Anomaly Detection for Smart Infrastructure: An Unsupervised Approach for Time Series Comparison

Harshitha Gandra

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Thidapet Chantem, Co-chair
Farrokh Jazizadeh, Co-chair
Ruoxi Jia

Dec 07, 2021
Blacksburg, Virginia

Keywords: Anomaly Detection, Asynchronous, Unsupervised, Matrix Profile, OCSVM

Anomaly Detection for Smart Infrastructure: An Unsupervised Approach for Time Series Comparison

Harshitha Gandra

(ABSTRACT)

Time series anomaly detection can prove to be a very useful tool to inspect and maintain the health and quality of an infrastructure system. While tackling such a problem, the main concern lies in the imbalanced nature of the dataset. In order to mitigate this problem, this thesis proposes two unsupervised anomaly detection frameworks. The first one is an architecture which leverages the concept of matrix profile which essentially refers to a data structure containing the euclidean scores of the subsequences of two time series that is obtained through a similarity join.It is an architecture comprising of a data fusion technique coupled with using matrix profile analysis under the constraints of varied sampling rate for different time series. To this end, we have proposed a framework, through which a time series that is being evaluated for anomalies is quantitatively compared with a benchmark (anomaly-free) time series using the proposed asynchronous time series comparison that was inspired by matrix profile approach for anomaly detection on time series . In order to evaluate the efficacy of this framework, it was tested on a case study comprising of a Class I Rail road dataset. The data collection system integrated into this railway system collects data through different data acquisition channels which represent different transducers. This framework was applied to all the channels and the best performing channels were identified. The average Recall and Precision achieved on the single channel evaluation through this framework was 92% and 54% respectively with an error threshold of 0.04 miles or 211 feet. A limitation that was noticed in this framework was that there were some false positive

predictions. In order to overcome this problem, a second framework has been proposed which incorporates the idea of extracting signature patterns in a time series also known as motifs which can be leveraged to identify anomalous patterns. This second framework proposed is a motif based framework which operates under the same constraints of a varied sampling rate. Here, a feature extraction method and a clustering method was used in the training process of a One Class Support Vector Machine (OCSVM) coupled with a Kernel Density Estimation (KDE) technique. The average Recall and Precision achieved on the same case study through this framework was 53% and 44%. In comparison to the first, the second framework does not perform as well. There will be future efforts focused on improving this classification-based anomaly detection method.

Anomaly Detection for Smart Infrastructure: An Unsupervised Approach for Time Series Comparison

Harshitha Gandra

(GENERAL AUDIENCE ABSTRACT)

Time series anomaly detection refers to the identification of any outliers or deviations present in a time series data. This technique could prove to be useful to mitigate any unplanned events by facilitating early maintenance. The first method proposed involves comparing an anomaly-free dataset with the time series of interest. The difference between these two time series are noted and the point with the highest difference will be considered to be an anomaly. The performance of this model was evaluated on a Rail road dataset and the cumuluative average Recall (how useful the predictions are) and average Precison (how accurate the predictions are) are 92% and 54% respectively. The second method proposed involves extracting all segments in the anomaly-free dataset and grouping them according to their similarity. Here, a OCSVM is used to train these individual groups. OCSVM is a machine learning algorithm which learns to classify a data as either anomalous or normal. It is then coupled with the KDE which creates a distribution across all the anomalies and identifies the anomaly as one with a high distribution of predictions.The performance of this model was evaluated on a Rail road dataset and the cumulative average Recall and cumulative average Precision 53% and 44% respectively with an acceptable error range of 0.04 miles or 211 feet.

Dedication

This work is dedicated to my family and friends who have supported me uncondionally along the way.

Acknowledgments

I would like to thank my advisor Dr.Farrokh Jazizadeh for his guidance and support. Thank you providing me with the opportunity to work with you in this interesting project. I would also like to thank Yueyan Gu who served as a constant mentor to me throughout and the members of the Inform Lab for giving me helpful feedback along the way. Lastly I would like to thank the Federal Railroad Administration's Research, Development and Technology for their financial support in funding this project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Infrastructure systems are the foundation stones of the national economy. However, these systems go through extensive strenuous stress due to environmental conditions and constant usage. Such conditions emphasize the importance of continuous monitoring and maintenance to ensure performance, resilience, and safety of infrastructure systems. This thesis has been motivated and investigated by a railroad infrastructure monitoring problem while its findings could be applied to other infrastructure systems with similar settings. Railroad infrastructure are critical to the US economy. In 2017, the Class I railroads generated $219 billion economic output and $26 billion tax revenues [1]. Moreover, monitoring and maintenance of the rail road track infrastructure are critical in preventing track defects that could lead to accidents and injuries.

It is crucial to maintain the infrastructure systems to improve safety and efficiency and to reduce associated costs. To achieve these objectives, monitoring systems for identification and quantification of the maintenance problems are extensively used for different infrastructure systems. These monitoring systems rely on a variety of sensing and measurement systems that enable the operators to collect and analyze a wide range of data. Therefore, data-driven infrastructure monitoring systems have gained popularity and are prevalent. However, although autonomous data acquisition and monitoring systems have been gaining popularity, the conventional methods of the data analytics still call for human-in-the-loop (HITL) interventions through visual or manual inspections. This may not be a convenient

method as labor is required and can potentially lead to human errors. In the case of a railroad track infrastructure, to detect anomalies, industries may make use of advanced sensing systems (heavily instrumented geometry cars that are equipped with a wide array of sensors to measure a variety of parameters) but human efforts are still required to monitor the data and assess the track based on prior knowledge. Hence, to overcome the need for experienced operators and extensive HITL interventions, artificial intelligence (AI) technologies can be leveraged to monitor and detect any changes in infrastructures through continuous data acquired through sensors. This enables automated inspections and allows to support the process of human decision making. Thus, the main focus of this thesis is to develop anomaly detection algorithmic frameworks for time series where unique or rare data is identified from the entire dataset and flagged as an anomaly relying on periodic and repeated inspection data.

Anomaly detection refers to the concept of finding patterns that do not conform to expected behavior [6] and has been extensively studied. In order to successfully implement an anomaly detection model, one needs to identify the problem characteristics : Nature of Data, Labels, Anomaly Type and Output (Scores and Labels) [6]. The following factors make the approach challenging [6]: (1)Scarcity of labeled data – training and validation will be a problem. (2) It might be difficult to establish a boundary encompassing normal behavior – high false positive rate.

As technology advances, infrastructure systems are becoming more complex and intricate. There are high labor costs associated with handling these systems. Moreover manual inspections and maintenances might lead to an increased human error.There have been numerous algorithms developed to address anomaly detection, but the two main approaches have been supervised and unsupervised learning. If the former approach is used, in order for classification to be optimum, the dataset needs to be balanced. This can be problematic since

anomalous events are rare leading to an imbalance. In order to avoid this, unsupervised methods have been proven to be more popular. Some common methods of anomaly detection using time series are: Density-based techniques, Cluster analysis, Bayesian Networks, Neural Networks, Support Vector Machines, Hidden Markov Models and Fuzzy Logic [2].

The advantage of the time series data is not just limited to time being used as a metric but also as a main component that helps analyze data and provides important insights. These insights help estimate future data based on the current data, which can then be used to detect anomalies by comparing them with normal data points. Time series data provides information on operating states of an infrastructure system. Time series data could be used to understand the normal patterns in the data and identify when these patterns are changing. Changes in these patterns could be reflection of abnormal behavior or defect. Comparison of the system states is an effective approach in inferring the potential anomalies. Such comparisons could be carried out by phase comparison , frequency comparison and euclidean distance. However, these comparisons usually call for time series synchronization, otherwise the shifts in patterns could lead to false positive detection.

Time series data is beneficial to anomaly detection because it can provide a lot of information regarding behaviors and patterns, where data acquisition plays a key role. In practice, however, it can be challenging to collect synchronous data from dynamic systems, because the sampling rate is not necessary to be consistent throughout. An example of this can be witnessed in the railroad industry, where there are varying speeds for each train, hence there is a difference in data collection (number of samples) during a certain period of time. In addition, real-time problems could also potentially lead to this behavior, such as a lag in data collection by sensors, environmental factors or power outage. Hence, the AI algorithms should be developed to enable anomaly detection while accounting for the asynchronous nature of the time series.

This research explores two anomaly detection methodologies.The first algorithmic frameworks enables asynchronous comparison of time series for anomaly detection. This framework draws on a time series analysis technique, called Matrix Profile Analysis[21] that enables relative comparison of subsequences in a time series for anomaly detection. While exploring this algorithm, it was noticed that the rate of false positives was higher where anomaly-free dataset yielded anomalies. Hence, to provide a more intuitive framework, the second framework was developed.The second framework relies on the identification and analysis of time series motifs for anomaly detection. Motifs are similar, repeated and expected patterns of data that could be leveraged for anomaly detection. The second framework uses time series motifs (identified through clustering) and one-class Support Vector Machine (SVM) classification as the main methods for anomaly detection.

The material presented in this thesis is as follows. In Section 2, a literature review presents the prior work that has been conducted on time series anomaly detection. Section 3 introduces the methodology, which describes the architecture of the proposed frameworks. Section 4 introduces a railroad case study along with the evaluation results from implementing the two frameworks and Section 5 and 6 provides the discussions and conclusions respectively.

Chapter 2

Review of Literature

Anomalous behaviour detection has been a very extensively studied topic in various areas such as healthcare [19], fraud detection [16], intrusion detection [10] and even ecosystem disturbances [5]. It is evident that anomaly detection is a very broad topic and can be extended to different types of data such as images, videos and data streams. This paper is focused on developing anomaly detection techniques for infrastructural health monitoring by using univariate time series.

In order to implement a smart infrastructure monitoring system, automating maintenance and enabling early defect detection is integral. Previously, conventional methods such as manual or visual inspections [11] have been employed. However, these could prove to be infeasible. Moreover, numerous factors need to be considered while making such manual analysis, such as environmental and infrastructure conditions, so as to reduce human errors. There have been direct track inspection methods such as guided wave ultrasound system [12]. This is a non intrusive approach incorporated numerical and signal analysis of the echo data.

Machine learning and deep learning methods have become a highly researched area of interest when it comes to anomaly detection. Supervised and unsupervised machine learning methods are the two main approaches. In supervised learning, there is a lot of dependence on the training data present. It is trained on labeled data to predict labels for testing data. The challenge with such an approach is that in real world anomaly detection problems, raw

data is highly imbalanced to the normal class. This could cause the biased training of the algorithm. A technique which has been proposed to overcome this is to under sample the normal data and/or to oversample the anomalous data [18][17]. An interesting deep learning approach was adopted to perform anomaly detection in time series with the use of stacked Long Short Term Memory (LSTM) systems [14]. Here, the neural network was trained on the anomaly-free data and used as a predictor over the testing series. The performance of this model was tested on four types of datasets: ECG, space shuttle, power demand, and multi-sensor engine dataset [14].

In unsupervised methods, unlike supervised methods, do not depend on any labeled data. They instead find interesting patterns in the unlabeled data. Unsupervised methods are preferred to detect anomalies because they do not rely on labelled data points. This is especially beneficial when the number of anomalies in the dataset are comparatively smaller than the other data points. Multi-Scale Convolutional Recurrent Encoder-Decoder (MSCRED) is a unsupervised approach based on Neural networks to detect and diagnose multivariate time series was developed by Zhang et al [23]. MSCRED leverages a convolutional encoder and incorporates temporal patterns with attention based ConvLSTM. This framework was tested using synthetic data and a real data from power plant. Even though this was efficient, it was seen that it was very susceptible to noise. Isolation forest is another unsupervised method that was proposed by Feremans et al [7]. This framework leveraged Jaccard similarity and derives an anomaly score using the Isolation forest classifier [7]. This algorithm was tested on various real-world univariate and multivariate time series.

The two frameworks that will be discussed in this paper are inspired by the concepts in the Matrix Profile approach [21] and Motifs [21]. Matrix Profile technique is essentially a data structure and associated algorithms aimed to detect discords (anomalies) and motifs for a given time series [21]. Motifs refer to the signature behaviours observed in a time

series and hence can be leveraged to anomalous patterns. A fast motif discovery method was introduced but this was only optimal for subseries of equal length [15]. There have been some unsupervised learning approach based on motifs that was proposed to perform anomaly detection using amplitudes and with the shape of subsequences. For example, C-means clustering [8] was evaluated using two real world dataset of Precipitation and Arrhythmia. Both of these datasets are very clean and do not have a lot of noise present so this approach could be a limitation since the railroad case study dataset used in the thesis contains noise.

In 2003, One Class Support Vector Machines(OCSVM) was introduced as a novel algorithm for outlier detection as well [13]. It was tested out on synthetic and measured data to evaluate the robustness of this approach. For the second framework, i.e. the motif based approach, clustering and OCSVM are leveraged.

In the frameworks proposed in the thesis, instead of focusing on supervised methods which require large amounts of labelled and balanced data, the main focus is to develop unsupervised algorithms to avoid issues associated with the imbalanced nature. The concept of Matrix Profile and Motifs were further explored and investigated to create two frameworks. Both of these address the problem of asynchrony that is present in the case study dataset which will be discussed later.

Chapter 3

Methodology

C.Yeh et al proposed a Matrix Profiling (MP) framework which incorporates the idea of discords (anomalies) and motifs (repeated patterns) [21]. These can be leveraged to conduct data mining which can be further used for machine learning or data analysis. According to C.Yeh et al. [21], the matrix profile of a time series refers to the z-normalized Euclidean scores between each subsequence of a selected window size and its nearest neighbor. Concurrently, subsequences with high Euclidean scores are labelled as anomalies and low Euclidean scores as motifs. This can be seen in Figure 3.2.

Figure 3.2 depicts the mechanism of the Matrix Profiling algorithm. The length of subsequences, also known as the window size (w), needs to be set. The i_{th} subsequence of index i refers to data points $(i + w)$. Assuming the length of the dataset is n, this window traverses across the entire dataset until the $(n - w)_{th}$ index is reached. As it traverses, the Euclidean scores are collected and compared for similarity. High Euclidean scores are considered to be discords or anomalies whereas smaller values to be motifs or repeated patterns.

Here, the Euclidean scores or euclidean distances refer to distance between two points on a 2 dimensional plane. The formula to calculate the Euclidean score between two time series is shown below.

If there exists two time series T_1 and T_2 of length n such that :

$T_1 = x_1, x_2, x_3, ..., x_n$

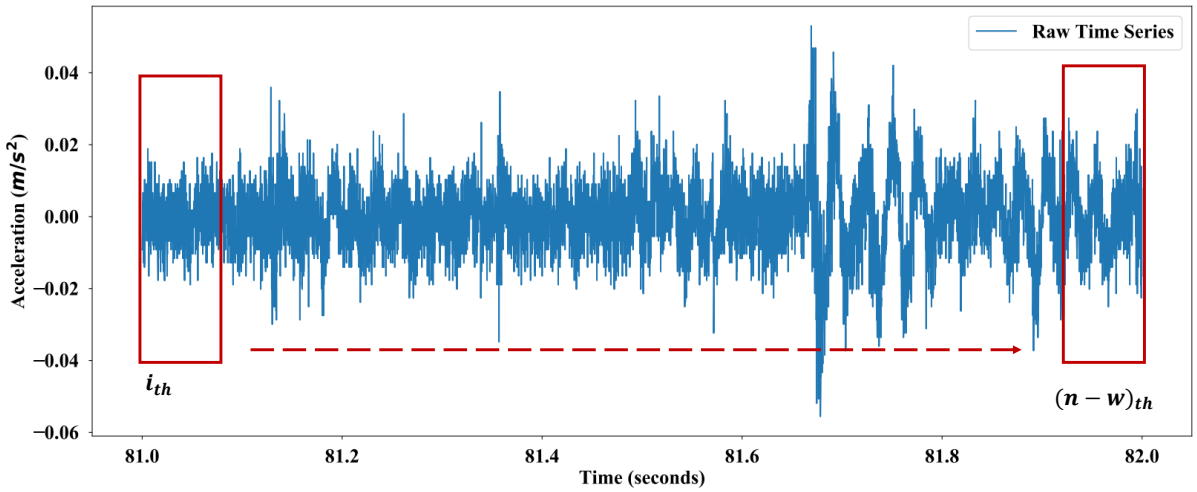


Figure 3.1: Visual representation of the sliding window approach to extract a Distance Profile graph

$T_2 = y_1, y_2, y_3, ..., +y_n$

$$Euclidean\ distance = \sqrt{\sum_{i=1]}^{n}(x_i - y_i)^2}$$

If small segments of two time series are being compared, Euclidean distance can prove to be efficient because there would not be a lot of distortion present in the data and it is a fast algorithm.

Initially, this exact algorithm was explored for anomaly detection on the time series data and two data comparison architectures were implemented as seen in Fig **??**.Given four different time series, it was hypothesized that applying the discord detection algorithm would yield the location of defect. For Architecture I, the discord algorithm was applied to a time series and the index of the subsequence with the highest Euclidean score is extracted and flagged

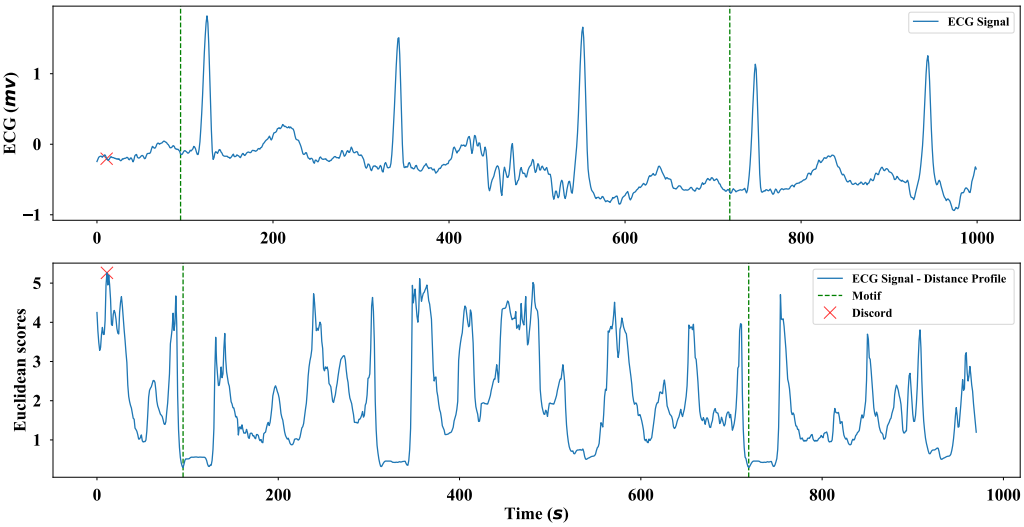


Figure 3.2: The top graph displays the raw ECG signal, and the bottom graph represents its matrix profile. High Euclidean scores – anomaly, Low Euclidean scores – motifs

as an anomaly. This process was continued for all the time series (T_A,T_B,T_C,T_D). However, this algorithm was not as efficient as expected. This could be due to the asynchronous nature of the time series or presence of natural variations in a track. For Architecture II, a different approach was used where an anomaly-free time series was identified and set as the "Benchmark", for instance T_A. The other time series are then considered to be evaluating time series (T_B,T_C,T_D). T_A is then concatenated with the remaining three time series and the discord algorithm was passed. This was hypothesized to work since the first half of the concatenated time series would serve as a comparison for the second half. However, a problem was observed with this approach that when two time series were combined as such, it led to a formation of a novel data at the concatenation point. This specific point could alter the behavior of the anomaly detection model since in many cases, it could potentially be flagged as an anomaly which is false.

In order to incorporate the concept of discord and motifs two alternative algorithms have

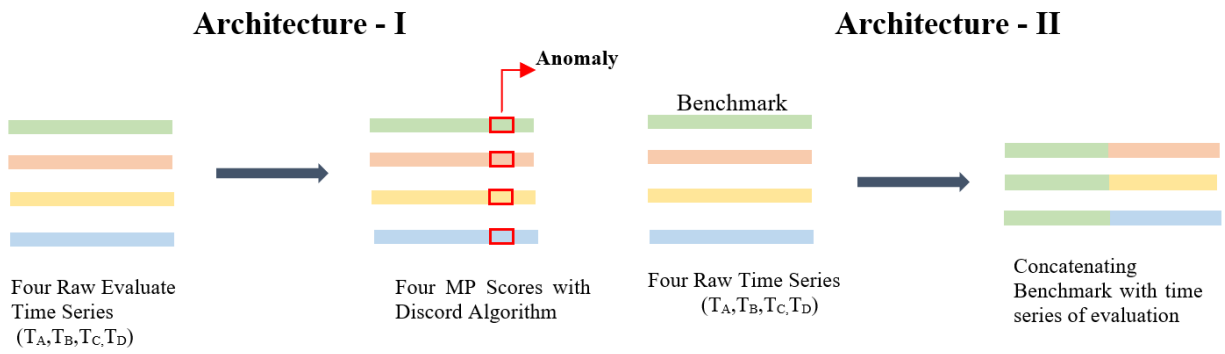


Figure 3.3: The two different Discord Architectures

been proposed:

- Matrix Profile Based Framework:

  Instead of concatenating two time series and implementing a matrix profile algorithm as seen as in Architecture II of 3.3 , an alternative algorithm was proposed which is based on this concept. Here, the subsequences of an evaluation time series of a window size (w) are compared directly with subsequences of a anomaly-free time series to identify anomalous behavior.

- Motif Based Framework:

  Instead of implementing the motif algorithm directly which is to identify the lowest euclidean scores to establish the normal behavior, the idea is to identify an anomaly-free time series and use all of its subsequences as motifs. This method ensures we have more data to leverage and still stays true to the definition of motifs.

3.1 Matrix Profile Inspired Framework

The Matrix Profiling inspired framework consists of three parts: Dataset Extraction, Matrix Profiling Algorithm, and Anomaly Selection.

A time series has underlying information which can be extracted to identify trends and patterns. The main motivation behind this proposed alternative framework is to leverage the information in a time series to detect presence of abnormalities. Here, these abnormalities typically refer to anomalies within the time series. These can be identified by examining unexpected changes in the sequence. The idea of anomaly detection in time series has garnered a lot of attention recently. A few problems which arise in regard to time series anomaly detection are [6]:

- There might not be enough labeled data to perform supervised learning.
- The normal behavior in a time series may change and conservation of the normal pattern might be difficult.
- Time series data may contain noise which might be detected as an anomaly, leading to high false positives.

Figure 3.4 is the representation of the framework and its modules. There are three main steps: Dataset Extraction, Matrix Profiling Algorithm, and Anomaly Selection. A detailed explanation of the three steps is provided below.

Once the dataset is extracted and segregated into the Benchmark (T_{BS}) and Evaluate (T_{ES}) time series, the next step is to implement the main matrix profiling algorithm. The subsequences of the time series are compared, and their Euclidean scores are stored. In this implementation, a Matrix Profile is defined as a vector of the minimum normalized Euclidean

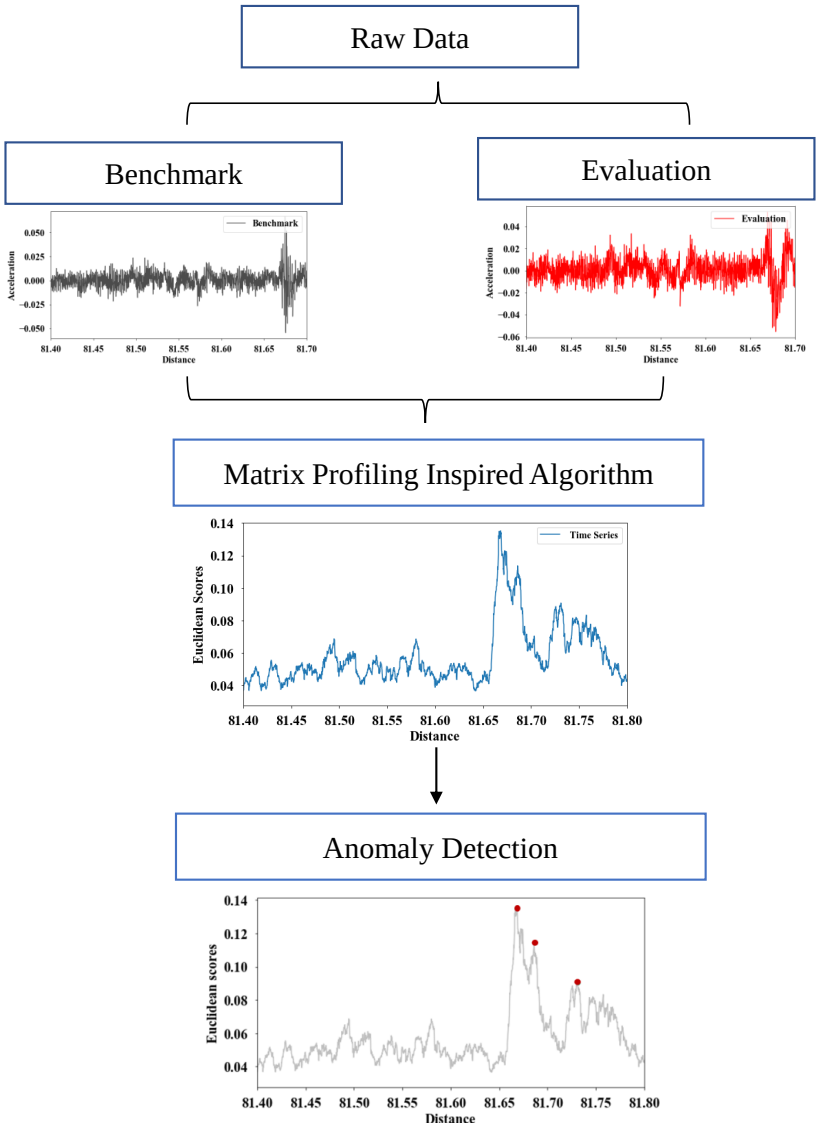


Figure 3.4: Matrix Profiling Inspired Anomaly Detection Framework

distances between each subsequence in the T_{ES} and the T_{BS} [22]. For instance, as seen in Figure 3.5, the first factor that needs to be selected is the window size (w). This is an integral part because setting a too low or high value can cause the anomaly to be overlooked. The i_{th} subsequence of T_{ES} is compared with all the subsequences of T_{BS} to identify its closest match (minimum Euclidean distance/score). This process is repeated until the $(n - w)_{th}$ index of the T_{ES} is reached. In the MP graph, the locations with high Euclidean scores can

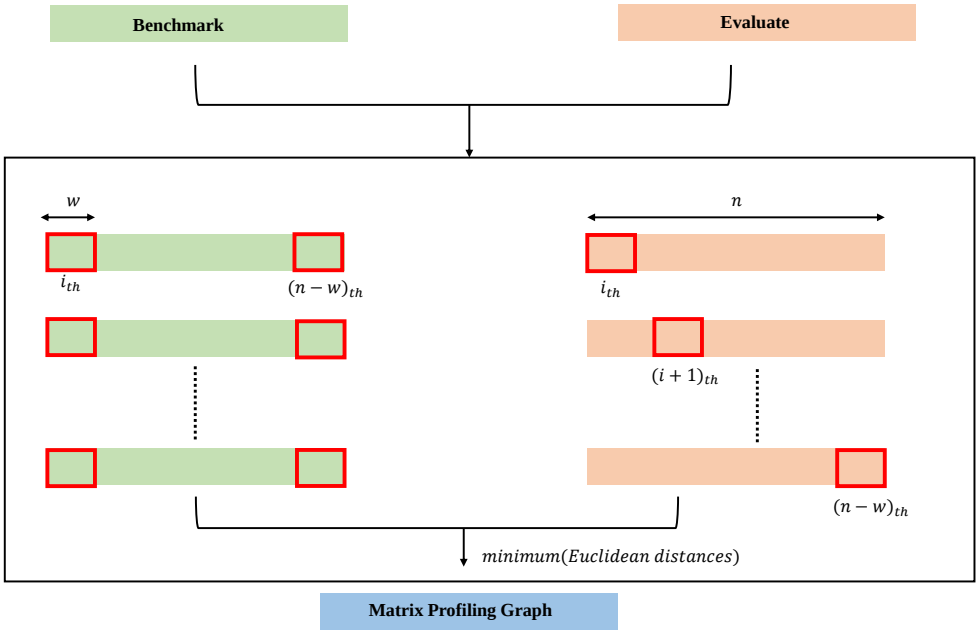


Figure 3.5: Description of Matrix Profiling Framework

therefore be extracted as anomalies.

3.2 Motif Based Framework

The concept of motifs was introduced in 2002 and it has been leveraged in different domains since then[15].A time series motif essentially refers to repeated conserved patterns present in a time series data [21] and this information can be leveraged to identify areas of concern or value. The similarities observed in these motifs could be of interest and can be used to understand the general behavior of the time series. These motifs extracted can be further used in various machine learning tasks such as classification and clustering [21]. The motif patterns are used to establish the normal behavior of the time series signal. These are then compared to the subsequences extracted from a time series where there is expected to be

abnormal behavior. The framework of the motif based analysis is shown in **??**.

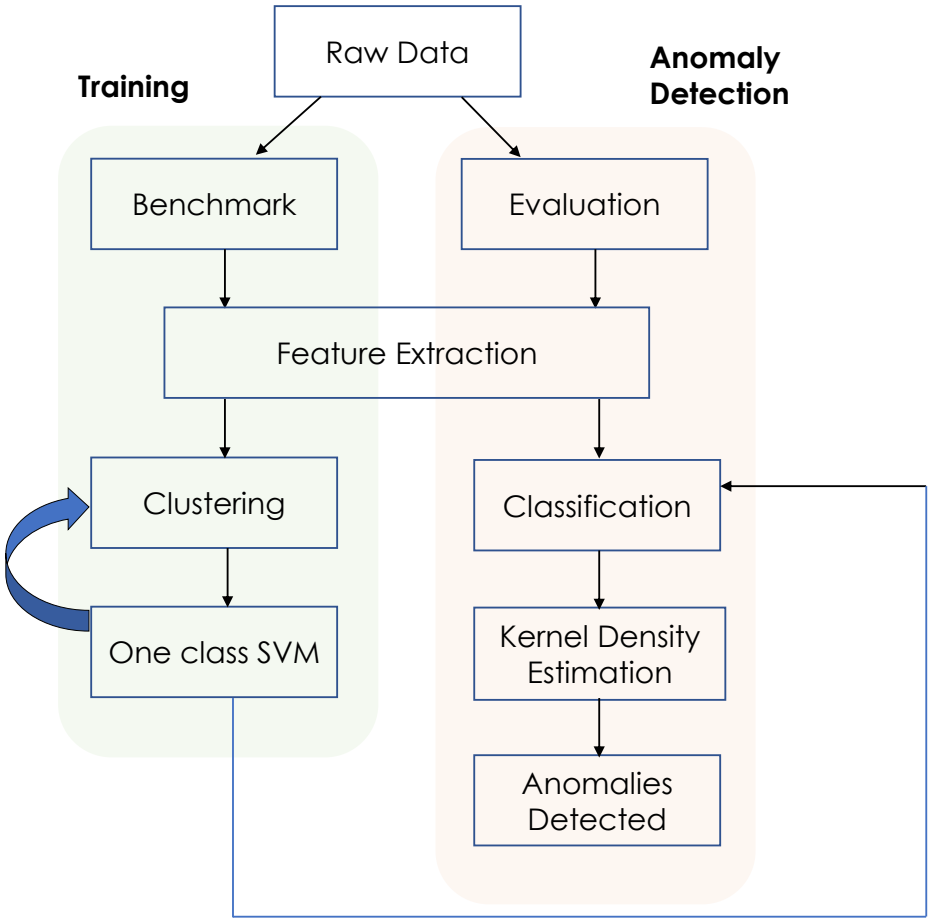


Figure 3.6: Motif Based Framework

Training

Here, the benchmark data serves as the training dataset which undergoes a feature extraction algorithm which comprises of extracting subsequences. In the training stage, a K means clustering and One class SVM models are leveraged. The following sections provide more detail.

1. Feature Extraction:

   Keeping in mind the definition of motifs, these are extracted from the benchmark dataset with a window size (w). Window size plays an important role since it determines the length of the subsequences. It should be set such that the time series pattern is conserved.Considering a time series of length of n as seen in Figure 3.1, the i_{th} subsequence comprises of the (i+w) subsequence. This process continues until the (n-w) subsequence is reached. In a similar way, the subsequences of the evaluating time series can be extracted. The subsequences extracted from the benchmark data can then be clusered to establish the motifs. This refers to training dataset while the subsequences extracted from the evaluation dataset refer to the testing dataset.

2. Clustering:

   Once the feature extraction method is completed, the next step is to establish the motifs. This can be done by using an unsupervised learning method such as K-Means Clustering. The concept of clustering refers to the grouping of data which exhibit similar behavior into clusters. It relies on the assumption: "Normal data instances lie close to their closest cluster centroid, while anomalies are far away from their closest cluster centroid [6]." The advantage of using K-means clustering is that it allows the algorithm to be implemented in an unsupervised manner and works well with data which cannot be separated using a plane or straight line. Here, time series data which is clustered together means that they have the same centroid. These were clustered according to their Dynamic Time Warping (DTW) distance. DTW is an algorithm made to measure similarity between two sequences. Even though this is a computationally expensive algorithm, it allows a many-to-one rather than a one-to-one comparison leading to more accurate comparison and predictions. DTW is beneficial for the time series we have because it handles the asynchronous trait that could be

present in a time series. In order to optimize the clustering algorithm, the elbow method was chosen to determine the best number (k) of clusters. Therefore, if five were to be considered the ideal number of clusters through elbow method, there will essentially be five clusters leading to five motifs. Once the clustering algorithm is executed, the cluster they belong to can be assigned as their respective cluster labels.

3. One Class Support Vector Machine:

   The motifs extracted in the previous step refer to normal patterns so this information can be leveraged to identify abnormal data.

   Once the training data is extracted and labelled using the K-Means clustering method, there are now k clusters with a OCSVM model for each. For instance, if there are five clusters, there will be five OCSVM models for each. These models use a hypersphere instead of hyperplanes to differentiate and encompass data. This model has two parameters (Nu, Gamma). Nu refers to the allowed training error and for this algorithm and Gamma refers to the radius of the kernel. Nu and Gamma are two hyperparameters which need to be tuned to avoid underfitting and overfitting. Their tuning methods are explained in the next section.

   For this implementation, a One class Support Vector Machine (OCSVM) has been applied due to its ability to perform unsupervised learning. This is a common algorithm which can be used for a one class classification problem where the region encompassing the training data can be learned [6]. OC- SVM consists of a kernel (linear, radius basis function(rbf) and polynomial) which can be invoked to learn and set complex boundaries. It has a L2 Regularization feature thereby preventing the data from overfitting and it can handle both classification and regression based problems[9]. The difference between SVM and OC-SVM is that the latter identifies a decision boundary which encompasses the instances of the class, and it allows only a few instances to be outliers.

If an instance from the testing data falls outside this boundary, it is labelled as an anomaly or an outlier. OC-SVM distinguishes instances from the origin given a feature space F and aims to maximize the distance from the hyperplane to the origin [9]. This creates a binary function which predicts a value of " +1 " if the testing instance is normal and " -1 " if the instance is an outlier. For instance, if there exists a data set with a feature space H, OCSVM works by traversing through the entire feature space to locate a subset Q with the use of a kernel to attain a function which satisfies the following function f(x) [4]:

$$f(x) = \begin{cases} +1 & x \in Q \\ -1 & x \notin Q \end{cases}$$

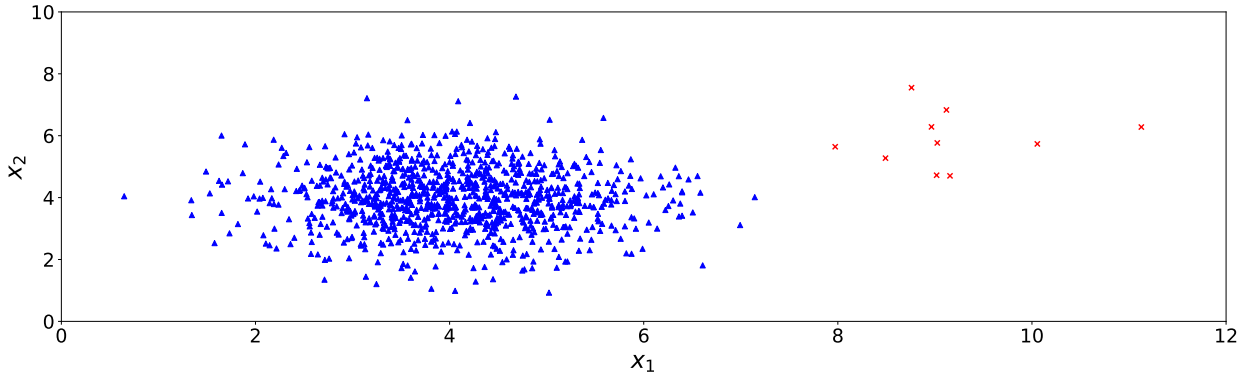


Figure 3.7: Visual representation of a One Class Support Vector Machine where the 'x' marks the outliers

Anomaly Detection

Once the training phase is completed, it is time to perform anomaly detection on the evaluation time series. Similarly, this also undergoes a subsequence extraction method to obtain its features. The following material outlines the final steps of this anomaly detection process.

1. Classification:

The model used from the trained clustering algorithm is applied to the testing dataset to provide cluster labels. Now that the testing dataset is segregated into their individual clusters, they are passed through their respective OCSVM model to identify the presence of anomalies. The data identified as an outlier using the SVM models are classified as anomalies. If there are multiple anomalies selected, a Kernel Density Estimation (KDE) explained in the following section could be used to eliminate the false positives.

2. Kernel Density Estimation:

Once the OCSVM classification is implemented, the next step is to identify the anomalies. To understand the distribution of the predictions, an important statistical tool leveraged is the Kernel Density Estimation (KDE). KDE works by plotting the data points and fits these with a curve to observe the distribution. The key factor which determines how smooth the distribution curve is the bandwidth value. Bandwidth plays an important role in prediction because a large value can over smooth the distribution causing a lot of real anomaly predictions to be overlooked whereas a small value can under smooth the distribution creating a lot of false predictions. Hence, tuning this parameter is vital for the implementation to be successful. The importance of bandwidth can be seen in the graph below. Taking into consideration the Euclidean scores, locations with high probabilities can be considered as abnormalities or anomalies.

Chapter 4

Results

4.1 Case Study: Rail track Anomaly Detection

Now that the framework is established, we evaluted its efficacy by implementing it on a real-world time series dataset from a Class I rail track. Rail tracks tend to go through stress from the rail car and the environment. Over time, this could lead to the formation of rail track defects, which need to be detected through frequent inspections and maintenance interventions. As noted, the objective of proposing the frameworks in this thesis is to facilitate data-driven inspections. It can be seen from both of the proposed frameworks that there is a need to identify a benchmark (anomaly-free) time series which serves as the base for comparison. The remaining time series serve as the evaluating or testing time series. These could comprise of anomalies or can be a clean time series. The following sections provides a brief description of the dataset, its segmentation into benchmark and evaluating time series along with a focus on how both of the proposed frameworks can be extended to implement anomaly detection for the railroad track.

4.2 Dataset Description and Segmentation

This dataset is collected through five regular inspections using a geometry car over a distance of 10 miles. It consists of data from various data acquisition channels, data from five

inspections (five different days) was provided for analysis and experimentation including two types: Acceleration (Horizontal and Vertical) and Geometry data. As established before, the data that is being analyzed is a time series sequence. The time series data was collected using accelerometer sensors and track geometry cars. Apart from this, a track maintenance log (ground truth) was also provided. This provided valuable information regarding the date and location of a reported maintenance. Another key observation was that there were more anomaly-free data compared to anomalous data which was expected. The goal here is to establish a benchmark dataset which was extracted with the help of the ground truth table. This table essentially refers to the labelled dataset which denotes the location and type of anomaly/defect. After analyzing the dataset, the location where no defect was recorded were extracted and concatenated to form the benchmark dataset. This is an anomaly-free dataset. Once this is established, the next step is to traverse through the remaining inspections and preprocess the evaluating time series. Here, the mile post and channel of interest is also specified. This will refer to the time series where anomaly detection is to take place. Tables 4.1, 4.2, 4.3 are a high-level representation of the dataset, and it also underscores its imbalanced nature.

Each of these inspections comprises of 10 Mile Posts (MP): 1,2,3,4,5,6,7,8,9. In order to establish the benchmark data, clean mile posts were extracted. For this analysis, it was noticed that the Third inspection consisted of the least number of anomalies and had no maintenance recorded for the following mile posts: 1,2,4,5,6,7 and 8. Thus, these seven mile posts from the third inspection served as the benchmark data and the other time series from the rest of the inspections serve as the evaluation data. Before passing these time series into the algorithm, the data was normalized through a zero-mean normalization method. The main idea behind this is that it works by averaging of all data points in the dataset to zero.

Table 4.1: Dataset Description

Parameters	Acceleration	Geometry
No. of Inspections	5	5
No.of Mileposts	10	10
No. of channels	14	19

Table 4.2: Distribution of clean (anomaly-free) and anomalous data over the five inspections across 10 Mile Posts

Inspection	No.of data points		No. of defects	Type of Time Series
	Acceleration	Geometry		
First	63325	52912	73	Evaluation
Second	63277	52923	73	Evaluation
Third	62486	52955	24	Benchmark
Fourth	62486	52943	67	Evaluation
Fifth	70399	52950	51	Evaluation

Table 4.3: Definition of important metrics used [3]

Metric	Definition
True Positive Defects (tp)	Number of real defects correctly predicted as anomalies
False Positive Predictions (fp)	Number of anomalies wrongly predicted as real defects
False Negative Defects (fn)	Number of real defects wrongly predicted as normal

4.3 Matrix Profiling Inspired Framework - Case Study Results

Now that the benchmark and evaluation data is established, the efficiency of the main framework needs to be evaluated. While conducting preliminary experiments, it was observed that parameter tuning was vital throughout the process. The following sections provide more details regarding the hyperparameters or factor which could alter the performance of

the Matrix Profiling Algorithm.

4.3.1 Matrix Profiling Parameter Selection

Referring to Figure 3.5, and based on a few implementations, a few parameters were identified which could impact the performance of the framework if not tuned in an optimal way. These are discussed below:

- Window Size (w)

  In order to perform the matrix profiling algorithm, a sliding window is used to traverse through the benchmark and evaluation datasets to identify minimum Euclidean distances. Hence, w is an important parameter to tune. The length of the window size chosen should reflect the general pattern, which means that the pattern of the time series should be conserved. Keeping this concept in mind, it can be assumed that a low window size would yield more accurate results. Experiments were conducted by using six window sizes values: 50, 75 ,100, 300, 500 and 700. The figures displayed below show the affect of different window sizes on the same time series and it can be observed that using window sizes of smaller value conserves the pattern of the original time series leading to less discrepancies. The peaks with the highest Euclidean score can then be extracted. From Figure 8, it can be seen that the time series is smoothened further as the window size value increases. The intricacy of the time series is maintained when smaller window sizes are used (50,75 and 100). The highest Euclidean score is reported as an anomaly across the various window sizes, but the intention is to conserve the pattern and not to cause over smoothing. Another key point that was observed is that the Euclidean scores (represented on the y-axis) tend to increase as the window size increases. This is an expected behavior because as the number of subsequences

increases, there is more discrepancy in the data leading to a higher Euclidean distance. For this implementation, it would be more optimal if majority of the features present in the time series are captured and smaller the subsequences, the more accurate the Euclidean score will be. Hence, considering the computational trade-offs, a window size of 75 was chosen for this implementation.

- Channel Selection

  There are two ways to approach anomaly detection with the Matrix Profile Based framework which is either by using a single channel implementation or a multi-channel implementation. As mentioned in Table 1, there are multiple channels present for Acceleration and Geometry which are extracted using different sensors on the geometry car respectively. In order to predict a defect across one Mile Post (e.g., MP1), it was theorized that using the time series from either all of the acceleration or the geometry channels would populate numerous anomaly locations but with the use of the KDE algorithm, the locations with the highest distribution of data can be isolated and flagged as anomalies. The geometry channels performed well but when this theory was experimented with all the acceleration channels present in the dataset, it was seen that not all channels contained valuable information causing the accuracy to drop. An example of this is shown in the figures below. From Figure 9, it can be observed that the matrix profiling graph for the Vertical Acceleration 1 channel accurately predicts the anomaly at MP 1.67 but the Vertical Acceleration 2 Channel does not produce a maximum Euclidean score at this location, but its highest peak is at MP1.84. Moreover, the Euclidean scores are very low which potentially could mean that this data might contain only noise. In conclusion, in order to identify the best channels, the Matrix profiling algorithm is performed for each time series across the acceleration (14 runs) and geometry channels (19 runs). Their anomalies are collected, and each channel

undergoes a performance evaluation. The channels with the highest performance can then be considered as the best channels.

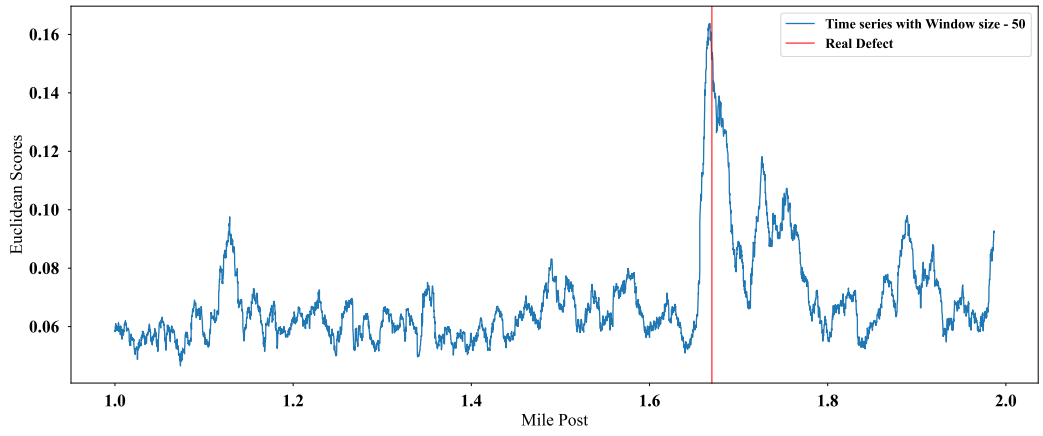
Matrix Profiling Results

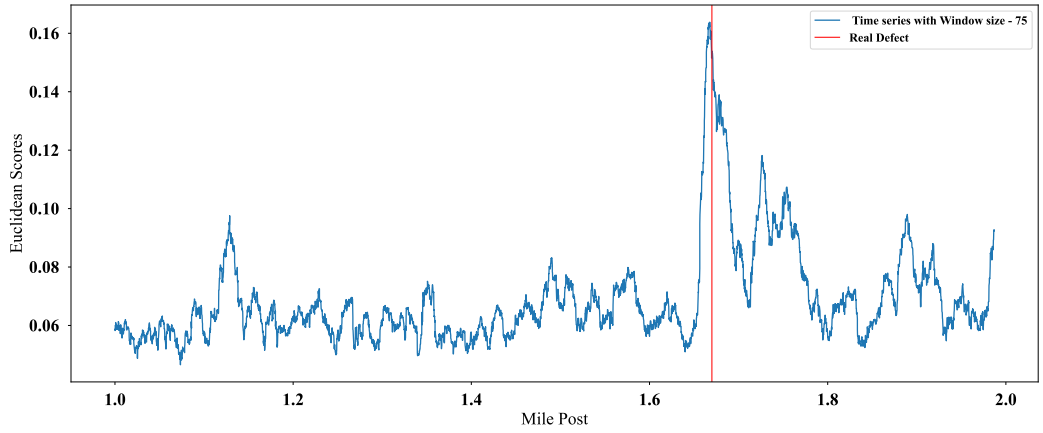
To evaluate the efficacy of this model, it was evaluated across multiple geometry and acceleration channels. The metrics used to conduct evaluation will be discussed in the following paragraphs and how this information can be leveraged to find the best channels as seen in parameter selection. MP3 was not considered because it did not have a single clean inspection. The following table represent the evaluation metrics used and their definitions. Recall and Precision are the two primary evaluation methods for a Machine Learning model, but the main focus of the evaluation relies on Recall. This is because there is more cost associated with predicting false negatives (anomaly) compared to false positives (normal). However, it is still necessary to achieve a tangible precision value. Our definitions of True positive (tp),False positive (fp) and False negative (fn) are seen in Table 4.3.

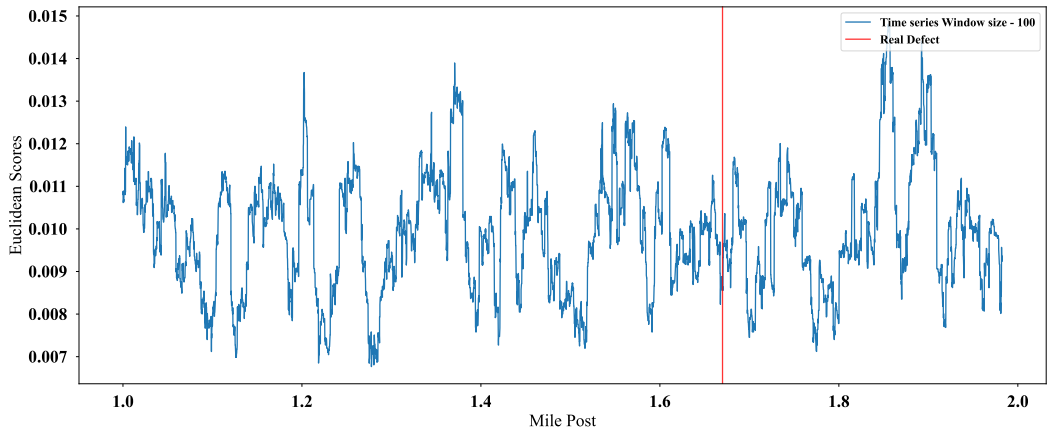
$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn}$$

To re-iterate the goal, which is to achieve a high recall, this would mean that the intent is to achieve a high True Positive Rate (TPR) and concurrently, a low False Negative Rate (FNR). Considering the above metrics, the recall and precision for the four different evaluation inspections is shown below. Since this case study deals with a real-world dataset, a threshold (t) was used. This refers to the accepted error range for an anomaly to be considered


Time series with Window size - 50
Real Defect
Euclidean Scores
0.16
0.14
0.12
0.10
0.08
0.06
1.0
1.2
1.4
1.6
1.8
2.0
Mile Post


Time series with Window size - 75
Real Defect
Euclidean Scores
0.16
0.14
0.12
0.10
0.08
0.06
1.0
1.2
1.4
1.6
1.8
2.0
Mile Post


Time series Window size - 100
Real Defect
Euclidean Scores
0.015
0.014
0.013
0.012
0.011
0.010
0.009
0.008
0.007
1.0
1.2
1.4
1.6
1.8
2.0
Mile Post

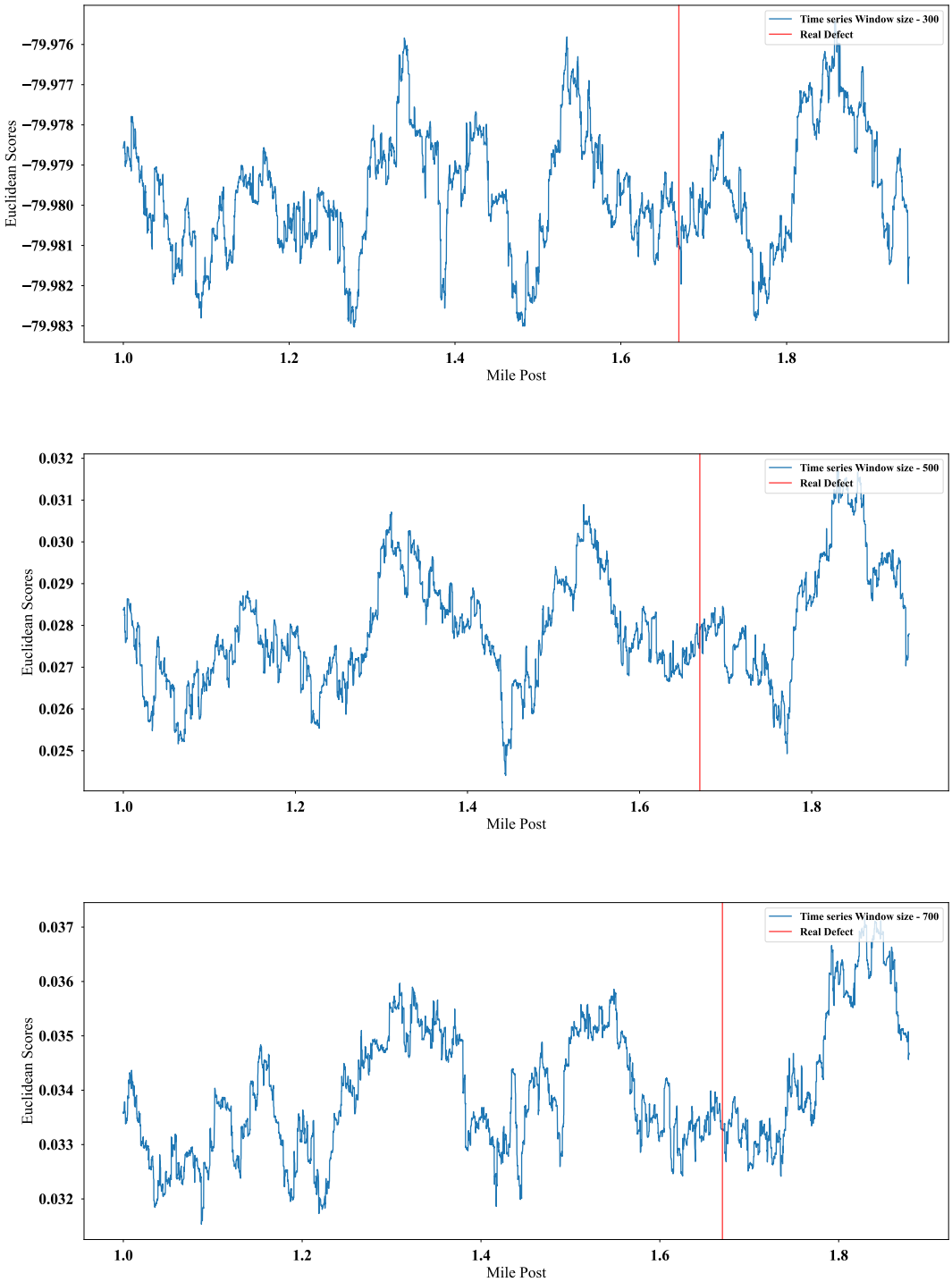


Figure 4.1: Effect of the window size values (50,75,100,300,500,700) parameter on a time series across Mile Post 1 of the Vertical Acceleration 1 Channel

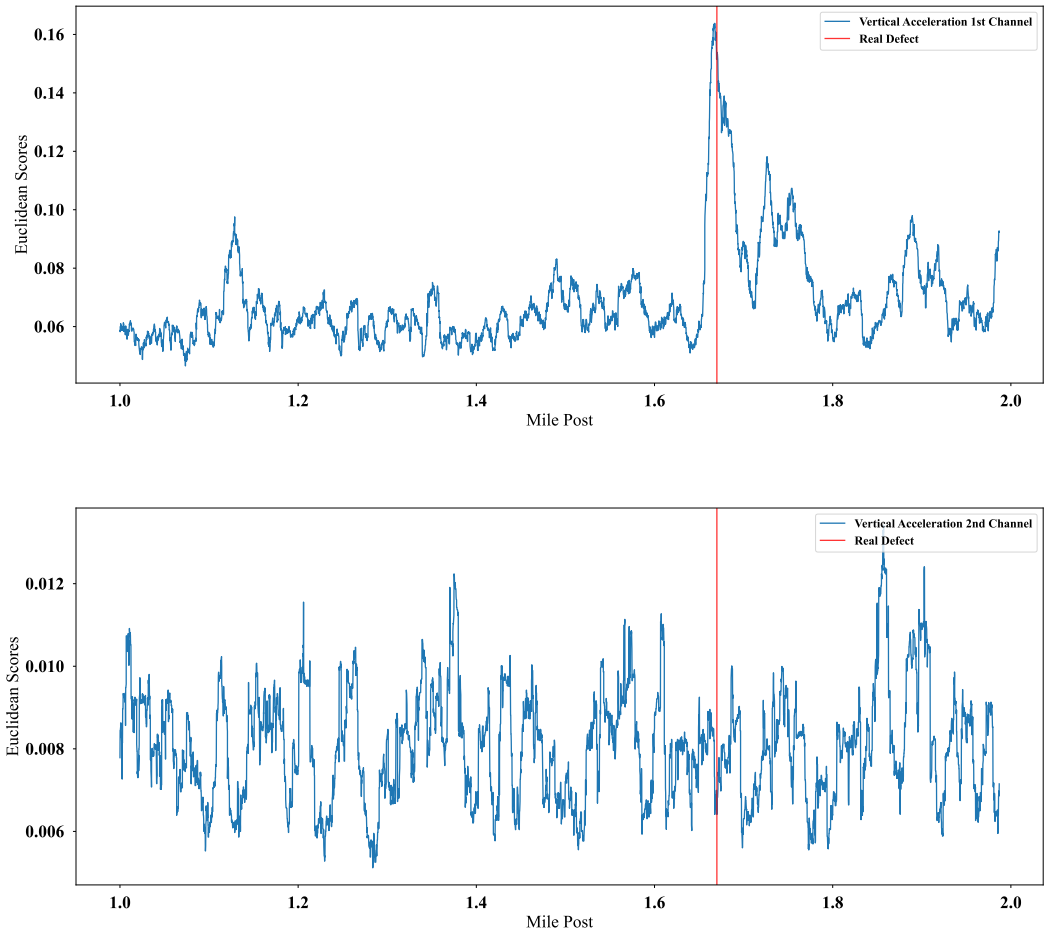


Figure 4.2: Comparison of a good performance with a bad performance channel from MP1.

as a true positive. P_{loc} are the anomalies detected from the model and D_{loc} are the real anomalies extracted from the ground truth provided. Using the above definitions of Recall and Precision, the following results in 4.4 were generated for evaluation purposes.

$$|P_{loc} - D_{loc}| \leq t \rightarrow True\ Positive$$

$$|P_{loc} - D_{loc}| \geq t \rightarrow False\ Positive$$

Four threshold values (t) : 0.01, 0.02, 0.03, 0.04 miles were used to analyze the recall and

Table 4.4: Results across the four inspections with the best channel and its performance with a threshold of 0.02 miles.

t=0.02 miles or ∼106 feet				
Data Source	**Inspection**	**Channel**	**Recall**	**Precision**
Acceleration	First	Vertical Acceleration 1	0.92	0.73
Acceleration	Second	Vertical Acceleration 1	0.67	0.58
Acceleration	Fourth	Vertical Acceleration 1	0.90	0.41
Acceleration	Fifth	Vertical Acceleration 1	0.71	0.21
Average Acceleration			0.8	0.48
Geometry	First	Right Profile 62	0.92	0.71
Geometry	Second	Left Profile	0.67	0.76
Geometry	Fourth	Right Align SC	0.9	0.43
Geometry	Fifth	CrossLevel	0.71	0.33
Average Geometry			0.98	0.56

Table 4.5: Results across the four inspections with the best channel and its performance with a threshold of 0.025 miles.

t=0.025 miles or ∼132 feet				
Data Source	**Inspection**	**Channel**	**Recall**	**Precision**
Acceleration	First	Vertical Acceleration 1	1.00	0.74
Acceleration	Second	Vertical Acceleration 1	0.74	0.61
Acceleration	Fourth	Vertical Acceleration 1	1.00	0.43
Acceleration	Fifth	Vertical Acceleration 1	0.71	0.21
Average Acceleration			0.86	0.50
Geometry	First	CrossLevel Rate	0.96	0.74
Geometry	Second	Left Profile	0.96	0.76
Geometry	Fourth	Right Align 62	1.00	0.48
Geometry	Fifth	CrossLevel	1.00	0.33
Average Geometry			0.98	0.58

precision. It is understandable that as the threshold (allowed tolerance) increases, the recall also increases. Table 4.4, 4.5, 4.6, 4.7, 4.8 gives information about the different inspections and which channel performed the best. It can be seen that across all the inspections, the geometry dataset outperformed the acceleration one. This could be due to the fact that

Table 4.6: Results across the four inspections with the best channel and its performance with a threshold of 0.03 miles.

t=0.03 miles or ∼158 feet				
Data Source	**Inspection**	**Channel**	**Recall**	**Precision**
Acceleration	First	Vertical Acceleration 1	1.00	0.74
Acceleration	Second	Vertical Acceleration 1	0.74	0.61
Acceleration	Fourth	Vertical Acceleration 1	1.00	0.45
Acceleration	Fifth	Vertical Acceleration 1	0.71	0.21
Average Acceleration			0.86	0.50
Geometry	First	CrossLevel Rate	1.00	0.76
Geometry	Second	Left Profile	1.00	0.79
Geometry	Fourth	CrossLevel	1.00	0.45
Geometry	Fifth	CrossLevel	1.00	0.33
Average Geometry			1.0	0.59

Table 4.7: Results across the four inspections with the best channel and its performance with a threshold of 0.04 miles.

t=0.04 miles or ∼211 feet				
Data Source	**Inspection**	**Channel**	**Recall**	**Precision**
Acceleration	First	Vertical Acceleration 1	1.00	0.74
Acceleration	Second	Vertical Acceleration 1	0.78	0.64
Acceleration	Fourth	Vertical Acceleration 1	1.00	0.45
Acceleration	Fifth	Vertical Acceleration 1	0.71	0.21
Average Acceleration			0.87	0.51
Geometry	First	CrossLevel Rate	1.00	0.76
Geometry	Second	Left Profile	1.00	0.79
Geometry	Fourth	CrossLevel	1.00	0.45
Geometry	Fifth	CrossLevel	1.00	0.33
Average Geometry			1.0	0.59

direct information about the track lies in the geometry dataset. The acceleration dataset on the other hand gave a decent recall and one key point that should be noted is that the "Vertical Acceleration 1 " channel was predicted to be the best channel unanimously.

Table 4.8: Grand Average across all inspections and the different thresholds

Threshold (t)	Data Source	Recall	Precision
0.02 miles	Acceleration	0.80	0.48
	Geometry	0.98	0.56
0.025 miles	Acceleration	0.86	0.50
	Geometry	0.98	0.58
0.03 miles	Acceleration	0.86	0.50
	Geometry	1.00	0.59
0.04 miles	Acceleration	0.87	0.51
	Geometry	1.00	0.59
Cumulative Average		0.92	0.54

4.4 Motif Based Framework- Case Study Result

As noted before, the Matrix Profile Based framework tends to generate false positives. In order to mitigate this problem, the Motif Based framework was proposed since it is more intuitive and involves machine learning models.

Similar to the Matrix Profile Based framework,there is a need to establish the benchmark and evaluating time series.The difference between both these frameworks is that the OCSVM requires parameter tuning. For this step, two benchmark data are needed as explained in 4.4.

Motif Based Parameter Selection

Based on some experimentation, the following parameters were identified as those which could alter the performance of the framework if not tuned.

- OCSVM Hyperparameters:

  The main hyperparameters in OCSVM that need to be tuned are the Gaussian Kernel (σ) and the Regularization coefficient (ν) [20]. νplays an important role in establishing

an upper bound for the allowed outliers [20]. Tuning this parameter helps to eliminate the presence of false positives. The σ parameter is important because it controls the flexibility of the decision boundary [20]. Keeping this in mind, the value of $\sigma$is crucial as setting a low or high value could lead to underfitting or overfitting respectively.

In order to overcome the problem of overfitting and underfitting, two benchmark(clean inspection) data were used. One would serve as training and another as testing. Here the parameters are tuned and the values which yield the least number of anomalies were extracted as the best hyperparameters. The following hyperparameters in Table 4.9 were used.

Table 4.9: Hyperparameter tuning for OCSVM with a Gaussian kernel

Parameter	**Values used for tuning**	**Step size**
σ	[1e^-6, 0.7]	0.01
ν	[1e^-6, 0.7]	0.01

- KDE Bandwidth:

If numerous anomalies are populated from the OCSVM models, it is necessary to identify the distribution of the predicted anomalies. KDE can be leveraged to extract the anomalies with a high distribution. An important parameter that needs to be tuned here is the bandwidth. A small or large bandwidth value can lead to undersmoothing or oversmoothing respectively.

In a undersmoothed plot, each data point would have a peak leading to a large number of false positives. On the contrary, in an oversmoothed plot, few peaks could be formed which could potentially cause the true positives to be overlooked. Based on visual experiments as shown below, an optimum bandwidth value of 0.01 was chosen.

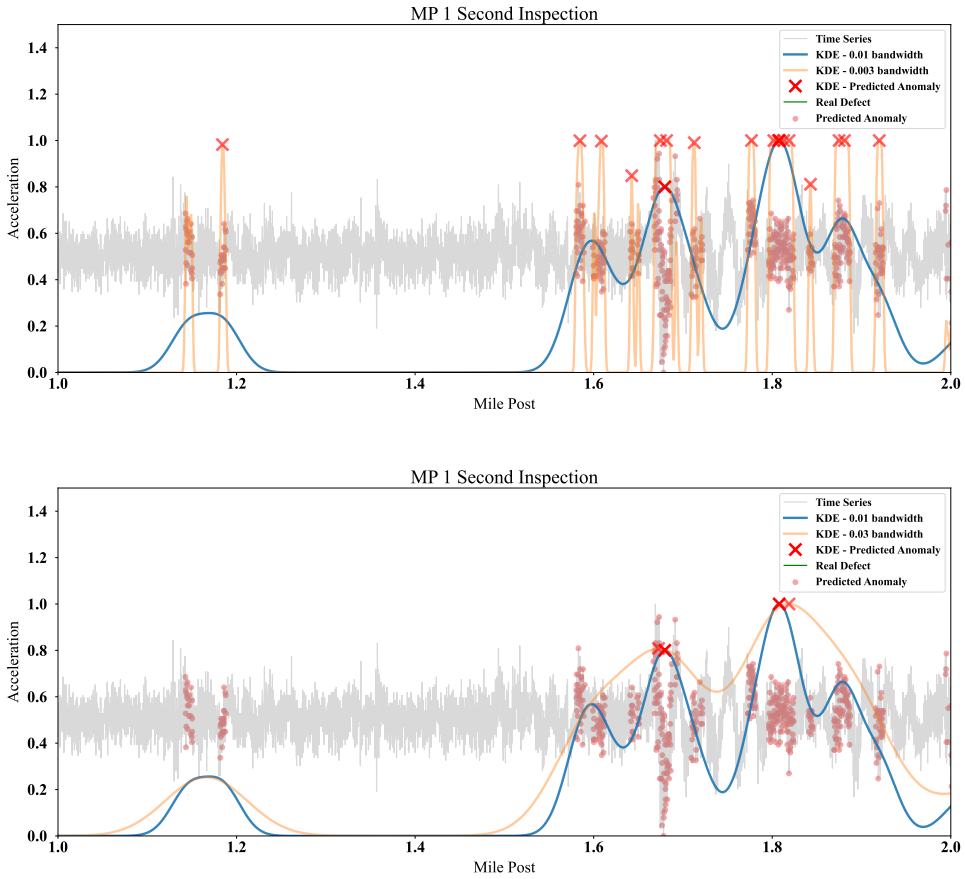


Figure 4.3: KDE graph for MP1 for a small bandwidth and large bandwidth value depicting undersmoothing and oversmoothing and respectively

Motif Based Results

In order to evaluate the performance of this framework, it was evaluated across all inspections of the Vertical Acceleration 1 Channel through the following mile posts : MP1, MP2, MP4, MP6 and MP8. These mileposts had atleast 2 clean inspections and hence were chosen. The metrics and along with the threshold conditions to identify True and False positives which were discussed in 4.3.1 were also used to evaluate this framework.

It can be seen that as the threshold value increases, the recall and precision increase as well and this phenomenon is understandable since the acceptable error range is increased. With

Table 4.10: OCSVM results across all inspections with respect to three threshold ranges

Threshold = 0.02 miles or ∼106 feet		
Inspection	**Recall**	**Precision**
First	0.12	0.17
Second	0.38	0.55
Fourth	0.5	0.21
Average	0.33	0.31
Threshold = 0.03 miles or ∼158 feet		
Inspection	**Recall**	**Precision**
First	0.47	0.5
Second	0.38	0.55
Fourth	0.67	0.29
Average	0.51	0.45
Threshold = 0.04 miles or ∼211 feet		
Inspection	**Recall**	**Precision**
First	0.82	0.70
Second	0.56	0.69
Fourth	0.83	0.33
Average	0.74	0.57
Cumulative Average	0.53	0.44

a threshold of 0.04 miles, the recall and precision reached is 0.74 and 0.57 respectively. This is lower than the recall and precision for Matrix Profile Based framework which was 0.87 and 0.51 (Acceleration) and 1.0 and 0.59 (Geometry) respectively.As mentioned earlier, the goal here is to achieve a high recall. It can be seen that the Motif Based framework does not have a good performance but these could potentially be further improved by experimenting with the feature extraction window size and by incorporating a multi-channel mode in this framework.

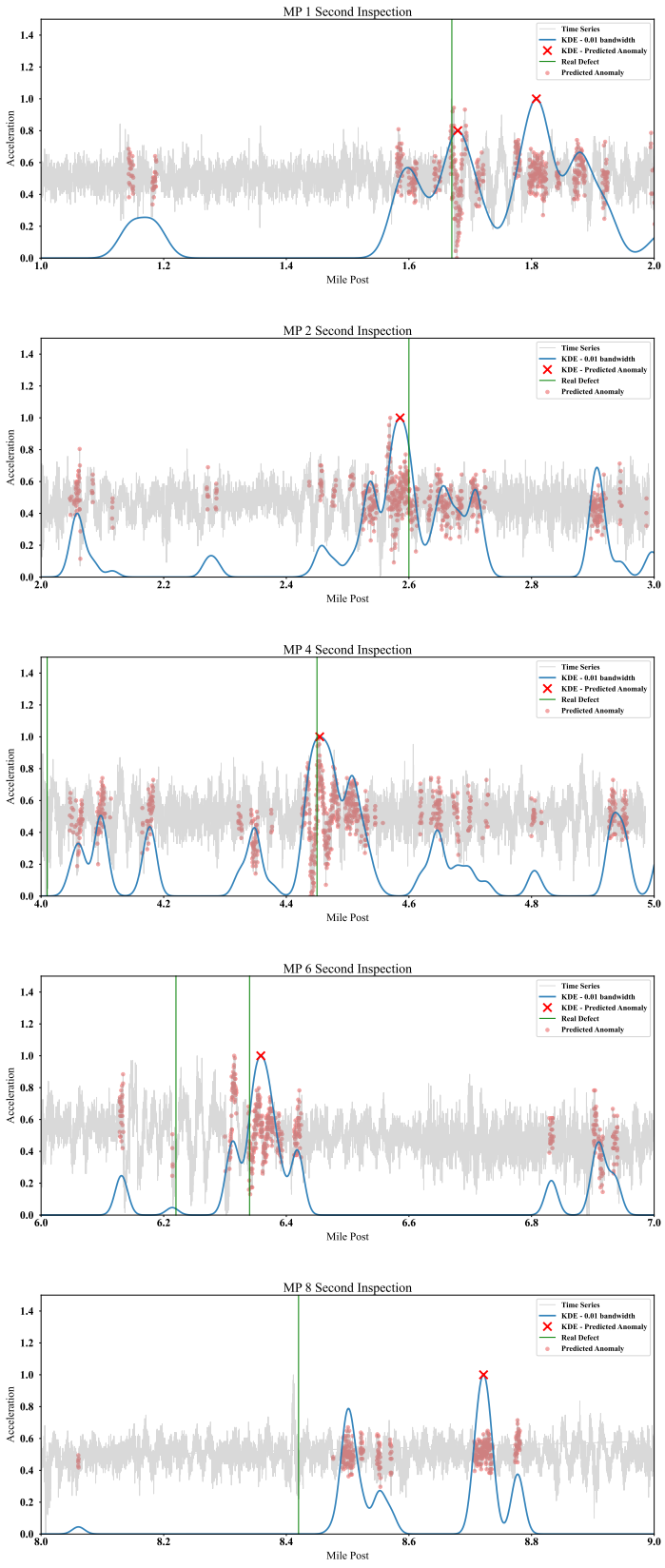


Figure 4.4: Examples of the Motif Based Results across the Second Inspection

Chapter 5

Discussion

This thesis presents two frameworks for time series anomaly detection. For the first framework, the geometry and acceleration channels in the railroad dataset were used for evaluation. The first framework yielded optimal results compared to the second framework. An important factor that could affect the results lies in hyperparameter tuning. The first framework involves tuning of more heuristic hyperparameters such as window size and channel selection. The second framework incorporates an OCSVM model which requires an optimal tuning of σ and ν. These parameters are less heuristic and need to be experimented on a broad range of values. Moreover, the scope of the OCSVM was limited to one vertical channel so this could be explored further.

From the results, it can be seen that the first framework performs better than the second framework. The reason for this could be due to the sensitive hyperparameter tuning process associated with the second framework. Even though the results are acceptable, the following are a few suggestions which could be used to improve the performance of the frameworks:

The matrix profile based framework yielded a cumulative average recall value of 0.92. This is proof that this algorithm is working remarkably well. This means that the number of false negatives is low which is the desired outcome of developing the algorithms. It has a tangible precision value of 0.59. This precision is promising but there is still scope for improvement. During the evaluation phase, it was seen that some of the matrix profile graphs derived had a very low euclidean score which may not be vital thus leading to high false positives. A

heuristic threshold method could be introduced here to eliminate these, thereby increasing precision.

The motif based framework yielded a recall value of 0.74 with an error range of 0.04 miles. This is not as promising as the first framework but it could be further enhanced by tuning the hyperparameters with a smaller step size. This method was also conducted by using only a single vertical acceleration channel so it could be extended to the geometry channels or a combination of both.

Chapter 6

Conclusions

Time series anomaly detection for smart infrastructures can be a very beneficial tool and could reduce redundancy in human efforts and cost. However, this is a very challenging domain owing to the insufficiency of labelled data. The main reason behind using the two frameworks proposed is to enable asynchronous comparison of time series that reflect operating states of a system across two time series that measure the system performance.

This thesis proposed two unsupervised machine learning methodologies - matrix profile based and motif based which rely only on anomaly-free data to perform anomaly detection. Since, the main goal is to detect anomalies, recall is given more importance compared to precision. This is because the main idea is to identify and locate all the defects due to which false positives could be generated .The performance of the matrix profile based framework is very note worthy with regards to its cumulative high recall of 0.92 and a tangible precision of 0.54. In order to evaluate the methodology, a real world case study of a railroad track was used to evaluate their performances. The precision could still be increased further by introducing a threshold heuristic. The second framework which is the motif based framework has a lower performance. The cumulative recall and precision was 0.53 and 0.44. It could be because there are overtly sensitive hyperparameters involved in the second framework.This could be further improved by performing a more intricate selection of OCSVM hyperparameters and extending it to other channels and by investigating alternative classification-based anomaly detection methods.

Bibliography

[1] 2019. URL https://www.aar.org/wp-content/uploads/2021/02/AAR-United-States-Fact-Sheet.pdf.

[2] 2020. URL https://towardsdatascience.com/introduction-to-anomaly-detection-c651f38cc

[3] Kyle D Anderson, Mario E Berges, Adrian Ocneanu, Diego Benitez, and Jose MF Moura. Event detection for non intrusive load monitoring. In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, pages 3312–3317. IEEE. ISBN 1467324213.

[4] Srinidhi Bhat and Sanjay Singh. One-class support vector machine for data streams. In *2020 IEEE REGION 10 CONFERENCE (TENCON)*, pages 1130–1135, 2020. doi: 10.1109/TENCON50793.2020.9293814.

[5] Yin Cai, Mei-Ling Shyu, Yue-Xuan Tu, Yun-Tian Teng, and Xing-Xing Hu. Anomaly detection of earthquake precursor data using long short-term memory networks. *Applied Geophysics*, pages 1–10, 2019.

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009. ISSN 0360-0300.

[7] Len Feremans, Vincent Vercruyssen, Boris Cule, Wannes Meert, and Bart Goethals. Pattern-based anomaly detection in mixed-type time series. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 240–256. Springer, 2019.

[8] Hesam Izakian and Witold Pedrycz. Anomaly detection in time series data using a fuzzy c-means clustering. In *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*, pages 1513–1518, 2013. doi: 10.1109/IFSA-NAFIPS.2013.6608627.

[9] Bouchra Lamrini, Augustin Gjini, Simon Daudin, Pascal Pratmarty, François Armando, and Louise Travé-Massuyès. Anomaly detection using similarity-based one-class svm for network traffic characterization. In *DX@ Safeprocess*.

[10] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the 2003 SIAM international conference on data mining*, pages 25–36. SIAM, 2003.

[11] Jingxiao Liu, Siheng Chen, George Lederman, David B Kramer, Hae Young Noh, Jacobo Bielak, James H Garrett, Jelena Kovačević, and Mario Bergés. Dynamic responses, gps positions and environmental conditions of two light rail vehicles in pittsburgh. *Scientific data*, 6(1):1–11, 2019. ISSN 2052-4463.

[12] Philip W Loveday and Craig S Long. Long range guided wave defect monitoring in rail track. In *AIP Conference Proceedings*, volume 1581, pages 179–185. American Institute of Physics. ISBN 0735412111.

[13] J. Ma and S. Perkins. Time-series novelty detection using one-class support vector machines. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1741–1745 vol.3, 2003. doi: 10.1109/IJCNN.2003.1223670.

[14] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, volume 89, pages 89–94, 2015.

[15] Abdullah Mueen, Eamonn Keogh, Qiang Zhu, Sydney Cash, and Brandon Westover. Exact discovery of time series motifs. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 473–484. SIAM, 2009.

[16] Afrooz Purarjomandlangrudi, Amir Hossein Ghapanchi, and Mohammad Esmalifalak. A data mining approach for fault diagnosis: An application of anomaly detection algorithm. *Measurement*, 55:343–352, 2014.

[17] Ingo Steinwart, Don Hush, and Clint Scovel. A classification framework for anomaly detection. *Journal of Machine Learning Research*, 6(2), 2005. ISSN 1532-4435.

[18] James P Theiler and D Michael Cai. Resampling approach for anomaly detection in multispectral images. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery IX*, volume 5093, pages 230–240. International Society for Optics and Photonics.

[19] Arijit Ukil, Soma Bandyoapdhyay, Chetanya Puri, and Arpan Pal. Iot healthcare analytics: The importance of anomaly detection. In *2016 IEEE 30th international conference on advanced information networking and applications (AINA)*, pages 994–997. IEEE, 2016.

[20] Siqi Wang, Qiang Liu, En Zhu, Fatih Porikli, and Jianping Yin. Hyperparameter selection of one-class support vector machine by self-adaptive data shifting. *Pattern Recognition*, 74:198–211, 2018.

[21] Chin-Chia Michael Yeh, Nickolas Kavantzas, and Eamonn Keogh. Matrix profile vi: Meaningful multidimensional motif discovery. In *2017 IEEE international conference on data mining (ICDM)*, pages 565–574. IEEE, . ISBN 1538638355.

[22] Chin-Chia Michael Yeh, Helga Van Herle, and Eamonn Keogh. Matrix profile iii: the

matrix profile allows visualization of salient subsequences in massive time series. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 579–588. IEEE, . ISBN 1509054731.

[23] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1409–1416, 2019.