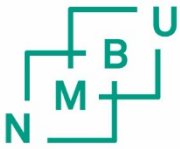


Norwegian University
of Life Sciences

Master's Thesis 2023 30 ECTS
Faculty of Science and Technology - REALTEK

Anomaly detection in industrial time series sensor data

Ivan Cherednikov
Data Science

Abstract

Anomaly detection in industrial time series data is essential for identifying and preventing potential issues in production processes, ensuring high product quality and reducing downtime. This master's thesis investigates the performance of two unsupervised machine learning algorithms, Local Outlier Factor (LOF) and DBSCAN, for detecting clogging events in the production process of Bioco, a company in the biotechnology industry. The main objective is to evaluate the algorithms' ability to provide early warnings for clogging events, enabling timely preventive actions.

The research process involves a thorough theoretical overview, data exploration and preprocessing, application of the selected algorithms, and evaluation of their performance. The study also examines the sensitivity of the algorithms to parameter tuning and the effectiveness of incorporating lagged variables as features in the anomaly detection models.

The results indicate that both LOF and DBSCAN can detect relevant anomalies in the time series data, but their performance in providing early warnings for clogging events is limited. While LOF requires careful parameter tuning, DBSCAN demonstrates more stable performance across different parameter settings. The inclusion of lagged variables does not improve the detection of clogging events, showcasing challenges in selecting the optimal lag length.

This study contributes to the existing literature on anomaly detection in industrial time series data by providing insights into the practical performance of LOF and DBSCAN algorithms in a specific industrial context. The findings highlight the importance of considering the effects of lagged variables and parameter tuning when designing anomaly detection models for industrial applications. Future research could explore other anomaly detection algorithms and their performance in different industrial settings to enhance the generalizability of the results.

Acknowledgements

This thesis marks the end of my five year journey at NMBU. Years filled with joy, tears, hope, ups and downs. I would like to express my deepest gratitude to my supervisor, Professor Kristian Hovde Liland, for his teaching, undivided attention, encouragement and immense help. Your dedication and commitment to my growth have been crucial in shaping my approach to research.

I would also like to thank my co-supervisor, senior researcher at Nofima Ingrid Måge for her invaluable advice, guidance and constructive feedback. I can not express how fortunate I am to have had the opportunity to learn from such a knowledgeable and inspiring mentor.

My deepest thanks to my incredibly kind flatmates for the countless laughter, shared meals, pep talks, and for allowing me to occupy a part of the living room sofa as my personal master place during the last weeks of writing.

I owe a debt of gratitude to my family, who have been my anchor in this storm. Galina and Victor, your unconditional support, your sacrifices and unwavering faith in me throughout have given me the strength to persevere. To my soulmate, Ester, thank you for being my cheerleader and for always reminding me that I am not alone in this journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Making sure that manufacturing processes are as efficient and successful as possible is essential in today's industrial climate. These procedures are susceptible to anomalies and unforeseen events that might result in significant financial losses and diminished productivity. To lessen the impact of such events, accurate and rapid anomaly detection is required. A good illustration is the continuous enzymatic hydrolysis process used by Bioco to generate food items by processing chicken and other birds. Clogging incidents can significantly interrupt this process, halting all production and incurring major expenditures for the organization. The goal of this thesis is to create a machine learning model that can use time series data from sensors positioned throughout the production process to identify and forecast clogging incidents. These sensors gather crucial data for identifying system irregularities, including flow rates, pressures, temperatures, torques, and velocities. This thesis discusses the difficulties connected with anomaly identification in industrial time series data. It takes a strong and scalable solution to address these industrial issues.

1.1 Motivation

The practical application of anomaly detection algorithms to actual industrial time series data serves as the driving force behind this thesis. By effectively detecting anomalies, companies like Bioco can implement proactive measures to prevent disruptions, reduce downtime, and minimize financial losses. A wide number of businesses may benefit from the development of effective anomaly detection algorithms, which can also advance the field of industrial process optimization. The growing importance of data-driven decision-making in manufacturing and the increasing complexity of production processes further emphasize the need for advanced techniques to identify and predict anomalies.

1.2 Challenges in anomaly detection

Anomaly detection on industrial time series data presents many challenges that have to be addressed before implementation to create effective solutions. These challenges include, but are not exclusive to:

Data Quality: Missing values, noise, and discrepancies may be present in time series data collected from industrial processes. It is essential to preprocess and clean the data to address these problems to guarantee the accuracy and reliability of the anomaly detection algorithms. To address data quality challenges, it is essential to apply appropriate data imputation algorithms

and noise reduction methods.

Feature Selection: A critical part of the procedure is choosing the features that are most relevant for anomaly detection. The performance of the algorithms can be negatively impacted by irrelevant or redundant features, which can also make the models' computations more complex. Advanced feature selection techniques, such as wrapper and embedded methods, can aid in selecting the most informative features.

Model Interpretability: It's critical to comprehend how an anomaly detection system functions in industrial settings. Model interpretability enables domain specialists to understand the discovered anomalies, identify the causes, and take the necessary actions to resolve the problems and resume the process. Better communication between data scientists and domain specialists in the industry can be facilitated by developing interpretable models or adding explainability approaches, resulting in more informed decision making.

Scalability: Industrial time series data can be large and high-dimensional. Anomaly detection methods should be scalable to handle extensive datasets and adapt to changes in the data over time. The scalability of the algorithms can be improved and the computational load can be decreased by using techniques for parallel processing, distributed computing, and efficient data storage. Incorporating adaptive model updating strategies can ensure that the algorithms remain effective as the data and processes evolve.

False positives and negatives: For the detection system to be effective, it is crucial to strike a balance between false positives, which are instances flagged as anomalies even when they are regular events and false negatives, cases where the system failed to detect real anomalies. To reduce the effect of both types of mistakes on the system's overall performance, an appropriate balance should be struck. This balance can be enhanced by creating methods to adjust and fine-tune model thresholds, and incorporating domain knowledge.

Real-time detection: In an industrial setting, real-time anomaly detection is essential for preventing disruptions and minimising the impact on the production process. The selected anomaly detection algorithms should be able to process the data and detect anomalies in a timely manner, enabling immediate corrective measures. Real-time streaming data processing and efficient model deployment techniques can contribute to achieving this.

1.3 Objectives

This thesis' main goal is to investigate and select appropriate anomaly detection algorithms for Bioco's time series data, encompassing both traditional statistical methods and deep learning strategies. The objective is to determine the best-performing algorithm that can accurately predict, flag, and detect clogging events before they affect production and result in high costs for the company. The effectiveness of these algorithms will be evaluated by observing how early anomalous behaviour can be detected prior to a clogging event, and the best algorithm will be suggested for implementation in Bioco's manufacturing process. The secondary goal is to present a thorough understanding of the difficulties encountered in anomaly identification for industrial time series data and to suggest solutions to these difficulties.

1.4 Scope and limitations

The analysis of time series data from sensors measuring flow rates, pressures, temperatures, and other metrics in Bioco's continuous enzymatic hydrolysis process is the sole focus of this thesis. While machine learning algorithms will be the main focus, a brief overview of conventional statistical techniques for anomaly identification will also be given. This thesis does not cover other data types or attempt to provide an exhaustive review of all possible anomaly detection algorithms. Instead, a selection of relevant algorithms will be tested and evaluated based on their performance within the specific context of Bioco's production process. Furthermore, the thesis acknowledges the limitations arising from only using data originating from the same industrial process and the potential impact of these limitations on the generalizability of the findings.

1.5 Thesis structure

This thesis will, in the following chapters, explore the theoretical foundations of existing approaches and discuss the challenges and limitations associated with current methods. It will dive into the theoretical foundations of anomaly detection and explore different methods for applying these to time series data. Furthermore, results of applying these methods to Bioco's data will be presented.

Chapter 2

Theory

2.1 Anomaly detection

Anomaly detection is a widely studied topic in various domains [1], and it refers to the process of identifying patterns or data points that significantly deviate from the regular, expected behaviour. When focusing on industrial systems, these anomalies may lead to equipment failure, inefficiencies in the process or safety hazards. Detection of such anomalies in a timely manner is crucial for maintaining the smooth operation of the system, preventing damage, added costs and ensuring worker safety. In the context of industrial big data, Caithness and Wallom (2018) emphasise the significance of developing and implementing efficient anomaly detection techniques at industrial scale [2]. In the case of Bioco's industrial process, clogging events halt the entire production for hours, resulting in high costs for the company. Evidently, anomaly detection is of high importance for preventing the clogging events.

An important aspect to consider in anomaly detection is that deviations from normal behaviour are not always observable before a major incident occurs. This can be attributed to complex industrial processes not having comprehensive data or sensors to monitor every aspect of the system. As such, subtle process changes can go unnoticed until they accumulate and cause major disruptions. Some anomalies may be the result of complex interactions between different system components. Such interactions can be difficult to recognise without a deep understanding of underlying relationships.

2.1.1 Anomaly types

Data anomalies can be split up into three categories: point, contextual and collective anomalies. Each of these possess unique characteristics and require different approaches for detection and managing. A comprehensive review of anomaly types and outlier detection methodologies and can be found in the work of Hodge and Austin (2004) [3]

2.1.1.1 Point anomalies

Point anomalies represent single data points that significantly deviate from the rest of the data. These can be identified by individual analysis of each data point, deciding if it lies outside the regular range or the data distribution. These anomalies are the easiest to detect, and can be discovered using even statistical methods.

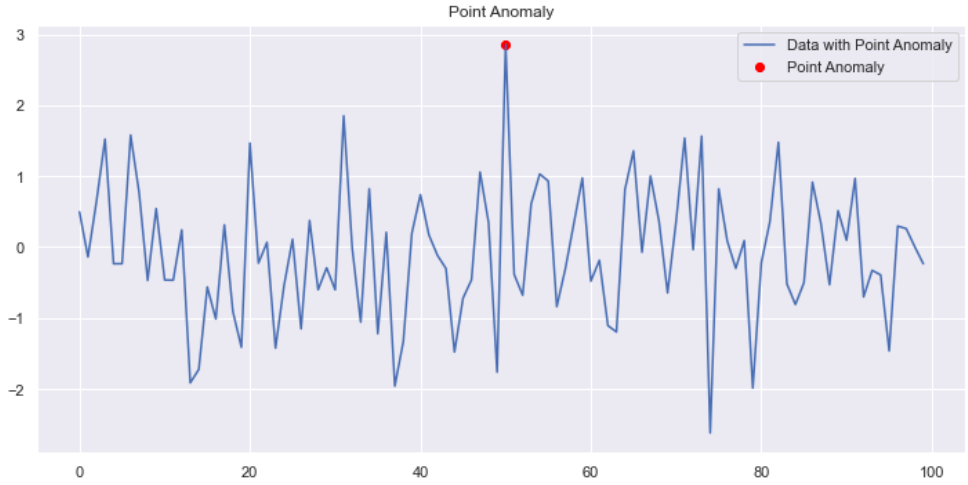


Figure 2.1: A figure illustrating a point anomaly in synthetic data

2.1.1.2 Contextual anomalies

Contextual anomalies are data points that are anomalous in specific contexts, and not necessarily when considered individually. To find these, one often needs to have an understanding of the underlying contextual information, making them more challenging to detect. These are commonly observed in time series, where the context is provided by the data points' relationships to their neighbouring data.

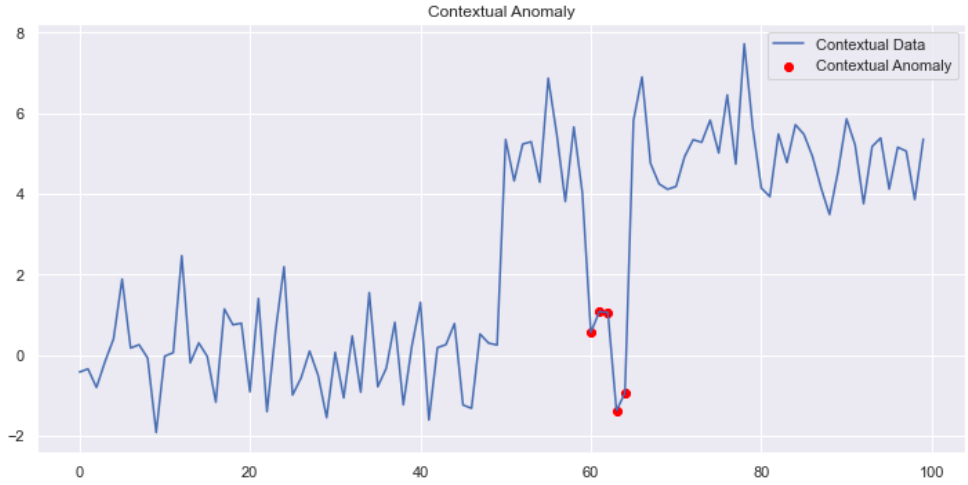


Figure 2.2: A figure illustrating contextual anomalies in synthetic data. The deviations here are with regard to the current state of the process and are not deviations from the total distribution of the measurements.

2.1.1.3 Collective anomalies

When groups of data points are considered anomalous when seen together, they are seen as collective anomalies. These are often not detected when examining individual data points, but require analysis as a whole group before the anomalous pattern emerges. Similar to contextual anomalies, these are common in time series data, where groups of data points may violate the regular temporal behaviour.

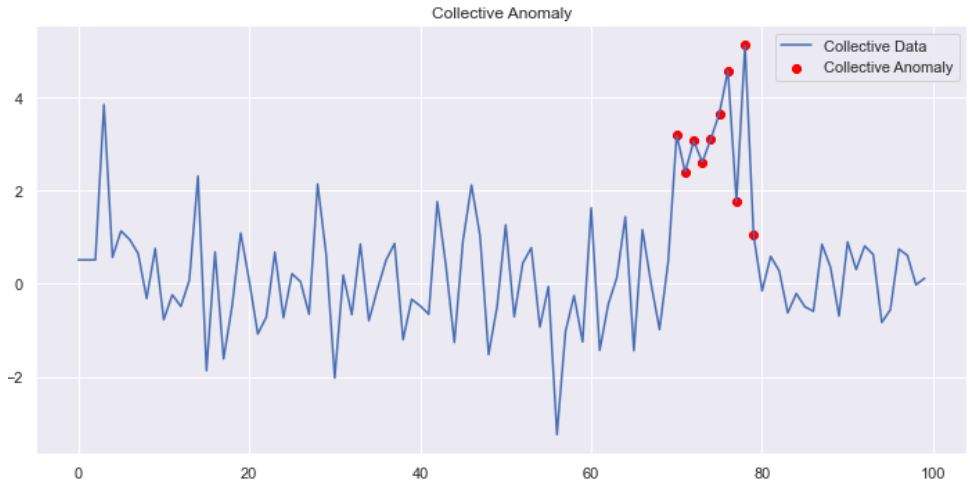


Figure 2.3: A figure illustrating collective anomalies in synthetic data

2.1.1.4 Multivariate anomalies

Multivariate anomalies are data points that appear anomalous when considering relationships between at least two features. These may not be visible when analysing each feature independently. These can be challenging to detect in real-world data, as they may be hidden among relationships between multiple variables. Detecting these is especially important in industrial settings, where interactions between process variables can have a significant impact on the overall performance of the industrial process.

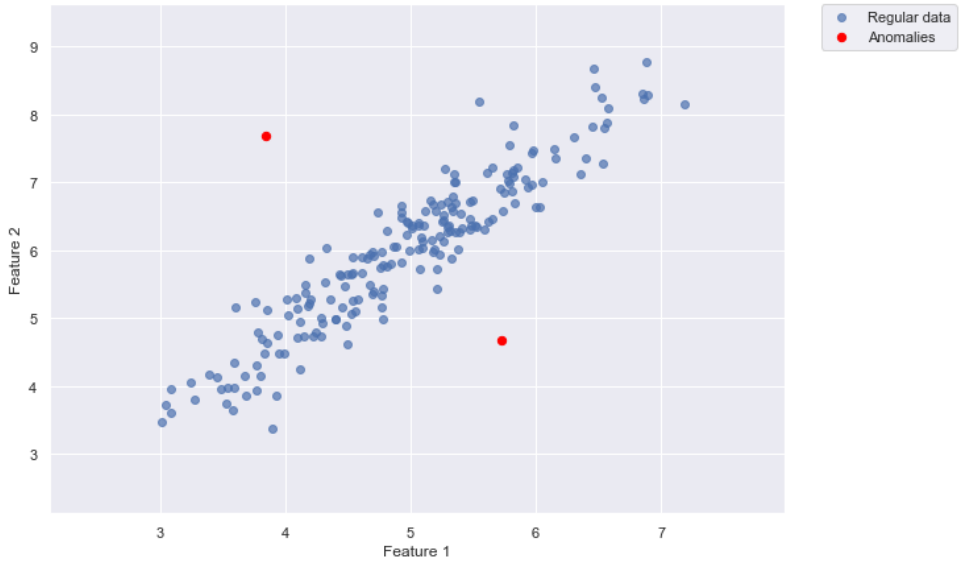


Figure 2.4: A figure illustrating multivariate (bivariate here) anomalies in synthetic data

2.2 Simple statistics and thresholds

Before the widespread adoption of machine learning techniques for anomaly detection, various statistical and threshold-based methods were used to identify outliers and irregular patterns in data. Such methods are often simple and interpretable, but may struggle to capture complex relationships between variables in the data and adapt to an evolving process and changing

conditions. Some of the most commonly used non-machine learning methods for anomaly detection include:

2.2.1 Threshold methods and statistical process control

Threshold-based methods involve manual setting of predefined limits for variables, and data points that exceed these thresholds are flagged as anomalies. These can be based on assumptions, experience, domain knowledge or other factors. Statistical process control is a collection of statistical methods for monitoring and controlling of industrial processes. Control charts plot process variables over time, and SPC uses statistical tests to determine if the process is running regularly or if any anomaly has occurred. The concept of statistical process control and the application of it in the industry can be contributed to Shewhart (1931) [4]. While these methods are easy to implement and effective for detecting changes in the mean or variance of a process, they are not effective when the anomalies are multivariate, or when non-linear patterns are introduced. Furthermore, they require manual tuning of thresholds and are not effective in identifying anomalies that result from subtle changes or complex interactions between variables.

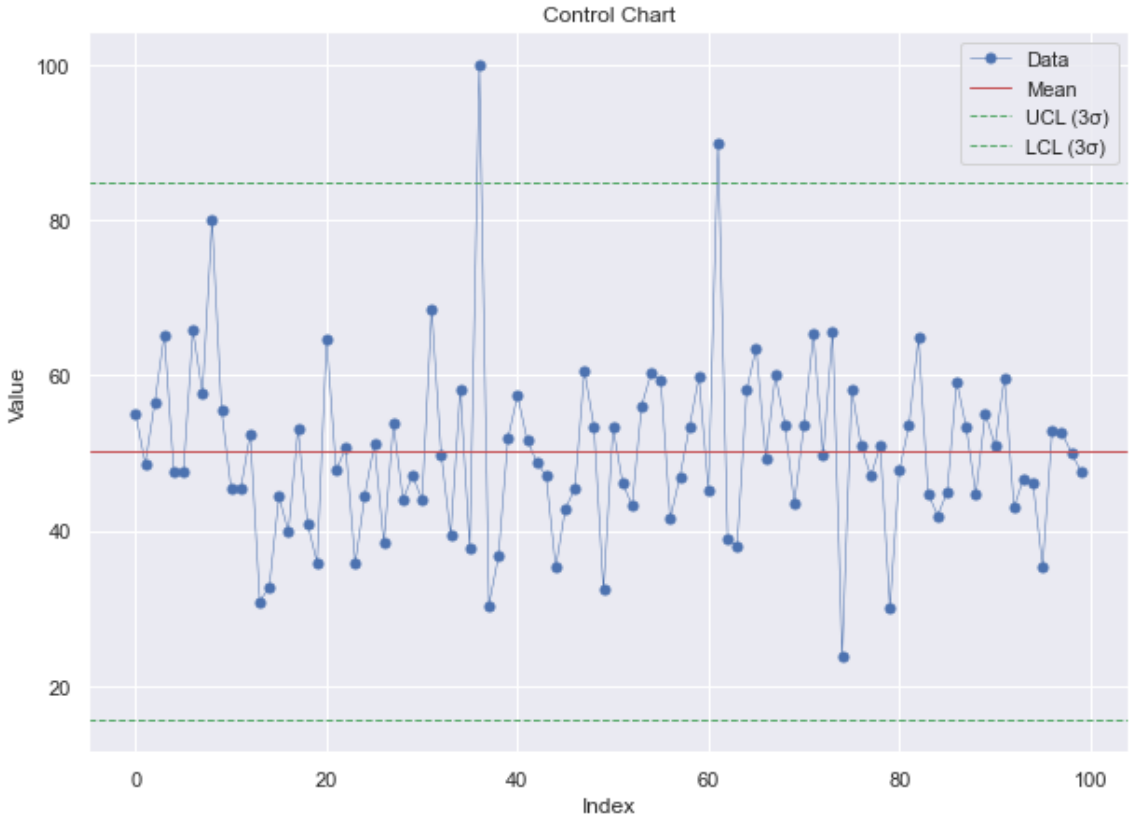


Figure 2.5: Control chart on randomly generated data

The control chart illustrates a threshold-based method for anomaly detection. The plot shows a randomly generated data set with anomalies, along with the mean, indicated by the red line, and upper and lower control limits, UCL and LCL, green dashed lines, based on 3 standard deviations from the mean. Data points exceeding the UCL or LCL are considered anomalies.

2.2.2 Z-score

Expressing in terms of the standard deviation, the Z-score measures how far a data point is from the mean of the whole dataset. Z-scores are a widely used statistical measure, making it

easier to compare values from different distributions [5].

$$Z = \frac{x - \mu}{\sigma}$$

Here, x is the data point, σ is the dataset mean and μ is the standard deviation. In the anomaly detection context, data points with high Z-scores, typically greater than 2, are considered outliers and are thus anomalous. It is a simple and widely used method, but is sensitive to extreme values, assuming that the data is normally distributed.

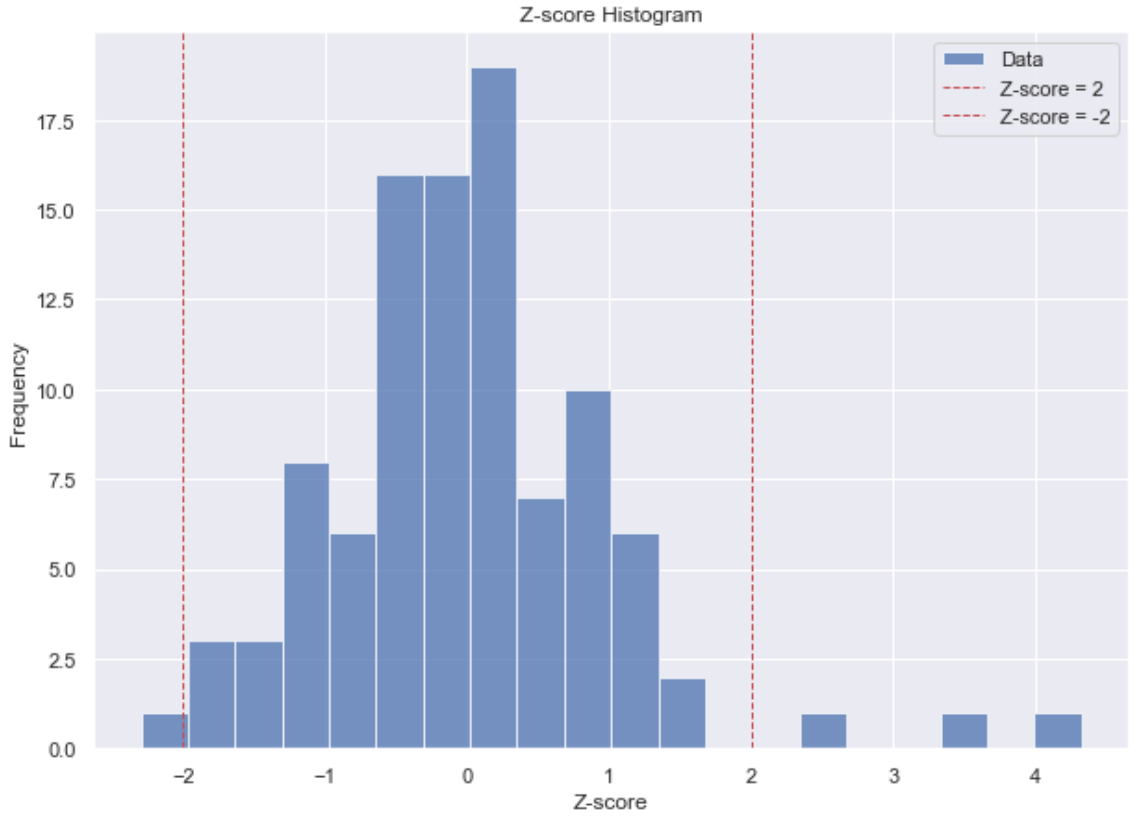


Figure 2.6: Z-scores of randomly generated data

The histogram illustrates the distribution of Z-scores of randomly generated data. The red dashed lines at Z-score = 2 and Z-score = -2 represent the threshold for anomaly detection. Data points with Z-scores greater than 2 or less than -2 are considered anomalies.

As an extension to the Z-score, Hotelling's T-square method is applied for multivariate data. The T-square method is a statistical measure that considers the correlation between variables, making it viable for multivariate data analysis [6].

2.2.3 Interquartile Range

The IQR measures statistical dispersion and is calculated as a difference between quartiles, representing the range between the first quartile, 25th percentile, and third quartile, 75th percentile. In the anomaly detection context, it can be used to identify outliers, by calculating the range within which most of the data points are expected to fall [7]. Typically, values above the third quartile plus 1.5 times the IQR, as well as values below the first quartile minus 1.5 times the IQR are considered outliers [8]. The IQR method is a robust measure of variability

and is less sensitive compared to standard deviation, making it better fit for non-normal data. However, it is not suitable for detecting subtle anomalies and multivariate, or time series data without heavy adaptation.

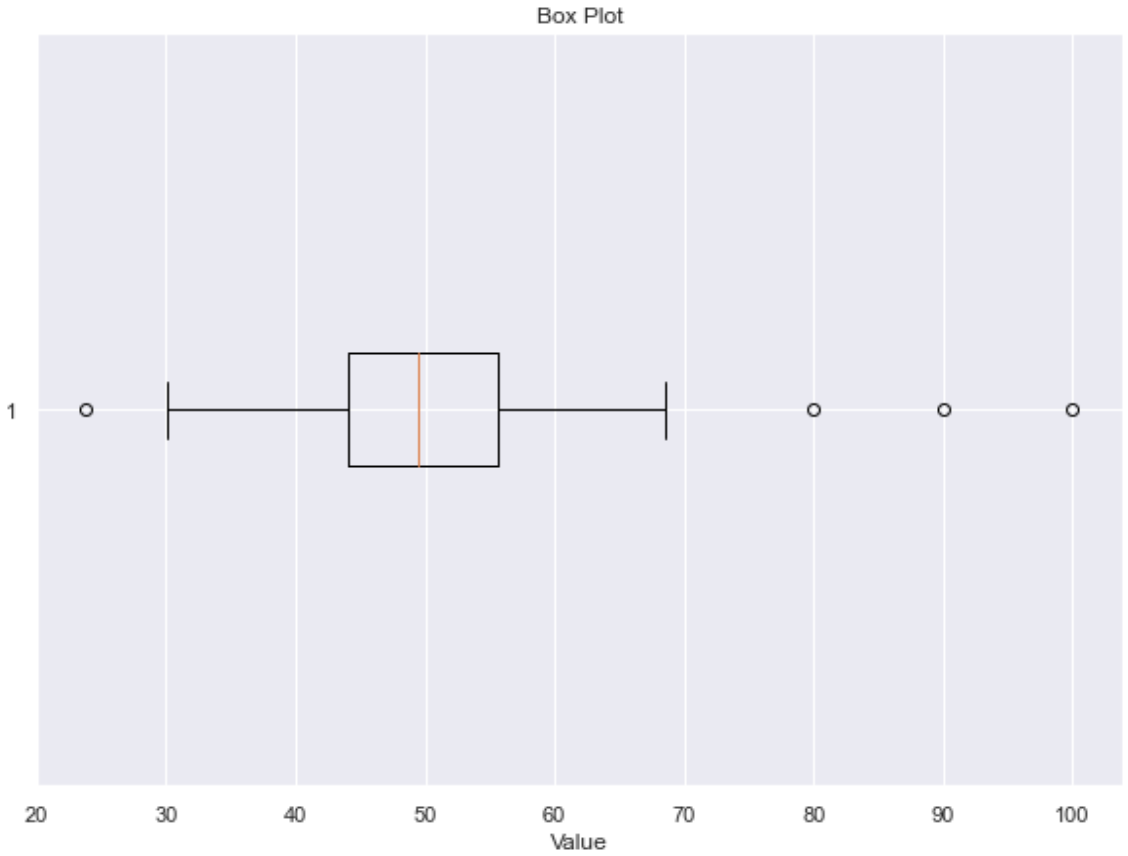


Figure 2.7: IQR method on randomly generated data

The box plot demonstrates the IQR method for anomaly detection. The box represents the interquartile range from the first quartile, Q1, 25th percentile, to the third quartile, Q3, 75th percentile. The line inside the box is the median. The whiskers extend to 1.5 times the IQR from the box. Data points outside the whiskers are considered anomalies.

2.3 Time series data

Time series data are a sequence of observations or measurements collected at regular time intervals. It is represented by a sequence of observations x_t, where x_t is the observed values at time t. The underlying processes in the data often have specific characteristics, such as trends, seasonalities and autocorrelations, which are to be accounted for in the potential analysis. The temporal ordering of time series data introduces unique challenges and opportunities for anomaly detection, and understanding there characteristics and features is important for the effectiveness of anomaly detection algorithms [9].

2.3.1 Trend

Trends are persistent movements in the data, over a long period of time. Trends can be upward, downward, or flat, showing the general direction of change in the data over a period of time. Identifying and modeling trends can be important to distinguish between variations and

anomalies in the data. A simple linear trend can be represented by a straight line in the form of $y_t = \alpha + \beta t + \epsilon_t$, where y_t is the observed value at time t, α is the intercept, β is the slope, and ϵ_t is the error term [10].

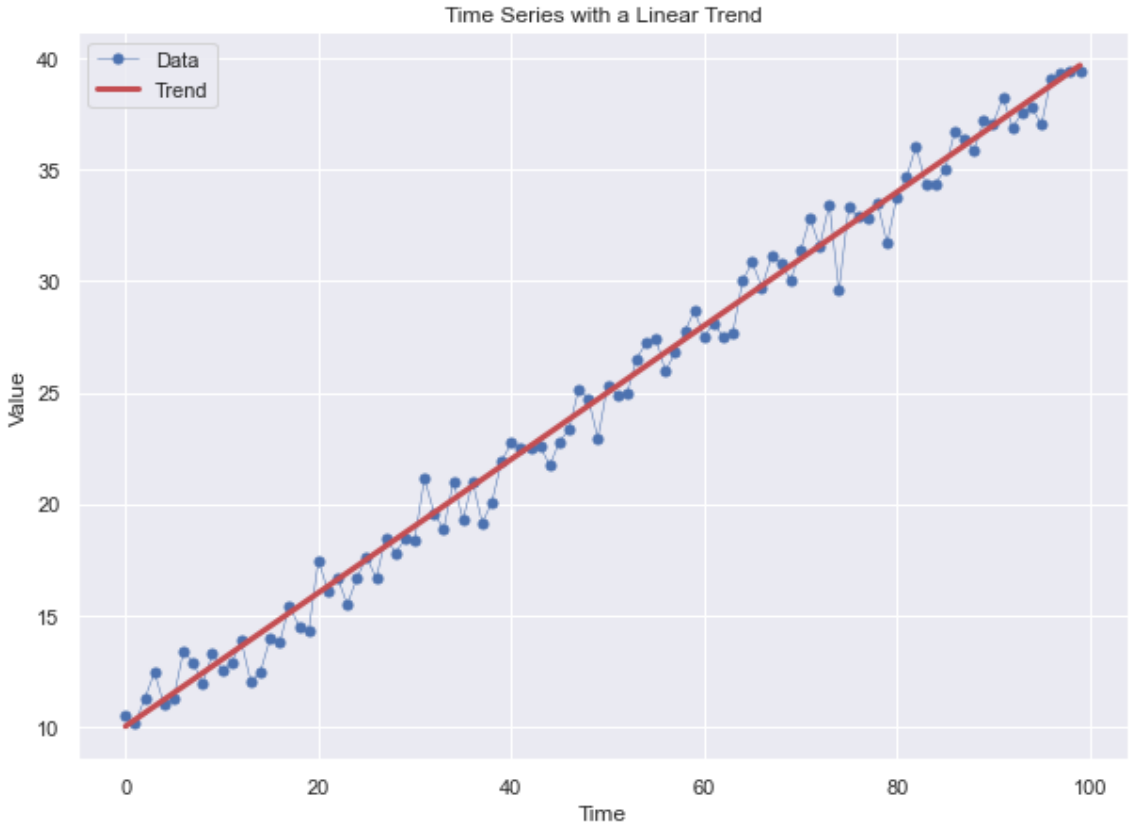


Figure 2.8: Trend in randomly generated time series data

The plot shows a synthetic time series with a linear trend. The blue points represent the observed values, and the red line represents the underlying linear trend.
In an industrial setting, an upward trend in pressure throughout the day might indicate a natural step in the process, while a downward trend in temperature might indicate an anomaly. Detrending techniques, such as differencing, can be applied to remove the trend from the data, making it easier to apply certain models.

2.3.2 Seasonality

Seasonality refers to regular, periodical and repeating fluctuations in the data. These are often related to as daily, weekly or annual cycles. Accounting for seasonality can be crucial for differentiating between normal and anomalous cycles in the data. Similar to the trend, decomposition techniques can be used to both isolate and remove seasonal components from data. Consequentially, this might assist anomaly detection algorithms by reduce the rate of false positives, caused by the periodical fluctuations.

Seasonality is the presence of regular, periodic patterns in a time series data. The seasonal component S_t can be represented as a function of time, with a fixed period p [10]:

$$S_t = f(t \mod p),$$

where f is a function that captures the seasonal pattern and $t \mod p$ represents the remainder of the division of t by the period p.

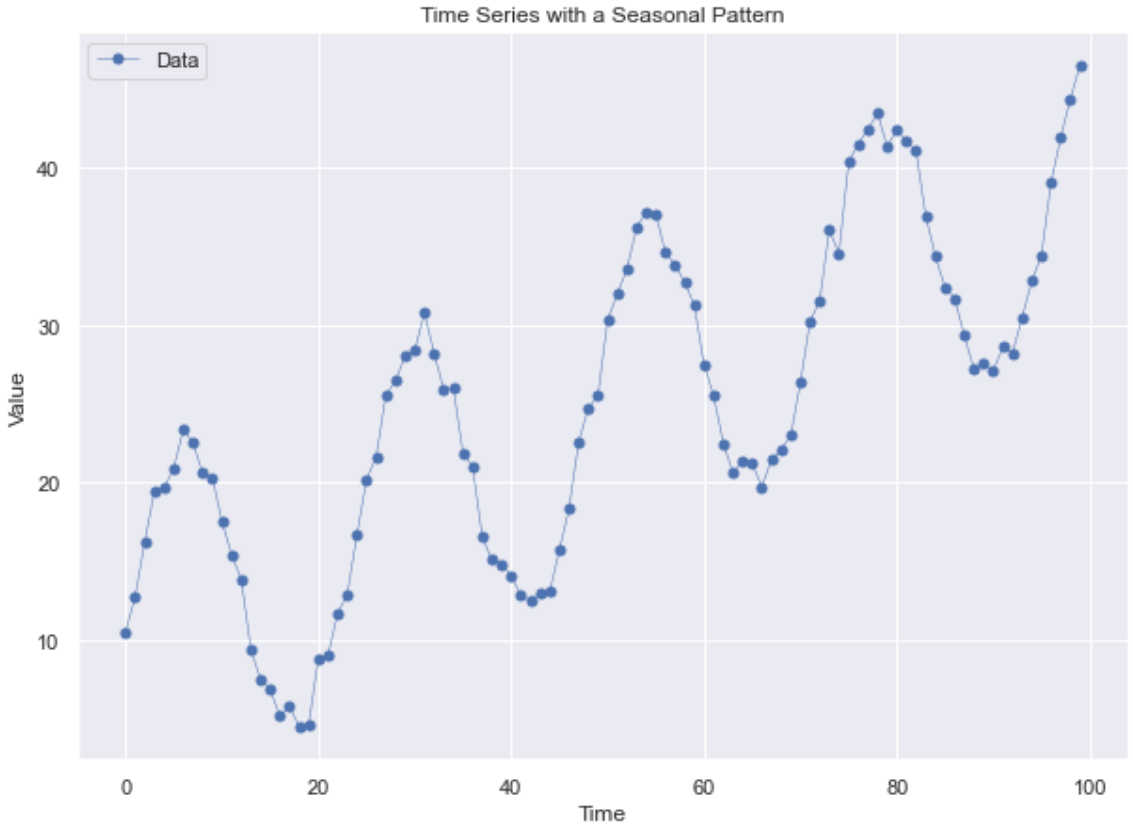


Figure 2.9: Seasonality in randomly generated time series data

The plot shows a synthetic time series with a seasonal pattern. The blue points represent the observed values, and the oscillation indicates the underlying seasonal pattern with a period of 24.

2.3.3 Autocorrelation

Autocorrelation simply refers to the correlation between a time and its own past values, measures a degree to which a given data point is related to data points at previous time steps. Analysis of autocorrelation can reveal important patterns and dependencies in the data. A strong autocorrelation, for example, indicates that values in a dataset are highly influenced by its recent past values. The existence of autocorrelation can be investigated using methods like the autocorrelation function (ACF) or the partial autocorrelation function (PACF).

The ACF is a measure of the correlation between a time series x_t and its lagged values x_{t-k}, where k is the lag. The autocorrelation at lag k, denoted as $ACF(k)$ or ρ_k, can be calculated as [10]:

$$\rho_k = \frac{\sum_{t=1}^{N-k}(x_t - \bar{x})(x_{t+k} - \bar{x})}{\sum_{t=1}^{N}(x_t - \bar{x})^2},$$

where $\bar{x}$ is the mean of the time series and N is the number of observations.

The PACF is the correlation between a time series and its lagged version after removing the effect of intermediate lags. The partial autocorrelation function (PACF) for a time series (x_t) at lag k can be represented as [10]:

$$\phi_{kk} = \text{Corr}(x_t - \hat{x}_t^{(k-1)}, x_{t+k} - \hat{x}_{t+k}^{(k-1)}),$$

where $\hat{x}_t^{(k-1)}$ and $\hat{x}_{t+k}^{(k-1)}$ are the linear least squares predictions of x_t and x_{t+k}, respectively, using lags $1, 2, ..., k-1$.

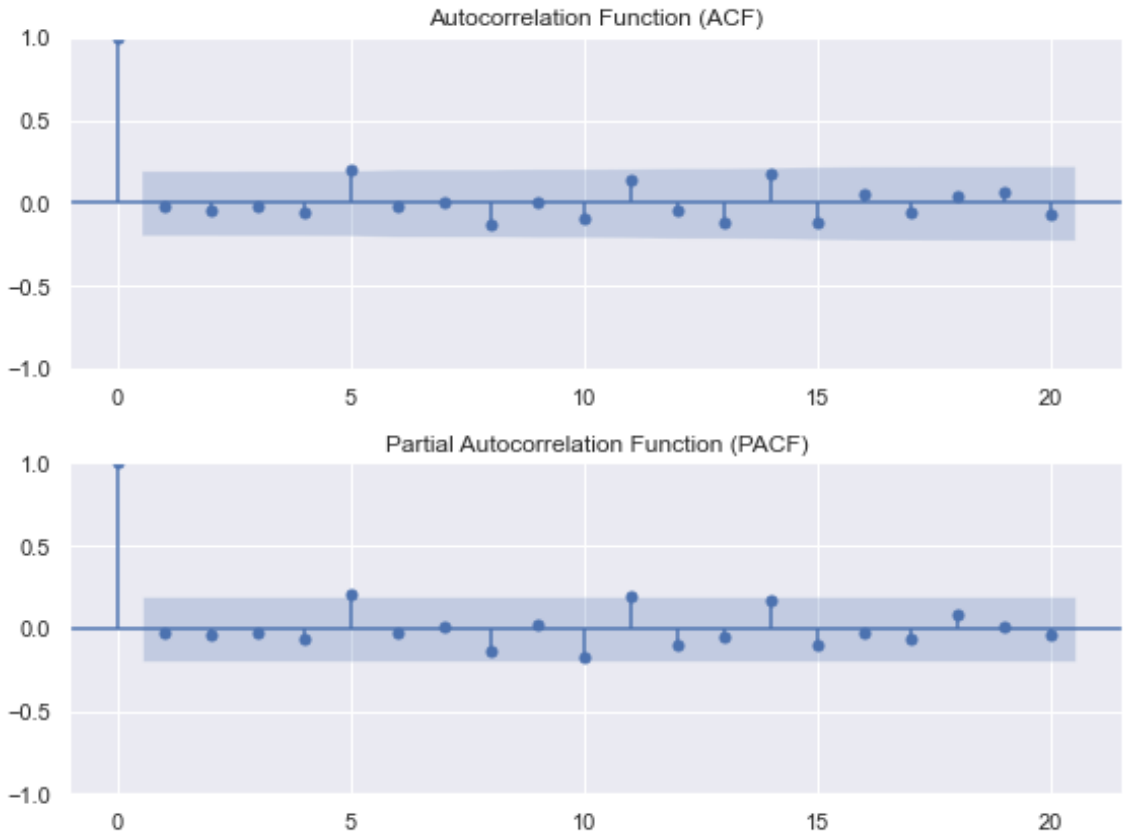


Figure 2.10: Autocorrelation in randomly generated time series data

The plots show the autocorrelation function, ACF, and the partial autocorrelation function, PACF, of a synthetic time series. The blue bars represent the correlation at each lag, and the blue shaded area represents the confidence interval under the null hypothesis of no autocorrelation. At lag 0, the autocorrelation is always 1 because a data point is perfectly correlated with itself. This is a fundamental property of the autocorrelation function. For the partial autocorrelation function, it also gives the value of 1 at lag 0.

These techniques can aid in determining the presence and degree of autocorrelation in the data and similar to trend and seasonality, accounting for autocorrelation, models can more accurately capture the actual underlying structure of the given data and improve their performance.

2.4 Types of machine learning

As a subfield of artificial intelligence, machine learning focuses on developing algorithms and models that learn from data, making decisions and predictions without explicit programming

[11]. It is is a powerful analysis tool for large data sets and complex data, and is useful is the task of anomaly detection [12]. Machine learning can be divided into three main types, supervised learning, unsupervised learning and reinforcement learning. In this thesis, focus will be laid on supervised an unsupervised learning, as reinforcement learning is not applicable for use on Bioco's industrial data.

2.4.1 Supervised and unsupervised learning

In supervised learning, an algorithm learns from a set of labeled data, consisting of input-output pairs, with the goal of learning a mapping from inputs to outputs, that then generalises well to unseen data [11]. This approach works well for tasks like classification, regression and forecasting with time series. For binary classification problems, we can define the labels as follows:

$$y_i = \begin{cases} 1 & \text{if } i \text{ is anomalous} \\ 0 & \text{else} \end{cases}$$

$D = (x_i, y_i)_{i=1}^n$ is a data set containing n data points, the goal of a supervised learning algorithm is to find a function $f(x)$ that maps the input features x_i to their corresponding labels y_i. Such a model is trained by minimisation of a loss function $L(y, f(x))$, measuring the difference between the predicted and the true labels.

In the context of anomaly detection, a supervised learning algorithm can be trained on a labeled data set with examples of both normal and anomalous data points, to then learn to classify data points as either anomalous or normal based on the data features. Supervised learning for anomaly detection can be challenging if the data is imbalanced, with a small amount of anomalies compared to normal data [12].

Advantages

- Because the model is trained on labeled data, high accuracy in identifying anomalies can be achieved, provided that the training data is representative and there are enough examples of normal data and anomalies [11]
- Clear decision boundaries between normal behaviour and anomalies can be learned, making interpretation of results easier

Disadvantages

- Getting labeled data can be expensive and sometimes impossible, especially in cases where anomalies are rare
- Model is more likely to overfit on the training data and not generalise well to new data with unknown anomalies

Unsupervised learning aims to discover structures and patterns hidden in the data, without the use of labeled examples [11]. It does not map inputs to inputs, but instead focuses on findings patterns, relationships or clusters within the data [12]. This is used in anomaly detection to identify deviating or unexpected patterns, indicating anomalies. Unsupervised learning is better fit for cases when obtaining labeled examples or a balanced data set is difficult.
The objective function here can be represented by minimising a measure of dissimilarity $d(x_i, x_j)$ or maximising a measure of similarity $s(x_i, x_j)$. The choice of this measure and function depends on the algorithm being used:

$$L(X) = \sum_{i=1}^{N} \sum_{j=1}^{N} f(d(x_i, x_j)) \quad \text{or} \quad L(X) = \sum_{i=1}^{N} \sum_{j=1}^{N} f(s(x_i, x_j))$$

Advantages

- Does not require labeled data, suitable for cases where acquiring labeled data is difficult
- Can potentially detect new or previously unknown types of anomalies that supervised methods might miss [12]

Disadvantages

- Relies on patterns and structures in the data, which might not be enough for effective anomaly detection. Thus, generally, lower accuracy is expected compared to supervised methods
- Results produced can be difficult to interpret, as clear decision boundaries between anomalies and normal behaviour are not provided. Thus, the interpretability is lower

2.5 Anomaly detection using machine learning

This section provides a theoretical overview of statistical and machine learning methods that can be employed for anomaly detection. Statistical methods normally assume that the data follows a specific distribution, and deviations are considered anomalies. On the other hand, ML techniques are widely used for anomaly detection, as they have the ability for adaptation to complex data structures and large data amounts.

2.5.1 Statistical methods

These methods are based on assumptions about the data distribution, and deviations from these distributions are considered anomalies [13]. The methodology involves fitting a statistical model to the data, and evaluating the likelihood of data points belonging to the model [14].

2.5.1.1 Gaussian Mixture Models

Probabilistic models assuming that the data is generated from a mixture of Gaussian distributions are called Gaussian mixture models [14]. Uses an expectation-maximization algorithm to estimate means, covariances and mixture weights of the components. Fits to the data and calculates the probability of each data point belonging to the model, where low scoring data points are anomalies. Can be used in both supervised and unsupervised manner depending on the task. Model might not be suitable for Bioco's data because of the assumptions the model makes, but is used to see if the underlying structure of the data can be described using it. The probability density function of a GMM can be written as:

$$P(x) = \sum_{i=1}^{k} \pi_i N(x|\mu_i, \Sigma_i)$$

P(x) represents the probability density of point x, π_i is the weight of the i-th Gaussian component, $N(x|\ \mu_i, \Sigma_i)$ is the Gaussian distribution with mean μ_i and covariance matrix Σ_i, and k is the number of Gaussian components [14].

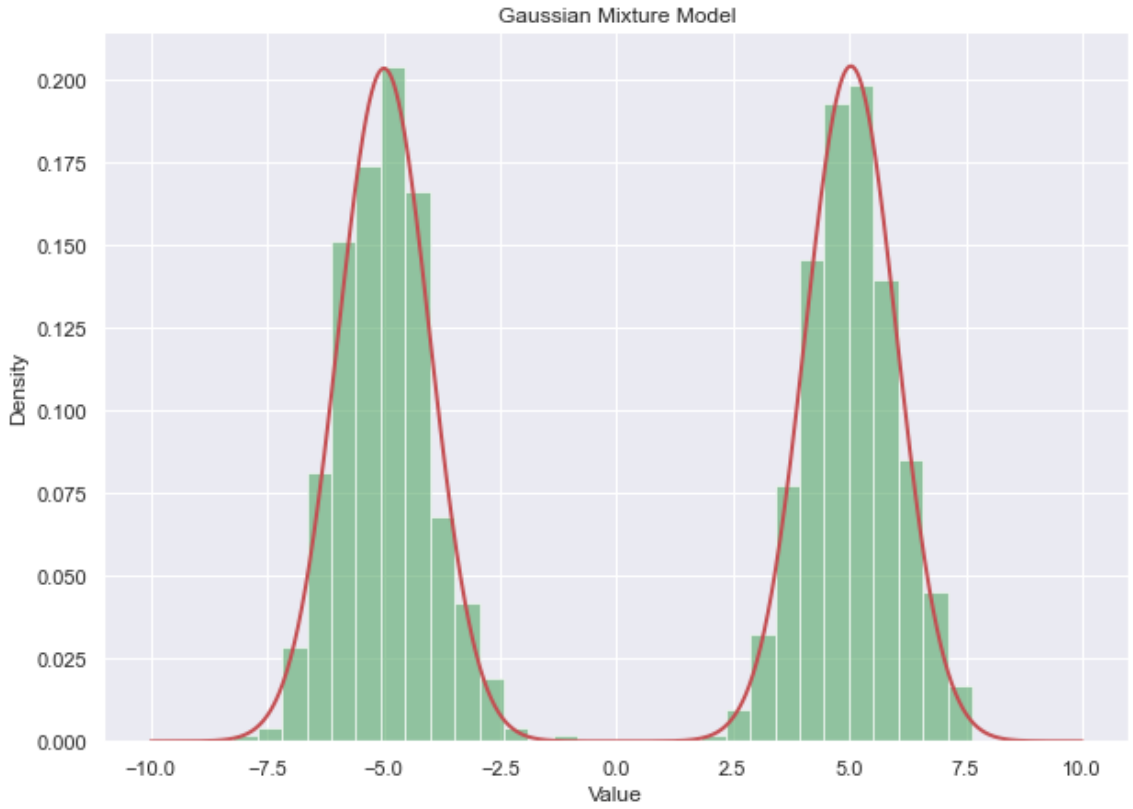


Figure 2.11: GMM model on randomly generated data

The histogram shows a synthetic dataset from a mixture of two Gaussian distributions, and the red line represents the probability density function of a Gaussian mixture model fitted to the data.

2.5.1.2 Autoregressive Integrated Moving Average (ARIMA)

ARIMA combines autoregression, differencing and moving average components. As a linear time series model, it is designed to work with stationary time series data [15]. Variations of the model incorporate differencing to remove seasonality and trends. Can be used for anomaly detection by forecasting future values based on historical data and comparing new actual values to the predicted ones. Strong deviations indicate anomalies, when data does not follow the regular pattern. For on-line applications, the model can be continuously fit on new data as it comes in, making constant forecasts and comparisons. Choice of model parameters (p,d,q) is critical for performance [15], where p is the autoregressive (AR) order, d is the degree of differencing and q is the moving average (MA) order. These are determined using various tests, and some variations of the model, like auto ARIMA, do this automatically. Deep understanding of the parameters is not relevant for our application.

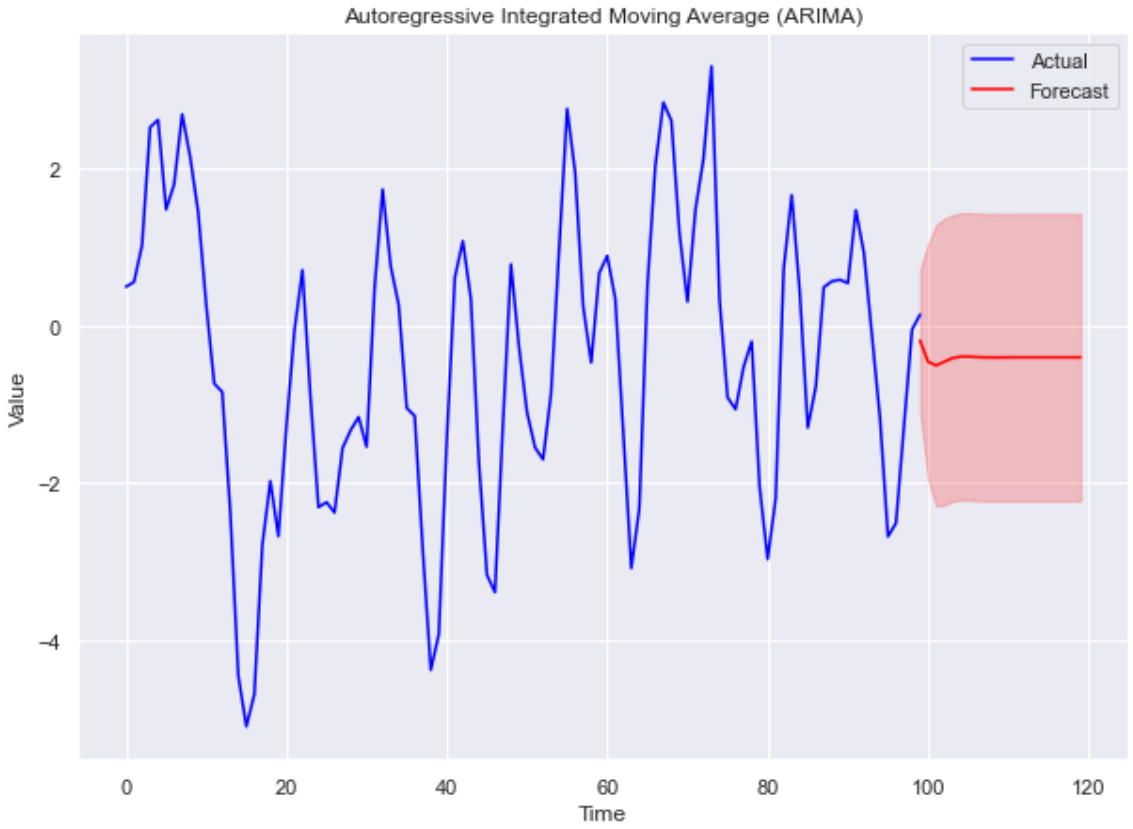


Figure 2.12: ARIMA model on randomly generated time series data

The plot illustrates the application of an ARIMA model on a synthetic time series. The actual values of the time series are plotted in blue, while the forecasted values are represented in red. The red shaded area surrounding the forecast represents the confidence interval, providing a range within which the future values are likely to fall. This showcases the ARIMA model's capacity to predict future data points based on historical data. Any deviations from the forecasted range are potential anomalies.

2.5.2 Clustering methods

Clustering methods, as an unsupervised learning technique, groups similar data points together into clusters based on features [14]. For anomaly detection, data points not belonging to any cluster, or too distant from cluster centroids are indentified as anomalies.

2.5.2.1 K-means

Simple technique partitioning data into K clusters based on features' mean values [16]. Initially starts with K centroids, iteratively updates centroids by minimising sum om squares distances between data and cluster centroids [14]. Data points far from assigned clusters are considered anomalous, deviating from general pattern. K-means allows for definition of thresholds for anomaly flagging. Algorithm is sensitive to the initial centroids [14], as well as spherical shape and equal size assumption about the clusters.
K-means aims to partition a dataset $X = \{x_1, x_2, \ldots, x_n\}$ into K distinct clusters, where each data point x_i belongs to the cluster with the nearest mean. The objective function to minimize sum of squares [16] within clusters :

$$J(C_1, C_2, \ldots, C_K) = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2 \tag{2.1}$$

where C_k represents the k-th cluster, and μ_k is the mean of the data points in cluster C_k. Updates the cluster assignments and mean iteratively until convergence. This process can be summed up in two steps:

1. **Assignment step:** Assign each data point x_i to the nearest cluster centroid:

$$C_k = \{x_i : \|x_i - \mu_k\|^2 \leq \|x_i - \mu_j\|^2 \ \forall j, 1 \leq j \leq K\} \tag{2.2}$$

2. **Update step:** Cluster centroids are updated by computing the mean of the points assigned to a cluster:

$$\mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i \tag{2.3}$$

Figure 2.13: K-means clustering on randomly generated data

The plot shows the result of K-means clustering on a synthetic dataset. The colored points represent data points assigned to different clusters, and the large red points are the centroids of the clusters.

2.5.2.2 DBSCAN

Density-based spatial clustering of applications with noise, an algorithm that groups data points based on density and proximity [17]. DBSCAN is more flexible since it does not require a prespecified amount of clusters like k-means. Thus, it is seen as the more flexible clustering algorithm for finding complex cluster structures. DBSCAN examines densities and proximities of data points, categorizing them based on local properties. Points that do not fit into dense regions are potential anomalies. The algorithm can be summarised as follows:

1. For each data point x_i, compute the number of neighbouring data points within a distance of ϵ. These are denoted as $N(x_i)$.
2. If $|N(x_i)| \geq MinPts$, x_i is marked as a core point. Otherwise, mark x_i as a border or noise point.
3. For each core point, if it is not already part of a cluster, create a new cluster and recursively add all its density-reachable points. Two points are density-reachable if there is a chain of core points between them, where each core point is a neighbour of the next core point in this chain.
4. Finally, assign border points to their nearest core point's cluster.

Main parameters of DBSCAN are ϵ and $MinPts$. Choice of these significantly affects the performance of the algorithm. Might require trial and error to find the optimal hyperparameters. DBSCAN can find clusters of arbitrary shapes, separating noise[17], making it robust for anomaly detection. Lacks performance on data with variying density, where ϵ and $MinPts$ are not optimal for all regions in the feature space.

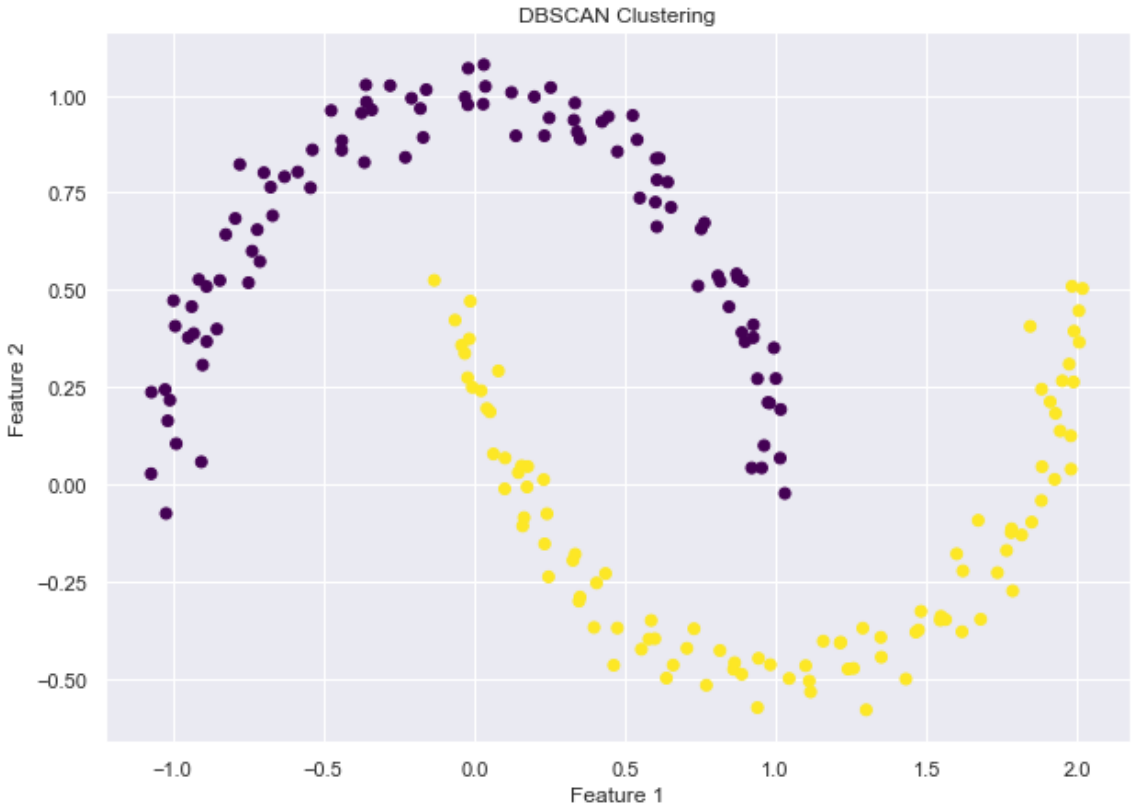


Figure 2.14: DBSCAN clustering on randomly generated data

The plot shows the result of DBSCAN clustering on a synthetic dataset. The colored points represent data points assigned to different clusters.

Furthermore, the next figure showcases the use of DBSCAN for anomaly detection in a different synthetic data set, mixed of normally distributed data points in three clusters and randomly placed anomalies.

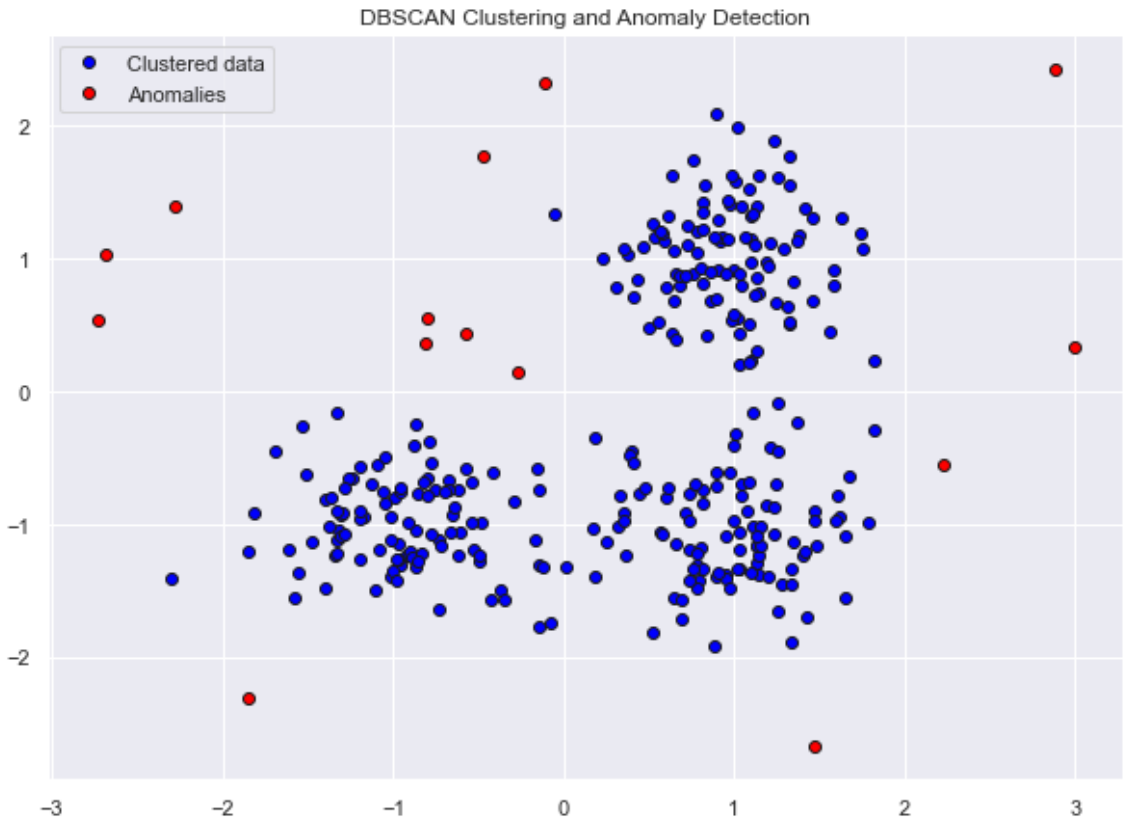


Figure 2.15: Anomaly detection using DBSCAN on randomly generated data

The blue points represent data points assigned to different clusters, while the red points represent the identified anomalies. This highlights the ability of DBSCAN to distinguish between clustered data and sparse anomalies.

2.5.3 Nearest neighbour methods

Nearest neighbour methods rely on between-point distance to determine similarity [18]. Used for both supervised and unsupervised learning tasks, depending on the data.

2.5.3.1 K-nearest neighbours

Non-parametric method used for classification and anomaly detection. Assumes that similar data points are located close to each other on the feature space. k-NN selects k closest neighbour by calculating distances between data points [18]. Average distance to the k-neighbours gives anomaly score. High anomaly score indicates that a point is likely an anomaly. Allows for definition of thresholds for classification of anomalous points. k-NN's strength is its simplicity, but the algorithm suffers from high computational complexity on large data sets, as it calculates distances between a point and all other points. The algorithm uses Euclidean distance $d(x, y)$ to measure the distance between two data points x and y in the feature space:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

The average distance to k nearest neighbours of the data point x is then computed as the anomaly score $S(x)$:

$$S(x) = \frac{1}{K} \sum_{i=1}^{K} d(x, y_i)$$

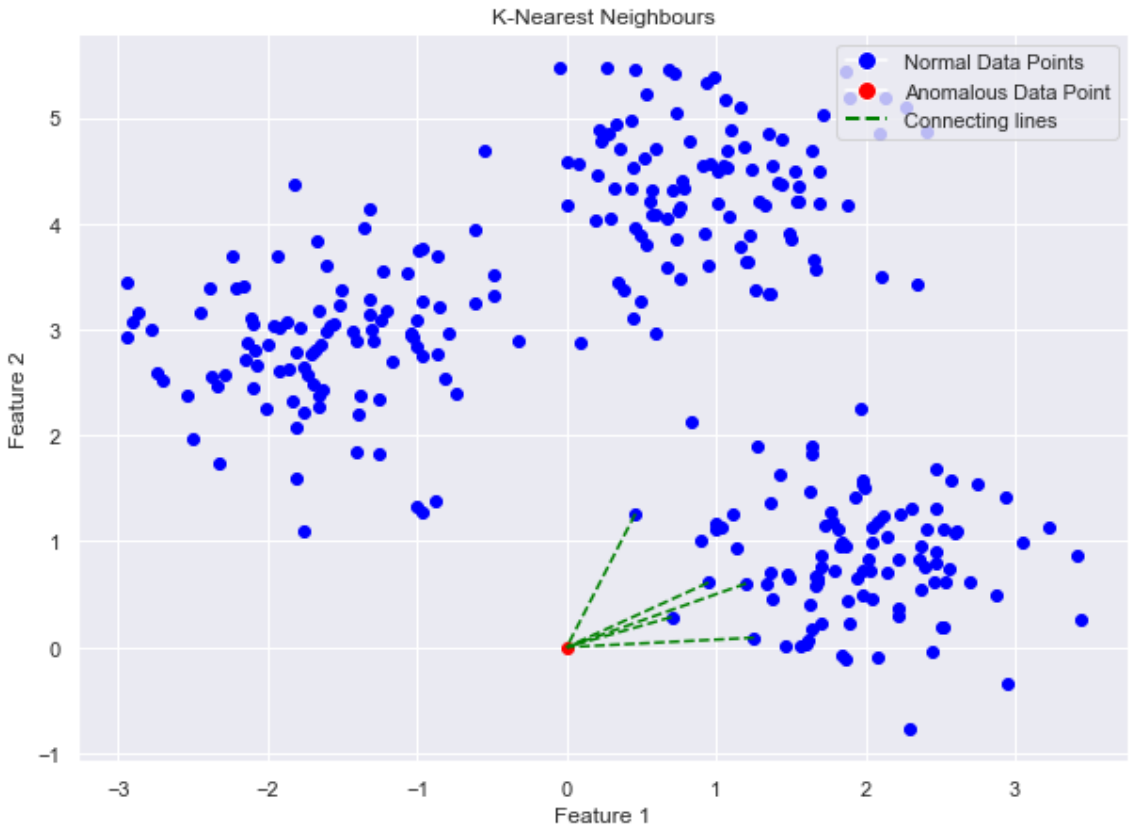


Figure 2.16: Application of k-nearest neighbours on randomly generated data

The plot shows the result of application of k-nearest neighbours on a synthetic data set. The blue points represent data points, the red point is an anomaly, and the green dashed lines connect the anomaly to its five nearest neighbours.

2.5.3.2 Local outlier factor

Density-based method for anomaly detection, working around local density for data in the feature space. Compares the ratio between the average local density of a data point's k nearest neighbours to its own local density, calculating a LOF score. It represents how much the local density of a data point differs from its neighbours. High LOF score for a data point indicate that it is located in a sparse region of the feature space, and thus has a significantly lower local density than the neighbours. Local outlier factor is particularly good for anomaly detection in data with varying densities, because it considers the local context. Choice of the k parameter

can have a significant impact and is often crucial for optimal performance [19].

Consider a data point x for which the LOF score is to be computed. The reachability distance $reachdist(x, y)$ between x and another data point y in LOF is defined as the maximum of the distance $d(x, y)$ and the k-distance of the data point y, which is the distance from y to its k-th nearest neighbour.

$$reachdist(x, xy) = \max\{d(x, y), k{-}\text{dist}(y)\}$$

The local reachability density $\text{lrd}(x)$ of a data point x is the inverse of the average reachability distance from x to its k nearest neighbours [19]:

$$\text{lrd}(x) = \frac{1}{\sum_{y \in N_k(x)} reachdist(x, y)}$$

Here, $N_k(x)$ represents the set of k nearest neighbours of the data point x. As the final step, the local outlier factor $\text{LOF}(x)$ is calculated as the average ratio of the local reachability densities of the k nearest neighbours of x to its own local reachability density:

$$\text{LOF}(x) = \frac{\sum_{y \in N_k(x)} \frac{\text{lrd}(y)}{\text{lrd}(x)}}{|N_k(x)|}$$

In this context, x is the data point for which LOF score is computed, while y represents the other data points in the dataset. Calculations start from x and consider the relationship between x and its k nearest neighbours (the set $N_k(x)$), which are the data points y.

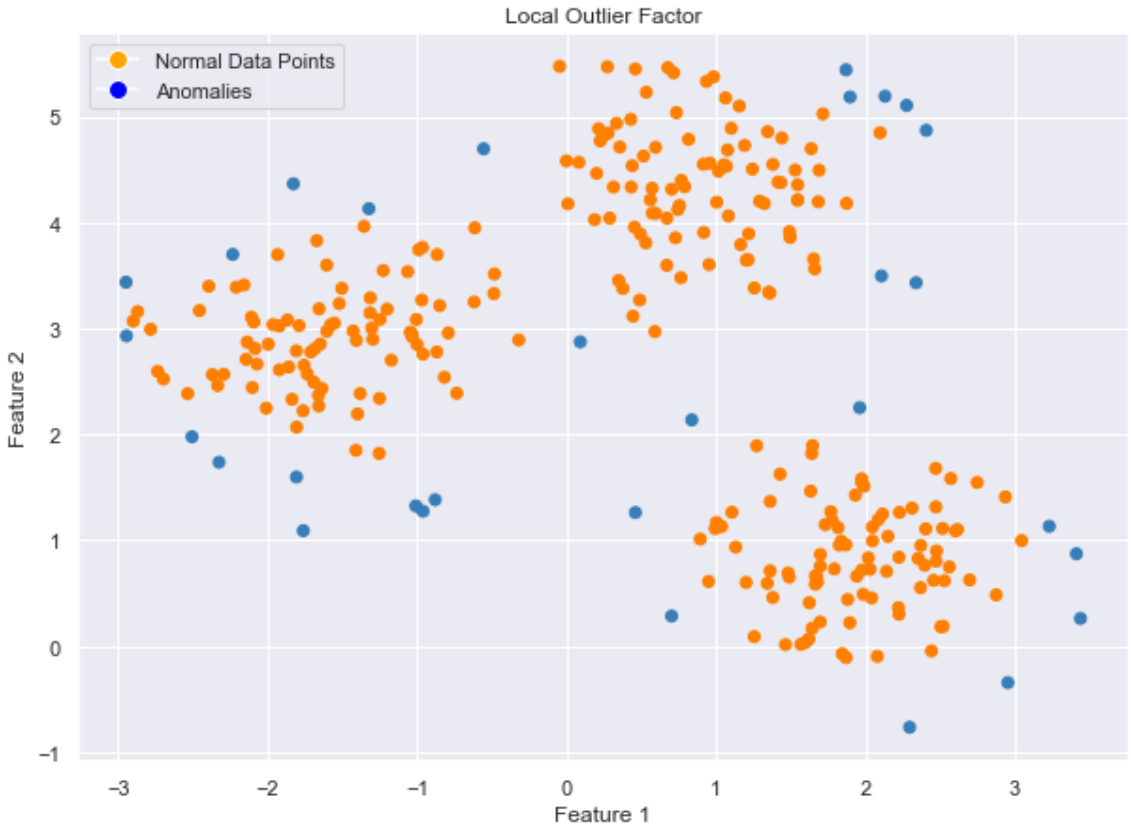


Figure 2.17: Application of LOF on randomly generated data

The plot shows the result of application of LOF on a synthetic data set. The orange points represent normal data points, and the blue points are considered as anomalies by the LOF model.

2.5.4 Decision tree based methods

Decision tree algorithms rely on tree structures, representing decisions and outcomes [20], and are used both for supervised and unsupervised tasks . Handle complex relationships between features well and the results are often easy to interpret, making them well suited for anomaly detection. Work through recursively partitioning the input spaces, making a series of binary splits on the data features. Results in tree-like structures, with leaf nodes corresponding the predictions or decisions [20]. In anomaly detection, a decision boundary is constructed around the normal data points, separating normal instances from anomalous points. Decision tree methods are robust to noise, missing values and are able to process high-dimensional data.

2.5.4.1 Isolation forest

An unsupervised anomaly detection algorithm, based on construction of multiple decision trees [21]. Randomly selects a feature and splits data based on a randomly chosen value for the feature. Isolation forest repeats this process recursively until all the data are isolated, forming a "forest" of trees. The algorithm does not make assumptions about the underlying data distribution, instead finding anomalies by comparing the path lengths of the data in the forest. Data points with shorter average path lengths in the trees are anomalies, as they are isolated more quickly than normal data [21]. A robust method with low memory requirement, effective for anomaly detection in high-dimensional and diverse data [21].

The path length $h(x)$ of a data point x in an isolation tree represents number of edges from the root node, to the node containing the data point. Essentially indicates how quickly a point of data can be isolated compared to others. It can be calculated as:

$$h(x) = \sum_{i=1}^{n} \frac{1}{2^{\lceil \log_2(n_i+1) \rceil}} \tag{2.4}$$

where n is the number of trees in the forest and n_i is the number of data points in the subtree rooted at node i. Then, the anomaly score $s(x)$ for a data point x is computed as:

$$s(x) = \frac{2^{-\frac{E(h(x))}{c(N)}}}{(1 - 2^{-\frac{1}{c(N)}})^{-1}} \tag{2.5}$$

where $E(h(x))$ is the average path langth of x across all of the trees. N is the total number of data points in the data, and $c(N)$ is the average path length of an unsuccessful search in a binary search tree. An unsuccefull search is defined as the search algorithm attempting to find a target value in a binary search tree, but with the value not existing in the tree. In the end, the average path langth of an unsuccessful search is used as a reference for comparing isolation abilities of data points.

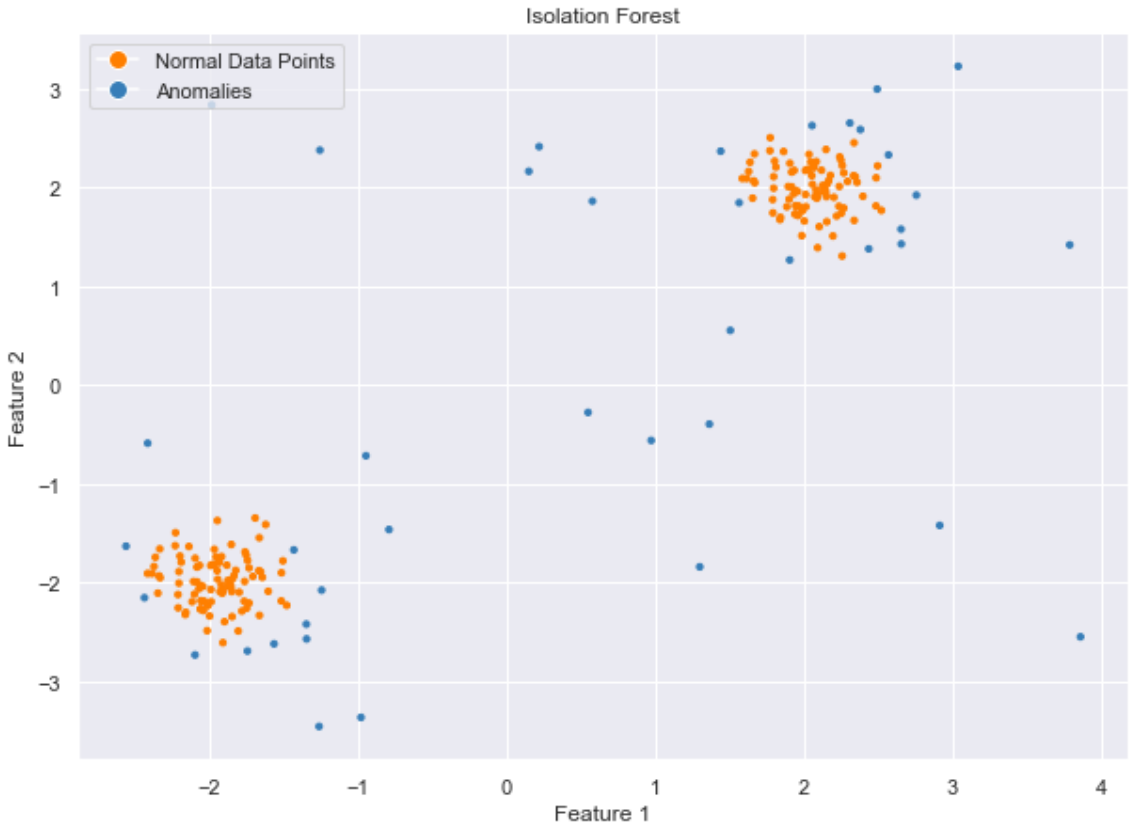


Figure 2.18: Isolation forest on randomly generated data

The plot illustrates the performance of an isolation forest model on a synthetic data set. The normal data points are shown in orange, while the anomalies are represented in blue. The model effectively separates the normal data points from the anomalies, as can be seen in the plot.

2.5.5 Deep learning and neural network methods

Deep learning and neural network methods are methods that have the ability to learn complex, non-linear patters from large data sets with vast variety [12]. Recently, these methods have gained popularity because of public availability and ease of use of some applications built on deep learning and neural networks. Flexible methods that are also applicable to anomaly detection tasks, where models are trained to recognise normal patterns and flag any deviations from these as anomalies.

2.5.5.1 Autoencoders

Unsupervised neural networks, learning to encode and decode the input data. Autoencoders effectively learns a representation of the data of a lower dimension [12]. For anomaly detection, the algorithm is trained on normal data, and the reconstruction error between the input and the decoded output is then used as an anomaly score [22]. When these errors are high the autoencoder fails or struggles with reconstruction of the input, indicating that anomalous data is present. Autoencoders have shown to be effective on anomaly detection in high-dimensional data, as they learn a compressed representation of the data, capturing the most relevant features. However, autoencoders may require large amounts of training data and resources for optimal performance. Anomaly scores for a given data point x are calculated by computing its reconstruction error, using a trained autoencoder:

$$S(x) = L(x, g(f(x))) \tag{2.6}$$

where $S(x)$ represents the anomaly score of the data point x, and $L(\cdot)$ is the loss function used to measure the reconstruction error.

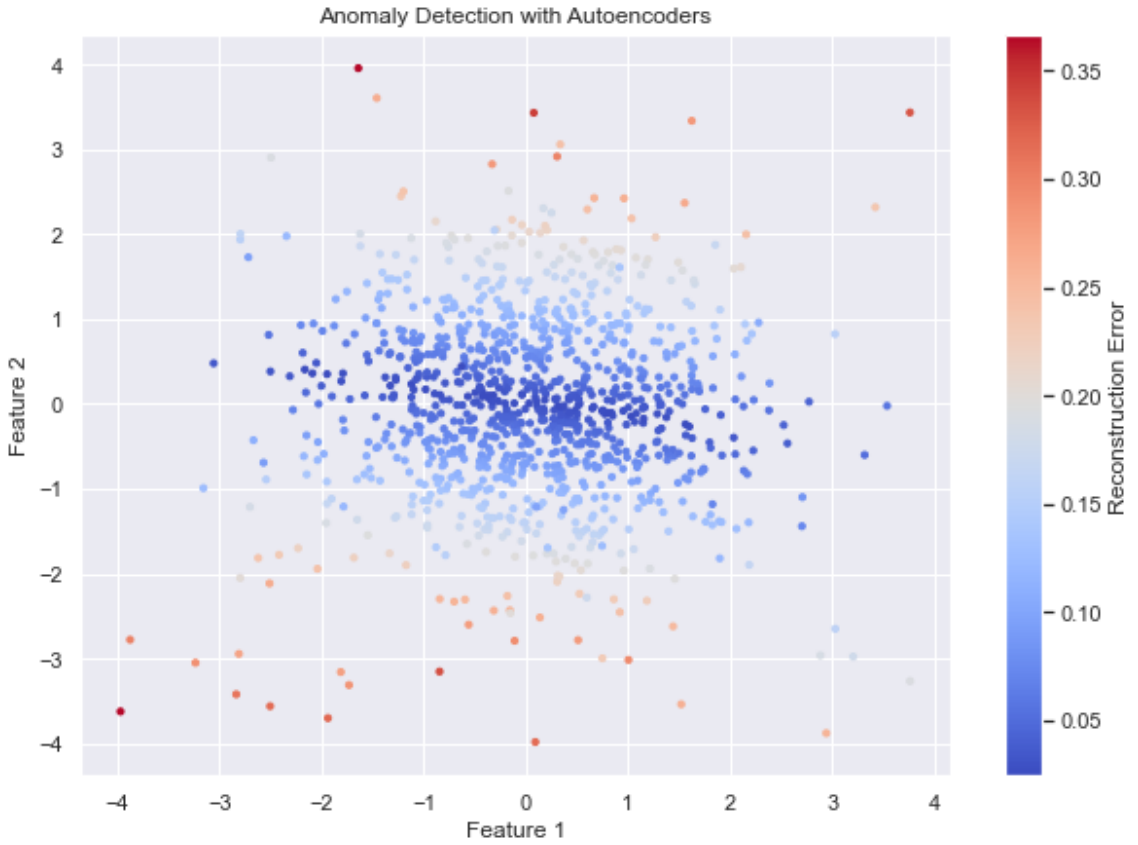


Figure 2.19: Application of an autoencoder on randomly generated data with a reconstruction error bar

The plot showcases anomaly detection using autoencoders. The normal data points are plotted with the colour representing the reconstruction error. The lower the reconstruction error, the more likely it is that the data point is normal. The anomalies, located outside the range of the normal data, are also coloured according to their reconstruction error. The anomalies have a higher reconstruction error because the autoencoder was not trained on these types of data points.

2.6 Related work

Many anomaly detection techniques have been proposed in the literature, as it has been widely studied in different fields. A comprehensive study on various anomaly detection techniques was done by Chandola et al. [1]. Statistical, clustering, classification-based methods are discussed and different types of anomalies are presented. Challenges in this field were also addressed.

Deep learning-based approaches were discussed by Chalapathy and Chawla [23] in a more recent survey about anomaly detection. Importance of deep learning in processing of high-dimensional data, as well as the use of different architectures, such as autoencoders and recurrent neural networks for anomaly detection purposes were discussed. Furthermore, future research directions

for deep learning-based anomaly detection methods were considered. As another deep learning approach, Malhotra et al. [24] proposed an encoder-decoder framework based on Long Short Term Memory networks for anomaly detection. Effectiveness of their approach was demonstrated on several data sets, including aerospace and automotive industry data.

Goldstein and Uchida [25] conducted an evaluation of different unsupervised anomaly detection algorithms. Statistical, clustering, nearest neighbour and subspace methods were considered. A framework for evaluation was proposed and advantages and disadvantages of every method were discussed, based on results of application to multivariate data.

The related works demonstrate continued interest in improvement and development of anomaly detection techniques for various types of applications. The exploration of new approaches and the application of current machine learning techniques provides a solid foundation for further research and advancements in the field of anomaly detection.

2.7 Summary

An overview of the theoretical background for anomaly detection in an industrial setting and related work has been provided in this chapter. The theory for time series data, statistical, non-machine learning and machine learning methods have been discussed. Furthermore, the challenges associated with these, as well as practical implementation considerations and evaluation of anomaly detection solutions were presented. This foundation of information will serve as the theoretical basis for the anomaly detection techniques suggested in this thesis.

Chapter 3

Data Exploration

This chapter presents the data exploration process for the industrial time series data provided by Bioco. A crucial first step for the analysis, particularly for complex industrial processes with interrelated variables. Ideally, the raw data is to be examined, cleaned and transformed into a more structured and understandable format, paving the way for further analysis and anomaly detection. A comprehensive overview of the data, its variables and structure is provided. For confidentiality reasons, some of the data description, plots and relations are altered, and an exact representation of the distributions in the data is not provided.

3.1 Data description

For the purpose of the thesis, 8 data sets were provided by Bioco. These comprise of various dimensionalities, with the data representing sensor measurements collected at various points in the industrial process, in the years 2020, 2021 and 2022. These include flow rates, pressures, temperatures and other critical process parameters. In general, variables called TT are temperatures, PT are pressures and F are flow rates. Additionally, the data includes a column that tracks the time during the process. The data sets are structured as follows:

Table 3.1: Data set dimensionalities

Dataset Name	**Rows**	**Columns**	**Sampling Frequency**
Process_2021	873105	37	Varying between 60 and 10 sec
Process_44	8640	49	1 min
Process_45	8640	49	1 min
Process_46	8640	49	1 min
Process_47	8640	49	1 min
Process_48	8640	45	1 min
Process_49	8640	45	1 min
Process_50	8640	45	1 min

The columns represent the different sensors placed across the industrial process, while the rows represent sensor measurements at a given time point.

The significant difference in data points in the 2021 data set is explained by the nature of the collection of that data, as well as the sampling frequency. The Process_2021 data set encompasses continuous sensor data from November 2020 until the end of March 2021 with a varying frequency, while the 2022 data, data sets Process_44-Process_-50, was collected on the weeks

corresponding to the data set name, at a lower frequency of sampling. Thus, the Process_2021 data set includes a significant amount of irrelevant data, such as days when the industrial process was not running. This results in varying, more extensive preprocessing requirements for 2021 data.

3.1.1 Clogging events

In addition to the data, dates and timestamps of previous clogging events were provided by Bioco. These are essential, serving as an evaluation benchmark for the performance of anomaly detection models. The following table presents the known clogging events at Bioco in 2021 and 2022:

Table 3.2: Known clogging events in 2021 and 2022

Date	Time	Clog Index
12/01/2021	05:45	clog 1
01/02/2021	08:40	clog 2
24/02/2021	08:00	clog 3
01/03/2021	17:00	clog 4
05/12/2022	15:05	clog 5
13/12/2022	01:35	clog 6

3.2 Visualisation

As Bioco's process is continuous, rest raw materials from poultry are constantly fed into the system, with the whole process taking approximately 60 minutes. The sensors collecting measurements are located at different points throughout the process, which gives a global overview of the process parameters at a time point. This means that a data point in a feature x and y at a time point t do not represent the same material, but rather the property of the material at the location of the sensor. This can be dealt with in a multitude of ways, discussed later. A sketch of the process, including an approximate timeline:

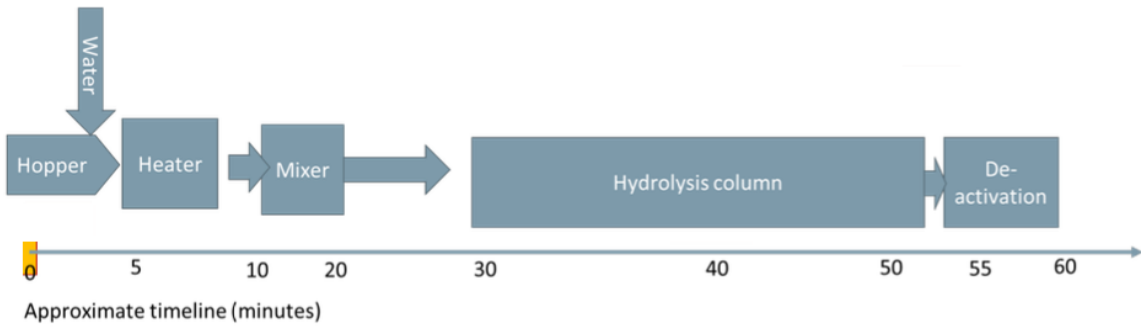


Figure 3.1: Process sketch with an approximate timeline

The hopper allows the process to have an even and steady flow rate, the heater warms raw material to a working temperature, the mixer blends the material, the hydrolysis column performs enzymatic hydrolysis and the deactivator heats the material and neutralises enzymes.

The clogging events showcased in table 3.2 are known to occur between the end of the hydrolysis column and the deactivation. This is significant in the context of anomaly detection, giving a

long period of time for analysis of the raw material going through the system. This property is less important if a clogging is an effect of a slow buildup or organic matter in the pipes, but plays a core role if it is caused by the properties of the raw material fed into the system.

A visualisation of selected sensors in the raw data is provided in the following figure to provide an opportunity to examine the inherent structure and patterns in the data. For confidentiality reasons, the y-axis is hidden:

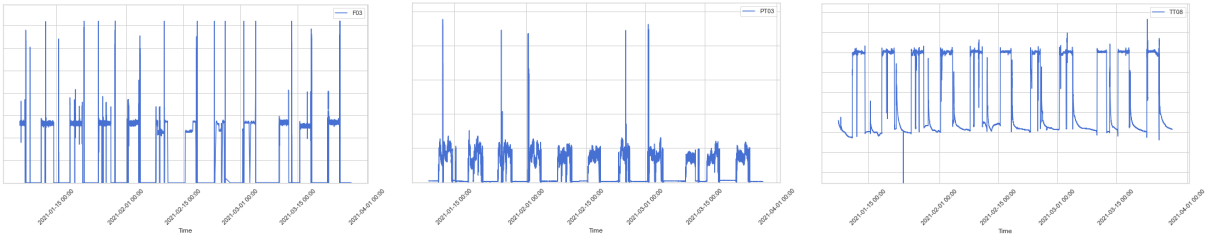


Figure 3.2: Plot for sensors F03, PT03, and TT08 in the Process_2021 dataset, representing flow rates, pressures and temperatures. F03 and PT03 are located in the hydrolysis column, while TT08 is in the mixer 3.1.

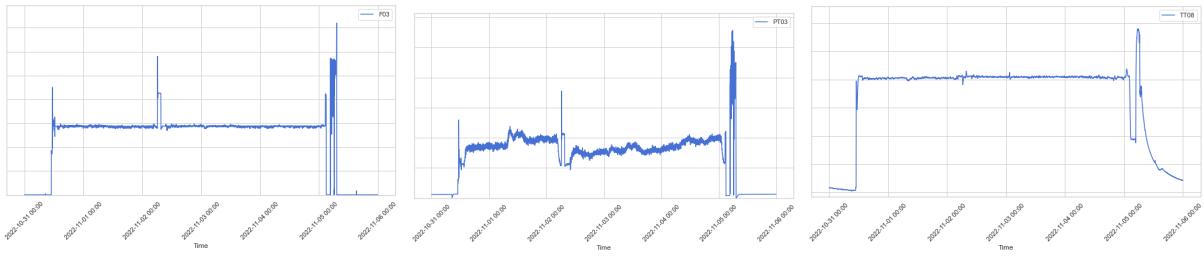


Figure 3.3: Plot for sensors F03, PT03, and TT08 in the Process_44 dataset, representing flow rates, pressures and temperatures. F03 and PT03 are located in the hydrolysis column, while TT08 is in the mixer 3.1.

The plots showcase the distribution of values in three of the multitude of sensors in the data sets, with data in figure 3.2 spanning over a significantly longer period of time. For the Process_44 data set, periods when the process is not running can be identified by a significant change in values and distribution, which is also noticeable in the figure 3.2 plots. For the subsequent data sets, Process_45-Process_50, the distribution of values is very similar to those presented in figure 3.3.

3.3 Preprocessing

In the context of analysing Bioco's industrial process data for anomaly detection, it is essential to recognise that treating this data as regular time series may not yield meaningful insight. Traditional methods for time series decomposition, such as Seasonal Decomposition of Time series (STL) are designed to capture seasonality, trends and residuals in the data. These components are largely irrelevant in a continuous industrial process, and applying STL or similar methods to such data in not the most appropriate approach for understanding the underlying behaviour. Thus, these techniques are not used in the thesis.

3.3.1 Frequency of data collection

As previously shown in table 3.1, the sampling frequency varies in the Process_2021 data set. Although the approach to this time series data differs from the traditional methods of handling a time series, a consistent frequency in the data is important for preserving the temporal dependencies and pattern in the data. Discrepancies in the sampling frequency are thus investigated using Python:

Algorithm 1 Pseudocode for finding frequency changes

Require: *df* - DataFrame
Ensure: Prints indices where frequency changes are detected
Calculate time differences between consecutive rows in $df\_find\_frequency$
Store time differences in a new column $time\_diff$
Convert time differences to seconds
Store the converted time differences in a new column $time\_diff\_seconds$
Calculate differences between consecutive time differences
Store the calculated differences in a new column $time\_diff\_delta$
Find indices where the absolute value of $time\_diff\_delta$ is greater than or equal to 1
Store these indices in a variable $change\_indices$
if length of $change\_indices > 0$ **then**
  print "Frequency changes in the dataset occur at the following indices:"
  $previous\_index \leftarrow None$
  for each $index$ in $change\_indices$ **do**
    if $previous\_index$ is None OR the location of the $previous\_index$ is not equal to the location of the $index - 1$ **then**
      $before\_index \leftarrow$ location of the $index - 1$
      $after\_index \leftarrow$ location of the $index + 1$
      if $before\_index$ is within the range of indices AND $after\_index$ is within the range of indices **then**
        print $index$, $index$ before it, and $index$ after it
      end if
    end if
    $previous\_index \leftarrow index$
  end for
else
  print "The frequency doesn't change throughout the dataset."
end if

The function investigated both frequency changes and time anomalies, which are instances where there is a sudden jump in the data sampling. For data sets other than Process_2021, the frequencies do not change, and are consistent throughout. For the Process_2021 data, the frequency changes at 2020-12-25 10:13:39, where the sampling frequency goes from once a minute, to six times a minute. This is however, true only for a part of the sensors, and other sensor data are still collected at 60 second intervals. Furthermore, the data set contains a variety of time anomalies, but the majority of these are outside the scope of relevance for anomaly detection, as they either occur when the industrial process is not running, or during intervals which are not later used for anomaly detection. The frequency change, however, introduces a variety of missing data, which is addressed in the next section.

3.3.2 Missing values

The presence of missing values is investigated using the open-source package MISSINGNO. The nullity matrix displays data completion patterns in the data:

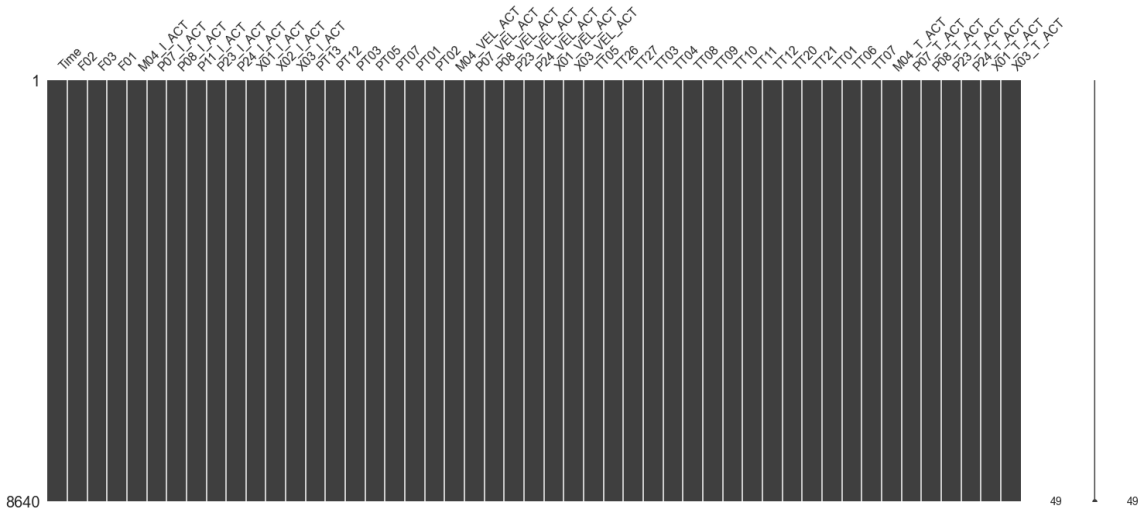


Figure 3.4: Missing value matrix for the Process_44 data set. The columns represent the features in the data, and the black areas showcase the completion of the data in the corresponding column. A full black column means no missing data

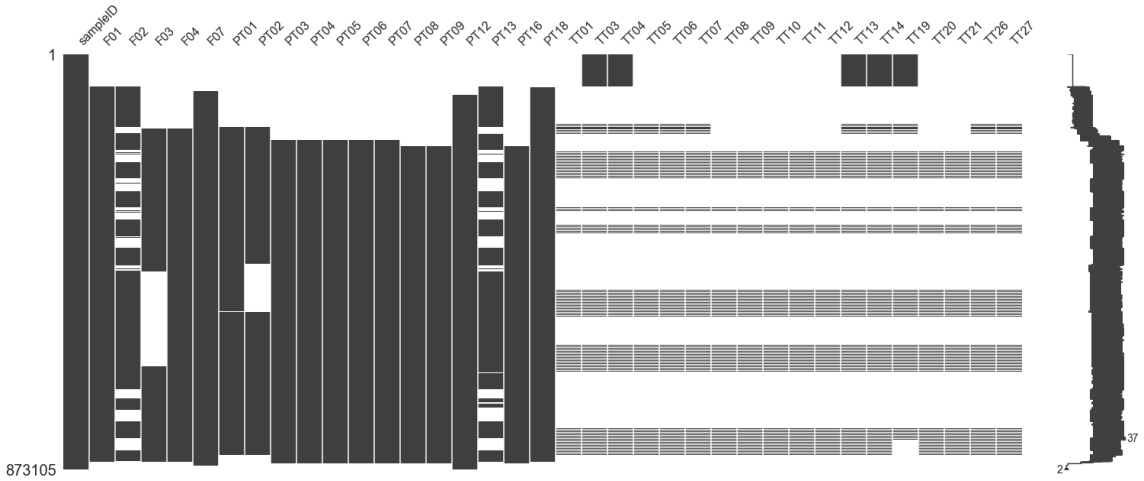


Figure 3.5: Missing value matrix for the Process_2021 data set. The columns represent the features in the data, and the black areas showcase the completion of the data in the corresponding column. A full black column means no missing values.

The columns represent the features in the data, and the black areas showcase the completion of the data in the corresponding column. For data sets other than Process_2021, there are no missing values. Process_2021 contains a vast variety of missing data, some of which are introduced during the frequency shift. A majority of the instances where there is a long sequence of missing data, such as in the column PT02 in figure 3.5, are within a range where the process is not running, or during a period irrelevant for the analysis. However, missing values in columns such as F02 at PT13 fall within the range of some of the clogging events defined in table 3.2.

These cannot be imputed in an effective way and the columns are discarded if selected within the data window chosen during further analysis.

As discussed previously, a frequency change occurs in the collection of the data. However, data in features such as TT03 and TT04 in figure 3.5 do not undergo this change, and are still collected once a minute. This is evident in the stripe pattern observed in these columns. For periods where this stripe pattern is present within a chosen data window, the values are linearly interpolated and brought to the same frequency as the other features.

3.3.3 Sensor location identification

Approximate locations of part of the sensors in the process were provided by Bioco. Approximate locations of flow rate sensors F01, F02, F03, temperature sensors TT06, TT07, TT08, TT09, TT12, TT12 and pressure sensors PT02, PT03, PT05 and PT07 are known. The exact locations of these and the rest of the sensors were further investigated by computing correlation coefficients in the data sets, with a varying degree of success:

Algorithm 2 Pseudocode for calculating lag and correlation between sensors

Require: *signal*1, *signal*2, *max_lag*
Ensure: Returns the best lag and maximum correlation between the two signals
function CALC_LAG(*signal*1, *signal*2, *max_lag*)
Initialize an empty list *corr_values*
for *lag* in range $-max\_lag$ to *max_lag* **do**
  Calculate the correlation between *signal*1 and the *lag*-shifted *signal*2
  Append the calculated correlation to *corr_values*
end for
Find the maximum correlation in *corr_values* and store it in *max_corr*
Find the index of the maximum correlation and store it in *best_lag*
Subtract *max_lag* from *best_lag*
return *best_lag*, *max_corr*
end function
Initialize an empty dictionary *lags*
for each i in the range from 1 to the number of columns in $data\_44\_find\_lag$ **do**
  for each j in the range from 0 to $i - 1$ **do**
    $signal1 \leftarrow data\_44\_find\_lag$ at column i
    $signal2 \leftarrow data\_44\_find\_lag$ at column j
    (*lag*, *corr*) ← CALC_LAG(*signal*1, *signal*2, 55)
    if $corr > 0$ **then**
      Add the result to *lags* with the key being the combination of column names
    end if
  end for
end for
Print the significant correlations
for each *pair*, *values* in *lags* **do**
  print "The lag between" *pair* "is" *values*[0] "minutes with a correlation of" *values*[1]
end for

The *max_lag* is subsequently adjusted depending on the data set used, representing the maximal lag value in data instances. For data sets other than Process_2021, this value is set to 55, corresponding to 55 minutes into the process. For the Process_2021 dataset, this value is set to 330, corresponding to the same amount of time adjusted for the changed frequency. The use

of the method on this data set is limited to a selected window in the data when the industrial process is running.

For a selection of sensors, the method produces desirable results, indicating toward values that are in line with the approximate sensor locations provided by Bioco. For sensors corresponding to velocities, torques and intensities, the results are unsatisfactory and random, thought to be caused by the noise and the nature of the data, and the correlations cannot be properly examined. The use of the obtained values is closely related to and further discussed in the following subsection.

3.3.4 Lagging

To incorporate the lag information obtained from the preceding subsection, correlated sensor signals can be adjusted according to their estimated lag. This process is done by shifting the time series of one sensor relative to the other by the computed lag value. Aligning the time series in this manner achieves an accurate representation of the relationship between the sensors and the raw material passing through the process.

There are a multitude of ways of implementing the lag process, depending on the sensor chosen as the reference point. For the purpose of this thesis, the temperature sensor $TT12$ at the end of the process was chosen as the reference point, resulting in sensor data located prior to deactivation 3.1 being shifted forward by a defined value, corresponding to the location of sensor in the process.

Algorithm 3 Pseudocode for lagging sensor data

Require: *df* - Original DataFrame
Require: *locations* - Dictionary containing sensor locations
Ensure: Produces a DataFrame with lagged sensor data
1: Initialize an empty DataFrame *lagged_df*
2: **function** SHIFT_DATA(*sensor*, *lag*)
3: Shift the *sensor* data by *lag* in the *df* DataFrame
4: **return** shifted sensor data
5: **for** each *sensor*, *lag* in *locations* **do**
6: $shifted\_sensor\_data \leftarrow$ SHIFT_DATA(*sensor*, *lag*)
7: Concatenate $shifted\_sensor\_data$ to *lagged_df*
8: **end for**

The sensor data is shifted according to the specified lag values in the *locations* dictionary. Twelve sensors, approximate locations of which are known are lagged accordingly to be in aligned with temperature sensor $TT12$ for the relevant data sets. These are subsequently used for anomaly detection, together with the unlagged data for a coherent representation of the underlying process. Throughout the rest of the thesis, the term "lagged" and "shifted" are used interchangeably to represent the same process.

3.4 Summary

The data exploration process for the Bioco data set has been discussed and the data structure presented. The necessary preprocessing steps have been outlined and data has been prepared for anomaly detection. Functions for identifying frequency anomalies, correlations between sensors and subsequent lagging have been defined. These processes set the foundation for further analysis and modeling in the subsequent chapters.

Chapter 4

Method

The purpose of this chapter is to provide an overview of the methods used for anomaly detection in the industrial time series data. The variations in data preprocessing steps depending on the model used are shown. The local outlier factor and DBSCAN algorithms are then presented and optimised, as they are the primary methods used. Criteria for evaluation of the models are defined to ensure that the results are credible. The results of application of these methods are presented in the following chapter.

4.1 Data selection

Refer to tables 3.1 and 3.2 for this section. Clog 5 is attributed to manual change in process parameters. It is caused by human error and not as a result of the industrial process. Thus, it is unsuitable for anomaly detection and excluded from the analysis to avoid biased results.

Prior to clog 3, process parameters were changed, and the clogging event is thought to be caused by this change. This occurs 40 minutes prior to the clogging. Despite this, decision is made to include this clogging in the analysis, as the industrial process continues regularly until the blockage. The impact of this is noted later in the thesis, as results are presented.

The clogging events 1,2,3,4 in the Process_2021 data are all relevant for anomaly detection. Varying windows of data surrounding these are selected for when the process is running. Features in data windows that include missing values discussed in the previous chapter 3.3.2 that cannot be imputed are discarded. From the 2022 data, only the Process_50 data set, containing clogging event 6, is used in anomaly detection. The remaining data sets serve as references for evaluation and generalizability testing of tuned algorithms.

Additional data is selected for analysis to use in parallel with other data windows, defined as "secondary data". As secondary data, the selected data windows are reduced to only contain a subset of features. These include only the sensors for which approximate locations in the process were provided by Bioco. The locations were subsequently confirmed using the correlation function defined in the previous chapter 2. As a result, the dimensions of the data are reduced to *window length* x 14 features, including the time column. Unimputable features in relevant data are similarly discarded.

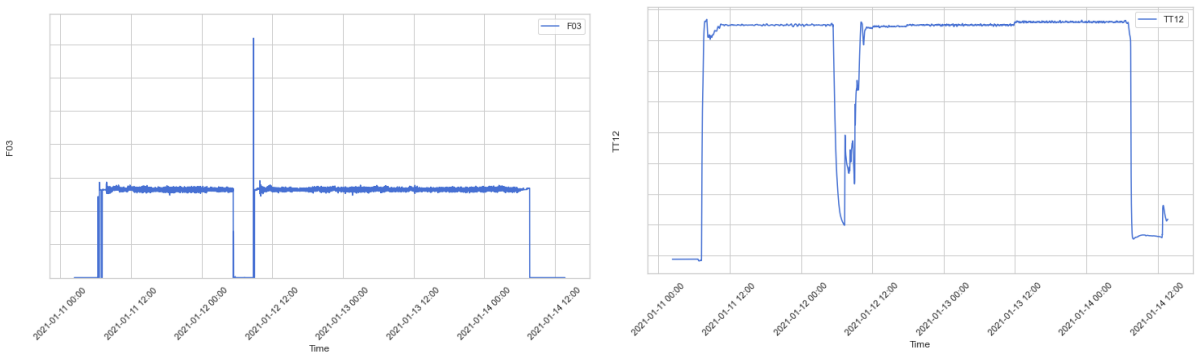


Figure 4.1: Flow rate sensor F03 and temperature sensor TT12 in a selected data window

The plots depict flow rate sensor F03 and temperature sensor TT12 data in the selected data window for analysis of clog 1. Clog 1 is the sudden drop in pressure and temperature in the figures. The rest of the features show a similar pattern. This is true for the rest of the clogging events. Starts and ends of the process are included for model evaluation, as these would naturally be seen as anomalies by algorithms used.

4.2 Selective denoising

A variation of denoising is done on a per-sensor basis after selection of data windows. Features of the data containing sensor errors and unnatural value deviations not related to the regular process are selected. Denoising is performed by defining noise intervals and calculating rolling means and standard deviations for a set amount of data points before and after an interval. The gap is subsequently filled with normally distributed values, aiming to preserve the similarity of structure to surrounding data. Following figures illustrate an example of applying this to flow rate sensor data in a selected data window.

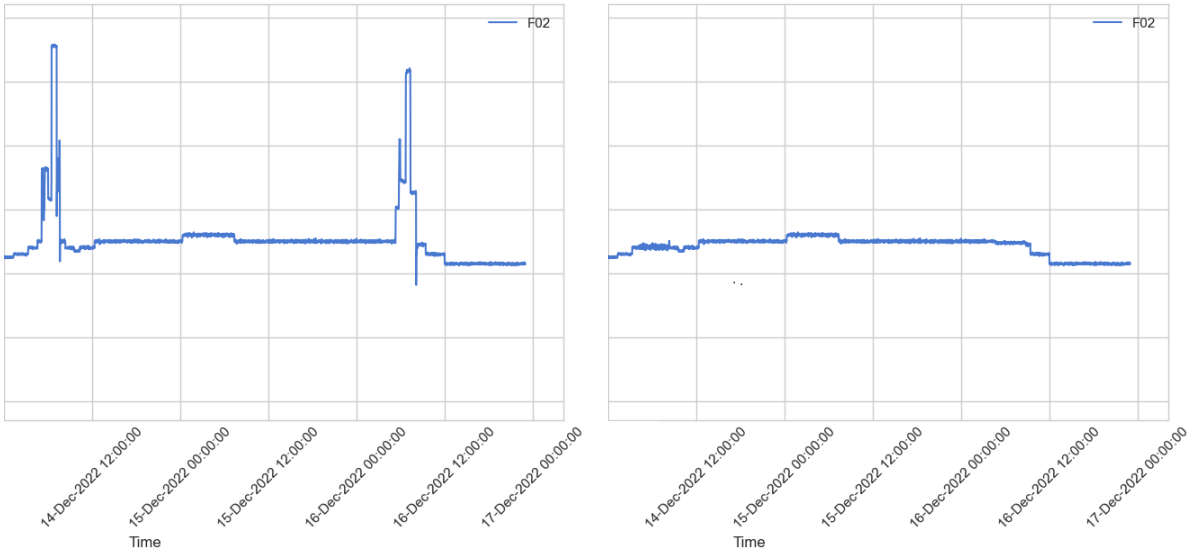


Figure 4.2: Flow rate sensor F02 before and after denoising in selected intervals

4.3 Dimensionality reduction for feature importance

Techniques for dimensionality reduction can be used for analysis of data structure. In this case, principal component analysis (PCA) is applied to access feature importance and to identify potential redundant features, before the application of anomaly detection. The original feature space is transformed into a lower-dimensional space and principal components are created, capturing the majority of the variance in the data set. The first principal component account for the largest variance in the data, followed by the second, and so on.
The PCA transformation can be summarised as:

$$Y = XW$$

Where X is the original data, W the eigenvectors and Y is the transformed data. Coefficients that define the linear combination of the original features in the data are called loadings. These are obtained from the PCA as:

$$Loadings = Eigenvectors \times \sqrt{Eigenvalues}$$

The eigenvectors W and eigenvalues λ are derived from the covariance matrix of the original data, either $XX^\top$ or $X^\top X$. The magnitude of these loadings then indicates how much each feature contributes to the principal component. The sum of the absolute loadings for each feature, across all principal components, gives feature importance:

$$Feature\ Importance_i = \sum_{j=1}^{n} |Loadings_{ij}|$$

Where i is the index of the feature, and n is the number of principal components.

Feature importance is calculated for the relevant data in the following manner:

Algorithm 4 Pseudocode for PCA Loadings for Feature Importance

1: Scale the data set
2: Fit PCA to the scaled data set, 95% explained variance
3: Obtain the PCA loadings
4: Calculate the absolute loadings
5: Compute the sum of the absolute loadings for each feature across all principal components
6: Display the importance of each feature

A threshold of 95% explained variance is used. This approach gives insight into the relative importance of features in the data. Redundant features are removed to make the input data more informative and less noisy for anomaly detection algorithms. As PCA is a linear technique, there are limitation to how effective this method is for feature importance analysis. Feature importance is further discussed in later parts of the thesis.

4.4 Local outlier factor algorithm

As local outlier factor is a distance-based algorithm, it is sensitive to scale of the features. Features in the selected windows of data are standardised by removing the mean and scaling to unit variance.

Initially, a baseline LOF model is fit to the data. Predicted anomaly points are attached to timestamps, inspected and visually analysed. Hyperparameter tuning is done using a custom

grid search process and a custom wrapper class for the model, as tuners such as *GridSearchCV* normally require labels.

Algorithm 5 Pseudocode for hyperparameter tuning of LOF algorithm with custom wrapper, scorer, and pipeline for time series data

```
Input: Preprocessed DataFrame
Output: Best model best_model
Output: Anomaly timestamps
Define custom scorer function lof_score(estimator, X)
Create custom wrapper class LOFWrapper with methods: __init__, fit, predict
Define parameter grid param_grid for LOFWrapper
Create pipeline with StandardScaler and LOFWrapper
Initialize best_score to ∞ and best_params to None
Create a parameter grid iterator param_grid_iterator from param_grid
for each parameter combination params in param_grid_iterator do
    Set pipeline parameters using params
    Initialize scores list
    for each train_index and test_index in TimeSeriesSplit(X) do
        Split X into X_train and X_test using train_index and test_index
        Fit pipeline on X_train
        Calculate score using lof_score function on pipeline model and X_test
        Append score to scores list
    end for
    Calculate mean_score as the mean of scores list
    if mean_score < best_score then
        Update best_score to mean_score and best_params to params
    end if
end for
Print best_params
Create best_model using best_params
Fit best_model on X
Predict anomalies using best_model on X
Get anomaly timestamps from X.index where predicted labels are -1
```

A custom wrapper *LOFWrapper* is created and tailored to work with unsupervised learning methods and unlabeled data. Allows to set the y parameter to None to work with LOF in anomaly detection tasks. Inherits from base class estimator BaseEstimator and ClassifierMixin which implements the scoring method, allows it to be used in a scikit-learn pipeline. The custom scorer *lof_score* is made so custom scoring can be defined. This takes an estimator and a data set as inputs, returning the inverse of the mean of *negative_outlier_factor_*, which is the negative of the standard local outlier factor. This type of scoring rewards models with low scores that try to optimally capture the structure of the data. Thereafter, a parameter grid can be used in the same manner a grid search is done for supervised methods.

Cross-validation is done using TimeSeriesSplit that splits the data into samples that preserve the temporal dependency between observations. Regular K-fold cross-validation is avoided as it does not preserve the relation, splitting the data randomly. Parameters are updated in a loop, and those with lower validation scores update the best parameters.

Optimal model is achieved by scaling and adjusting the best parameters found during the tuning. A middle ground is found between an out-of-the box and the tuned model. The performance is

evaluated by plotting the data and examining how early anomalous behaviour was seen before a clogging. This is done iteratively while model parameters are changed. Precision is analysed by the number of non-anomalous instances flagged as anomalous. Ideal model performance is seen as achieving a state where only the pre-clogging anomalous behaviour and starts and ends of the process are flagged as anomalies.

Feature importance is investigated after an optimal performing model is achieved. Two methods are used, with one utilising cluster labels and a supervised classifier model, and a manual method where features are iteratively removed and LOF scores are calculated. The method utilising supervised learning is divided into multiple variations.

Algorithm 6 Pseudocode for feature importance calculation for LOF algorithm using feature removal

```
Define LOFModel with tuned hyperparameters
Standardize data set using StandardScaler
Calculate LOF scores for the full dataset
Initialize FeatureImportance dictionary
for each feature in the dataset do
    Remove the current feature from the standardised dataset
    Calculate LOF scores without the current feature
    Calculate mean absolute difference between original and reduced LOF scores
    Store the score difference in FeatureImportance dictionary
end for
Convert FeatureImportance dictionary to a DataFrame
Sort the feature importances
Print the sorted DataFrame
```

Most optimal model is used here. The method essentially performs a feature removal analysis. Mean absolute differences in LOF scores are calculated with and without each feature, and the features are ranked by their importance in the process.

Additionally, a secondary analysis is done using a RandomForestClassifier and data labeled by the LOF algorithm.

Algorithm 7 Pseudocode for feature importance calculation for LOF algorithm using a decision tree classifier

```
Scale data using StandardScaler
Fit the LOF model to the scaled data and obtain LOF labels
Create a new DataFrame with LOF labels as the target variable
Split the labeled data into features (X) and target (y)
Train a RandomForestClassifier on the labeled data (X, y)
Calculate permutation importance using the trained RandomForestClassifier and permutation_importance
Display feature importances
```

The most optimal LOF algorithm is first applied to generate labels for the data. A RandomForestClassifier is trained on the data and feature importance is calculated using permutation importance. This method uses permutation_importance which permutes the features one at a time. The change in the model's accuracy score is used to estimate the importance of that feature.

There are considerations to keep in mind using this method. As a clogging occurs, the industrial process comes to a halt. The regular process is disturbed and sensor values greatly differ before the pipes are flushed as seen in figure 4.1. As PERMUTATION_IMPORTANCE randomly shuffles values of a single feature, this can lead to misleading estimates. A workaround is to modify the method to use windowed permutations, where data are shuffled within windows instead of permuting the whole feature.

Algorithm 8 Pseudocode for windowed permutation importance for LOF Algorithm

```
Scale data using StandardScaler
Fit the LOF model to the scaled data and obtain LOF labels
Create a new DataFrame with LOF labels as the target variable
Split the labeled data into features (X) and target (y)
Train a RandomForestClassifier on the labeled data (X, y)
Define windowed_permutation_importance function with inputs: model, X, y, window_size, n_repeats, and random_state
Initialize scores_decreases as an empty list
for each feature in X do
    Initialize decrease as an empty list
    for i in range 0 to n_repeats do
        Create a copy of X, called X_permuted
        for j in range 0 to number of samples in X, with step window_size do
            Permute the values of the feature within the window in X_permuted
        end for
        Calculate the accuracy score using the RandomForest model on X_permuted and y
        Append the drop in accuracy to decrease
    end for
    Append decrease to scores_decreases
end for
Return scores_decreases as an array
Calculate windowed permutation importance using the trained RandomForestClassifier and windowed_permutation_importance function
Display feature importances
```

The new method acts as an extension of standard permutation importance. It works similarly, where the difference between the original model's accuracy and the accuracy after permuting a feature is used to estimated feature importance. However, the feature values are permuted within windows of fixed size. This change aims to avoid misleading results and give a more accurate estimate of important features.

An optimal window size needs to be determined for optimal performance of this method as the structure of the data needs to be preserved. This is done by iterating through different window sizes and inspecting the results. The procedure is as follows:

Algorithm 9 Pseudocode for testing different window sizes for windowed permutation importance

```
Define a list of window sizes to test
Define try_window_size function with input: window_size
Calculate windowed permutation importance using window_size
Compute mean importances and standard deviations of importances
Return mean importances and standard deviations
for each window_size in the list do
    Call try_window_size function with the current window_size
    Print window_size and the corresponding feature importances
    Calculate and print the sum of feature importances
end for
```

As Bioco's industrial process lasts around 60 minutes, this is used as an anchor point for finding the optimal window size, scaling the values in the list of window sizes to test with how many process cycles are passed. The results for each window size are inspected. The criterion is to find a window size that has stable importance scores, where scores do not significantly change between several gaps. Goal is to minimise the trade-off between misleading results and preserving data structure.

In summary, manual feature removal in algorithm 6 assumes that the mean absolute difference in LOF scores is an adequate metric of importance. This method does not capture complex relationships between features and is used as a baseline algorithm for feature scoring. Methods for feature importance calculation showcased in algorithms 7 and 8 depend on the quality of the LOF model. As labeled data are unavailable and the model is tuned by visual reference, these methods can underestimate and overestimate feature importances.

4.5 DBSCAN algorithm

Similarly to local outlier factor, DBSCAN is sensitive to the scale of the features. Features are appropriately scaled in the selected windows of data to mitigate the influence of varying magnitudes.

Starting procedure is identical to LOF, a baseline DBSCAN model is fit to the data, predicted anomaly points are attached to timestamps, inspected and visually analysed. In contrast to LOF, which evaluates the degree of local densities, DBSCAN can identify clusters of arbitrary, varied shapes. This difference should allow DBSCAN to better adapt to industrial sensor data. However, as mentioned in section 2.5.2.2, proper selection of hyperparameters $MinTps$ and ϵ is crucial for optimal results.

As $MinPts$ represents the minimum number of points required in a neighbourhood for it to be considered a cluster, a nearest neighbours approach can be used. K-nearest neighbours' distances for each point in the data are calculated. Density of data points in the feature of data points is then analysed. A sharp change in density is an indication of an appropriate $MinPts$ value.

Algorithm 10 Pseudocode for finding optimal MinPts value and distances

Require: data, min_k, max_k
Ensure: optimal_k, k_values, avg_distances_all
Initialize optimal_k, max_curvature, k_values, and avg_distances_all
for k in range(min_k, max_k + 1) **do**
    Instantiate NearestNeighbors with n_neighbors=k and fit to data
    Calculate distances to k-nearest neighbors
    Compute average distances
    Sort average distances
    Compute second derivative (curvature) of sorted average distances
    Find the index of the maximum curvature
    if maximum curvature > max_curvature **then**
        Update max_curvature and optimal_k
    end if
    Append k to k_values and the mean of average distances to avg_distances_all
end for
return optimal_k, k_values, avg_distances_all

NearestNeighbors is used to calculate distances. Second derivative, or curvature, of sorted average distances is used to measure how sharply average distances change to identify optimal k. This method helps to automate the process by iterating through a set of given k-values efficiently and find optimal $MinPts$ parameter. Domain knowledge is often used in determining the optimal value for this parameter and a rule of thumb is to use a value of at least $Minpts > Features$. As such, this method is seen as appropriate as this condition holds true for optimal k-values determined.

An example of results derived from this implementation is illustrated in the following figure, illustrating relationship between k-values and average distance to the k-th nearest neighbour. In this iteration, the optimal $MinPts$ value was determined to be 43.

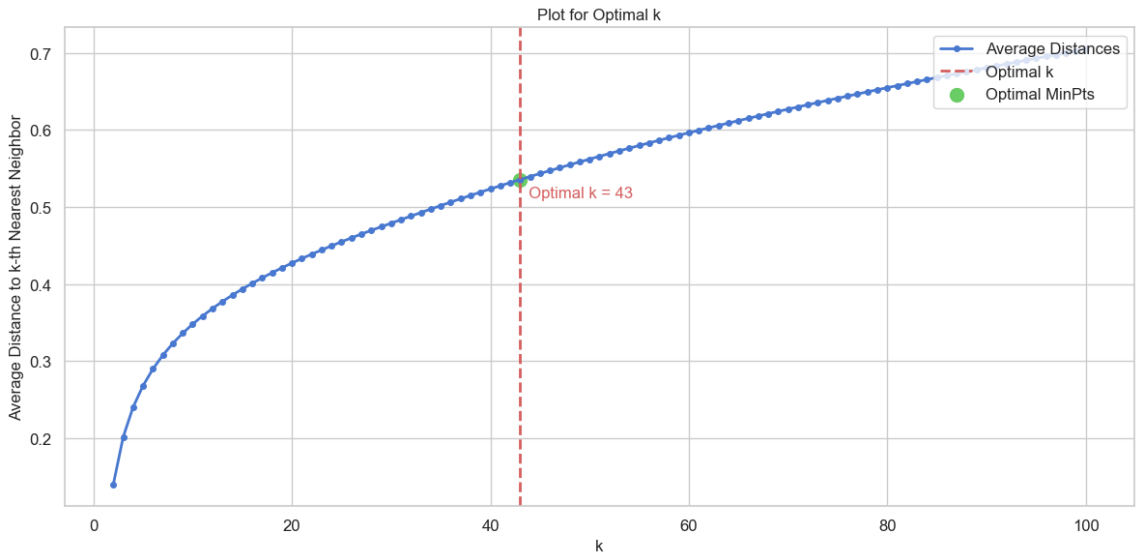


Figure 4.3: Relationship between k-values and average distances to k-th nearest neighbour, with optimal k-value derived illustrated

The method essentially derives from analysing elbow/knee plots to determine the optimal k-

value, but these sharp curvature changes are not visually observed in such plots in data used. Thus, values obtained from algorithm 10 are used.

Acquired $MinPts$ parameter can then be used to obtain optimal value of ϵ. NearestNeighbors can be reused, similarly calculating k-nearest neighbour distances. The distances to the k-th nearest neighbour for each data point are utilised instead of average distances, as epsilon represents the radius of neighbourhood around each point.

Algorithm 11 Pseudocode for finding optimal epsilon value

Require: data, k
Ensure: optimal_epsilon
function find_optimal_epsilon(df, k)
Instantiate NearestNeighbors with n_neighbors=k and fit to data
Calculate distances to k-nearest neighbors
Extract the distance to the k-th nearest neighbor for each point
Sort the k-nearest neighbor distances
Find the elbow point using KneeLocator with curve='convex' and direction='increasing'
Plot the KNN distance plot and mark the elbow point
return The k-th nearest neighbor distance at the elbow point
end function
Set the value of k
Call find_optimal_epsilon with scaled data and k
Print the optimal epsilon value

An example of results derived from this implementation is illustrated in the following figure.

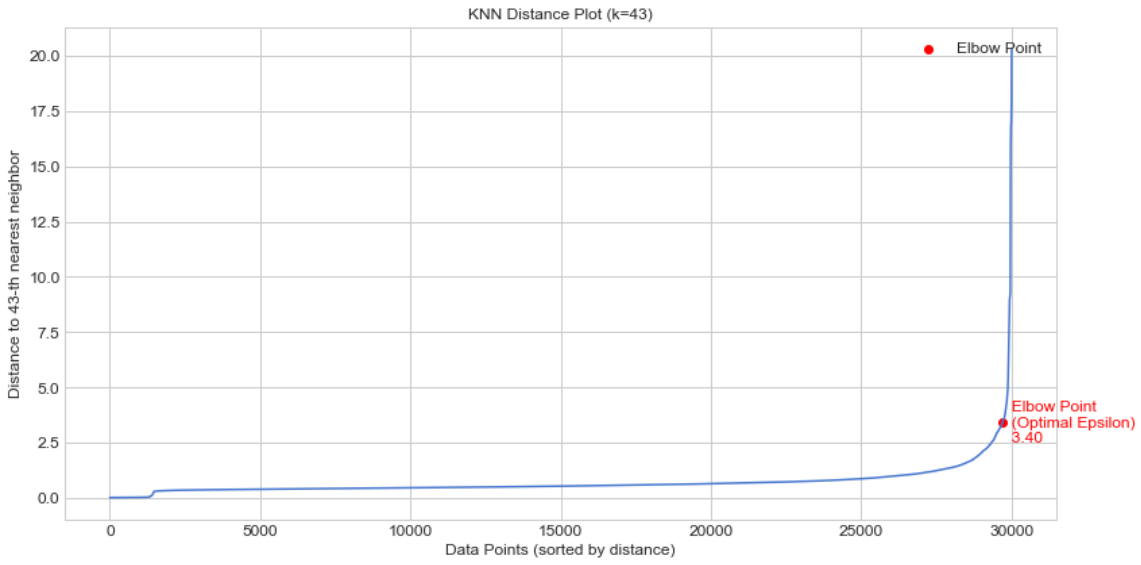


Figure 4.4: KNN distance plot with optimal value of ϵ marked in red

The plot showcases the optimal ϵ value found for an iteration of parameter tuning for a data window. KneeLocator is used to automatically find the point for maximum curvature in the plot, which is marked with the red circle.

An optimal model is achieved utilising the same process used for local outlier factor algorithm in

4.4. A state where anomalous behaviour is detected as early as possible, while also minimising the amount of false positives is seen as ideal model performance. Same logic is used for starts and ends of the process, and rates of false positives for these are not relevant.

Feature importance is similarly investigated utilising the same methods used for LOF, applying procedures described in algorithms 6, 7 and 8. The LOF model is exchanged for the best performing DBSCAN model. Adjustments are made to algorithm 6 to use silhouette scores as the evaluation metric. Differences between silhouette scores, measuring how well points are assigned to clusters, are used as the importance value for each feature. Similarly to LOF, DBSCAN cluster labels are used for a supervised learning approach. As the labels only vaguely represent the ground truth, feature importances can be over- and underestimated.

4.6 Implementation flow

The entire workflow of the anomaly detection process can be summarised in the following workflow.

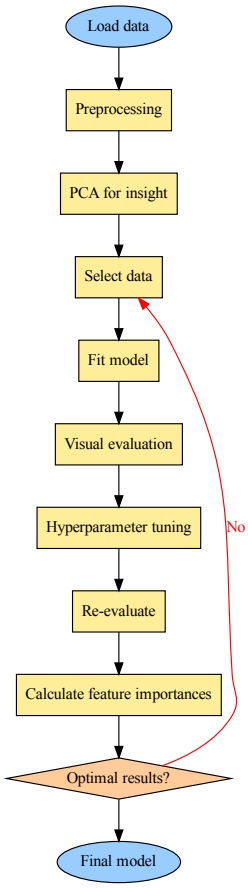


Figure 4.5: Anomaly detection workflow for LOF and DBSCAN algorithms

4.7 Summary

The chapter presents two unsupervised learning algorithms for anomaly detection, local outlier factor and DBSCAN, and the approach of tailoring these for industrial time series data.

The next chapter presents the results of application of these. The anomalous points identified by LOF and DBSCAN are attached to the original data with minimal preprocessing for an optimal representation of the industrial process. Identified anomalies are presented using figures captured from interactive Plotly plots, that allow for a better visualisation of data surrounding these.

Chapter 5

Results

This chapter showcases the identified anomalous points prior to clogging events in relevant data windows. Primary focus is to showcase the performance of two unsupervised learning methods, local outlier factor and DBSCAN, on the relevant data windows containing clogging events. For locations of the clogging events in the data, refer to table 3.2. As previously mentioned, clog 5 is not included, as it is attributed to human error.

The chapter only showcases the results for clog 1. For readability reasons, the rest of the results are moved to the Appendix. These results are presented in a similar fashion and can be found in Appendix B.

The anomaly timestamp tables presented showcase the earliest anomalous behaviour detected by an algorithm prior to a clogging and relevant in the context. The index column represents the position of these values in a list of points flagged as anomalies by the algorithms, used as a reference point for evaluation.

Timestamps of identified anomalous points are attached to the original data and plotted together with selected sensors. To get an accurate representation of the process, anomalies are shown together with sensor data of flow rate sensor F01, located early in the process, in the hopper, and temperature sensor TT12, located late in the process, in deactivation 3.1.

For confidentiality reasons, data values in the plots are removed or hidden. As a result, the figures can be challenging to interpret. However, all clogging events in the figures can be identified by a sudden decrease of values, followed by an irregular distribution in relation to the rest of the figure.

Additionally, results of using lagged variables are presented in a similar format.

5.1 Local outlier factor results

Following tables and figures present relevant timestamps flagged as anomalies by the most optimal LOF model. As previously mentioned, the index columns show the position of these points in a list of data flagged as anomalous. Results for clog 2,3,4 and 6 can be found in Appendix B.1

5.1.1 Results without lagging of variables

5.1.1.1 Clog 1

For the first clogging, earliest anomalies are identified right as the event happens. No anomalous behaviour is detected a notable amount prior to the clogging.

Table 5.1: LOF anomaly timestamps for clog 1, occurring at 05:45

Anomaly index	Timestamp
4	2021-01-12 05:22:34
5	2021-01-12 05:22:44
6	2021-01-12 05:22:54
7	2021-01-12 05:23:04
8	2021-01-12 05:23:14
9	2021-01-12 05:23:44
10	2021-01-12 05:23:54

The following plots showcase the points presented in the table, as well as additional false positives, anomaly indexes 1-3, prior to the clogging event. Further anomalous points displayed are not relevant, as they are marked during the clogging.

Figure 5.1: Clog 1, anomalies detected using LOF, showcased on F01 sensor data

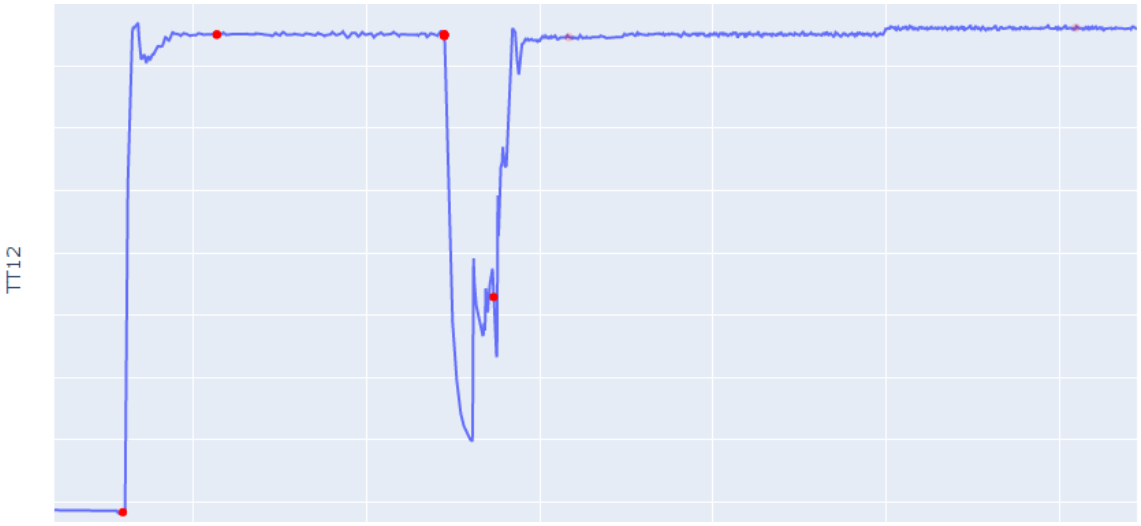


Figure 5.2: Clog 1, anomalies detected using LOF, showcased on TT12 sensor data

The following figure presents the earliest relevant timestamp prior to the clogging, zoomed in for visibility.

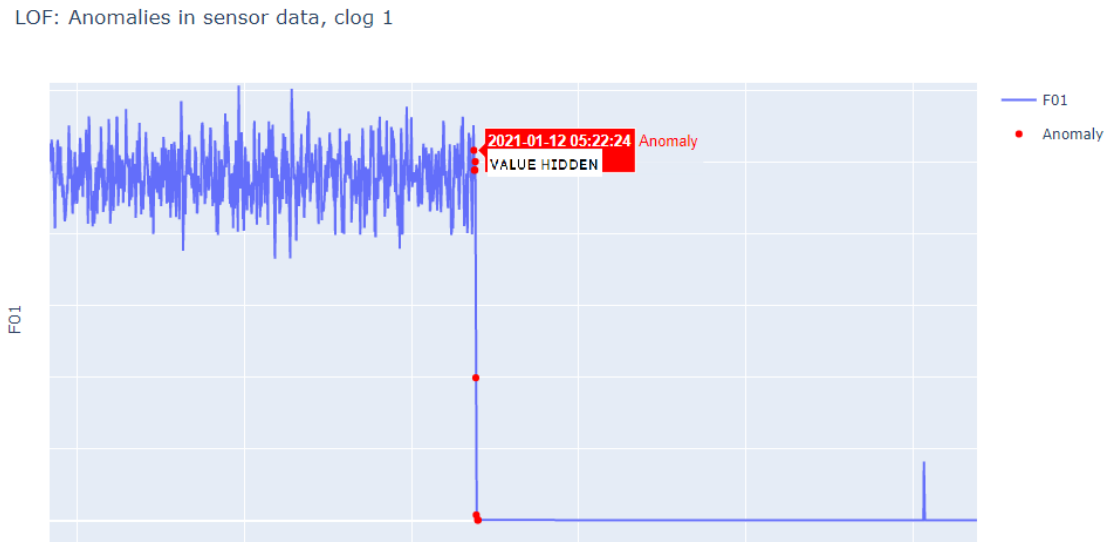


Figure 5.3: Clog 1, earliest detected relevant anomaly timestamp, showcased on F01 sensor data

5.1.2 LOF results from using lagged variables

This part presents results from utilising data shifted by the method showcased in section 3.3.4.

5.1.2.1 Lagged variables and clog 1

Using lagged variables for clog 1 does not provide optimal results. No anomalous behaviour is detected prior to the clogging. The earliest detected timestamps during the clogging are showcased in the following table.

Table 5.2: LOF anomaly timestamps for clog 1 using lagged variables, occurring at 05:45

Anomaly index	Timestamp
20	2021-01-12 05:46:34
21	2021-01-12 05:46:44
22	2021-01-12 05:46:54
23	2021-01-12 05:47:04

The following plots showcase the points presented in the table. Additional anomalies displayed are marked during the clogging and startup of the process.

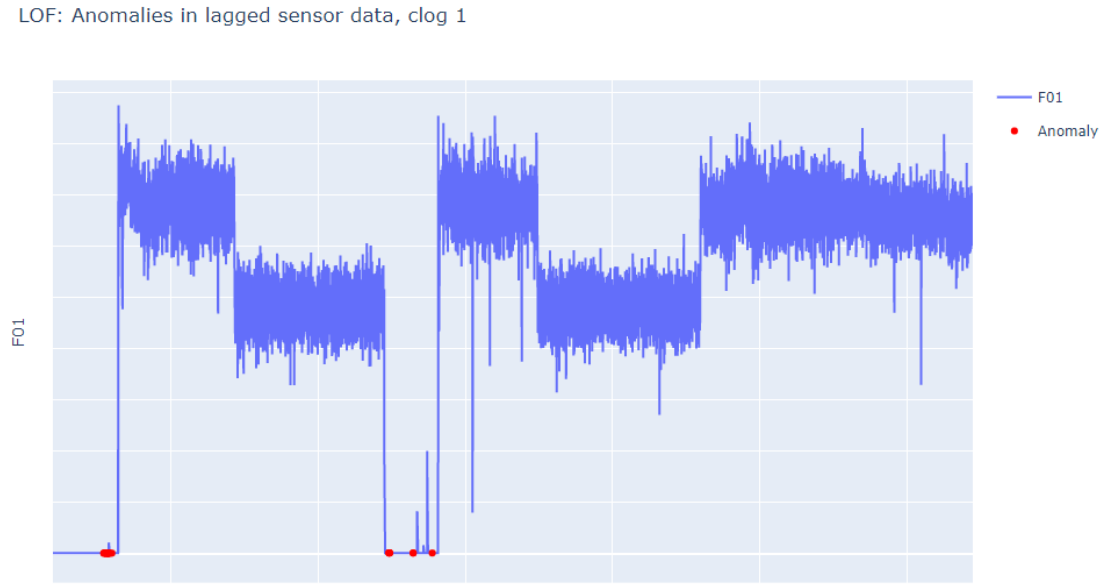


Figure 5.4: Clog 1, anomalies detected using LOF and lagged variables, showcased on F01 sensor data

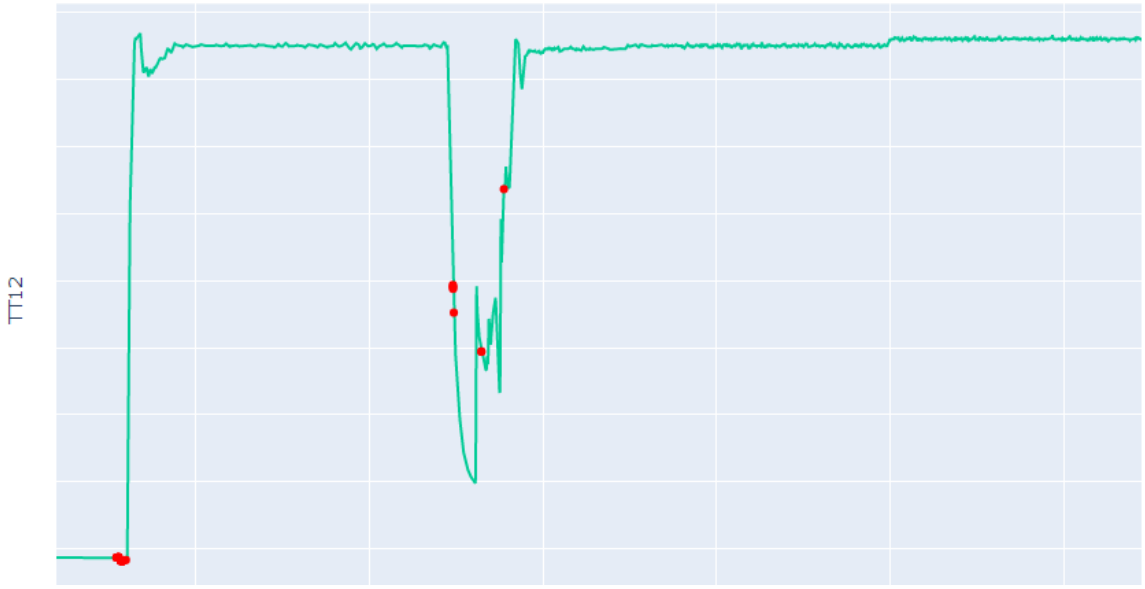


Figure 5.5: Clog 1, anomalies detected using LOF and lagged variables, showcased on TT12 sensor data

The following figure presents anomaly index 20 in table 5.2, zoomed in for visibility.

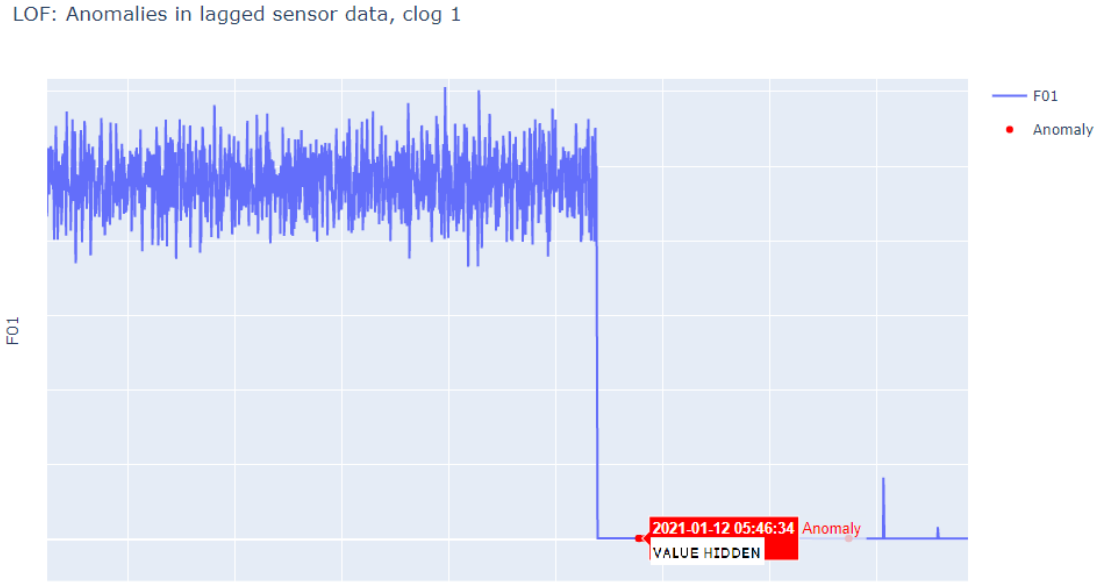


Figure 5.6: Clog 1, earliest detected relevant anomaly timestamp in lagged data, showcased on F01 sensor data

5.2 DBSCAN results

Following tables and figures present relevant timestamps flagged as anomalies by the most optimal DBSCAN model. As previously mentioned, the index columns show the position of these points in a list of data flagged as anomalous. Results for clog 2,3,4 and 6 can be found in Appendix B.2

5.2.1 Results without lagging of variables

5.2.1.1 Clog 1

For the first clogging using DBSCAN, earliest anomalies are identified right as the event happens. However, the anomalous behaviour is detected a minute earlier compared to utilising LOF 5.1. No anomalous behaviour is detected a notable amount prior to the clogging.

Table 5.3: DBSCAN anomaly timestamps for clog 1, occurring at 05:45

Anomaly index	Timestamp
2	2021-01-12 05:21:34
3	2021-01-12 05:21:44
4	2021-01-12 05:21:54
5	2021-01-12 05:22:04
6	2021-01-12 05:22:14
7	2021-01-12 05:22:24
8	2021-01-12 05:22:34

The following plots showcase the points presented in the table, as well as a false positive, anomaly index 1, prior to the clogging event. Additional anomalous points displayed are not relevant, as they are marked during the clogging.

Figure 5.7: Clog 1, anomalies detected using DBSCAN, showcased on F01 sensor data

Figure 5.8: Clog 1, anomalies detected using DBSCAN, showcased on TT12 sensor data

The following figure presents the earliest relevant timestamp prior to the clogging, anomaly index 2 in table 5.3, zoomed in for visibility.

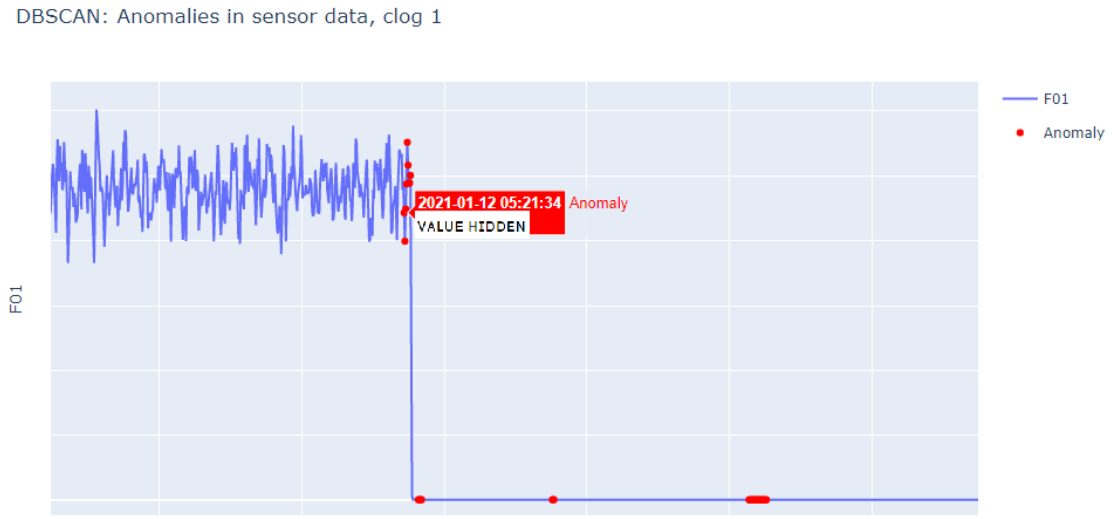


Figure 5.9: Clog 1, earliest detected relevant anomaly timestamp, showcased on F01 sensor data

5.2.2 DBSCAN results from using lagged variables

Similarly to results obtained from using the LOF algorithm and shifted variables, this part presents results from utilising data shifted by the method showcased in section 3.3.4.

5.2.2.1 Lagged variables and clog 1

Using lagged variables for clog 1 does not provide optimal results. No anomalous behaviour is detected prior to the clogging. No false positives are produced. The earliest detected timestamps during the clogging are showcased in the following table.

Table 5.4: DBSCAN anomaly timestamps for clog 1 using lagged variables, occurring at 05:45

Anomaly index	Timestamp
1	2021-01-12 05:36:34
2	2021-01-12 05:36:44
3	2021-01-12 05:36:54
4	2021-01-12 05:37:04

The following plots showcase the points presented in the table. Additional anomalies displayed are marked during the clogging event.

Figure 5.10: Clog 1, anomalies detected using DBSCAN and lagged variables, showcased on F01 sensor data

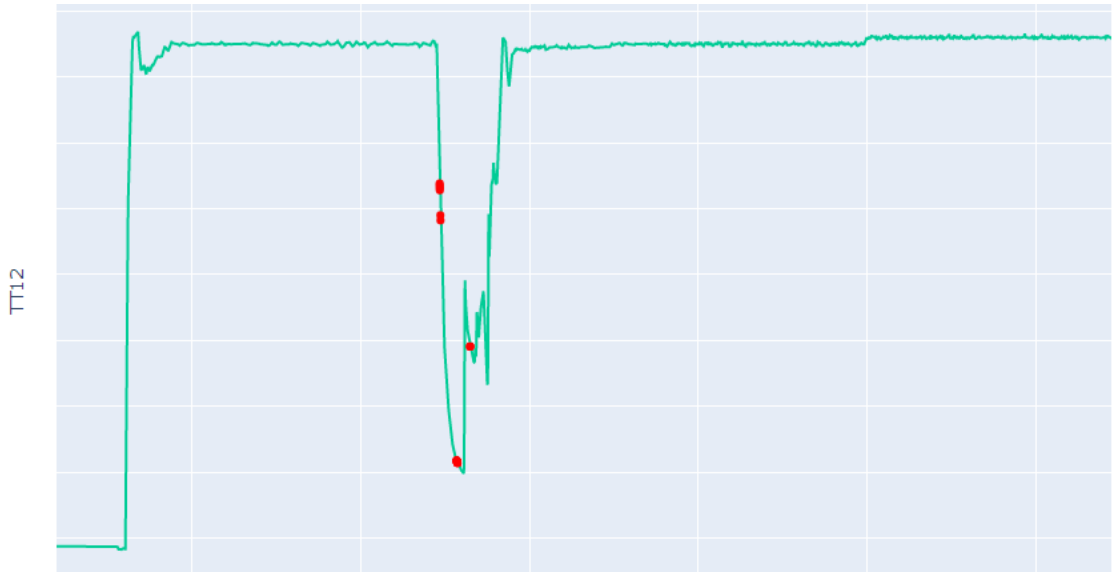


Figure 5.11: Clog 1, anomalies detected using DBSCAN and lagged variables, showcased on TT12 sensor data

The following figure presents anomaly index 1 in table 5.4, zoomed in for visibility.

Figure 5.12: Clog 1, earliest detected relevant anomaly timestamp in lagged data, showcased on F01 sensor data

5.3 Table of results

For ease of reference, the reported clogging times and earliest detected anomalies by LOF and DBSCAN are showcased in the following table. For tables and figures showcasing full results of respective algorithms, refer to Appendix B

Table 5.5: Reported clogging event times and earliest detected anomalies

Clogging event	Reported clogging event time	LOF: earliest detected relevant anomaly timestamp	DBSCAN: earliest detected relevant anomaly timestamp
1	2021-01-12 05:45	05:22:34	05:21:34
2	2021-02-01 08:40	08:37:56	08:37:56
3	2021-02-24 08:00	08:02:46	08:02:26
4	2021-03-01 17:00	16:58:34	16:58:24
6	2022-12-13 01:35	01:33	01:33

5.4 Feature importance results

The following tables present the most important features for achieving the results presented. A table with combined features importance across all clogging events for LOF and DBSCAN, and a secondary table of combined feature importance using the same algorithms on shifted data are showcased. These are derived using feature importance techniques described in chapter 4.

Table 5.6: Top 8 most important features for LOF and DBSCAN

Feature	Relative importance (%)
PT02	16.5
PT03	13.4
PT05	11.3
F01	10.0
PT07	9.3
TT12	7.6
TT06	5.1
TT20	4.6

The second table, presenting feature importance using shifted variables:

Table 5.7: Top 8 most important features for LOF and DBSCAN using shifted variables

Feature	Relative importance (%)
PT02	25.1
PT07	23.6
PT03	15.9
PT05	10.6
TT06	8.6
TT20	8.4
TT09	7.7
F01	6.5

Chapter 6

Discussion

This chapter aims to discuss the results and findings presented in chapter 5, focusing on the performance of the local outlier factor (LOF) and density-based clustering of applications with noise (DBSCAN) algorithms on industrial time series data provided by Bioco. The discussion revolves around the effectiveness of these methods in detecting anomalous behaviour prior to clogging events, the impact of using shifted variables, as well as potential future directions for research. As previously stated in section 3.3.4, the terms ”shifted” and ”lagged” are used interchangeably.

6.1 Performance in detecting anomalies

Both LOF and DBSCAN algorithms perform similarly in detecting anomalous behaviour in most cases. Neither algorithm consistently detected anomalies a notable time prior to the clogging events, which limits their practical use for early warning in Bioco's industrial process.

Based on the results obtained, both LOF and DBSCAN algorithms demonstrated some success in detecting anomalous behaviour prior to a clogging event. However, the ability to detect such behaviour in a timely manner prior to an event was limited. In most cases, the earliest detected anomalies coincided with the occurrence of the clogging events themselves.

For example, for clog 1, LOF and DBSCAN detected anomalous behaviour at 05:22:34 and 05:21:34, respectively. As shown in tables 5.5 and 3.2, clogging event 1 occurs at 5:45, which indicates that anomalous behaviour was detected a significant amount of time prior to the event. However, in reality, the process hits a point of no return around 05:23, and the clogging is in full motion at this point. This is indicated by the extreme drop in pressure in sensor F01 after the detected anomaly timestamps in the following figure.

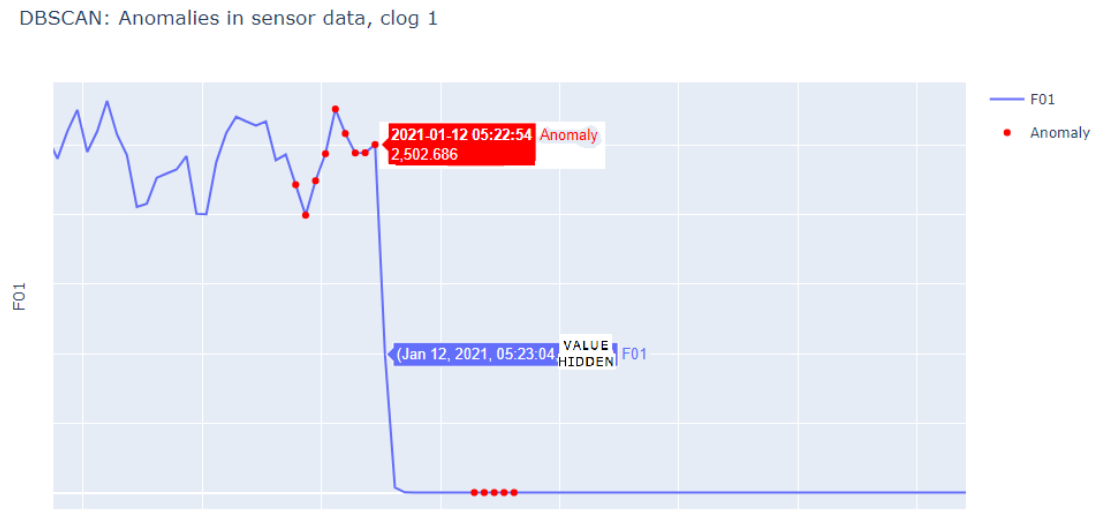


Figure 6.1: DBSCAN, clog 1, point of no return showcased on F01 data. The point of no return is between the marked anomaly and the blue-labeled data point

Similar behaviour is present for clog 3, 4 and 6. For clog 3, the point of no return occurs later, between 08:02:46 and 08:03:06. For clog 4, it is around 16:59:14. For clog 6, it is similarly earlier, at 1:33. For clog 2, the given time is correct, and the clogging occurs at 08:40. To make these easier to follow, these are indicated in the following table.

Table 6.1: Reported clogging events compared to points of no return

Clogging event	**Reported clogging event time**	**Point of no return**
1	2021-01-12 05:45	2021-01-12 05:23
2	2021-02-01 08:40	2021-02-01 08:40
3	2021-02-24 08:00	2021-02-24 08:02:46
4	2021-03-01 17:00	2021-03-01 16:59:14
6	2022-12-13 01:35	2022-12-13 01:33

This removes any bias related to results and clarifies the points at which detected anomalous behaviour is too late to prevent a clogging. A table comparing these points of no return to earliest relevant anomalous timestamps detected by LOF and DBSCAN algorithms is shown below.

Table 6.2: Points of no return compared to earliest detected anomalies

Clogging event	**Point of no return**	**LOF: earliest detected relevant anomaly timestamp**	**DBSCAN: earliest detected relevant anomaly timestamp**
1	2021-01-12 05:23	05:22:34	05:21:34
2	2021-02-01 08:40	08:37:56	08:37:56
3	2021-02-24 08:02:46	08:02:46	08:02:26
4	2021-03-01 16:59:14	16:58:34	16:58:24
6	2022-12-13 01:33	01:33	01:33

Examining this, earliest relevant detected anomalies by both LOF and DBSCAN are generally close to points of no return for most clogging events. This indicates that the algorithms were able to detect anomalous behaviour, but not sufficiently early to be considered as reliable early warning indicators. An early warning of 30 seconds will most likely have a limited value in a real production, as a human operator still has to process this warning and perform preventive action. Even an automatised response to such warnings implemented into the process will probably not have time to prevent a clogging event, unless a system is in place that directly injects a fluid or some sort of appropriate solvent, like water, into the area where a clogging regularly occurs.

The most notable result is seen for clog 2, where both algorithms detected anomalous behaviour 2 minutes prior to the points of no return. This is the most promising result, providing a potentially useful early warning time window. However, this is just a single instance. For the rest of the clogging events, anomalous behaviour was detected under a minute to, or at the exact same time of the point of no return, failing to provide sufficient time for preventive measures. Overall, DBSCAN outperformed LOF, albeit to a very small degree.

Preprocessing steps affected the performance of the algorithms. For example, the application of LOF to unfiltered data for clog 4 (table B.4) gave additional relevant anomalous timestamps. However, this variation did not generalise well to the rest of the data, resulting in a high amount of false positives during normal production. This highlights the need for carefully selected preprocessing steps that enhance a model's performance, while at the same time minimising sensitivity to noise.

The impact of false positives on the performance of the algorithms in Bioco's industrial setting should be considered. While a few false positives might be acceptable, an excess could lead to unintended consequences. In a decision support context, an operator receiving occasional false positives alert may perceive themselves as "smarter than the system" and remain engaged in the process, providing a sense of purpose and utility. However, a high frequency of false positives may cause the operator to lose focus and even ignore genuine alerts, potentially leading to overlooked critical events.

One approach to address this issue of false positives could be to develop a secondary filtering method, distinguishing between true and false positives. This could involve examining the flagged time steps and applying additional criteria or algorithms to determine their validity. Such a method could help reduce the number of false positives presented to an operator, increasing the reliability of the system.

Another approach involves the use of an ensemble of anomaly detectors. By incorporating multiple anomaly detection algorithms, the final decision of whether to flag an event as anomalous could be based on an agreement between the detectors. Only if a given proportion of the detectors agree that an event is anomalous, is it flagged for an operator's attention. This ensemble approach could enhance the overall robustness and accuracy of the system, leveraging the strengths of each individual algorithm and compensating for their weaknesses.

6.2 Effectiveness of using shifted variables

Incorporating variable shifting was done as an attempt to capture a synced representation of raw material passing through the system at a given time point. However, the results show that using lagged variables does not lead to improved performance for either LOF or DBSCAN. In most cases, the algorithms failed to detect anomalous behaviour prior to the clogging events.

Additionally, even the clogging events were not marked as anomalies by the algorithms in a timely manner.

It is possible that the choice of lag values or their incorporation into the feature set could be suboptimal. The values by which the features were shifted were based on the approximation of the location of these in Bioco's industrial process, which may have been inaccurate, indicating the need of further exploration. Using lagged variables with incorrect shift values might have a detrimental effect on the performance of the used algorithms. If the chosen shift values do not accurately reflect the time it takes for raw material to pass through different stages of Bioco's process, the resulting features essentially introduce noise and misleading information, causing the algorithms to miss important patterns related to the clogging events.

Additionally, the variation in input properties, such as tougher, more viscous raw material and other process conditions, such as changed process parameters, could lead to fluctuations in time it takes for the material to pass between different points in the process. This further complicates the selection of appropriate lag values, limiting the effectiveness of using shifted variables. An approach to solve this could involve analysing how the relationship between the lags of different sensors changes over time. Doing this, it might be possible to identify deviations in Bioco's process that can serve as early warning indicators for clogging events.

As it stands, it is beneficial to reassess the use of shifted variables in the current implementation, and explore alternative methods for capturing dynamic relationships between sensor data. This could involve further correlation testing for set data windows, assisted by using near infrared sensors to capture data about raw material properties. Additionally, other feature engineering techniques, better suited for variations in raw material properties and process conditions should be considered.

Another approach would be a proper implementation of autoencoders (2.5.5.1). These were briefly used in the analysis, but not explored on a proper level because of limitations in computational resources, discussed later. However, as autoencoders learn a lower-dimensional representation of the input data, this might reveal hidden patterns or relationships not evident in the raw sensor data.

In conclusion, while the use of shifted variables in the current implementation has not given significant improvements in the performance of LOF and DBSCAN algorithms, there are several alternative approaches that should be explored to better capture the relationships between sensor data in Bioco's process. By combining the mentioned techniques with domain expertise, it may be possible to develop more effective strategies for shifting sensor data and improving the detection of anomalous behaviour in Bioco's industrial process.

6.3 Sensitivity to parameter tuning

The optimal $n\_neighbors$ parameter in LOF and $MinPts$ parameter in DBSCAN essentially represent the same thing. In general, for Bioco's data, setting $n\_neighbors$ and $MinPts$ parameters between 1.2-1.5 times the number of features used gives a model that tries to capture pre-clogging behaviour, with a minimal amount of false positives. Setting this parameter to values lower than the amount of features in the data used yields a model that seems to detect anomalous behaviour earlier than the results showcased in chapter 5. However, such a model essentially flags much of the regular production as anomalous behaviour, produces false positives, and does not generalise well between data windows. The parameter ϵ for DBSCAN is derived from finding the optimal $MinPts$ parameter. No notable improvement in performance

is noted from adjusting this parameter manually.

For LOF, the sensitivity of setting the *contamination* parameter is an interesting topic. This parameter essentially sets the proportion of anomalies in the data set, directly influencing number of detected anomalies. As we have prior knowledge of the clogging events, setting this parameter value based on known events essentially introduces hindsight bias, resulting in biased results. The trick used to avoid this was to include starts, end, and inactive periods in the industrial process in data used. The idea is that the pre-clogging, anomalous behaviour is caught in the results, and the rest of the "anomaly quota" is filled by irrelevant periods. For implementation in the actual industrial process, where data from inactive periods might not be utilised, setting the parameter to a very low value, below 0.001, should yield optimal results. In praxis, a value of 0.001 means that there is an anomalous point for every 1000 points. With a sample rate of 6 sensor values a minute, this translates to an anomaly every 167 minutes, or roughly 3 hours of production. If a window size of at least 30000 data points, or just above 83 hours of production is used, 30 anomalous points are detected. This value should, in theory, be appropriate for potential clogging prevention. For a sampling rate of 1 sensor value a minute, the *contamination* parameter should be adjusted to 0.006 and lower, representing the same amount of production time. However, this is only a speculation, as this was not thoroughly tested.

6.4 Feature importance evaluation

Feature importance results shown in 5.4 provide insight into which features contribute the most to the performance of the LOF and DBSCAN algorithms. However, it is important to consider that these results are based on the suboptimal performance achieved in detecting anomalies prior to clogging events. Thus, the identified important features may not necessarily be the optimal features for detecting anomalous behaviour early enough to prevent such events.

Examining the top features presented in table 5.6, PT02, PT03, and PT05 have the highest relative importance. These pressure-related features seem to be more informative for the algorithms compared to the temperature and flow-related features, which might be attributed to the nature of clogging events being more closely related to changes in pressure. However, the fact that the algorithms did not provide satisfactory early warnings for clogging events suggests that relying solely on these features might not be sufficient.

Incorporating shifted variables, as shown in table 5.7, PT02 and PT07 become the most important features, followed by PT03 and PT05. The increase of importance of PT07 and the emergence of TT06 and TT20 among the top features suggest that using lagged variables may provide additional information to the algorithms. Nevertheless, the use of shifted variables did not produce any reliable early warnings for clogging events, indicating the previously mentioned need of an alternative approach to lagging these variables.

6.5 Limitations and challenges

One of the primary limitations encountered in this work was the lack of sufficient computational resources. The machine used for the analysis was an outdated unit, which significantly impacted the feasibility of employing more complex models, such as autoencoders. Due to the considerable amount of time and computational resources required to set up and utilise these advanced models, a decision was made to focus on simpler models that appeared more promising, given the available resources.

This limitation influenced the choice of algorithms and methods employed, potentially affecting the success in detecting pre-clogging anomalous behaviour. As a result, the findings may not fully represent the capabilities of more advanced models that could have been explored if computational resources were less limited.

6.6 Future Work

Future work in this area should prioritise addressing the limitations encountered in this work and exploring new techniques and strategies that could enhance the performance of anomaly detection algorithms in industrial time series data.

With increased computational resources, more sophisticated and computationally expensive methods, such as autoencoders and other deep learning techniques, can be thoroughly investigated and tested on Bioco's data. This would also allow for the use of larger data windows, enabling more extensive performance evaluation and better understanding of the relationships within the sensor data.

The idea of using ensemble methods or combining different anomaly detection algorithms should be explored further. By leveraging the strengths of multiple algorithms, it is thought to be possible to achieve better overall performance in detecting anomalous behaviour prior to a clogging event.

Future research should explore additional feature engineering and selection strategies that could improve the performance of the algorithms. Looking into alternative techniques for incorporating lagged variables may yield better results, and ensemble methods that utilise both regular and lagged data should be considered.

The impact of different preprocessing steps on algorithm performance should be systematically evaluated, as the results in this work indicate that these steps can significantly influence anomaly detection outcomes. By understanding how preprocessing choices affect the results, more informed decisions can be made when selecting the most appropriate preprocessing techniques.

Additionally, future work should focus on minimising the occurrence of false positives while maintaining the ability to detect true anomalies. This, as previously discussed, could involve developing advanced warning systems that can differentiate between true and false positives, improving the reliability and usability of the anomaly detection system.

Chapter 7

Conclusion

The aim of this thesis was to investigate the application of anomaly detection algorithms for early detection of clogging events in an industrial setting, specifically for Bioco's production process. The main goal was to identify anomalous behaviour in time series sensor data prior to clogging events, with the aim of allowing preventive actions to be taken. To achieve this, two unsupervised learning algorithms, LOF and DBSCAN, were employed and their performance was evaluated in terms of their ability to detect anomalies in the data.

The results revealed that both LOF and DBSCAN were able to detect anomalous behaviour in most cases, but their performance was not consistent in providing sufficient early warnings for clogging events. The earliest detected anomalies often coincided with the point of no return or occured too close to the clogging event itself, limiting the practical use of these algorithms for early warning purposes.

Feature importance analysis indicated that certain features contributed more to the detection of anomalies than others. However, the most important features identified were not necessarily the most effective in providing early warnings of clogging events. This highlights the need for further investigation into feature selection and engineering techniques that may enhance the performance of anomaly detection algorithms.

Several limitations were encountered during this research, including limited computational resources, which constrained the exploration of more advanced techniques such as autoencoders and other deep learning methods. Additionally, preprocessing steps were found to have a significant impact on the performance of the algorithms, indicating that a systematic evaluation of different preprocessing techniques is needed to optimise anomaly detection results.

Future work should focus on addressing these limitations and exploring new strategies for improving anomaly detection in industrial time series data. This includes investigating more advanced algorithms and feature engineering techniques, ensemble methods, and refining warning systems to address false positives. By building upon the findings of this thesis, it is hoped that more effective and reliable anomaly detection systems can be developed to aid in the prevention of clogging events in Bioco's industrial process and improve overall process control in their industrial setting.

Bibliography

[1] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM Computing Surveys (CSUR)* 41.3 (2009), pp. 1–58. URL: https://dl.acm.org/doi/10.1145/1541880.1541882.

[2] Neil Caithness and David Wallom. "Anomaly Detection for Industrial Big Data". In: *Proceedings of the 7th International Conference on Data Science, Technology and Applications*. SCITEPRESS - Science and Technology Publications, 2018. DOI: 10.5220/0006835502850293. URL: https://arxiv.org/abs/1804.02998.

[3] Victoria J Hodge and Jim Austin. "A survey of outlier detection methodologies". In: *Artificial Intelligence Review* 22.2 (2004), pp. 85–126. URL: https://link.springer.com/article/10.1007/s10462-004-4304-y.

[4] Walter A Shewhart. "Economic control of quality of manufactured product". In: *Bell System Technical Journal* 10.2 (1931), pp. 500–501.

[5] David S Moore, George P McCabe, and Bruce A Craig. *Introduction to the Practice of Statistics*. 7th ed. W.H. Freeman, 2012.

[6] Harold Hotelling. "The Generalization of Student's Ratio". In: *The Annals of Mathematical Statistics* 2.3 (1931), pp. 360–378. DOI: 10.1214/aoms/1177732979. URL: https://doi.org/10.1214/aoms/1177732979.

[7] Frederik Michel Dekking et al. *A Modern Introduction to Probability and Statistics*. 1st ed. Springer London, 2010, pp. 234–236. DOI: 10.1007/1-84628-168-7.

[8] Steven Walfish. "A review of statistical outlier methods". In: *Pharmaceutical technology* 30.11 (2006), p. 82. URL: http://www.statisticaloutsourcingservices.com/Outlier2.pdf.

[9] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. 3rd ed. OTexts, 2021.

[10] G.E.P. Box et al. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley, 2015. ISBN: 9781118674925. URL: https://books.google.no/books?id=rNt5CgAAQBAJ.

[11] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016. URL: https://mitpress.mit.edu/9780262035613/deep-learning.

[13] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT Press, 2012.

[14] Trevor Hastie, Rober Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. 2nd ed. Springer Science & Business Media, 2009. DOI: https://doi.org/10.1007/978-0-387-84858-7.

[15] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. 3rd ed. Springer, 2016. DOI: https://doi.org/10.1007/978-3-319-29854-2.

[16] J. B. MacQueen. "Some Methods for Classification and Analysis of MultiVariate Observations". In: *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1967, pp. 281–297. URL: http://projecteuclid.org/euclid.bsmsp/1200512992.

[17] Martin Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[18] T. Cover and P. Hart. "Nearest neighbor pattern classification". In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27. DOI: 10.1109/TIT.1967.1053964.

[19] Markus Breunig et al. "LOF: Identifying Density-Based Local Outliers." In: vol. 29. June 2000, pp. 93–104. DOI: 10.1145/342009.335388.

[20] J Ross Quinlan. "Induction of decision trees". In: *Machine learning* 1.1 (1986), pp. 81–106.

[21] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation Forest". In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[22] Chong Zhou and Randy C. Paffenroth. "Anomaly Detection with Robust Deep Autoencoders". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '17. Halifax, NS, Canada: Association for Computing Machinery, 2017, pp. 665–674. ISBN: 9781450348874. DOI: 10.1145/3097983.3098052. URL: https://doi.org/10.1145/3097983.3098052.

[23] Raghavendra Chalapathy and Sanjay Chawla. "Deep learning for anomaly detection: A survey". In: *arXiv preprint arXiv:1901.03407* (2019). URL: https://arxiv.org/abs/1901.03407.

[24] Pankaj Malhotra et al. "LSTM-based encoder-decoder for multi-sensor anomaly detection". In: *arXiv preprint arXiv:1607.00148* (2016). URL: https://arxiv.org/pdf/1607.00148.pdf.

[25] Markus Goldstein and Seiichi Uchida. "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data". In: *PloS one* 11.4 (2016), e0152173. URL: https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0152173&type=printable.

Appendix A

Table of Python-packages

Python-package name	Version	Purpose of use
”kneed”	0.8.3	Knee point identification
”matplotlib”	3.7.1	Plots and figures
”missingno”	0.5.2	Missing data visualisation
”numpy”	1.24.3	Array computing
”pandas”	2.0.1	Data analysis
”plotly”	5.14.1	Interactive plots and figures
”scikit-learn”	1.2.2	Mainly for anomaly detection, model tuning and feature importance
”seaborn”	0.12.0	Mainly for correlation and lag identification and figure styling
”statsmodels”	0.14.0	Statistical computations and data exploration

Table A.1: Python-packages used during data exploration, data processing, data visualisation, feature selection, feature engineering, anomaly detection and feature importance, the respective version used as well as the purpose of their use.

Appendix B

Continuation of results

B.1 Local outlier factor results

B.1.1 Results without lagging of variables

B.1.1.1 Clog 2

For the second clogging, earliest anomaly is identified a significant amount prior to the clogging event. However, this is attributed to the process being in the startup phase, indicating a false positive. Subsequent points are identified right as the event happens.

Table B.1: LOF anomaly timestamps for clog 2, occurring at 08:40

Anomaly index	Timestamp
8	2021-02-01 07:43:46
9	2021-02-01 08:37:56
10	2021-02-01 08:38:06
11	2021-02-01 08:38:46
12	2021-02-01 08:38:56
13	2021-02-01 08:39:06
14	2021-02-01 08:39:16

The following plots showcase the points presented in the table, as well as additional false positives, prior to the clogging event. Additional anomalous points displayed are not relevant, as they are marked during the clogging.

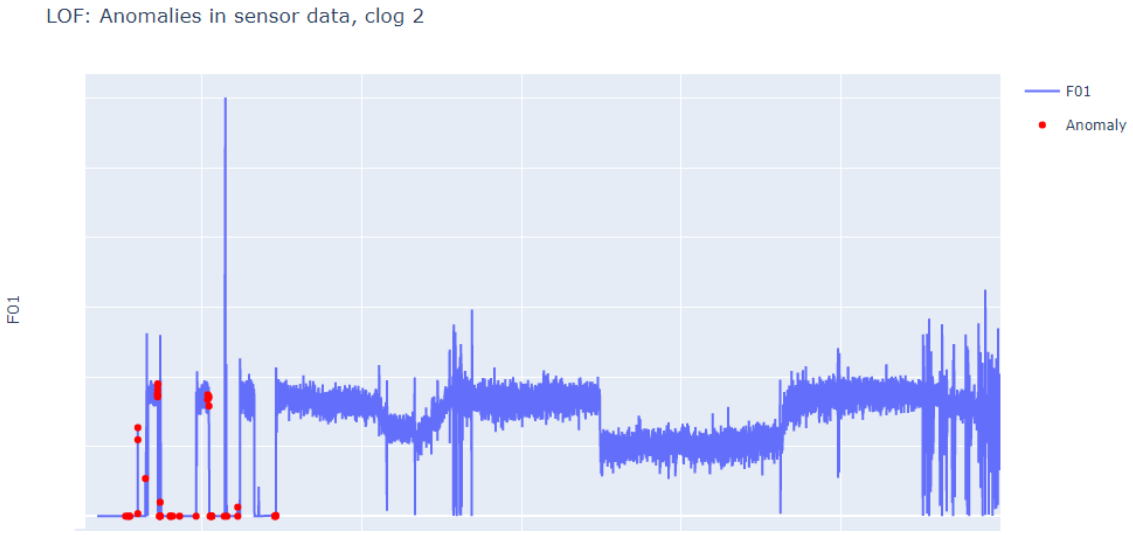


Figure B.1: Clog 2, anomalies detected using LOF, showcased on F01 sensor data

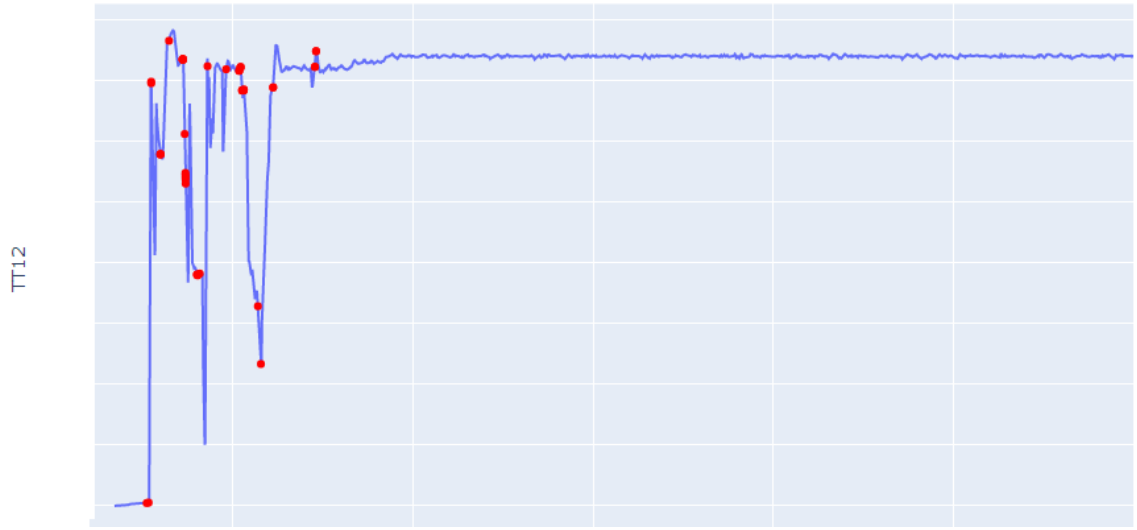


Figure B.2: Clog 2, anomalies detected using LOF, showcased on TT12 sensor data

The following figure presents the earliest relevant timestamp prior to the clogging, anomaly index 9 in table B.1, zoomed in for visibility.

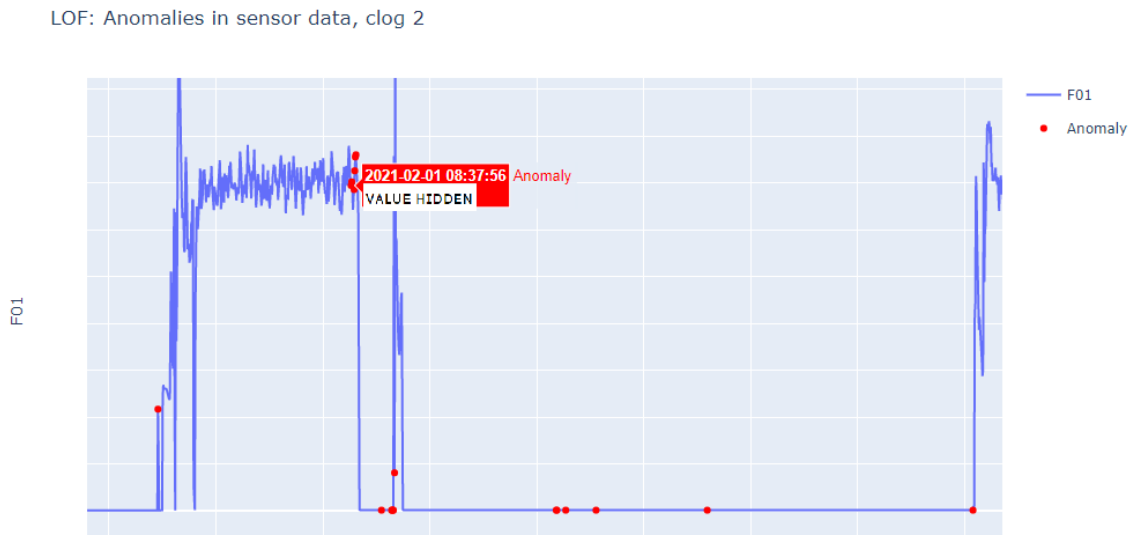


Figure B.3: Clog 2, earliest detected relevant anomaly timestamp, showcased on F01 sensor data

B.1.1.2 Clog 3

For the third clogging, earliest anomalies are identified more than 40 minutes prior to the event. However, this is attributed to a manual change in process parameters in that period. Subsequent anomalies are detected as the clogging occurs.

Table B.2: LOF anomaly timestamps for clog 3, , occurring at 08:00

Anomaly index	Timestamp
1	2021-02-24 07:15:56
2	2021-02-24 07:16:06
3	2021-02-24 07:24:06
4	2021-02-24 08:02:46
5	2021-02-24 08:02:56

The following plots showcase the points presented in the table. Additional anomalous points displayed after the significant drop in values are not relevant, as they are marked during the clogging.

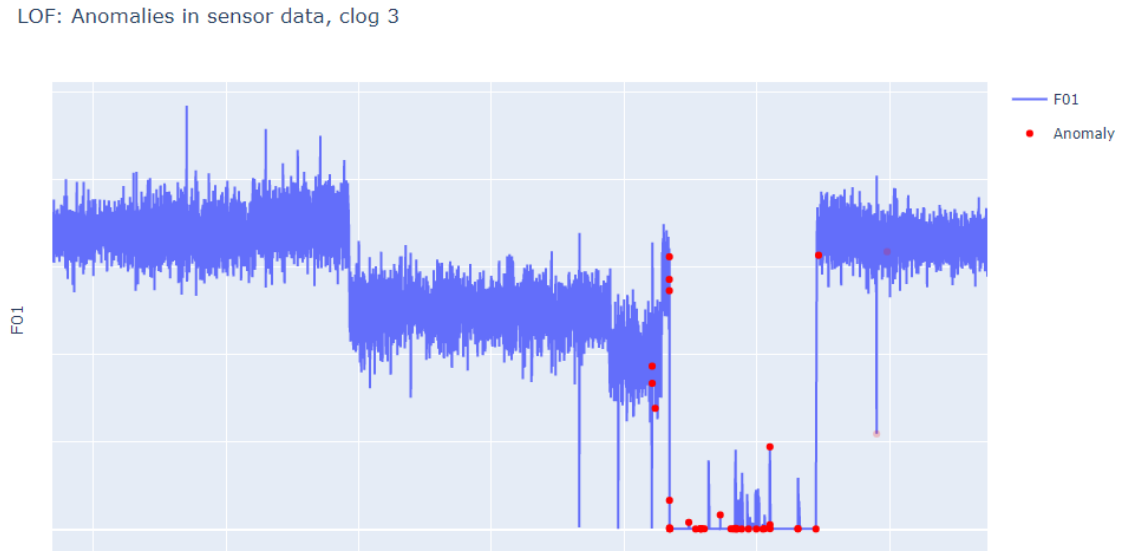


Figure B.4: Clog 3, anomalies detected using LOF, showcased on F01 sensor data

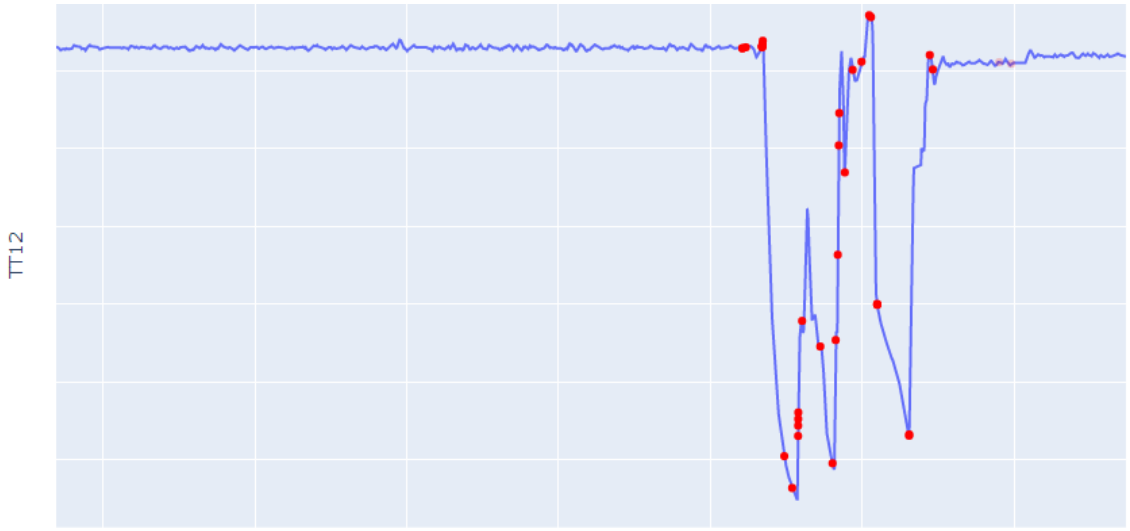


Figure B.5: Clog 3, anomalies detected using LOF, showcased on TT12 sensor data

The following figure presents the earliest timestamp prior to the clogging, anomaly index 1 in table B.2, zoomed in for visibility.

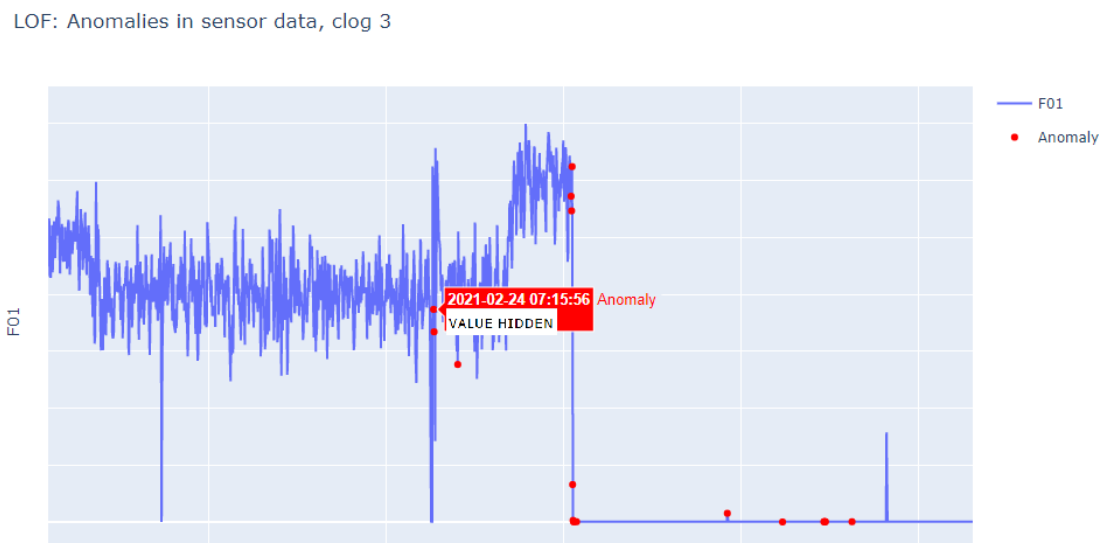


Figure B.6: Clog 3, earliest detected relevant anomaly timestamp, showcased on F01 sensor data

B.1.1.3 Clog 4

Utilising the most optimal model on filtered and denoised data, the results are as shown in the following table.

Table B.3: LOF anomaly timestamps for clog 4, occurring at 17:00

Anomaly index	Timestamp
4	2021-03-01 16:58:34
5	2021-03-01 16:58:44
6	2021-03-01 16:58:54

No anomalous behaviour is detected prior to the clogging. The anomalous points are flagged too late, right as the clog happens. The following plots showcase this.

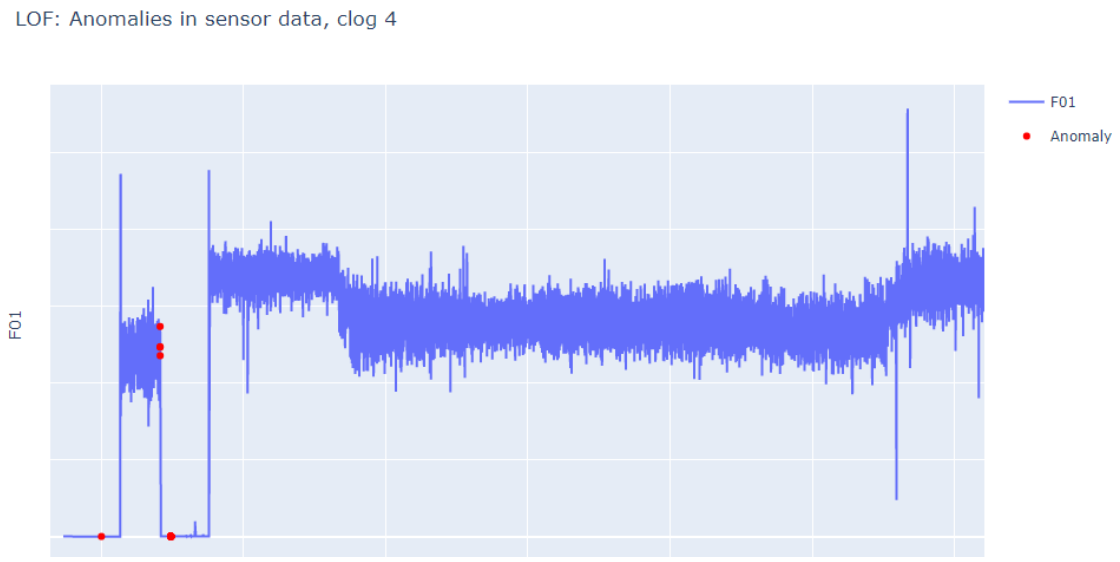


Figure B.7: Clog 4, anomalies detected using LOF, showcased on F01 sensor data

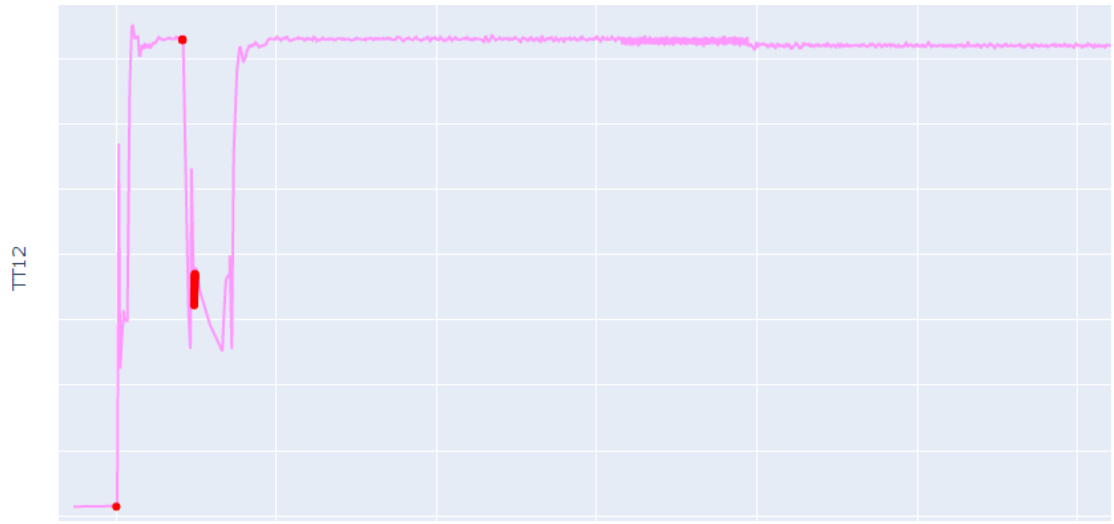


Figure B.8: Clog 4, anomalies detected using LOF, showcased on TT12 sensor data

The following figure presents the earliest timestamp detected, anomaly index 4 in table B.3, zoomed in for visibility.

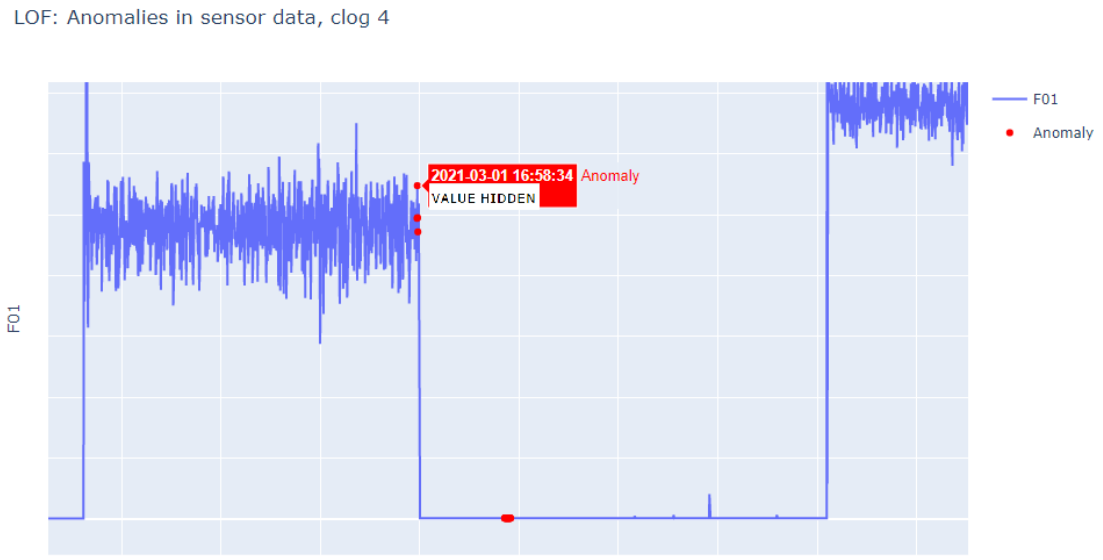


Figure B.9: Clog 4, earliest detected relevant anomaly timestamp, showcased on F01 sensor data

However, if utilised on unfiltered data, two additional timestamps prior to the clogging are produced.

Table B.4: LOF anomaly timestamps for clog 4 on unfiltered data, occurring at 17:00

Anomaly index	**Timestamp**
13	2021-03-01 15:34:44
14	2021-03-01 16:18:44
15	2021-03-01 16:58:24
16	2021-03-01 16:58:34

However, this variation of implementation does not generalise well on the rest of the data, resulting in false positives during normal production and a sensitive model. This is showcased in the following plots of temperature sensor TT12.

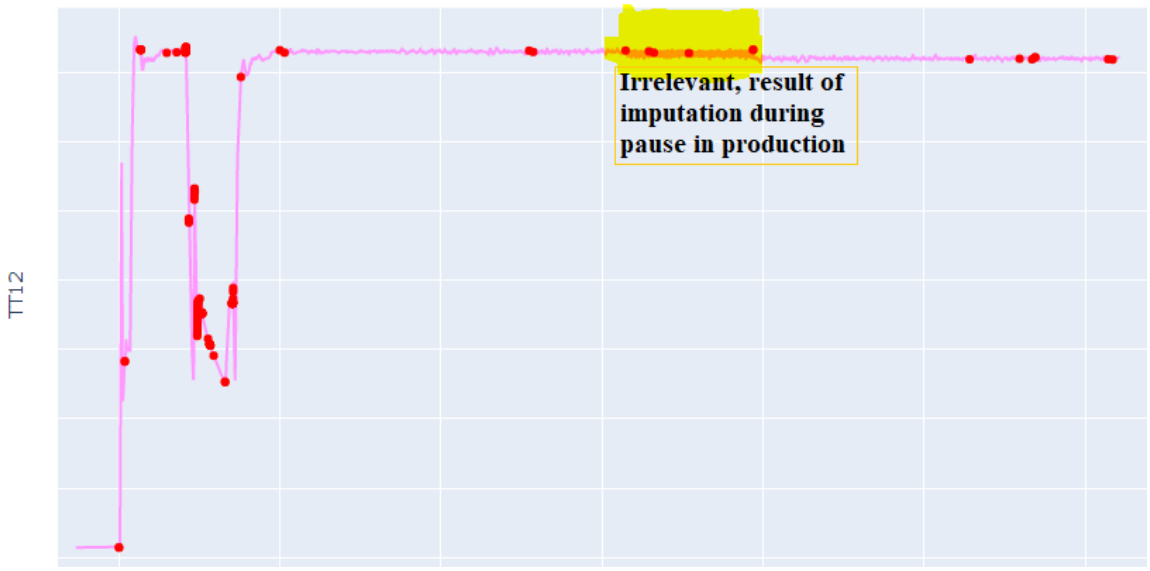


Figure B.10: Clog 4, anomalies detected using LOF on unfiltered data, showcased on sensor TT12

The yellow-marked area is irrelevant, caused by choice of imputation technique. All other anomalous points detected prior to and after the clogging, excluding this, are false positives. The following figure presents the earliest timestamps prior to the clogging detected on unfiltered data, anomaly indexes 13 and 14 in table B.4, zoomed in for visibility.

Figure B.11: Clog 4, earliest detected relevant anomaly timestamp using unfiltered data, showcased on sensor TT12

B.1.1.4 Clog 6

For the sixth clogging and 2022 data, earliest anomalies are identified right as the event happens. No anomalous behaviour is detected a notable amount prior to the clogging. No false positives are produced.

Table B.5: LOF anomaly timestamps for clog 6, occurring at 01:35

Anomaly index	Timestamp
1	2022-12-13 01:33:00
2	2022-12-13 01:34:00

The following plots showcase the points presented in the table. Additional anomalous points displayed are not relevant, as they are marked during the clogging.

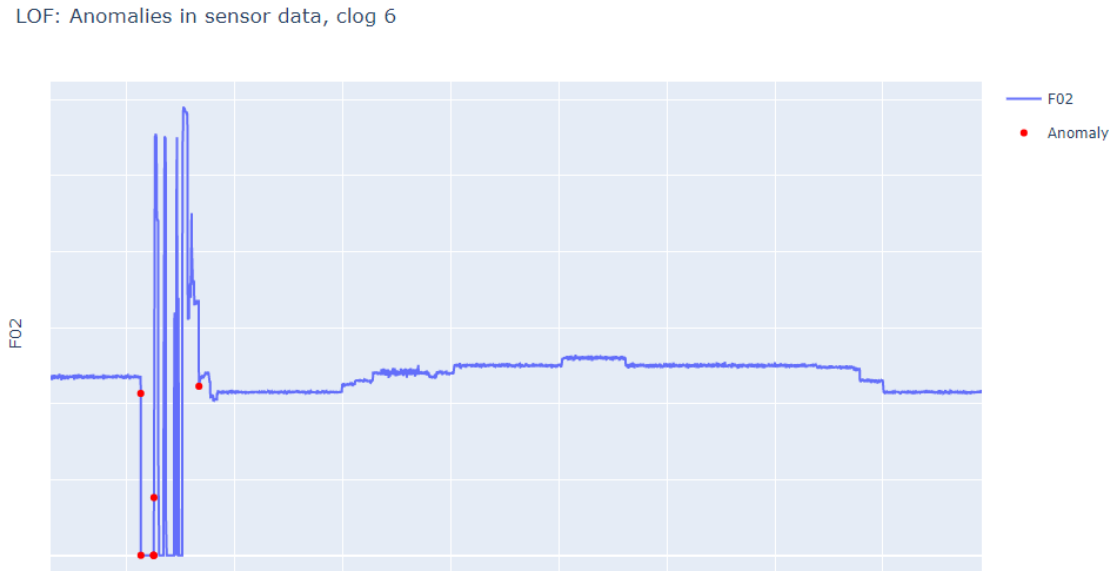


Figure B.12: Clog 6, anomalies detected using LOF, showcased on F01 sensor data

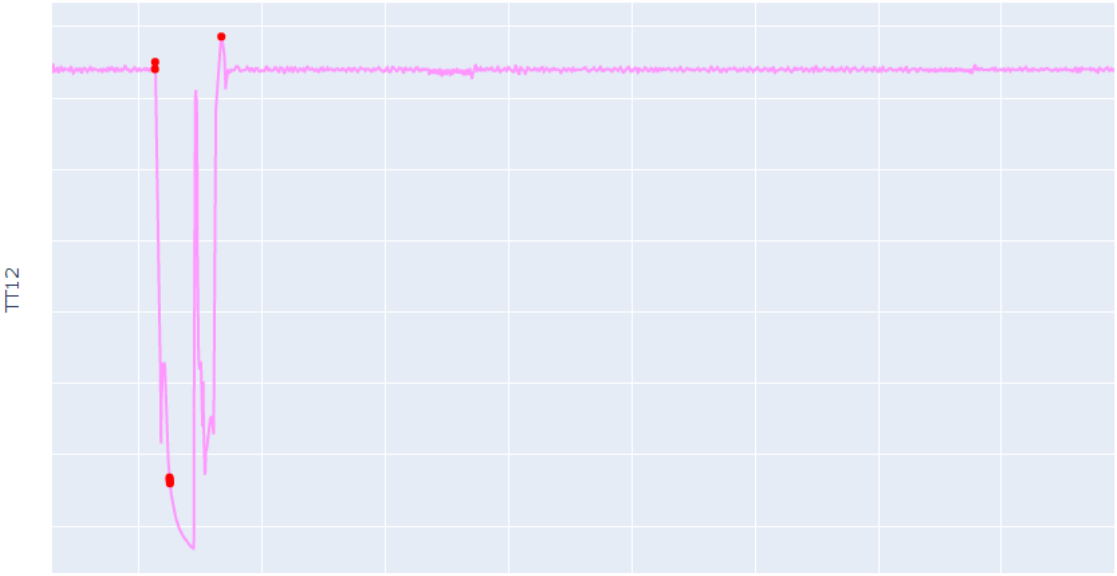


Figure B.13: Clog 6, anomalies detected using LOF, showcased on TT12 sensor data

The following figure presents the earliest timestamp prior to the clogging, anomaly index 1 in table B.5, zoomed in for visibility.

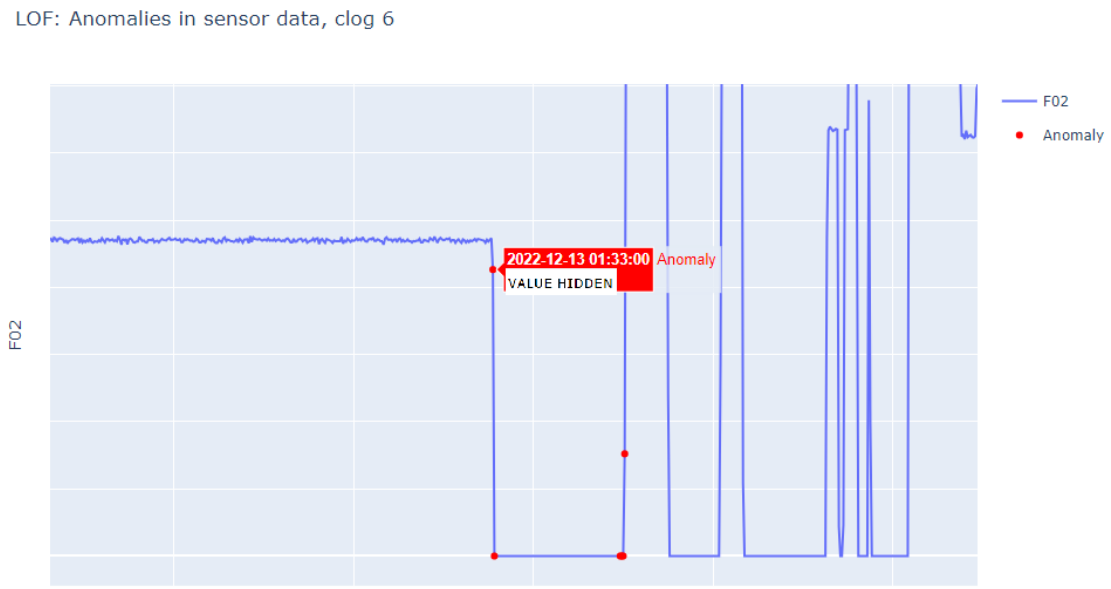


Figure B.14: Clog 6, earliest detected relevant anomaly timestamp, showcased on F01 sensor data

B.1.2 LOF results from using lagged variables

B.1.2.1 Lagged variables and clog 2

For clog 2, the lagged variables provide interesting results, showcased in the following table.

Table B.6: LOF anomaly timestamps for clog 2 using lagged variables, occurring at 08:40

Anomaly index	Timestamp
16	2021-02-01 07:54:56
17	2021-02-01 08:05:06
18	2021-02-01 08:05:16
19	2021-02-01 08:05:36
20	2021-02-01 08:48:56

Anomalous behaviour is detected more than 40 minutes prior to the clogging event, with multiple consecutive timestamps. However, this is attributed to the clogging happening early after the start of the industrial process. The lagged variables early in the process are using data from the start of the process, while variables later are using data from regular operation. This is indicated in the following figures, showing an abundance of points flagged as anomalies.

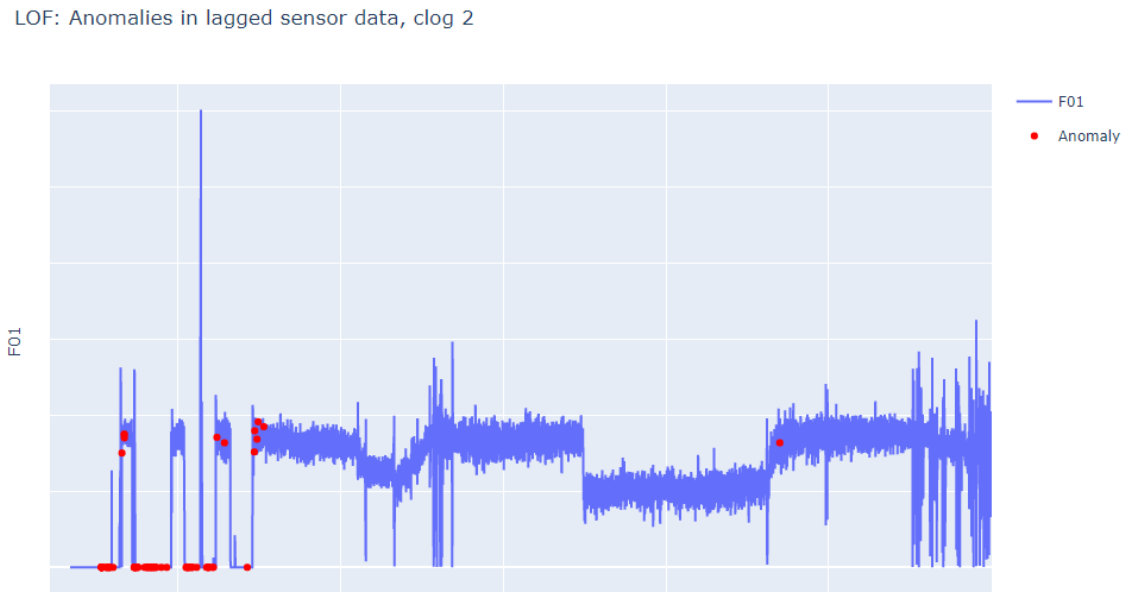


Figure B.15: Clog 2, anomalies detected using LOF and lagged variables, showcased on F01 sensor data

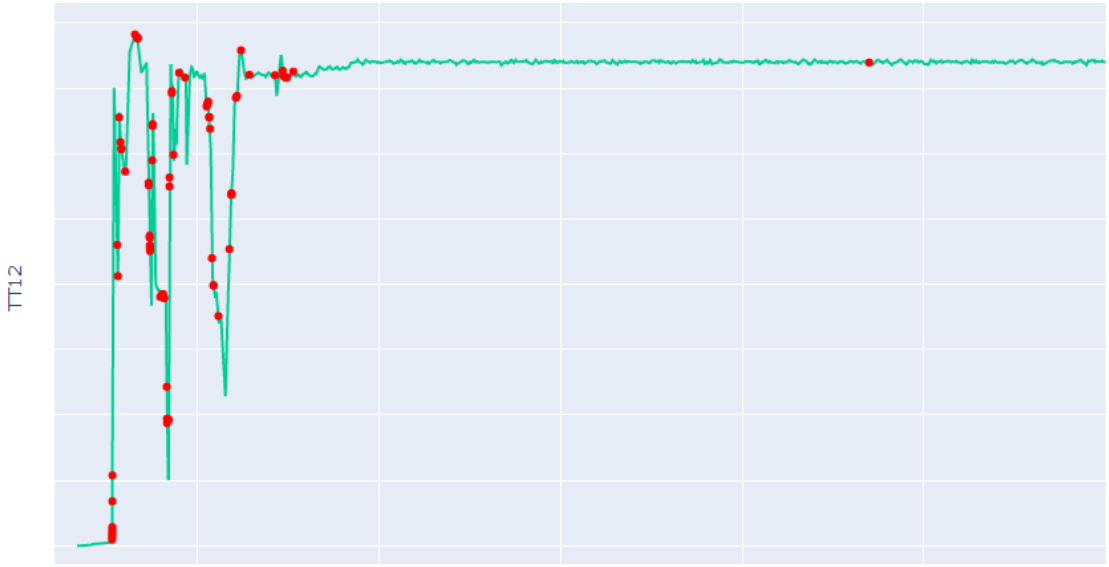


Figure B.16: Clog 2, anomalies detected using LOF and lagged variables, showcased on TT12 sensor data

The following figure presents anomaly index 16 in table B.6, zoomed in for visibility.

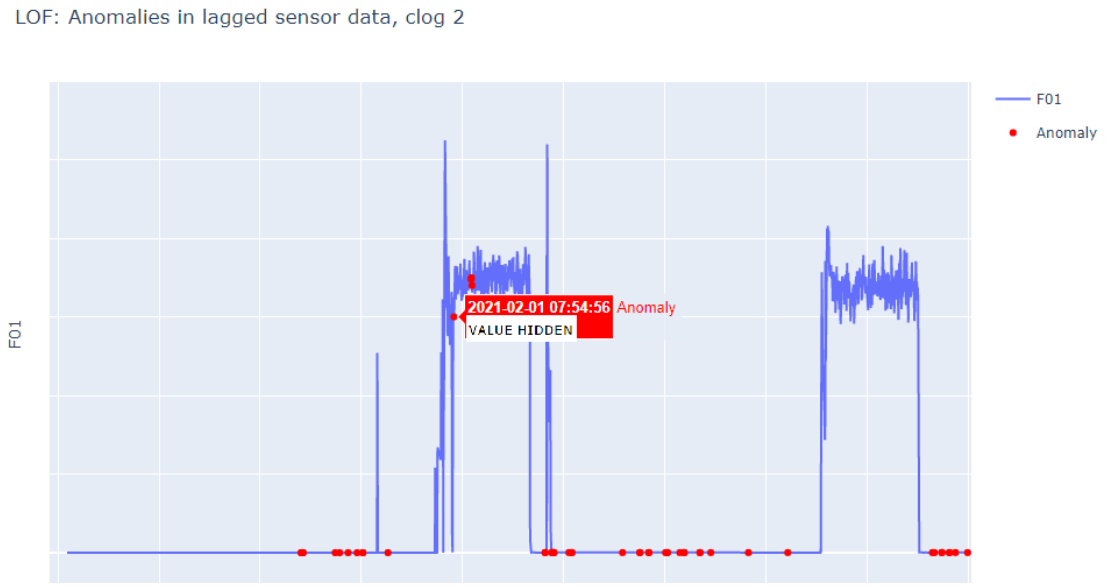


Figure B.17: Clog 2, earliest detected relevant anomaly timestamp in lagged data, showcased on F01 sensor data

B.1.2.2 Lagged variables and clog 3

For clog 3, using lagged variables does not provide any relevant results. Only the clogging event is detected by the model, as showcased in the following table.

Table B.7: LOF anomaly timestamps for clog 3 using lagged variables, occurring at 08:00

Anomaly index	Timestamp
2	2021-02-24 08:10:56
3	2021-02-24 08:11:06

The following plots showcase the points presented in the table. Additional anomalies displayed are marked during the clogging. A false positive is also present during normal operation.

Figure B.18: Clog 3, anomalies detected using LOF and lagged variables, showcased on F01 sensor data

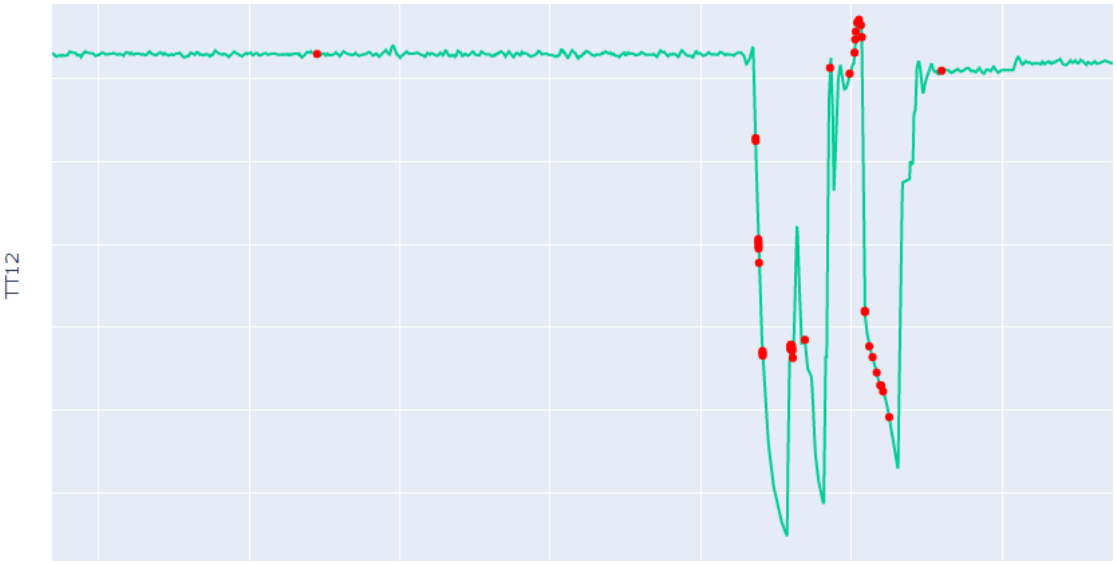


Figure B.19: Clog 3, anomalies detected using LOF and lagged variables, showcased on TT12 sensor data

The following figure presents anomaly index 2 in table B.7, zoomed in for visibility.

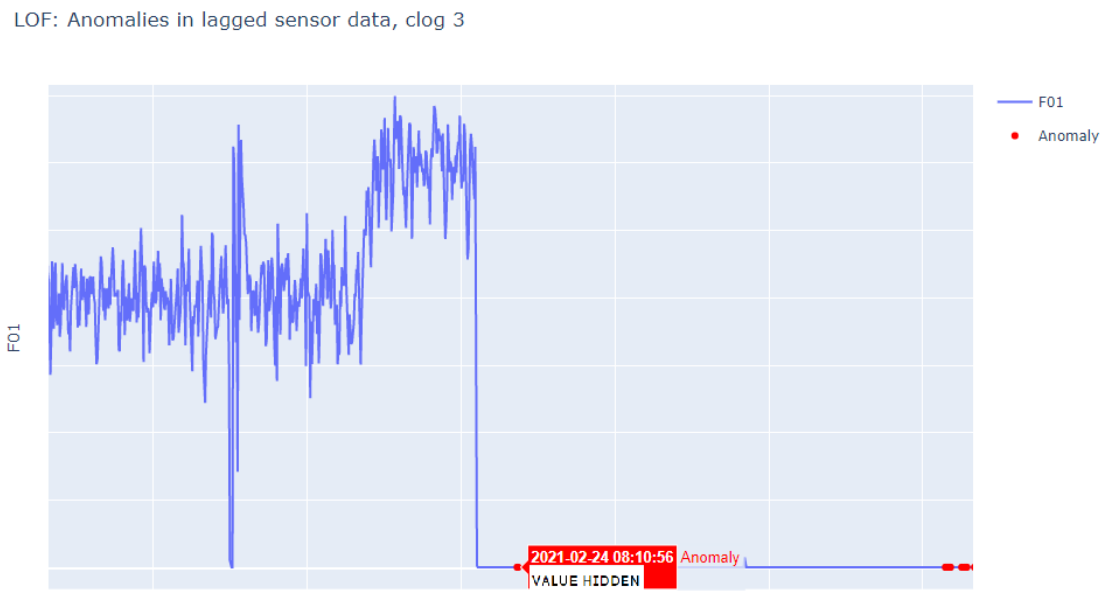


Figure B.20: Clog 3, earliest detected relevant anomaly timestamp in lagged data, showcased on F01 sensor data

B.1.2.3 Lagged variables and clog 4

Using lagged variables for clog 4 similarly does not produce optimal results. No anomalous behaviour is detected prior to the clogging. The earliest anomalous instances are detected during the event itself. Additionally, using lagged variables creates multiple false positives during regular production.

Table B.8: LOF anomaly timestamps for clog 4 using lagged variables, occurring at 17:00

Anomaly index	Timestamp
9	2021-03-01 17:23:24
10	2021-03-01 17:23:34
11	2021-03-01 17:23:44

The following plots showcase the points presented in the table. Additional anomalies displayed are marked at the start of the process and during the clogging. Mutiple false positives are also present during normal operation.

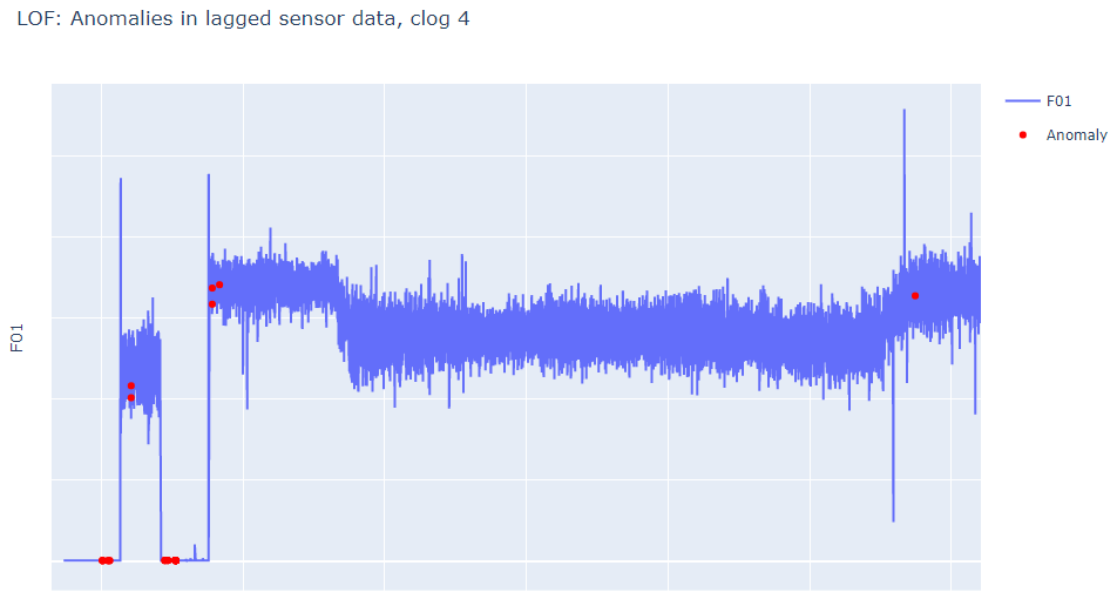


Figure B.21: Clog 4, anomalies detected using LOF and lagged variables, showcased on F01 sensor data

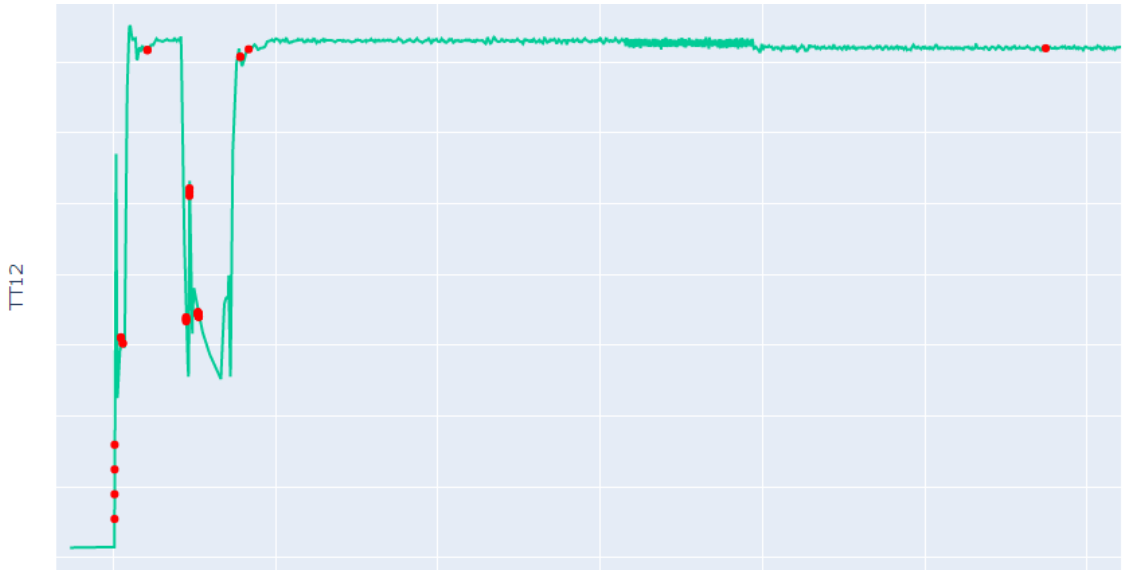


Figure B.22: Clog 4, anomalies detected using LOF and lagged variables, showcased on TT12 sensor data

The following figure presents anomaly index 9 in table B.8, zoomed in for visibility.

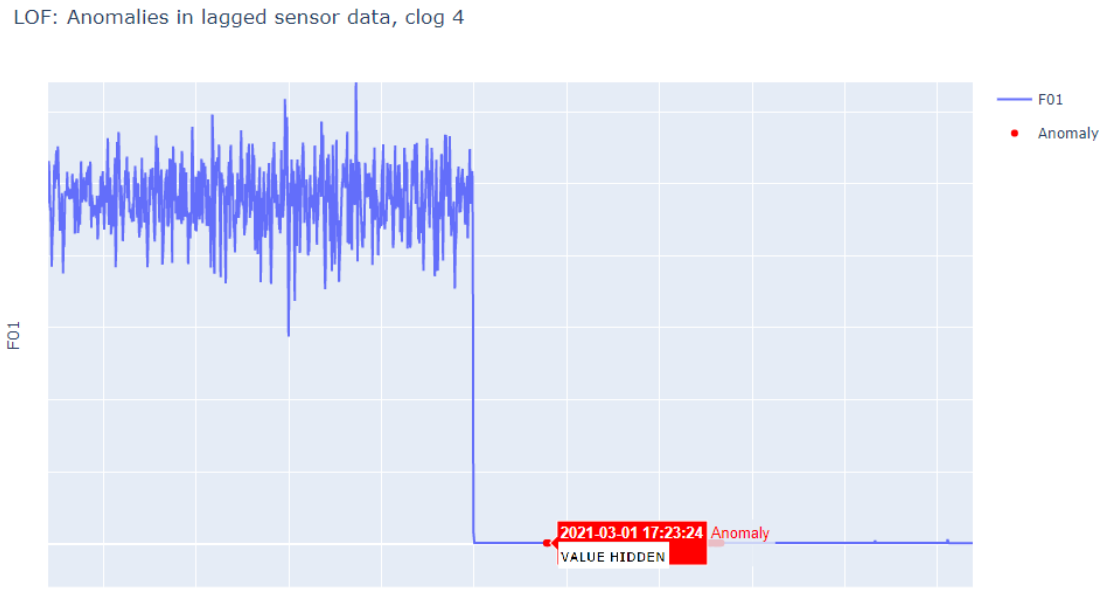


Figure B.23: Clog 4, earliest detected relevant anomaly timestamp in lagged data, showcased on F01 sensor data

B.1.2.4 Lagged variables and clog 6

Unsatisfactory results of applying the model to clog 6 data with lagged variables, no anomalous behaviour detected prior to the clogging. The table showcases earliest anomalous timestamps.

Table B.9: LOF anomaly timestamps for clog 6 using lagged variables, occurring at 01:35

Anomaly index	Timestamp
1	2022-12-13 01:48:00
2	2022-12-13 01:58:00
3	2022-12-13 02:00:00

The following plots showcase the points presented in the table. Additional anomalies displayed are marked during the clogging.

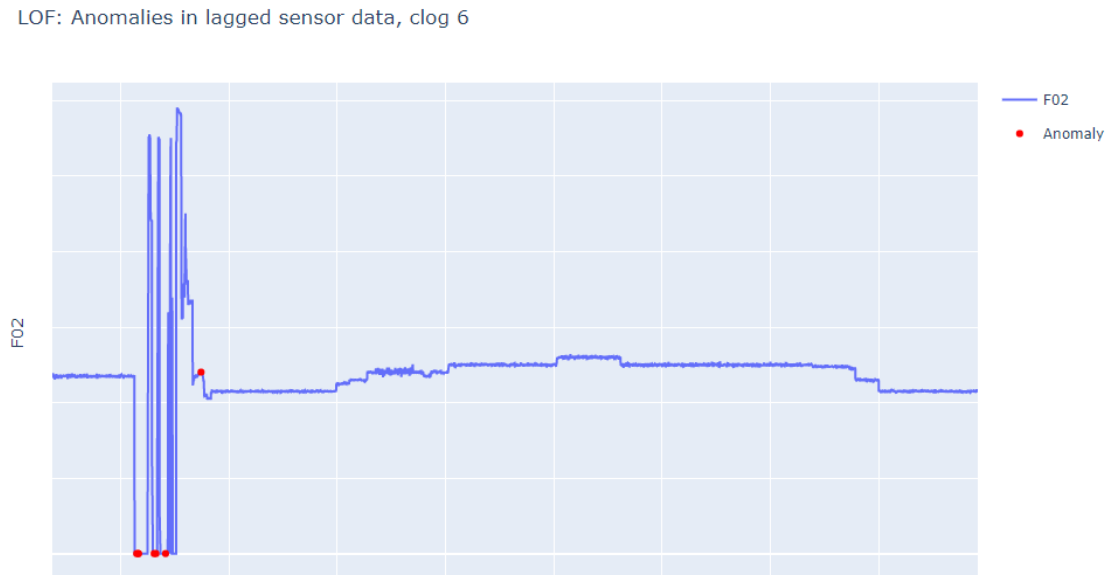


Figure B.24: Clog 6, anomalies detected using LOF and lagged variables, showcased on F01 sensor data

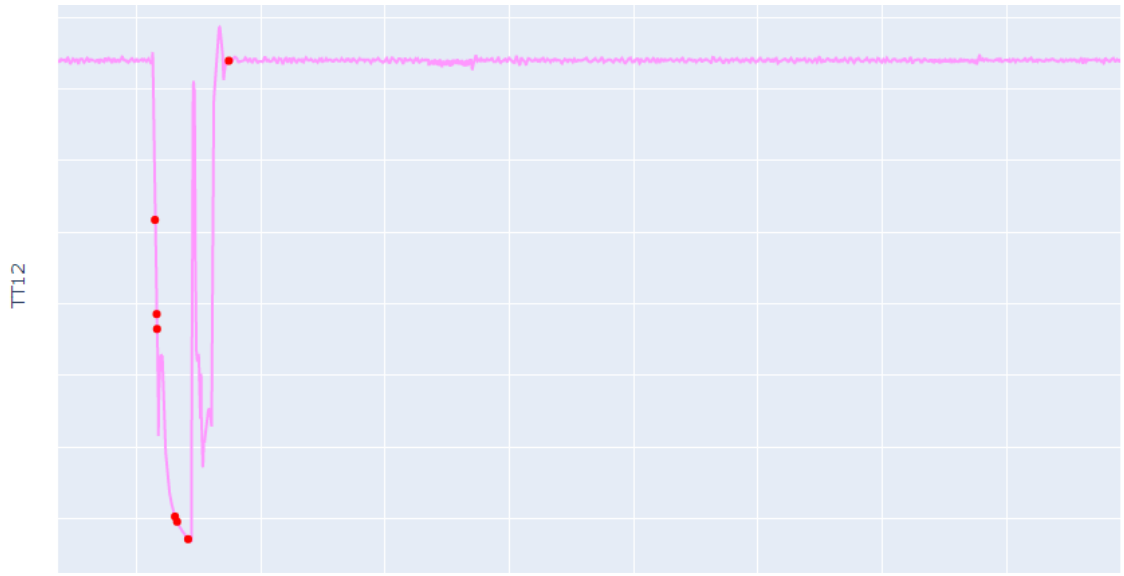


Figure B.25: Clog 6, anomalies detected using LOF and lagged variables, showcased on TT12 sensor data

The following figure presents anomaly index 1 in table B.9, zoomed in for visibility.

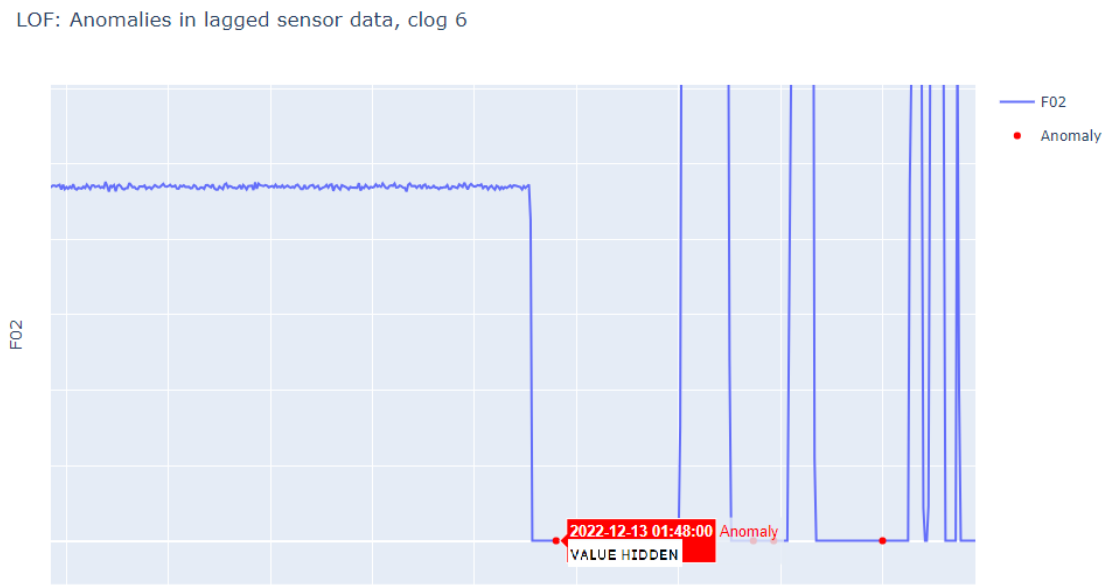


Figure B.26: Clog 6, earliest detected relevant anomaly timestamp in lagged data, showcased on F01 sensor data

B.2 DBSCAN results

B.2.1 Results without lagging of variables

B.2.1.1 Clog 2

For the second clogging using DBSCAN, earliest anomalies are identified right as the event happens. Results are almost identical to that of LOF, excluding the false positive with anomaly index 8 in table B.1. No anomalous behaviour is detected a notable amount prior to the clogging.

Table B.10: DBSCAN anomaly timestamps for clog 2, occurring at 08:40

Anomaly index	Timestamp
4	2021-02-01 08:37:56
5	2021-02-01 08:38:06
6	2021-02-01 08:38:46
7	2021-02-01 08:38:56
8	2021-02-01 08:39:06

The following plots showcase the points presented in the table, as well as additional false positives, anomaly indexes 1-3, caused by process startup. Further anomalous points displayed are not relevant, as they are marked during the clogging.

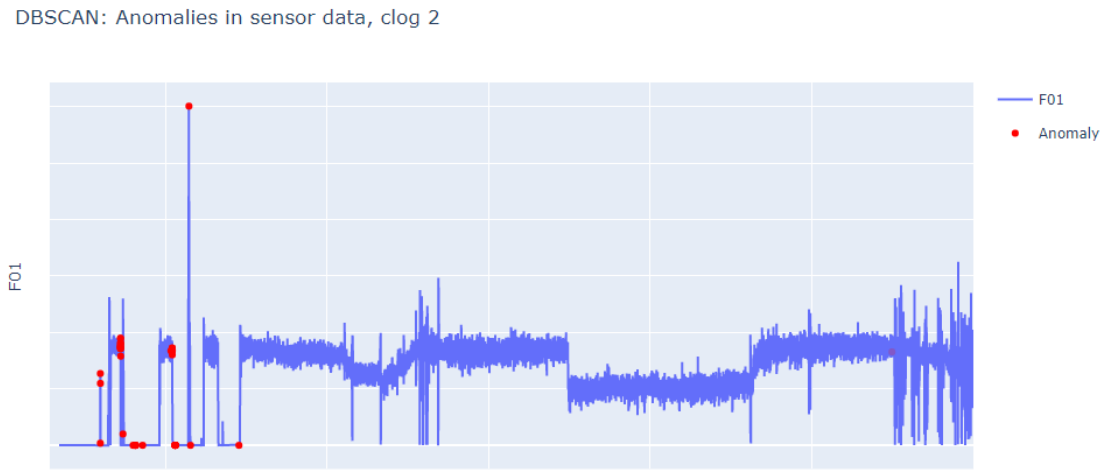


Figure B.27: Clog 2, anomalies detected using DBSCAN, showcased on F01 sensor data

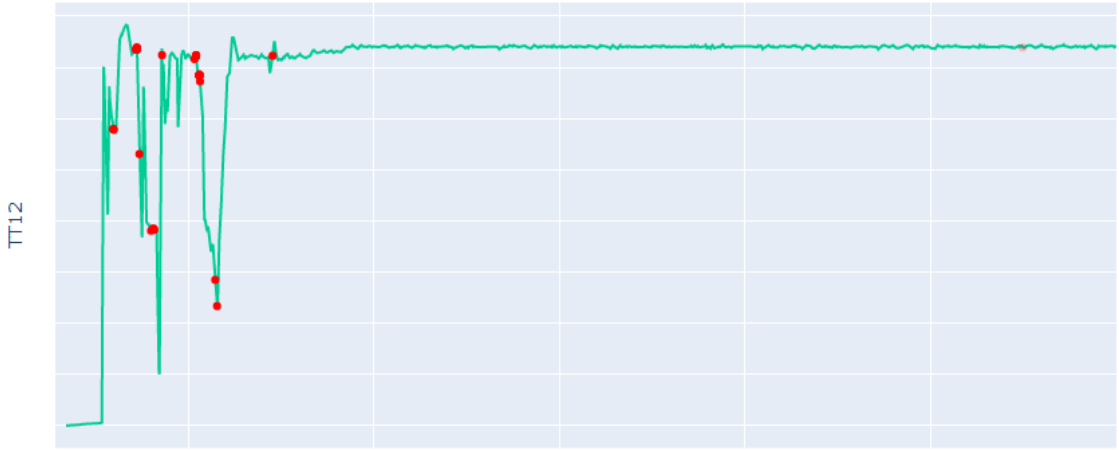


Figure B.28: Clog 2, anomalies detected using DBSCAN, showcased on TT12 sensor data

The following figure presents anomaly index 4 in table B.10, zoomed in for visibility.

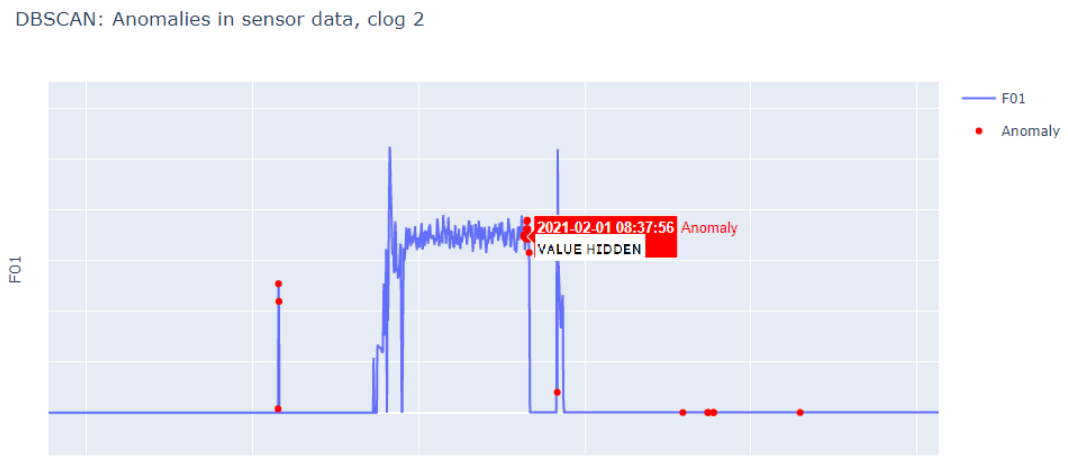


Figure B.29: Clog 2, earliest detected relevant anomaly timestamp, showcased on F01 sensor data

B.2.1.2 Clog 3

For the third clogging, DBSCAN provides identical results to those of LOF for the same event in section B.1.1.2, where earliest anomalies are identified more than 40 minutes prior to the event. Similarly, this is attributed to a manual change in process parameters in that period. Subsequent anomalies are detected as the clogging occurs.

Table B.11: DBSCAN anomaly timestamps for clog 3, occurring at 08:00

Anomaly index	Timestamp
3	2021-02-24 07:15:56
4	2021-02-24 07:16:06
5	2021-02-24 07:24:06
6	2021-02-24 08:02:26
7	2021-01-12 08:02:36

The following plots showcase the points presented in the table, as well as false positives during regular production, anomaly indexes 1 and 2, prior to the clogging event. Additional anomalous points displayed are not relevant, as they are marked during the clogging.

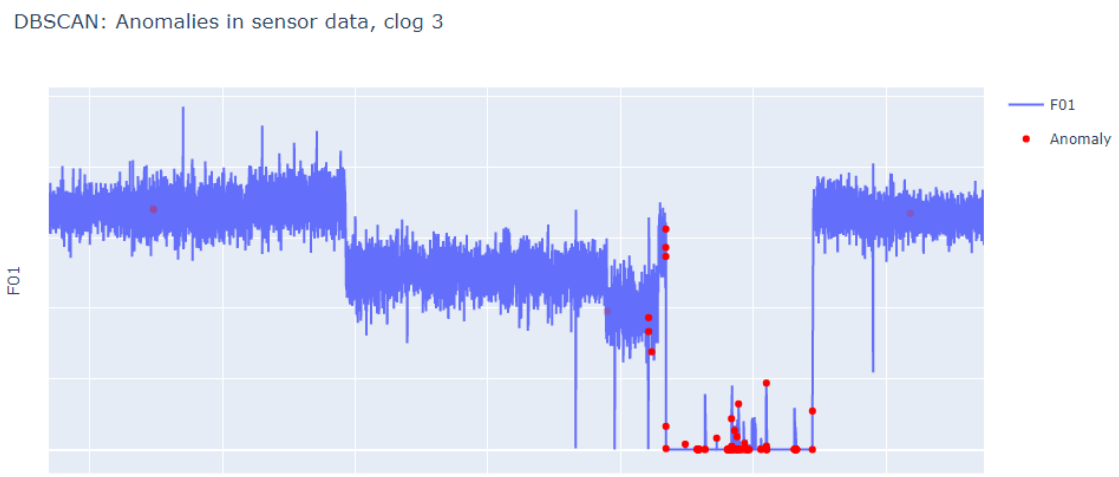


Figure B.30: Clog 3, anomalies detected using DBSCAN, showcased on F01 sensor data

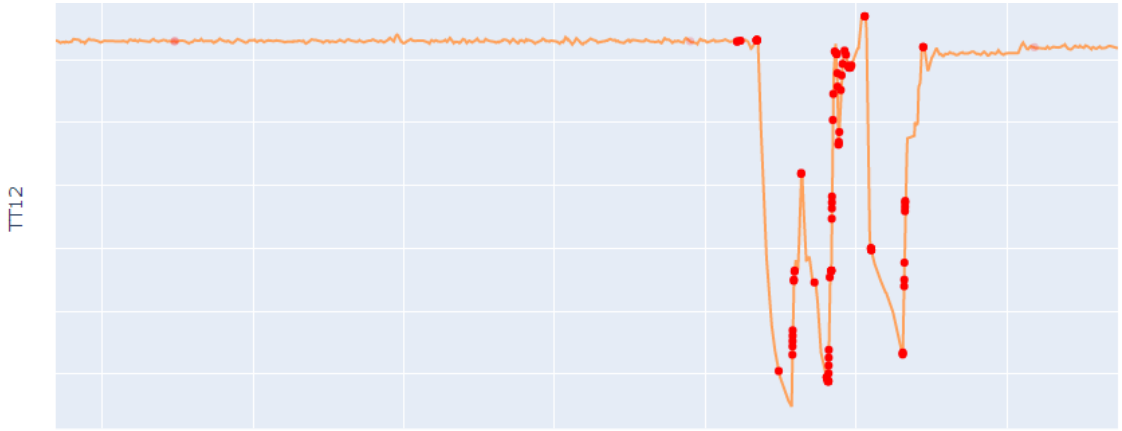


Figure B.31: Clog 3, anomalies detected using DBSCAN, showcased on TT12 sensor data

The following figure presents the earliest relevant timestamps prior to the clogging, anomaly indexes 3 and 5 in table B.11, zoomed in for visibility.

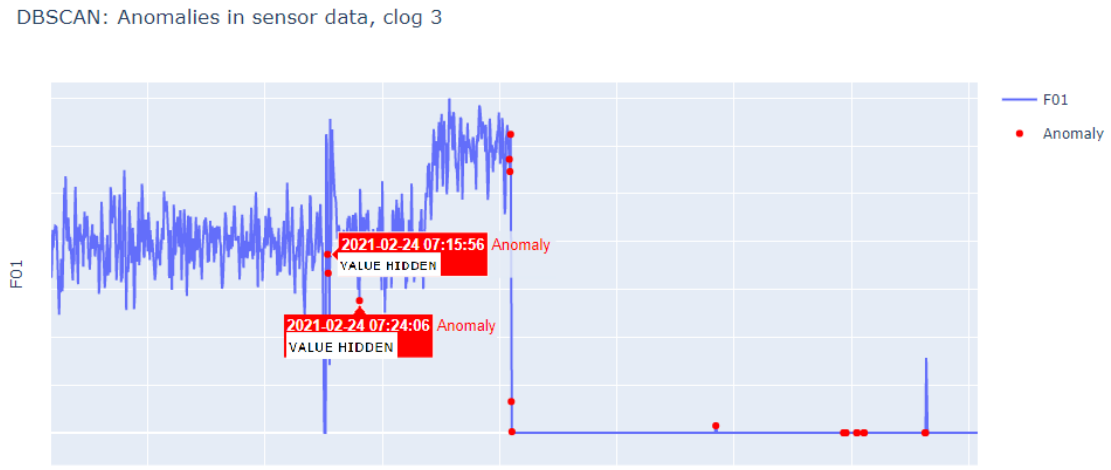


Figure B.32: Clog 3, earliest detected relevant anomaly timestamp, showcased on F01 sensor data

B.2.1.3 Clog 4

No anomalous behaviour is detected prior to the clogging. The anomalous points are flagged too late, right as the clog happens. The following plots showcase this. Opposed to the variation in results using LOF B.1.1.3, no variation in preprocessing provides additional anomalous timestamps.

Table B.12: DBSCAN anomaly timestamps for clog 4, occurring at 17:00

Anomaly index	Timestamp
5	2021-03-01 16:58:24
6	2021-03-01 16:58:34
7	2021-03-01 16:58:44
8	2021-03-01 16:58:54

The following plots showcase the points presented in the table. Additional anomalous points displayed are not relevant, as they are marked during the clogging, starts and ends of the process, and as a result of filtering.

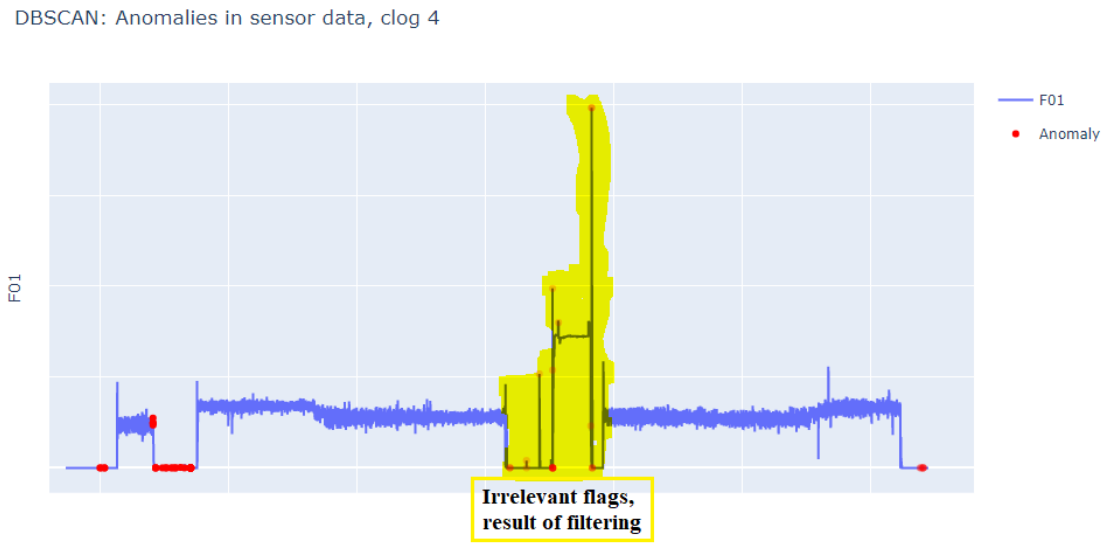


Figure B.33: Clog 4, anomalies detected using DBSCAN, showcased on F01 sensor data

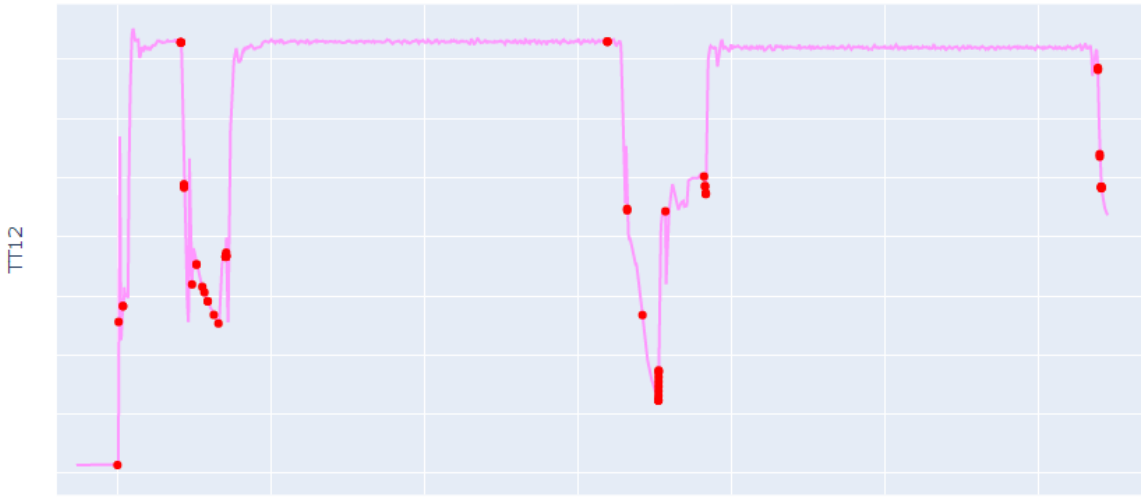


Figure B.34: Clog 4, anomalies detected using DBSCAN, showcased on TT12 sensor data

The following figure presents the earliest relevant timestamp, anomaly index 5 in table B.12, zoomed in for visibility.

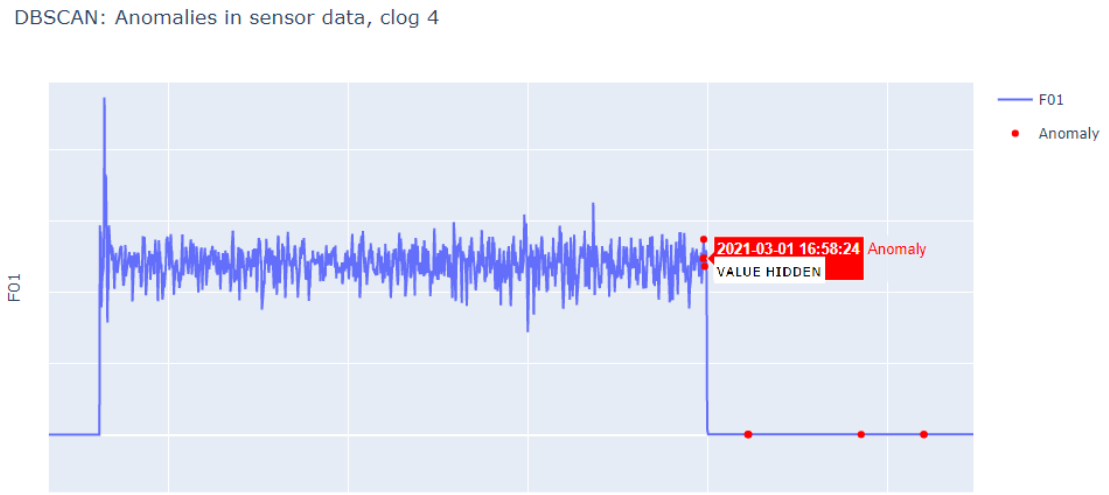


Figure B.35: Clog 4, earliest detected relevant anomaly timestamp, showcased on F01 sensor data

B.2.1.4 Clog 6

DBSCAN gives identical results to utilising LOF on the same data in section B.1.1.4. No anomalous behaviour is detected a notable amount prior to the clogging. No false positives are produced.

Table B.13: DBSCAN anomaly timestamps for clog 6, occurring at 01:35

Anomaly index	Timestamp
1	2022-12-13 01:33:00
2	2021-12-13 01:34:00

The following plots showcase the points presented in the table. Additional anomalous points displayed are not relevant, as they are marked during the clogging.

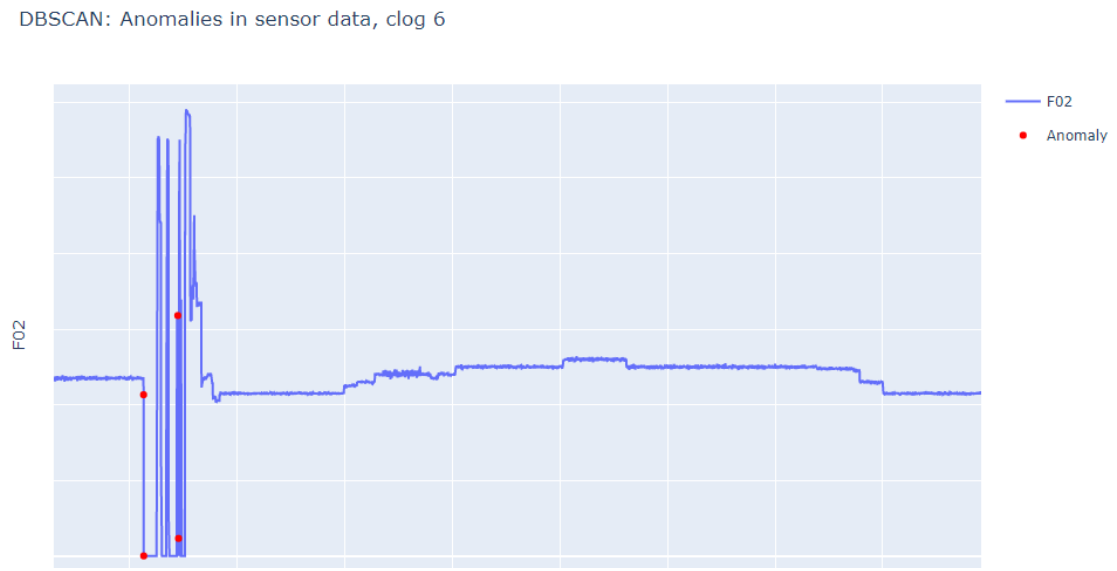


Figure B.36: Clog 6, anomalies detected using DBSCAN, showcased on F01 sensor data

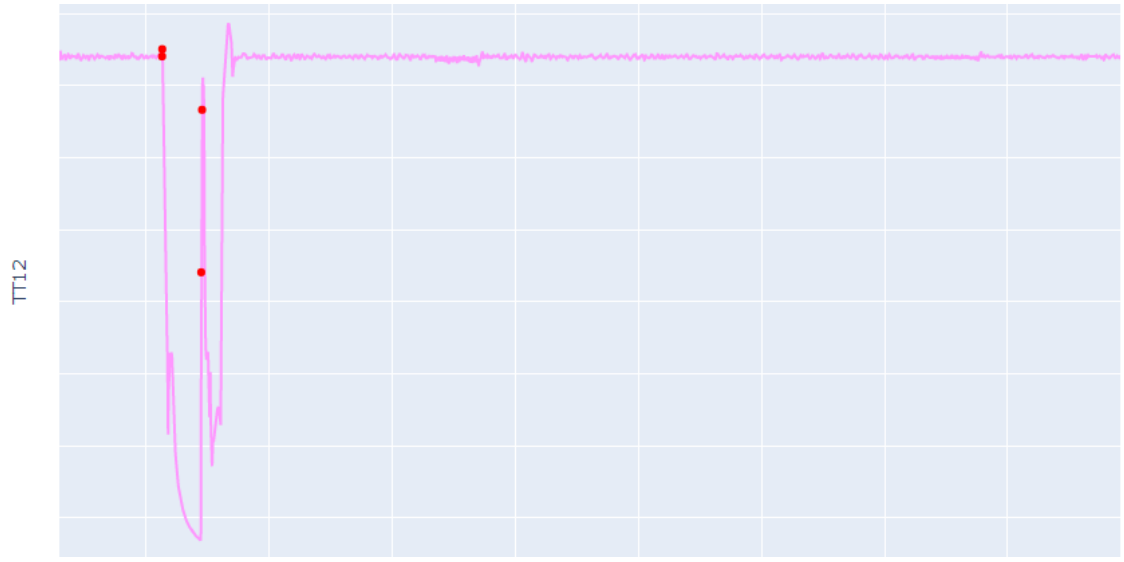


Figure B.37: Clog 6, anomalies detected using DBSCAN, showcased on TT12 sensor data

The following figure presents the earliest timestamp detected, anomaly index 1 in table B.13, zoomed in for visibility.

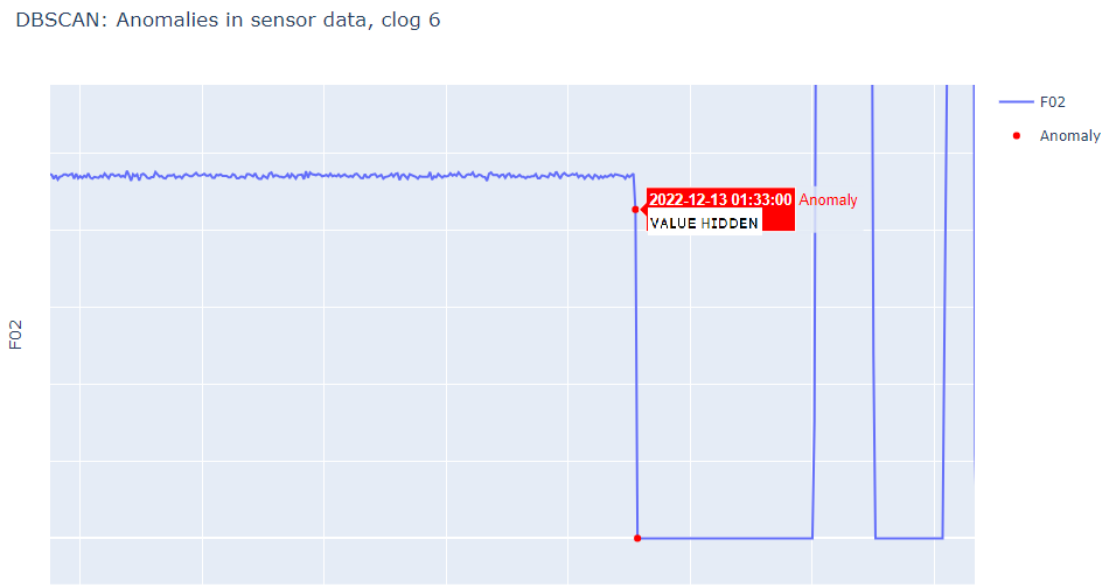


Figure B.38: Clog 6, earliest detected relevant anomaly timestamp, showcased on F01 sensor data

B.2.2 DBSCAN results from using lagged variables

B.2.2.1 Lagged variables and clog 2

Using lagged variables for clog 2 does not provide optimal results. No anomalous behaviour is detected prior to the clogging. No false positives are produced. The earliest detected timestamps during the clogging are showcased in the following table.

Table B.14: DBSCAN anomaly timestamps for clog 2 using lagged variables, occurring at 08:40

Anomaly index	Timestamp
1	2021-02-01 08:53:46
2	2021-02-01 08:53:56
3	2021-02-01 08:54:06
4	2021-02-01 08:54:16

The following plots showcase the points presented in the table. Additional anomalies displayed are marked during the clogging event.

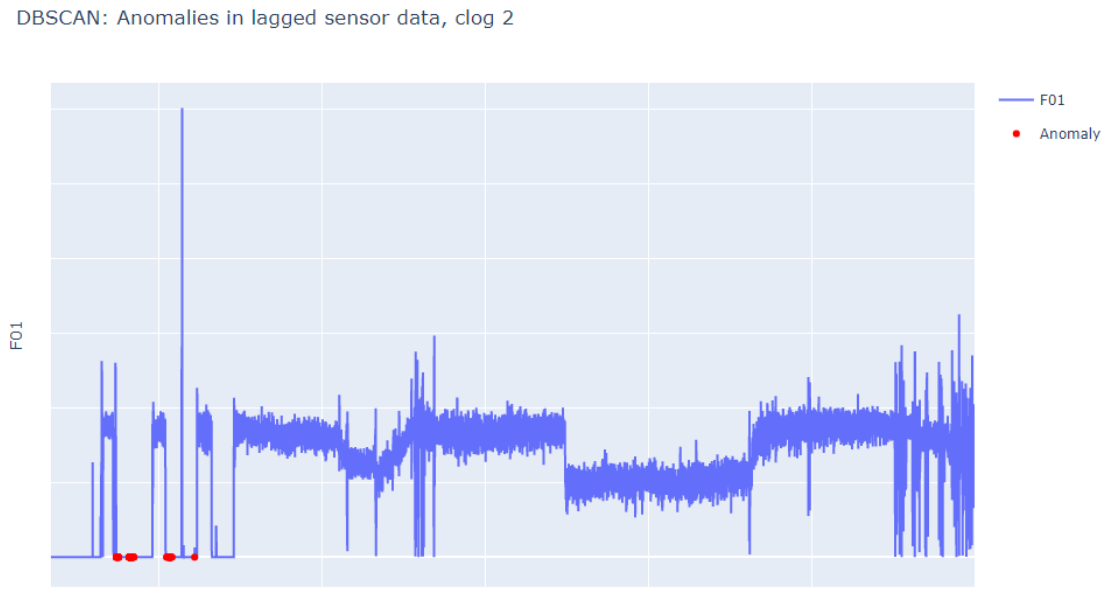


Figure B.39: Clog 2, anomalies detected using DBSCAN and lagged variables, showcased on F01 sensor data

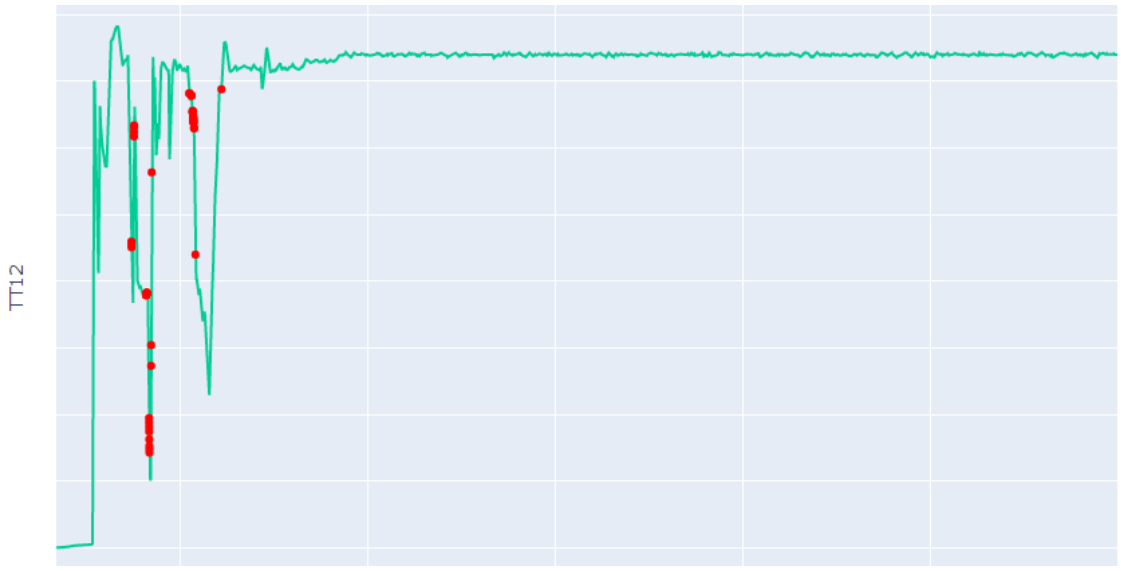


Figure B.40: Clog 2, anomalies detected using DBSCAN and lagged variables, showcased on TT12 sensor data

The following figure presents anomaly index 1 in table B.14, zoomed in for visibility.

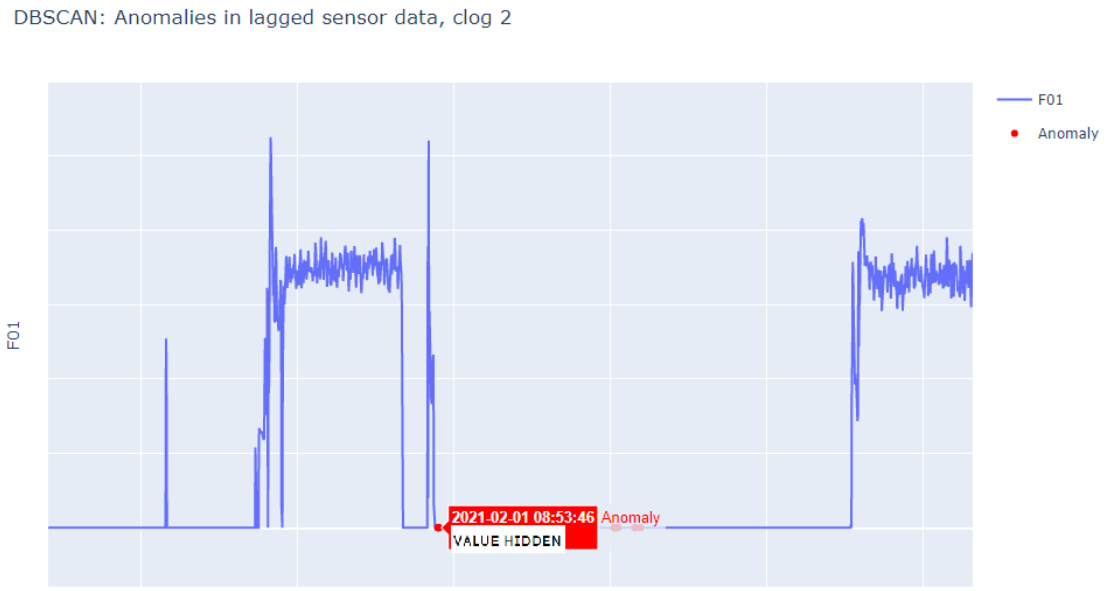


Figure B.41: Clog 2, earliest detected relevant anomaly timestamp in lagged data, showcased on F01 sensor data

B.2.2.2 Lagged variables and clog 3

For clog 3, using lagged variables does not provide any relevant results. Only the clogging event is detected by the model, as showcased in the following table.

Table B.15: DBSCAN anomaly timestamps for clog 3 using lagged variables, occurring at 08:00

Anomaly index	Timestamp
2	2021-02-24 08:09:56
3	2021-02-24 08:10:06
4	2021-02-24 08:10:16
5	2021-02-24 08:10:56

The following plots showcase the points presented in the table. Additional anomalies displayed are marked during the clogging. A false positive is also present during normal operation.

Figure B.42: Clog 3, anomalies detected using DBSCAN and lagged variables, showcased on F01 sensor data

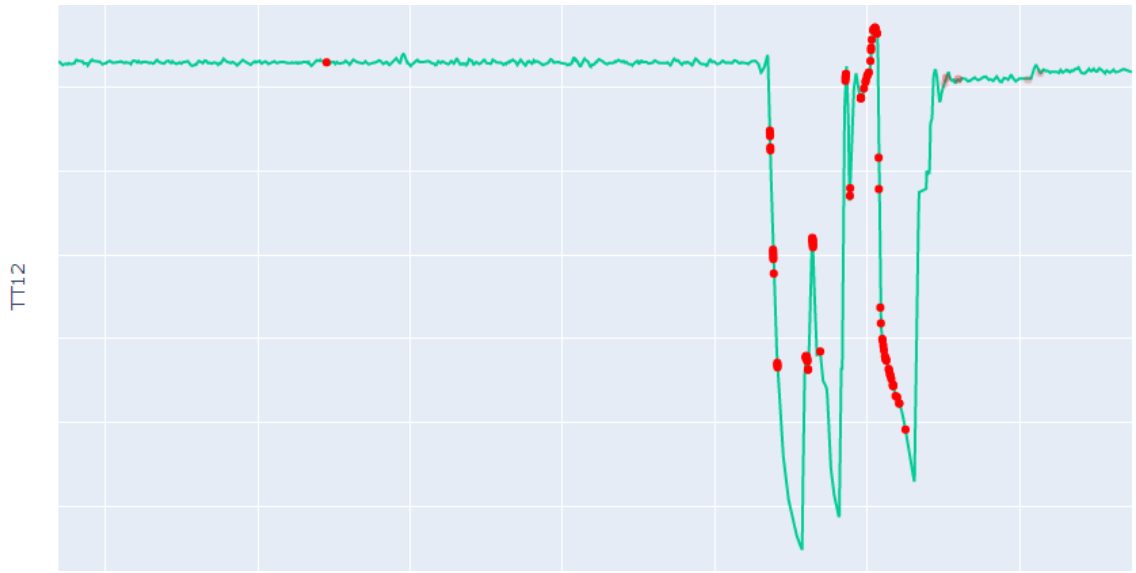


Figure B.43: Clog 3, anomalies detected using DBSCAN and lagged variables, showcased on TT12 sensor data

The following figure presents anomaly index 2 in table B.15, zoomed in for visibility.

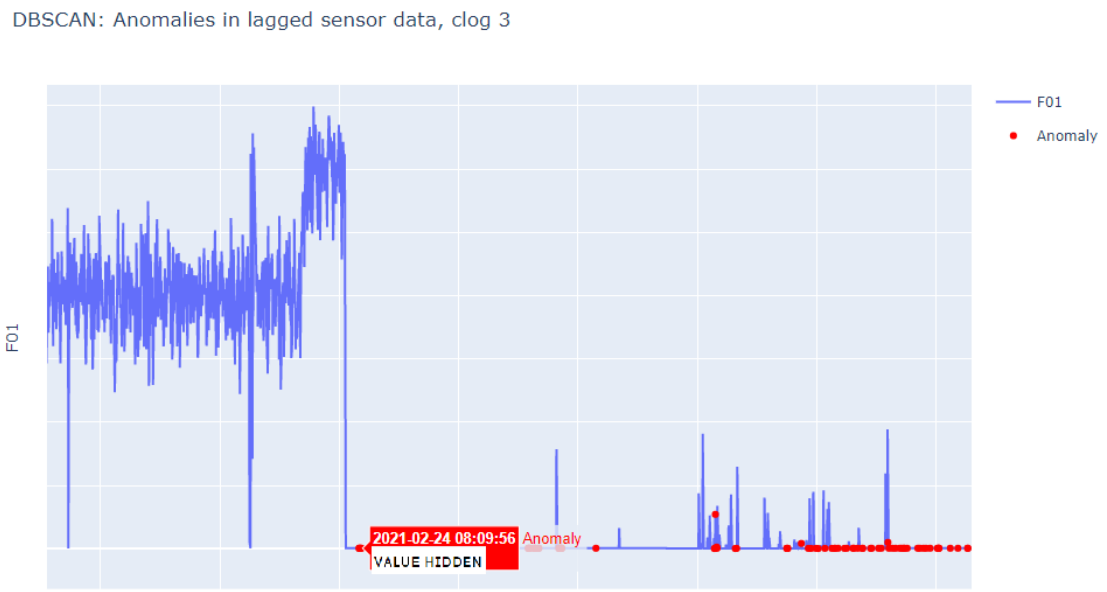


Figure B.44: Clog 3, earliest detected relevant anomaly timestamp in lagged data, showcased on F01 sensor data

B.2.2.3 Lagged variables and clog 4

Using lagged variables for clog 4 similarly does not produce optimal results. No anomalous behaviour is detected prior to the clogging. The earliest anomalous instances are detected late into the event. The results are almost identical to LOF for the same data in section B.1.2.3, excluding false positives during regular operation.

Table B.16: DBSCAN anomaly timestamps for clog 4 using lagged variables, occurring at 17:00

Anomaly index	Timestamp
1	2021-03-01 17:23:24
2	2021-03-01 17:23:34
3	2021-03-01 17:23:44

The following plots showcase the points presented in the table. Additional anomalies displayed are marked during the clogging.

Figure B.45: Clog 4, anomalies detected using DBSCAN and lagged variables, showcased on F01 sensor data

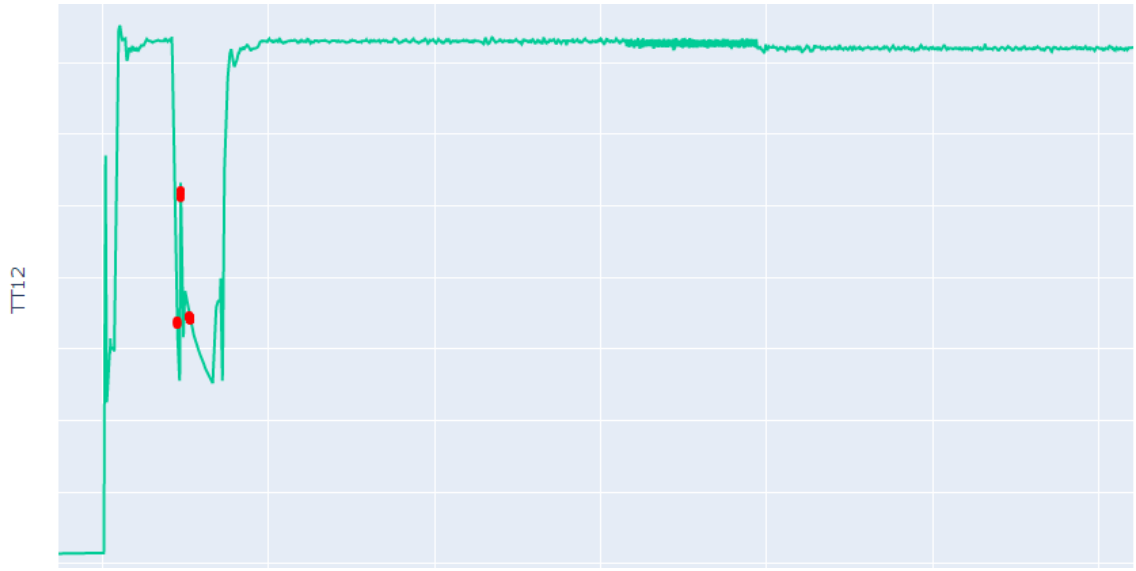


Figure B.46: Clog 4, anomalies detected using DBSCAN and lagged variables, showcased on TT12 sensor data

The following figure presents anomaly index 1 in table B.16, zoomed in for visibility.

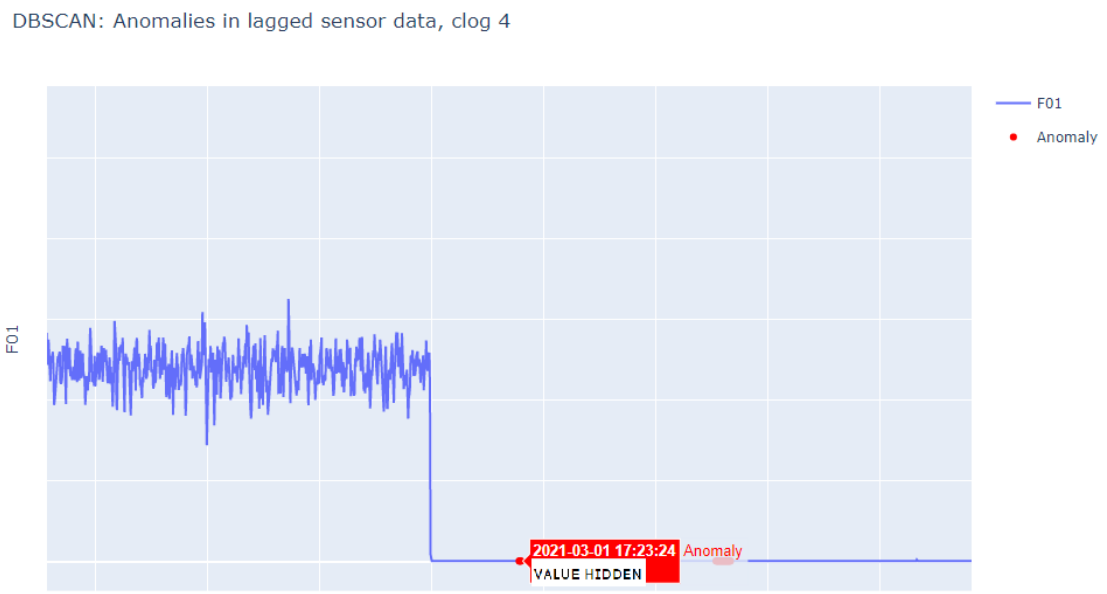


Figure B.47: Clog 4, earliest detected relevant anomaly timestamp in lagged data, showcased on F01 sensor data

B.2.2.4 Lagged variables and clog 6

The model is unsuccessful in identifying anomalous behaviour prior to clog 6 using lagged variables. The earliest identified anomaly timestamps are presented in the following table.

Table B.17: DBSCAN anomaly timestamps for clog 6 using lagged variables, occurring at 01:35

Anomaly index	Timestamp
1	2022-12-13 01:48:00
2	2022-12-13 01:49:00

The following plots showcase the points presented in the table. Additional anomalies displayed are marked during the clogging.

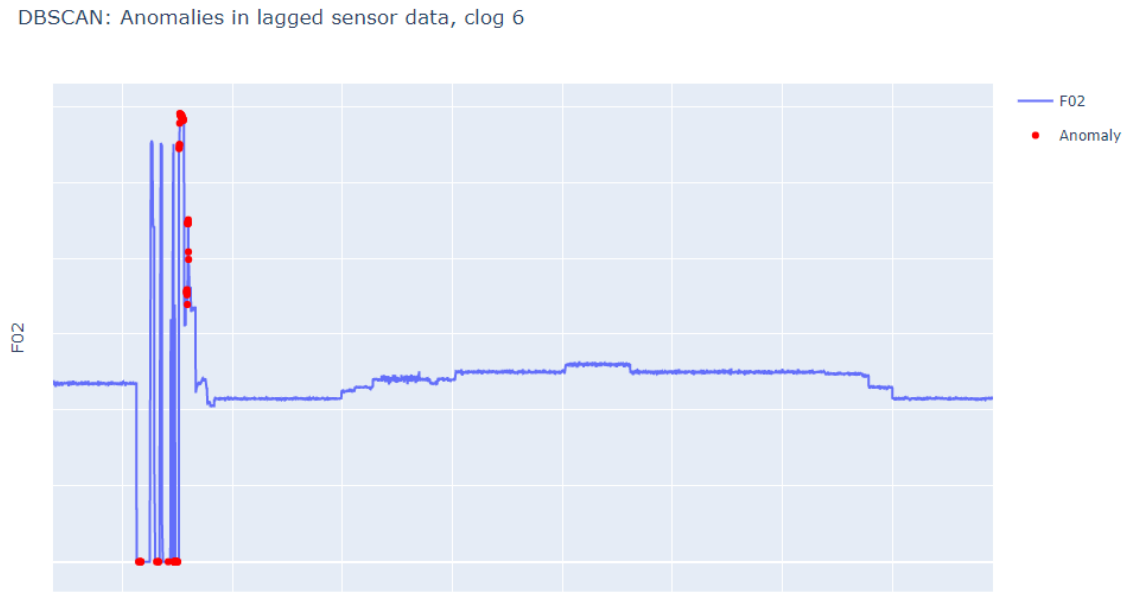


Figure B.48: Clog 6, anomalies detected using DBSCAN and lagged variables, showcased on F01 sensor data

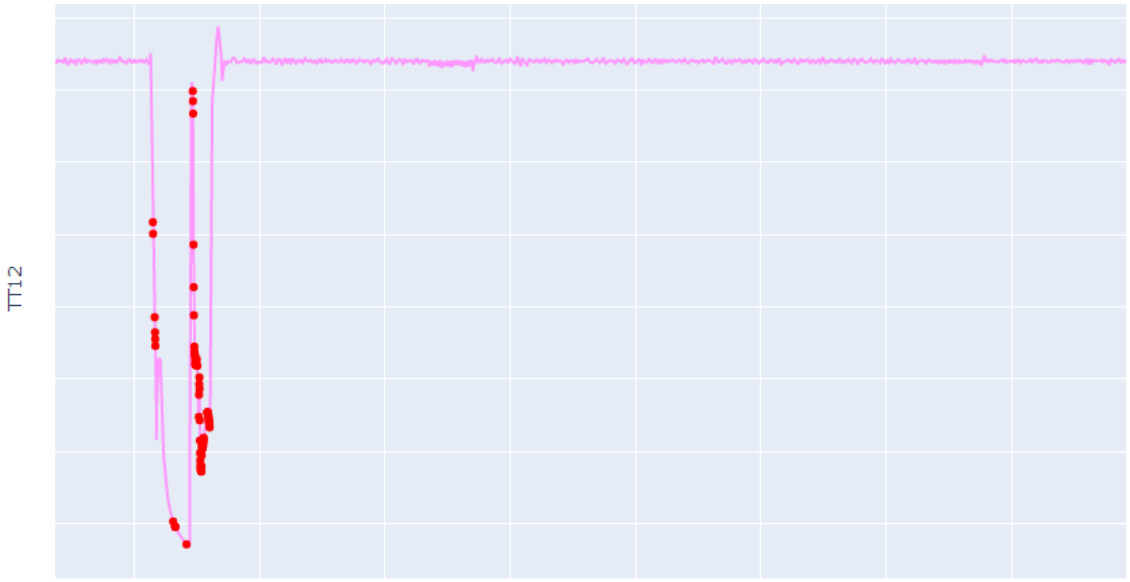


Figure B.49: Clog 6, anomalies detected using DBSCAN and lagged variables, showcased on TT12 sensor data

The following figure presents anomaly index 2 in table B.17, zoomed in for visibility.

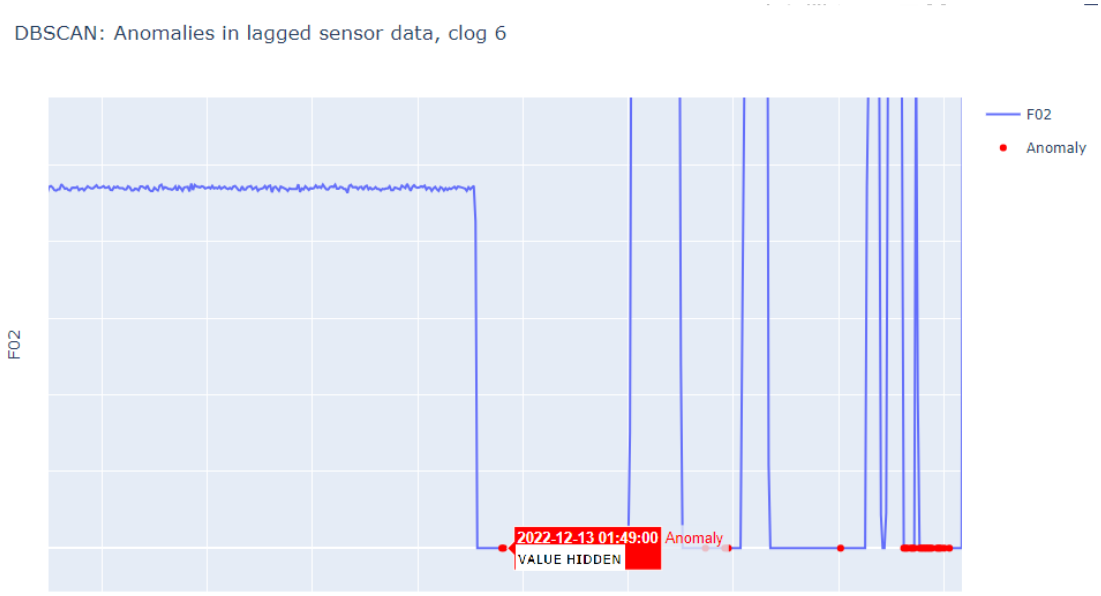
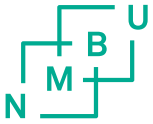


Figure B.50: Clog 6, earliest detected relevant anomaly timestamp in lagged data, showcased on F01 sensor data



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway