YRKESHÖGSKOLAN
NOVIA

Performance Comparison of One-Class Classifiers in Faulty Sanding Machine Detection Using Sound

MD. Zubairul Haque

Master's Thesis

Thesis for a NOVIA (UAS) - Master

Master of Engineering

Vasa, Wolffskavägen 33

DEGREE THESIS

Author: MD. Zubairul Haque

Degree Programme and place of study: Automation Technology / Intelligent Systems, Vaasa

Specialisation:

Supervisor(s): Ray Pörn

Title:

Date: 18.9.2020   Number of pages: 57   Appendices: 1

Abstract

Nowadays, product quality is a paramount concern in the modern machine manufacturing industry. Millions of dollars are being spent by modern machine manufacturing industries to maintain product quality. This expenditure on quality control can be reduced, and the efficiency of quality control can be enhanced, by reducing human contact and utilizing technology in quality control. Furthermore, automatic mechanical failure detection is an important technology in the fourth industrial revolution. Sound from the machine is used as the primary way to detect mechanical faults by quality control technicians. The concept behind this is that faulty machines must sound different from normal machines. After converting the sound data into a suitable digital format, machine learning algorithms can be used to identify abnormal sounds.

In this thesis, we have tried several machine learning algorithms to find the most suitable one for anomalous sound detection. Additionally, we have discussed the working principle and optimization process of these machine learning algorithms. To provide a reliable comparison between these machine learning algorithms, the anomaly scores for the same data were shown. Our aim was to provide a comprehensive understanding of the performance of these anomaly detection algorithms by analyzing the accuracy of detecting faulty machines.

Language: English
Key Words: anomaly, faulty machine, machine-learning, sound

Table of Contents

1 Introduction

During the past couple of years, manufacturing industries have been getting interested in integrating AI (Artificial Intelligence) based technology into their industries due to the power of AI in improving efficiency, accuracy, and decision-making. Researchers are working on implementing AI in various industrial aspects, for instance, researchers proposed an AI-based solution for machine health monitoring (R. Liu et al., 2018). AI was also utilized by another group of researchers for fault diagnosis in rotating machinery (Zhao et al., 2019). However, usually, most of these proposals do not completely comply with the exact requirements of a particular problem. This thesis is about analyzing the performance of machine learning algorithms in the sound-based detection of faulty sanding machines.

This chapter explains the importance and motivation behind this thesis. Additionally, it offers a summary of this thesis. Finally, it provides a short description of every chapter along with the arrangement of these chapters.

1.1 Relevance and motivations for the work

Detecting faulty products is extremely important for any machine manufacturing industry to maintain quality. Product quality is an important factor that has a significant impact on the growth of any business. David Garvin mentioned in his article that quality is essential to meet the customer’s needs and expectations (Garvin, 1987). Defective products also lead to a waste of time, money, and resources. Moreover, faulty products increase the possibility of damaging a company’s reputation and may lead to business loss. To maintain the quality of the product and to make sure that products meet the required standard, detecting faults in products can play a great role. Though technicians can detect mechanical faults in products, they do have limitations like making mistakes or missing faults due to human error, especially when they are tired or distracted. Moreover, skilled technicians require training, which is costly and time-consuming.

However, recent advancements in technology have introduced many automation systems to do industrial tasks more effectively, accurately, and quickly. Moreover, the integration of AI (Artificial Intelligence) technologies in the manufacturing industry has enabled machines to do such industrial tasks that were previously impossible without human intelligence. For example, advancements in artificial intelligence have resulted in a system that can identify mechanical faults in machines based on their sound. This kind of system identifies abnormal or anomalous sounds and from that, it identifies faulty machines. In most cases, anomalous sounds have been identified using a variety of techniques, the majority of which are machine learning based.

Machine learning is an important subset of Artificial Intelligent that allows a computer to be programmed with the self-learning ability to increase the performance of doing a specific task. Machine learning algorithms are mathematical expressions that can analyze large data sets of inputs and outputs to recognize the patterns to make autonomous decisions (Jordan & Mitchell, 2015).

The concept of detecting faulty machines based on sound is that normal machines sound almost the same. On the other hand, machines that have mechanical faults sound different from normal machines, which can be defined as an anomaly compared to normal machine sounds. Solution for detecting different types of anomalies is considered crucial and useful in many cases (Sharma et al., 2018). Even though anomaly detection is a well-studied problem, developing effective and accurate methods has proven difficult. In machine learning, anomaly detection can be defined as the process of recognizing patterns in data that do not match expected behavior (Chandola et al., 2009). An anomaly detection method must be able to model patterns in normal data to identify a typical sample.

Technological evolution is continuously increasing the demand for convenient solutions for new or existing problems and detecting sound anomalies can play an important role in this case. Many researchers have utilized various machine learning algorithms, including One Class SVM, Isolation Forests, and Autoencoders, for anomalous sound detection. For example, Rabaoui used one-class SVM to detect abnormal events in the audio surveillance context (Rabaoui et al., 2008). In another article, an autoencoder was used for detecting anomalous sound in a surface-mounted device (Oh & Yun, 2018). In sound anomaly detection, machine learning models are being trained with only regular sounds and then it can detect anomalous sounds. This type of solution minimizes required data, costs, and time.

1.2 Summary of the thesis

In this thesis, we have analyzed the performances of several one-class classifiers in detecting faulty sanding machines based on their sound. Sanding machines are revolving machines that are commonly used in factories to smooth surfaces of materials such as metal, wood, or concrete by abrasion with sandpaper. Figure 1 represents a picture of a sanding machine.

Figure 1: Sanding Machine

In this work, a set of sanding machines have been used as the monitoring target to see the performance of machine learning algorithms in faulty machine detection. These sanding machines are widely used in many industries for smoothing and shaping surfaces. A normal microphone was used to record sound from the sanding machines. To analyze the performance of machine learning algorithms in detecting anomalous sound from sanding machines, this thesis provides a quantitative evaluation of several machine learning algorithms which are one class support vector machine, one class K nearest neighbors, Isolation Forest, Principal component analysis (PCA) based autoencoder and one class K nearest neighbor with PCA, Local Outlier Factor (LOF), LOF with PCA. However, no anomalous sound was used to train these machine-learning models. The anomalous sound was only used in validation and to check the performance of these machine learning models.

The objective of this thesis is to provide an automated solution for detecting faulty machines using a normal microphone. One of the main contributions of this thesis is that it shows the way of using sound for detecting faulty machines and evaluates the performance of proposed machine learning algorithms in sound anomaly detection. In addition, it describes how to train machine learning models for detecting anomalous sound by using only normal sound. Moreover, this thesis provides an overview of the basic concepts of proposed machine learning algorithms and the way of optimizing proposed machine learning algorithms for better performance.

1.3 Thesis outline

This thesis has been divided into five chapters. The first chapter is the introduction which describes the importance of this thesis and the motivation behind this thesis. This chapter also contains a short overview of the whole thesis. In addition, it describes the organization of the thesis chapters.

The second chapter is the literature review, and it covers the previous work on data preprocessing for detecting acoustic sound. Moreover, it contains a literature review of previous work on acoustic anomaly detection.

The third chapter is the fundamental concepts and background, and it contains fundamental concepts and background information, which are essential to understand this work. It discusses the basics of digital audio signal processing. It explains the basic theory of machine learning algorithms for anomaly detection and the working principle of One class SVM, N nearest neighbors, Isolation Forest, and PCA.

The fourth chapter provides a detailed explanation of the techniques that were used in this project to detect faulty machines using anomalous sound and it is named methodology. This chapter describes how the sample was collected and organized for the project. Furthermore, it explains the way of extracting meaningful features from digital sound for further use. In addition, this chapter includes the details of the hyperparameter selection process for getting optimum performance from the machine learning algorithms. Finally, it describes what the ROC curve is and how it shows the performance of several machine learning algorithms.

In the fifth chapter, results have been shown and a quantitative comparison between targeted machine learning algorithms has been presented. It also shows the response of these algorithms towards different machine learning algorithms.

Then the performances have been evaluated and the reasons behind the difference in performance have been described in the sixth chapter, evaluation.

The last chapter is about the conclusions and future work. This chapter contains the complete analysis of this thesis topic. Besides that, in this chapter, a summary of the fundamental concepts has also been provided.

2 Literature review

2.1 Data preprocessing

When analyzing rotating machine components such as rolling element bearings or gears, frequency information is crucial for identifying potential issues and detecting anomalies. To capture the frequency information of the machine's behavior, it is often necessary to convert the time-domain signals into the frequency domain using techniques such as the Fourier Transform. Once the signal has been transformed into the frequency domain, feature extraction techniques can be applied to extract meaningful features that can be used to analyze the behavior of the machine. Lei et al. grouped rolling element-bearing conditions by using statistical features based on their frequency domain characteristics (Lei et al., 2008). In calculating frequency domain features, they also used the bearing signal model from (McFadden & Smith, 1984). In another article Minhua et al suggested a frequency domain multi-channel acoustic modeling to identify distant speech(Minhua et al., 2019).

2.2 Acoustic anomaly detection in the industrial context

Anomaly detection of sounds in an industrial context can help identify potential issues with machinery or equipment before they become serious problems, allowing for proactive maintenance, and avoiding downtime. There has been a significant amount of research in recent years on using machine learning algorithms to detect anomalies in industrial tools, applications, and processes. Chou et al. used Support Vector Data Description (SVDD) as part of a novelty detection module in a wafer quality prediction system for semiconductor manufacturing (Chou et al., 2010). In some articles, the One-class Support Vector Machine (OCSVM) technique is utilized to detect anomalies in various industrial applications like vibration-based fault detection in a kinematic chain(Cariño-Corrales et al., 2016). Another study examines and compares the results of abnormality detection techniques, including k-Nearest Neighbor (kNN), OCSVM, Local Outlier Factor, Principal Component Analysis, and Maximum Mean Discrepancy, in Prognostics and health management (Jia et al., 2017).

A survey paper on machine learning for industrial prognosis provides a review of different anomaly detection techniques and fault detection methods employed in industrial processes (Diez-Olivan et al., 2019). Moreover, another study discusses the use of in-process quality control (IPQC) during the process of inertia friction welding for critical components (Hartman, 2012). In this process, they use sound to maintain the quality of the inertia friction welding. The paper “Acoustic fall detection using one-class classifier” is one of the articles that discussed the use of a one-class classifier to detect anomalous sound. (Popescu & Mahnot, 2009). In this paper, the author used a one-class classifier to classify sounds as either falling or non-falling. In our project, we used sound to detect faulty machines. We tried to see the performance of a One-class Support Vector Machine (OCSVM), One-class nearest neighbors, Isolation Forest, and Principal Component Analysis (PCA) based autoencoder for acoustic anomaly detection.

With the development of deep learning techniques, Auto encoders have become increasingly popular for anomaly detection tasks, particularly when working with acoustic data. In a study researchers used a convolutional autoencoder and one-class SVM to detect faulty electric motors using machine’s sound (Son et al., 2022). Furthermore, in another study, the author made a comparison between a convolutional autoencoder (CAE) and OCSVM in acoustic anomaly detection in industrial processes and showed that CAE performs better than OCSVM (Duman et al., 2020). Some other articles on acoustic anomaly detection used several machine learning algorithms to detect acoustic anomalies, for instance, Salamon & Bello used deep convolutional neural networks and data augmentation for environmental sound classification (Salamon & Bello, 2017). While another group of authors offers an approach that utilizes a denoising autoencoder in conjunction with a bidirectional LSTM (Long Short-Term Memory) neural network to detect acoustic anomalies (Marchi et al., 2015). There is hardly any article that directly worked on the performance analysis of one-class classifiers in faulty sanding machine detection. We did a performance comparison of OCSVM, One-class nearest neighbor, isolation forest, LOF, and PCA-based autoencoder.

3 Fundamental concepts and background

3.1 Digital sound processing

Digital Signal Processing (DSP) refers to the application of mathematical algorithms and techniques to interpret and utilize digital signals. Digital Sound Processing (DSP) is the implementation of digital signal processing techniques to audio signals. It entails using mathematical algorithms to analyze, manipulate, and synthesize sound signals. FFT (Fast Fourier Transform) is a branch of digital signal processing (DSP) technique for analyzing and manipulating digital signals.

3.1.1 Fast Fourier transformation

The Fourier transform is a useful mathematical technique to transform a signal from the time domain to the frequency domain. Joseph Fourier, a well-known mathematician, first introduced this concept(Debnath, 2012). The Fourier transform is a widely used technique for analyzing and modifying continuous signals. However, when transforming digital signals such as digital audio from the time domain to the frequency domain, the discrete Fourier transform is used. The DFT can be defined as the mathematical operation that transforms a sequence of N complex numbers from the time domain to the frequency domain.

Mathematically, the DFT of a sequence x[n] of length N is defined as:

$$X[k] = \sum_{n=0}^{N-1} \left(x[n] \times exp\,(\frac{-2\pi ikn}{N}) \right) \tag{3.1}$$

Here, X(k) is the kth frequency component of the sequence x(n), and $exp\,(\frac{-2\pi ikn}{N})$ is a complex exponential function that represents the contribution of the nth sample to the kth frequency component.

In the DFT a straightforward matrix multiplication approach is being used, which takes O(N^2) time, here N is the size of the input sequence. The Fast Fourier Transform (FFT) is a mathematical algorithm that performs the same function as the Discrete Fourier Transform (DFT) but in a much faster manner. While the DFT takes O(N^2) time, the FFT has a much more efficient time complexity than O(N log N). To do that the FFT algorithm divides the sequence x[n] into two smaller sequences of even and odd-indexed elements. These subsequences are then recursively transformed using the FFT algorithm, and the results are combined to produce the final DFT of x[n]. In the field of digital signal processing, data analysis, and scientific computing, the FFT is a widely used technique. Figure 2 shows the FFT of a 10 Hz sinusoidal signal and its Fourier transformation.

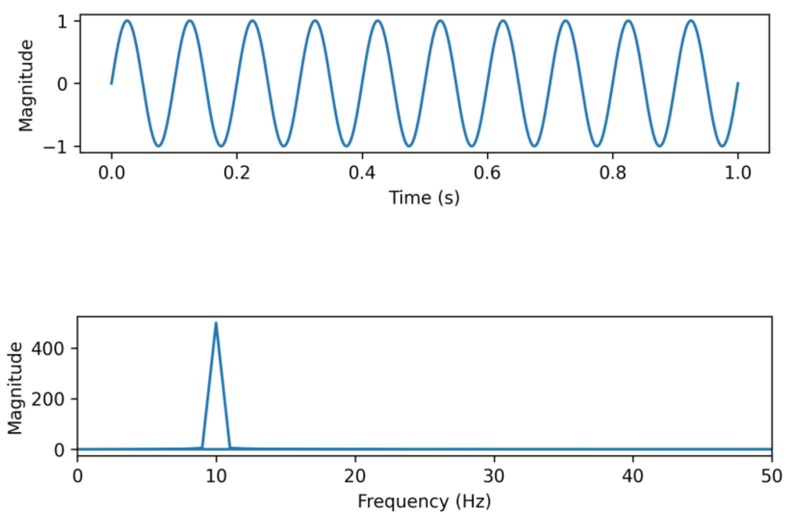

Figure 2: Sinusoidal signal (10 Hz) and its Fourier transformation.

3.1.2 Welch's method

Welch's method is a method that is used to estimate the power spectral density (PSD) of a signal using a periodogram of the signal. A periodogram is a mathematical tool that involves analyzing a windowed segment of a signal and computing the squared magnitude of its discrete Fourier transform (DFT) to estimate the power spectral density (PSD) of that signal (Rahi & Mehra, 2014).

Mathematically, according to Welch's method, the periodogram of a signal $x[n]$ of length N obtained by dividing the signal into K overlapping segments of length L can be written as:

$$P_{\{welch\}}[k] = \left(\frac{1}{K}\right) \sum_{(i=0)}^{(K-1)} |Xi[k]|^2 \tag{3.2}$$

Here, $Xi[k]$ = DFT of the i^{th} segment,

and $P_{\{welch\}}[k]$ = Welch PSD estimate at frequency k.

Window function $w[n]$ can be used to taper the signal before computing the periodogram. It improves the Welch PSD estimation. In this case, the Welch PSD estimate is given by:

$$P_{\{welch\}}[k] = (\frac{1}{KLw^2}) \times \sum_{(i=0)}^{(K-1)} |Xi[k]w[n]|^2 \tag{3.3}$$

Here, $w[n]$ = window function,

Lw = effective length of the window.

Usually, the Welch method is used to reduce the variance of the periodogram method by averaging. Figure 3 shows how the Welch method reduces the variance.

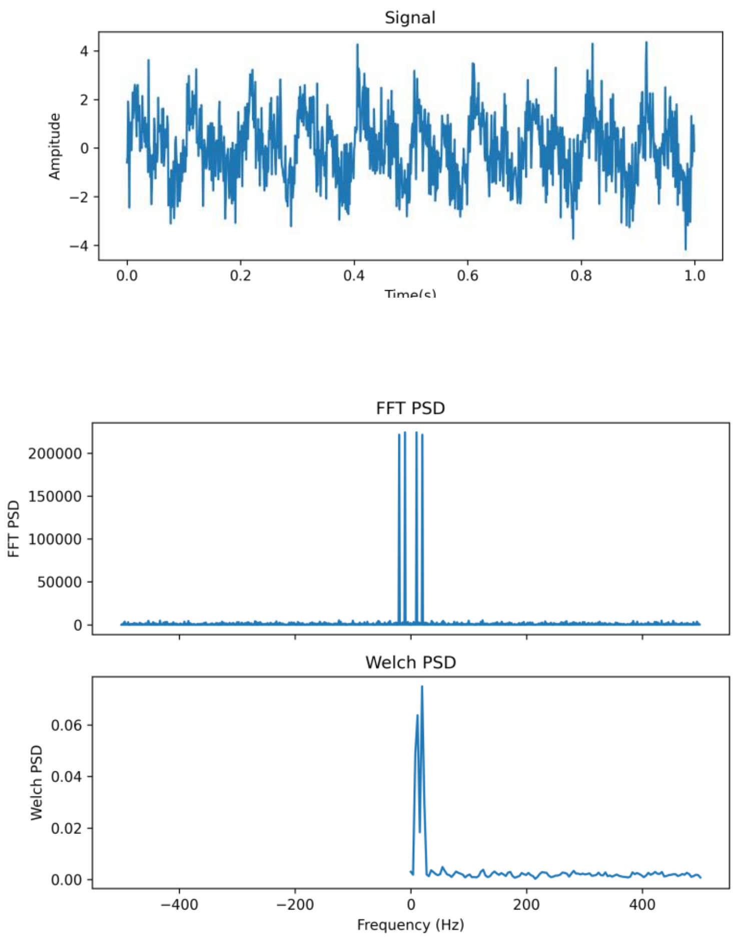

Figure 3: Signal, FFT PSD and welch PSD.

3.2 Data standardization

Data Standardization is a commonly used preprocessing technique in data analysis and machine learning. This technique transforms the data set so that it has a mean of 0 and a standard deviation of 1. To do that this method subtracts the mean from each data point and divides the result by the standard deviation. The formula of standardization is as follows:

$$S_i = \frac{(Xi - mean)}{std} \tag{3.4}$$

Here, X is the set of original data and i = 1,2, 3……. n, where n is the amount of data in the set X.

Mean is the mean of set X and std is the standard deviation of set X.

S is the set of standardized data

Data standardization is necessary if the features in the dataset have different scales. Data standardization is a very useful technique to improve the performance of many machine learning algorithms and to make the model faster. For example, the neural network gives better results for standardized datasets (Shanker et al., 1996).

3.3 Anomaly

In simple words, an anomaly is something that is not normal, standard, or expected. It can be referred to as an abnormal behavior, characteristic, occurrence, or event. In data analysis, an anomaly refers to a set of data that does not match the normal data in particular characteristics.

In this case, normal data can be defined as a set of data that has at least one similarity in a particular characteristic that can be used to distinguish them from anomalies. However, it is not mandatory that all the normal data must have similarities in all the characteristics. Normal data may have some differences in some of their characteristics, but they must have one or more similar characteristics that are common among them.

On the other hand, anomalous data can be defined as a set of data that contains data that deviate from the characteristics that are common in normal data. Here, an important thing is that anomalous data may have some similar characteristics to normal data, but those characteristics must not be the characteristics that define normal data. In short, anomalous data must have some characteristics that can be used to differentiate them from normal data, no matter whether anomalous data have other similarities with normal data or not.

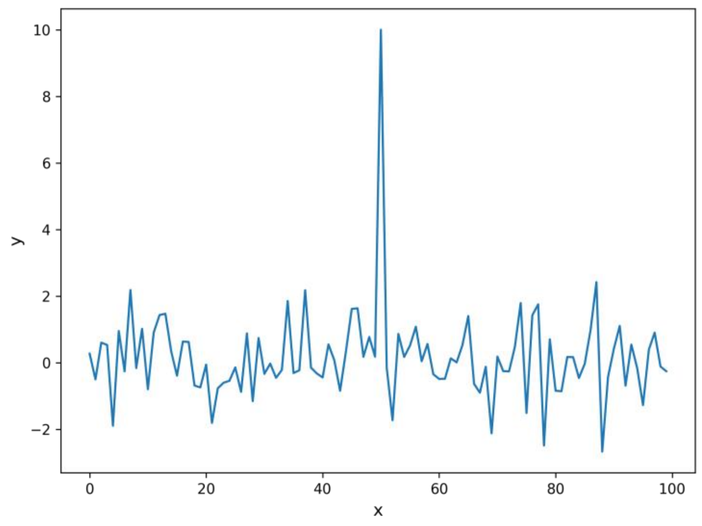

Figure 4: Example of anomaly

Figure 4 provides an example of what anomaly data looks like. In this example, normal data can be defined as data that has a y value between -3 to 3. Any data that has a y value more than 3 is considered anomalous in this example.

There are arguments about the similarity in characteristics among anomalous data. For instance, there are researchers who believe that it is not mandatory for anomalous data to have a common characteristic in order to be considered an anomaly (Data et al., 2016).On the other hand, others consider anomalies to be different from normal data but to have common characteristics among them (Günnemann et al., 2014). In our case, we adopted the first definition, where scholars state that it is not important for anomalous data to have a common characteristic in order to be considered an anomaly.

3.4 Anomaly detection

Anomaly detection refers to the process of identifying data points or observations that deviate significantly from the expected or normal pattern of a dataset. These anomalies can be caused by various factors, such as measurement errors, data corruption, or genuine outliers in the data. In machine learning, anomaly detection means the way of finding out the patterns of normal data and then detecting anomaly data that do not have the same pattern (Hawkins, 1980). In anomaly detection, an anomaly score is a quantitative measure that indicates the level of deviation of a data instance from the expected or normal pattern. Typically, higher anomaly scores are associated with data instances that are more anomalous, meaning that they deviate more from the normative behavior of the data. On the other hand, a lower anomaly score indicates the data instance is not that much anomalous and the amount of deviation from the normal data pattern is comparatively lower.

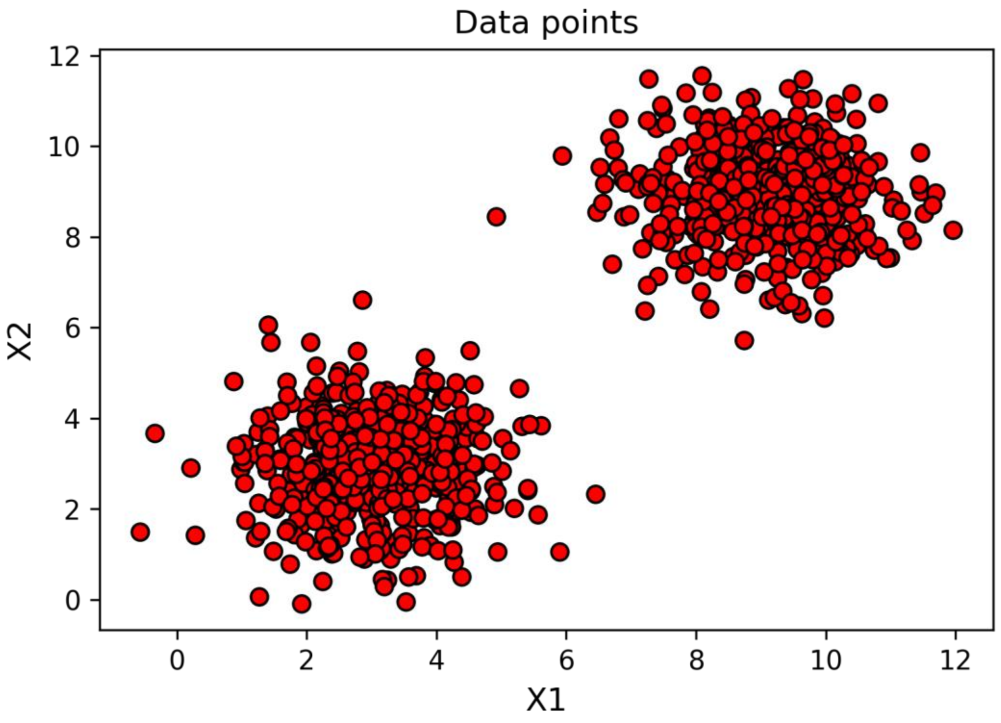

Figure 5: Sample Data

To describe the working principle of machine learning algorithms in anomaly detection, I have considered some synthetic data to train suitable machine learning algorithms for anomaly detection and tried to illustrate the approach of detecting anomaly by showing the anomaly score for each data point of the corresponding contour plot. The synthetic data are shown in Figure 5**.** These data are considered normal data during the training of the machine learning model. The entire contour plot was divided into 10 regions, labeled as levels 0 to 9, where each region represents the anomaly score for the data within it. I also used a black line to show the specific region mentioned in the corresponding section to demonstrate the phenomenon more specifically and clearly. ROC AUC scores have been used to measure the performance of these algorithms. This section also describes ROC AUC scores.

3.4.1 Proximity to mean

In general, considering proximity to mean is a very common process of defining any cluster. However, data from the same class could be distributed in a way that the mean of these data can be any value far away from the class. Let's consider a set of data illustrated in Figure 5 belonging to the same class. If we consider the proximity to mean here, it gives a lower anomaly score, for the data points that are far away from the normal class. Moreover, it gives comparatively higher anomaly scores for the data points that are nearer to the normal data points, which is illustrated in Figure 6. In Figure 6 the region inside the black line represents the region where anomaly scores are lowest. So, it is clear that considering the mean value for clustering may not be a good idea every time.

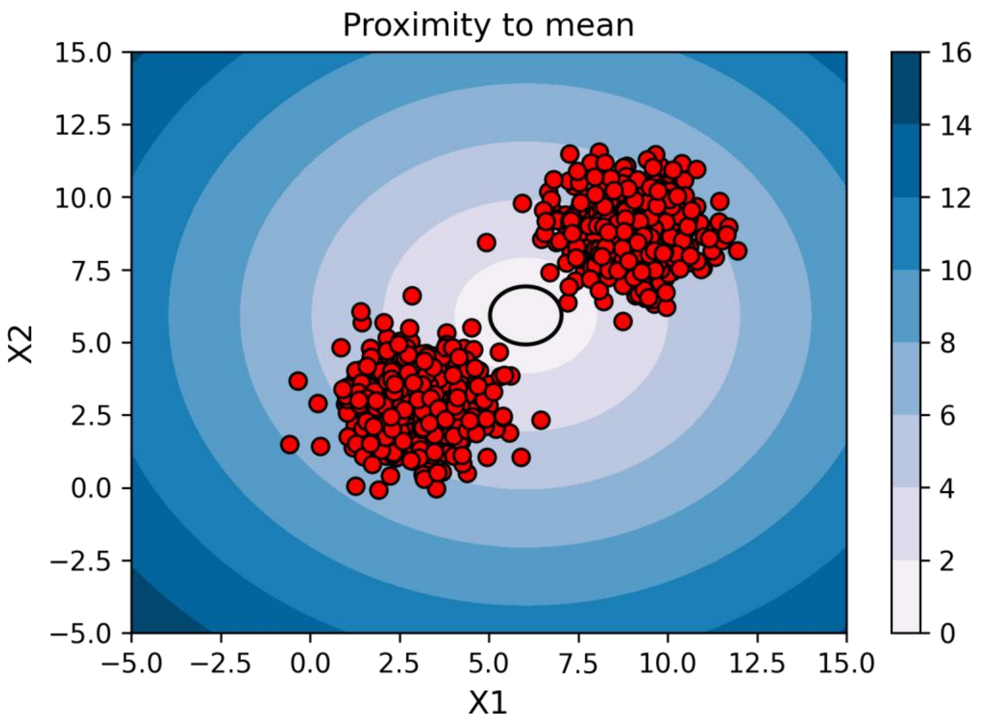

Figure 6: Anomaly Scores for proximity to mean.

3.4.2 One class SVM

In 1992 Vapnik introduced a supervised algorithm, which is known as a Support vector machine nowadays (Kao, 2008). Support vector machine (SVM) is a machine learning algorithm, which uses hyper-plane to distinguish different classes. The concept behind choosing the hyperplane is that the hyperplane should be chosen in a way that keeps the maximum marginal distance from the classes.

One class SVM is the modified version of the SVM machine learning algorithm, where only one class of data is considered normal data and any data out of this class is considered an anomaly. In the case of One-class SVM, as it supports only one class, it draws the hyperplane between the origin (zero vector) and the selected class. It is a semi-parametric one-class classification method, which means that it does not require any functional form of the distribution of the normal data to be explicitly specified beforehand. Instead, the method creates a boundary around the normal data to separate them from the anomalous data.

Using One class SVM, if training features vectors consisting of $xi \in R^d$ $(i = 1, 2, ... \iota)$ have been projected into a higher dimensional space with the help of a feature map ϕ, a hyperplane is then found that separates the projected examples from the origin with a maximum margin. For doing so, one class SVM has to solve an optimization problem, which is as follows:

$$min_{\omega,\xi,\rho} \frac{1}{2}||\omega||^2 + \frac{1}{\upsilon\iota}\sum_{i=1}^{\iota} \xi_i$$

$$subject\ to : \left(\omega, \varphi(X_i)\right) \geq \rho - \xi_i$$

$$\xi_i \geq 0\forall_i \qquad (3.5)$$

Here, $\omega \in R^d$, ι is the number of training samples, and $0 \leq \upsilon \leq 1$. Moreover, $\xi i \geq 0$, which loses the constraints of the problem so that a few examples can fall outside of the boundary. This problem must have a solution with three separate example sets: the set of examples that are inside the boundary (non-support vectors), the set of examples that are on the boundary (border support vectors), and the set of examples that are outside the boundary (outliers or bounded support vectors). The one class SVM has the property that the user-defined hyperparameter υ defines the fraction of training data to be allowed to fall on or inside the decision boundary. The kernel trick can be applied to distinguish normal data points and outliers even which are not linearly distinguishable in the input space by using a kernel function as φ (Ghafoori et al., 2016).

3.4.3 Hyperparameters of One class SVM

υ (nu): υ (nu) is the hyperparameter for the underlying SVM model. υ decides the fraction of training data be allowed to fall on or inside the decision boundary. Since the training errors could not be more than the number of total training data, υ must be in the interval of 0 to 1.

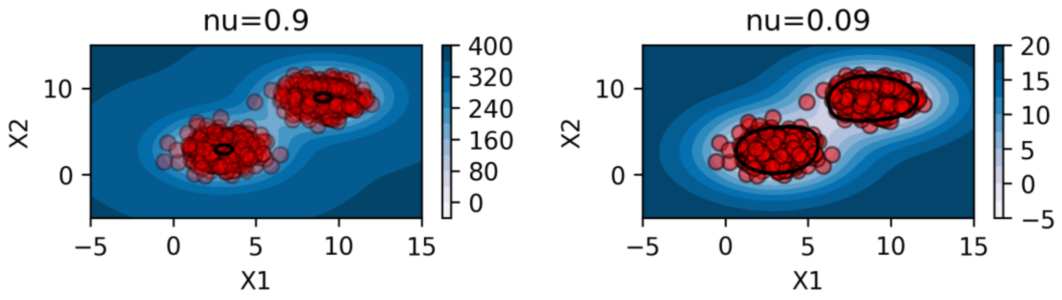

Figure 7: Effect of nu value on one class SVM

In Figure 7 both the contour plots were divided into 9 regions. Every region represents the anomaly score for the points in that region. In Figure 7 the black line represents the level 1 region (decision boundary). This black line helps to understand how the levels get changed for different nu values. Data that are inside the line belong to the same level. If we want to classify the data in the contour plot using this black line, the data that are inside the line can be considered data from the same class. Figure 7 clearly shows that when υ value is 0.9 almost 90 percent of the training data is considered as out of level 1 region. On the other hand, when υ value is 0.09 only 9 percent of the normal training data is out of level 1 region. So, from here, we can understand how training data is considered during the training of the model. Moreover, this figure also shows that when the υ value is 0.9, the level 1 region is shorter than that when the υ value is 0.09.

Gamma: Gamma is a parameter for a particular kernel (rbf). Gamma defines the sensitivity of the decision function to feature variation. It can be any real number. Gamma decides the curvature in the decision boundary (Pedregosa et al., 2011). To understand the effect of gamma I have considered synthetic data shown in Figure 5 and plotted a contour plot in

Figure 8, where the anomaly score of every point of the contour plot for corresponding gamma values has been plotted. It shows that for gamma=0.3 curvature in the decision boundary is more than the curvature in the decision boundary for gamma=0.09. In Figure 8, the black line represents the level 0 region (decision boundary), which shows that for gamma = 0.3, the level 0 region is more precise and correct compared to gamma = 0.09. Concisely, a higher gamma value means testing data must match more accurately to the training data features than the model with lower gamma values.

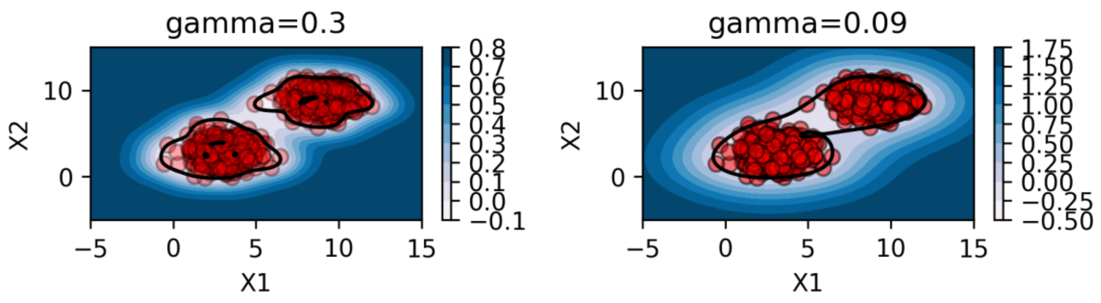

Figure 8: Effect of gamma on One-class SVM

3.5 K-nearest neighbors

Fix and Hodges introduced a non-parametric method of pattern classification in 1951 in an unpublished report of the US Air Force School of Aviation Medicine, which has since become known as the k-nearest neighbor rule (Peterson, 2009). It assumes that similar things have less distance. Therefore, it considers the distance from the neighbors' data to classify the data sets. It considers the distances between the N number of neighbor samples to decide whether the data instance belongs to the class or not. If the data instance has a smaller average distance from the instances of a particular class than the instances of other class or classes, then the data instance belongs to the class that has a smaller average distance. In the case of one class K-Nearest Neighbors, after considering n number of training samples, if the average distance from the instances of the class is smaller than the average distance from the origin, then the data instance belongs to the class. On the other hand, for larger average distance from the class means the data instance does not belong to the class. The Euclidian distance that is considered to find the distance between the specified training samples x_l *(l = 1, 2, 3…., n)* and a test sample x_i can be expressed as follows:

$$d(x_i, x_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + \dots \dots \dots \dots .. + (x_{ip} - x_{lp})^2} \qquad (3.6)$$

Here, x_i is an input sample that has p number of features *(x_{i1}, x_{i2}, x_{i3}, ……, x_{ip})* and the total number of input samples is *n (i = 1,2,3….,n)* with the features number of *(j = 1,2,3,….,p)* (Peterson, 2009).

3.5.1 Hyperparameters of N-nearest neighbors

N-Nearest neighbors: Number of neighbors to be used for n neighbors' queries. It must be an integer. A higher n neighbors value lets the algorithm consider more neighbors during n neighbors queries like training the model or during validation. In other words, N-Nearest neighbors define the sensitivity of the decision function to feature variation. Considering more samples has both positive and negative impacts on the algorithm's performance. N Nearest neighbors should be chosen in such a way that it lets the algorithm give better results. Figure 9 shows that for n=160 the anomaly score is more sensitive than the anomaly score for n=2. This means, in the case of n = 160 anomaly scores get higher for more little distances than that of n = 2. It also controls the curvature in the decision boundary. It is clearly visible in Figure 9 that curvature in the decision boundary for n = 160 is much smoother than the curvature in the decision boundary for n = 2. In this figure, the black line represents the level 1 region (decision boundary). Here we can see that level 1 is smoother and smaller for n-neighbors = 160 compared to n-neighbors = 2.

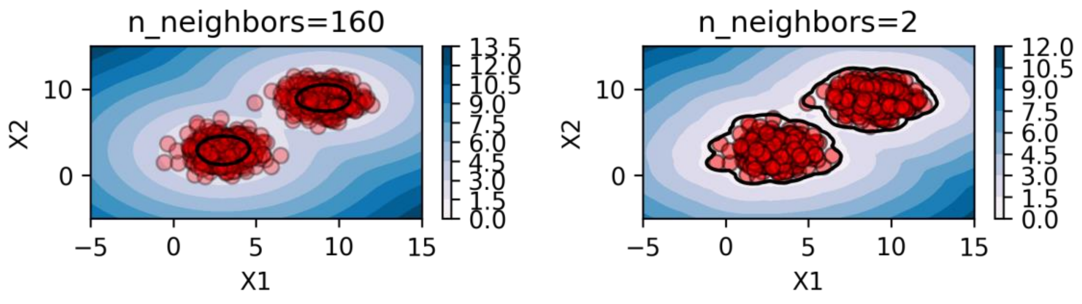

Figure 9: Effect of neighbor numbers on Nearest Neighbors

3.6 Isolation forest

The decision tree is a tool, which employs a tree-like representation of options and their potential outcomes for decision-making. This decision tree is the base estimator of the Isolation Forest, which is an anomaly detection algorithm. Isolation forest builds an ensemble of isolation trees for a given data set, then anomalies are those instances that have short average path lengths on the isolation Trees. At first, it selects a feature at random and then selects a split value between the maximum and minimum values of the selected feature to isolate the observation. Recursive segmentation can be portrayed as a tree structure. So, the splitting number to isolate a sample is equal to the length of the path from the root node to the terminating node. The measure of normality and the decision function is obtained by averaging the length of this path among a forest of similarly random trees. For anomalies, random partitioning results in considerably shorter pathways. As a result, shorter path lengths for specific samples produced by a forest of random trees are quite likely to be anomalies. Every instance has a score that is determined by the average depth of isolation trees in the isolation forest, which is a measure of how likely it is to be an anomaly. Formula 2.3 is used to determine the anomalous score S(x, N).

$$S(x,N) = 2^{\frac{-E(h(x))}{c(N)}} \tag{3.7}$$

Here, N stands for the subsample size, *and* $h(x)$ is the average search height for instance x from the isolation trees. $E(h(x))$ is the average of $h(x)$ from a group of isolation trees, and $c(N)$ is the average of $h(n)$, where $h(N)$ is the average search height for any instance N from the isolation trees. Isolation forest does not use density or distance measures like One class SVM or K-Nearest neighbors. As a result, it overcomes the issues regarding density or distance-based isolation processes. Most anomaly detectors identify normal data rather than identify anomalies. The outcomes of anomaly detection may therefore not be as good as anticipated, resulting in too few anomalies detection. In this case, Isolation Forest could be a good option to detect anomalies more precisely.

3.6.1 Hyperparameters of isolation forest:

The number of isolation trees and the size of the subsample are the two variables in this method (F. T. Liu et al., 2008).

Max-sample or subsampling size: Max-sample or subsampling size is the maximum number of samples that will be taken to train each base estimator of the model. A higher max-sample ensures the model’s accuracy. There is no need to increase subsampling size after it reaches a desired value because doing so increases processing time and memory size without improving detection performance. Figure 10 clearly shows that when the Max-sample=10, the anomaly score for anomalous data is quite low. On the other hand, the anomaly score for anomalous data is comparatively higher for max sample=100. In addition, the model can define the level 0 region more precisely for max sample = 100, as indicated by the black line representing the level 0 region (decision boundary). From Figure 10, the level 0 region for max-samples = 100 is more accurate compared to the level 0 region for max-samples = 10.

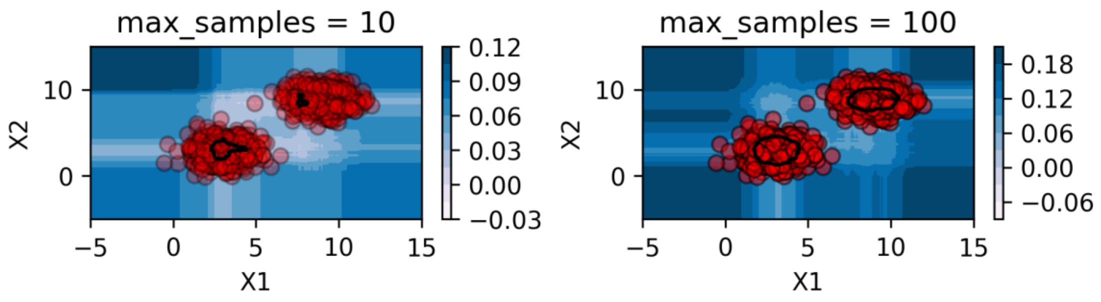

Figure 10: Effect of subsampling size on Isolation Forest

N-estimators or the number of isolation trees: N-estimators or the number of itrees (isolation trees) is the number of base estimators that will be used in the ensemble to build the isolation forest. It has a great impact on the performance of isolation forest. The isolation forest takes the average result from n number of decision trees and every decision tree considers features and samples randomly. As a result, if the number of trees increases, the chances of getting biased results decrease. The accuracy of the isolation forest is linearly proportional to the number of itrees (isolation trees). If the number of itrees (isolation trees) increases, the accuracy of the isolation forest also increases. Figure 11 clearly shows that the anomaly score of anomalous data is quite higher for n-estimators = 600 and vice versa for normal data compared to the anomaly score for n-estimators= 2. Furthermore, it is noticeable that the model gives higher accuracy in classifying data with n-estimator = 2 compared to n-estimator = 600. In Figure 11, the black line represents the level 0 region (decision boundary), and the figure clearly shows that the level 0 region is defined more accurately by the n-estimators = 600 compared to n-estimator = 2. In addition, the level 0 region is smoother when the n-estimator = 600.

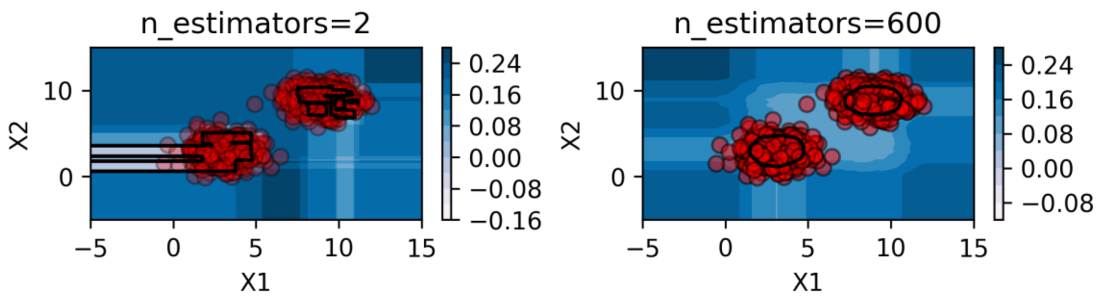

Figure 11: Effect of the number of itrees on Isolation Forest

3.7 Local outlier factor

The local outlier factor is a density-based outlier detection algorithm proposed by Breuning et al. (Breunig et al., 2000). It is an unsupervised machine learning algorithm that measures the distances among data to find the density of that data point. A point's local density is determined by its average distance from its closest neighbors, with higher densities indicating points that are closer to their neighbors. The LOF approach operates under the assumption that anomalies are typically more isolated than regular data points, resulting in anomalies having a lower local density. The local outlier factor (LOF) employs a metric known as the local outlier factor score, which is determined by dividing the local density of a given sample point by the average local density of its nearest neighbors. The operation of the Local Outlier Factor can be divided into five steps.

1) In the first step of calculating the local outlier factor, *k* nearest neighbors of the target data point p is determined. Nearest neighbors include all the data points that are within the range of *k* distance. Let’s consider all the data objects within the k distance of target data *p* are $N_k(p)$. Then we can express $N_k(p)$ as follows:

$$N_k(p) = \{q \in D \setminus \{p\} / d(p, q) \leq k\text{-}dis(p)\} \tag{3.8}$$

Here, q is the set of data that are within *k*-distance of target data p.

D is the data set and *d(p,q)* is the distance between *p* and *q*

2) The second step is calculating the reachability distance of each neighbor point to the target data point. The reachability distance between a point *o* and the target point *p* can be written as

$$reach\text{-}dis_k(p, o) = max\{k\text{-}dis(o), d(p, o)\}$$

Here, if the distance between *o* and *p* is equal to or less than the k-distance the reachability distance is equal to the k-distance and if the distance between *o* and *p* exceeds the k-distance than the reachability distance is the actual distance between *o* and *p*.

3) After that the local reachable density of point *p* is calculated. Local reachable density is basically the inverse of the average reachability distance of its k nearest neighbors. Reachable density can be written as follow:

$$lrd(p) = \frac{1}{\frac{\sum_{i=1}^{n} reach - dis_k(p, o_i)}{N_k(p)}}$$

$$= \frac{N_k(p)}{\sum_{i=1}^{n} reach - dis_k(p, o_i)} \tag{3.9}$$

Here, $o_i \in N_k(p)$ and i= *1,2,3, 4...n*. The *reach-dis*$_k$ *is d(p, o*$_i$*)*, when *p* has a large deviation. In this case, the neighborhood data are fewer which means the $N_k(p)$ is lower, and it causes lower reachable density. On the other hand, reach-dis$_k$ is k-distance *(o)*, when *p* is in a more clustered group. In this case, the neighborhood data are more, the $N_k(p)$ is higher, and it causes higher reachable density.

4) In the last step local outlier factor score for target *p* is determined by taking the ratio of the average density of the data objects in the neighborhood of target *p* to the local reachable density of target *p*. The LOF score of target *p* can be written as:

$$LOF(p) = \frac{\frac{\sum lrd(o_i)}{N_k(p)}}{lrd(p)} \tag{3.10}$$

3.7.1 Hyperparameter of local outlier factor

k-neighbors: k-neighbors are the number of neighbors considered for calculating the k-distance in the LOF algorithm. The local density of a point is determined by calculating the average distance between that point and its "k" closest neighbors. All data objects that are within the k-distance are considered as "k" closest neighbors.

If the value of k is smaller, the sensitivity to local density changes gets higher. This means that smaller values of k are suitable for detecting local outliers. On the other hand, when the value of k is larger, sensitivity to local density changes gets lower. This means that larger values of k are suitable for detecting global outliers.

From Figure 12 the black region represents the level 0 region in both the pictures. This figure shows that when k-neighbors are higher the decision boundary (level 0 region) is smoother compared to the decision boundary for lower k-neighbors. In addition, the LOF score is distributed more uniformly for higher k-nearest neighbor values.

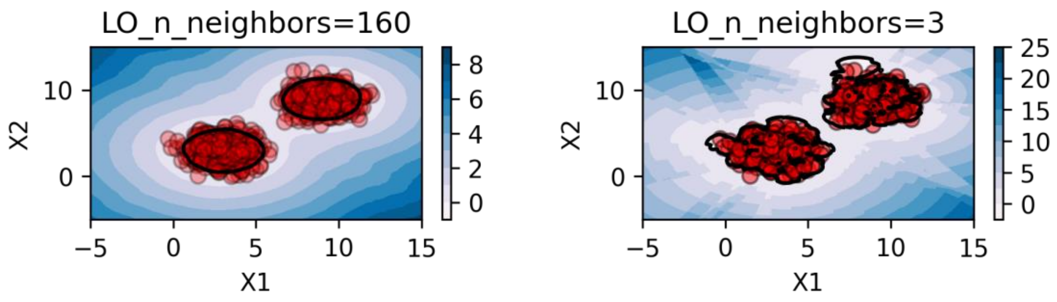

Figure 12: Effect of K-neighbors on Local Outlier Factor

Contamination: in the LOF, contamination refers to the proportion of outliers in the training dataset. It is a hyperparameter that specifies the expected percentage of anomalous data points in the dataset. This contamination has a direct impact on the performance of LOF. When the contamination is high, LOF misclassifies normal data points as outliers. On the other hand, when the contamination is low, LOF may consider outliers as normal data points.

In Figure 13, the black line represents the level 0 region (decision boundary) for both pictures. In Figure 13 when the contamination is high, LOF considered less training data points compared to the training data points at the time of lower contamination value. At the same time, we can see that if the contamination value is high, the misclassification rate of anomaly is comparatively higher for the decision boundary.

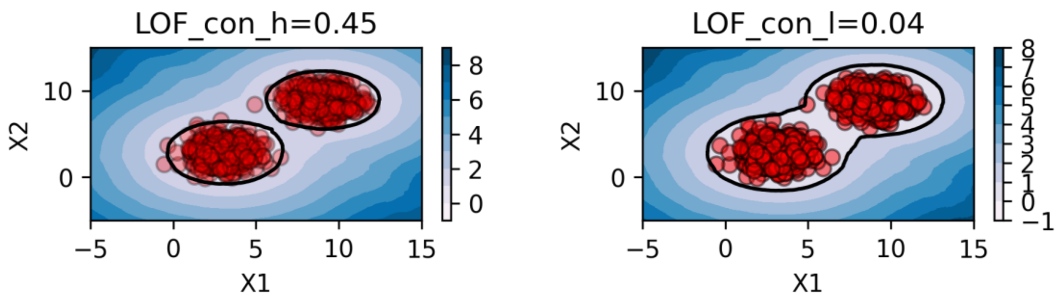

Figure 13: Effect of contamination on Local Outlier Factor

3.8 PCA

Principal component Analysis (PCA) is a useful and simple method to extract important features from high-dimensional vectors by reducing the dimensionality (Bo & Wu, 2009; Smith, 2002). This method creates a set that contains orthogonal-uncorrelated variables, and these sets are known as principal components (PCs) (Gorgoglione et al., 2021). These sets have a linear relation with the original dimensions (Arriola et al., 2020). The importance of each principal component can be found in the eigenvalue. PCA can be used to find out possible emerging patterns that can be invisible when one original feature is considered at a time (Gorgoglione et al., 2018).

PCA defines a line that minimizes the average squared distance from the data points. Then PCA projects those data points on that line. This line is known as the principal component and the unit vector across the PCA is known as the eigenvector. The main goal of PCA is to represent these data sets in a lower dimension with the most possible variation.

3.9 Autoencoder

An autoencoder consists of two networks, the first one is an encoder, which compresses the data into a lower-dimensional representation. The other one is the decoder, which reconstructs the data from the compressed representation. Figure 14 represents the block diagram of an autoencoder. PCA can be used as an autoencoder because it has the property of compressing data into a lower dimension and then reconstructing that data from the compressed representation.

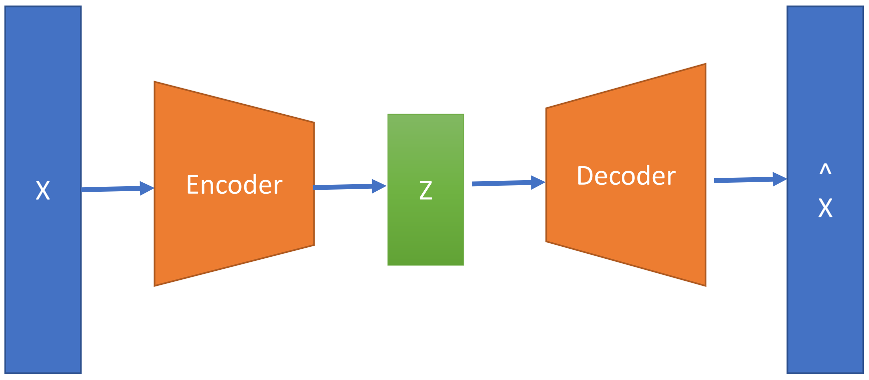

Figure 14: Auto Encoder

Data that are projected in the lower dimension during PCA, could be reconstructed with some error, which is known as reconstruction error. Data that has less deviation from the pattern of the training set also has low reconstruction errors. In contrast, data that have more deviation from the pattern of the training set also have high reconstruction errors. Here, I have used PCA as an autoencoder and I considered reconstruction errors to find the anomaly. Any anomaly must be out of this line of normal data distribution pattern and that results in high reconstruction error.

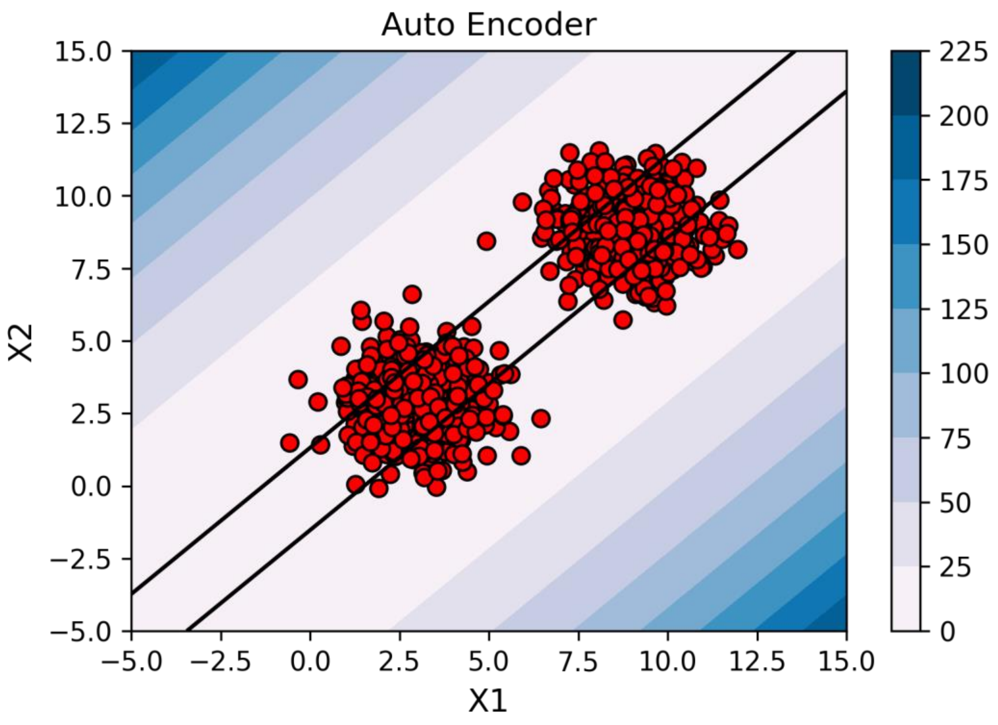

Figure 15: Anomaly score for PCA-based autoencoder.

Figure 15 shows the reconstruction error that is considered an anomaly score for every point of the contour plot. In this figure, the black line represents the level 1 region (decision boundary). If we analyze this figure, we can see that the anomaly scores are the same for the area between the two black lines, and the changes in the anomaly score follow a parallel path in this case. This gives us an idea of how the reconstruction error works in a PCA-based autoencoder.

4 Methodology

4.1 Overview

This chapter discusses how the project was conducted. First, it describes the sample selection process, the sound recording process, and the process of listing the recordings. Next, it outlines the steps taken to convert the recorded sound into a suitable format for analysis. Finally, the chapter covers the hyperparameter tuning of machine learning algorithms for the project.

4.2 Sample selection

To conduct this project, thirty sanding machines were acquired. Of these thirty machines, twenty are in good condition and ten have some kind of fault. Each machine was assigned an ID number. The conditions of the machines are presented below:

Old machine List		
Machine ID	State	Fault ID
1	Good	0
2	Good	0
3	Faulty	1
4	Faulty	2
5	Faulty	2
6	Faulty	2
7	Faulty	3
8	Faulty	3
9	Faulty	3
10	Faulty	4
11	Good	0
12	Good	0
13	Good	0
14	Good	0
15	Good	0
16	Good	0
17	Good	0
18	Good	0
19	Good	0
20	Good	0
21	Good	0
22	Faulty	2

23	Faulty	5
24	Good	0
25	Good	0
26	Good	0
27	Good	0
28	Good	0
29	Good	0
30	Good	0

4.3 Recording process

The length of each recording was 30 seconds. Each recording was tagged with the corresponding machine number, the machine's rotational speed, and their corresponding tags (z and z1) and (j and j2). A normal microphone was used to record the sound and the recordings were done in six steps as follows:

Recording List					
Sl. No	Data set	Recordings	RPM	Tags	Machine's Position
1	Fixed1	recordings-22-02-21	4k, 10k	z	Fairly Fixed
2	Fixed2	recordings-22-02-23	4k, 10k	z2	Fairly Fixed
3	Distance	recordings-22-03-02	4k	z	At different distances
4	Moving1	recordings-22-03-16	4k	z	Moved randomly
5	Moving2	recordings-22-03-29	4k	z2	Moved randomly
6	Fixed3	recordings_09_20	4k,10k	j, j2	Fairly Fixed

4.4 Data preprocessing

Data preprocessing is a crucial step in the data analysis pipeline, where raw data is transformed into a more usable and understandable format. It involves a series of techniques and methods to clean, transform, and organize the data with the goal of improving its quality and making it suitable for analysis.

The process of data preprocessing includes several important steps, such as data cleaning, data integration, data transformation, data reduction, and data normalization. These steps help ensure that the data is consistent, complete, and accurate and can be analyzed effectively. Overall, data preprocessing is a critical step in the data analysis process, and it requires careful planning and attention to detail to ensure that the data is accurate and reliable.

In this project, which focuses on anomalous sound detection, transforming the sound signal into meaningful and suitable data is crucial for the analysis. The data preprocessing for this project can be divided into two parts. The first part involves dividing the signal into equal segments. In this process, segment borders were calculated to divide the signal into equal parts. The second part involves finding the spectrum of each segment and storing it in a suitable format. The entire process is described in detail in the following sections.

4.4.1 Segmentation

The raw sound data consists of recordings that last approximately 30 seconds at both 4000 RPM and 10000 RPM. Each recording underwent a Fourier transformation process to convert the time domain signal into the frequency domain.

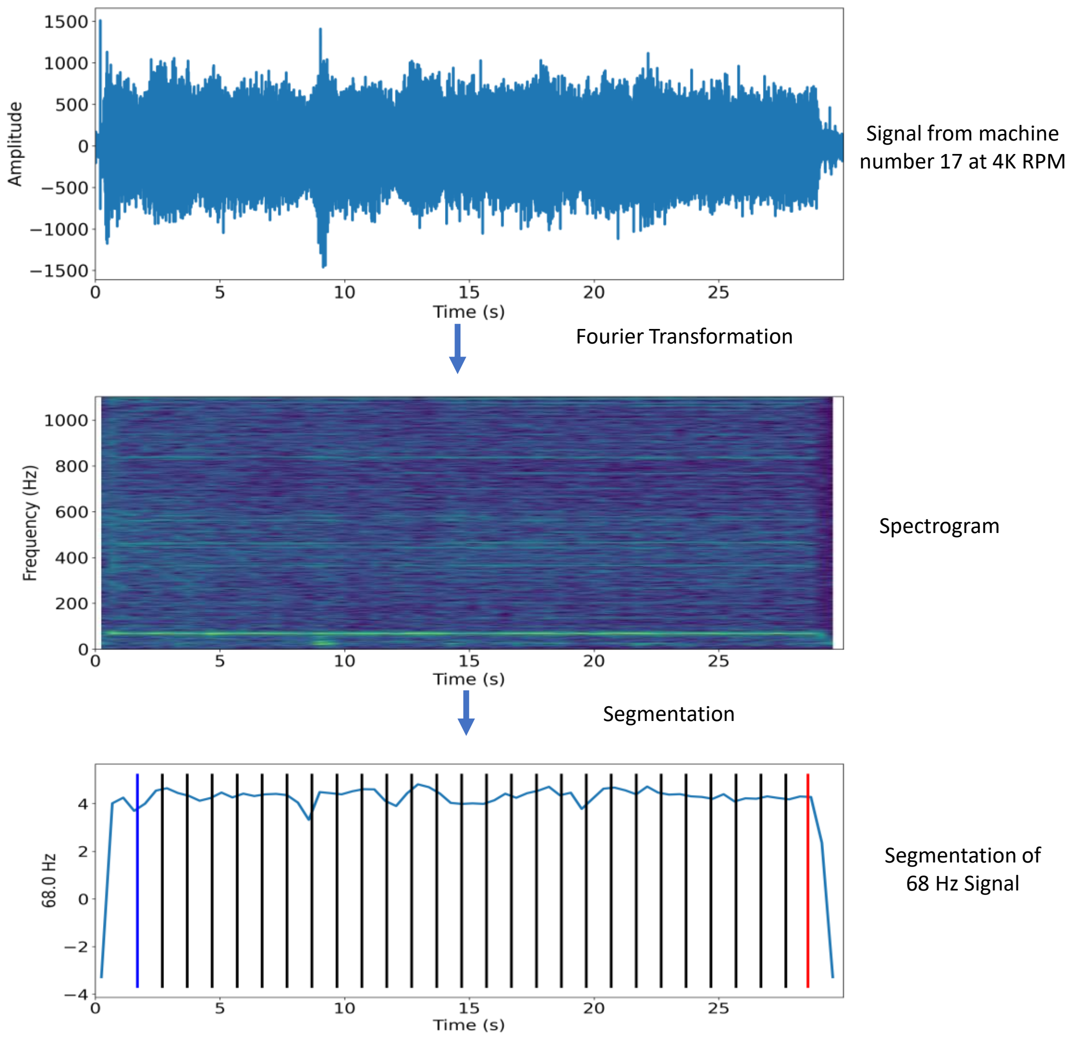

Figure 16: Fourier Transformation of the sound signal from machine number 17

Figure 16 depicts the sound signal from machine number 17. After collecting the sound signal in the time domain, a Fourier transformation process was applied to convert the signal into the frequency domain and generate the spectrogram. To expedite the computation, the sampling rate of the spectrogram was set to 20 times lower than the original recording’s sampling rate. Following the Fourier transformation, a logarithmic transformation was applied to each frequency of the spectrogram to account for the high amplitude differences between frequencies and enable easier comparison.

Next, the RPM frequency was identified as the frequency with the highest amplitude. The onset and offset times of the RPM frequency were determined by identifying the time at which the frequency's amplitude exceeded and then dropped below 50% of its highest amplitude, respectively. These onset and offset times were then used to extract the onset and offset times of the entire signal. The full signal was then divided into non-overlapping segments, with each segment lasting one second. A spectrogram was computed separately for each segment.

4.4.2 Spectra calculation for each segment

After obtaining the segments, the spectrogram for each segment was computed separately. The frequency resolution of each spectrogram was set to 10 Hz, and the cutoff frequency was set to 6000 Hz. To achieve the desired frequency resolution that includes the RPM frequency data, the data was processed in several steps.

First, the frequency with the highest peak value was calculated by performing a Fourier transformation of the segment. This frequency corresponds to the speed of the machine. Then, this frequency was used to determine the actual RPM of the machine. The actual RPM was rounded up to obtain the intended RPM. Next, the time axis was stretched to ensure that all segments had the same RPM by considering the ratio between the actual and intended RPMs. This stretched time axis was multiplied by the sampling rate to obtain a new sampling rate that ensured the presence of the RPM frequency in the spectrogram. Using the new sampling rate and Welch's method for averaging periodograms, the desired frequency resolution could be obtained in each spectrum. Figure 17 shows the spectrum of each segment for machine number 17.

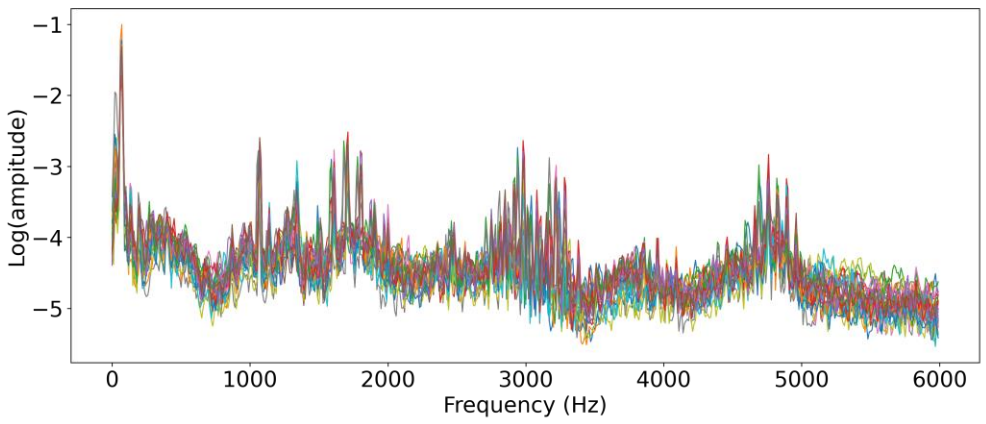

Figure 17: Spectra of each segment for machine nr 17

Afterward, all spectral data of each segment were stored in a data frame. Each row in the data frame, except for the header row, represents a segment. The corresponding machine ID, status, fault code, tag, segment ID, RPM, onset index, offset index, and frequency for each segment were listed in the data frame. Finally, the data frame was saved as a comma-separated values (CSV) file for future analysis. All training, validation, and testing data were standardized prior to validation and testing.

4.5 Hyperparameter tuning.

4.5.1 ROC curve:

One way of measuring the performance of machine learning algorithms is by analyzing the confusion matrix or truth table. However, analyzing the truth table for every single classification threshold to find the performance of machine learning models can be quite challenging. A ROC curve is a graphical representation of the classification model's performance for different classification threshold settings. It shows the rate of correctly predicted positive data as well as the rate of false positives of the model's prediction. Therefore, the ROC curve can be considered a performance measurement of the classification model across the entire range of class distribution as well as error cost. (Rakotomamonjy, 2004)

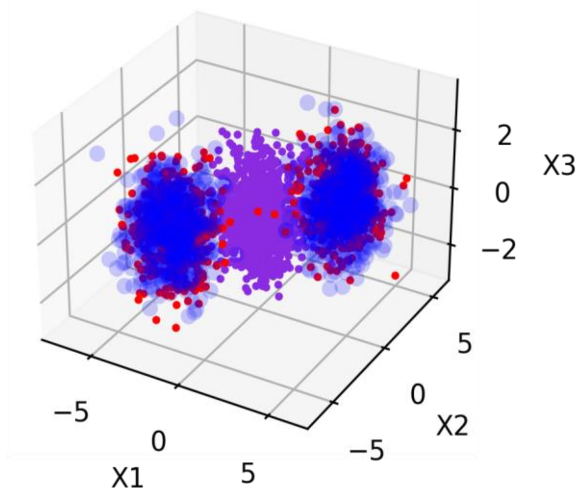

Figure 18: Sample 150-dimensional data (represented using first 3 dimensions)

4.5.2 Working principle

In general, the process of performing decision rules is to select a decision threshold that separates the classes, so this decision threshold must be selected in a way that minimizes the classifier error. However, the optimal threshold varies within a large range. Every decision threshold gives a pair of true positive and false positive performance rates. Using these true positive and false positive performance rates, a ROC curve can be found for every decision threshold. As the ROC curve uses only true positive and false positive performance rates, the model's performance can be determined without knowing the class distributions and error costs. The value of the area under the curve (AUC) is the most used performance measure extracted from the ROC curve. AUC = 1 means that the classifier's accuracy is 100 percent. Therefore, the corresponding decision threshold is the best choice for the model. (Rakotomamonjy, 2004)

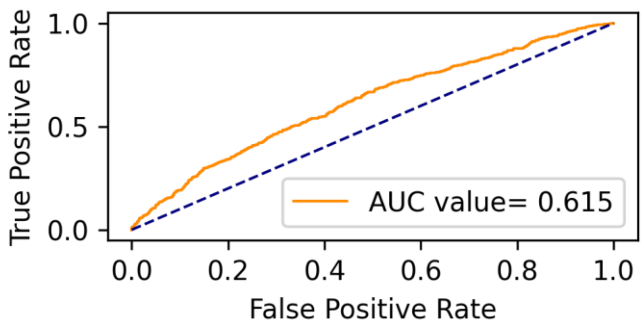

Figure 19: Receiver operating characteristic for SVM

I have considered some sample data shown in Figure 18. The sample data illustrated in Figure 18 have the following combination:

1) Training data (red)

2) Testing data (blue)

3) Noise (blue violet)

Figure 19 is an example of the ROC curve of the SVM classifier with nu=0.1 and gamma= 0.1 for the data illustrated in Figure 18. To demonstrate the receiver operating characteristic (ROC) curve, we have randomly chosen the nu and gamma values. The orange curve is the ROC curve for the model. The diagonal blue dotted line shows the ROC curve of SVM that predicts the class at random. The performance gets better as the curve goes near the upper left corner of the plot.

4.5.3 Data selection for hyperparameter tuning:

The hyperparameters must be selected in a way that enables the model to accurately predict the maximum amount of data, and the ROC AUC score is an excellent metric for evaluating the percentage of true positive predictions. While grid search is a common method for finding suitable hyperparameters, I tried to illustrate how the ROC AUC score changes for each hyperparameter. A contour plot can display the optimal hyperparameter settings for the algorithm and their corresponding ROC AUC values. To demonstrate the hyperparameter tuning process, I utilized spectral data from the 'recordings_09_20' dataset. For training, I used machines 1, 2, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 24, and 25 with tag 'j', and for validation, I used machines 6, 7, 8, 9, 10, 23, 26, 27, 28, 29, and 30 with tag 'j2'.

4.5.4 Hyperparameter of one class SVM:

Figure 20 illustrates the corresponding ROC AUC scores for hyperparameters gamma and nu, with the color bar indicating the corresponding colors for ROC AUC scores. The figure shows that possible nu values have a minimal effect on the ROC AUC score, while gamma values have a significant impact, as demonstrated in the figure. Consequently, the best hyperparameters for the model can be selected from the figure. When using the one-class SVM algorithm, the optimal threshold value for separating normal and anomalous data results in the best performance. The highest ROC AUC value indicates the optimal threshold value, and thus, the gamma and nu values corresponding to the highest ROC AUC value are the best hyperparameter values for the model.

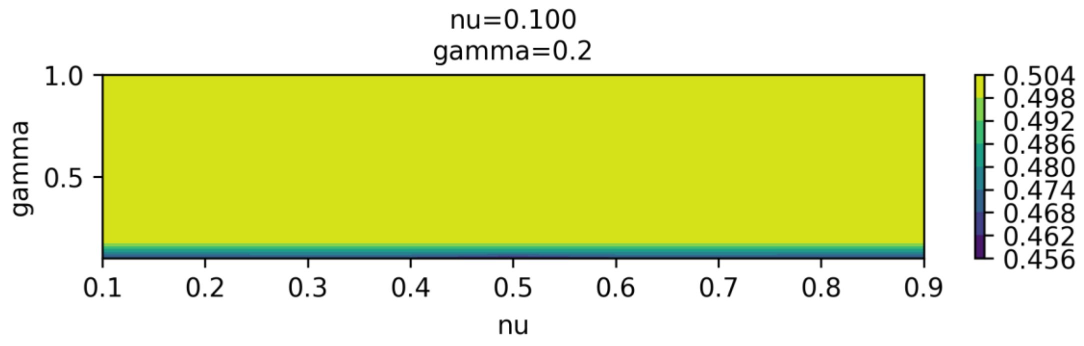

Figure 20: ROC AUC score for hyperparameter gamma and nu

4.5.5 Hyperparameter of nearest neighbors:

To determine the optimal number of nearest neighbors (N) for the model, Figure 21 displays the ROC AUC values for each N neighbor. From this figure, the best N value can be identified as the one that produces the highest ROC AUC value. The highest ROC AUC value indicates that the N nearest neighbor model provides the best performance for this particular number of nearest neighbors.

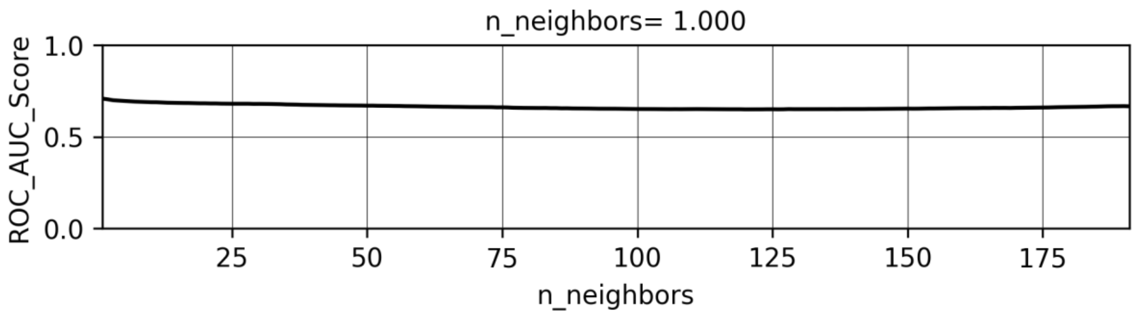

Figure 21: AUC values for different N neighbor's value

4.5.6 Hyperparameter of Isolation Forest:

Figure 22 illustrates the ROC AUC values for the hyperparameters of the Isolation Forest. Hyperparameters that result in high ROC AUC values are considered more suitable for the model. The figure displays the optimal number of estimators and max samples for which the ROC AUC value is the highest. Consequently, it can be concluded that using these values for the number of estimators and max samples will yield the best model performance.

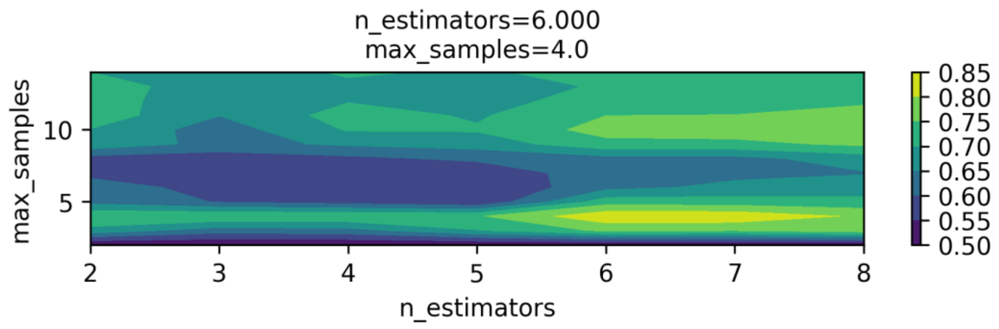

Figure 22: AUC scores for different hyperparameters combinations for isolation forest

4.5.7 Hyperparameter of autoencoder:

Figure 23 displays the change in ROC-AUC values for each n-components value in an autoencoder model. Since it is evident from the figure that different n-component values yield different ROC-AUC values for the autoencoder model, selecting the appropriate n-component value is crucial.

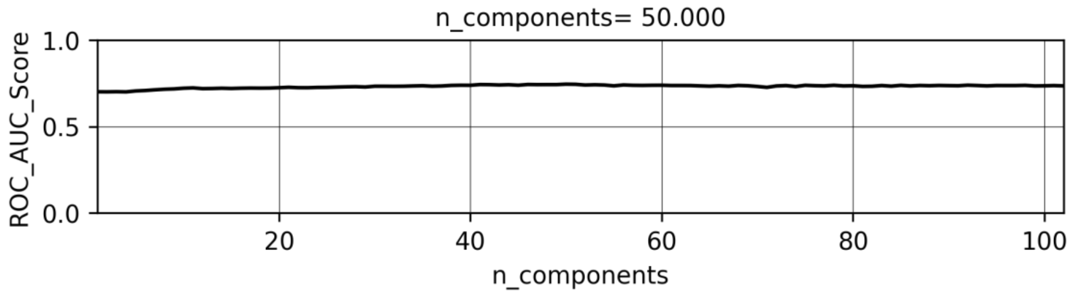

Figure 23: AUC scores for different hyperparameters combinations for autoencoder

4.5.8 Hyperparameter of nearest neighbor with PCA:

To improve the performance of the nearest neighbor, the dimension of the dataset was reduced using PCA. Figure 24 illustrates the effect of the number of PCA components and the number of nearest neighbors on the ROC-AUC values of the hybrid model.

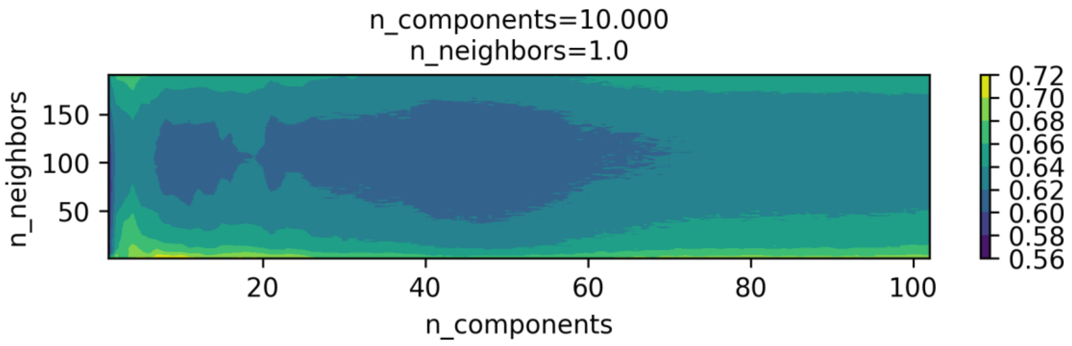

Figure 24: AUC scores for different hyperparameters combinations for the nearest neighbor with PCA

4.5.9 Hyperparameter of local outlier factor:

Figure 25 illustrates the effect of hyperparameters of the local outlier factor on ROC-AUC values. The figure indicates that contamination does not have a significant effect on the ROC-AUC values. Moreover, the optimal value of n-neighbors for achieving the highest ROC-AUC value can be determined from the figure.

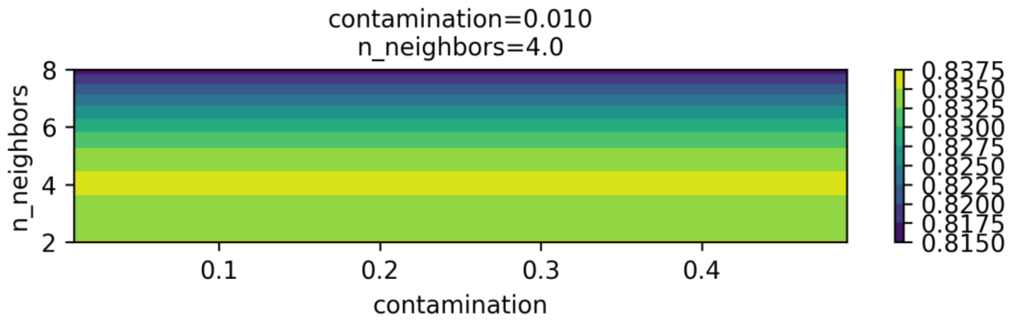

Figure 25: AUC scores for different hyperparameters combinations for local outlier factor

4.5.10 Hyperparameter of local outlier factor with PCA:

To improve the performance of the local outlier factor, the dimension of the dataset was reduced using PCA. As Figure 25 demonstrates that contamination has no impact on the ROC-AUC, I set the contamination level to 0.1 and searched for the optimum values of n-neighbors of the local outlier factor and n-component of the PCA. Figure 26 shows the effect of hyperparameters on ROC-AUC. From Figure 26 the optimum hyperparameter can be found. In this case, the optimal configuration is 24 PCA components and 3 nearest neighbors.

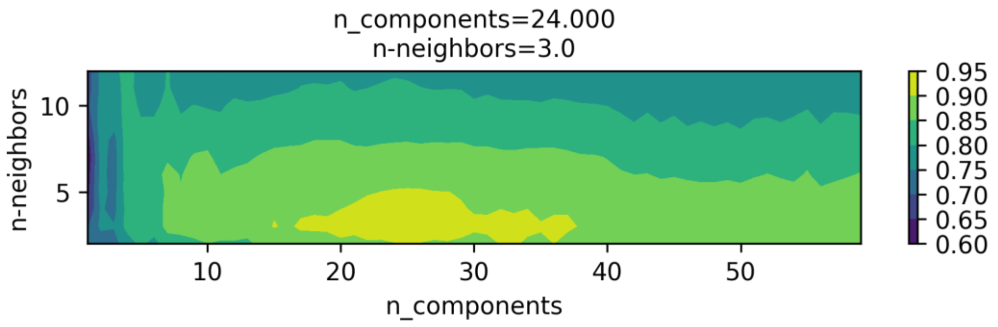

Figure 26: AUC scores for different hyperparameters combinations for local outlier factor with PCA

5 Experiments

This chapter presents the results obtained by applying different machine-learning algorithms to various experimental datasets. For each algorithm, the hyperparameters were selected according to the methods described in the methodology section. The hyperparameters, as well as the training data used for each algorithm, are listed in this chapter.

5.1 Case1:

In the hyperparameter tuning process, machine numbers 1, 2, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 24, and 25 with the tag ‘j’ were used for training, while machine numbers 6, 7, 8, 9, 10, 23, 26, 27, 28, 29, and 30 with the tag ‘j2’ were used for validation. In the validation phase, hyperparameters were selected for each machine learning algorithm using the same process described in the hyperparameter tuning section. In the testing phase, the same data were used for training, and machine numbers 6, 7, 8, 9, 10, 23, 26, 27, 28, 29, and 30 with the tag ‘j’ were used for testing.

A summary of this section is presented in the following chart for better understanding.

Performance analysis			
Sl. No	Machine learning	Hyperparameter	ROC-AUC
1	One class SVM	nu=0.1, gamma=0.2	0.500
2	Isolation Forest	n-estimators=6 max-samples=4	0 .722
3	Nearest neighbors	n-neighbors=1	0.767
4	Autoencoder	n-component=50	0 .788
5	PCA+ K-nearest neighbor	n-component=10, n-neighbors=1	0.827
6	Local Outlier Factor	n-neighbors = 4, contamination= 0.01	0.847
7	PCA+ Local Outlier Factor	n-component=24, n-neighbors = 3,	0.816

5.1.1 Performance of machine learning algorithms

The use of the ROC curve for analyzing the model performance has already been discussed in the previous section named ROC curve. In this section, the ROC curve has been used to show the model performance in sound anomaly detection. I divided the models into two categories. The first category includes machine learning algorithms that give ROC-AUC values less than 0.80. The second category contains machine learning algorithms that have ROC-AUC values of more than 0.80. Figure 27 shows the corresponding ROC-AUC of the machine-learning models in the first category. Figure 28 shows the ROC-AUC of the machine learning models in the second category. From Figure 27 One-class SVM gives a ROC-AUC of 0.500, which means it cannot distinguish anomalies at all. However, Isolation Forest, K-nearest neighbors, and autoencoder give ROC-AUC values between 0.72 and 0.79, which is satisfying. On the other hand, Figure 28 shows the ROC-AUC values of machine learning algorithms that have ROC-AUC values of more than 0.80. From Figure 28, dimensionality reduction using PCA improves the performance of K-nearest neighbors. In addition, the same figure shows that Local Outlier Factor (LOF) gives the best performance in detecting sound anomalies. Another notable fact about LOF, in this case, is that dimensionality reduction using PCA has a negative impact on LOF.

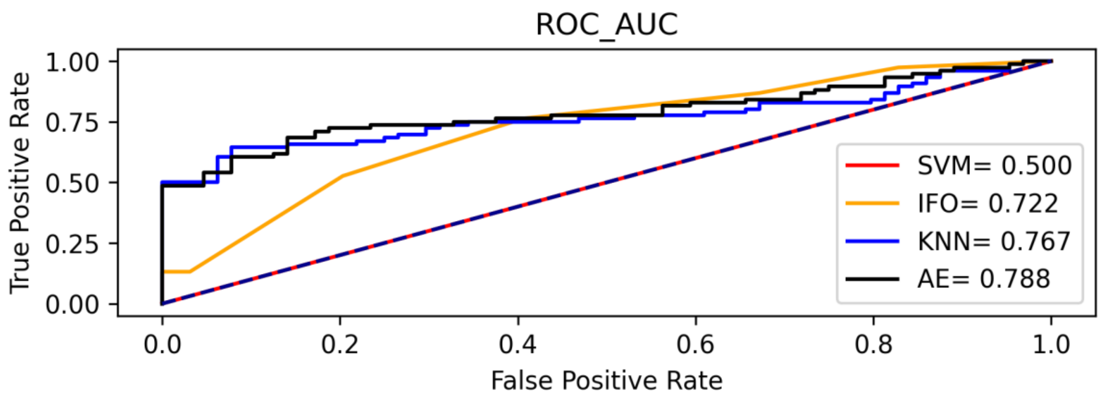

Figure 27: ROC_AUC

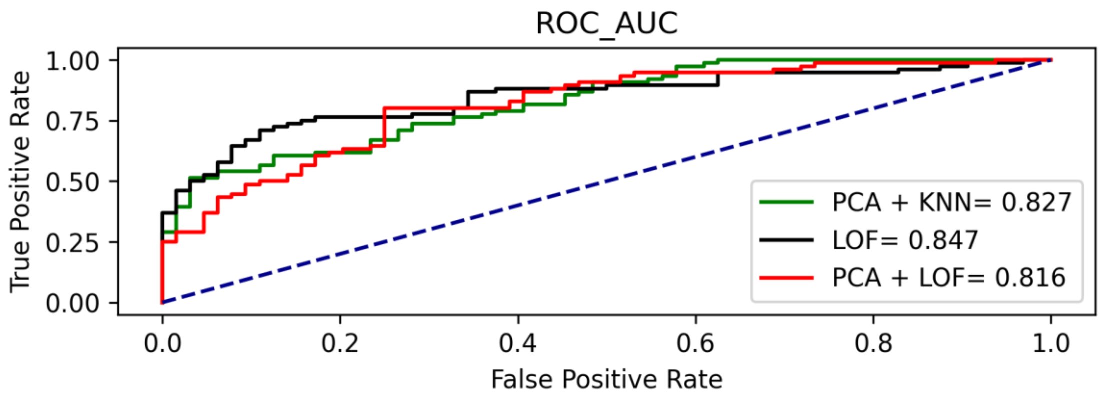

Figure 28: ROC_AUC_2

5.1.2 Anomaly score distribution for LOF

We have used the anomaly score to identify faulty machines, and the distribution of anomaly scores for each faulty machine is illustrated below. The black bins represent the anomaly scores for normal machines, while the scores for faulty machines are denoted by their corresponding colors mentioned in the legend.

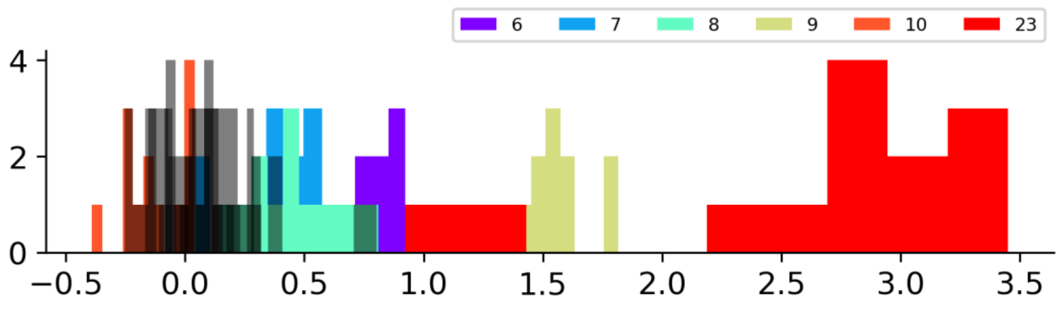

Figure 29: Score distribution for Local Outlier Factor

The local outlier factor gives noticeably higher anomaly scores for machine numbers 23, 9, and 6. However, the anomaly scores for machine numbers 7, 8, and 10 are not significantly higher than those of normal machines. Specifically, machine number 10 gives an almost identical anomaly score to that of normal machines.

6 Evaluation

The results of the experiments reveal that the one-class SVM fails to detect anomalous data in this particular case. This could be attributed to the unsuitability of the data distribution for distinguishing anomalies using a hyperplane, even with the utilization of kernel tricks. The one-class SVM considers the origin as anomalous data and draws a hyperplane between the normal data and the origin. However, in this experiment, there is no guarantee that the anomalous data are located close to the origin. Consequently, the one-class SVM completely fails to detect anomalous data in this case.

Furthermore, although the performance of the isolation forest is better than that of the one-class SVM, it is still not adequate for the practical utilization of the model. The isolation forest comprises several isolation trees that consider features randomly. Since the model was trained only with normal data in this experiment, the isolation trees were unable to identify the most relevant features from the normal data for anomaly detection. Consequently, when the model is tested with anomalous data, the result from one isolation tree may be mitigated by the result from another isolation tree. Therefore, if our anomalous data exhibit variation in fewer features, the results from fewer isolation trees may be mitigated as well. Thus, the isolation forest is suitable for cases where the training data contains anomalous data.

However, this experiment also demonstrates that the autoencoder performs well in anomaly detection in this particular case. Additionally, nearest neighbor-based classification methods such as k-nearest neighbors (KNN) and LOF yield satisfactory results. Another noteworthy finding is that dimensionality reduction using PCA enhances the performance of the k-nearest neighbor algorithm.

From this experiment, it is observed that LOF achieves the best classification results compared to other machine learning algorithms examined. Another significant finding is that while dimensionality reduction enhances the performance of LOF in the validation phase, it negatively affects its performance in the testing phase. This discrepancy can be attributed to overfitting, which can be mitigated by adjusting the n-components of PCA.

Nearest neighbor-based one-class classifiers such as KNN, LOF, and PCA-based autoencoder prioritize every feature during classification. Consequently, even a slight variation in any single feature can lead to a significant difference in the anomaly scores. In this experiment, since the sound from faulty machines can cause changes in any feature (frequency), the important features for anomaly detection remain uncertain. Thus, classification algorithms that assign importance to every feature, like nearest neighbor-based algorithms and PCA-based autoencoders, yield better results.

Analyzing the anomaly scores for LOF, it is evident that machine numbers 6, 9, and 23 exhibit significantly higher scores. Machine number 6 experiences a fan problem, machine number 9 has a bearing problem, and machine number 23 suffers from an issue denoted by fault no 5. Machine numbers 7 and 8, both having the same bearing problem, produce almost identical anomaly scores, slightly higher than those of the normal machines. Conversely, machine number 10, which encounters a spider-bearing problem, generates anomaly scores similar to those of normal machines.

7 Conclusion and future work

Overall, this thesis provides a comparative analysis of several one-class classifiers for detecting anomalous sound, aiming to make a reliable contribution to the research community in terms of their performance in acoustic anomaly detection. Additionally, the impact of hyperparameters on machine learning algorithm performance is also examined.

The experiment demonstrates that neighbor-based classifiers, such as k-nearest neighbor and LOF, deliver satisfactory results in acoustic anomaly detection. It also reveals that dimensionality reduction has a positive effect on the performance of the k-nearest neighbor algorithm. However, the performance of LOF during the testing phase is lower than that during the validation phase, indicating an overfitting issue. The use of PCA for feature reduction can lead to overfitting in LOF. Autoencoder, on the other hand, yields satisfactory results. The experiment suggests that the sounds from normal machines form a distinct cluster, with any data outside that cluster being considered anomalous.

However, one-class SVM fails to classify data accurately in this experiment, indicating that anomalous data cannot be effectively classified using a hyperplane, even with the utilization of kernel tricks. The performance of the isolation forest algorithm is also unsatisfactory.

When comparing successful and unsuccessful machine learning algorithms in this experiment, the main difference lies in their feature prioritization for anomaly detection. Successful algorithms prioritize every feature, whereas unsuccessful algorithms focus on specific features. In this experiment, since the sound from faulty machines is uncertain, all features are equally important. As a result, algorithms such as KNN, LOF, and autoencoder yield better results in this case.

The observation reveals that the sound from normal machines forms a cluster, and any sound with deviations in any frequency (feature) can be considered anomalous. Therefore, a one-class anomaly detection algorithm that takes into account deviations in every dimension is well-suited for sound anomaly detection.

The anomaly score also depends on the amplitude of the anomalous sound. The higher amplitude of the anomalous sound leads to greater variation in feature values, resulting in higher Euclidean distances in KNN and LOF, as well as higher reconstruction errors in PCA-based autoencoder. This, in turn, leads to a higher anomaly score. The experiment demonstrates that the anomaly scores for machine number 23 are significantly higher than those for the other machines, as the amplitude of the anomalous sound caused by fault no 5 is comparatively higher. Conversely, machine number 10, with a lower amplitude of anomalous sound, yields almost equal anomaly scores to the normal machines.

The main focus of this thesis was to demonstrate the behavior of various one-class classifiers in detecting anomalous sound. This research can be valuable for future studies on sound-based classification models. Furthermore, while this experiment was conducted using static data, our next target is to perform it using streaming data. One finding of this work is that LOF suffers from an overfitting problem when using PCA for feature reduction. In the future, we will strive to improve the performance of LOF and address the overfitting issue.

Frequency amplitudes have been utilized as the dataset for anomaly detection in this thesis, and the results indicate that the amplitude of sounds directly affects the anomaly detection process. Therefore, expanding the experiments conducted in this thesis to evaluate the methodologies using diverse domain-specific datasets would be a promising avenue for future research.

To the best of our knowledge, no previous work has been done on comparing one-class classifiers for anomalous sound detection. Thus, this is the first study to compare such classifiers, and we believe it will provide valuable insights for future research in sound anomaly detection.

8 References

Arriola, A., Pastorini, M., Capdehourat, G., Grampín, E., & Castro, A. (2020). Large-Scale Internet User Behavior Analysis of a Nationwide K-12 Education Network Based on DNS Queries. *Computational Science and Its Applications–ICCSA 2020: 20th International Conference, Cagliari, Italy, July 1–4, 2020, Proceedings, Part I 20*, 776–791.

Bo, C., & Wu, M. (2009). Research of intrusion detection based on principal components analysis. *2009 Second International Conference on Information and Computing Science*.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: identifying density-based local outliers. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 93–104.

Cariño-Corrales, J. A., Saucedo-Dorantes, J. J., Zurita-Millán, D., Delgado-Prieto, M., Ortega-Redondo, J. A., Alfredo Osornio-Rios, R., & de Jesus Romero-Troncoso, R. (2016). Vibration-Based Adaptive Novelty Detection Method for Monitoring Faults in a Kinematic Chain. *Shock and Vibration*, *2016*, 2417856. https://doi.org/10.1155/2016/2417856

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, *41*(3), 1–58.

Chou, P.-H., Wu, M.-J., & Chen, K.-K. (2010). Integrating support vector machine and genetic algorithm to implement dynamic wafer quality prediction system. *Expert Systems with Applications*, *37*(6), 4413–4424. https://doi.org/https://doi.org/10.1016/j.eswa.2009.11.087

Data, M. I. T. C., Salgado, C. M., Azevedo, C., Proença, H., & Vieira, S. M. (2016). Noise versus outliers. *Secondary Analysis of Electronic Health Records*, 163–183.

Debnath, L. (2012). A short biography of Joseph Fourier and historical development of Fourier series and Fourier transforms. *International Journal of Mathematical Education in Science and Technology*, *43*(5), 589–612.

Diez-Olivan, A., del Ser, J., Galar, D., & Sierra, B. (2019). Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0. *Information Fusion*, *50*, 92–111. https://doi.org/https://doi.org/10.1016/j.inffus.2018.10.005

Duman, T. B., Bayram, B., & İnce, G. (2020). Acoustic Anomaly Detection Using Convolutional Autoencoders in Industrial Processes. In F. Martínez Álvarez, A. Troncoso Lora, J. A. Sáez Muñoz, H. Quintián, & E. Corchado (Eds.), *14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019)* (pp. 432–442). Springer International Publishing.

Garvin, D. (1987). Competing on the eight dimensions of quality. *Harv. Bus. Rev.*, 101–109.

Ghafoori, Z., Rajasegarar, S., Erfani, S. M., Karunasekera, S., & Leckie, C. A. (2016). Unsupervised parameter estimation for one-class support vector machines.

Advances in Knowledge Discovery and Data Mining: 20th Pacific-Asia Conference, PAKDD 2016, Auckland, New Zealand, April 19-22, 2016, Proceedings, Part II 20, 183–195.

Gorgoglione, A., Bombardelli, F. A., Pitton, B. J. L., Oki, L. R., Haver, D. L., & Young, T. M. (2018). Role of sediments in insecticide runoff from urban surfaces: Analysis and modeling. *International Journal of Environmental Research and Public Health*, *15*(7), 1464.

Gorgoglione, A., Castro, A., Iacobellis, V., & Gioia, A. (2021). A comparison of linear and non-linear machine learning techniques (PCA and SOM) for characterizing urban nutrient runoff. *Sustainability*, *13*(4), 2054.

Günnemann, N., Günnemann, S., & Faloutsos, C. (2014). Robust multivariate autoregression for anomaly detection in dynamic product ratings. *Proceedings of the 23rd International Conference on World Wide Web*, 361–372.

Hartman, D. A. (2012). Real-time detection of processing flaws during inertia friction welding of critical components. *Turbo Expo: Power for Land, Sea, and Air*, *44717*, 1–10.

Hawkins, D. M. (1980). *Identification of outliers* (Vol. 11). Springer.

Jia, X., Zhao, M., Di, Y., Yang, Q., & Lee, J. (2017). Assessment of data suitability for machine prognosis using maximum mean discrepancy. *IEEE Transactions on Industrial Electronics*, *65*(7), 5872–5881.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.

Kao, M.-Y. (2008). *Encyclopedia of algorithms*. Springer Science & Business Media.

Lei, Y., He, Z., Zi, Y., & Chen, X. (2008). New clustering algorithm-based fault diagnosis using compensation distance evaluation technique. *Mechanical Systems and Signal Processing*, *22*(2), 419–435.

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. *2008 Eighth Ieee International Conference on Data Mining*, 413–422.

Liu, R., Yang, B., Zio, E., & Chen, X. (2018). Artificial intelligence for fault diagnosis of rotating machinery: A review. *Mechanical Systems and Signal Processing*, *108*, 33–47. https://doi.org/https://doi.org/10.1016/j.ymssp.2018.02.016

Marchi, E., Vesperini, F., Eyben, F., Squartini, S., & Schuller, B. (2015). A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural networks. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1996–2000. https://doi.org/10.1109/ICASSP.2015.7178320

McFadden, P. D., & Smith, J. D. (1984). Model for the vibration produced by a single point defect in a rolling element bearing. *Journal of Sound and Vibration*, *96*(1), 69–82. https://doi.org/https://doi.org/10.1016/0022-460X(84)90595-9

Minhua, W., Kumatani, K., Sundaram, S., Ström, N., & Hoffmeister, B. (2019). Frequency domain multi-channel acoustic modeling for distant speech recognition. *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6640–6644.

Oh, D. Y., & Yun, I. D. (2018). Residual error based anomaly detection using auto-encoder in SMD machine sound. *Sensors*, *18*(5), 1308.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., & Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825–2830.

Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia*, *4*(2), 1883.

Popescu, M., & Mahnot, A. (2009). Acoustic fall detection using one-class classifiers. *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 3505–3508. https://doi.org/10.1109/IEMBS.2009.5334521

Rabaoui, A., Kadri, H., Lachiri, Z., & Ellouze, N. (2008). One-class SVMs challenges in audio detection and classification applications. *EURASIP Journal on Advances in Signal Processing*, *2008*, 1–14.

Rahi, P. K., & Mehra, R. (2014). Analysis of power spectrum estimation using welch method for various window techniques. *International Journal of Emerging Technologies and Engineering*, *2*(6), 106–109.

Rakotomamonjy, A. (2004). Optimizing Area Under Roc Curve with SVMs. *ROCAI*, 71–80.

Salamon, J., & Bello, J. P. (2017). Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification. *IEEE Signal Processing Letters*, *24*(3), 279–283. https://doi.org/10.1109/LSP.2017.2657381

Shanker, M., Hu, M. Y., & Hung, M. S. (1996). Effect of data standardization on neural network training. *Omega*, *24*(4), 385–397.

Sharma, V., Kumar, R., Cheng, W.-H., Atiquzzaman, M., Srinivasan, K., & Zomaya, A. Y. (2018). NHAD: Neuro-fuzzy based horizontal anomaly detection in online social networks. *IEEE Transactions on Knowledge and Data Engineering*, *30*(11), 2171–2184.

Smith, L. I. (2002). *A tutorial on principal components analysis*.

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

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, *115*, 213–237. https://doi.org/https://doi.org/10.1016/j.ymssp.2018.05.050

9 Appendix

9.1 Appendix A

This section provides an idea of how Python coding is done during the project. The entire coding for this project, including coding for data processing and generating results, was done using Python. The code responsible for generating all the results can be found on GitHub using this link https://github.com/Zubairul/Project.

9.2 Appendix B

In this section some sample coding have been provided for better understanding.

Code that was used for generating the plotting function has been provided below.

```
# Importing ploting function

import matplotlib.pyplot as plt

import operator

import numpy as np

import matplotlib.cm as cm

from sklearn.metrics import roc_auc_score

from os.path import sep

import sys

sys.path.append('../')

from Code1.project_configuration import get_parameter

from Code1.plotting_functions import get_figure_win

novia_red = get_parameter('color_novia_red')
```

```
import os

figure_panel_dir = 'Figure panels'

if not os.path.isdir(figure_panel_dir):

    os.makedirs(figure_panel_dir)

if not os.path.isdir('Tikz'):

    os.makedirs('Tikz')

fig_size_cm = [17, 6]                    # [width, height]

plot_rect_cm = [1.6, 1.2, 14, 2.5]  # [left, bottom, width, height]

n_subfigs = [1, 2]                       # [n_rows, n_cols]

hor_ver_sep_cm = [2, 0]

# Function for ploting and saving figuer for meshplot

def MeshPlot(Z,Title,X_normal):

    xx, yy = np.meshgrid(np.linspace(-5, 15, 200), np.linspace(-5, 15, 200))

    Z = Z.reshape(xx.shape)

    fig, ax = plt.subplots(1, 1, figsize=[6,4]);

    ax.set(title=Title,xlabel='X1', ylabel='X2')

    ax.xaxis.label.set(fontsize=12)

    ax.yaxis.label.set(fontsize=12)

    im_l=ax.contourf(xx, yy, Z, levels=9, cmap=plt.cm.PuBu)

    ax.contour(xx, yy, Z, levels=[1], colors='k')
    plt.scatter(X_normal[:, 0], X_normal[:, 1], c="red", s=40, edgecolors="k")

    plt.colorbar(im_l,)

    plt.show()

    fig.savefig(figure_panel_dir + os.sep + Title+'.png', dpi=300)  # Raster image

# Function for ploting and saving figuers in a single window

def MeshPlot2(Z_h,Z_l,Title1,Title2,X_normal,lv):

    xx, yy = np.meshgrid(np.linspace(-5, 15, 200), np.linspace(-5, 15, 200))

    Z_h = Z_h.reshape(xx.shape)

    Z_l = Z_l.reshape(xx.shape)

    # Get the figure window and the axes

    fig, axs = get_figure_win(fig_size_cm, plot_rect_cm, n_subfigs, hor_ver_sep_cm)    #
Ploting in a specific frame

    im_h=axs[0].contourf(xx, yy, Z_h, levels=9, cmap=plt.cm.PuBu)

    axs[0].contour(xx, yy, Z_h, levels=[lv], colors='k')

    axs[0].scatter(X_normal[:, 0], X_normal[:, 1], c="red", s=40, edgecolors="k",alpha=.3)

    axs[0].set(title=Title1,xlabel='X1', ylabel='X2')

    fig.colorbar(im_h, ax=axs[0])



    im_l=axs[1].contourf(xx, yy, Z_l, levels=9, cmap=plt.cm.PuBu)

    axs[1].contour(xx, yy, Z_l, levels=[lv], colors='k')

    axs[1].scatter(X_normal[:, 0], X_normal[:, 1], c="red", s=40, edgecolors="k",alpha=.5)
```

```
   axs[1].set(title=Title2,xlabel='X1', ylabel='X2')

   fig.colorbar(im_l, ax=axs[1])

   fig.savefig(figure_panel_dir + os.sep + Title1+'.png', dpi=300)  # Raster image

  def plot_roc_curve(fpr, tpr,y_test, scores):

   auc = roc_auc_score(y_test, scores)

   plt.plot(fpr, tpr, color='orange', label='ROC')

   plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')

   plt.xlabel('False Positive Rate')

   plt.ylabel('True Positive Rate')

   plt.title('ROC_AUC_Score = %1.3f' %auc,fontsize=5)

   plt.legend()

   plt.show()

def plot_Hyp_Parameter(Name,xlabel, ylabel,Set_x,Set_y,ROC,a,b=None,c=None):

   if b == None:

       n_subfigs = [1, 1]

       fig, ax = get_figure_win(fig_size_cm, plot_rect_cm, n_subfigs, hor_ver_sep_cm);

       ax[0].plot(Set_x,ROC,'-',color='black')

       ax[0].set_xlim(np.amin(Set_x),np.amax(Set_x))

       ax[0].set_ylim(np.amin(Set_y),np.amax(Set_y))

       ax[0].set_ylabel(ylabel)

       ax[0].set_xlabel(xlabel)
```

```
        ax[0].set_title(xlabel+"="+"     %1.3f"     %a+"             ""MAX_ROC"+"="+"     %1.3f"
%c,fontsize=10)

        plt.grid(color='black', linestyle='-', linewidth=0.3)

        fig.savefig(figure_panel_dir + os.sep + xlabel+Name+'.png', dpi=300)  # Raster image

        plt.show()

    else:

        n_subfigs = [1, 1]

        fig, ax = get_figure_win(fig_size_cm, plot_rect_cm, n_subfigs, hor_ver_sep_cm);

        im=ax[0].contourf(Set_x,Set_y,ROC)  #PLoted contour plot

        ax[0].set_ylabel(ylabel)

        ax[0].set_xlabel(xlabel)

        ax[0].set_title(xlabel +"="+"%1.3f" %a +"\n"+ ylabel+"="+"%1.1f"%b  , fontsize=10)

        plt.colorbar(im)

        fig.savefig(figure_panel_dir + os.sep + xlabel+Name+'.png', dpi=300)  # Raster image

        plt.show()

def plot_classification_stats(conf_matrix, machine_acc, unique_ids):

    fig, axs = get_figure_win(fig_size_cm, plot_rect_cm, n_subfigs, hor_ver_sep_cm)

    f = operator.itemgetter(2,3,4,5,6,7,8,9,20,21)

    o = operator.itemgetter(0,1,10,11,12,13,14,15,16,17,18,19)

    im = axs[0].imshow(conf_matrix, cmap=cm.jet_r)

    im.set_clim([0,1])
    axs[0].set_xlabel('Predicted class')

    axs[0].set_ylabel('True class')

    axs[0].set_xticks([0, 1], labels=['Faulty', 'OK'],fontsize=8)

    axs[0].set_yticks([0, 1], labels=['Faulty', 'OK'],fontsize=8)

    plt.colorbar(im, ax=axs[0])

    axs[1].bar(o(unique_ids), o(machine_acc), 0.5, fc=np.append(novia_red, 0.5),
ec=novia_red, lw=2)

    axs[1].bar(f(unique_ids), f(machine_acc), 0.5, fc='b', ec='b', lw=2)

    axs[1].set_xlabel('Machine ID')

    axs[1].set_ylabel('Machinewise \n Accuracy (%)')

    Model_Acc = sum(machine_acc)/machine_acc.size

    axs[1].set_title("Overall Model Accuracy = %1.3f" %Model_Acc,fontsize=5)
```