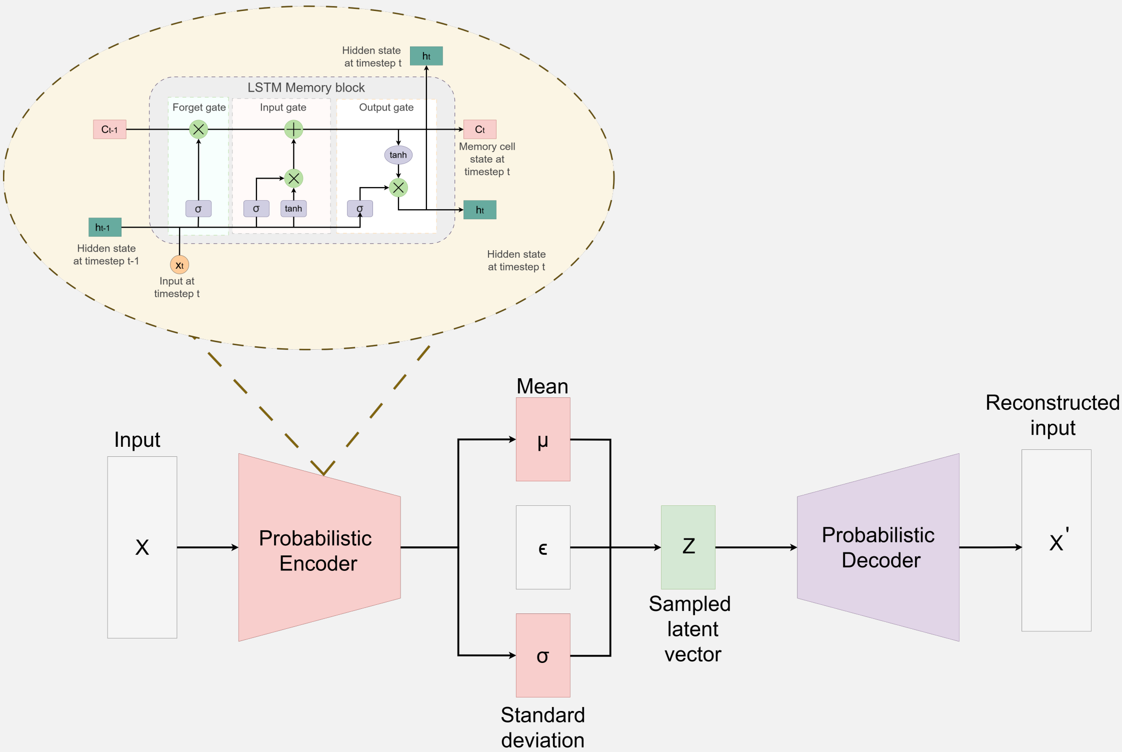
CHALMERS
UNIVERSITY OF TECHNOLOGY



Multivariate anomaly detection with LSTM layered Variational Autoencoder

A thesis performed with unsupervised learning on unlabeled boat signal data.

Master's thesis in Complex Adaptive Systems

EMILY BLIXT
LINUS KULLMYR

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Multivariate anomaly detection with LSTM layered Variational Autoencoder

A thesis performed with unsupervised learning on unlabeled boat signal data.

Emily Blixt & Linus Kullmyr

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2025

Multivariate anomaly detection with LSTM layered Variational Autoencoder
A study performed with unsupervised learning on unlabeled boat signal data.
EMILY BLIXT
LINUS KULLMYR

Supervisor: Gökcen Özcelik, Volvo Penta, Connectivity
Examiner: Mats Granath, Institution of physics at Gothenburg University Department

Master's Thesis 2025
Department of Physics
Division of Complex Adaptive Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Architecture of an LSTM-based Variational Autoencoder (VAE), highlighting the internal structure of an LSTM layer and its role in encoding and decoding probabilistic representations.

Typeset in LaTeX
Gothenburg, Sweden 2025

Multivariate anomaly detection with LSTM layered Variational Autoencoder
A study performed with unsupervised learning on unlabeled boat signal data.
EMILY BLIXT
LINUS KULLMYR
Department of Some Subject or Technology
Chalmers University of Technology

Abstract

The aim of this thesis was to develop and evaluate the effectiveness of a recurrent neural network layered autoencoder model for detecting anomalies in multivariate time-series data, with a focus on improving the accuracy and reliability of diagnostic data for Volvo Penta's boats. The primary goal was to leverage the relationships and correlations between signals to identify deviations that traditional models may fail to detect. The model's performance was assessed in terms of its ability to learn the structure of normal data, detect synthetic anomalies, and provide meaningful insights without relying on labeled datasets.

The study highlights the limitations of traditional evaluation metrics, which are often unsuitable for unsupervised learning approaches like the model used. Instead, the model's effectiveness was demonstrated through reconstruction error analysis and its ability to handle the complexities of multivariate time-series data. Challenges such as data dimensionality, sequence length optimization, and noise handling were addressed to enhance the model's robustness. The findings suggest that while the model excels at identifying synthetic anomalies and capturing temporal relationships, further work is needed to generalize its capabilities to real-world scenarios. This research lays the groundwork for improving diagnostic processes and supports the development of more adaptive and reliable anomaly detection systems.

Keywords: Anomaly, detection, LSTM, VAE, correlation, unsupervised.

Acknowledgements

We would like to express our gratitude to Volvo Penta for providing us with the opportunity to write this thesis, as well as the resources and support necessary to conduct our research. Moreover, we are grateful for the opportunity to gain valuable office experience and learn more about the marine industry. The work environment was exceptionally pleasant and conducive to productivity. We are especially grateful to our supervisor, Gökcen Özcelik, for her invaluable guidance, support, and dedication throughout the project. Additionally, we would like to thank our examiner, Mats Granath, for his insightful guidance and contributions which have helped us complete this project.

Emily Blixt & Linus Kullmyr, Gothenburg, May 2025

Glossary

Anomaly A data point that deviates past a threshold from expected patterns in a dataset. Anomalies may indicate meaningful irregularities, and can be classified as point, contextual, or collective depending on the setting.

Corrected Anomaly An anomaly data point that has been replaced with the model's reconstructed estimate.

Z-score A statistical measure that expresses how far a data point is from the mean of a dataset in terms of standard deviations. It is commonly used to identify anomalies, with values significantly higher or lower than zero indicating outliers.

IQR Interquartile Range (IQR) indicates the range within which the middle 50% of the data points fall. It is calculated as the difference between the third quartile (Q3) and the first quartile (Q1). Outliers are identified as points beyond 1.5 times the IQR from Q1 or Q3.

Mahalanobis Distance A distance metric that measures how far a point is from the mean of a distribution, accounting for the correlations within the dataset. It is commonly used in anomaly detection and classification of multivariate data.

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

VAE	Variational Autoencoder
LSTM	Long Short Term Memory
PCA	Principle Component Analysis
t-SNE	t-Distributed Stochastic Neighbor Embedding
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
ELBO	Evidence Lower Bound
KL	Kullback-Leibler
RL	Reconstruction loss
ADF	Augmented Dickey-Fuller
ACF	Autocorrelation
PACF	Partial autocorrelation
SMA	Simple Moving Average
k-NN	k-Nearest Neighbor
GAN	Generative Adversarial Network
SVM	Support Vector Machine
RNN	Recurrent Neural Network
BPTT	Backpropagation Through Time

Contents

List of Figures

List of Tables

1

Introduction

Anomaly detection is a well-established area of research, yet it continues to present several significant challenges [1] [2]. A primary difficulty lies in the definition of an anomaly, as its characterization is highly context-dependent and may vary across different domains. Additionally, acquiring high-quality labeled data is often expensive and time-consuming. This is particularly problematic, as anomalies are inherently rare and unpredictable, with new variations continually emerging that may not have been previously encountered, complicating the task of detection. Various approaches and models have been proposed to address these challenges, including statistical methods, machine learning techniques, and deep learning architectures. Unsupervised models for anomaly detection have several advantages, especially in terms of scalability and cost effectiveness. Since they don't require labeled datasets, unlike supervised methods, the time consuming process of manually annotating data can be avoided. This is particularly useful when anomalies are rare, complex, or difficult to define and annotating the data accurately requires specialized knowledge. One notable strength of unsupervised models are their adaptability. They can more easily adapt to changing environments without needing to be updated with new labeled examples. At the same time, there are certain challenges. One of the biggest is interpretability. Because unsupervised models aren't trained on actual anomalies, it can be difficult to understand why certain data points are flagged as anomalous. Without labeled data, it's also harder to identify the cause of an anomaly, which can complicate troubleshooting and follow up actions [2] [3].

The data is from a product developed by Volvo Penta for leisure boat users that enables vessel owners to monitor vessel and engine data in real-time, track routes, receive alerts, and more on their smart device. The product is tested using simulation rigs and boats at sea, where diagnostic data is collected. However, as with any extensive dataset, outliers and faulty data are inevitable, making their detection a crucial aspect of the testing process.

This thesis explores the possibility to increase the reliability of the captured data by utilizing anomaly detection through unsupervised machine learning, specifically Long Short-Term Memory (LSTM)-based Variational Autoencoders (VAE), as a stepping stone to optimize the testing process of the product and enhance data quality. Given that data readings may be imperfect, anomaly detection helps identify these errors. This approach lays the groundwork for potential system improvements, contributing to more reliable and efficient rigs over time.

1.1 Aim

The aim of this thesis is to investigate the potential of LSTM-VAE models for identifying anomalies within multivariate time-series data collected from Volvo Penta's diagnostic systems. This research seeks to enhance the reliability and accuracy of diagnostic data by leveraging the model's ability to detect deviations and flag them as anomalies. By focusing on the relationships between signals, the model aims to provide a robust foundation for identifying faulty data, which can support further work in improving the diagnostic reliability and overall functionality of Volvo Penta's systems.

1.2 Delimitations

The scope was intentionally limited in several areas to maintain a clear focus. The corrected signals, which would originally reflect what the model thought the values should be, lacked foundation and instead remained for further insight in model behavior and performance. Additionally, labeled datasets were not created, as the research prioritized exploring the model's unsupervised capabilities in identifying deviations based solely on data patterns.

The dataset used for the study was restricted to a single boat and included only 10 signals, representing a small subset of the total available signals and boats within the system. This limitation was chosen to ensure a manageable and focused investigation as a proof of concept rather than achieving comprehensive coverage and providing a product.

2

Theory

This section provides the theoretical background necessary for the reader to understand the problem, the model and the evaluation methods.

2.1 Time Series Data

Data points collected sequentially over time are called time series data. Time series are usually divided into two types of categories: *univariate* and *multivariate* time series. A univariate time series consists of data points for a single variable over time, whereas a multivariate time series tracks multiple variables over the same period. Since time series measures behavior over time for one or several variables, it can be decomposed into seasonal patterns, long-term trends, cyclical fluctuations, and residual noise. Seasonal patterns are recurring variations that occur at regular fixed intervals, such as annually or monthly, and repeat with a consistent frequency over time. Trends refers to the long-term changes in the data, such as an increase, stability or decrease over time. Cyclical fluctuations are prolonged shifts that diverge from the ordinary pattern of a series, showing repeated irregular fluctuations, their duration depending on the characteristics of the time series. Residual noise corresponds to random or irregular occurrences in the series that are left after removing regular patterns [4] [5].

2.1.1 Stationarity

Another characteristic of time series is the distinction between *stationarity* and *non-stationarity*. Non-stationary means that the distribution of the time series changes over time, and the relationships between observations depend on when they are made. Stationarity means that the statistical properties of the time series do not depend on time, and the patterns controlling the process stay the same over time. This results in a stable, unchanging behavior. Stationarity is commonly classified into *weak stationarity* and *strict stationarity*. A time series $\{Y_t\}_{t=1}^{T}$ is a sequence of observations indexed by time t, where $t = 1, 2, \ldots, T$. At each time point, Y_t is a random variable whose value can vary according to some probability distribution, reflecting the inherent randomness in the process over time. Such a time series is said to be *weakly stationary* if the following conditions hold [6]:

$$\mathbb{E}[Y_t] = \mu \quad \text{for all}\, t$$

where $\mathbb{E}$ represents the expected value (mean) of the time series, and μ is a constant.

$$\mathrm{Var}(Y_t) = \sigma^2 \quad \text{for all}\, t$$

where $\mathrm{Var}(Y_t)$ is the variance and σ^2 is a constant.

$$\mathrm{Cov}(Y_t, Y_{t+h}) = \gamma(h) \quad \text{for all}\, t \text{ and fixed lag } h$$

where $\mathrm{Cov}(Y_t, Y_{t+h})$ is the autocovariance, $\gamma(h)$ is a function of the lag h and does not depend on t, meaning the covariance between Y_t and Y_{t+h} only depends on the difference in time (the lag) and not the specific time points.

A time series Y_t is strictly stationary if the joint distribution is the same for any set of time points $t_1, t_2, \ldots, t_n$ and for all time lags k. This means that the distribution of $Y_{t_1}, Y_{t_2}, \ldots, Y_{t_n}$ is identical to the distribution of $Y_{t_1-k}, Y_{t_2-k}, \ldots, Y_{t_n-k}$, where k is any time shift. This means that the statistical properties of the process (such as mean, variance, autocovariance etc.) do not change over time, regardless of when the observations are made. Strict stationarity is a stronger condition than weak stationarity, as it requires the entire distribution of the process to be the same at all time points [7].

To determine whether a process is stationary, it is common to use the *Augmented Dickey-Fuller* (ADF) test [8]. The test is based on the autoregressive process equation:

$$Y_t = \alpha Y_{t-1} + X_t \quad \text{for} \quad t = 1, 2\cdots \quad \text{where:}$$

- X_t is a stationary process,
- Y_t is the value of the time series at time t,
- α is the coefficient that determining how much of the past value Y_{t-1} influences the current value Y_t

The process Y_t has a unit root if $\alpha = 1$, indicating that the time series is non-stationary and stationary if $|\alpha| < 1$. The ADF test evaluates the null hypothesis that the process is non-stationary, with the alternative hypothesis being that the process is stationary. When the p-value is small (usually below 0.05), you reject the null hypothesis and conclude that the series is stationary. If the p-value is large, you fail to reject the null hypothesis, indicating that the series is likely non-stationary [8].

2.1.2 Correlation

Correlation is a measure of the relationship between variables, showing both the strength and direction of their connection. There are different methods to measure correlation in data, with Pearson and Spearman correlation being two commonly used approaches. Pearson measures the linear relationship between two continuous variables, with the values of the Pearson correlation coefficient ranging from -1 to 1. Two variables with a correlation coefficient of 1 indicate a perfect positive linear relationship, meaning that an increase in one variable will result in a proportional increase in the other. The opposite is true for a correlation coefficient of -1, which indicates a perfect negative linear relationship. In this case, an increase in one variable will result in a proportional decrease in the other. A coefficient of 0 means that there is no linear relationship between the variables. Values between the extremes show varying degrees of positive or negative linear relationships, with values closer to 1 or -1 indicating stronger correlations. Spearman's rank correlation works similarly to Pearson's, in that it produces a correlation coefficient ranging from -1 to 1, indicating the intensity and direction of the relationship between two variables. Instead of measuring linear relationships, it evaluates the strength and direction of the monotonic relationship between the ranks of the variables. A monotonic relationship means that as one variable increases, the other either consistently increases or consistently decreases [9].

Since time series consists of measurements taken at consecutive time points, each variable can be correlated with its own past values over time, a relationship known as *autocorrelation* (ACF). The equation for the ACF coefficient ρ_k at lag k between the values of the time series at time t and $t+k$ is [6]:

$$\rho_k = \frac{\sum_{t=1}^{n-k}(y_t - \bar{y}_1)(y_{t+k} - \bar{y}_2)]}{\sqrt{\left[\sum_{t=1}^{n-k}(y_t - \bar{y}_1)^2\right]\left[\sum_{t=1}^{n-k}(y_{t+k} - \bar{y}_2)^2\right]}} = \frac{\operatorname{cov}(y_t, y_{t+k})}{\sigma(y_t)\sigma(y_{t+k})} \quad \text{where:}$$

- y_t is the value of the time series at time t,
- $\bar{y}_1 = \frac{1}{n-k}\sum_{t=1}^{n-k} y_t$ is the mean of the time series values from time 1 to $n-k$,
- $\bar{y}_2 = \frac{1}{n-k}\sum_{\tau=1+k}^{n} y_\tau$ is the mean of the time series values from time $1+k$ to n,
- n is the total number of time periods in the sample, where $t = 1, 2, ..., n$,
- k is the lag, representing the shift of the time series y relative to itself,
- $\sigma(y_t)$, $\sigma(y_{t+k})$ are the standard deviations of y_t and y_{t+k} respectively

The ACF assumes constant variance and a stable mean, so while it can be applied to non-stationary data, its interpretation becomes more challenging. For non-stationary series, the ACF typically declines slowly or remains high, reflecting trends or drifts. In contrast, for stationary series, the ACF drops quickly to zero, indicating no long-term dependencies [10].

Partial autocorrelation (PACF) is a variation of autocorrelation that measures the direct influence of the past value at lag k on the current value at timestep t, after removing the effects of the intermediate lags (1 to k-1). In other words, it quantifies the correlation between y_t and y_{t-k} after removing the effect of the intervening variables $y_{t-1}, y_{t-2}, y_{t-3}, \cdots, y_{t-k+1}$. The PACF coefficient is defined as [10]:

$$\phi_{kk} = \operatorname{Corr}(y_t, y_{t-k} \mid y_{t-1}, y_{t-2}, \ldots, y_{t-k+1})$$

One can compare two time series, consisting of any pair of related data sets, to examine their relationship over time, which is called *cross-correlation*. The cross-correlation function $r_k(xy)$ is defined as [11]:

$$r_k(xy) = \frac{\sum(x_t - \bar{x})(y_{t-k} - \bar{y})}{\sqrt{\sum(x_t - \bar{x})^2}\sqrt{\sum(y_t - \bar{y})^2}} \quad \text{where:}$$

- $\bar{x}$ is the mean of the time series x,
- $\bar{y}$ is the mean of the time series y,
- k is the lag, which represents the shift of time series y relative to x

Moving averages or sliding windows are common in time series analysis to reduce noise and better highlight underlying trends. This smoothing technique can be applied in various ways, with one common approach being the Simple Moving Average (SMA), where the average of the N most recent data points in the time series is calculated. There are also methods that assign weights to data points according to a defined rule or method in order give more importance to certain observations [12].

2.2 Anomaly Detection

Anomaly detection refers to identifying data points, known as anomalies or outliers, that deviate from the usual patterns or trends within a dataset. These deviations can often point to important issues or unexpected behaviors that require further investigation, making anomaly detection a very important area of research. The interest and relevance of anomalies is a critical aspect of anomaly detection. While closely related to noise handling, anomaly detection focuses on identifying and interpreting irregularities that may offer useful information, in contrast to noise, which is typically discarded or adjusted. The specific kind of irregularity or unusual pattern to identify in the data varies across contexts or applications, which is why anomalies are typically classified into the following three categories [2]:

- **Point anomalies.** Data points that are considered anomalous or outliers when compared to the rest of the dataset.
- **Contextual anomalies.** Data points that are anomalous only within a certain context or condition. This context can be such things as behaviors, spatial locations, or time.
- **Group or collective anomalies.** Individual data points that may not be anomalous on their own, but together they form a pattern that is unusual or unexpected.

Common techniques for detecting anomalies include neighbor-based methods like k-NN and LOF, clustering-based methods such as DBSCAN and k-means, one-class classification approaches like One-Class SVM and Isolation Forest, and deep learning models such as autoencoders and GANs [2] [13].

There are several challenges in anomaly detection, as it requires accurately identifying rare and often unpredictable patterns within large, complex datasets. Defining what is considered normal behavior versus anomalous behavior is often not straightforward, and the distinction between them is not always clear. In addition, typical patterns or trends may change over time. New anomalies can also occur over time and what qualifies as an anomaly may change dynamically. These challenges can be approached using different anomaly detection techniques, depending on the specific task at hand [2].

2.2.1 Supervised vs unsupervised learning

Supervised learning: refers to training a model on labeled data, where each input is paired with a correct output, allowing it to learn patterns and make predictions on new data based on this reference. So in the case of supervised anomaly detection, it means that the model has data labeled both normal and anomalous. Anomalies are often rare events that deviate from the normal pattern or expected data distribution, meaning they occur infrequently compared to regular data. This imbalance between normal and anomalous observations leads to class imbalance, where the majority of the data used to train a model belongs to the normal class. Moreover, labeling of anomalies is required, which means each individual deviation must be identified and correctly classified as either an anomaly or normal data, which is often a labor-intensive and time-consuming process. [2] [3].

Semi-supervised learning: Refers to a training approach where the model is trained on a small subset of labeled data. In the context of anomaly detection, this typically means training the model using only normal data, without any labeled anomalies. The model learns the patterns and behaviors of the normal data to distinguish between normal and anomalous data. This approach is often more practical than supervised methods, as it is easier to obtain data that is considered normal or non-anomalous [2] [3].

Unsupervised learning: Refers to training performed on completely unlabeled data, thus the model has no predefined outputs to learn from. Instead, it learns by identifying patterns, structures, or relationships within the data on its own. Unsupervised anomaly detection has a broader range of applications, as a lot of data is difficult to label. It can be challenging to identify what qualifies as an anomaly, especially since new, previously unknown anomalies may emerge. The primary goal is often to discover what constitutes an anomaly, rather than relying on pre-labeled data. Unsupervised anamoly detection is based on the assumption that normal instances occur much more frequently than anomalies in the data and that the model learns the structure of normal data to detect deviations. Since there are no labels the data is evaluated by measuring deviations from the learned patterns and assigning an anomaly score based on statistical, distance-based, or density-based methods to identify outliers [2] [3].

2.2.2 Anamoly detection in time series

Since time series data is sequentially ordered, the main challenge in time series anomaly detection lies in the fact that data points are not independent. They are influenced by previous values and temporal dependencies, complicating the identification of anomalies. Contextual anomalies in time series data refer to data points whose behavior is considered unusual based on their position or context within the sequence. Contextual anomaly detection identifies these anomalies, particularly when data points resemble each other within a given context, revealing anomalies that point anomaly detection might overlook. However, it may not always be adaptable as context cannot always be clearly defined. Collective anomalies in time series data involve a set of subsequences that, when viewed together, differ from the expected behavior, even if individual points appear normal. It can be difficult to detect anomalous sequences because they often vary in length, and the anomalous patterns may not always appear in the same order across different sequences.

2.3 Recurrent neural networks

In machine learning, artificial neural networks are models with interconnected layers of nodes that process inputs, similar to the structure of the brain. They consist of an input layer, one or more hidden layers, and an output layer. These models process and transform input data through weighted connections. This enables them to learn complex patterns and relationships within the data, which allows them to make predictions based on the learned patterns. A *Recurrent Neural Network* (RNN) is a neural network that loops back to its hidden states, which can also be called memory states, enabling it to process sequential data and retain contextual information, essentially creating a form of memory. The output at each time step is fed back as input for the next, which means the network incorporates past information when processing new data [14].

The most basic architecture of an RNN is made up of only one input layer, a single hidden layer with a recurrent connection, and one output layer. The recurrent connection feeds the hidden state from the previous time step into the hidden layer at the current time step in order to capture the temporal dependencies. An example of such network can be seen to the left in figure 2.1. On the right in the figure, a time-unfolded representation of the recurrent connection is show. This visualization provides a clearer view of the dynamic interactions over time within the network, better illustrating how the system progresses and interacts at each time step [14] [15].

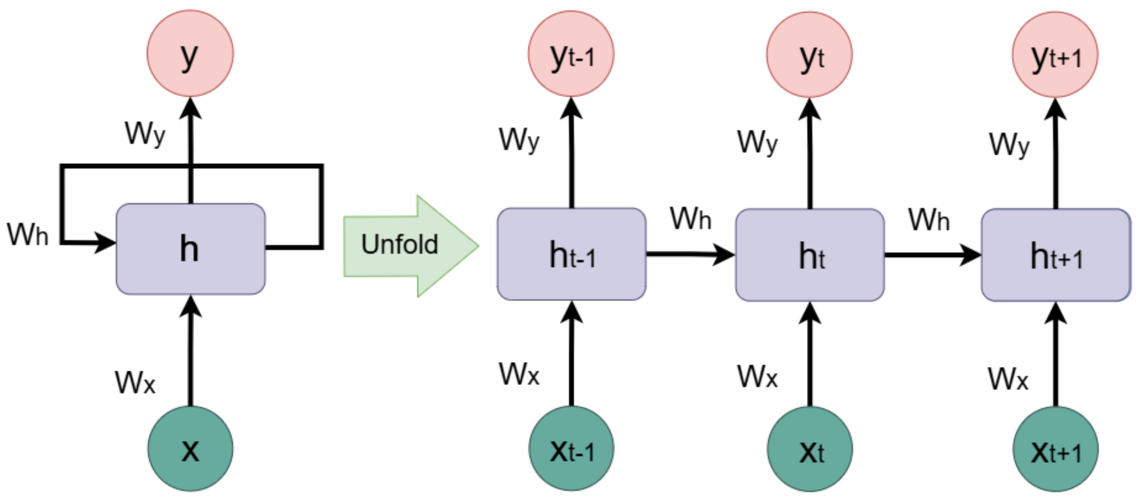


Figure 2.1: Recurrent Neural Network

In the step that is called the forward pass, the input at time step t, x_t, is processed by the RNN and the hidden state h_t is updated based on the current input and the previous hidden state h_{t-1} as following [14] [15]:

$$h_t = \phi_h(W_x x_t + W_h h_{t-1} + b_h) \quad \text{where:}$$

- W_x is the weight matrix that determines the contribution of the input x_t to the hidden state,

- W_h is the weight matrix that determines the contribution of the previous hidden state h_{t-1} to the hidden state,
- b_h is the bias for the hidden state,
- ϕ_h the activation function for the hidden state (varies based on the problem)

The activation function is applied to the output of the hidden layer to introduce non-linearity, helping the model to better learn the underlying structures. The bias is added to shift the activation function, improving the model's ability to adjust to the data. Then, with the updated hidden state the output at time step t, y_t is generated by the equation [14] [15]:

$$y_t = \phi_y(W_y h_t + b_y) \quad \text{where:}$$

- W_y is the weight matrix that determines the contribution of the hidden state h_t to the output,
- b_y is the bias for the output
- ϕ_y the activation function for the output (varies based on the problem)

Moreover, with the calculated output, the weights are updated through backpropagation, which in RNNs is dependent on time and is therefore referred to as Backpropagation Through Time (BPTT). The loss $\mathcal{L}$ is calculated at each time stamp by the difference between the output and the true target [14] [15]:

$$\mathcal{L} = \sum L(y_t, t) \quad \text{where:}$$

- $\mathcal{L}$ is the loss or objective function,
- t is the true target

Which loss function that is used can vary depending on the problem. Afterwards the gradients of the loss with respect to each weight and bias parameter is computed by propagating the error backward through time, starting from the final time step and moving backwards in time. Then, the computed gradients are used to update the weights and biases at each time step, typically by using an optimization algorithm like gradient descent [16].

One issue with RNNs is that during backpropagation through time, the gradient of the loss can either become very small or very large, a problem referred to as the vanishing or exploding gradient problem. The more hidden layers a network has, the more prominent the vanishing gradient problem becomes, as the gradient decreases with each calculation. As a result, the updates for the earlier layers become ineffective and the model fails to learn correctly. The same applies to exploding gradients, but the gradient grow exponentially instead of decreasing. This issue causes RNNs to be difficult to train, resulting in both poor model performance and slow convergence, especially when handling long-term dependencies [14] [17].

2.4 Long Short-Term Memory networks

Long Short-Term Memory (LSTM) networks are a version of RNNs created to address the issues associated with long-term dependencies. It does so by introducing what is called a memory block [18].

2.4.1 Memory Block

A memory block consists of one or more self-connected memory cells and three gates: input, output, and forget. The memory cells store and retain information over time, allowing the network to remember past inputs and make decisions based on that stored knowledge. The gates manage the flow of information and selectively regulates which information should be allowed through, stored, or discarded at each time step. LSTMs are otherwise very similar to regular RNNs, but instead of basic activation units in the hidden layer, they utilize memory blocks to efficiently control and process data over time [18].

Figure 2.2 is a schematic diagram of the memory block and its parts. Here C_t means the state of the memory cell, also called the *cell state*, at time step t, h_t is the hidden state at time step t, x_t is the input at time step t and σ and tanh are the respective sigmoid and hyperbolic tangent activation functions.

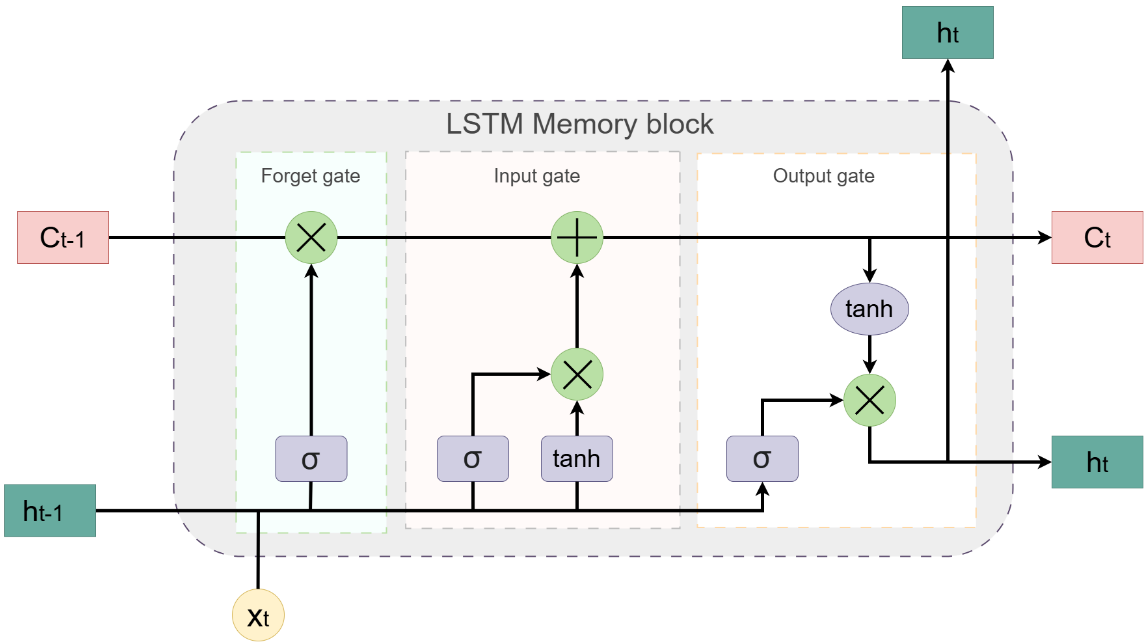


Figure 2.2: LSTM memory block

2.4.1.1 Forget Gate

The forget gate determines which information should be retained and which should be discarded from the cell state. The amount of information retained is determined by the sigmoid function, which produces an output ranging from 0 to 1. A value of 1 results in all the information being retained, and a value of 0 means that all the information will be discarded. Values in between function as a scalar, determining the proportion of information to be retained. The equation for the forget gate is [19][20]:

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \quad \text{where:}$$

- f_t is the forget gate output at time step t,
- W_{fx} is the weight matrix that determines the contribution of the input x_t to the forget gate's activation,
- W_{fh} is the weight matrix that determines the contribution of the previous hidden state h_{t-1} to the forget gate's activation,
- b_f is the bias term for the forget gate

2.4.1.2 Input Gate

The input gate controls whether the memory cell is updated and, if so, the extent of the update. The input gate is calculated by [19][20]:

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \quad \text{where:}$$

- i_t is the input gate output at time step t,
- W_{ix} is the weight matrix that determines the contribution of the input x_t to the input gate's activation,
- W_{ih} is the weight matrix that determines the contribution of the previous hidden state h_{t-1} to the input gate's activation,
- b_i is the bias term for the input gate

In the input gate, the sigmoid function determines the proportion of new information that should be used to update the cell state. The selected information is then combined with *the candidate cell state*, which is the propose potential new memory information derived from the current input and the previous hidden state. The candidate cell state is calculated as follows [19][20]:

$$\tilde{C}_t = \tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c) \quad \text{where:}$$

- $\tilde{C}_t$ is the candidate cell state at time step t,
- W_{cx} is the weight matrix that determines the contribution of the input x_t to the candidate cell state,
- W_{ch} is the weight matrix that determines the contribution of the previous hidden state h_{t-1} to the candidate cell state,
- b_c is the bias term for the candidate cell state

The tanh activation function produces outputs in the range of -1 to 1, which helps prevent the cell state from reaching extreme values. This bounded range stabilizes the memory and reduces the risk of exploding and vanishing gradients during backpropagation. The memory cell is then updated by forgetting a portion of the previous state using the forget gate and adding a new candidate value, influenced by the current input. The equation for this update is [19][20]:

$$C_t = f_tC_{t-1} + i_t\tilde{C}_t \quad \text{where:}$$

- C_t is the updated cell state at time step t,
- C_{t-1} is the previous cell state

2.4.1.3 Output gate

The output gate decides how much of the stored memory should be output based on the current input and previous state. It achieves this by calculating the next hidden state of the memory block, which is then used as the output of the memory block at the current time step. The equation for the output gate is given by [19][20]:

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \quad \text{where:}$$

- o_t is the output gate output at time step t,
- W_{ox} is the weight matrix that determines the contribution of the input x_t to the output gate's activation,
- W_{oh} is the weight matrix that determines the contribution of the previous hidden state h_{t-1} to the output gate's activation,
- b_o is the bias term for the output gate

The sigmoid function here works again like a scaler, deciding how much of the cell state's memory should be output. The hidden state h_t at time step t is then updated with the equation [19][20]:

$$h_t = o_t \tanh(C_t)$$

2.5 Variational autoencoder

Autoencoders are another type of artificial neural networks that learns to replicate its input by first encoding the input data into a lower-dimensional latent representation, then decoding the latent representation back to reconstruct the original input. It consists of two parts, an encoder and a decoder, which perform respective tasks of compressing the input and reconstructing it. Standard autoencoders are primarily used in unsupervised learning, which means that the model is not trained with labeled data. Labeled data provides direct guidance by indicating the correct output for each input, helping the model to learn the mapping between the inputs and their corresponding targets. In unsupervised learning, the model instead learns by identifying patterns and structures within the data itself, without any predefined labels. By discovering underlying features, such as similarities and differences, within the data, and the model can use those features for drawing meaningful conclusions or making decisions based on the inherent structure of the data [21].

Variational autoencoders (VAE) are a type of autoencoders with added probabilistic element by modeling the encoding process as a distribution. The input data is mapped to a latent space where each point represents a probability distribution instead than a fixed vector. This makes the latent space continuous, allowing for smooth interpolation between points. As a result, VAE are also able to generate new data rather than just reconstructing existing inputs, making them a type of generative model. VAEs are primarily used for unsupervised learning, but they have also been shown to be adapted for semi-supervised and supervised learning tasks [22] [23] [24].

2.5.1 Encoder – Decoder

The encoder transforms the input x into a lower-dimensional representation, producing two outputs: the mean and the standard deviation of a latent distribution. These parameters define a probability distribution over the latent variable z, typically modeled as Gaussian. To generate or reconstruct data effectively, the encoder must determine the likelihood of different values of z given the observed input. This requires computing the posterior distribution $p(z|x)$, which represents the probability of z taking specific values after observing x. Computing the posterior distribution directly is difficult because of its complexity, particularly when the latent variable is high-dimensional. Therefore, the encoder instead approximates the posterior distribution by learning a variational distribution $q_\phi(z|x)$. The equation for the variational distribution is [24] [25]:

$$q_\phi(z|x) = \mathcal{N}(z \mid \mu(x), \sigma^2(x)), \quad \text{where:}$$

- $\mu(x)$ is the mean of the latent distribution over z for a given input x,
- $\sigma^2(x)$ is the variance of the latent distribution over z for a given input x,
- $\mathcal{N}(z \mid \mu(x), \sigma^2(x))$ is the Gaussian distribution with mean $\mu(x)$ and variance $\sigma^2(x)$

A z is then sampled using what is called the *reparametrization trick* in the equation:

$$z = \mu(x) + \sigma(x) \cdot \epsilon, \quad \text{where:}$$

- ϵ is a random sample from the standard normal distribution $\mathcal{N}(0, 1)$

In probabilistic models with random variables, backpropagation can be difficult because sampling from probabilistic distributions is non-differentiable. The reparametrization trick adds noise through the variable ϵ, in order to express z as a deterministic function instead of a random variable. This separates the randomness from the learned parameters, making the sampling process differentiable and enabling smooth backpropagation [24] [25].

The decoder then reconstructs the data by mapping the latent representation z back to the original input space, producing a probability distribution over the possible reconstructions. This distribution reflects the likelihood of each possible output x', given the latent variable z. In figure 2.3 the architecture of the VAE is demonstrated.

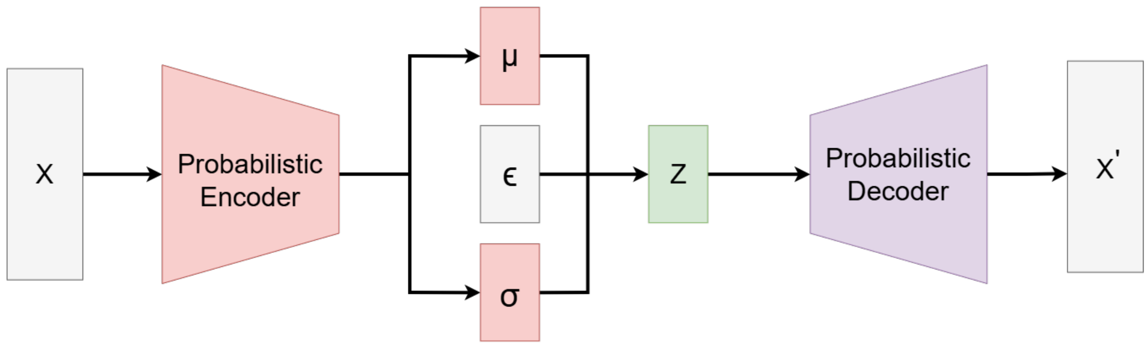


Figure 2.3: Variational autoencoder architecture

2.5.2 Loss function

The reconstructed data from the decoder is then propagated back through the network. VAEs objective is to maximize the Evidence Lower Bound (ELBO) in order to optimize the variational distribution and better approximate the true posterior. It is a lower bound on the data log-likelihood, which assesses the probability of observing the given data under a specific probabilistic model. Maximizing it improves the model's ability to better explain or fit the data. ELBO is composed out of the *reconstruction loss* and the *Kullback-Leibler divergence*. The reconstruction loss is the loss function used for standard autoencoders, measuring the difference between the decoder's output and the original input. The reconstruction loss can be computed using various algorithms, with the choice depending on the type of data and task requirements. When the decoder reconstructs data, it samples from the particular points in the latent space that correspond to the original inputs. For the decoder to be able to generate new data, it needs to be able to sample from any region within the latent space. To enable the decoder to not only reconstruct data, but also successfully generate data, a supplementary regularization term, the Kullback-Leibler divergence or KL divergence, is added. The KL divergence is a measurement of how much one probability distribution differs from another. The prior distribution $p(z)$ is the initial assumption about the latent variable z, before observing any data. In VAEs, the prior is typically chosen as a standard multivariate Gaussian distribution $\mathcal{N}(0, I)$. The aim is to minimize the KL divergence between the variational distribution $q_\phi(z|x)$ and the prior distribution $p(z)$ to create a structured latent space that ensures smooth and consistent representations. ELBO is calculated by [24] [25]:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi(z|x)}[\log p(x|z)] - \text{KL}(q_\phi(z|x)\|p(z)) \quad \text{where:}$$

- $\mathcal{L}_{\text{ELBO}}$ is the objective function,
- $\mathbb{E}_{q_\phi(z|x)}[\log p(x|z)]$ is the reconstruction loss. This term computes the expected log-likelihood of reconstructing the data x from the latent variable z, averaged over the variational distribution $q_\phi(z|x)$,
- $\log p(x|z)$ measures how well the model reconstructs data, where $p(x|z)$ represents the probability of generating the observed data x given the latent variables z,
- $\text{KL}(q_\phi(z|x)\|p(z))$ is the KL divergence between the variational distribution $q_\phi(z|x)$ and the prior distribution $p(z)$

2.5.3 Self-attention

Self-attention is a mechanism that helps models to capture dependencies between different positions in a sequence, allowing it to generate a more meaningful representation. It is commonly used in transformer models but has also been applied in VAE models [26] [27].

Attention is computed as the weighted sum of three vectors: queries (Q), keys (K), and values (V), where the weights are determined by the relationships between the queries and keys. A common form of attention is the scaled dot-product attention, where the input consists of queries and keys, both of dimension d_k, and values of dimension d_V. It is given by the equation [28]:

$$\text{Attention(Q, K, V)} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

The queries and keys are first transformed through linear mappings to calculate their compatibility. These compatibility scores are then used to determine the weights for each value. Multi-head attention is an extension where multiple attention functions (referred to as heads) are computed in parallel, rather than just a single one. This is done by linearly projecting the queries, keys, and values into multiple smaller dimensions. The model can then capture dependencies across different representation spaces and positions within the input. Afterward, the outputs from all attention heads are concatenated and passed through a final linear layer to produce the final output. The equation for Multi-Head Attention is given by [26] [28]:

$$\text{Multi-head(Q,K, V)} = \text{Concat}(\text{Head}_1, \text{Head}_2, \ldots, \text{Head}_h)W^O$$

$$\text{Head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad \text{where:}$$

- W_i^Q is the Query weight matrix for the i-th attention head,
- W_i^K is the key weight matrix for the i-th attention head,
- W_i^V is the Value weight matrix for the i-th attention head,
- W^O is the output weight matrix,
- h is the number of attention heads

3
Method

This section aims to provide an overview of the data handling process and the development of the model as well as reasoning behind model architecture choices made. While Chapter 2 provides a theoretical background and an understanding of the techniques used in this report, this section will primarily focus on the justification for these techniques and their practical implementation. The framework used in this project is PyTorch, an open-source machine learning library, with Python as the programming language [29]. Additionally, Ray Tune, a hyperparameter tuning library, has been utilized to optimize model performance [30]. Furthermore, GPT-4 was utilized as a suggestion tool when writing the thesis, used to overall increase the grammatical degree and provide suggestions for sentence building.

3.1 System

Supervised and semi-supervised methods were considered but deemed unsuitable due to the need for labeled data and the difficulty in obtaining expert knowledge. Given the goal of detecting previously unobserved anomalies, an unsupervised approach was selected.

3.1.1 Dataset

The data used in this project consist of time series data from vessel engine signals which are retrieved from a *SSMS-database* (SQL Server Management Studio). The datasets are structured and given in a tabular format with defined attributes such as vessel, engine, signal names and mean value, recorded at specific timestamps. An example of the raw data format, with dummy values, can be seen in table 3.1. For this project one vessel was chosen and total of 10 signals were selected for analysis.

Table 3.1: Example table illustrating the structure of raw data (dummy values).

Timestamp	Vessel Name	Mean Value	Engine Name.Signal Name
2024-01-01 12:00:00	Vessel X	123.45	Engine A.Signal 1
2024-01-01 12:00:01	Vessel X	125.32	Engine B.Signal 1
2024-01-01 15:23:04	Vessel Y	39.89	Engine A.Signal 2
2024-01-01 15:23:05	Vessel Y	41.12	Engine B.Signal 2
2024-01-01 15:23:06	Vessel Y	305.23	Engine A.Signal 3

During the initial data exploration, an analysis of the time series characteristics, examining aspects such as seasonality, stationarity, correlation, autocorrelation, and cycles was conducted. However, upon further evaluation, it was determined that only correlation and autocorrelation significantly impacted the modeling and the research objectives of the project. The reason for not including the other metrics was that the boat data is mostly independent of seasons, has no evident cycles, only smaller trends and the ADF tests presented in table 3.2 indicated that all signals are stationary. As a result, it was concluded that it was only necessary to focus on correlation and autocorrelation, and the other aspects were excluded from further analysis.

Table 3.2: ADF test results for signal configurations. Note: Signals 1–3 differ between the 3-signal and 7-signal configurations.

3-Signal Configuration		
Signal	**ADF p-value**	**Stationarity**
Signal 1	0.000	Stationary
Signal 2	0.033	Stationary
Signal 3	7.475×10^{-22}	Stationary
7-Signal Configuration		
Signal	**ADF p-value**	**Stationarity**
Signal 1	0.0349	Stationary
Signal 2	1.786×10^{-12}	Stationary
Signal 3	2.169×10^{-20}	Stationary
Signal 4	2.405×10^{-23}	Stationary
Signal 5	0.01	Stationary
Signal 6	8.878×10^{-17}	Stationary
Signal 7	0.003	Stationary

3.1.2 LSTM Variational Autoencoder model

Several models used for anomaly detection were considered, including Transformers, Generative Adversarial Network (GAN), and simpler Autoencoders. Although GAN and Transformer-based models may prove successful in anomaly detection, they are not necessarily the optimal choice for the current system. GANs, although powerful for generating realistic data distributions, may suffer from instability in training leading to extensive hyperparameter tuning and careful model balancing.

Transformer-based models, on the other hand, excel in capturing global dependencies but come with high computational costs, especially when processing longer sequences which was required during training to capture larger behavioral patterns. Autoencoders can be effective for anomaly detection by highlighting outliers through reconstruction error, though variational autoencoders are more powerful due to their probabilistic framework, which better models uncertainty and outliers.

Considering this, along with numerous studies highlighting the efficiency and accuracy of LSTM-based encoder-decoders, this approach was deemed appropriate. Additionally, incorporating a self-attention mechanism, known for its effectiveness in handling multivariate sequential time series, ensures that the model's architecture is well-suited to the system's requirements. Self-attention offers a balance between effective representation learning, sequential data handling, and improved attention to relevant features over time. The models architecture can be seen below in figure 3.1 which illustrates a simpler overlay of the components of the autoencoder.[5]

Input Time-Series Data

$\downarrow$

Encoder:

- LSTM Layer 1: Extracts temporal features.
- LSTM Layer 2: Refines temporal representations.
- Dense Layers: Outputs z_{mean} and $z_{\text{log\_var}}$.

$\downarrow$

Latent Space Sampling:

$$z = z_{\text{mean}} + \exp(0.5 \cdot z_{\text{log\_var}}) \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1)$$

$\downarrow$

Decoder:

- LSTM Layer 1: Processes latent variables.
- Self-Attention Mechanism: Enhances temporal focus.
- LSTM Layer 2: Refines intermediate representations.
- Dense Layer: Reconstructs the original signals.

$\downarrow$

Reconstructed Data

$\downarrow$

Reconstruction Error Calculation:

$$\mathcal{L}_{\text{recon}} = \frac{1}{N} \sum_{i=1}^{N} \|x_i - \hat{x}_i\|^2$$

$\downarrow$

Anomaly Detection:

Data points with high reconstruction error are flagged as anomalies.

Figure 3.1: Architecture of the Variational Autoencoder (VAE) Model.

The LSTM layers handle the temporal dynamics of the input data by extracting and refining key temporal features, as illustrated in figure 3.1. Specifically, the encoder consists of two LSTM layers: the first extracts temporal features, while the second refines these representations. Dense layers then output the latent variables, z_{mean} and $z_{\text{log\_var}}$, which are sampled to form the latent space representation. The VAE framework leverages this latent space to learn representations of normal behavior, enabling efficient detection of deviations that signify anomalies.

The decoder processes the latent variables through an optional attention mechanism, enhancing temporal focus, followed by additional LSTM layers and dense layers to reconstruct the original signals. Reconstruction errors are calculated as:

$$\mathcal{L}_{\text{recon}} = \frac{1}{N} \sum_{i=1}^{N} \|x_i - \hat{x}_i\|^2,$$

where data points with high deviations are flagged as anomalies. The reconstruction error measures how well the model can recreate the input, while the overall loss optimized during training combines this reconstruction error with a KL divergence term to regularize the latent space. The integration of self-attention further enhances the model's ability to focus on relevant sequence elements and capture long-range dependencies, ensuring a resource-efficient and timely detection of faulty signals.

3.2 Model development and data flow

3.2.1 Data Preprocessing

The data preprocessing pipeline begins by filtering the raw data based on the engine name, signals, and the mean value of these signals in the SSMS database. Afterward, the data is split into training and test sets, and normalization is applied separately by scaling each signal to the range [0,1], based on its minimum and maximum values. Normalization was applied before most other preprocessing steps, so outliers in the training data were included in the normalization calculations. This normalization process ensures that all signals have comparable ranges, which improves model convergence during training. Normalization is important in time series analysis as it preserves the correlations between signals, which are essential for maintaining the structural relationships required for effective anomaly detection. Correlation, using Pearson's method, captures the strength and direction of relationships between variables, as can be seen later in figure 3.3.

To handle time-series data, the datasets are divided into sequences. Each sequence represents a fixed number of consecutive timestamps, determined by a sequence length parameter (e.g., 30 timestamps). A sliding window approach is used to create overlapping sequences, with a configurable step size to control the degree of overlap. This approach captures temporal dependencies and provides context for each observation.

Although the dataset is complete, there are instances where only some sensors produce outputs, resulting in missing values when the signals are aligned on a timeline. The missing values, represented as `NaN`, are replaced using a forward-fill and backward-fill approach, followed by interpolation to compute the mean of neighboring non-`NaN` values. This ensures that gaps in the data do not disrupt model training or cause inference. For time-series data, this interpolation is performed within each sequence to maintain temporal consistency, aligning with the sequential nature of time series data.

Outliers are a specific type of anomaly, meaning extreme individual data points. Anomalies also include more complex deviations such as unusual temporal or correlational patterns, which were deemed more important and interesting to examine. Therefore, initial outlier detection and smoothing were applied to the training data to reduce the impact of these extreme points and ensure the model is not trained on obvious outliers.

Outlier detection, smoothing, and robust scaling are applied to the training data to prevent the model from training on anomalies. Outliers are detected using a combination of the z-score and interquartile range (IQR) methods. The z-score measures how far a data point deviates from the mean in terms of standard deviations, while the IQR identifies outliers based on the range of the middle 50% of the data. Additionally, percentile clipping is applied to remove extreme values outside a specified percentile range (e.g., 1st to 99th percentile).

Smoothing is performed using a moving average with a configurable window size (e.g., 5 timestamps), which reduces noise while preserving underlying trends in the data. For signals with high variability, an exponential moving average is used to give more weight to recent observations. After outlier handling and smoothing, robust scaling is applied to further reduce the influence of extreme values, ensuring a more stable data distribution, particularly for signals with heavy-tailed distributions. Furthermore, compression techniques are applied to both train and test data for dimensionality reduction and data aggregation. This is applied to optimize the data for storage and processing. For example, signals may be aggregated over fixed time intervals to reduce the dataset size while retaining essential information.

Before preprocessing, the raw data also undergoes validation to ensure quality and consistency. This includes checking for duplicate timestamps, verifying consistent sampling rates, and ensuring that all required signals are present. After preprocessing, additional checks are performed to confirm that no `NaN` or infinite values remain in the dataset. The preprocessing pipeline is implemented in a distributed environment using Apache Spark, enabling it to handle large datasets efficiently. The final result, shown in table 3.3, presents an example of a fully preprocessed data sample with dummy values.

Table 3.3: Example table illustrating the structure of the preprocessed data (dummy values).

Timestamp	Signal 1	Signal 2	Signal 3	Signal 4	Signal 5
2024-01-01 12:00:00	0.25	0.78	0.63	0.12	0.55
2024-01-01 12:00:01	0.30	0.81	0.60	0.15	0.50
2024-01-01 12:00:02	0.28	0.79	0.65	0.11	0.52
2024-01-01 12:00:03	0.31	0.80	0.64	0.14	0.53
2024-01-01 12:00:04	0.29	0.77	0.62	0.13	0.54

3.2.2 Model Training

The training begins by splitting the input data into training and validation sets. The training and validation data are then loaded into PyTorch DataLoader objects, which handle batching and shuffling of the data. The VAE model is trained using the Adam optimizer, with gradient clipping applied to prevent exploding gradients. Each epoch consists of a training and validation phase. During the training phase, the model processes batches of data, computes the forward pass, and calculates the total loss, which is a weighted sum of the reconstruction loss and the KL (Kullback-Leibler) divergence loss. The reconstruction loss measures the difference between the input and the reconstructed output, while the KL divergence loss regularizes the latent space to approximate a standard normal distribution.

3.2.3 Anomaly Detection

Following the training phase, the system transitions into the anomaly detection stage, where the Variational Autoencoder (VAE) encodes input data into a compact latent representation and reconstructs it back into the original data space. Anomalies are identified by calculating the reconstruction error, measured as the mean squared error (MSE) between the original input and its reconstructed output. Higher reconstruction errors signify potential anomalies, as they indicate instances where the VAE struggles to accurately recreate the data. By analyzing these errors over time and across signals, the system captures both individual signal issues and interactions between signals, effectively detecting irregularities while considering temporal correlations and relationships within the sequence.

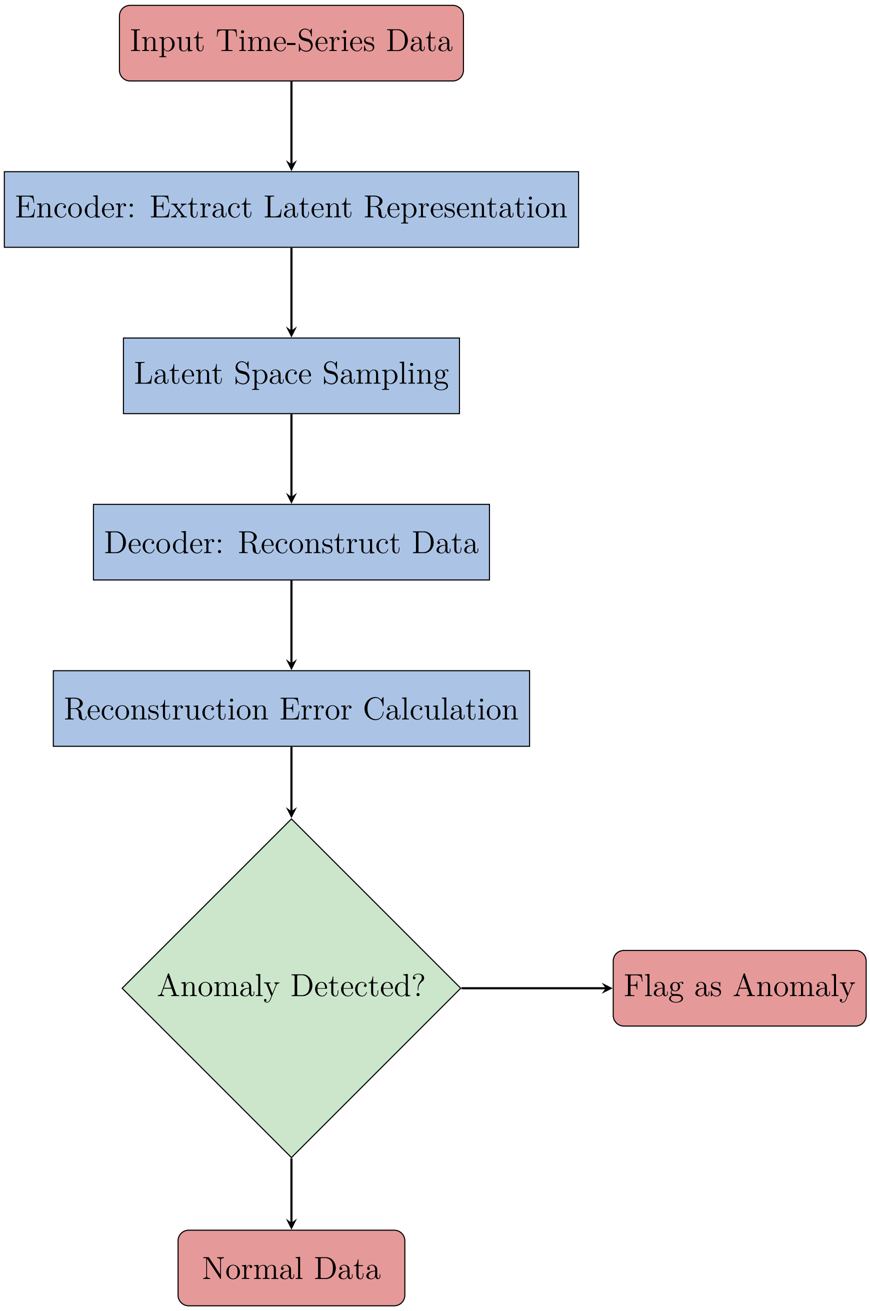


Figure 3.2: Pipeline of the Variational Autoencoder (VAE) Model for Anomaly Detection.

The process of anomaly detection using the Variational Autoencoder (VAE) model is illustrated in figure 3.2. The pipeline begins with the input time-series data, which is passed through the encoder to extract a compressed latent representation. The latent space sampling step ensures that the latent variables are sampled in a differentiable manner, enabling optimization during training. The decoder then reconstructs the data back to its original space. The reconstruction error is calculated as the difference between the original input and the reconstructed output. Finally, a decision is made based on the reconstruction error: if the error exceeds a predefined threshold, the data point is flagged as an anomaly; otherwise, it is classified as normal data. This pipeline effectively captures both temporal dependencies and multivariate relationships, making it well-suited for anomaly detection in time-series data.

Deviations in the latent space are quantified using the Mahalanobis distance, which measures how far a data point deviates from the learned latent distribution. This approach is particularly effective for multivariate time series data, as it accounts for correlations between variables. The correlations between the signals are substantial to the model in utilizing all available information. This can be seen in figure 3.3, which shows the Pearson correlations for seven trained signals. The signals demonstrate strong correlations with one another, providing support for the importance of incorporating these correlations in the analysis. The seven signals show only positive correlations with each other, resulting in correlation values ranging from 0 to 1 in the current analysis. It is important to note that the correlation coefficient ranges from -1 to +1; however, since no negative correlations were detected in this dataset, the lower bound of the scale is not represented in the plot.

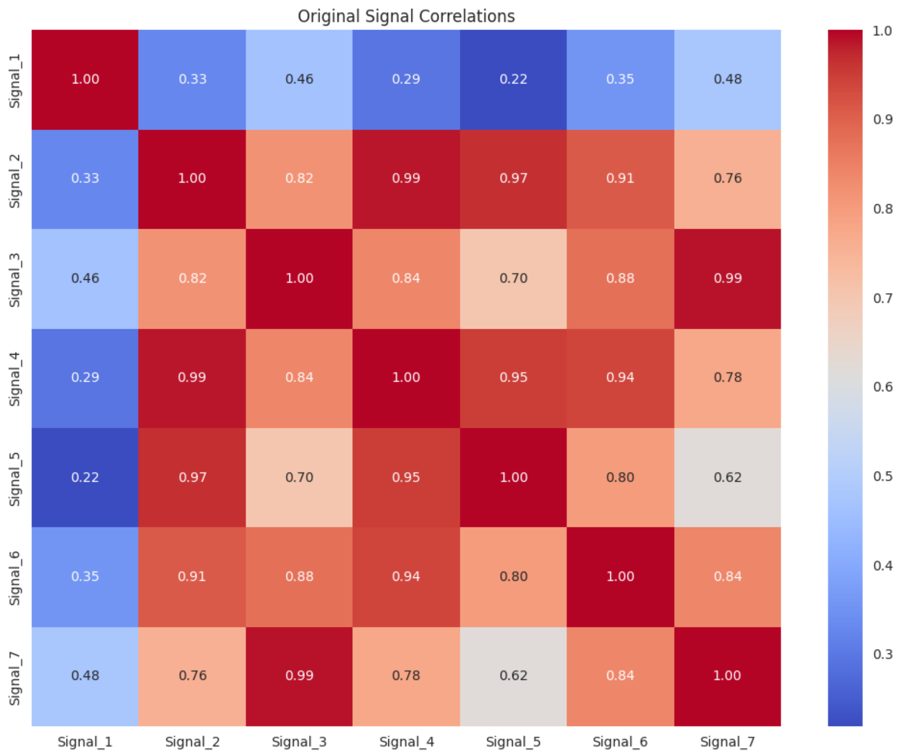


Figure 3.3: A correlation matrix illustrating the Pearson correlations between the seven signals used in one of the trained model runs.

3.2.4 Hyperparameter Tuning with Ray

To optimize the performance of the VAE model, Ray Tune is utilized, a scalable and flexible hyperparameter tuning library. Ray Tune allows for efficient exploration of a wide range of hyperparameter configurations while leveraging distributed computing resources.

The Asynchronous Successive Halving Algorithm (ASHA) scheduler is used to terminate poorly performing trials early and allocate more resources to promising configurations, thereby reducing the overall computation time. The results of the tuning process are tracked and stored, with the best-performing configuration identified based on the lowest validation loss. This configuration is saved for future use, along with a detailed history of all trials, including their hyperparameters and corresponding losses.

3.3 Evaluation Process

3.3.1 Introducing synthetic anomalies

Anomalies were iteratively removed in three stages by detection and removal phases. After cleaning the data, synthetic anomalies were introduced to test model robustness. These were then detected and removed in two additional passes, following the same threshold-based approach. This iterative process ensured improved data quality and validated the model's anomaly detection capabilities.

3.3.2 Anomaly Correction

The purpose of anomaly correction is to enhance the understanding of the anomaly detection and to highlight a potential direction for further development using the model. The process leverages the VAE model's ability to reconstruct input data and analyze its latent space representation. For each detected anomaly, the correction process involves replacing the anomalous data point with the corresponding reconstructed value generated by the VAE model. This ensures that the corrected data adheres to the learned patterns of the system while maintaining temporal consistency.

In cases where anomalies persist or the reconstruction is insufficient, additional techniques such as forward-filling and moving average smoothing are applied to maintain temporal consistency in the corrected data. Furthermore, signal-specific thresholds are dynamically calculated to identify and correct anomalies on a per-signal basis, allowing for adjustments tailored to the characteristics of each signal.

For domain-specific applications, physical consistency checks are performed to ensure the corrected data adheres to real-world constraints. For example, in vehicle data, corrections may be validated to ensure that speed and position changes are consistent with physical laws. By combining reconstruction-based corrections, latent space analysis, and domain-specific validation, the anomaly correction process ensures that the corrected data may be more accurate and consistent with the underlying patterns of the system.

3.3.3 Model comparison

To evaluate the performance of the proposed LSTM-VAE model for unsupervised anomaly detection, a comparative analysis against eight widely used baseline methods was conducted. These methods are *k-Nearest Neighbors (KNN)*, *Local Outlier Factor (LOF)*, *DBSCAN*, *Isolation Forest*, *One-Class SVM*, *K-Means*, and *ARIMA* ([31] [32] [33] [34] [35] [36] [37]). To effectively apply these models, dimensionality reduction was necessary due to the high dimensionality of the data. *t-distributed Stochastic Neighbor Embedding (t-SNE)* was chosen for this purpose [38]. Given the fully unsupervised setting and absence of ground truth labels, all models were evaluated using internal metrics commonly applied in anomaly detection, such as anomaly scores and clustering-based consistency. In certain cases, qualitative analysis was also conducted through visual inspection of the detected anomalies and their temporal patterns.

4

Results

This chapter provides an in-depth exploration of the results produced by the anomaly detection model, showcasing its ability to identify anomalies effectively. The chapter is structured to present results for datasets with three and seven signals, followed by evaluations using reconstruction errors, synthetic anomalies, and comparisons with baseline models. The aim is to provide not only a representation of the model's performance but also an understanding of the reason behind the chosen evaluation metrics and methodologies.

4.1 Model output

The model was trained on two distinct datasets containing three and seven signals, respectively. The first dataset includes signals labeled as signal 1, signal 2, and signal 3, while the second dataset comprises signals labeled as signal 1 through signal 7. To avoid confusion, it is important to note that the labels signal 1, signal 2, and signal 3 in the second dataset refer to different signals than those in the first dataset. In all figures, the x-axis represents time and the y-axis indicates the signal values, illustrating how each signal evolves over time. Across all plots, anomalies are marked with red dots along the signal curve. These red dots highlight points where the signal deviates significantly from its expected pattern, including outliers such as sudden spikes or drops, as well as contextual and group anomalies that arise from temporal dependencies and correlations across signals. Figure (4.11) and (4.10) contain the distribution for the plots includes the original signal and the corrected signal. The corrected signal is where anomalies or outliers have been replaced with the model's reconstructed data. These represent what the model estimated the values to be but lack sufficient confidence to be considered reliable. They should be considered more as additional insights into the model's behavior and performance rather than definitive results.

This section illustrates the output of the model for these datasets, highlighting its ability to detect anomalies. The results presented here are complemented by additional visualizations in Appendix B. Beyond merely illustrating the outputs, this section also discusses the evaluation methods employed to validate the model's reliability and effectiveness. It is crucial to note that the nature of the signals and their correlations vary between the three-signal and seven-signal configurations, which is reflected in the analysis.

4.1.1 Three signals

Figures 4.1 and 4.2 each contain three subplots, with each subplot representing a different signal. In figure 4.1, at the center bottom of all three figures, a anomalies can bee seen in all three signals, indicating that the overall timestamp is flagged as anomalous. Additionally, in signal 3 in the top left, a larger sequence anomaly is visible, indicating that while the other two signals follow a similar pattern, signal 3 deviates significantly at this point.

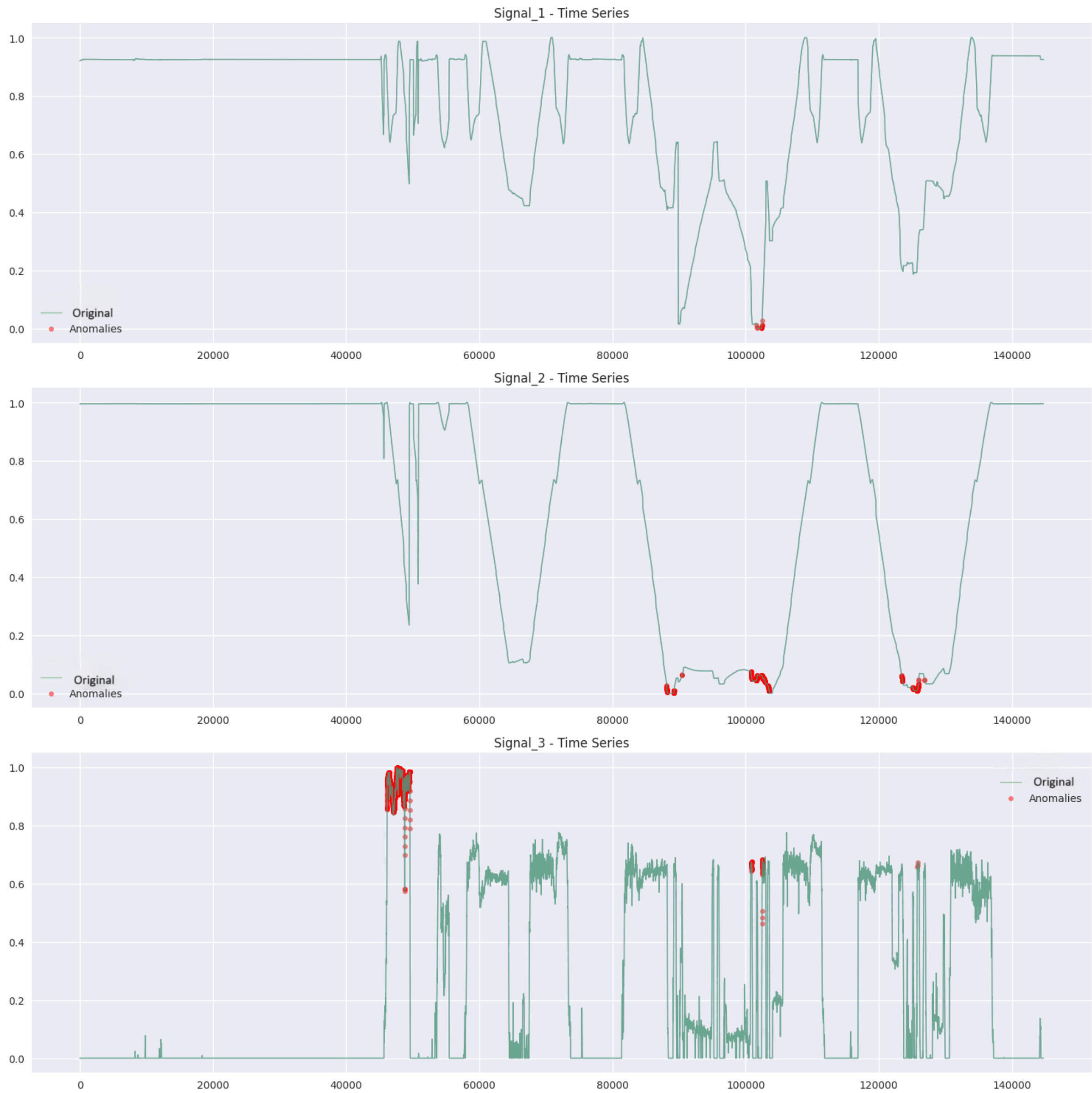


Figure 4.1: Original Data with Detected Anomalies for three signals

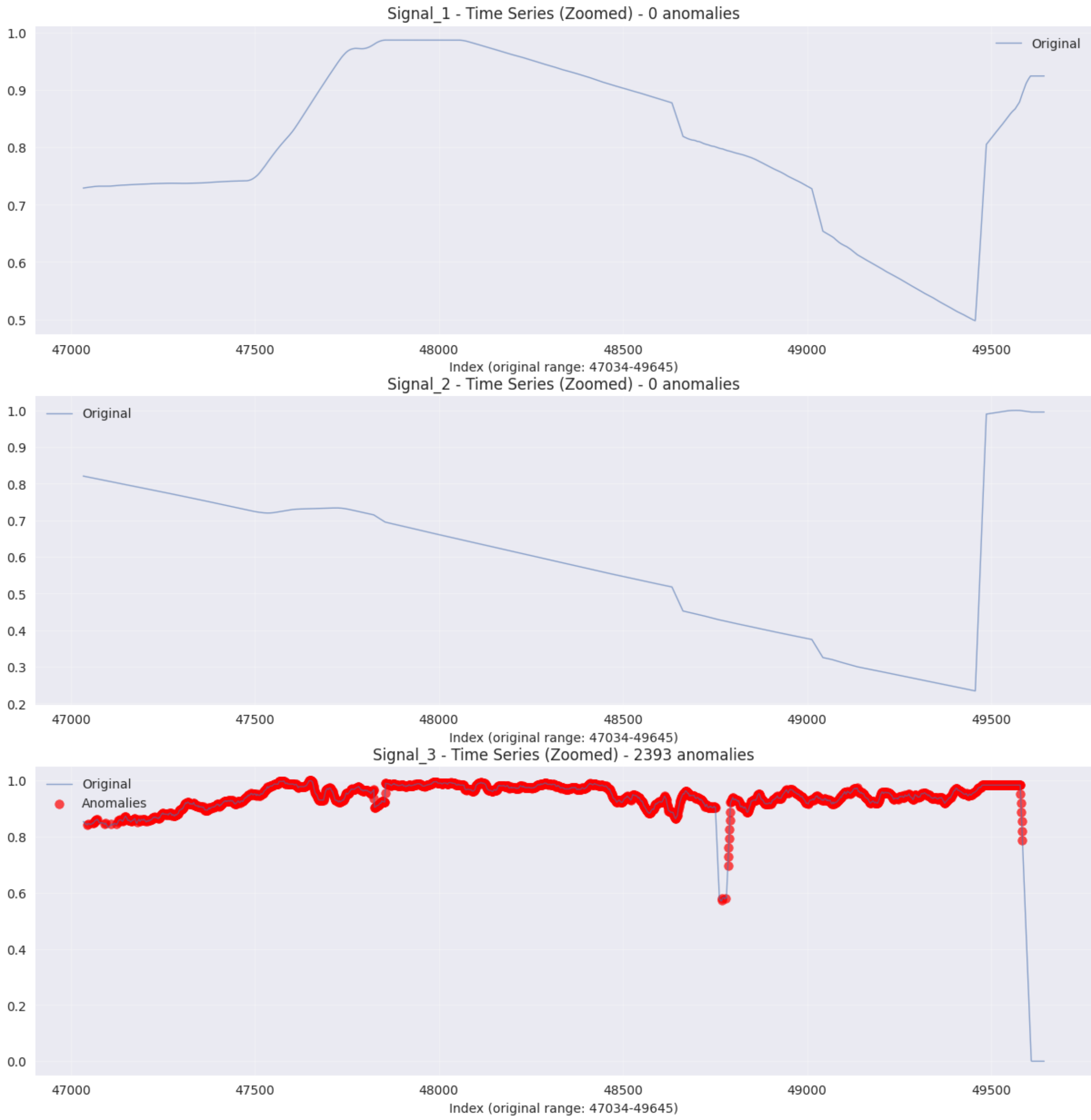


Figure 4.2: Zoomed-in view of an anomaly-dense section

Figures 4.2 and 4.3 present zoomed-in views of the signals, providing a closer look at the anomaly behavior. In the bottom subplot of figure 4.2, a distinct sequence of anomalies can be observed.

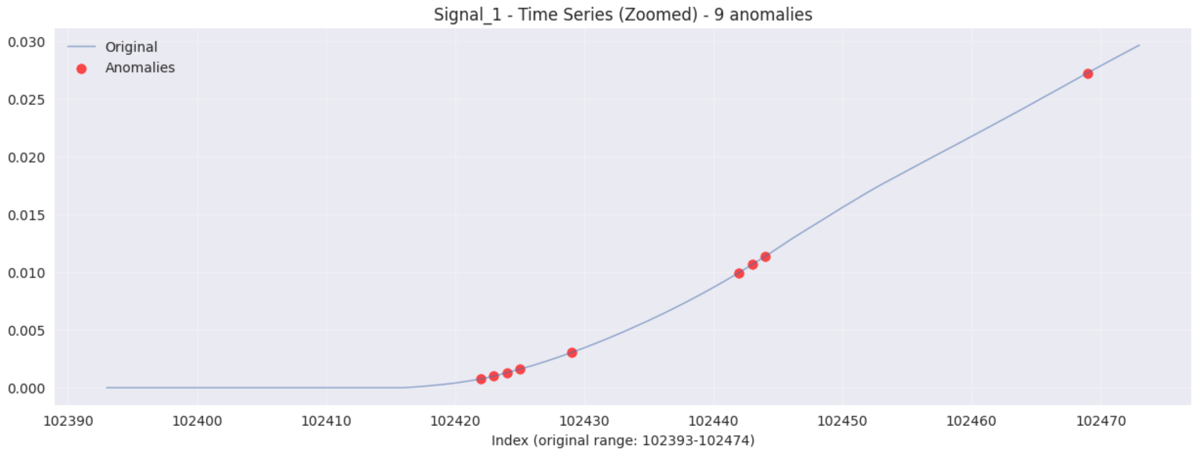


Figure 4.3: Zoomed-in view of a section of the signal

Figure 4.4 illustrates the correlation between signal 1, signal 2, and signal 3. This plot shows the relationships and dependencies among the three signals and how variations in one signal can impact the others. The plot shows a strong positive correlation between signal 1 and signal 2, while signal 3 has a moderate negative correlation with both. This type of analysis is vital for understanding how anomalies in one signal might influence or relate to anomalies in others, thereby providing a more comprehensive evaluation of the dataset.

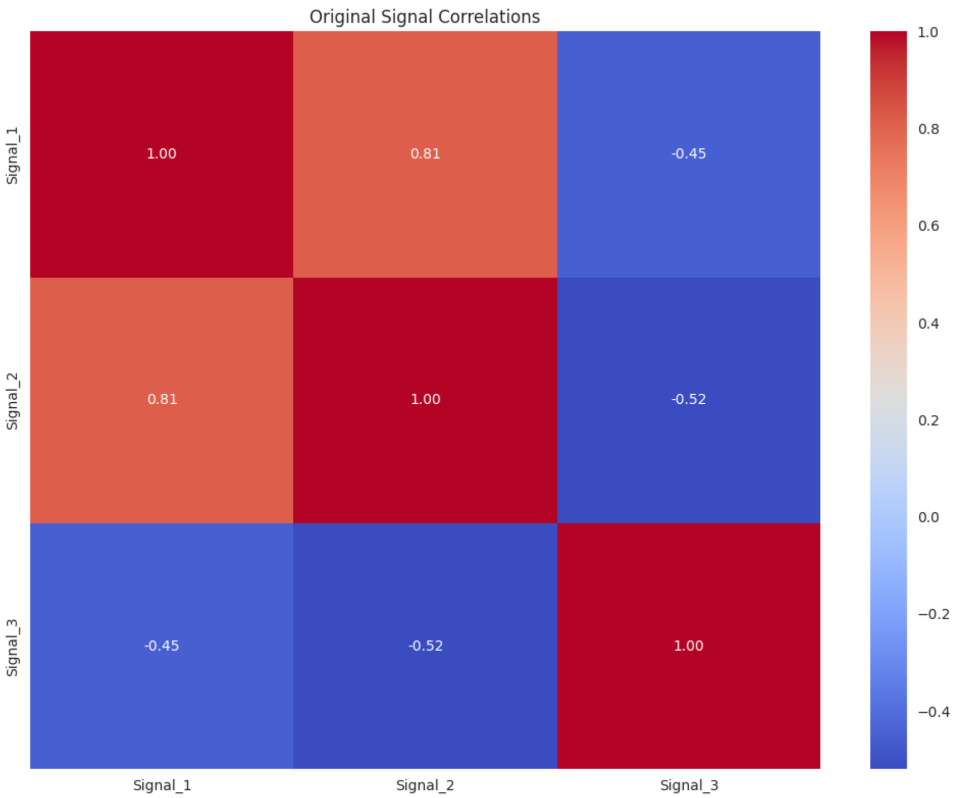


Figure 4.4: Pearsons correlation between signal 1, signal 2 and signal 3

4.1.2 Seven signals

Figure 4.6 and 4.7 follows the configuration of figure 4.1, but instead of three subplots, it includes seven subplots, each representing a different signal. It is important to note that the three-signal dataset and the seven-signal dataset together represent 10 distinct signals, as previously described. As in figure 4.1, the x-axis represents time and the y-axis shows signal values, with anomalies marked by red dots. In the top subplot of figure 4.6 anomalies are spread out over the entire signal, not just concentrated in one area. In the other six subplots, the highest number of anomalies appears at the far right end of the plot, with a few others scattered throughout. A noticeable drop is visible in subplots two, four, and five, where the anomalies occur.

Anomalies are identified and marked across all seven signals, with some signals exhibiting anomalies spread throughout their entirety, while others show concentrated anomaly clusters. This difference helps highlighting the model's adaptability to varying signal behaviors. Additionally, further visualizations can be found in Appendix B, which may provide further insight of how anomalies are distributed across different signals and sequences.

Figure 4.5 shows a zoomed-in section of signal 6, where a continuous region is flagged as anomalous. The anomalies appear grouped along a smooth portion of the signal, rather than as isolated outliers. This may illustrate that the model can detect subtle, sustained deviations in patterns, in comparison to outliers which can be detectable by the human eye as more obvious spikes in the data.

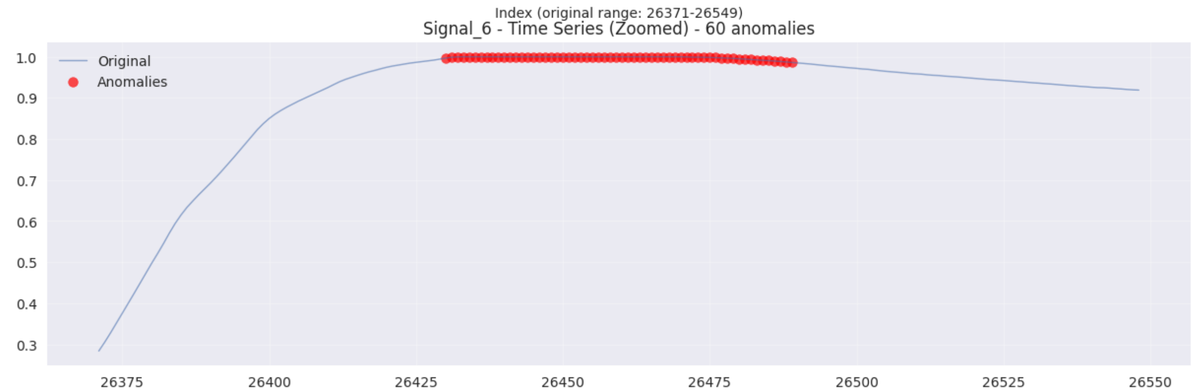


Figure 4.5: A zoomed in graph of signal 6, which is centralized around the area with anomalies, illustrating a grouping of on a smooth line.

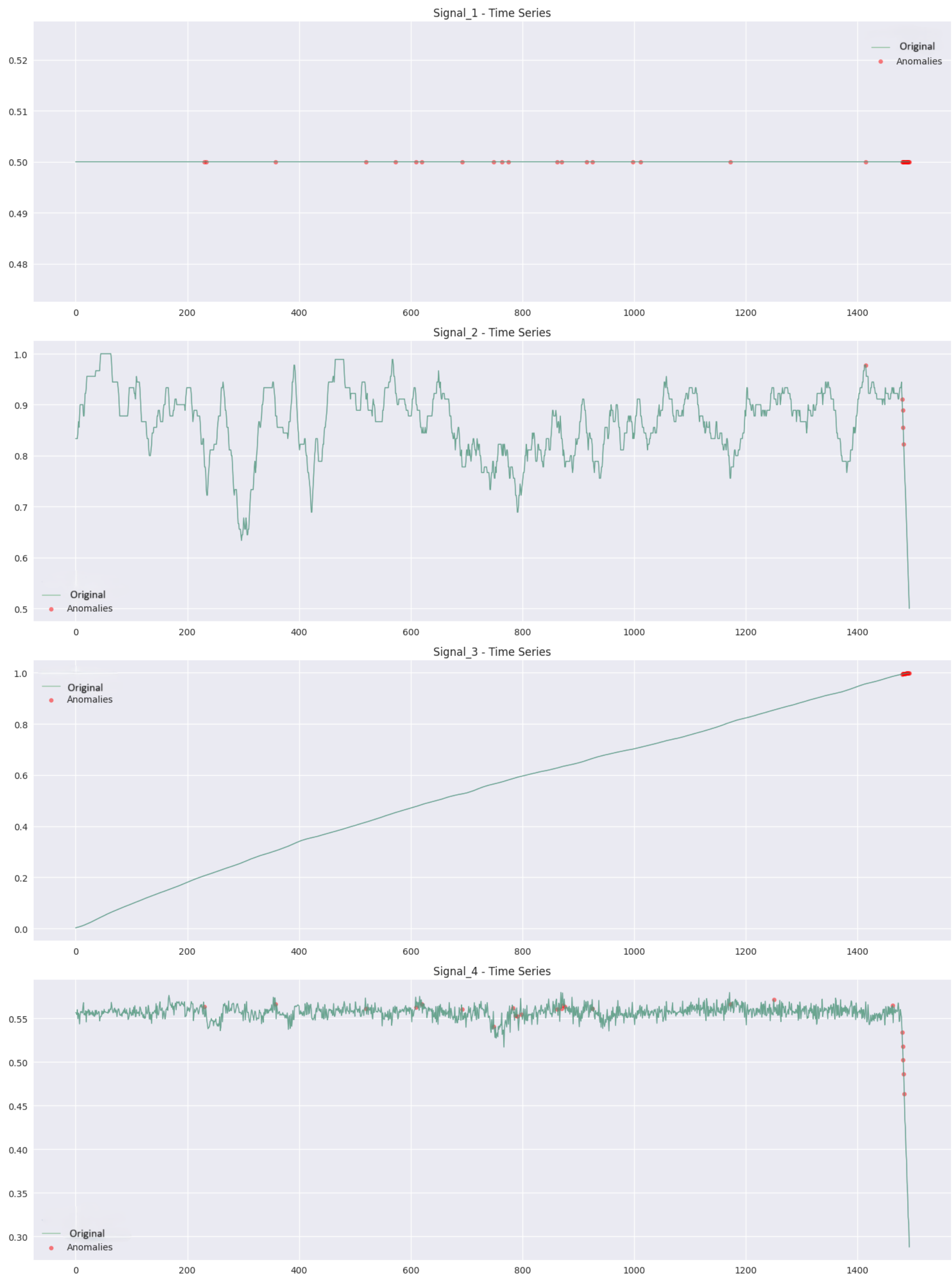


Figure 4.6: The first 4 signals for the anomaly detection of 7 signals with original data included.

Figure 4.7: The last 3 signals for the anomaly detection of 7 signals with original data included.

4.2 Evaluation

The evaluation section seeks to provide reliability to the models results with assessments of the anomaly detection for the model's performance. By analyzing reconstruction errors, synthetic anomalies, and comparing with statistical methods, this section aims to provide a comprehensive understanding of the model's strengths and limitations.

4.2.1 Reconstruction Error

The anomaly detection process consists of calculating reconstruction errors, which measure the difference between the original input and the reconstructed output. These errors are used to identify anomalies, with thresholds dynamically determined based on statistical properties such as the mean and standard deviation of the errors. Additionally, the latent space is analyzed to detect deviations from the learned distribution, using metrics like the Mahalanobis distance to quantify how far a data point lies from the expected latent representation.

Figure 4.8 and 4.9 contains two subplots each. The upper subplots visualizes how reconstruction error varies over time and identifies anomalies based on the threshold θ. The x-axis represents time, while the y-axis shows the error magnitude, which measures the difference between original data and its reconstruction. The horizontal line across the plot represents the threshold θ. Any point where the reconstruction error rises above this threshold is considered an anomaly and is marked with a red dot. The lower subplots shows the distribution of reconstruction errors, with the error magnitude plotted on the x-axis and the density of occurrences on the y-axis. This plot visualizes how frequently different error magnitudes occur, evaluating how accurately the model reconstructs the data. The threshold is marked by a red dotted line, and the region beyond the threshold is shaded in red, indicating the anomaly region where significant reconstruction errors occur. In the top subplot in 4.8 the error magnitude stays below the threshold, indicating that the model's reconstructions are within the acceptable range and consistent with the original data. The bottom subplot shows a primary normal distribution with a secondary smaller one, suggesting the presence of two distinct groups. Most errors are small, but a portion of the errors show larger anomalies. In figure 4.9 the top subplot, the error magnitude remains mostly below the threshold, but there are some anomalies that exceed the threshold. The bottom subplot show that the error distribution follows a normal distribution, with most errors small and centered around the mean, indicating generally accurate reconstructions. However, the anomaly region is relatively large with a long distribution tail.

Figure 4.8: Reconstruction Error with Anomaly Threshold θ and Detected Anomalies

Figure 4.9: Reconstruction Error with Anomaly Threshold θ and Detected Anomalies

Figure 4.10 and 4.10 shows the distribution of the original data and the corrected data for seven signals. The two distributions mostly overlap, suggesting that the model has reconstructed the data accurately. Furthermore, sections where they don't overlap indicate areas where the model might flag anomalies, depending on the reconstruction error at these points.

Figure 4.10: The first 4 signals of the comparison of original and corrected data distribution.

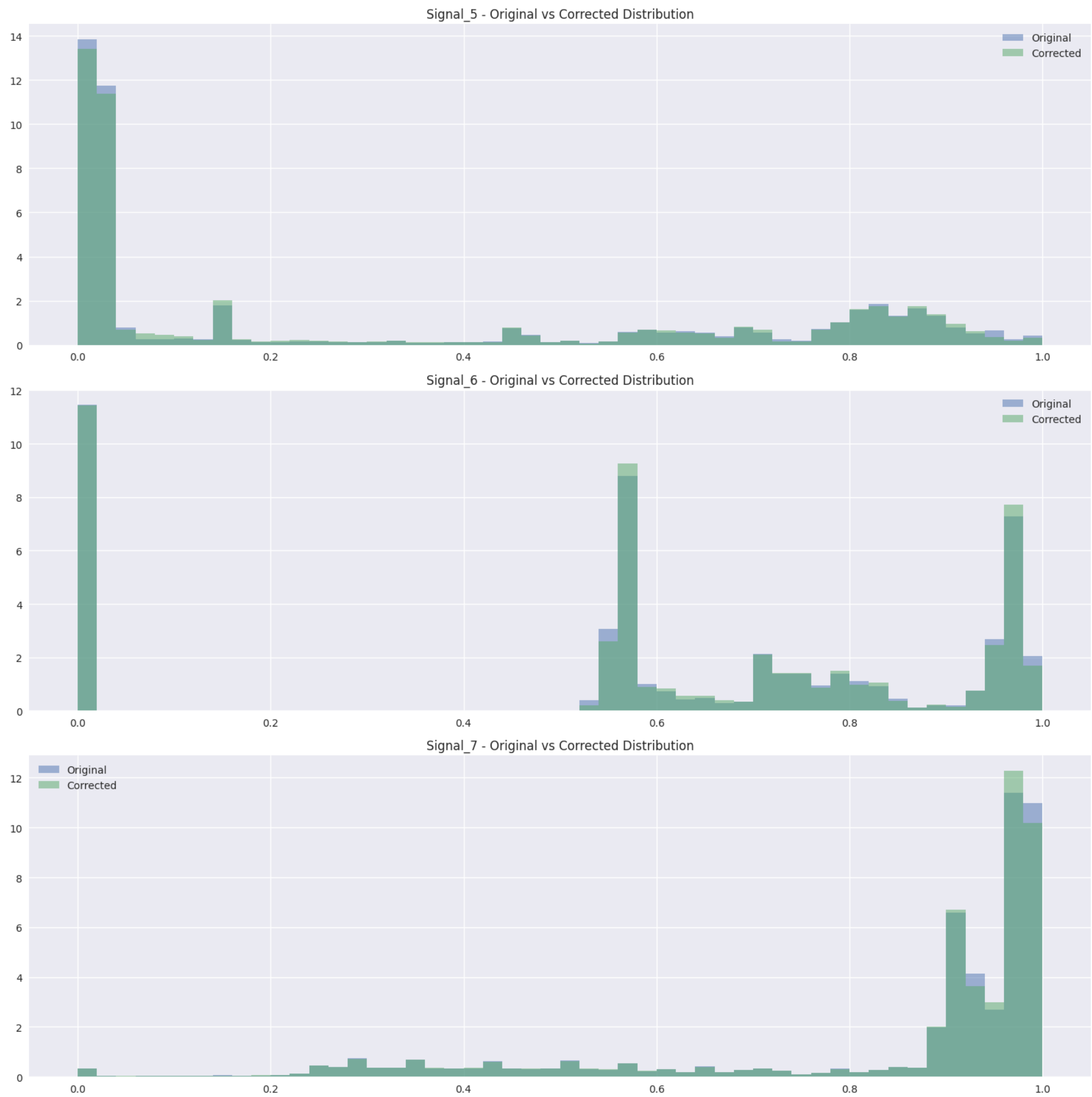


Figure 4.11: The last 3 signals of the comparison of original and corrected data distribution.

4.2.2 Synthetic Anomalies

The use of synthetic anomalies serves to test the robustness and adaptability of the anomaly detection model. The synthetic anomalies are artificially introduced into the dataset to simulate realistic scenarios where abnormal data points may occur. This approach allows for further insight and evaluation of the model's ability to detect anomalies that were deliberately designed to challenge its detection capabilities. By iteratively inserting and removing anomalies, the methodology ensures a comprehensive assessment of the model's strengths and limitations in handling diverse anomaly patterns.

The figures below show five stages from the iterative removal and synthetical adding of anomalies, each of which is further divided into three smaller subplots, each representing a different signal. The figure shows the results of anomaly detection performed by the model on the original data. The second then displays the output of the model after removing the anomalies found in the first run. The third shows the model's output from the third round of anomaly detection, this time applied to the dataset with synthetic anomalies inserted. The remaining two subplots show the fourth and fifth stages, where anomaly detection was repeated after progressively removing detected anomalies in each iteration.

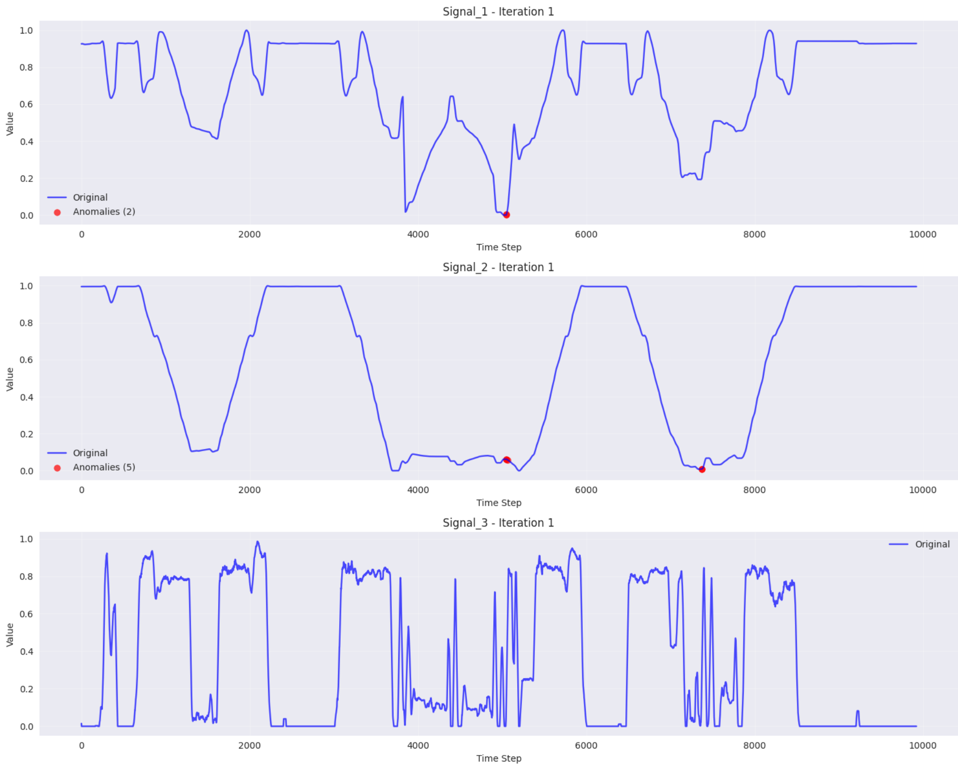


Figure 4.12: Anomaly detection on original data.

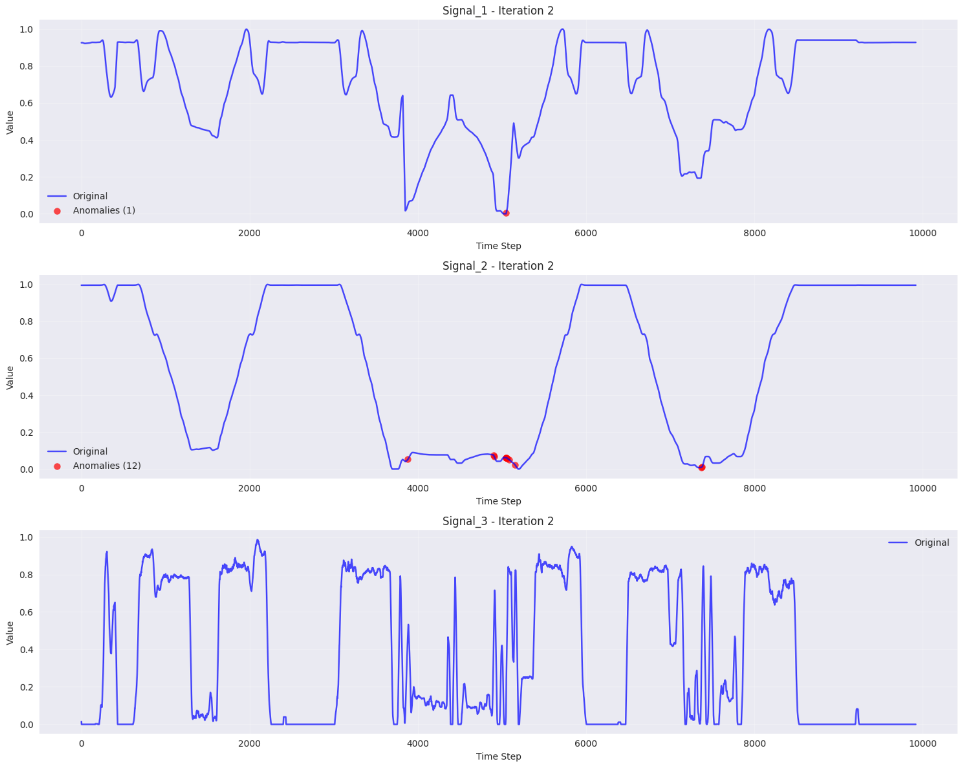


Figure 4.13: Anomaly detection after removing the anomalies found in the original data.

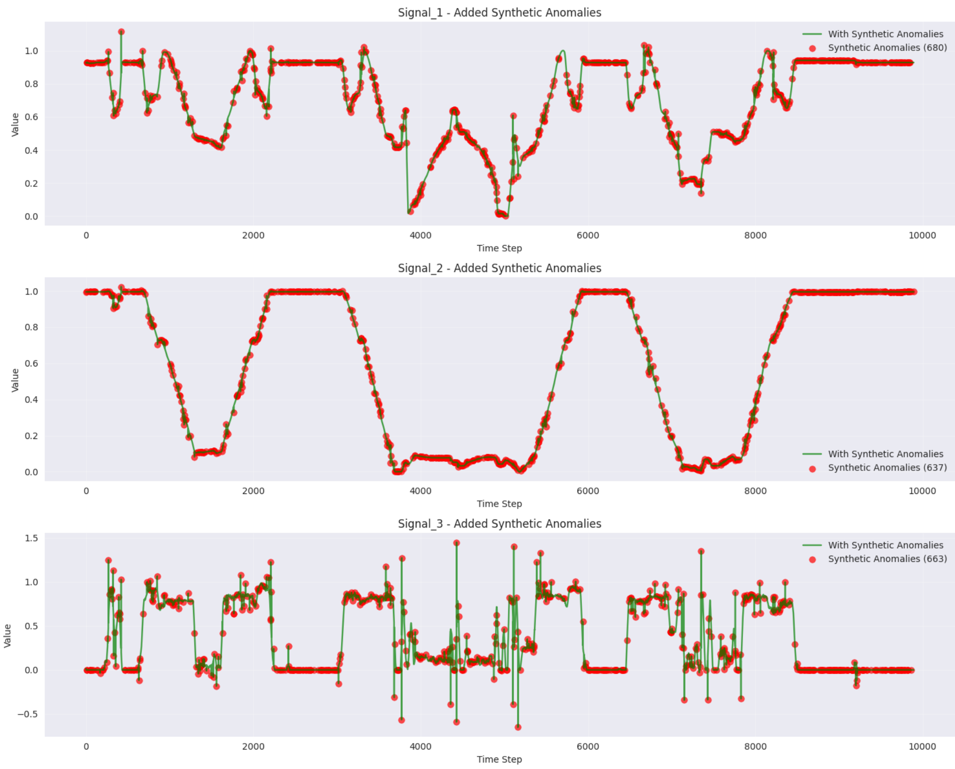


Figure 4.14: Synthetic anomalies inserted into the dataset for evaluation.

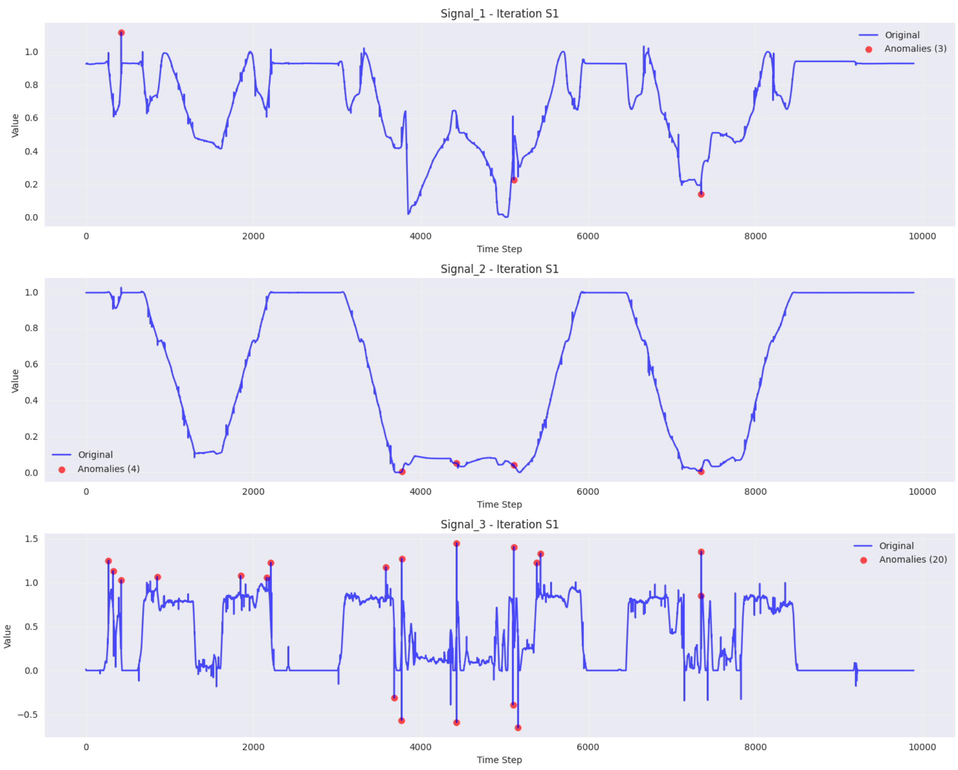


Figure 4.15: Anomaly detection after removing synthetic anomalies.

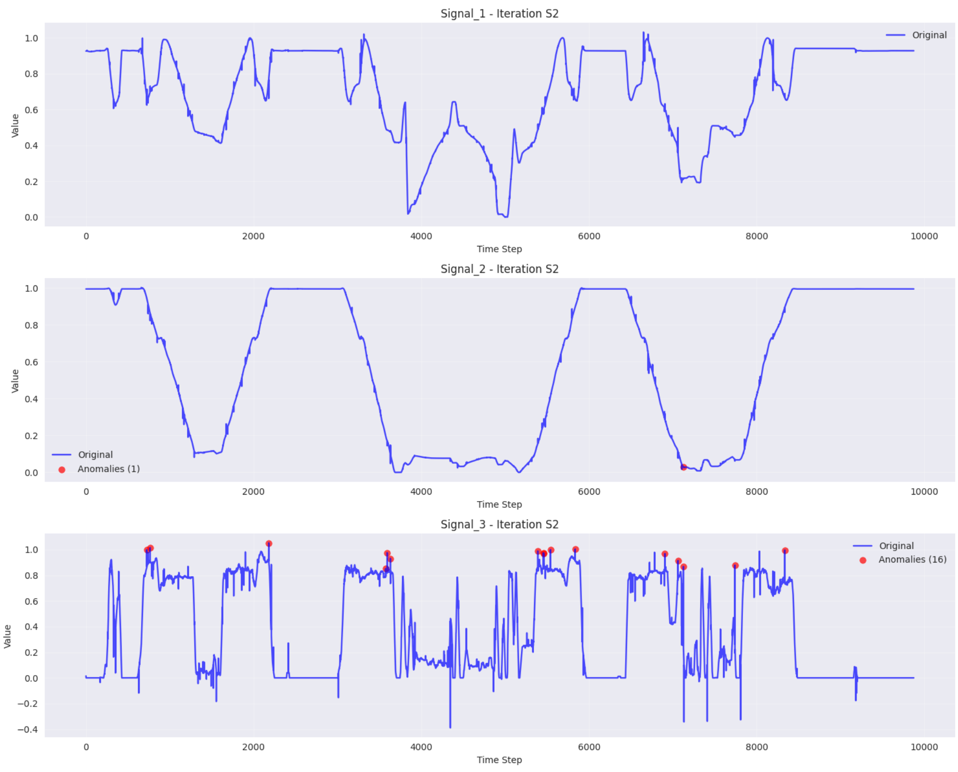


Figure 4.16: Anomaly detection after removing synthetic anomalies and reapplying anomaly detection.

4.2.2.1 Traditional Metrics

Threshold-based performance metrics, such as precision, recall, F1 score, and ROC curves, are often used to evaluate the trade-off between sensitivity and specificity in anomaly detection. While these metrics could be calculated for the model, they offer limited value in this case. The model consistently produces results dominated by true positives, with only a negligible fraction of false negatives, false positives, or true negatives, where further understanding of why this happens can be found in Appendix A. This skewed distribution renders the metrics uninformative, as they fail to provide meaningful additional insights into the model's performance. Consequently, the calculation of such metrics does not contribute to a deeper understanding of the model's capabilities or limitations in this context as can be seen below.

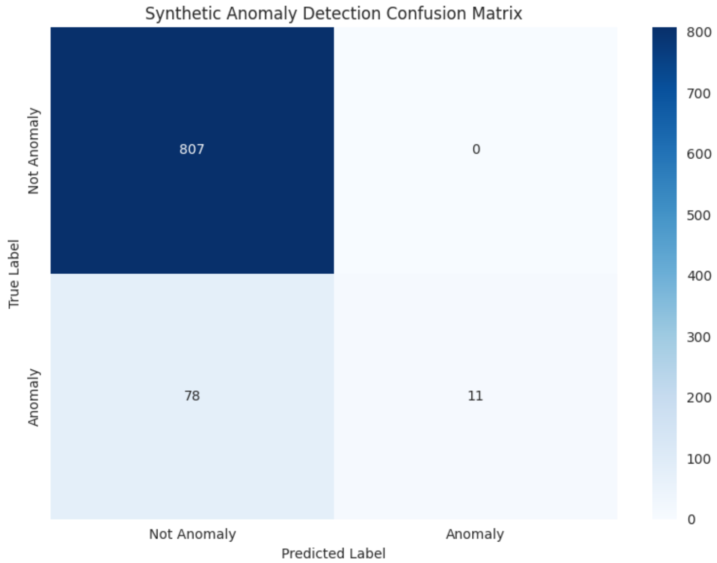


Figure 4.17: Confusion matrix illustrating the distribution of true positives, false positives, true negatives, and false negatives for the synthetic anomaly detection test.

Table 4.1: A table of the metrics from a trial run on the synthetic anomalies, calculated using the values from figure 4.17.

Precision	0.999
Sensitivity	0.001
Specificity	0.000

The confusion matrix presented in Figure 4.17 illustrates the outcomes following the second iteration of synthetic anomaly removal. These figures serve to complement the analysis by visually depicting the distribution of true positives, false positives, true negatives, and false negatives in the synthetic anomaly test. Although traditional metrics are used sparingly, the matrix offers a clearer perspective on the model's detection patterns. Additionally, since most of the more apparent anomalies were eliminated in earlier iterations, only the subtler anomalies remain. With the robustness threshold set sufficiently high, the model appropriately refrains from flagging these subtle points, as intended.

The iterative process of adding and removing synthetic anomalies, combined with statistical evaluations, demonstrates the model's robustness and adaptability. This approach further helps ensure that reliability of the model and that is not only capable of detecting anomalies but also of iteratively improving its detection accuracy in the presence of controlled distortions.

4.3 Baseline Comparison

The simpler models KMeans, KNN, DBSCAN, ARIMA, Local Outlier Factor, Isolation Forest, and One-Class Support Vector Machine were used as baseline models, establishing the minimum performance level that the LSTM VAE model should meet or exceed. Figure 4.18 visualizes the data distribution, where t-SNE has been applied to reduce the dimensionality of the data for easier visualization. The plot shows the data projected onto two dimensions (t-SNE Component 1 and Component 2). Because the baseline models rely on dimensionality reduction to handle the high-dimensional data, and the LSTM-VAE processes the original signals directly, their outputs are based on fundamentally different representations. This makes it difficult to fully compare their anomaly detection performance, as the dimensionality reduction can distort relationships that the LSTM-VAE preserves.

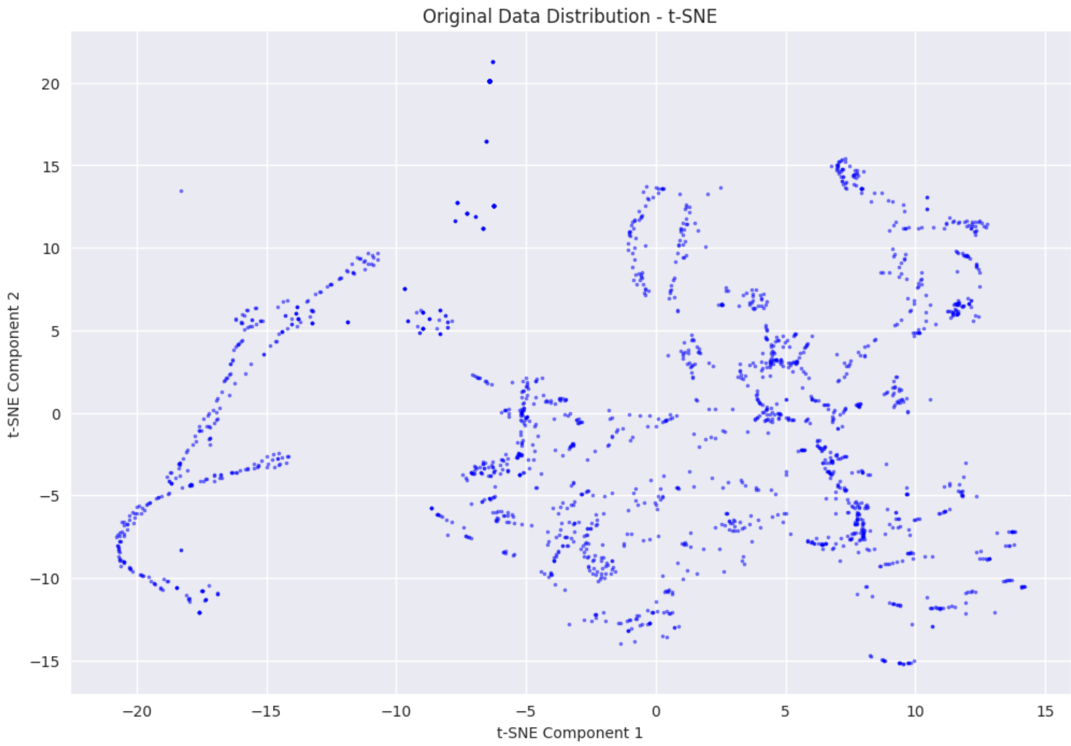


Figure 4.18: t-SNE visualization of the dataset, with components 1 and 2

Figure 4.19 displays eight subplots. Each subplot visualizes the anomaly detection performance of the models KMeans, KNN, DBSCAN, ARIMA, Local Outlier Factor, Isolation Forest and One-Class SVM using the data transformed through t-SNE. Table 4.2 presents an overview of the results shown in the subplots. Overall, most models identify around 5% of the data as anomalous. However, One-Class SVM and DBSCAN differ, detecting approximately 9% and 32% of anomalies, respectively. The eight subplots exhibit anomalies in different regions, with no clear discernible pattern across them. However, subplots one, five, seven, and eight display more pronounced anomalies in the lower-left quadrant.

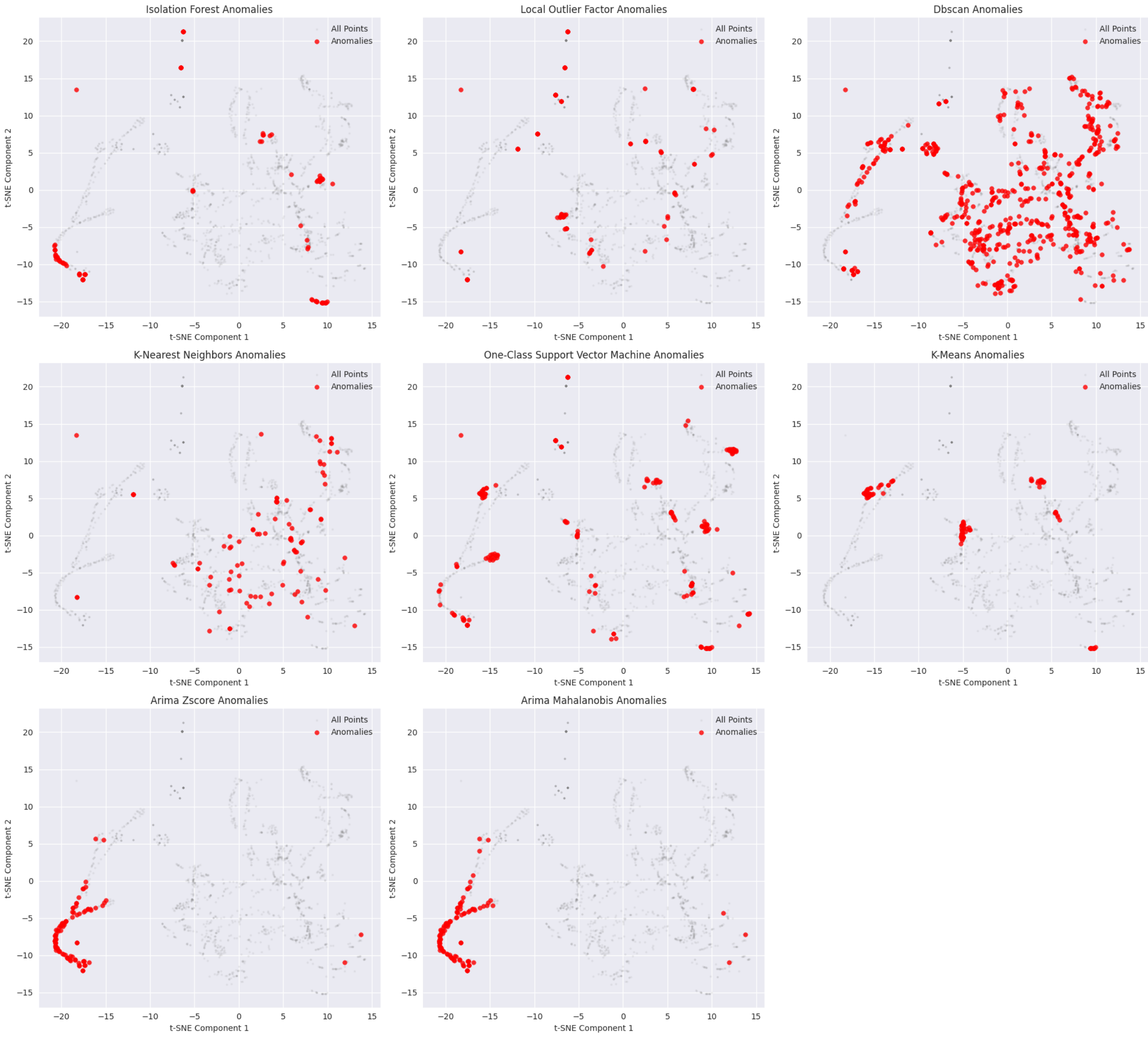


Figure 4.19: Anomaly detection performance of various models (KMeans, KNN, DBSCAN, ARIMA, Local Outlier Factor, Isolation Forest, and One-Class Support Vector Machine) visualized using t-SNE.

Method	**Anomalies Detected**	**Percentage of Data**
Isolation Forest	97	5.00%
Local Outlier Factor	97	5.00%
DBSCAN	616	31.75%
K-Nearest Neighbors	97	5.00%
One-Class SVM	183	9.43%
K-Means	97	5.00%
ARIMA Z-score	95	4.90%
ARIMA Mahalanobis	111	5.72%

Table 4.2: Number and Percentage of Anomalies Detected

Table 4.3 presents a statistical analysis of the baseline models, comparing their performance across several metrics. One-Class SVM stands out with the highest mean score and number of clusters, but also exhibits substantial variability, indicating that it identifies a wide range of anomalies. LOF and K-Means show similarly high cluster counts but differ in score behavior. LOF demonstrates extremely high variance, while K-Means appears more stable. Isolation Forest and K-Nearest Neighbors show moderate scores with relatively consistent behavior and small cluster sizes. DBSCAN and both ARIMA based methods (Z-Score and Mahalanobis) produce fewer clusters and have low average cluster sizes, indicating a conservative detection pattern that emphasizes distinct, isolated anomalies. Notably, across almost all methods, the average cluster size is around 1, indicating that the models perceive most anomalies as isolated rather than as part of larger clusters.

Table 4.3: Statistical Analysis for Anomaly Detection Methods

(a) Isolation Forest

Mean Score	0.486
Std. Dev	0.040
Clusters	74
Avg. Cluster Size	1.01

(b) Local Outlier Factor

Mean Score	$1.40 \cdot 10^7$
Std. Dev	$2.76 \cdot 10^8$
Clusters	88
Avg. Cluster Size	1.10

(c) DBSCAN

Mean Score	0.011
Std. Dev	0.103
Clusters	21
Avg. Cluster Size	1.00

(d) K-Nearest Neighbors

Mean Score	0.047
Std. Dev	0.072
Clusters	95
Avg. Cluster Size	1.02

(e) One-Class SVM

Mean Score	3.562
Std. Dev	3.226
Clusters	181
Avg. Cluster Size	1.19

(f) K-Means

Mean Score	0.922
Std. Dev	0.471
Clusters	88
Avg. Cluster Size	1.10

(g) ARIMA Z-Score

Mean Score	1.836
Std. Dev	1.123
Clusters	49
Avg. Cluster Size	1.00

(h) ARIMA Mahalanobis

Mean Score	0.852
Std. Dev	0.523
Clusters	73
Avg. Cluster Size	1.01

5
Discussion

This section underlines key aspects of the LSTM-VAE model for anomaly detection, starting with the limitations of traditional evaluation metrics, which are often unsuitable for unsupervised tasks due to the lack of labeled data and the complexity of time-series anomalies. The section highlights the uncertainty in anomaly detection, emphasizing the importance of domain expertise and visual analysis. It argues the use of synthetic anomalies to evaluate the model's learning capabilities while acknowledging their limitations. The interpretation of reconstruction errors is also analyzed, focusing on the impact of threshold selection on detection outcomes.

A comparison with simpler statistical models demonstrates the advantages of leveraging temporal and cross-signal correlations in advanced models like LSTM-VAE. Finally, the section outlines future directions, including exploring alternative architectures, integrating ensemble methods, and refining anomaly detection through iterative validation and collaboration with domain experts.

5.1 Limitations of Traditional Evaluation Metrics for LSTM-VAE

Traditional evaluation metrics such as accuracy, precision, recall, F1-score, and ROC AUC are commonly used in supervised and semi-supervised learning tasks to assess model performance. These metrics rely on the availability of labeled data, where each data point is explicitly categorized as normal or anomalous. However, these metrics are not directly applicable to the LSTM-VAE model due to its unsupervised learning approach and the nature of the data it processes. In the context of anomaly detection, such labels are often unavailable or unreliable, as anomalies are rare and may not be accurately labeled due to the subjective nature of defining what constitutes an anomaly. This becomes very evident from the investigation of removing found anomalies and adding synthetic anomalies. The outcome was that the model performed as expected in most cases successfully identifying most rougher anomalies and refraining from looking at the more subtle cases, as is intended with a robust threshold.

Even if labels are available, anomalies are typically rare events, leading to significant class imbalances. This makes metrics such as accuracy misleading, as a model could achieve high accuracy simply by predicting the majority class (e.g., normal data) while failing to detect anomalies. Furthermore, anomalies in time-series data are often contextual or collective, meaning their detection depends on temporal dependencies and relationships between the signals. Traditional metrics do not account for these complexities, as they are designed for static data. Additionally, the LSTM-VAE model operates by learning the structure of normal data and identifying deviations based on reconstruction errors and latent space representations. Metrics like confusion matrices or ROC curves, which rely on binary classification thresholds, do not align with the model's objective of assigning anomaly scores based on deviations from said learned patterns. Thus, the approach is instead to investigate the outputs of the detection as well as comparing with simpler models to achieve a deeper understanding of the trained models and the potential the model might have with labeled data.

5.2 Uncertainty in Anomaly Detection and Visual Analysis

It is essential to recognize that, in this approach to data analysis, no method is 100% reliable in determining whether a particular observation is an anomaly. While patterns in the data can provide a foundation for educated guesses, a degree of uncertainty will always remain. This uncertainty stems from the natural variability within the data, the limitations of the detection algorithms, and the possibility that normal and anomalous observations share overlapping characteristics. To provide a deeper understanding of the detection process and the complexities associated with anomalies, Appendix B presents additional visual representations of the resulting anomaly detection and potential issues with data.

Figures 4.6 and 4.7 provide examples of an expected anomaly detection. In the figure the errors in the data are both point wise and cross signal, where overall it is not possible for the human eye to detect any outlying features, indicating the deviations are probably temporal or correlated in nature. These results demonstrate more subtle anomalies occurring across longer sequences, where each data point, for all signals in a series are flagged as anomalous. These examples demonstrate how anomalies can manifest in various ways: sometimes as sharp, isolated deviations, and other times as gradual, sustained drifts over larger chunks of a signal. Importantly, these figures also demonstrate that anomalies can impact all signals simultaneously or affect a single signal in isolation. This variability emphasizes the need to consider correlations between signals when designing detection methods, as inter-signal relationships may provide critical context for distinguishing between true anomalies and natural fluctuations. Additionally, these sequences often exhibit errors on each signal individually, reinforcing the importance of analyzing both the collective behavior of signals and their individual patterns. In contrast, there are also cases which how sequences that are largely clean from anomalies across most signals, yet certain signals display stretches where nearly every data point is flagged. This suggests that the model perceives these specific signals as deviating from the shared pattern of all the signals at that sequence, underlining the importance of correlations in anomaly detection.

In some cases there would be test runs where data points over larger time periods are flagged as anomalous, which would indicate that the model thinks it deviates from the overall trend. These deviations may be stem from the temporal or correlating patterns, where a signal relationship usually behaves differently. However, without additional context or domain knowledge, it is impossible to confirm the sequences' nature with absolute certainty.

To gain valuable insight into how well the model captures temporal dependencies and underlying patterns the the original and corrected sequences was compared. Although the term corrected may suggest that the data has been modified or improved, in this context it refers to the reconstructed data generated by the LSTM-VAE model. Rather than being truly corrected, this reconstructed data reflects the model's internal representation and understanding of the input. Discrepancies between the two can reveal which features the model emphasizes or finds challenging, providing valuable perspective on its behavior and limitations. Assessing the model's performance with reconstructed or corrected data provides a qualitative view of which aspects of the input the model considers significant. However, since ground truth labels are absent, this approach is unreliable and serves only as a visual indication of performance. If ground truth labels were available to the model, the corrections could potentially leverage the predictive capabilities typically associated with RNNs. The predictive capabilities typically associated with RNNs refer to their strength in modeling and forecasting sequences by capturing temporal dependencies in data.

In all cases, the identification of anomalies relies on a combination of statistical techniques, domain expertise, and an understanding of the data context. While patterns in the data can guide the analysis, the inherent uncertainty means that some anomalies may go undetected, while others may be falsely identified. This underlines the importance of iterative analysis, validation, and collaboration with domain experts to refine the anomaly detection efforts.

5.3 Synthetic Anomalies and Model Learning

Synthetic anomalies offer a controlled means of evaluating of the model's capacity for anomaly detection. As illustrated in section 4.2.2, there is further insight to be gained from investigating how the model evaluates the consistency and integrity of patterns for the signals. Introduction of synthetic anomalies into the dataset enables the observation of the model's response. The significant rise in detected anomalies during anomaly introduction, compared to the relatively few detections before and after, suggests that the model has successfully learned the underlying structure of the normal data. This behavior suggests that the model is not simply overfitting to the training data but has developed a general understanding of the data structure. However, it is important to note that synthetic anomalies may not fully capture the complexity of real-world anomalies, and further validation is needed.

A potential issue with the technique of removing detected anomalies and then adding new synthetic ones is that, as shown in figure 4.13, removing a data point can cause the points on either side to be flagged as anomalies. This occurs because the neighboring points may no longer align smoothly, creating an artificial deviation. Despite this limitation, the technique still provides valuable information on how the model processes anomalies. For instance, in figure 4.16, multiple smaller synthetic anomalies are not detected by the model, as their deviations are insufficient to exceed the set threshold for the signals during training. This showcases the importance of threshold selection in determining the sensitivity of anomaly detection.

5.4 A baseline comparison for the model

The comparison between simpler statistical models and more advanced approaches highlights an important distinction in anomaly detection research. While the simpler models such as KMeans, KNN, DBSCAN, and Isolation Forest, offer straightforward implementations and computational efficiency, their ability to detect anomalies is typically limited by the simplicity of their assumptions. These models often rely on nearest-neighbor calculations, density estimates, or statistical thresholds, which may miss complex patterns or fail to account for inter-dependencies between signals. An representation for how these models view and find the anomalies is figure B.4, where the circles represent the distance to other data nodes.

On the other hand, advanced models like the LSTM-VAE model leverages the ability to learn temporal and cross-signal correlations, possibly capturing a better representation of the data. This approach enables the detection of more nuanced and meaningful deviations that simpler models may overlook. For instance, statistical methods that combine signals into principal components may lose critical information about the relationships between those signals, whereas models that explicitly account for these correlations are better suited to uncovering anomalies embedded in complex, high-dimensional data.

Although it is challenging to determine definitively which approach identifies the most accurate anomalies in the absence of ground truth, the incorporation of signal relationships and temporal dynamics provides a compelling case for the superior performance of advanced models. The observed variability in anomaly detection across simpler models further underscores their reliance on specific assumptions and methodologies, which may not generalize well to all datasets or anomaly types. In contrast, models that learn from the underlying structure of the data have the potential to outperform these baseline approaches, particularly in scenarios where intricate data relationships play a critical role.

5.5 Interpreting Reconstruction Error for Anomaly Detection

The reconstruction error distribution quantifies how well the model reconstructs input data, with low errors indicating normal patterns and high errors suggesting potential anomalies. By analyzing this distribution, thresholds can be set to classify data points as normal or anomalous, but the choice of threshold critically impacts sensitivity and specificity.

Figures 4.8 and 4.9 illustrate this relationship, showing how the bulk of the data represents normal patterns, while the tail corresponds to significant anomalies. The *3-sigma rule* is a common thresholding method, effective for normally distributed data but less suitable for skewed or heavy-tailed distributions. Alternatively, the Mahalanobis distance accounts for correlations between variables, making it more robust for multivariate data.

Threshold selection involves a trade-off. Lower thresholds increase sensitivity but risk more false positives, while higher thresholds reduce false positives but may miss subtle anomalies. Furthermore, these figures demonstrate how different thresholds affect anomaly detection outcomes, where the second figure might have a to low threshold, making larger chunks of the data be flagged as anomalies. Meanwhile, the first figure has similarly sporadic pattern, but since the threshold is higher for this data, not a single data point was flagged.

The reconstruction error distribution provides a solid foundation for anomaly detection, with the choice of an appropriate threshold being crucial to achieving a balance between sensitivity and specificity. However, it is important to consider the varying levels of severity among anomalies. While some anomalies may be flagged, significant spikes in the data can substantially raise the threshold, potentially allowing smaller anomalies to go undetected. This phenomenon is particularly evident in figure B.1, which illustrates the reconstruction error of a huge spike in the data. In this case, faulty data causes the threshold to spike, leading to a failure in detecting other outlying data points.

5.6 Challenges in Modeling Multivariate Time-Series Data

The model's performance on sequences may be affected by several factors. Firstly, if the sequences are too short, the model may not have enough context to accurately capture temporal dependencies. Conversely, if the sequences are too long, the model may struggle with vanishing gradients or memory limitations, particularly in architectures like LSTMs. Additionally, noise in the data can lead to incorrect reconstructions, as the model may interpret random fluctuations as meaningful patterns. While outlier handling and smoothing techniques applied during data preprocessing can help mitigate this, there is no guarantee that all such noise is fully removed. Another issue arises when the data contains overlapping anomalies or subtle changes that are difficult to distinguish from normal variations. These challenges highlight the importance of carefully preprocessing the data and selecting appropriate sequence lengths for training.

Furthermore, another challenge may be that training a model which accounts for the correlation between multiple signals is inherently difficult due to the exponential increase in computational complexity as more signals are added. Each additional signal introduces new dimensions to the data, which significantly increases the number of relationships and dependencies between the signals, very quickly creating quite complex data. This complexity only increases further as time sequences are extended to capture broader trends, resulting in a rapid growth in the computational demands necessary to identify even a fraction of the underlying patterns.

Moreover, increased data dimensionality poses significant challenges for models like LSTM-VAEs. One major issue is overfitting, where the model becomes overly specialized to the training data and struggles to generalize to unseen data, as illustrated in figure 4.2, where multiple points across neighboring sequences are incorrectly classified as anomalies. High-dimensional data also demands greater memory and computational resources, leading to slower and more resource-intensive training. Additionally, as the number of signals increases, the risk of introducing noise or irrelevant features grows, degrading overall model performance. For LSTM-VAEs, which rely on sequential processing and latent space representations, this complexity is particularly problematic. The encoder and decoder must process higher-dimensional inputs and outputs while ensuring the latent space effectively captures the increased data complexity, often requiring larger models with more parameters and additional data for effective training, further compounding the computational burden.

5.7 Future work

In future work, it would be valuable to compare the performance of the LSTM-VAE model with other advanced architectures, such as Transformers and GANs. Such comparisons would help evaluate the differences in each model's ability to capture complex patterns, reconstruct data accurately, and maintain robustness across various applications. Additionally, incorporating alternative architectures like Bi-LSTMs, switching to GRU layers or CNN-based approaches could further enhance the model's performance and adaptability to different use cases.

Improving the reliability of the results should also be a priority. If multiple advanced models consistently identify the same data points as anomalous, their agreement would increase confidence in the findings, suggesting that the detected anomalies are highly likely to be valid. This consensus among models would provide a more dependable and robust approach to anomaly detection, offering greater reliability and accuracy across diverse datasets and scenarios without the need for manual labeling of anomalies.

6
Conclusion

This thesis has demonstrated that it is possible to create a model capable of detecting anomalies in multivariate time-series data by leveraging relationships between signals. The model shows clear promise in surpassing traditional classifiers by adapting to the inherent dependencies and correlations within the data. However, given the lack of a guaranteed method for labelling anomalies as true or false, the model's purpose shifts from being a definitive anomaly detection tool to serving as a suggestion mechanism that provides informed estimations.

Although the model achieves notable success in identifying deviations and approximating thresholds, further refinement is needed to improve its interpretability and robustness. The introduction of labelled datasets could significantly enhance its performance by enabling semi-supervised learning approaches. This would allow for a more precise evaluation of the model's outputs and a better understanding of its decision-making process.

Furthermore, this thesis has shown that the model excels in identifying synthetic anomalies, which suggests that it has effectively learnt the structure of normal data. However, the complexity of real-world anomalies presents additional challenges that require further investigation. To address these challenges, future research could explore more advanced architectures, such as Transformers and GANs, which may offer improved capabilities in capturing complex patterns and handling high-dimensional data. Comparing these architectures with the LSTM-VAE model could provide valuable insights into their relative strengths and weaknesses.

In conclusion, while this model demonstrates high potential in anomaly detection, it serves as a foundation for further development and refinement. By incorporating additional data, leveraging semi-supervised learning, and exploring more advanced architectures, the model could evolve into a more robust and versatile tool for identifying anomalies in increasingly complex scenarios.

Bibliography

[1] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, Mar. 2021, ISSN: 0360-0300. DOI: `10.1145/3439950`. [Online]. Available: `https://doi.org/10.1145/3439950`.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009, ISSN: 0360-0300. DOI: `10.1145/1541880.1541882`. [Online]. Available: `https://doi.org/10.1145/1541880.1541882`.

[3] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLoS One*, vol. 11, no. 4, e0152173, 2016. DOI: `10.1371/journal.pone.0152173`. [Online]. Available: `https://doi.org/10.1371/journal.pone.0152173`.

[4] E. Bee Dagum, "Time series modeling and decomposition," *Statistica*, vol. 70, no. 4, pp. 433–457, Jan. 2010. DOI: `10.6092/issn.1973-2201/3597`. [Online]. Available: `https://rivista-statistica.unibo.it/article/view/3597`.

[5] Z. Zamanzadeh Darban, G. I. Webb, S. Pan, C. Aggarwal, and M. Salehi, "Deep learning for time series anomaly detection: A survey," *ACM Computing Surveys*, vol. 57, no. 1, pp. 1–42, Oct. 2024, ISSN: 1557-7341. DOI: `10.1145/3691338`. [Online]. Available: `http://dx.doi.org/10.1145/3691338`.

[6] E. R Rosca, "Stationary and non-stationary time series," *The USV Annals of Economics and Public Administration*, vol. 10, no. 1, pp. 177–186, 2011.

[7] J. D. Cryer and K.-S. Chan, "2.3 stationarity," in *Time Series Analysis: With Applications in R*, ser. Springer Texts in Statistics, 2nd ed., Springer New York, NY, 2008, pp. 16–19, ISBN: 978-0-387-75958-6. DOI: `10.1007/978-0-387-75959-3`.

[8] J. D. Cryer and K.-S. Chan, "6.4 nonstationarity," in *Time Series Analysis: With Applications in R*, ser. Springer Texts in Statistics, 2nd ed., Springer New York, NY, 2008, pp. 128–130, ISBN: 978-0-387-75958-6. DOI: `10.1007/978-0-387-75959-3`.

[9] P. Schober, C. Boer, and L. A. Schwarte, "Correlation coefficients: Appropriate use and interpretation," *Anesthesia & Analgesia*, vol. 126, no. 5, pp. 1763–1768, May 2018. DOI: `10.1213/ANE.0000000000002864`.

[10] J. D. Cryer and K.-S. Chan, "6.2 the partial and extended autocorrelation functions," in *Time Series Analysis: With Applications in R*, ser. Springer Texts in Statistics, 2nd ed., Springer New York, NY, 2008, pp. 112–115, ISBN: 978-0-387-75958-6. DOI: `10.1007/978-0-387-75959-3`.

[11] C. Chatfield, "Cross-covariance and cross-correlation functions," in *The Analysis of Time Series: Theory and Practice*, C. Chatfield, Ed., Published: 01 January 1975, Springer New York, NY, 1975, ch. 8, pp. 169–174, ISBN: 978-0-412-14180-5. DOI: `https://doi.org/10.1007/978-1-4899-2925-9`.

[12] S. Hansun, "A new approach of moving average method in time series analysis," in *2013 Conference on New Media Studies (CoNMedia)*, 2013, pp. 1–4. DOI: 10.1109/CoNMedia.2013.6708545.

[13] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20, Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 3395–3404, ISBN: 9781450379984. DOI: 10.1145/3394486.3403392. [Online]. Available: https://doi.org/10.1145/3394486.3403392.

[14] C. Aggarwal, "Chapter 8: Recurrent neural networks," in *Neural Networks and Deep Learning: A Textbook*, Springer, 2018, pp. 265–278, ISBN: 978-3-319-94462-3. DOI: 10.1007/978-3-319-94463-0.

[15] I. Goodfellow, Y. Bengio, and A. Courville, "Sequence modeling: Recurrent and recursive nets," in *Deep Learning*, Cambridge, MA: MIT Press, 2016, ch. 10. [Online]. Available: https://www.deeplearningbook.org/contents/rnn.html.

[16] S. Kostadinov, *Recurrent Neural Networks with Python Quick Start Guide*. Packt Publishing, 2018, ISBN: 9781789132335.

[17] C. Aggarwal, "4.3.2 the vanishing and exploding gradient problems," in *Neural Networks and Deep Learning: A Textbook*, Springer, 2018, pp. 124–125, ISBN: 978-3-319-94462-3. DOI: 10.1007/978-3-319-94463.

[18] A. Graves, "Supervised sequence labelling with recurrent neural networks," in *Studies in Computational Intelligence*, Chapter 4: Long Short-Term Memory, 2012, ch. 4, pp. 37–44. [Online]. Available: https://api.semanticscholar.org/CorpusID:2118350.

[19] A. Anwarsha and T. Narendiranath Babu, "Intelligent fault detection of rotating machinery using long-short-term memory (lstm) network," in *Proceedings of the 2nd International Conference on Emerging Technologies and Intelligent Systems*, M. A. Al-Sharafi, M. Al-Emran, M. N. Al-Kabi, and K. Shaalan, Eds., Cham: Springer International Publishing, 2023, pp. 76–83, ISBN: 978-3-031-20429-6.

[20] C. Aggarwal, "8.5 long short-term memory (lstm)," in *Neural Networks and Deep Learning: A Textbook*, Springer, 2018, pp. 285–287, ISBN: 978-3-319-94462-3. DOI: 10.1007/978-3-319-94463.

[21] C. Aggarwal, "3.4 unsupervised learning with autoencoders," in *Neural Networks and Deep Learning: A Textbook*, Springer, 2018, pp. 88–89, ISBN: 978-3-319-94462-3. DOI: 10.1007/978-3-319-94463.

[22] W. Xu, H. Sun, C. Deng, and Y. Tan, "Variational autoencoder for semi-supervised text classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Feb. 2017. DOI: 10.1609/aaai.v31i1.10966. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/10966.

[23] R. Torres-Calderon, B. Gonzalez-Torres, and R. Menchaca-Mendez, "Generating universum instances from variational autoencoder latent space for semi-supervised learning," in *Telematics and Computing*, M. F. Mata-Rivera, R. Zagal-Flores, and C. Barria-Huidobro, Eds., Cham: Springer Nature Switzerland, 2024, pp. 90–104, ISBN: 978-3-031-77293-1.

[24] R. Xie, N. M. Jan, K. Hao, L. Chen, and B. Huang, "Supervised variational autoencoders for soft sensor modeling with missing data," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2820–2828, 2020. DOI: 10.1109/TII.2019.2951622.

[25] L. Cinelli, M. Marins, E. da Silva, and S. Netto, "5 variational autoencoder," in *Variational Methods for Machine Learning with Applications to Deep Networks*, Springer, 2021, pp. 111–120, ISBN: 978-3-030-70678-4. DOI: 10.1007/978-3-030-70679-1.

[26] L. Correia, J.-C. Goos, P. Klein, T. Bäck, and A. Kononova, "Ma-vae: Multi-head attention-based variational autoencoder approach for anomaly detection in multivariate time-series applied to automotive endurance powertrain testing," in *Proceedings of the 15th International Joint Conference on Computational Intelligence*, SCITEPRESS - Science and Technology Publications, 2023, pp. 407–418. DOI: 10.5220/0012163100003595. [Online]. Available: http://dx.doi.org/10.5220/0012163100003595.

[27] T. Xie, Q. Xu, C. Jiang, Z. Gao, and X. Wang, "A robust anomaly detection model for pumps based on the spectral residual with self-attention variational autoencoder," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 6, pp. 9059–9069, 2024. DOI: 10.1109/TII.2024.3381790.

[28] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[29] P. Contributors, *Pytorch: An open-source deep learning framework*, https://pytorch.org, Accessed: 2025-04-24, 2025.

[30] R. Team, *Ray tune: Hyperparameter tuning at scale*, https://docs.ray.io/en/latest/tune/index.html, Accessed: 2025-04-24, 2025.

[31] S.-l. developers, *Neighbors module — scikit-learn 1.0.2 documentation*, https://scikit-learn.org/stable/modules/neighbors.html, Accessed: 2025-04-24, 2025.

[32] S.-l. developers, *Localoutlierfactor — scikit-learn 1.0.2 documentation*, https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html, Accessed: 2025-04-24, 2025.

[33] S.-l. developers, *Dbscan — scikit-learn 1.0.2 documentation*, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html, Accessed: 2025-04-24, 2025.

[34] S.-l. developers, *Isolationforest — scikit-learn 1.0.2 documentation*, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html, Accessed: 2025-04-24, 2025.

[35] S.-l. developers, *Oneclasssvm — scikit-learn 1.0.2 documentation*, https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html, Accessed: 2025-04-24, 2025.

[36] S.-l. developers, *K-means clustering — scikit-learn 1.0.2 documentation*, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html, Accessed: 2025-04-24, 2025.

[37] S. developers, *Arima — statsmodels 0.13.0 documentation*, https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima.model.ARIMA.html, Accessed: 2025-04-24, 2025.

[38] S.-l. Developers, *Sklearn.manifold.tsne — scikit-learn 0.24.2 documentation*, https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html, Accessed: 2025-04-24, 2025.

[39] G. Naidu, T. Zuva, and E. M. Sibanda, "A review of evaluation metrics in machine learning algorithms," in *Artificial Intelligence Application in Networks and Systems*,

R. Silhavy and P. Silhavy, Eds., Cham: Springer International Publishing, 2023, pp. 15–25, ISBN: 978-3-031-35314-7.

[40] A. Painsky, "Quality assessment and evaluation criteria in supervised learning," in *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*, L. Rokach, O. Maimon, and E. Shmueli, Eds., Springer Cham, 2023, pp. 171–195. DOI: 10.1007/978-3-031-24628-9_9. [Online]. Available: https://doi.org/10.1007/978-3-031-24628-9_9.

A

Appendix-Theory

A.1 Traditional Evaluation methods

There are various methods to evaluate the performance of a machine learning model. For supervised and semi-supervised learning common ones are accuracy, precision, recall, f1-score, confusion matrix, ROC AUC curve and mean square error [39].

A confusion matrix assesses the performance of a classification model by presenting an in-depth comparison of predicted and actual outcomes in a table format, distinguishing where the model makes mistakes [39].

	Predicted Positive	**Predicted Negative**
Actual Positive	**TP (True Positives)**	**FN (False Negatives)**
Actual Negative	**FP (False Positives)**	**TN (True Negatives)**

- **TP (True Positives)**: The actual positive cases that the model correctly identifies as positive.
- **TN (True Negatives)**: The actual negative cases that the model correctly identifies as negative.
- **FP (False Positives)**: The actual negative cases that the model incorrectly classifies as positive.
- **FN (False Negatives)**: The actual positive cases that the model incorrectly classifies as negative.

Accuracy measures the proportion of correct predictions made out of all predictions made.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Specificity or True Negative Rate (TNR) measures the proportion of correctly predicted negatives among all instances that were predicted as negative. A high specificity indicates that the model effectively distinguishes negative cases from positive ones, which is crucial when it is important to avoid false positives [40].

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Precision measures the proportion of correctly predicted positive cases among all instances that were predicted as positive. A high precision score is significant when false positives have a high cost, ensuring that when the model predicts something as positive, there is a high likelihood it is correct [40].

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall, Sensitivity, or True Positve Rate (TPR) measures the proportion of correctly predicted positive cases among all actual positive instances in the data. A high recall score indicates that the model reliably detects positive cases, which is of great weight when it is critical not to miss any positive cases, even if that means allowing some false positives [40].

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-score is the harmonic mean of precision and recall. A high F1-score means that there is a good balance between precision and recall, making it particularly valuable when both false positives and false negatives should be minimized [40].

$$F1\text{-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

False Positve Rate (FPR) measures the proportion of incorrectly predicted positive cases among all actual negative instances in the data. If the FPR is high, it means that a large number of negative cases are incorrectly classified as positive, which is beneficial to minimize when false positives are costly [40].

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$

The *Receiver Operating Characteristic* (ROC) curve and *Area Under the Curve* (AUC) provide a visual representation of a model's performance in classification tasks, with the AUC representing the area beneath the ROC curve. The ROC curve plots the TPR is on the y-axis and the FPR on the x-axis, for various classification thresholds. The thresholds determine the cut-off point for classifying a sample as positive or negative. A lower threshold increases the number of cases classified as positive, raising both true positives and false positives. In contrast, a higher threshold reduces the number of cases classified as positive, leading to fewer true positives and false positives. The ROC AUC score is takes values from 0 to 1. If the score is 1, the model perfectly distinguishes between the positive and negative classes, correctly classifying all instances. If the score is 0.5, indicates that the model is unable to differentiate between the classes, performing at the level of random guessing. A score between 0.5 and 1 the model demonstrates the ability to correctly distinguish between positive and negative classes, as it is able to identify both true positives and true negatives. Lastly, a score below 0.5 indicates that the model is performing worse than random guessing, meaning it is frequently misclassifying positive and negative instances [39].

MSE (Mean Squared Error) is calculated by taking the average of the squared differences between the actual input data and the predicted values in the dataset. MSE should be minimized, with lower values reflecting better model performance and a more accurate representation of the true input data [39].

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \quad \text{where:}$$

- n is the total number of data points
- Y_i is the actual observed value at index i
- $\hat{Y}_i$ is the predicted value at index i

B

Appendix-Results

B.1 Anomaly plots

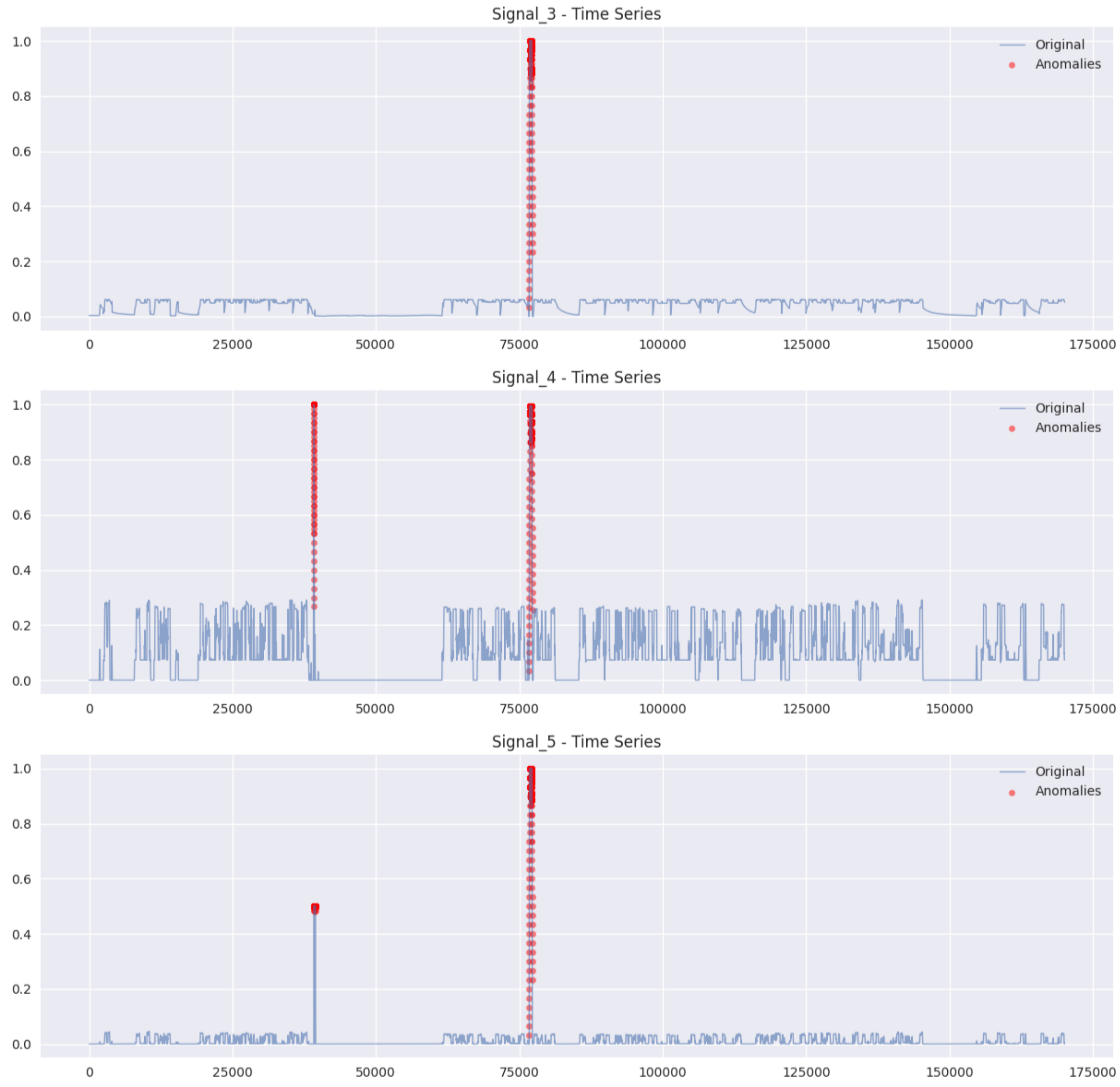


Figure B.1: A trail of the anomaly detection for 7 signals, showing 3 of the signals with a visually obvious case of faulty data.

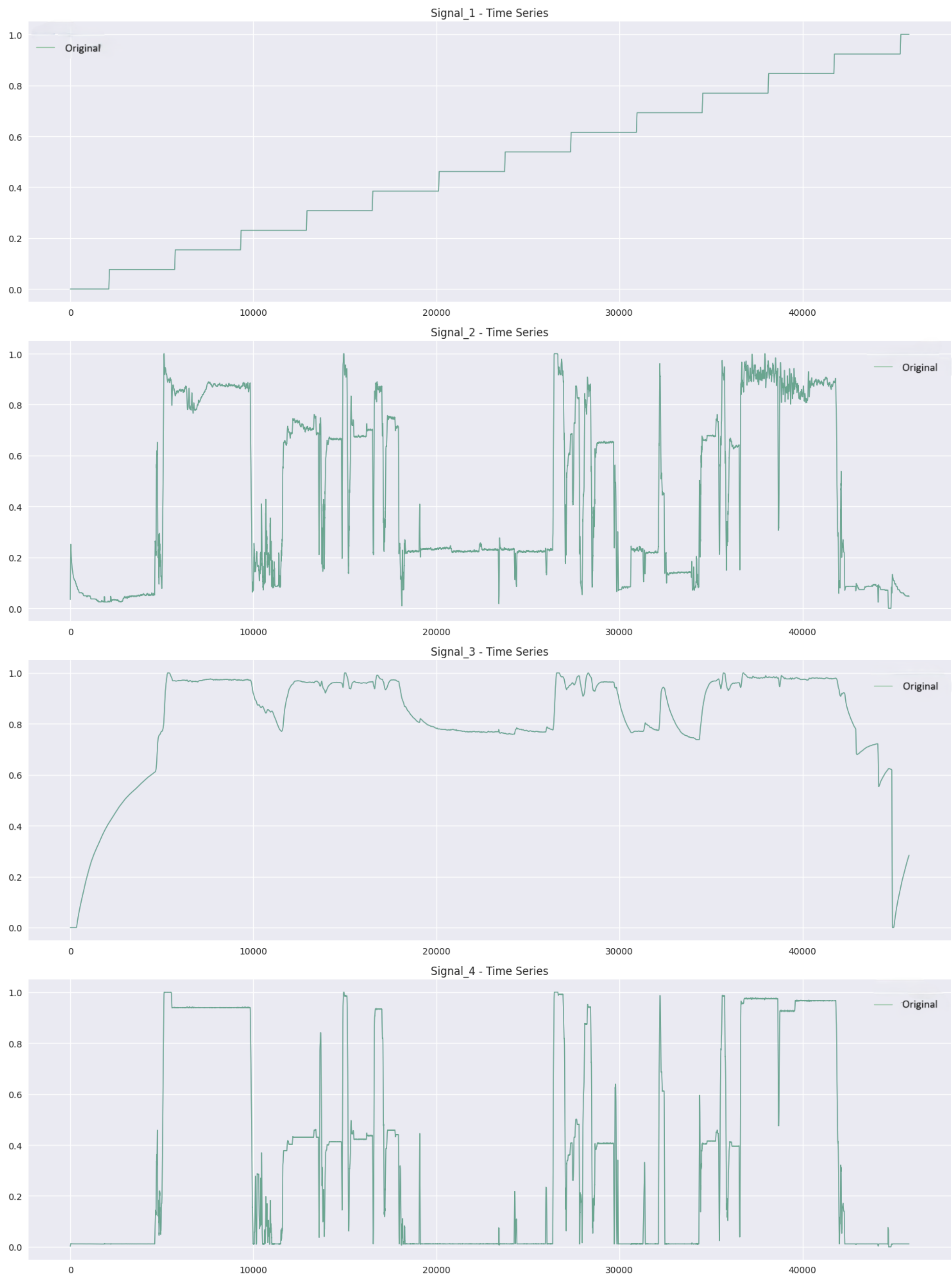


Figure B.2: The first 4 signals for the anomaly detection of 7 signals with no detected anomalies for the signals, including the reconstructed data which doesn't deviate from the original.

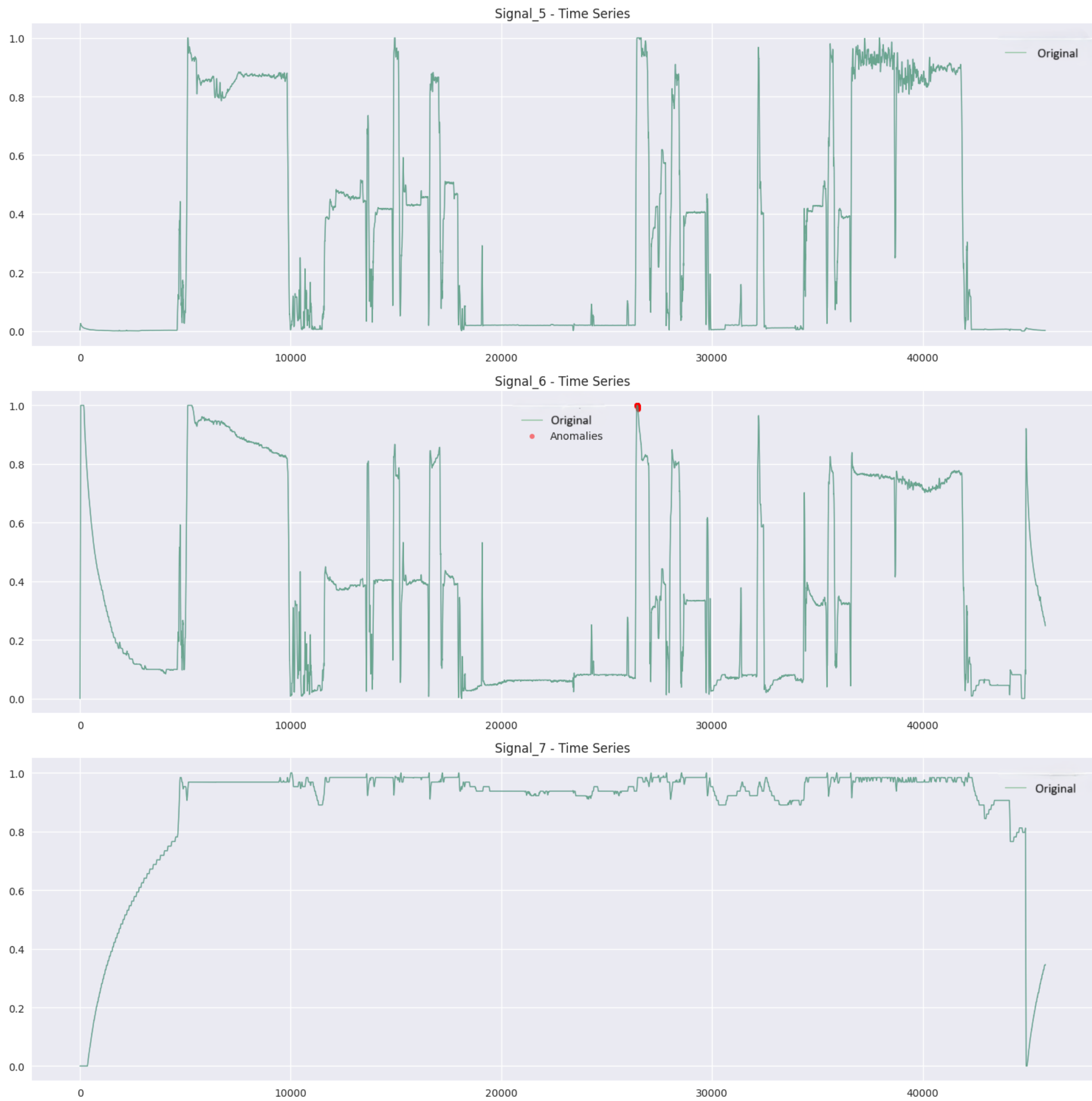


Figure B.3: The last 3 signals for the anomaly detection of 7 signals with only 2 detected anomalies for the signals, including the reconstructed data which doesn't deviate from the original.

However, while many of these cases strictly show where anomalies are found, the model is also capable of returning clean datasets. Figure B.2 and B.3 which are from the same run, depict larger portions of the data where the model did not flag nearly as many anomalies as the other plots, which may indicate at the very least that the model has memorized a larger portion of the possible patterns.

B.2 Baseline

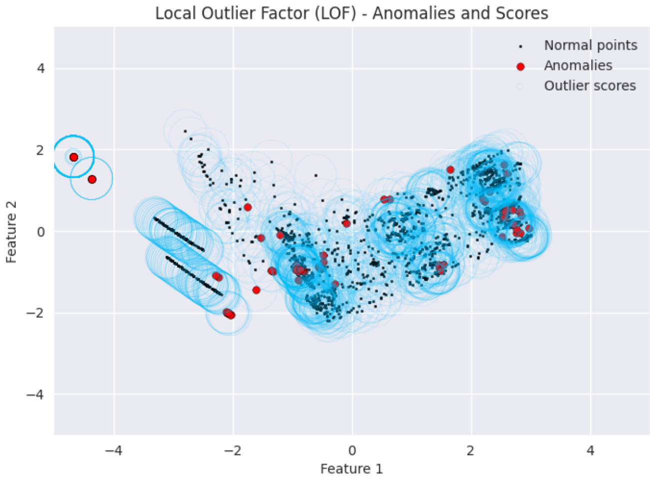


Figure B.4: Visualization of the outlier detection of the LOF.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY