studies is that careful feature extraction from audio greatly improves classification accuracy, whether the target classes are spoken words, musical genres, or machinery conditions. In our project, we will similarly extract informative acoustic features and apply a range of classifiers. Prior research suggests that this combination can successfully distinguish between subtle differences in sound that correlate with the condition of the mechanical device. For example, a "broken" engine may have extra noise components or different frequency peaks as compared to a "healthy" engine [3]. Overall, the literature provides a strong foundation indicating that audio-based fault diagnosis is feasible and can achieve high accuracy when utilizing the right features and models. Furthermore, the existing literature supports the idea that established sound analysis techniques from fields like speech and music processing can be effectively adapted to the unique challenges of industrial noise and fault detection.

## Data Description

The dataset used in this project is the IDMT-ISA-ELECTRIC-ENGINE dataset, which contains audio recordings of a 2ACT Motor Brushless DC 42BLF01 engine operating under three distinct conditions: “good,” “heavy load,” and “broken.” The goal is to classify the engine’s operational state by analyzing these sound signals, which are informative indicators of performance and potential faults.

The dataset consists of 2,378 audio clips, each 3 seconds long, recorded at a 44.1 kHz sampling rate with 32-bit resolution. The class distribution is as follows:

*Table 1: Label description and class distribution of the dataset*

| Label | Description | Count |
|---|---|---|
| Good | Engine operating normally | 774 |
| Heavy Load | Engine under mechanical or electrical stress | 815 |
| Broken | Engine exhibiting fault symptoms | 789 |

The states were created by varying the engine's voltage and load, which simulates real-world conditions where engines may experience stress or wear.

Furthermore, the dataset is pre-divided into training and testing subsets. The training set consists exclusively of pure engine recordings (no additional noise). In contrast, the testing set contains recordings under various background noise conditions, providing a realistic challenge that mimics actual industrial environments. The noise types include:

- talking – people speaking around the engine casing
- white_noise – white noise played through external speakers
- atmo – factory atmospheric noise at three loudness levels (low, medium, high)
- stress test – recordings with altered input gains and physical interactions like knocking on the casing

To further illustrate the structure of the dataset, we present bar plots showing the distribution of the three engine states in both the training and testing sets.


*Figure 1: Class distribution of the training data*

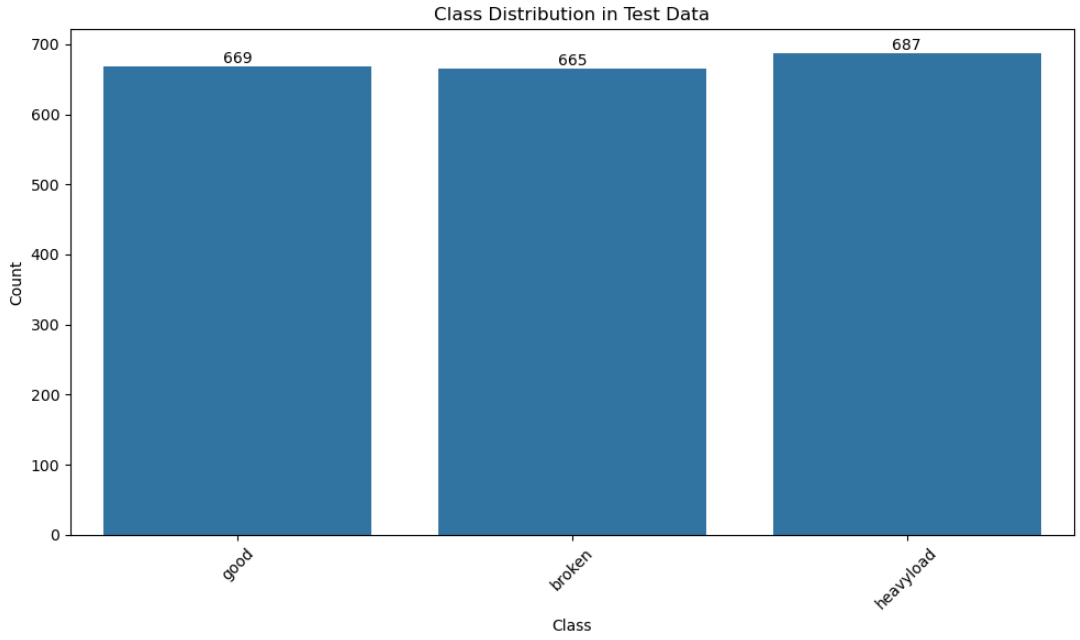


*Figure 2: Class distribution of the testing data*

We can observe that the distribution of the 3 classes is almost equal for both training and testing data. We also visualized the test data noise type distribution in the bar plot below.

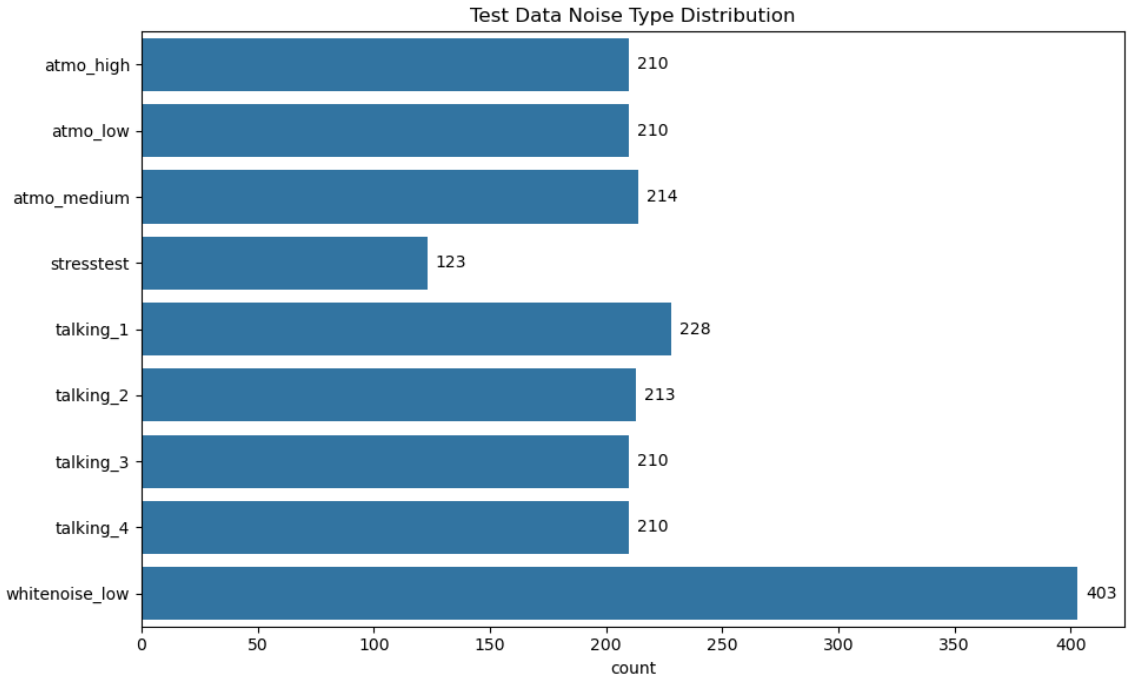


*Figure 3: Bar plot of the testing data noise type distribution*

The noise levels are mostly uniformly distributed, except for 'stresstest,' which is the least common, and 'whitenoise_low,' which occurs twice as frequently as the other types.

To better understand the audio data, we present visualizations about a sample from each class of the dataset. We first show the raw audio waveform (amplitude vs time). This form of data is the simplest form but is usually not useful for machine learning tasks because it has a huge number of data points per sound file (3sec at 44kHz = 132,000 data points). We then present the amplitude spectrum of the signal, which shows the signal in the frequency domain. This form might be less intuitive for humans but is a better candidate for machine learning algorithms. The amplitude spectrum shows the sinusoidal amplitudes present in the signal versus the frequency of the sinusoids using the Fourier Transform. However, the main drawback of the amplitude spectrum is that it forgoes the temporal data in the signal. That is, it does not have any information about the time at which the specific frequencies are present in the signal. To mitigate this, the Short Time Fourier Transform (STFT) spectrum is used. It is a three-dimensional plot that has frequency on the vertical axis, time on the horizontal axis, and color to represent the intensity of the frequency. It does so by applying Fourier Transform on a small region in a sliding manner. However, STFT has much information that might overwhelm the machine learning models. Therefore, a compressed version of the STFT called log-Mel-spectrogram is also shown. The log-Mel-spectrogram applies a perceptual scaling to the short-time Fourier transform (STFT) using filter banks that approximate the frequency sensitivity of the human auditory system. Consequently, if two sounds are distinguishable to human listeners, it is reasonable to expect that a machine learning model can also learn to distinguish them.

“Good” class:

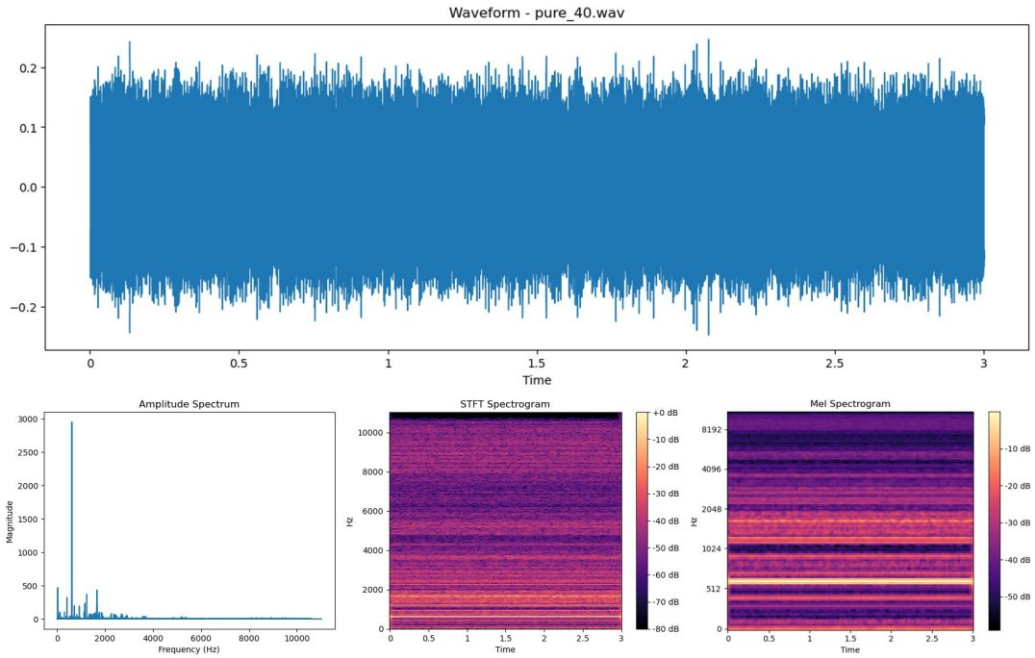


*Figure 4: Time domain waveform, amplitude spectrum, STFT spectrogram, and log-Mel spectrogram of "good" class sample*

“Broken” class:

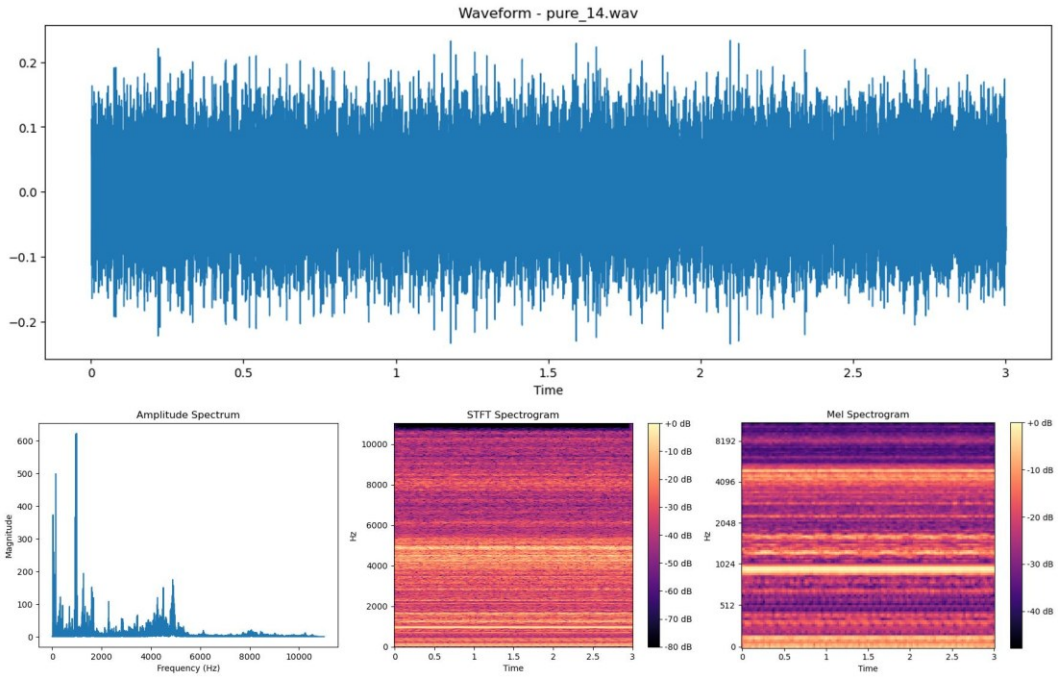


*Figure 5: Time domain waveform, amplitude spectrum, STFT spectrogram, and log-Mel spectrogram of "broken" class sample*

“Heavy load” class:

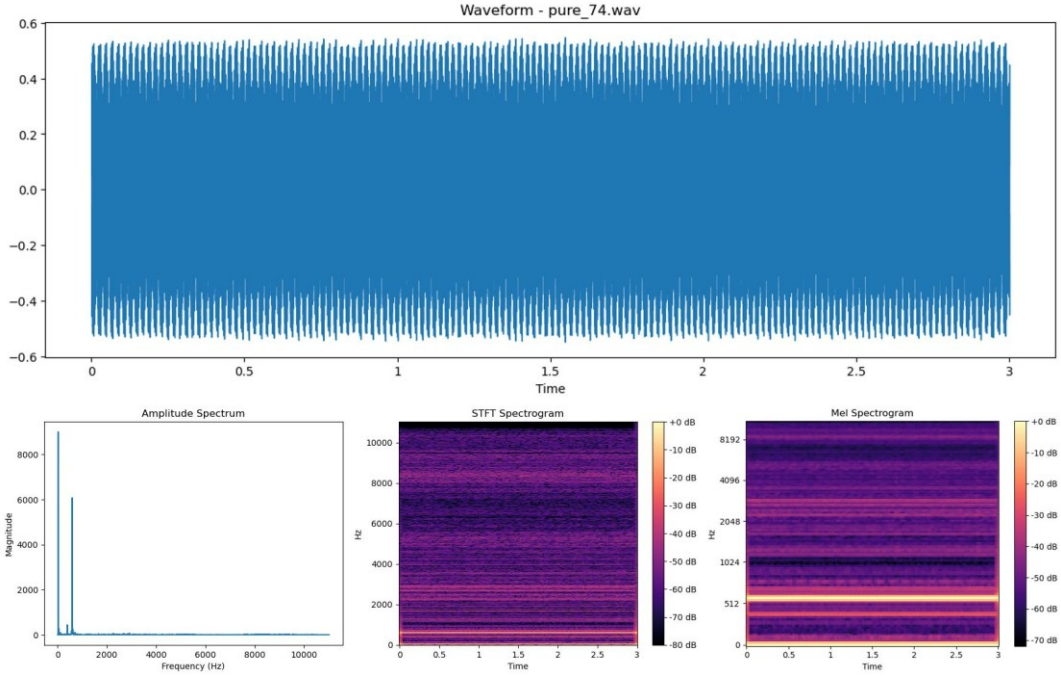


*Figure 6: Time domain waveform, amplitude spectrum, STFT spectrogram, and log-Mel spectrogram of "heavy load" class sample*

We can clearly see a difference at the spectra level and so we expect the machine learning models to perform relatively well. Additionally, we visualize the features extracted using the various methods explained in the Methodology section using PCA, t-SNE, and UMAP. t-Distributed Stochastic Neighbor Embedding (t-SNE) is a statistical technique that is used to visualize high dimensional features into two or 3 dimensions for visualization [6]. It is better than PCA because it is not restricted to linear transformations, and hence it is expected to work better for nonlinear and complex data. Uniform Manifold Approximation and Projection (UMAP) is a nonlinear dimensionality reduction technique that, like t-SNE, is well-suited for visualizing high-dimensional data [7]. However, unlike t-SNE, it is a general dimensionality reduction technique making it applicable to a broader range of dimensionality reduction tasks, including those typically addressed by linear methods such as PCA.

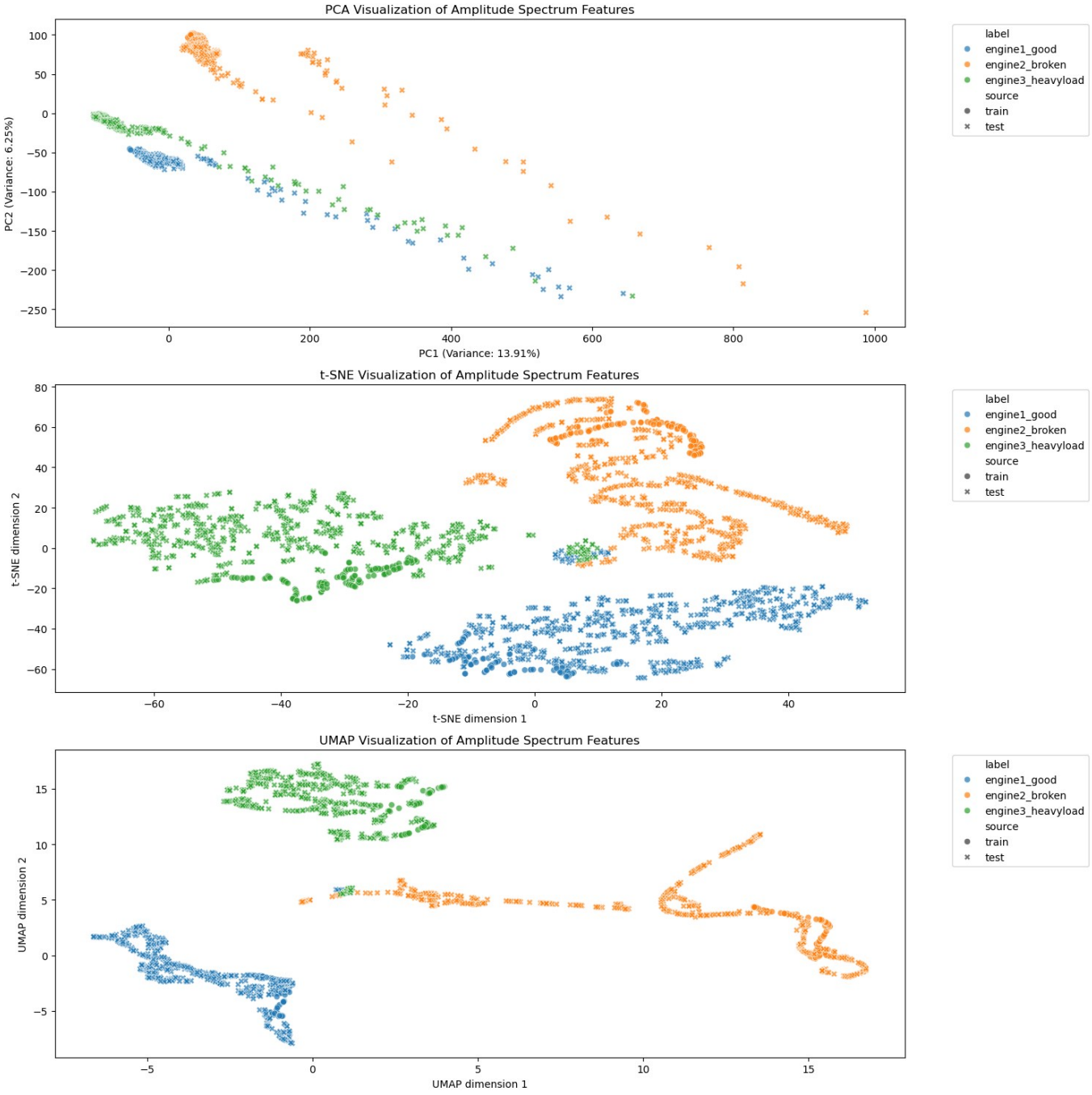


*Figure 7: 2-D PCA, t-SNE, and UMAP visualizations of the amplitude spectrum features*

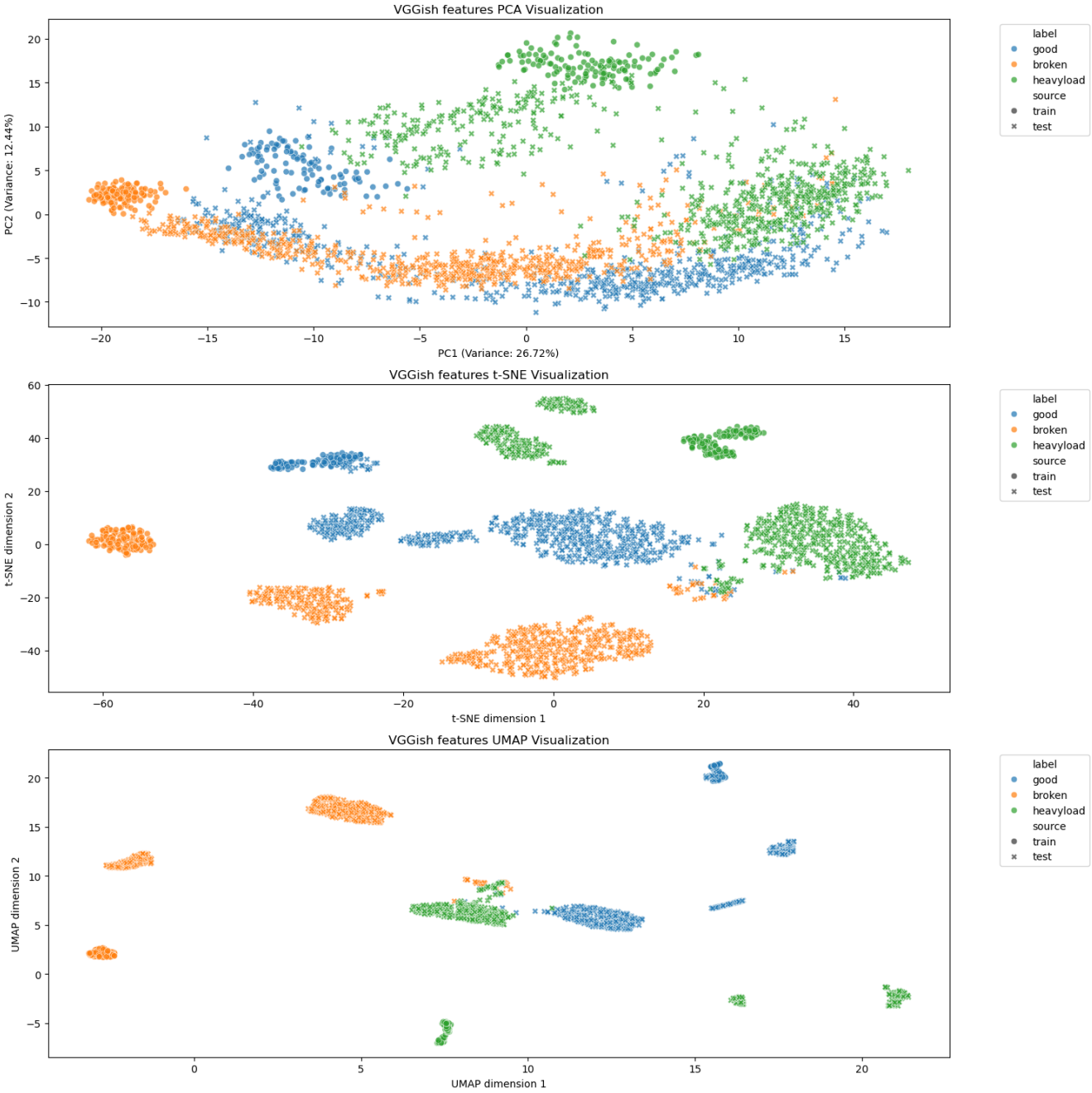


*Figure 8: 2-D PCA, t-SNE, and UMAP visualizations of the VGGish features*

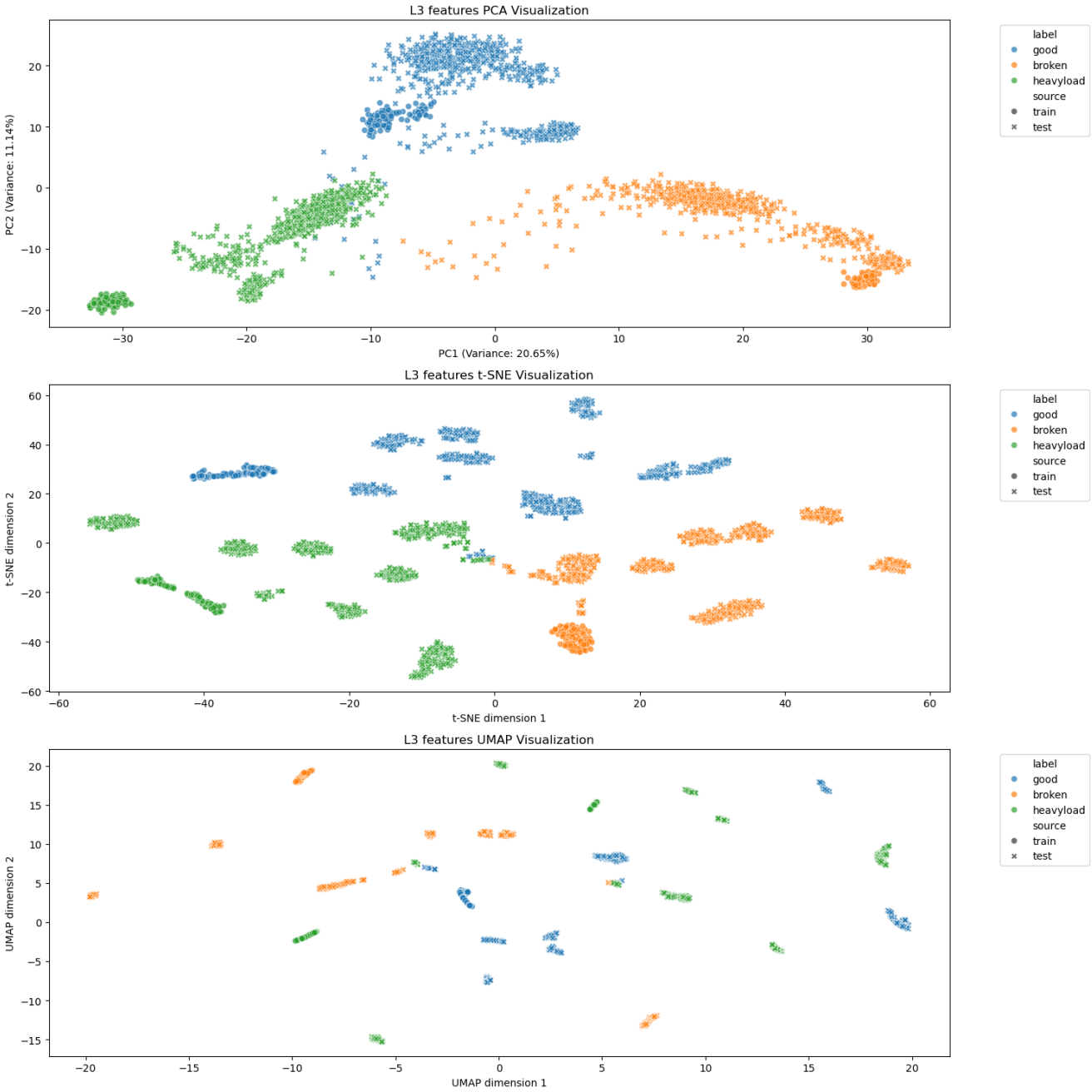


*Figure 9: 2-D PCA, t-SNE, and UMAP visualizations of the OpenL3 features*

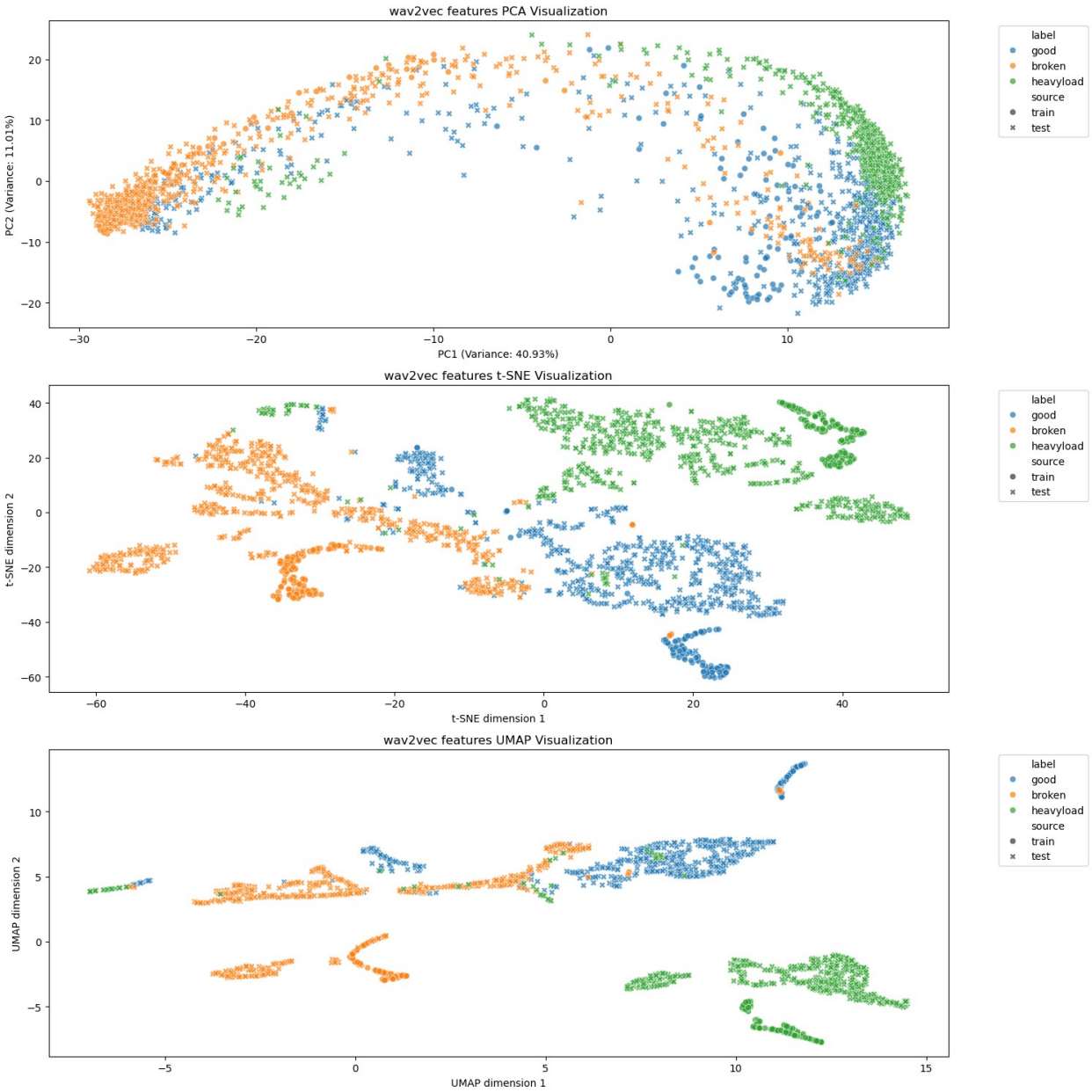


*Figure 10: 2-D PCA, t-SNE, and UMAP visualizations of the Wav2Vec features*

Note the difference between the linear technique PCA and the nonlinear t-SNE and UMAP. It can be clearly seen that the nonlinear techniques offer better separation between the classes across all feature types by mapping them into distinct regions, while PCA often has overlapping data points. Another observation is that the training data points are clustered in compact regions while testing data are scattered outside these clusters. This is likely because the testing data has noise added to it, when training data is pure sound. This problem will lead the classifiers to perform well on the training data but very poorly on the testing data because they were not trained on that domain of input. This problem will be addressed in the Methodology section.

## Methodology

The methodology employed in this project follows a structured machine learning pipeline that starts with the raw audio data and ends with the evaluation of several classification models. Our goal is to accurately classify different engine sounds into three operation states ("good", "broken", or "heavy load"). The process is divided into stages including data preparation, audio preprocessing and feature extraction, dimension reduction using Principal Component Analysis (PCA), model training and cross validation, and performance evaluation.

### Data Preparation

Since the training dataset initially consisted solely of clean audio samples, a domain gap arose when testing on samples that contained background noise. To address this mismatch, we augmented the training data by injecting noise, thereby making it more representative of the testing conditions. As the exact noise sources in the test set are unknown, we approximated them by introducing five common noise types: Gaussian noise, conversational (talking) noise, crowd noise, atmospheric noise, and traffic noise. Except for Gaussian noise, we sourced four audio samples for each noise type from Freesound.org. During augmentation, each original audio sample was combined with a randomly selected noise sample from each of the five categories. For each combination, a Signal-to-Noise Ratio (SNR) was randomly chosen between 10 and 20 dB to modulate the noise level. This augmentation strategy not only reduced the domain gap but also increased the size of the training dataset by a factor of five. The class distribution of the training data after adding noisy samples is shown below.

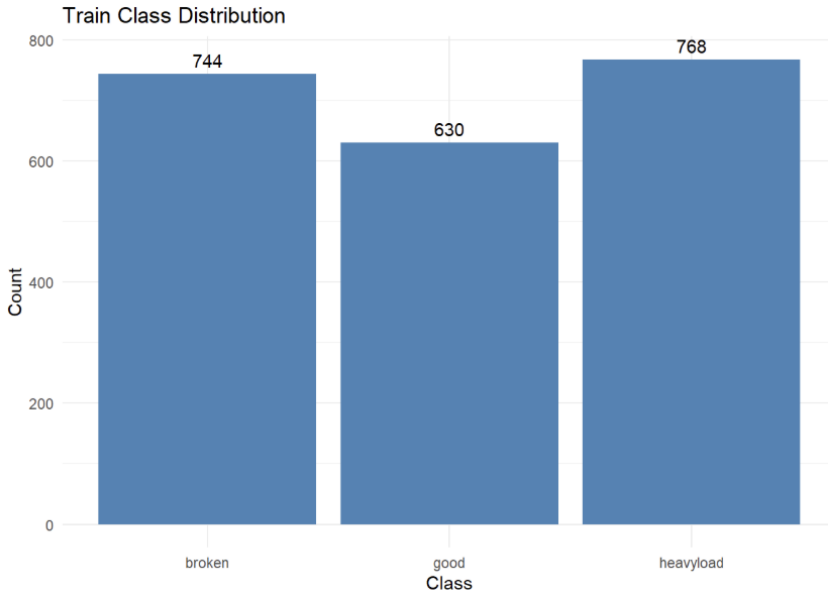


*Figure 11: Class distribution of the training data after adding noise*

**Audio Preprocessing and Feature Extraction**

The IDMT-ISA-ELECTRIC-ENGINE dataset provided raw .wav audio files that are organized into labelled subfolders that correspond to the engine state. As explained earlier, this audio format provides 132,000 features per audio file. Therefore, we experimented with multiple feature extraction techniques starting with the basic amplitude spectrum and moving into audio feature extraction neural network models. The reason we use neural networks for feature extraction is that we don't have the time and computational resources to train them, and so we opt for using the popular pretrained models VGGish, OpenL3, and Wav2Vec 2.0.

*Amplitude Spectrum*

We developed a custom R function, wav_to_amplitude_spectra(), to automate the extraction of the frequency-domain features. The function reads each audio file using the tuneR package and performs an FFT on the time-domain waveform to extract the amplitude spectrum. FFT converts time-domain audio signals to the frequency domain, where each sample becomes a long vector representing signal amplitude at different frequencies. The amplitude spectrum is the magnitude of the FFT components, and it provides a compact summary of the energy present at each frequency. This is suitable for capturing the differences in engine noise patterns between the different operation states. All spectra were normalized to ensure that the recordings were comparable, and an optional sample size was applied for standardizing the frequency resolution. We implemented parallel processing to accelerate the feature extraction processes due to the large number of files from the dataset.

*VGGish features*

The VGGish model architecture is adapted from the VGG neural network originally developed for image classification tasks [8][9]. It consists of deep 2D convolutional layers designed to extract high-level features and is modified to suit audio analysis by adjusting the input shape and replacing the final fully connected layer. We use an open-source PyTorch implementation of VGGish with pretrained weights on the YouTube-8M dataset to extract audio embedding. Since the model expects input in the form of log-Mel spectrograms, we preprocess all raw audio accordingly. VGGish outputs 128-dimensional feature vectors for each non-overlapping 0.96-second segment of audio. Given our 3-second audio clips, this results in a total of 384 features per audio sample.

*OpenL3 features*

OpenL3 is an open-source implementation of the L3-Net architecture, which is trained using audio-visual correspondence (AVC) through self-supervised learning. The model was trained on approximately 60 million samples of music and environmental sounds from the AudioSet dataset [10], making it well-suited to extract semantically meaningful features for our classification task. OpenL3 requires input in the form of log-Mel spectrograms, and we applied

the necessary preprocessing to match this format. The model outputs 512-dimensional feature vectors for each 1-second segment of audio. Consequently, for our 3-second audio clips, we obtain a total of 1536 features per sample.

*Wav2Vec 2.0*

Wav2Vec 2.0 is a self-supervised model for learning speech representations directly from raw audio waveforms. Developed by Facebook AI, the model is pre-trained on large-scale unlabeled audio and fine-tuned for downstream tasks such as speech recognition [11]. Unlike VGGish and OpenL3, Wav2Vec 2.0 does not require handcrafted features such as log-Mel spectrograms; instead, it operates directly on the raw waveform, allowing it to capture more nuanced temporal and frequency information. We use a pre-trained version of Wav2Vec 2.0 available in the torchaudio library. The model outputs contextualized embeddings at a fixed rate of approximately 50 Hz, producing 768-dimensional feature vectors for each frame. Given our 3-second audio clips, this results in approximately 150 vectors per sample, which are subsequently averaged to obtain a single 768-dimensional feature representation per audio file.

**Dimension Reduction using PCA**

After we have extracted meaningful features from the audio data, the dimensionality of the features is still high, the least of which is 384. Therefore, we utilize PCA to reduce the dimensionality of the data, preventing the classifiers from overfitting and avoiding needless long training and cross validation time. For almost all cases, we notice that the knee of the scree plot and the Kaiser rule (Kaiser-Guttman criterion) yield a number of principal components with low explained variance. Therefore, we decided to select the number of principal components based on 95% of explained variance for each feature extraction method.

**Model Training and Cross Validation**

The transformed dataset is then used to train and evaluate several classification models. We selected the below models to capture a range of classification techniques.

- Multinomial Logistic Regression, Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA) were used for better interpretation when data distributions follow linear or quadratic decision boundaries.
- K Nearest Neighbors (KNN) was used as a simple, distance-based classifier that adapts to local patterns in the feature space.
- Decision Tree provided a clear rule-based model that was pruned to avoid overfitting the data.
- Bagging and Random Forest were used to improve stability and accuracy by averaging over multiple decision trees.

- Support Vector Machines (SVM) were used to evaluate both linear and nonlinear (radial) decision boundaries in the transformed feature space.
- Extreme Gradient Boosting (XGBoost) was selected for its high performance and ability to model complex relationships using boosting of decision trees.

Cross-validation using 5-folds was used to choose the values of the model hyperparameters when possible. For example, the number of splitting variables “mtry” for Random Forest models, cost for SVMs, and the number of nearest neighbors for KNN. The caret library was used for cross-validation training routines.

**Performance Evaluation**

In the end, we evaluate the performance of each classifier using standard metrics: Accuracy, Recall, Precision, and F1-score, computed on the test dataset. The Precision, Recall, and F1-scores are aggregated by macro-averaging the class-specific metrics, providing a comprehensive view of model performance. We also include confusion matrices for each classifier to visualize the distribution of correct and incorrect predictions across all classes. This helps identify specific misclassification patterns and offers insight into which classes are most often confused.

# Results

## Amplitude Spectrum

Multinomial Logistic Regression:

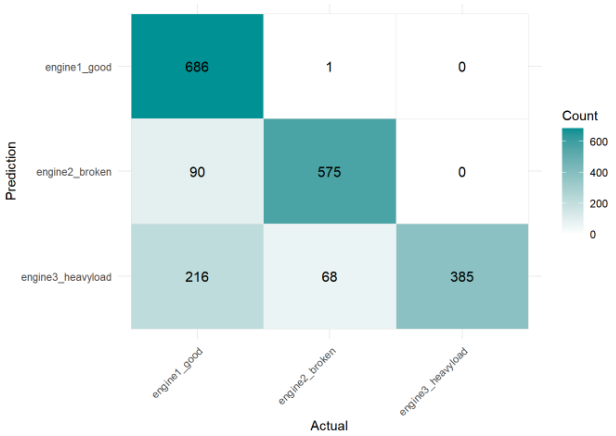


*Figure 12: amplitude spectrum - logistic regression confusion matrix*

LDA:

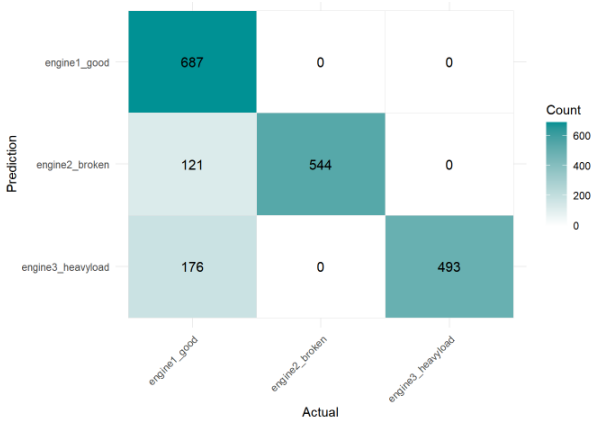


*Figure 13: amplitude spectrum - LDA confusion matrix*

QDA:

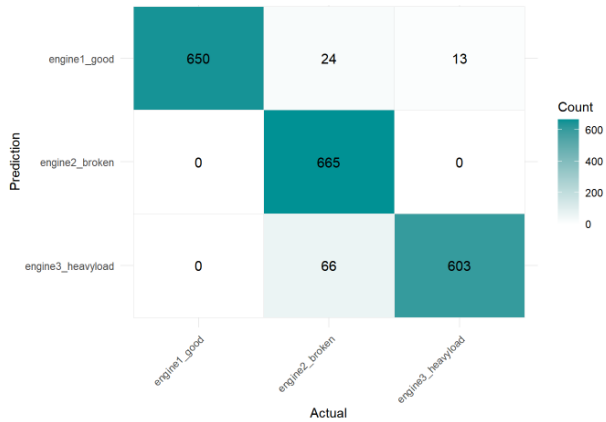


*Figure 14: amplitude spectrum - QDA confusion matrix*

Bagging:

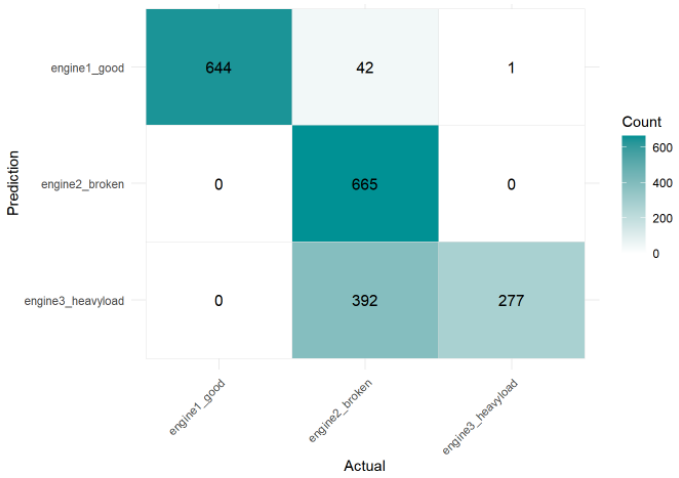


*Figure 15: amplitude spectrum - bagging confusion matrix*

Random Forest:

5-fold cross validation was used to find the "mtry" parameter based on the highest accuracy. According to the plot below, the selected value is 2.

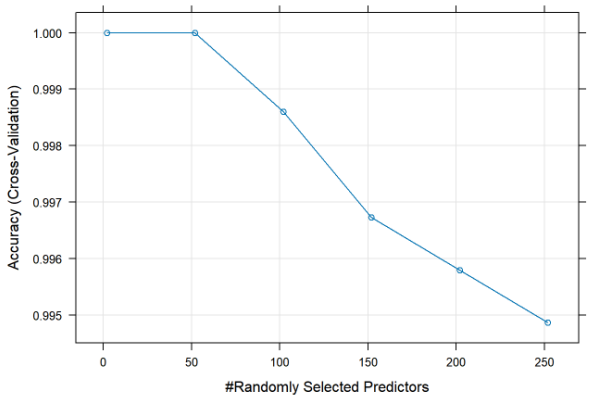


*Figure 16: amplitude spectrum – 5 fold CV plot of random forest*

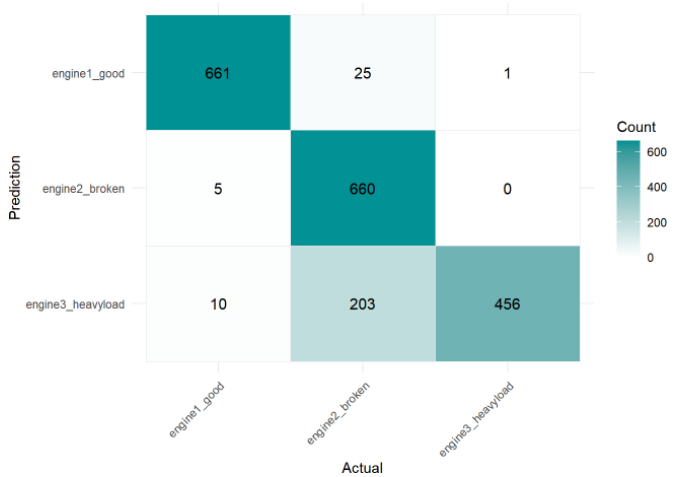


*Figure 17: amplitude spectrum - random forest confusion matrix*

SVM (radial):

5-fold cross validation was used to find the “sigma” and “cost” parameters based on the highest accuracy. According to the plot below, the selected values are sigma = 0.01 and cost = 1.

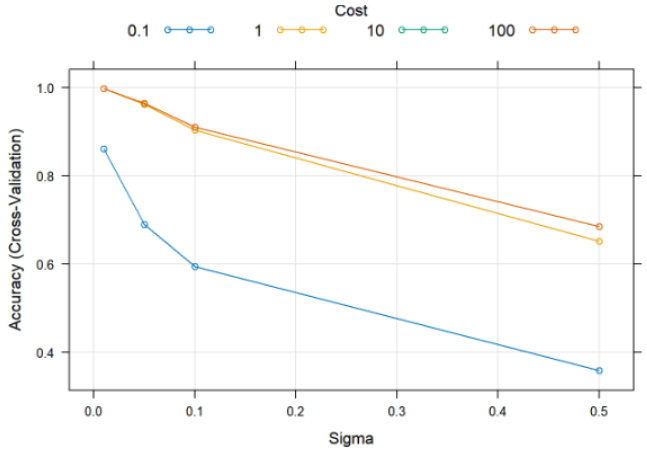


*Figure 18: amplitude spectrum – 5 fold CV plot of radial SVM*

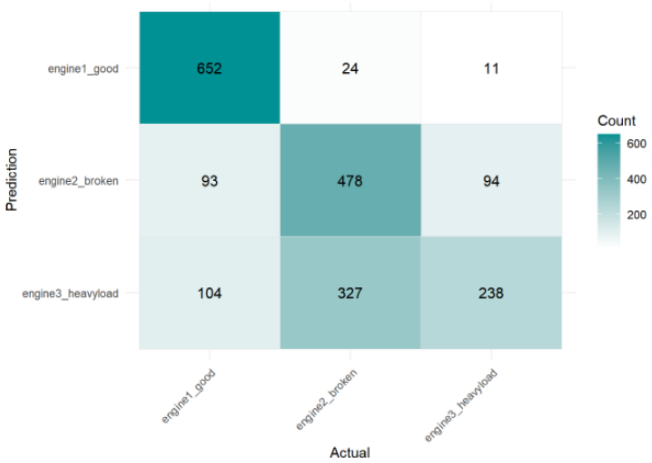


*Figure 19: amplitude spectrum - radial SVM confusion matrix*

SVM (linear):

5-fold cross validation was used to find the “cost” parameter based on the highest accuracy. According to the plot below, the selected value is 0.1.

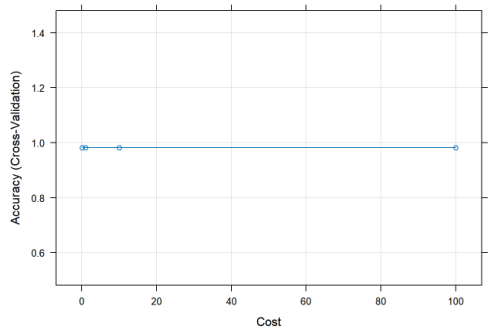


*Figure 20: amplitude spectrum – 5 fold CV plot of linear SVM*

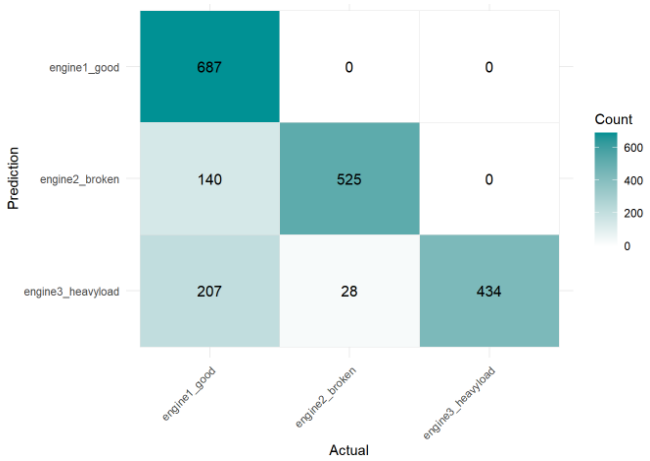


*Figure 21: amplitude spectrum - linear SVM confusion matrix*

XGBoost:

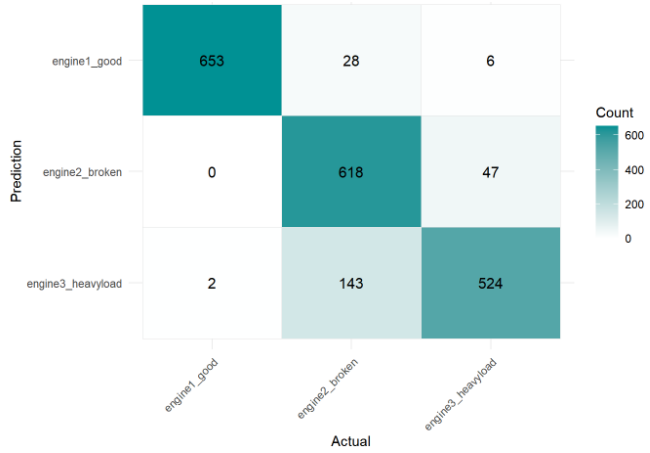


*Figure 22: amplitude spectrum - XG boost confusion matrix*

Decision Tree:

The decision tree is pruned based on the complexity parameter. The final pruned model is the same as the unpruned tree according to the cp plot below:

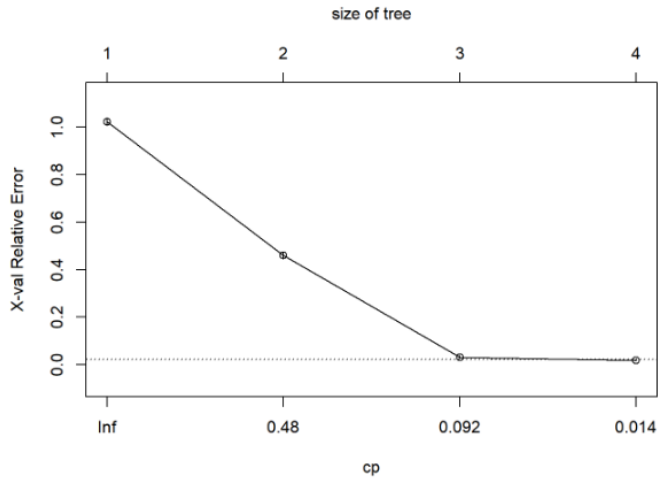


*Figure 23: amplitude spectrum - cp plot of the decision tree*

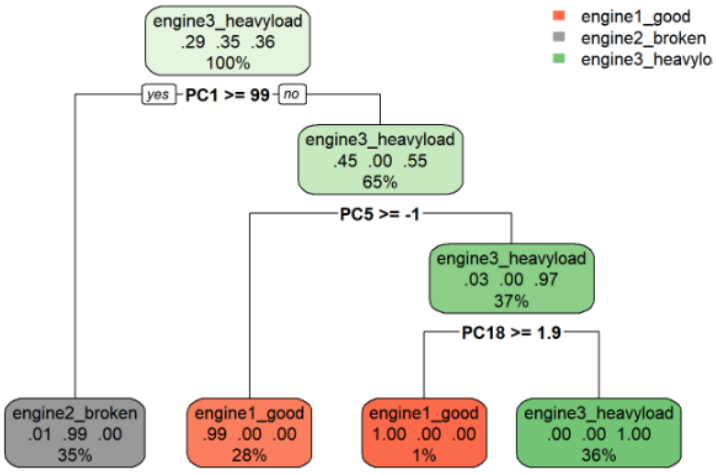


*Figure 24: amplitude spectrum - pruned decision tree*

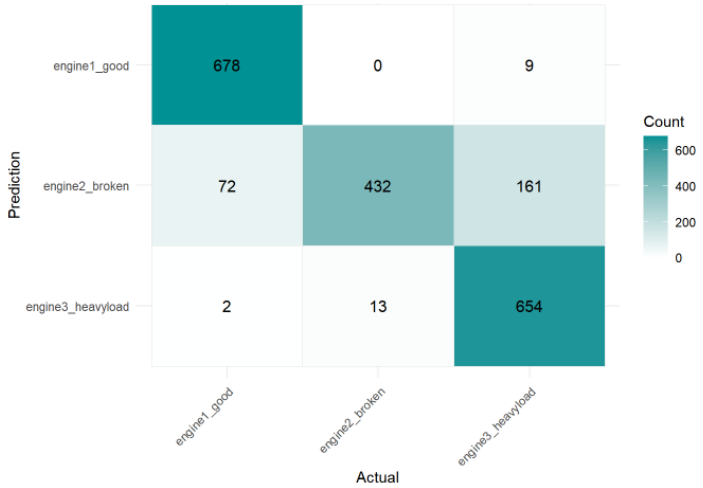


*Figure 25: amplitude spectrum - decision tree confusion matrix*

KNN:

5-fold cross validation was used to find the "k" parameter based on the highest accuracy. According to the plot below, the selected value is 5.

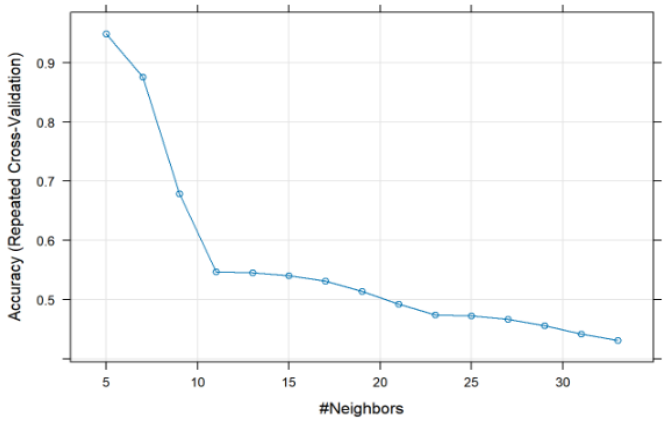


*Figure 26: amplitude spectrum - 5 fold CV plot of KNN*


*Figure 27: amplitude spectrum - KNN confusion matrix*

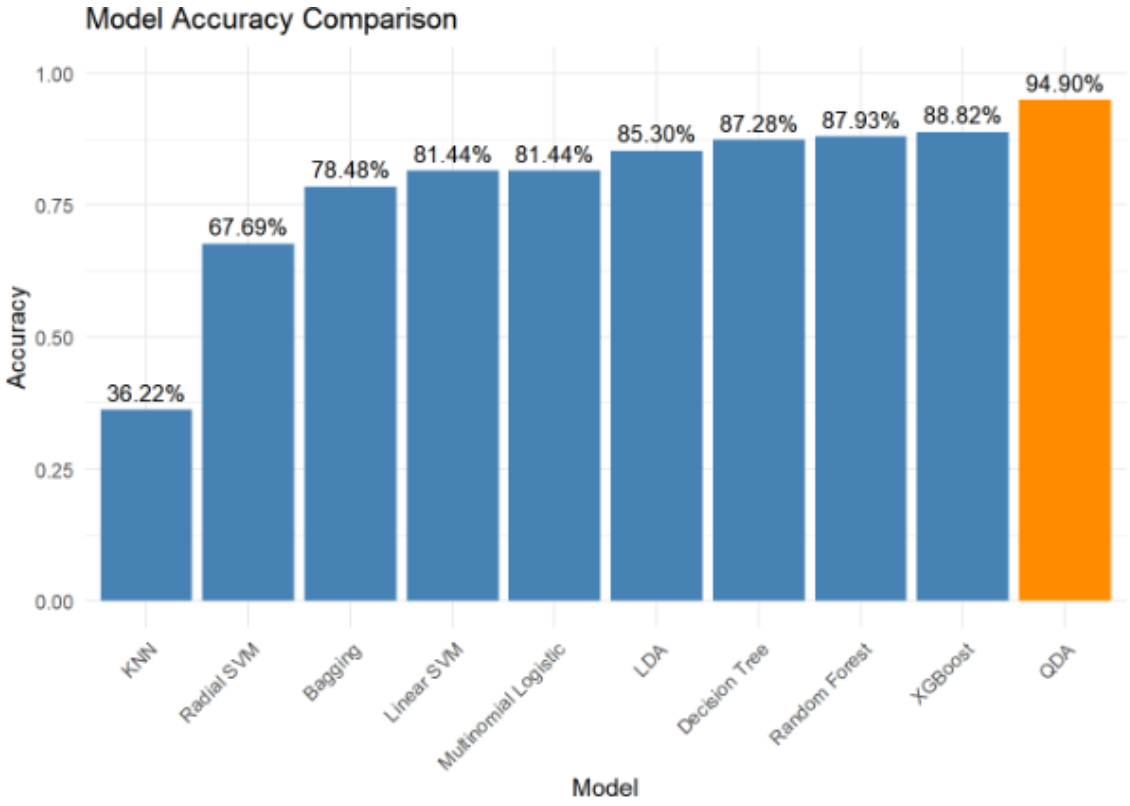


*Figure 28: amplitude spectrum - model accuracy comparisons*

*Table 2: Classification performance of various models using Amplitude Spectrum feature extraction.*

| Model | Accuracy (%) | Precision | Recall | F1 Score |
|---|---|---|---|---|
| LDA | 85.30 | 0.8993902 | 0.8516553 | 0.8569055 |
| QDA | 94.90 | 0.9532303 | 0.9491626 | 0.9491557 |
| Multinomial Logistic | 81.44 | 0.8614631 | 0.8128973 | 0.8087455 |
| Bagging | 78.48 | 0.8671661 | 0.7838199 | 0.7688890 |
| Random Forest | 87.93 | 0.9062886 | 0.8787499 | 0.8766113 |
| Radial SVM | 67.69 | 0.6794794 | 0.6745352 | 0.6530690 |
| Linear SVM | 81.44 | 0.8712590 | 0.8127344 | 0.8157956 |
| Decision Tree | 87.28 | 0.8886905 | 0.8713674 | 0.8655959 |
| XGBoost | 88.82 | 0.8961207 | 0.8876971 | 0.8881115 |
| KNN | 36.22 | 0.6738360 | 0.3557549 | 0.3210049 |

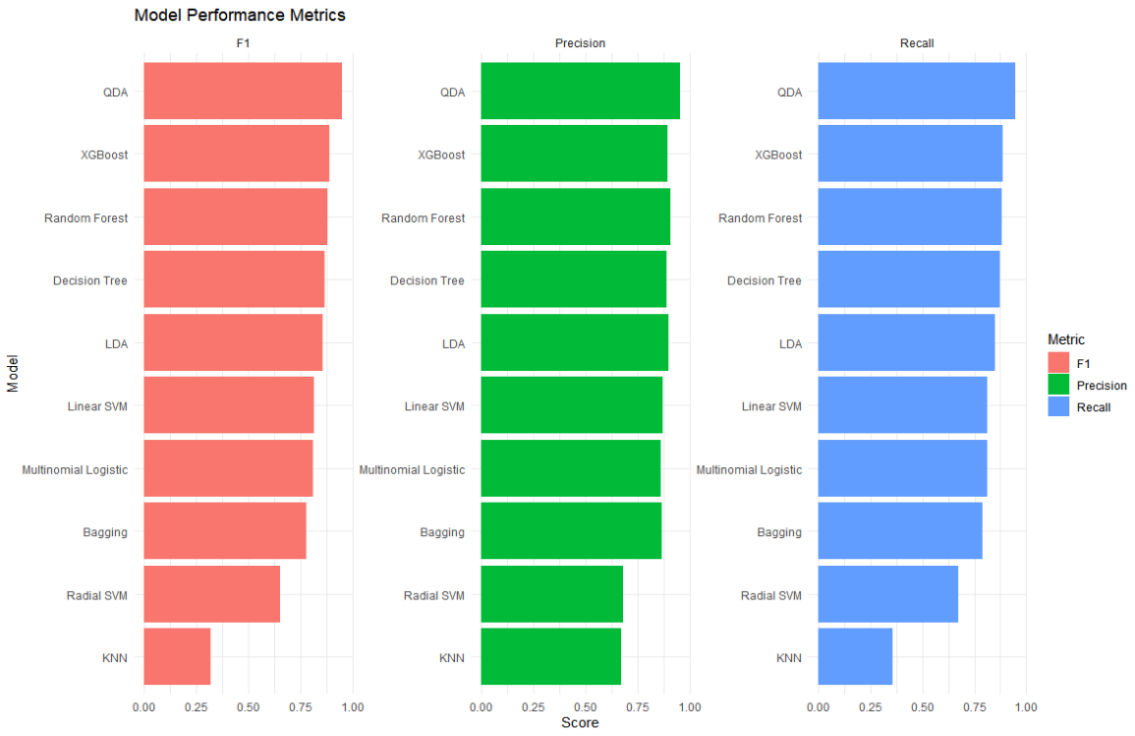


*Figure 29: amplitude spectrum - f1, recall, and precision comparisons*

**VGGish**

Multinomial Logistic Regression

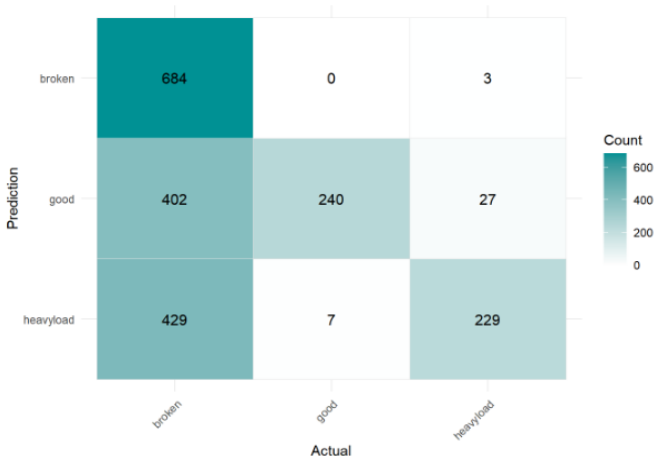


*Figure 30: VGGish - logistic regression confusion matrix*

LDA:

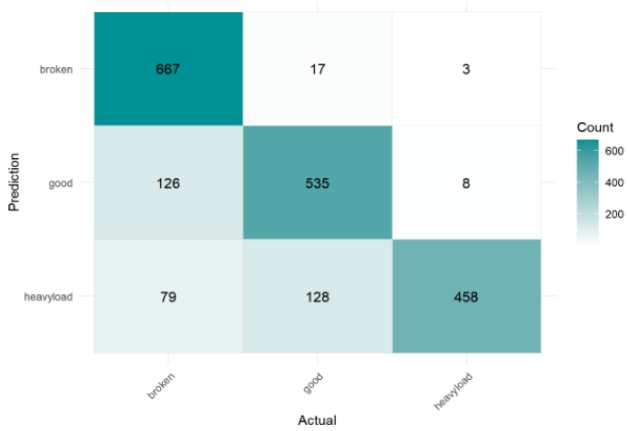


*Figure 31: VGGish - LDA confusion matrix*

QDA:

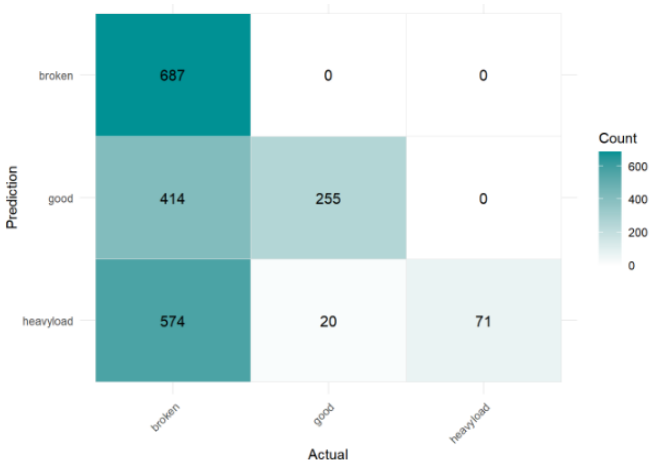


*Figure 32: VGGish - QDA confusion matrix*

Bagging:

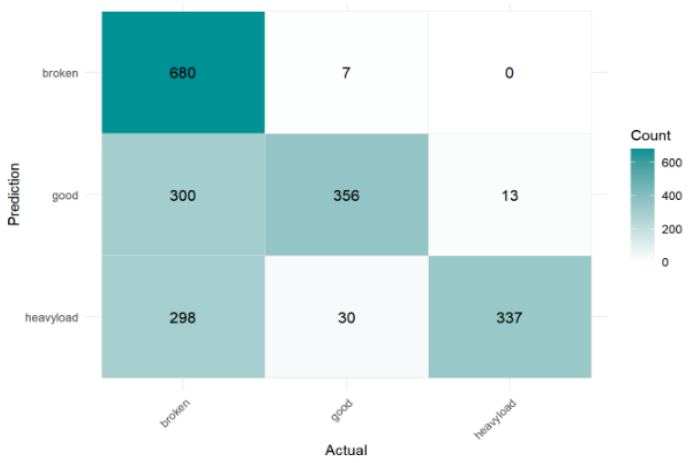


*Figure 33: VGGish - bagging confusion matrix*

Random Forest:

5-fold cross validation was used to find the "mtry" parameter based on the highest accuracy. According to the plot below, the selected value is 12.

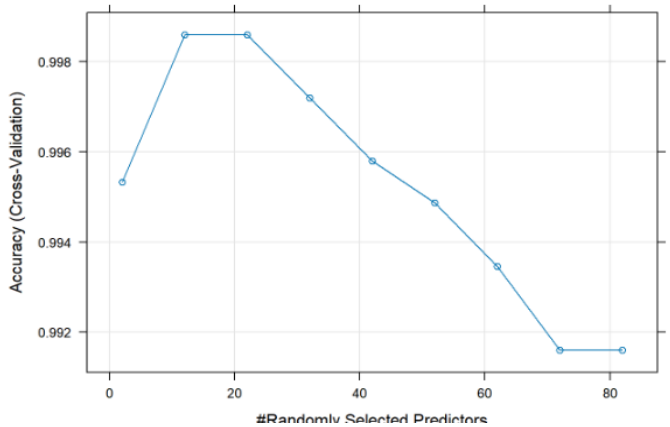


*Figure 34: VGGish – 5 fold CV plot of random forest*

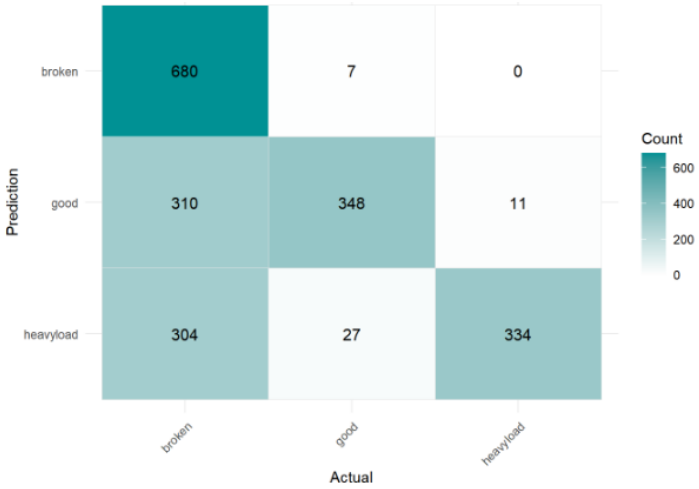


*Figure 35: VGGish - random forest confusion matrix*

SVM (radial):

5-fold cross validation was used to find the “sigma” and “cost” parameters based on the highest accuracy. According to the plot below, the selected values are sigma = 0.01 and cost = 10.

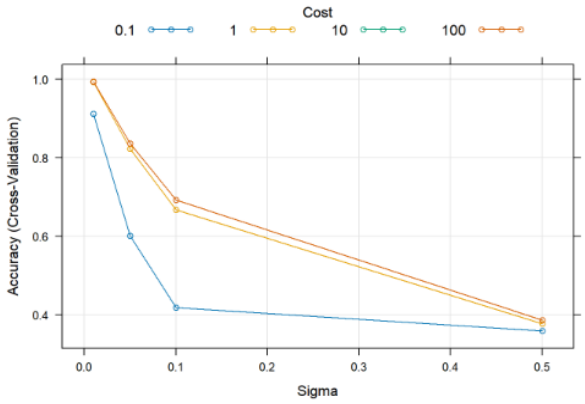


*Figure 36: VGGish – 5 fold CV plot of radial SVM*


*Figure 37: VGGish - radial SVM confusion matrix*

SVM (linear):

5-fold cross validation was used to find the “cost” parameter based on the highest accuracy. According to the plot below, the selected value is 0.1.

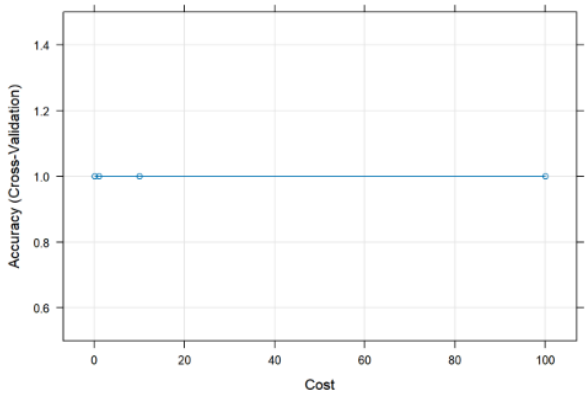


*Figure 38: VGGish – 5 fold CV plot of linear SVM*

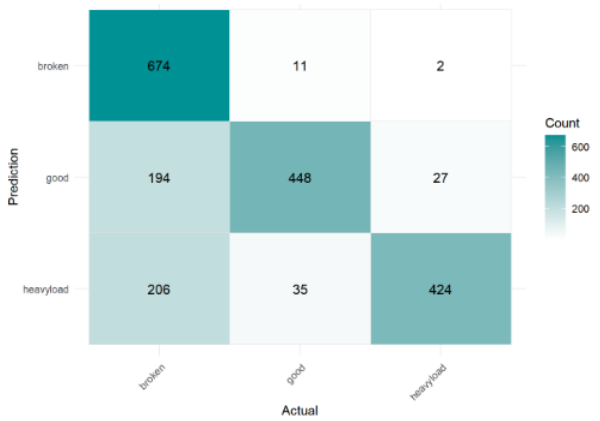


*Figure 39: VGGish - linear SVM confusion matrix*

XGBoost:

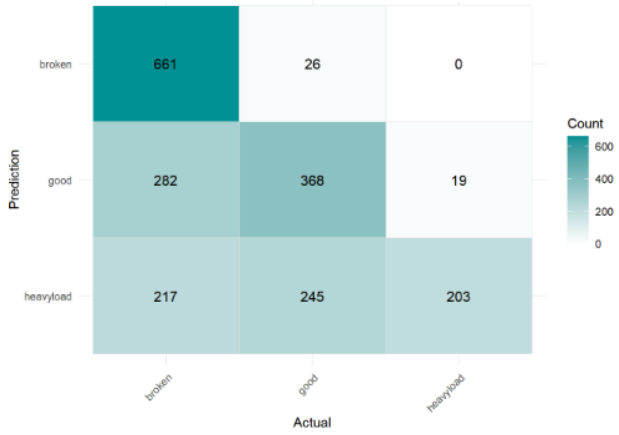


*Figure 40: VGGish - XG boost confusion matrix*

Decision Tree:

The decision tree is pruned based on the complexity parameter. The final pruned model is the same as the unpruned tree according to the cp plot below:

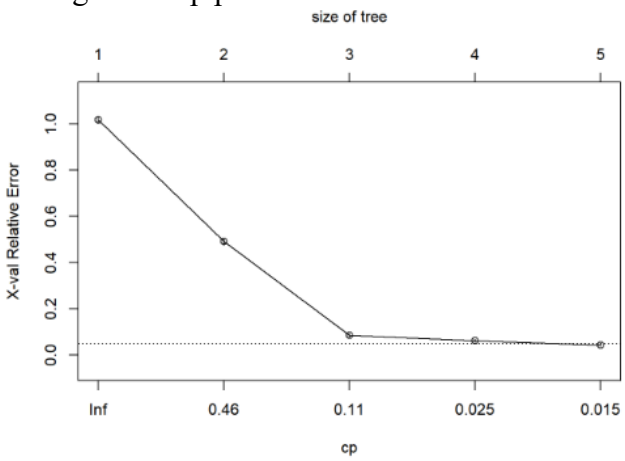


*Figure 41: VGGish - cp plot of the decision tree*

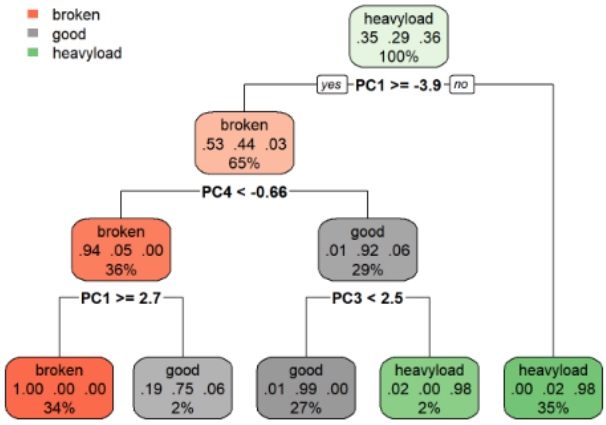


*Figure 42: VGGish - pruned decision tree*

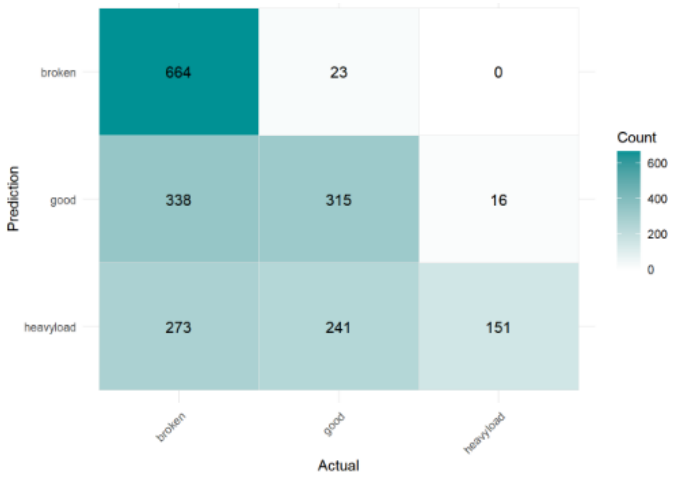


*Figure 43: VGGish - descison tree confusion matrix*

KNN:

5-fold cross validation was used to find the “k” parameter based on the highest accuracy. According to the plot below, the selected value is 5.

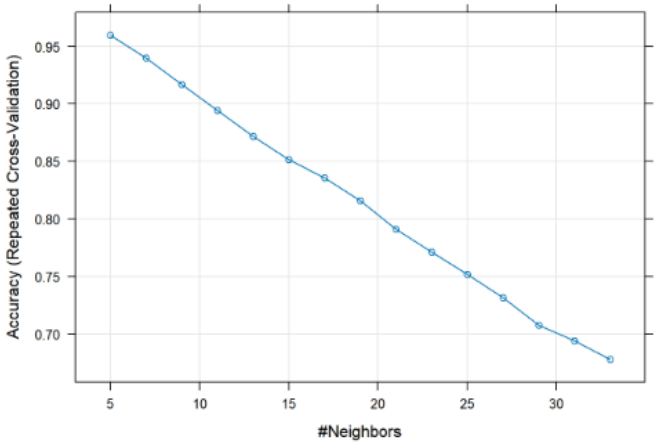


*Figure 44: VGGish – 5 fold CV plot of KNN*

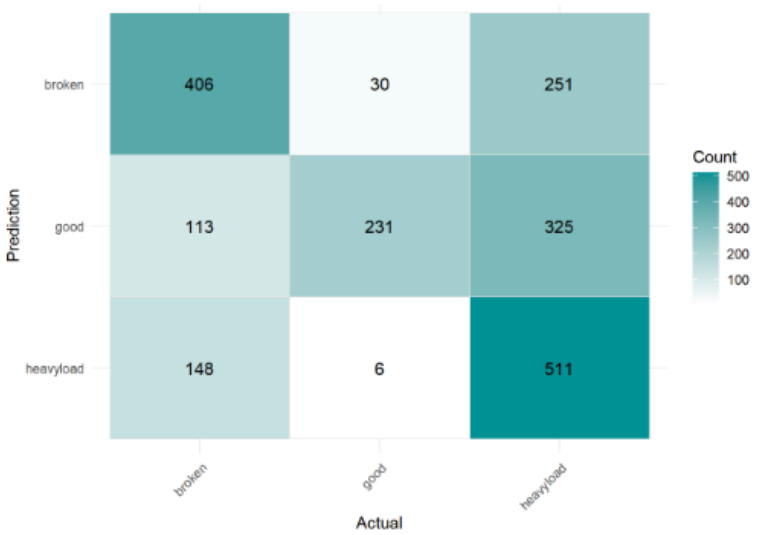


*Figure 45: VGGish - KNN confusion matrix*

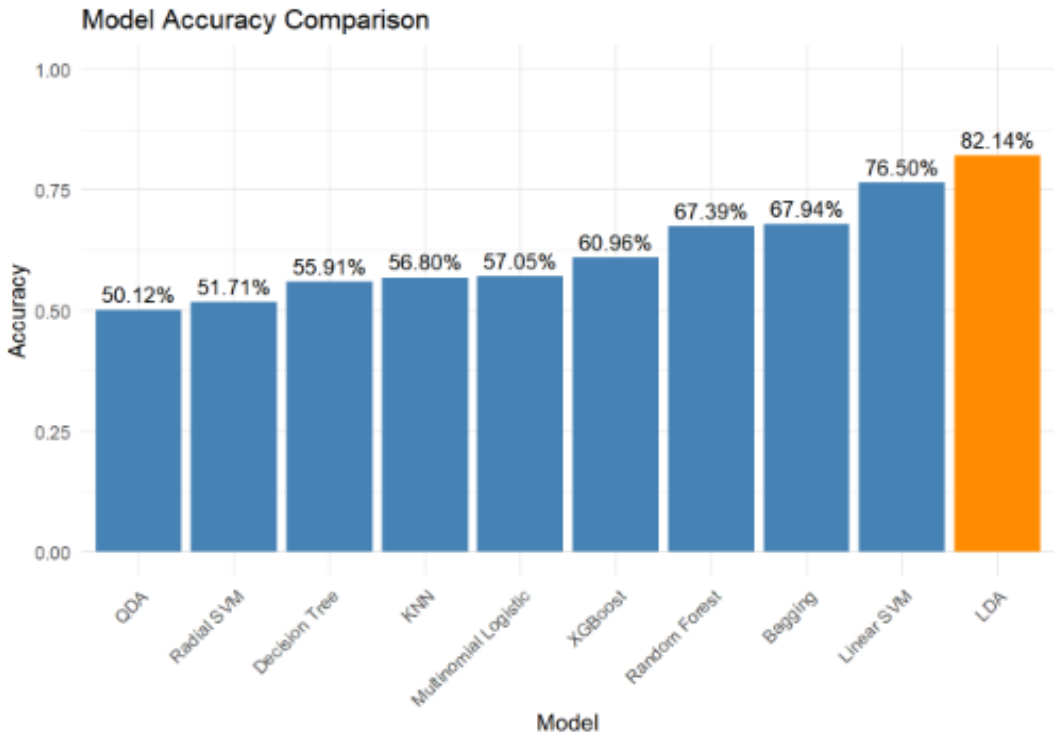


*Figure 46: VGGish - model accuracy comparisons*

*Table 3: Classification performance of various models using VGGish feature extraction.*

| Model | Accuracy (%) | Precision | Recall | F1 Score |
|---|---|---|---|---|
| LDA | 82.14 | 0.8427396 | 0.8197703 | 0.8188723 |
| QDA | 50.12 | 0.7791407 | 0.4959776 | 0.4382998 |
| Multinomial Logistic | 57.05 | 0.7691050 | 0.5662462 | 0.5469806 |
| Bagging | 67.94 | 0.8002636 | 0.6762384 | 0.6755282 |
| Random Forest | 67.39 | 0.8015377 | 0.6707486 | 0.6700448 |
| Radial SVM | 51.71 | 0.7556418 | 0.5121847 | 0.4747122 |
| Linear SVM | 76.50 | 0.8234752 | 0.7627758 | 0.7647976 |
| Decision Tree | 55.91 | 0.6563391 | 0.5548136 | 0.5148829 |
| XGBoost | 60.96 | 0.6867139 | 0.6058307 | 0.5787230 |
| KNN | 56.80 | 0.6479885 | 0.5682293 | 0.5588759 |

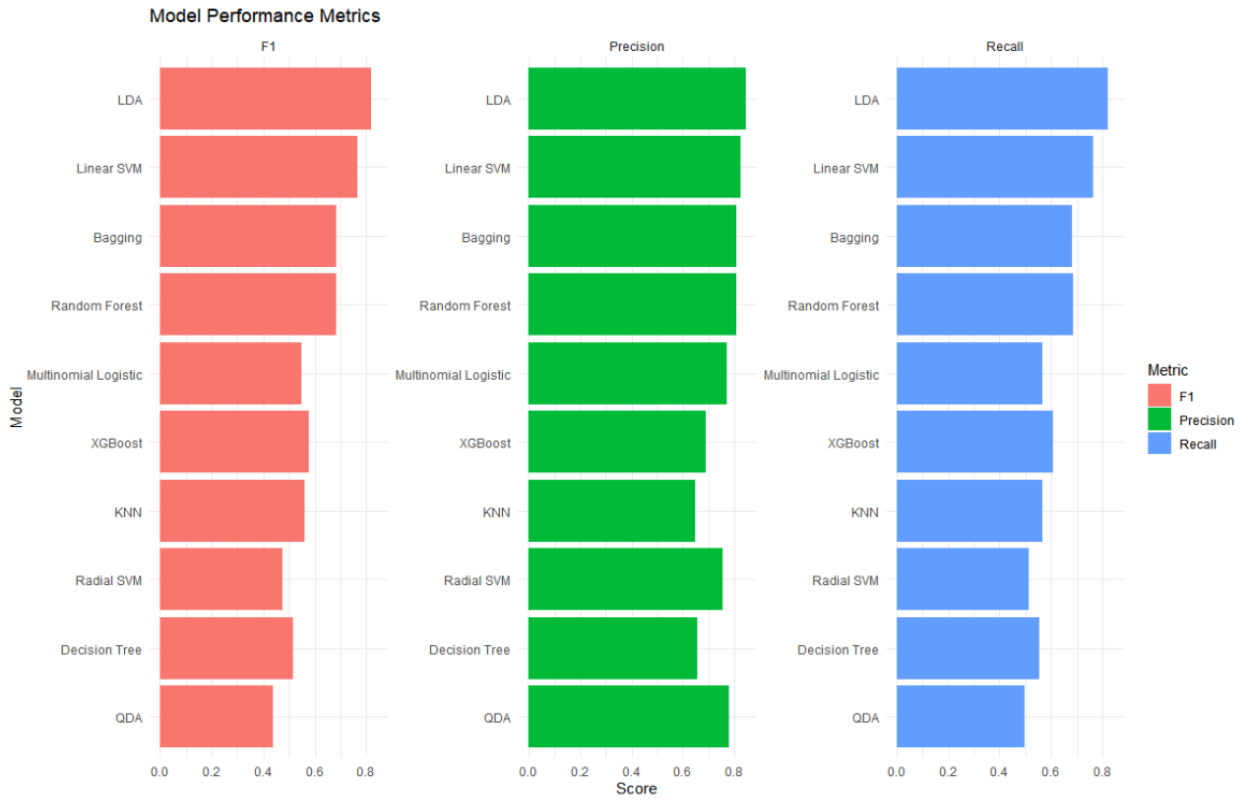


*Figure 47: VGGish - f1, recall, and precision comparisons*

**OpenL3**

Multinomial Logistic regression:

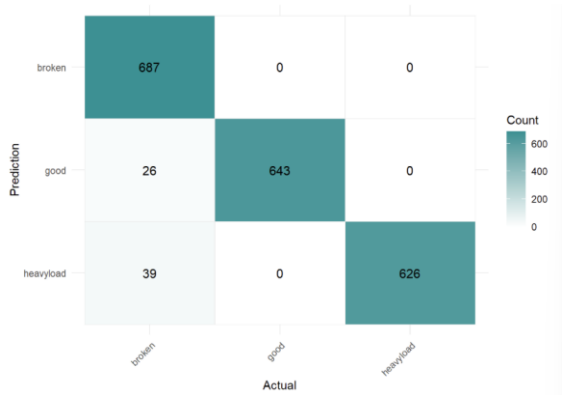


*Figure 48: OpenL3 - logistic regression confusion matrix*

LDA:

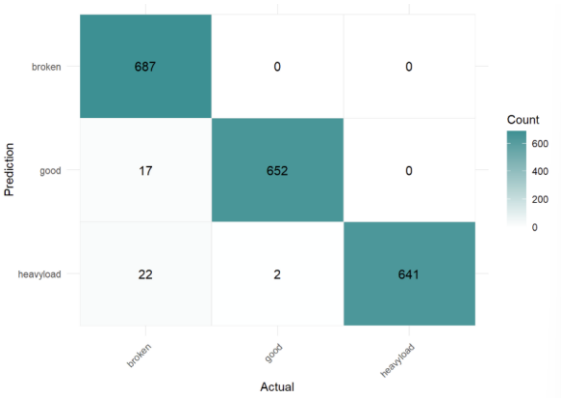


*Figure 49: OpenL3 - LDA confusion matrix*

QDA:


*Figure 50: OpenL3 - QDA confusion matrix*

Bagging:

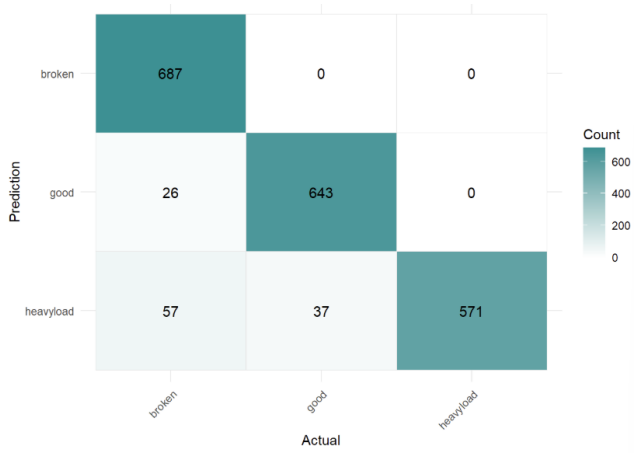


*Figure 51: OpenL3 - bagging confusion matrix*

Random Forest:

5-fold cross validation was used to find the “mtry” parameter based on the highest accuracy. According to the plot below, the selected value is 62.

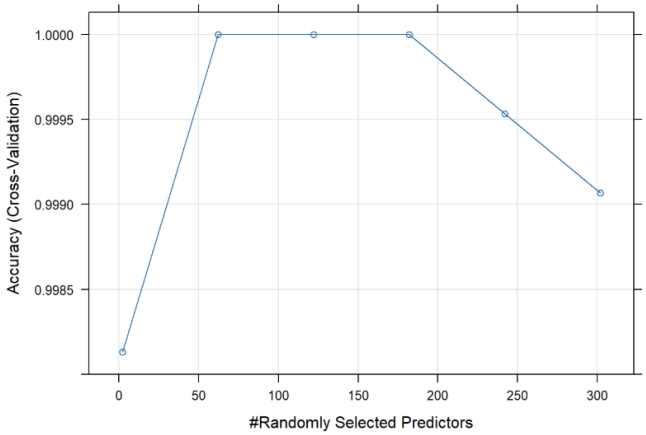


*Figure 52: OpenL3 - 5 fold CV plot of random forest*

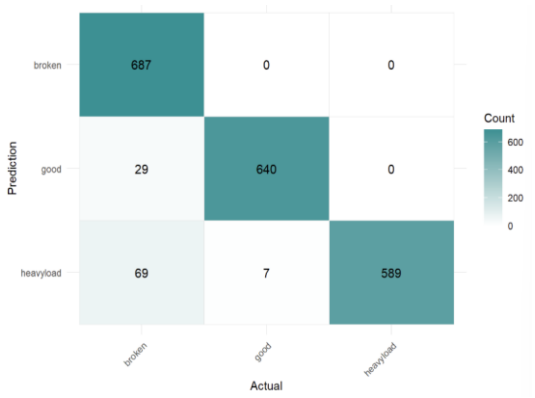


*Figure 53: OpenL3 - random forest confusion matrix*

SVM (radial):

5-fold cross validation was used to find the “sigma” and “cost” parameters based on the highest accuracy. According to the plot below, the selected values are sigma = 0.01 and cost = 10.

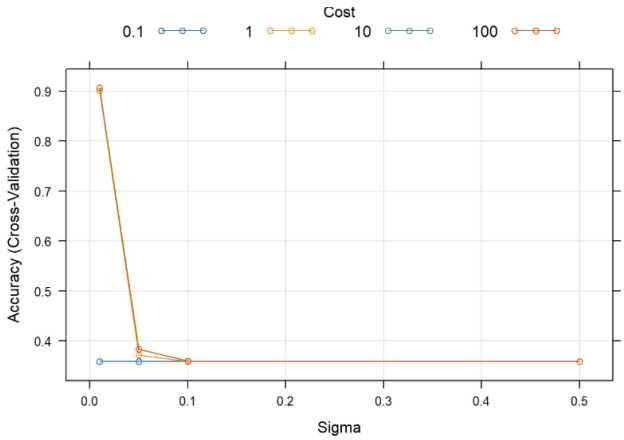


*Figure 54: OpenL3 - 5 fold CV plot of radial SVM*

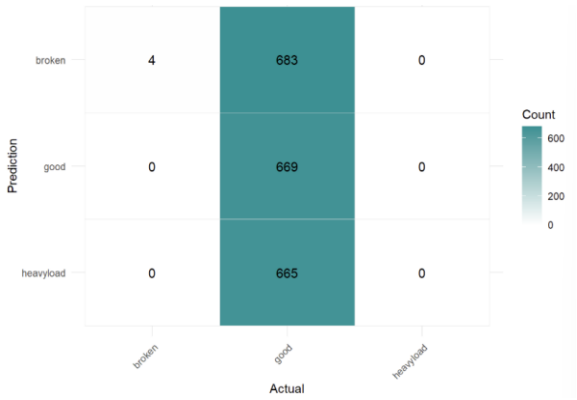


*Figure 55: OpenL3 - radial SVM confusion matrix*

SVM (linear):

5-fold cross validation was used to find the “cost” parameter based on the highest accuracy. According to the plot below, the selected value is 0.1.

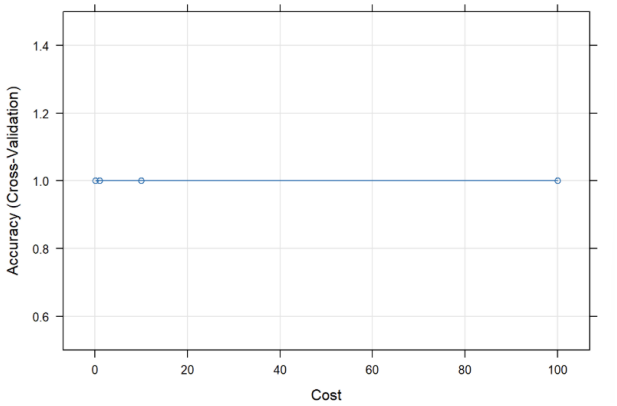


*Figure 56: OpenL3 - 5 fold CV plot of linear SVM*

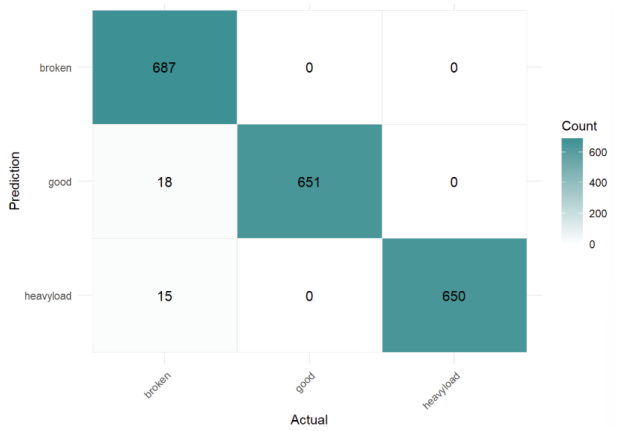


*Figure 57: OpenL3 - linear SVM confusion matrix*

XGBoost:


*Figure 58: OpenL3 - XG boost confusion matrix*

Decision Tree:

The decision tree is pruned based on the complexity parameter. The final pruned model is the same as the unpruned tree according to the cp plot below:

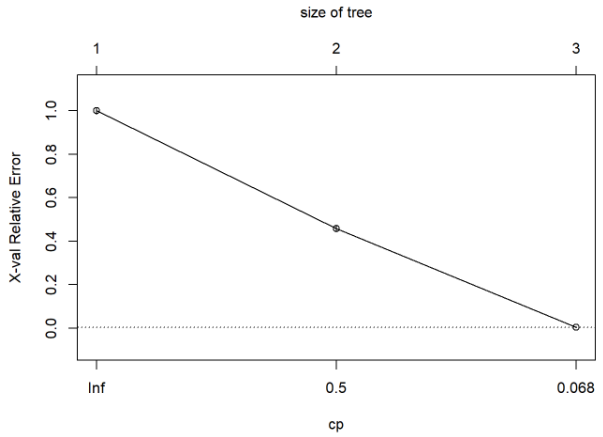


*Figure 59: OpenL3 - cp plot of the decision tree*

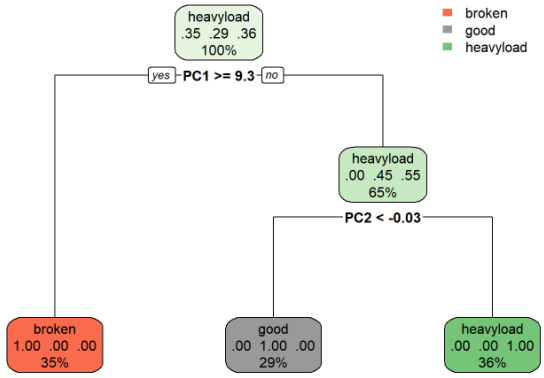


*Figure 60: OpenL3 - pruned decision tree*

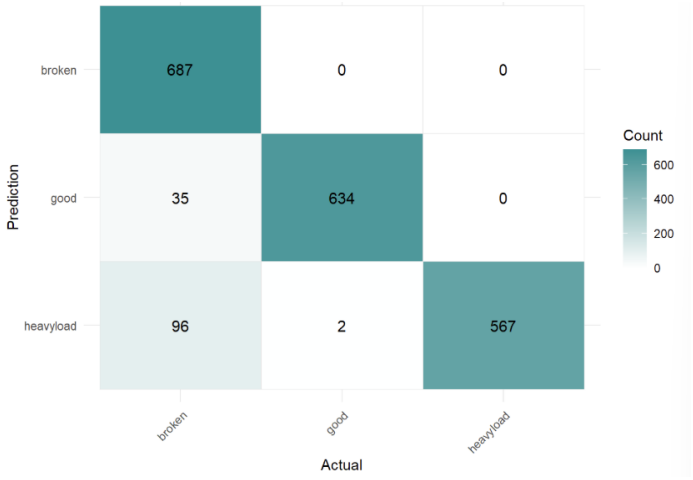


*Figure 61: OpenL3 - decision tree confusion matrix*

KNN:

5-fold cross validation was used to find the “k” parameter based on the highest accuracy. According to the plot below, the selected value is 5.

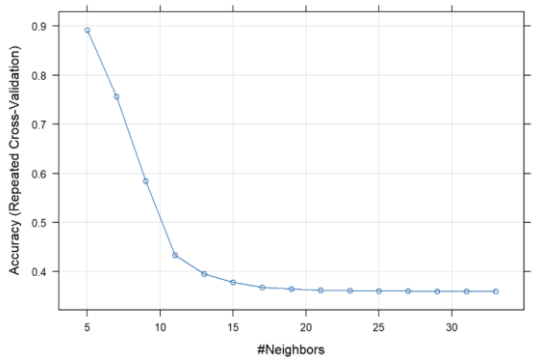


*Figure 62: OpenL3 - 5 fold CV plot of KNN*


*Figure 63: OpenL3 - KNN confusion matrix*

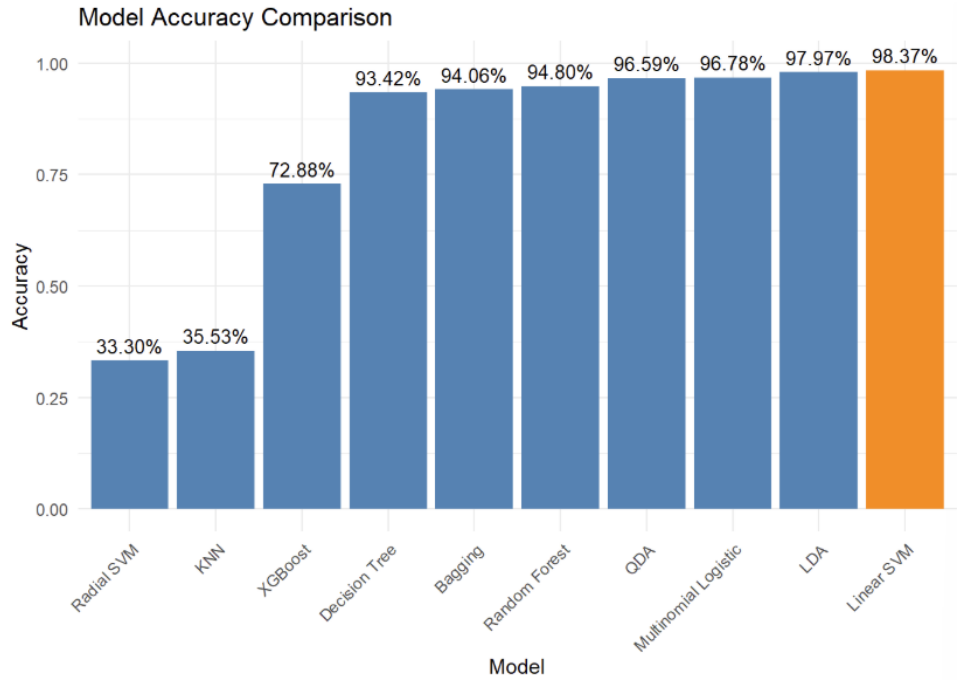


*Figure 64: OpenL3 - model accuracy comparisons*

*Table 4: Classification performance of various models using OpenL3 feature extraction.*

| Model | Accuracy (%) | Precision | Recall | F1 Score |
|---|---|---|---|---|
| LDA | 97.97 | 0.9810743 | 0.9794996 | 0.9798870 |
| QDA | 96.59 | 0.9694627 | 0.9654825 | 0.9662602 |
| Multinomial Logistic | 96.78 | 0.9711879 | 0.9674965 | 0.9682678 |
| Bagging | 94.06 | 0.9459320 | 0.9399275 | 0.9400935 |
| Random Forest | 94.80 | 0.9547800 | 0.9474553 | 0.9484874 |
| Radial SVM | 33.30 | 0.6658404 | 0.3352741 | 0.2548580 |
| Linear SVM | 98.37 | 0.9847222 | 0.9835126 | 0.9838342 |
| Decision Tree | 93.42 | 0.9455695 | 0.9334382 | 0.9350196 |
| XGBoost | 72.88 | 0.8337893 | 0.7323669 | 0.7173822 |
| KNN | 35.53 | 0.6819738 | 0.3488602 | 0.2011988 |

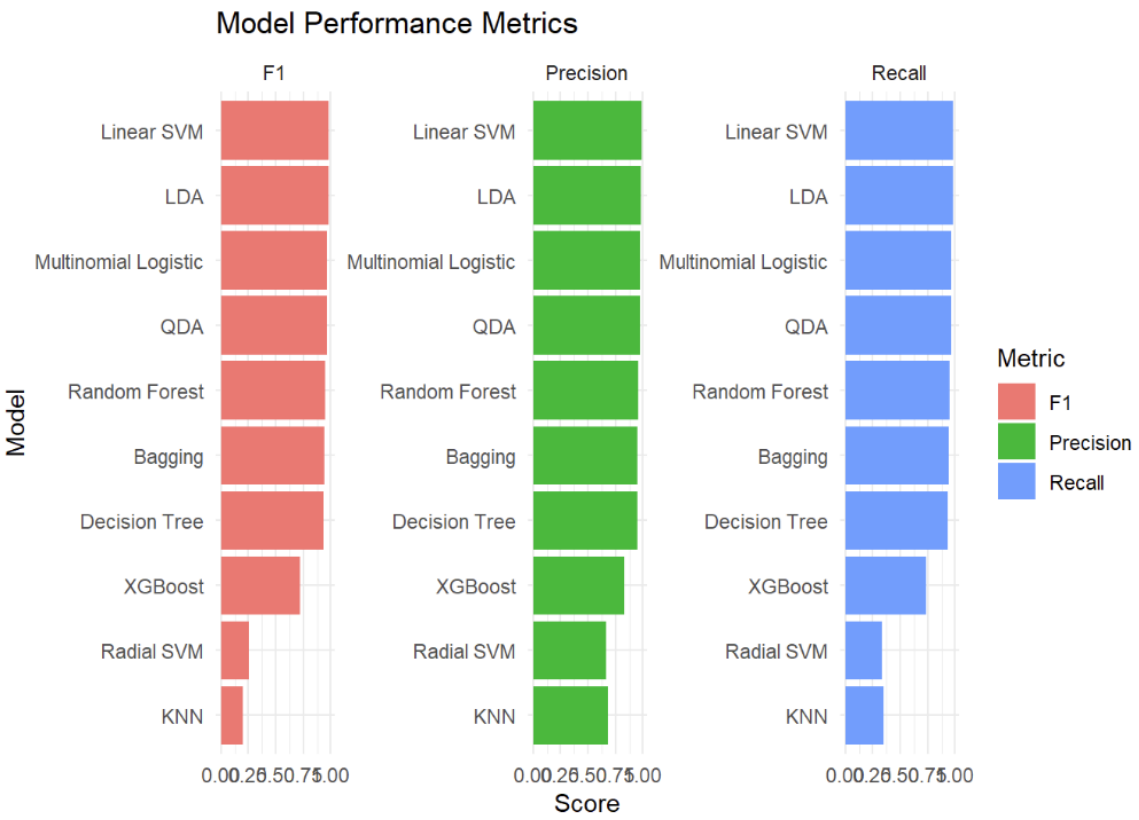


*Figure 65: OpenL3 - f1, recall, and precision comparisons*

**Wav2Vec**

Multinomial Logistic regression:

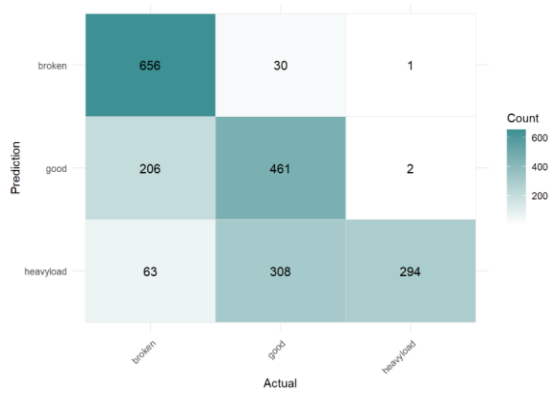


*Figure 66: Wav2Vec - logistic regression confusion matrix*

LDA:

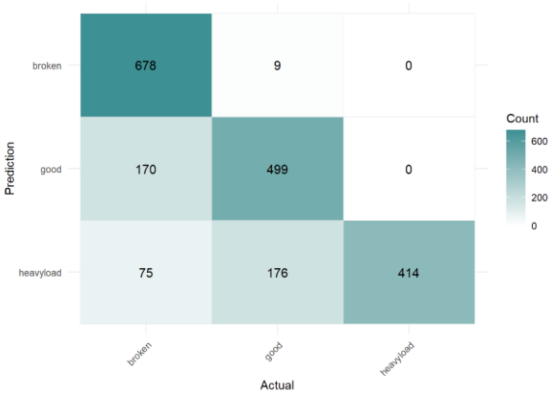


*Figure 67: Wav2Vec - LDA confusion matrix*

QDA:

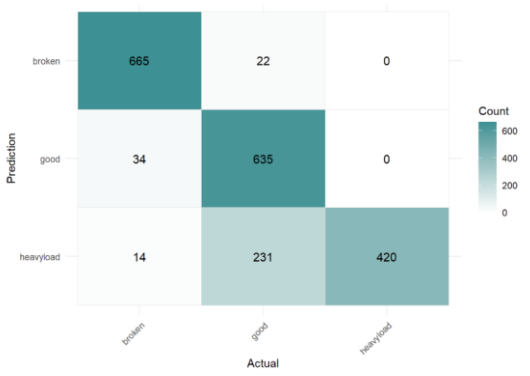


*Figure 68: Wav2Vec - QDA confusion matrix*

Bagging:

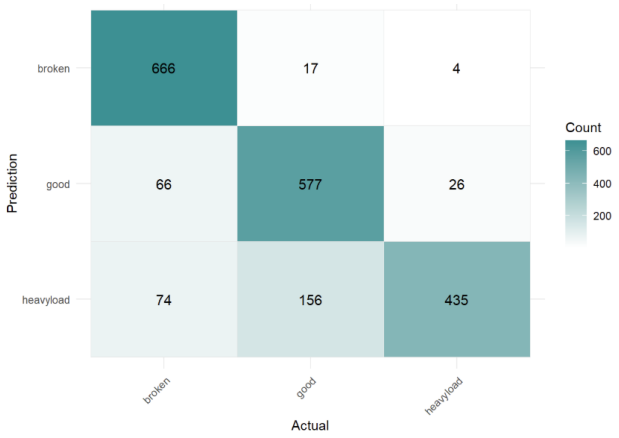


*Figure 69: Wav2Vec - bagging confusion matrix*

Random Forest:

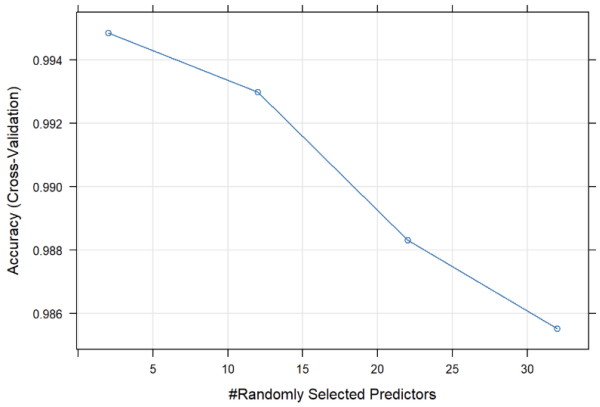


*Figure 70: Wav2Vec - 5 fold CV plot of random forest*

Accuracy was used to select the optimal model using the largest value. The final value used for the model was mtry = 2.

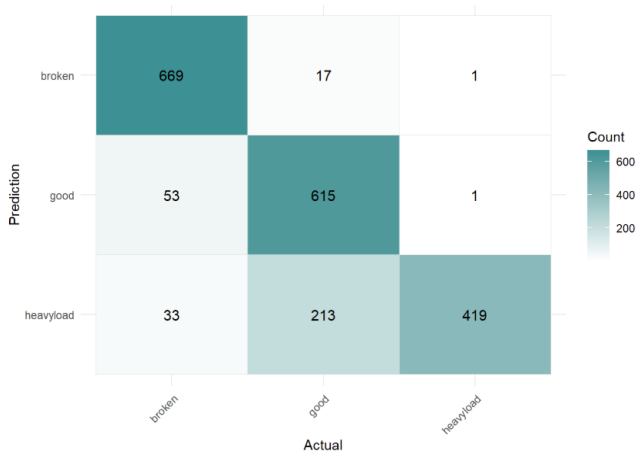


*Figure 71: Wav2Vec - random forest confusion matrix*

SVM (radial):

5-fold cross validation was used to find the "sigma" and "cost" parameters based on the highest accuracy. According to the plot below, the selected values are sigma=0.01 cost=1.

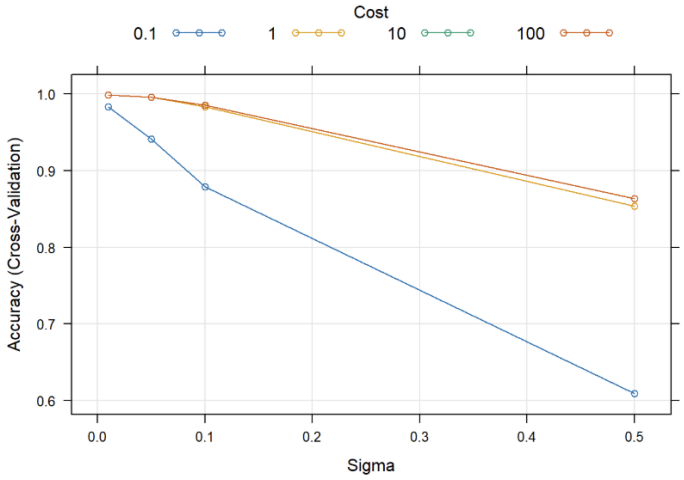


*Figure 72: Wav2Vec - 5 fold CV plot of radial SVM*

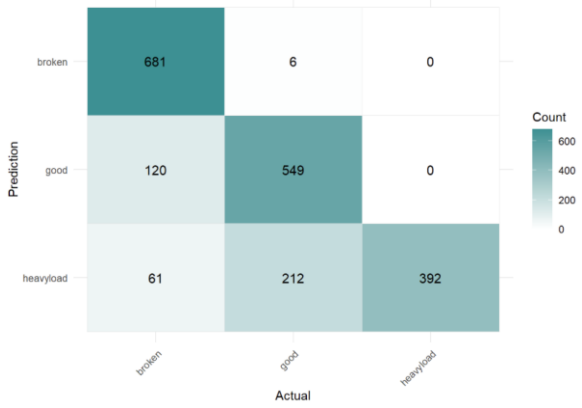


*Figure 73: Wav2Vec - radial SVM confusion matrix*

SVM (linear):

5-fold cross validation was used to find the “cost” parameter based on the highest accuracy. According to the plot below, the selected value is 0.1.

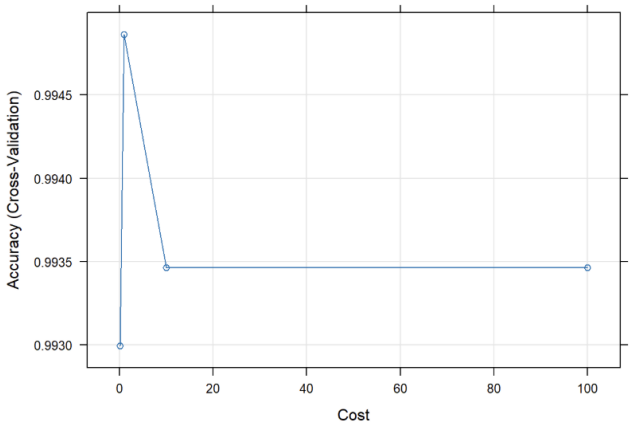


*Figure 74: Wav2Vec - 5 fold CV plot of linear SVM*

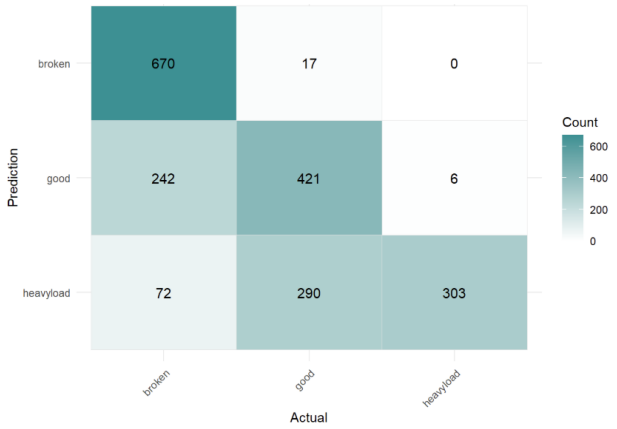


*Figure 75: Wav2Vec - linear SVM confusion matrix*

XGBoost:

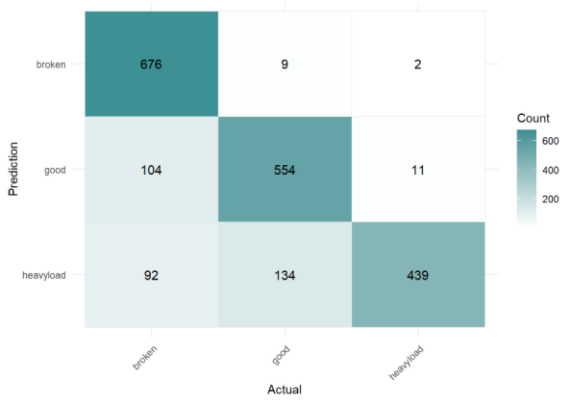


*Figure 76: Wav2Vec - XG boost confusion matrix*

Decision Tree:

The decision tree is pruned based on the complexity parameter. The final pruned model is the same as the unpruned tree according to the cp plot below:

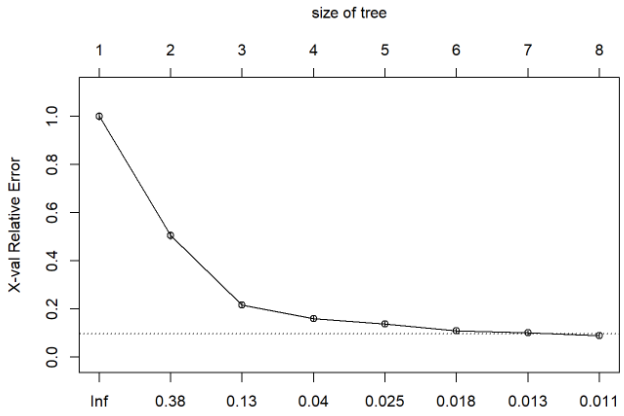


*Figure 77: Wav2Vec - cp plot of the decision tree*

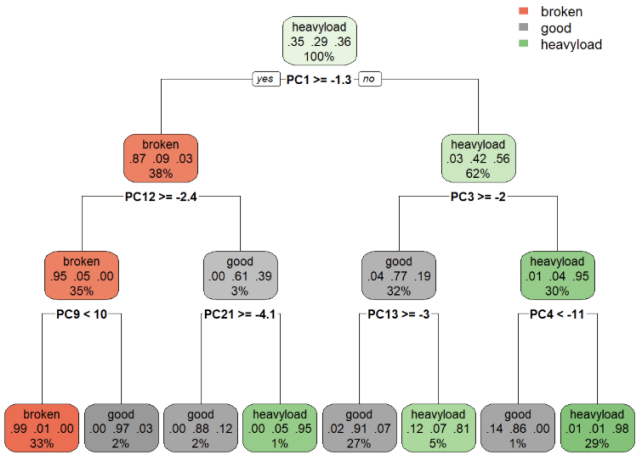


*Figure 78: Wav2Vec - pruned decision tree*

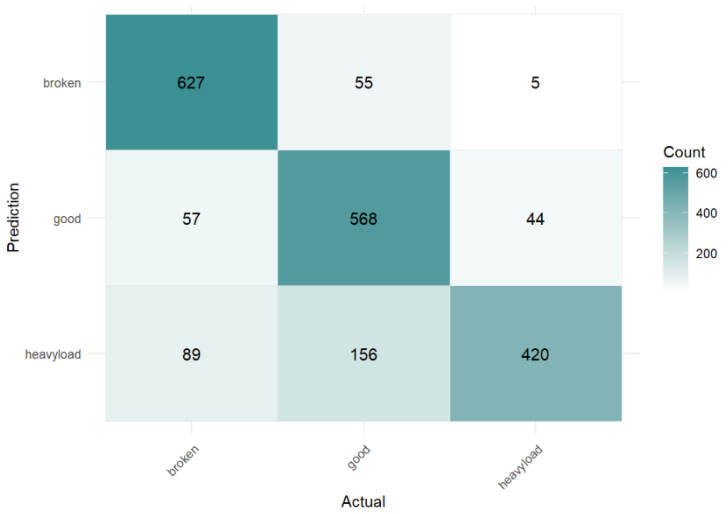


*Figure 79: Wav2Vec - decision tree confusion matrix*

KNN:

5-fold cross validation was used to find the "k" parameter based on the highest accuracy. According to the plot below, the selected value is 5.

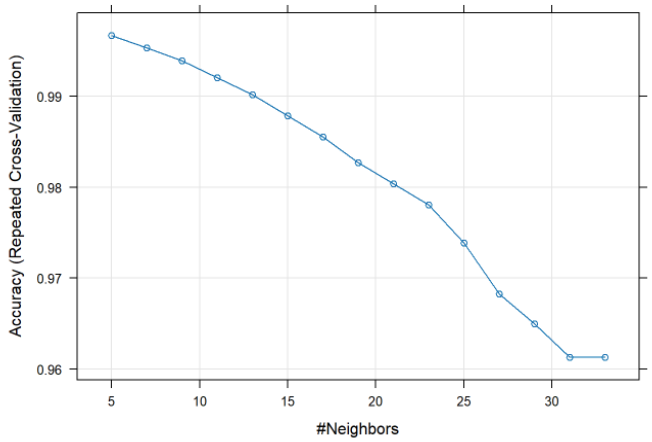


*Figure 80: Wav2Vec - 5 fold CV plot of KNN*

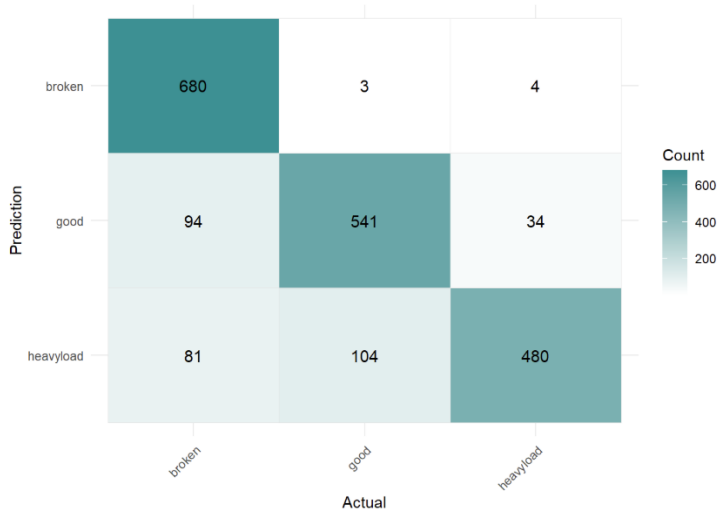


*Figure 81: Wav2Vec - KNN confusion matrix*

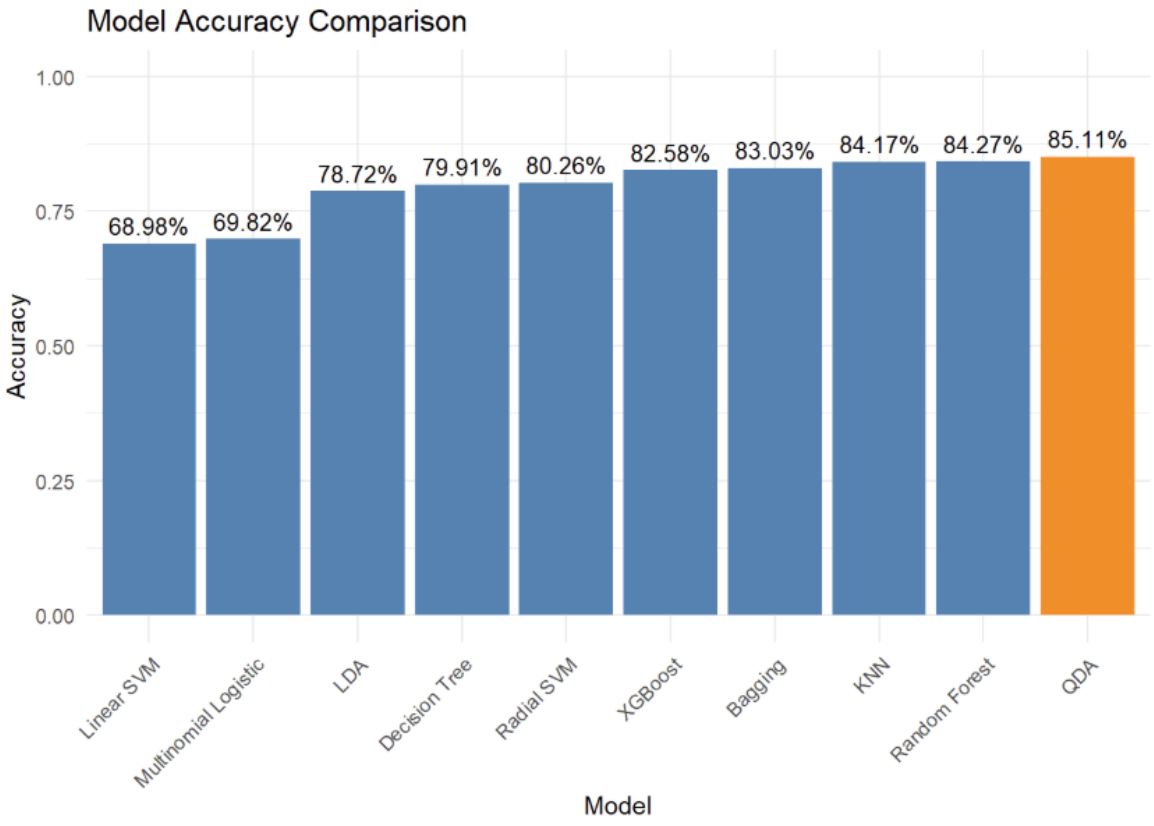


*Figure 82: Wav2Vec - model accuracy comparison*

*Table 5: Classification performance of various models using Wav2Vec feature extraction.*

| Model | Accuracy (%) | Precision | Recall | F1 Score |
|---|---|---|---|---|
| LDA | 78.72 | 0.8213645 | 0.7851151 | 0.7824111 |
| QDA | 85.11 | 0.8825896 | 0.8495778 | 0.8466216 |
| Multinomial Logistic | 69.82 | 0.7586865 | 0.6953566 | 0.6843959 |
| Bagging | 83.03 | 0.8437066 | 0.8286830 | 0.8251079 |
| Random Forest | 84.27 | 0.8697176 | 0.8410523 | 0.8373115 |
| Radial SVM | 80.26 | 0.8352663 | 0.8004560 | 0.7952076 |
| Linear SVM | 68.98 | 0.7465912 | 0.6867304 | 0.6756039 |
| Decision Tree | 79.91 | 0.8119293 | 0.7977570 | 0.7947251 |
| XGBoost | 82.58 | 0.8471011 | 0.8240801 | 0.8214613 |
| KNN | 84.17 | 0.8522797 | 0.8400950 | 0.8383439 |

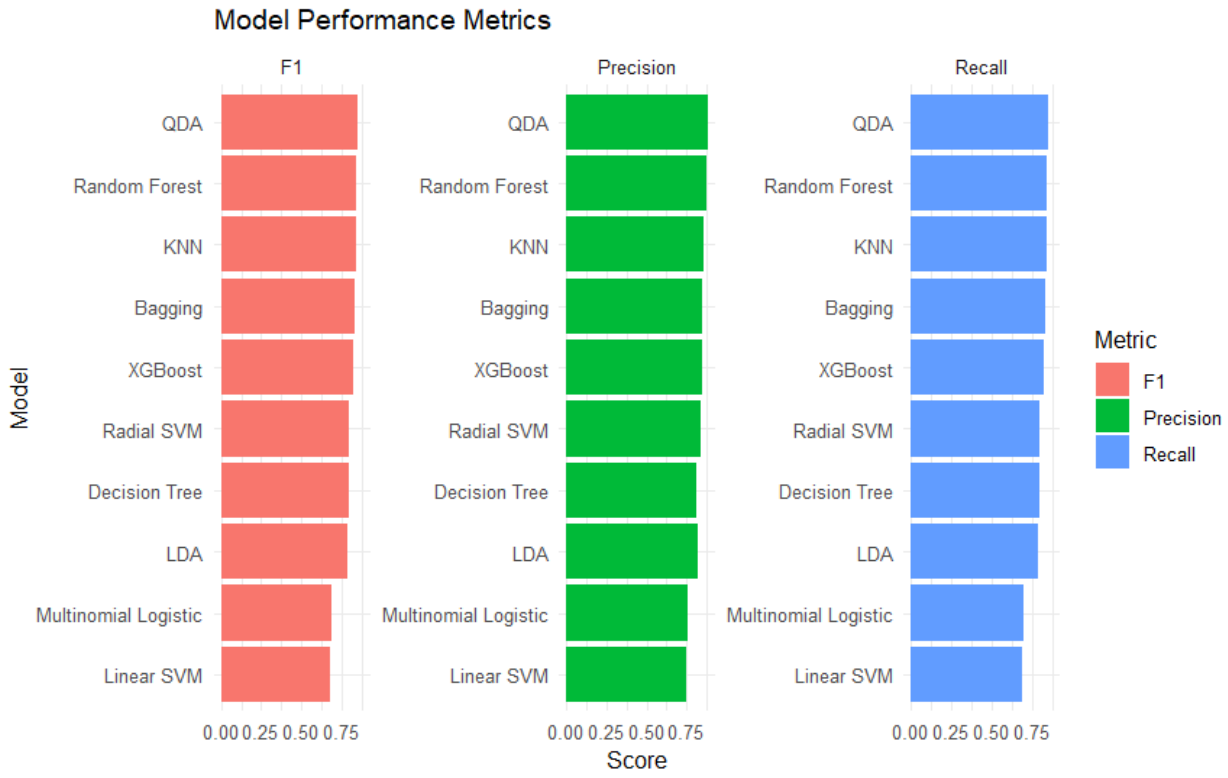


*Figure 83: Wav2Vec - f1, recall, and precision comparisons*

**Overall**

*Table 6: Overall performance metric comparisons between all feature extraction methods.*

| Feature Type | Accuracy (Mean) | Accuracy (Median) | Precision (Mean) | Precision (Median) | Recall (Mean) | Recall (Median) | F1 Score (Mean) | F1 Score (Median) |
|---|---|---|---|---|---|---|---|---|
| **Amplitude Spectrum** | 78.95% | 83.37% | 0.850 | 0.880 | 0.788 | 0.832 | 0.780 | 0.836 |
| **VGGish** | 62.65% | 59.00% | 0.756 | 0.774 | 0.623 | 0.587 | 0.604 | 0.569 |
| **Wav2Vec** | 78.17% | 80.09% | 0.827 | 0.839 | 0.795 | 0.812 | 0.790 | 0.808 |
| **OpenL3** | 81.37% | 94.43% | 0.893 | 0.950 | 0.813 | 0.944 | 0.790 | 0.944 |

## Discussion

This section contains the interpretation of the classification results that were obtained through four different feature extraction techniques (Amplitude Spectrum, VGGish, OpenL3, and Wav2Vec 2.0). We will evaluate how different machine learning methods and feature extraction techniques impact the performance of the models and discuss the accuracy and assess the models based on certain performance metrics such as F1 score, Recall, and Precision.

### Amplitude Spectrum

Despite being the simplest method, the Amplitude Spectrum features produced surprisingly strong results. XGBoost (88.82%), Random Forest (87.93%), and Decision Tree (87.28%) achieved high accuracy and F1-scores above 0.86 which demonstrates that frequency-domain representations can effectively capture the discriminative patterns in engine audio signals. QDA (94.90%) performed the best, indicating that spectral features may naturally form clusters with distinct covariance patterns. In contrast, KNN (36.22%) and Radial SVM (67.69%) were less effective which could be due to the high dimensionality in this representation. Looking at the PCA, t-SNE, and UMAP visualizations, it is also clear that simple discriminators like LDA, QDA, and Logistic Regression performed better than more complex ones. These findings show that the classical signal processing methods can still be competitive and still relevant for machine learning.

### VGGish

VGGish exhibited the most variability in performance across models as compared to the other neural network-based methods. While LDA (82.14%) and Linear SVM (76.50%) performed well, models such as QDA (50.12%), Radial SVM (51.71%), and Decision Tree (55.91%) dropped in performance. The inconsistencies suggest that VGGish features may be less adaptable to variations in data and are more prone to overfitting or noise. Ensemble methods such as Bagging (67.94%) and Random Forest (67.39%) yielded moderately stable results. Nevertheless, the relatively lower F1-scores in most of the classifiers point to weaker class separability within the VGGish feature space compared to OpenL3 or even Wav2Vec. This is also clear by looking at the PCA, t-SNE, and UMAP visualizations, as it is harder to discriminate between the classes than OpenL3 and Amplitude Spectrum features. Nonetheless, fine-tuning the VGGish model on our data might yield better results.

### OpenL3

OpenL3 consistently yielded the highest performance across nearly all classifiers. The Linear SVM model achieved 98.37% accuracy, which was the highest among all the experiments, with precision and F1-scores exceeding 0.98. LDA (97.97%), Multinomial Logistic Regression (96.78%), and QDA (96.59%) also exhibited very strong performances, indicating that OpenL3 produces highly linearly separable features that allow even simpler models to perform well. However, certain non-parametric methods such as Radial SVM (33.30%) and KNN (35.53%) have

low accuracy and F1-scores. This may indicate that the high dimensionality of OpenL3 embeddings can negatively affect algorithms that depend on distance metrics or kernel bandwidths. By looking at the PCA visualization, it is clear that a linear decision boundary can discriminate well on this feature type. Overall, OpenL3 appears to provide the most discriminative and stable representations, particularly when paired with linear models.

**Wave2Vec**

Wav2Vec 2.0 embeddings led to strong overall performance across multiple classifiers. The highest accuracy was achieved by QDA (85.11%), followed closely by Random Forest (84.27%) and Bagging (83.03%), each paired with F1-scores above 0.82. These results suggest that Wav2Vec effectively captures meaningful representations of engine sounds, especially when leveraged by models that can handle nonlinear decision boundaries. The improved performance of Radial SVM (80.26%) over Linear SVM (68.98%) supports this interpretation. Collectively, these outcomes demonstrate that Wav2Vec is a valuable approach for audio feature extraction, particularly when combined with flexible classifiers. Note that although Wav2Vec was trained for speech related tasks, it still performed relatively well for motor sounds. This indicates that Wav2Vec generalizes well, and if paired with fine-tuning on the dataset, it might perform better than OpenL3.

Across all the feature sets, OpenL3 embeddings proved to be the most effective due to their strong classification with linear models. Wav2Vec 2.0 offered balanced performance, particularly for ensemble and nonlinear classifiers. While being the simpler method, Amplitude Spectrum delivered strong results with tree-based and quadratic models which display its ability as a lightweight alternative. VGGish was the least consistent method, highlighting the importance of model-feature compatibility. Notably, Radial SVM and KNN underperformed across multiple settings, especially with OpenL3 and VGGish, due to their sensitivity to high-dimensional and densely clustered data. Whereas methods like Random Forest and XGBoost consistently offered strong performance across feature sets, suggesting that they can adapt well to varying representations. These results suggest that pre-trained neural embeddings are powerful tools for audio classification but must be carefully paired with appropriate modeling strategies. Future work may explore combining multiple embeddings, fine-tuning models for specific tasks, or incorporating sequential models such as temporal CNNs or recurrent networks to further enhance performance.

We further quantified the overall effectiveness of each feature extraction method by calculating the mean and median performance metrics across all classifiers (see Table 6). OpenL3 was the most consistent since it had the highest median values across all four metrics. Interestingly, while Wav2Vec and Amplitude Spectrum had similar mean scores, Amplitude Spectrum exhibited higher medians which suggest that it is more stable performance across the classifiers. VGGish, displayed the lowest averages and the widest variation, indicating its inconsistency and sensitivity

to model selection. These aggregated metrics reinforce the earlier interpretation that OpenL3 provides the most linearly separable and discriminative embeddings for this classification task.

## Conclusion

In this study, we investigated the effectiveness of different audio feature extraction techniques for classifying electric engine conditions using machine learning. We compared four distinct approaches: Amplitude Spectrum (a classical frequency-domain method) and three neural network-based embeddings (VGGish, OpenL3, and Wav2Vec 2.0). Each method was evaluated across ten different models, and performance was assessed using accuracy, precision, recall, and F1-score on a labeled dataset of engine sound recordings. The results demonstrate that OpenL3 consistently outperformed all other feature sets, achieving near-perfect classification with linear models such as LDA and SVM. Wav2Vec 2.0 also showed strong, balanced performance across ensemble and nonlinear classifiers. Amplitude Spectrum delivered competitive results, particularly with tree-based and boosting models despite being a handcrafted feature set. VGGish, on the other hand, produced more variable outcomes, highlighting the importance of feature–model compatibility. Our results show that pre-trained neural models like OpenL3 can be very effective for classifying engine sounds, especially when used with the right type of classifier. We also found that even simple features like amplitude spectrum can work well if combined with strong models like Random Forest or XGBoost. Future work could consist of trying to combine different feature types, using models that can handle time-based patterns (like RNNs or CNNs), or applying this system to real-time monitoring of machines to help predict and prevent failures. Another interesting area of research involves using unsupervised anomaly detection methods to distinguish well-functioning machines from faulty ones. Unlike supervised classification, which requires labeled examples of all possible fault types—a task that is often impractical due to the vast and unpredictable range of potential failures—unsupervised learning adopts a one-class approach. It models the normal behavior of machines and flags any significant deviation as anomalous. This approach is particularly advantageous in real-world settings, where the unknown nature and diversity of faults make it difficult to obtain comprehensive labeled datasets.

## References


[1] K. Strantzalis et al, "Operational State Recognition of a DC Motor Using Edge Artificial Intelligence," *Sensors (Basel)*, vol. 22, (24), pp. 9658, 2022. . DOI: 10.3390/s22249658.

[2] M. Engeler et al, "Condition-based Maintenance: Model vs. Statistics a Performance Comparison," *Procedia CIRP*, vol. 57, pp. 253–258, 2016. DOI: 10.1016/j.procir.2016.11.044.

[3] A. Glowacz, "Acoustic-Based Fault Diagnosis of Commutator Motor," *Electronics*, vol. 7, (11), pp. 299, 2018. DOI: https://doi.org/10.3390/electronics7110299.

[4] F. Alharbi et al, "A Brief Review of Acoustic and Vibration Signal-Based Fault Detection for Belt Conveyor Idlers Using Machine Learning Models," *Sensors (Basel)*, vol. 23, (4), pp. 1902. DOI: 10.3390/s23041902.

[5] F. Akbalık et al, “Engine Fault Detection by Sound Analysis and Machine Learning,” *Appl. Sci.* 2024, 14(15), 6532; https://doi.org/10.3390/app14156532.

[6] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE." Journal of Machine Learning Research, vol. 9, (11), 2008. .

[7] L. McInnes, J. Healy and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," arXiv Preprint arXiv:1802.03426, 2018. .

[8] S. Hershey *et al.*, “CNN Architectures for Large-Scale Audio Classification,” *arXiv.org*, Sep. 29, 2016. https://arxiv.org/abs/1609.09430

[9] K. Simonyan and A. Zisserman, “Very deep convolutional networks for Large-Scale image recognition,” *arXiv.org*, Sep. 04, 2014. https://arxiv.org/abs/1409.1556.

[10] A. L. Cramer, H. -H. Wu, J. Salamon and J. P. Bello, "Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, 2019, pp. 3852-3856, doi: 10.1109/ICASSP.2019.8682475.

[11] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations,” *arXiv.org*, Jun. 20, 2020. https://arxiv.org/abs/2006.11477.