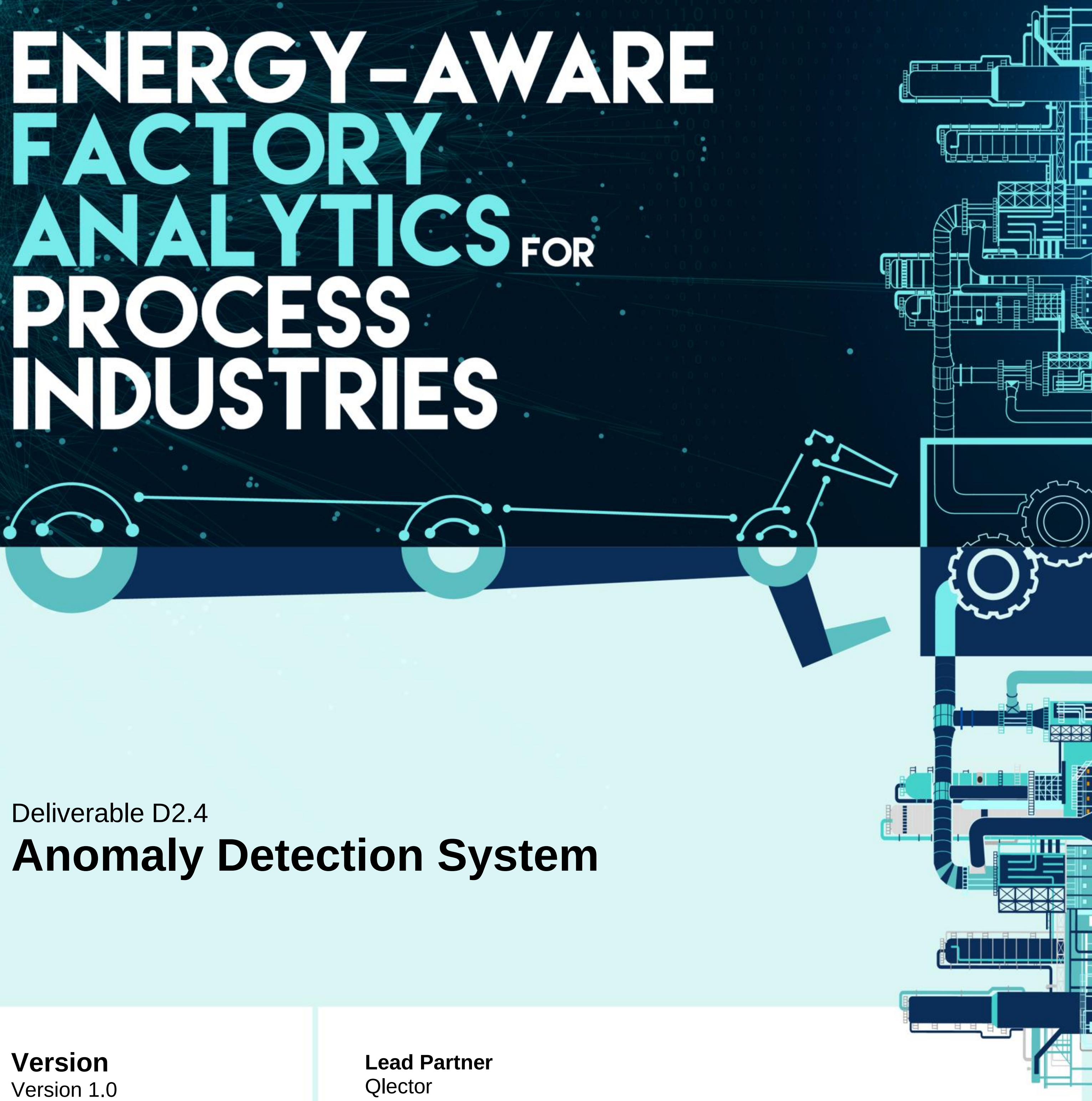
FACTLOG
www.factlog.eu

ENERGY-AWARE FACTORY ANALYTICS FOR PROCESS INDUSTRIES



Deliverable D2.4

Anomaly Detection System

Version Version 1.0	**Lead Partner** Qlector
Date 30/06/2021	**Project Name** FACTLOG – Energy-aware Factory Analytics for Process Industries

FACTLOG has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Nº 869951

Call Identifier H2020-NMBP-SPIRE-2019	**Topic** DT-SPIRE-06-2019 - Digital technologies for improved performance in cognitive production plants
Project Reference 869951	**Start date** November 1st, 2019
Type of Action IA – Innovation Action	**Duration** 42 Months

Dissemination Level

X	**PU**	Public
	CO	Confidential, restricted under conditions set out in the Grant Agreement
	CI	Classified, information as referred in the Commission Decision 2001/844/EC

Disclaimer

Executive Summary

This document presents the anomaly detection system of the FACTLOG project. In the FACTLOG architecture anomaly detection methods play the role of the eyes and ears. They monitor the operation of the manufacturing environments and raise alerts when unexpected situations occur. Other components such as process modelling and optimisation are then invoked to take over and find appropriate actions to address whatever may have happened.

The anomaly detection methods must be broad and flexible enough to be able to detect a wide variety of anomalies and offer ways for calibration, so they detect all the critical issues while not flooding the system with spurious and meaningless alerts. Section 2 describes the selection of methods chosen for use. This includes a wide selection univariate and multivariate algorithms, from the simpler approaches based on running averages and thresholds, through detection using machine learning models such as the Isolation Forest, to deep-learning based methods such as the Generative Adversarial Networks (GANs). For complex event processing, where more meaningful events need to be identified, the StreamStory is presented, which builds a hierarchical Markov model of a multivariate system. The model can be used to run Monte-Carlo simulations of system state transitions or inspected manually in the graphical interface to gain insights into the system operation.

In Section 3 deployments of the anomaly detection methods in the FACTLOG use cases is presented. These examples include detecting parameter value drift in the Continental use-case, the unsupervised detection of anomalies in the JEMS use-case, both on the level of individual readings as well as on the level of system-wide events, and finally an example of detecting organisational anomalies through the QlectorLEAP application.

The implementation details of the anomaly detection module developed during FACTLOG are presented in Section 4. The section covers some of the methodological details of the implemented methods, the specification of their APIs, and the technical guidelines needed for their deployment.

Revision History

Revision	Date	Description	Organisation
0.1	03/03/2021	Table of contents	QLECTOR
0.2	05/03/2021	Initial inputs	QLECTOR
0.3	29/04/2021	Anomaly detection description.	JSI, QLECTOR
0.4	10/06/2021	Methodology section.	QLECTOR
0.5	18/06/2021	Implementation section	JSI, QLECTOR
0.6	26/06/2021	Final draft ready for internal review.	QLECTOR
0.7	29/06/2021	Peer review	NISSA
1.0	30/06/2021	Final version ready for submission	QLECTOR

Contributors

Organisation	Author	E-Mail
QLEC	Klemen Kenda	klemen.kenda@qlector.com
JSI	Aljaž Košmerlj	aljaz.kosmerlj@ijs.si
QLEC	Viktor Jovanoski	viktor.jovanoski@qlector.com
QLEC	Mario Karlovčec	mario.karlovcec@qlector.com
QLEC	Blaž Fortuna	blaz.fortuna@qlector.com
JSI	Matic Erznožnik	matic.erznoznik@ijs.si
JSI	Gal Petkovšek	gal.petkovsek@ijs.si
JEMS	Johannes Krmc	johannes@jems.eco

Table of Contents

List of Figures

1 Introduction

1.1 Purpose and Scope

This deliverable presents the anomaly detection system of the FACTLOG project. Anomaly detection is a significant part of the analytics component and handles monitoring of the manufacturing systems and raising alerts when unexpected and/or unwanted situations happen. In a sense, this system plays the role of the eyes and ears of the cognitive digital twin. Once an anomaly is detected, it can be displayed to a human operator for inspection or other components of the cognitive digital twin can be invoked to handle the situation (e.g. optimisation or process modelling).

The document explains the theoretical background of the methods used – from the simpler statistics-based approaches to the more complex machine learning methods. The use of the methods is demonstrated in examples from the use-cases where some of them are already in deployment. Though more work is needed to reach full operation, so far, the results are promising.

The technical details of the implemented anomaly detection module are also presented. This includes the API specification and the deployment technical details.

1.2 Relation with other Deliverables

This deliverable is strongly coupled with FACTLOG D2.3 Holistic Model of Uncertainty and Causal Relations [1]. Some of the baseline models differ between anomaly detection and approaches described in D2.3, however – when trying to extract certain more complex events based on uncertainty, approaches from D2.3 also play a crucial role.

Some of the examples in Section 3 lean on use-case deployment information contained in FACTLOG deliverable D7.1 Installation and Initial Testing [2]. The key information is repeated in this document, but all the details are available there.

1.3 Structure of the Document

After the introduction in Section 1, the methodology is explained in Section 2, covering univariate methods, multivariate methods, as well as Complex Event Processing using multi-level modelling with the StreamStory tool. Section 3 contains examples of use of the anomaly detection functionality in different FACTLOG use-cases. Following that, Section 4 explains the technical details of the implementation of the anomaly detection module. Finally, the conclusions are drawn in Section 5.

2 Methodology

Anomaly detection is a subfield of artificial intelligence with many possible definitions, methodologies and goals. Quite often anomalies are defined as outliers, previously unseen or rarely seen events. On the other hand, in practice quite often we want to identify the states that are not unknown but are undesirable. Methodologically, approaches may differ. We have explored some of the possible methods that solve concrete problems, identified in the manufacturing and process industries. These events might not be anomalies in the traditional sense and therefore we attack them differently.

We have explored traditional uni- and multi-variate anomaly detection approaches and tested its usability on real world use cases. The approaches improve the current state of the use cases, but do not necessarily improve state of the art in the field.

More added value can be extracted from the data when not only single-point anomaly detectors are used but with a combination of a knowledge graph and with methodologies of modelling uncertainty through the knowledge graph (as described in FACTLOG deliverable D2.3 [1]). With the knowledge graph, many of the useful insights can be extracted out of very simple models and this is where the strength of FACTLOG approach is demonstrated in practice.

2.1 Univariate Anomaly Detection

In time series data, anomaly detection becomes a slightly better-defined task and can be defined as "an outlier data point which does not follow the collective common pattern of the majority of the data points and hence can be easily separated or distinguished from the rest of the data" [3]. In practice, anomalies can refer to single events, multiple events (changing offset) and changing trends.

Examples of anomalies are depicted in Figure 1. In the first subfigure, (a), we can observe a time series without any anomalies. The measurements are depicted in blue, the predicted values in orange and an admissible interval in light blue. Kalman filter (which models prediction in the first phase and uncertainty in the second phase of the algorithm) can produce such nice characteristics, which can be used for detecting out-of-order data pointes. Kalman filter is classified as **predictive confidence level approach**. Instead of a generic Kalman filter any similar approach can be used such as ARIMA, SARIMA, GARCH, VAR or any other regressive model from the field of machine learning. Standard machine learning models, however, might not be able to model variance in a such an elegant way as Kalman filter does, but they might implicitly acquire more underlying knowledge about the modelled process and would therefore yield much more accurate results.

The second subfigure, (b), depicts single spikes, which might be a consequence of bad measurements, temporal sensor malfunction or some other data-transfer related interference. Also, potentially misclassified anomalous event is depicted in this subfigure. Similarly, 3 possible anomalous events can be observed in the next subfigure, (c), where an expert user should check, whether these are true anomalies or not. Finally, the last subfigure, (d), depicts multiple obvious anomalies that emerged due to system malfunction. Anomalies that happen due to slow change of target value distributions are depicted in the next subsection, in Figure 11. Kalman filter, for example, cannot detect such anomalies but statistical profiling approaches can.

In Continental use-case, for example, each process has defined upper and lower borders of a particular process measurement. Their MES system is able to address this issue and can issue alarms, when a particular measurement falls outside the allowed interval. These static approaches are easy to understand, robust and easy to implement. However, this set-up is not useful for predictive maintenance, as a single anomaly does not really talk much about the state of a particular machine/tool. Quite often, the machine produces OK parts up until it fails. And there are no (or only a few) anomalies. When indicates the potential failure is the slow deterioration of measured parameters (that still stay within the allowed range).

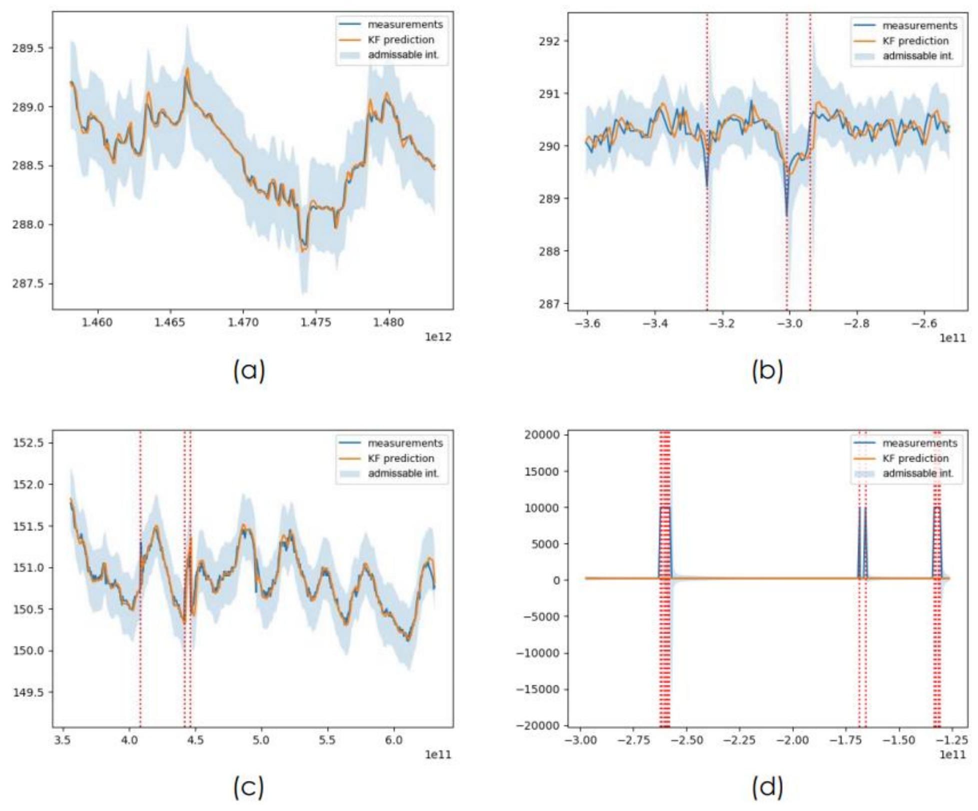


Figure 1: Types of anomalies that occur in univariate time series [4]. (a) a time series without any anomalies, (b) single spikes, (c) possible false positives, (d) multiple obvious anomalies

For these, often **statistical profiling approaches** are appropriate. Their best feature is that they do not need extensive setup in the beginning. For example, there is no need to set up the upper and lower boundaries of the system.

One typical example would be histogram anomaly detector, which yields an alert when a measurement falls outside the 99-percentile limit, for example. The detector would need some grace period in which baseline distribution of the target values would be acquired and from then on it would function automatically, alerting the user whenever something non-typical would happen.

In the case of predictive maintenance use-case the intuition was also to handle the noise and therefore prior to histogram anomaly detector we performed an EMA (exponential moving average) smoothing on the signal.

Anomaly detection is an unsupervised machine learning tasks and as such, clustering would be a good fit for solving anomaly detection problems. In fact, a family of such algorithms composes the **clustering-based unsupervised approach**. Whenever a data point, which does not belong to the local cluster, is detected (for example by using rolling window based DBSCAN [3]) it can be marked as an anomaly.

Changes of trends over time is sometimes referred to as concept drift detection. We indicated, that **concept-drift unsupervised approach** can be used for detecting changes of target value distributions over time. Potential methods include, but are not limited to, DDM, EDDM, PHT, STEPD, DoF and ADWIN. Our early experiments, however, have indicated poor quality of results (unstable) from these methods.

2.2 Multivariate Anomaly Detection

Among methods that support multivariate data, Isolation Forest [5] shows the most promising results, especially in a streaming scenario [6]. Isolation forest builds upon two premises: that anomalies in the dataset are only few and that they are very different from the normal instances. Such instances are easier to isolate. The algorithm builds an ensemble of “Isolation” trees (which are technically equivalent to Half-Space trees) for the dataset. It finds anomaly according to the shorter average path length in the trees as depicted in Figure 3. Anomalous path is depicted in red, non-anomalous in blue.

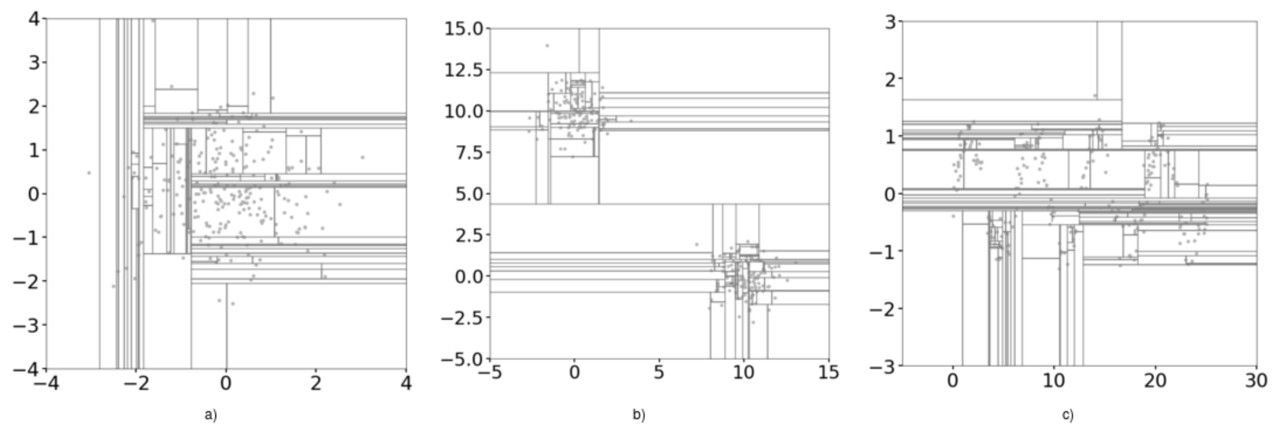


Figure 2: Splitting of data space with an “Isolation” tree [7].

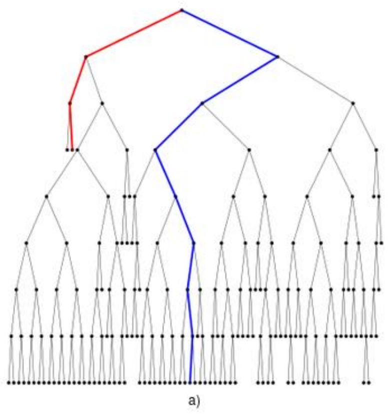


Figure 3: Shorter path to anomalous point than to non-anomalous [7].

Inconsistencies of Isolation Forest (with the “anomaly score”) are solved in Extended Isolation Forest algorithm [7]. The latter uses a branching process, that can occur in every direction.

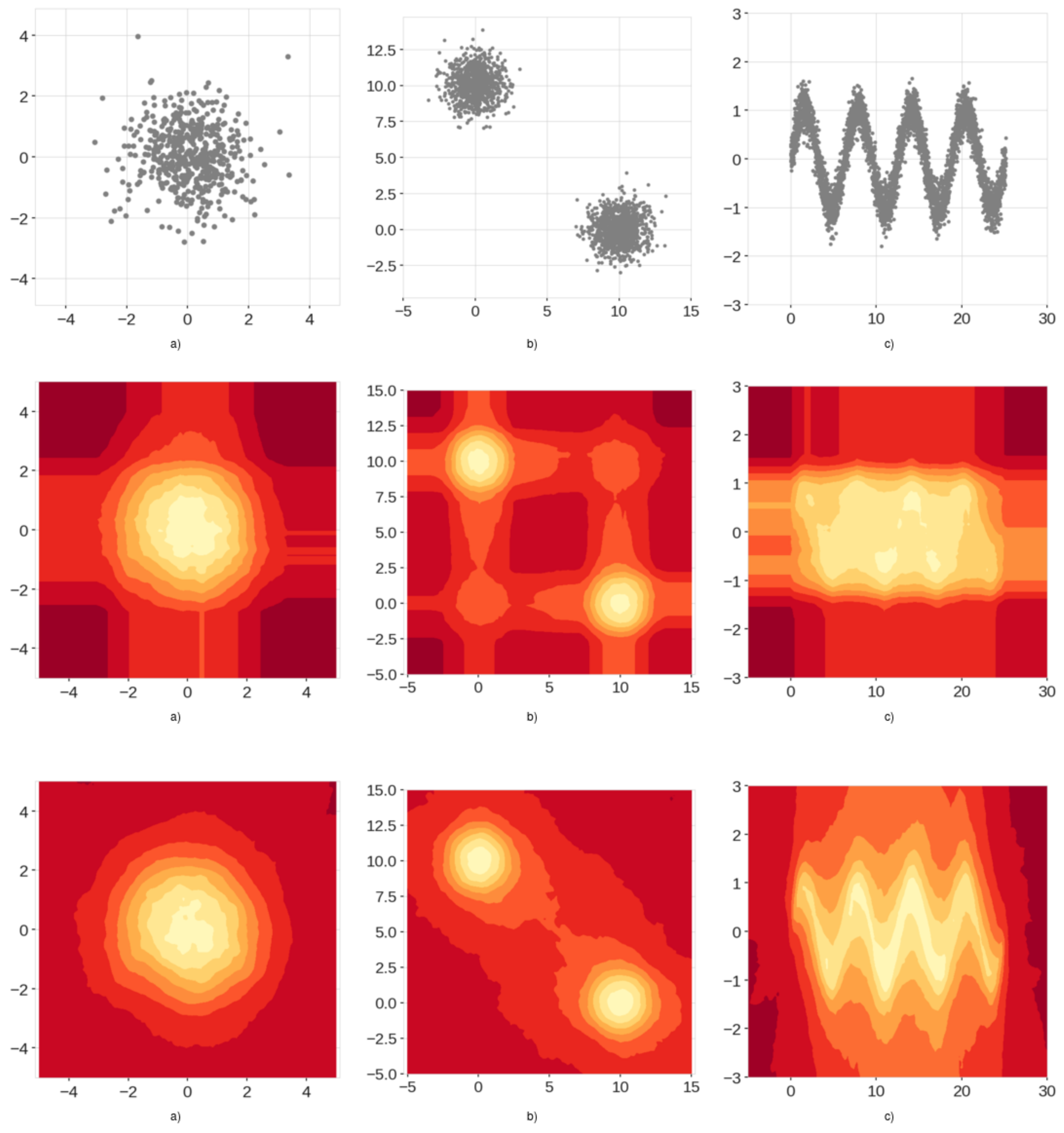


Figure 4: Dataset (top), Isolation forest scores (middle) and Extended isolation forest scores (bottom) [7].

Figure 4 depicts differences between anomaly score maps generated by Isolation Forest and Extended Isolation Forest. Artefacts of the horizontal/vertical splitting are clearly visible in the middle row.

The GAN (Generative Adversarial Networks) mechanism can also be used for anomaly detection purposes. GAN networks are based on the encoder – decoder principle. The neural network is comprised of two parts – the encoder and decoder. The general architecture is depicted in Figure 5.

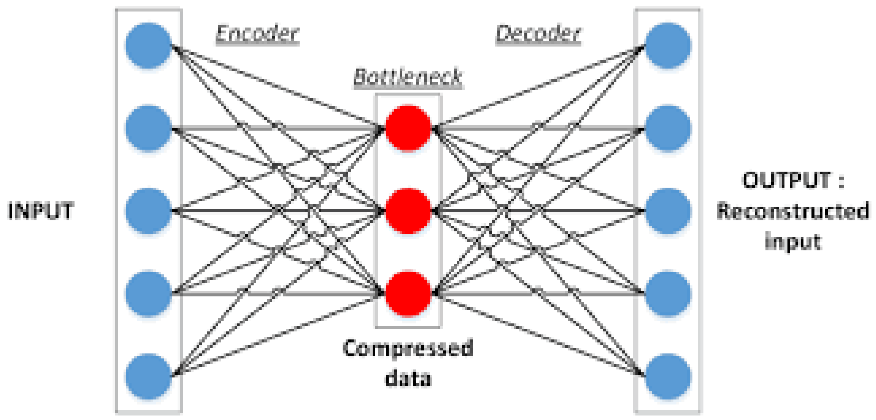


Figure 5: GAN network architecture.

The idea of GAN is to reduce the dimensionality of the input data, in order to extract the most important features and characteristics of the input vector via the encoder. The decoder part of the network then attempts to reconstruct the initial input based on the compressed data in the bottleneck. We can think of the input as the raw data and the information in the bottleneck layer as the 'logic' of the input. If the 'logic' can accurately describe the entire input, then the reconstructed output can be very similar to the input. In anomaly detection, we observe the reconstruction error – the difference between the input and output. The network is trained on a dataset, which is considered to describe the normal behaviour of the observed system. It learns to reproduce the inputs very closely, by learning the 'logic' of the system. After it is trained, it is applied to the real data on which we perform anomaly detection. If the input is abnormal, the bottleneck will not be able to extract the key characteristics correctly, which will result in a skewed reconstruction by the decoder. The difference between the input and output will thus be large, which will indicate an anomaly.

Promising results in data stream scenarios have been observed by making the input of the GAN a feature vector which consists of N consecutive values of a single variable in the stream.

2.3 Complex Event Processing and Risk Prediction through Multi-level State Graph Analysis

Complex event processing (CEP) observes data streams at a higher level than methods in previous sections. The goal of CEP is to process the data streams (from possibly multiple sources) and identify meaningful events in them – for example, a machine malfunction, a warm-up routine, loading materials for batch processing, end-of-day shutdown and similar. These events and the relations between them offer insights into the observed process and help understand what is going on. By understanding the significance of events, we can identify which represent risks and then form strategies to avoid them.

The tool we use for the purposes described above is StreamStory[1] [8]. StreamStory was developed at JSI as part of previous work on stream data mining. It performs an analysis of

[1] http://streamstory.ijs.si/

a multivariate time series and builds a hierarchical Markov chain model of its qualitative behaviour. This model is a multi-level state graph that captures the typical states if the system and the transitions between them at different levels of detail. An example of a model is shown in Figure 6. The pictured model is for a simple case of two variables, rainfall and temperature, measured over the course of several years and shows an expected yearly cycle.

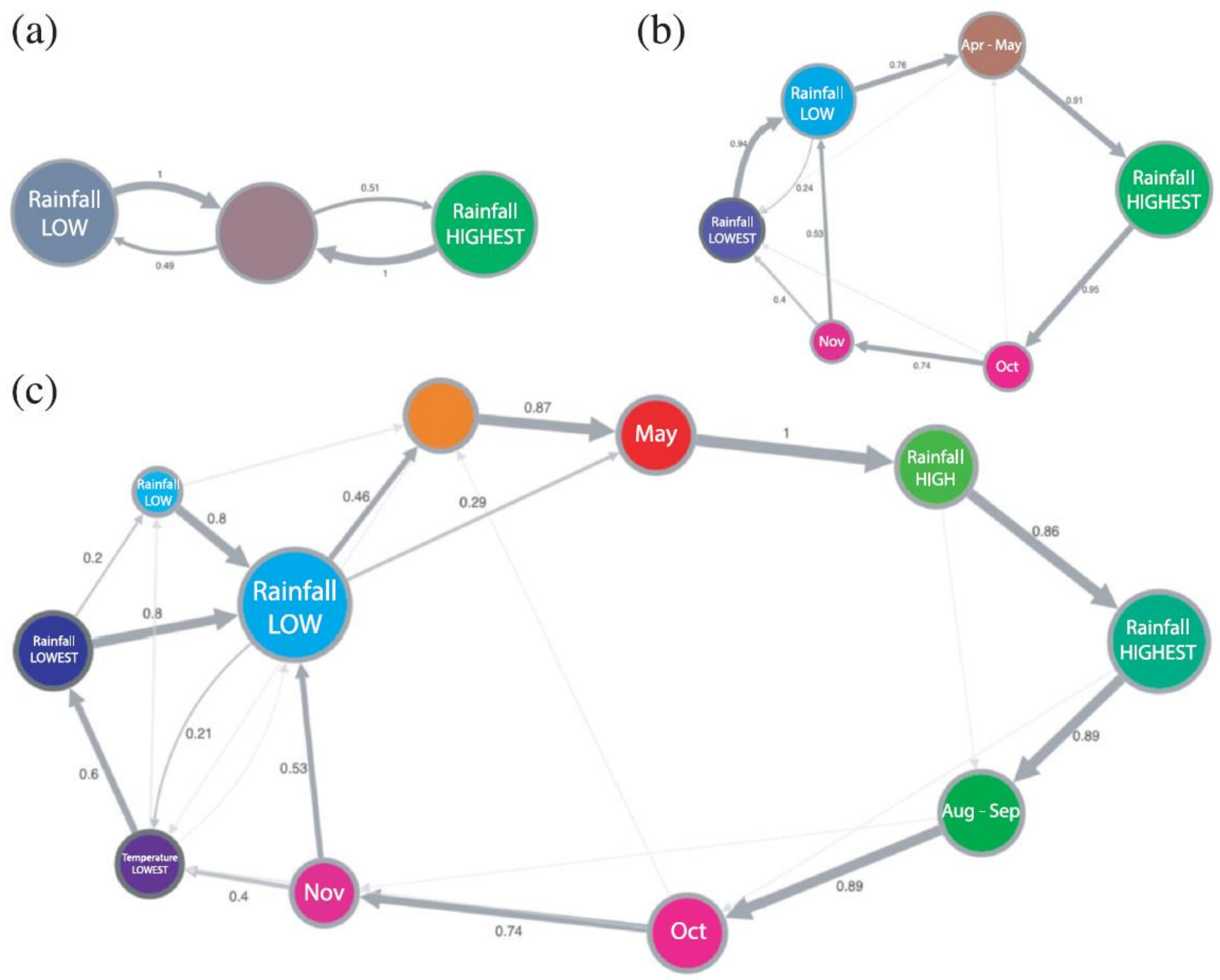


Figure 6: An example of a model different-levels of resolution. The model captures the dynamics of the yearly rainfall and temperature cycles at (a) low, (b) medium and (c) high levels of detail.

The StreamStory methodology works in several steps illustrated in Figure 7:

a) First the time-series values are transformed into a point cloud with each time series representing a different dimension of the points. The points have a temporal ordering as they correspond to the series.
b) The point space is then partitioned using a clustering algorithm. Currently, the K-Means and DP-Means algorithms are supported. The clusters obtained represent the system states (i.e. model nodes) at the highest level of detail.
c) The transitions between the states are modelled by aggregating the trajectories of the time series between them. By computing the probabilities of transitions from the trajectory frequencies from the data a Markov model is built.

d) Finally, the hierarchy is constructed by aggregating the most related states and recomputing the transitions. The relatedness of the states can be measured based on the distances between states using mean-linkage agglomerative clustering or by splitting the transition Markov chain in a top-down manner by solving the min-cut problem. The result is a model of the process at several levels of aggregation (i.e. detail).

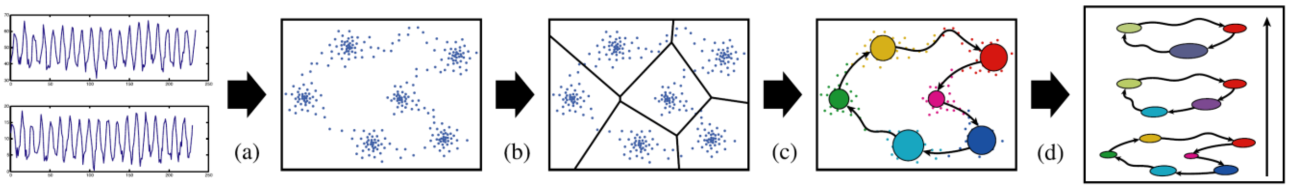
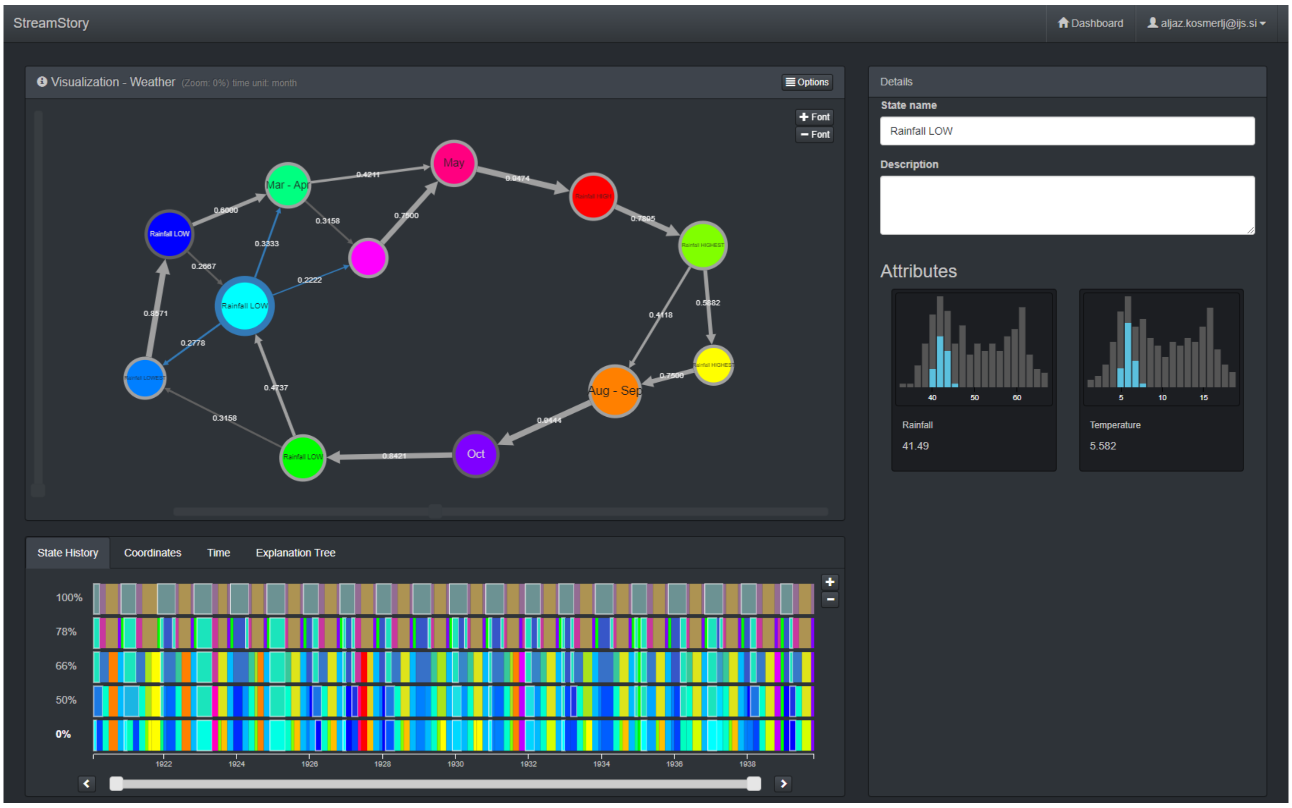


Figure 7: The steps of the StreamStory methodology: (a) constructing a point cloud to represent the multivariate time series, (b) constructing the states by partitioning the ambient space, (c) modelling the transitions between the states and (d) aggregating the states and transitions into a hierarchy.

The StreamStory interface (shown in Figure 8) supports interactive inspection of the model at all the different resolutions and the properties of the states, such as the distributions of the variables' values (shown in the histograms on the right). This allows domain experts to interpret the states, define their semantics and identify those that represent potential risks. StreamStory also supports automatic state labelling by highlighting the most prominent features in the states. Using the transition probabilities Monte-Carlo simulations can be run of future system dynamics which allow us to estimate the likelihood of entering a "risk" state or for some meaningful event to occur. Though currently unsupported from the interface, the transitions could also be modelled by building a classifier of the next state using a supervised learning approach that uses the time-series values as features (e.g. SVM, RNN, LSTM).



State name: Rainfall LOW

Description:

Rainfall 41.49

Temperature 5.582

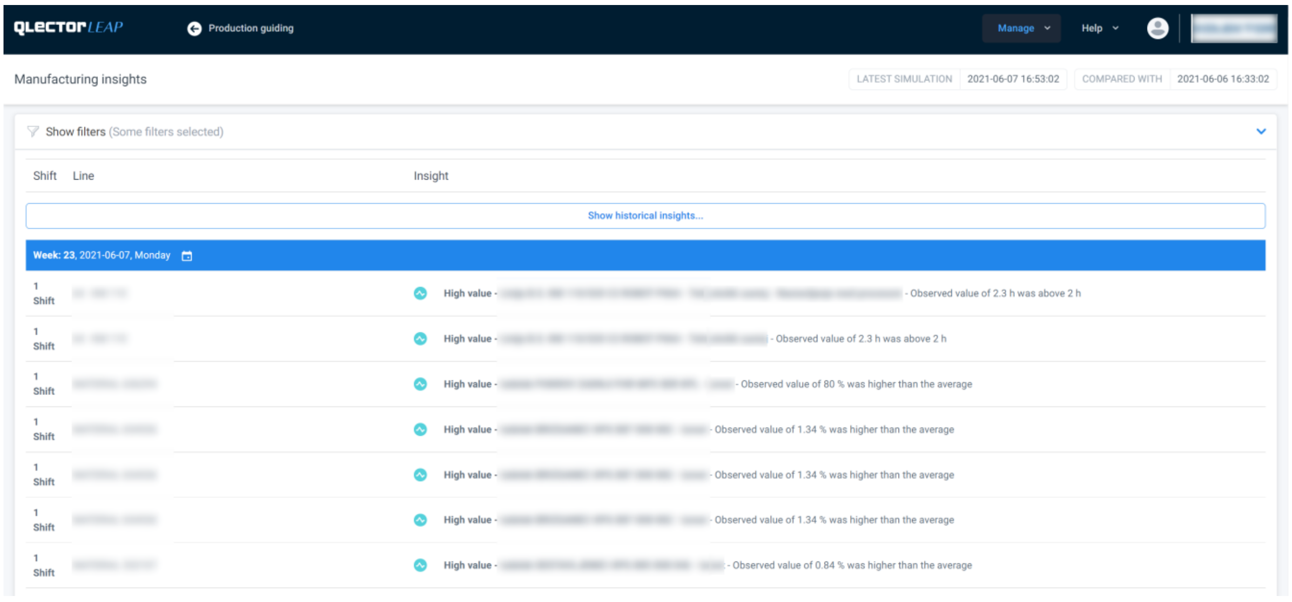
Figure 8: The StreamStory user interface.

3 Scenarios

3.1 Simple Anomalies

Implemented system has a potential to detect anomalies on streaming data. The component that is described in this deliverable, offers plenty of advanced functionalities, some of them have also been deployed to production.

Below in Figure 9 and Figure 10 we present screenshots from QlectorLEAP, where we demonstrate generation of insights that inform the shopfloor manager. These are the detection of unusually high and unusually low values of particular data streams. Mostly, these anomalies are based on the histogram approach and only report the events on a univariate stream, which fall into a pre-defined category of unlikely events (based on a threshold).


QLECTOR LEAP
Production guiding
Manage
Help
[redacted]
Show filters (Some filters selected)
Show historical insights...

Manufacturing insights

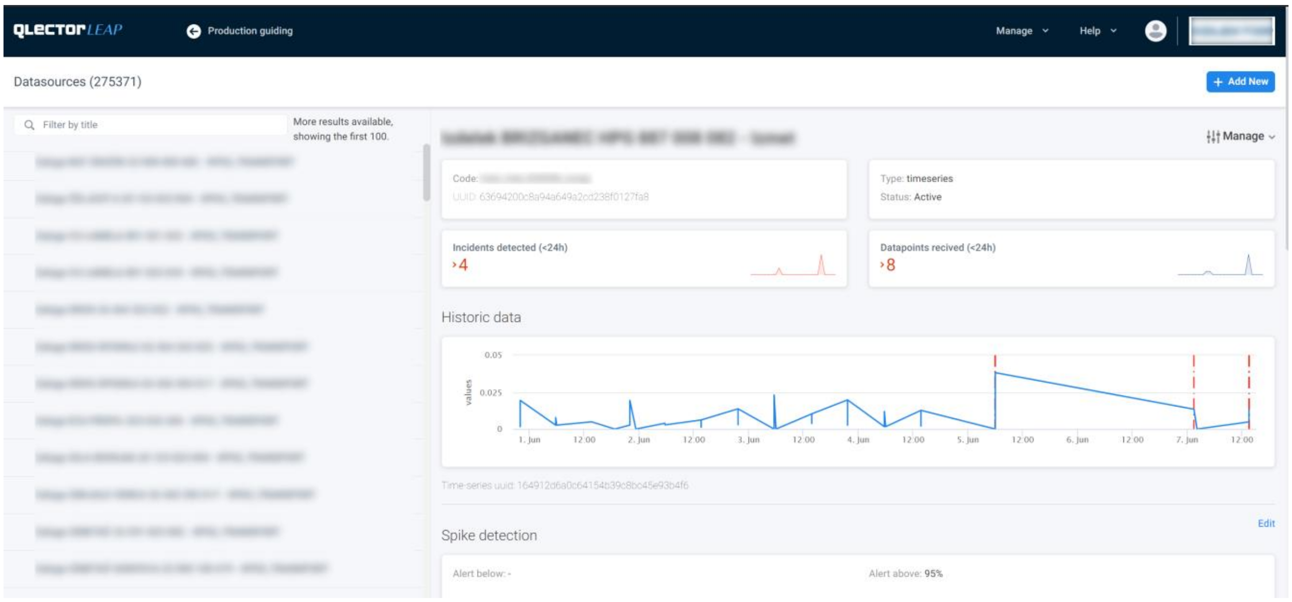
LATEST SIMULATION 2021-06-07 16:53:02 COMPARED WITH 2021-06-06 16:33:02

Shift	Line	Insight
Week: 23, 2021-06-07, Monday		
1 Shift	[redacted]	**High value** - [redacted] - Observed value of 2.3 h was above 2 h
1 Shift	[redacted]	**High value** - [redacted] - Observed value of 2.3 h was above 2 h
1 Shift	[redacted]	**High value** - [redacted] - Observed value of 80 % was higher than the average
1 Shift	[redacted]	**High value** - [redacted] - Observed value of 1.34 % was higher than the average
1 Shift	[redacted]	**High value** - [redacted] - Observed value of 1.34 % was higher than the average
1 Shift	[redacted]	**High value** - [redacted] - Observed value of 1.34 % was higher than the average
1 Shift	[redacted]	**High value** - [redacted] - Observed value of 0.84 % was higher than the average

Figure 9: List of insights based on adaptive anomaly detectors in QlectorLEAP.

Each anomaly is reported in a rule-based NLP generation module among various insights that are reported and can serve as an additional datasource when trying to debug the production flow.

By clicking the particular insight, the following screen in Figure 10 is offered to the user, where it is possible to visualize the time series and detected anomalies.



[redacted]

Datasources (275371)

More results available, showing the first 100.

[redacted]

[redacted]

Code: [redacted]
UUID: 63694200c8a94a649a2cd238f0127fa8

Type: timeseries
Status: Active

Incidents detected (<24h)
›4

Datapoints recived (<24h)
›8

Historic data

Time-series uuid: 164912d6a0c64154b39c8bc45e93b4f6

Spike detection

Alert below: -

Alert above: 95%

Figure 10: Anomaly details in QlectorLEAP.

3.2 Predictive Maintenance

In predictive maintenance the task is to detect the optimal time for machine/tool service. This means that we try to use the machine/tool as long as possible before service/replacement, but to still avoid cases, where malfunction would cause a technical and/or organisational downtime.

Another important aspect of anomaly detection in these scenarios is the interpretability of the results. This means that a detected anomaly should be explained to the end user and that he should be able to comprehend the reasoning of the automated anomaly detection system.

This scenario will be implemented in the Continental use-case. Currently, they use the “border-check” approach on a set of more than 10.000 time series. Border check has limited usability as there are a lot of time series and because per-part anomaly detection can be caused by a normal distribution of values and not some systemic failure. They lack an adaptive approach which would successfully detect the shift in the distribution of values towards one of the extremes. According to a domain expert, that is the best indicator that something is about to go wrong.

3.2.1 Results

We consider three approaches of stream data analysis in order to detect anomalous behaviour, preferably before the measurements surpass the specified tolerance limits. Generally, the idea is to detect the trend shift in the incoming data in order to recognise early, that the data might be approaching the specified upper or lower bound. The real data from Continental was slightly modified in order to better demonstrate the functionality of the proposed methods. Figure 11 illustrates the functionality of Border check on sample data. The yellow dashed lines represent the warning stages, and the black lines are the upper and lower limit. The advantage of this method is that we can receive warnings before measurements start to fall out of the specified range.

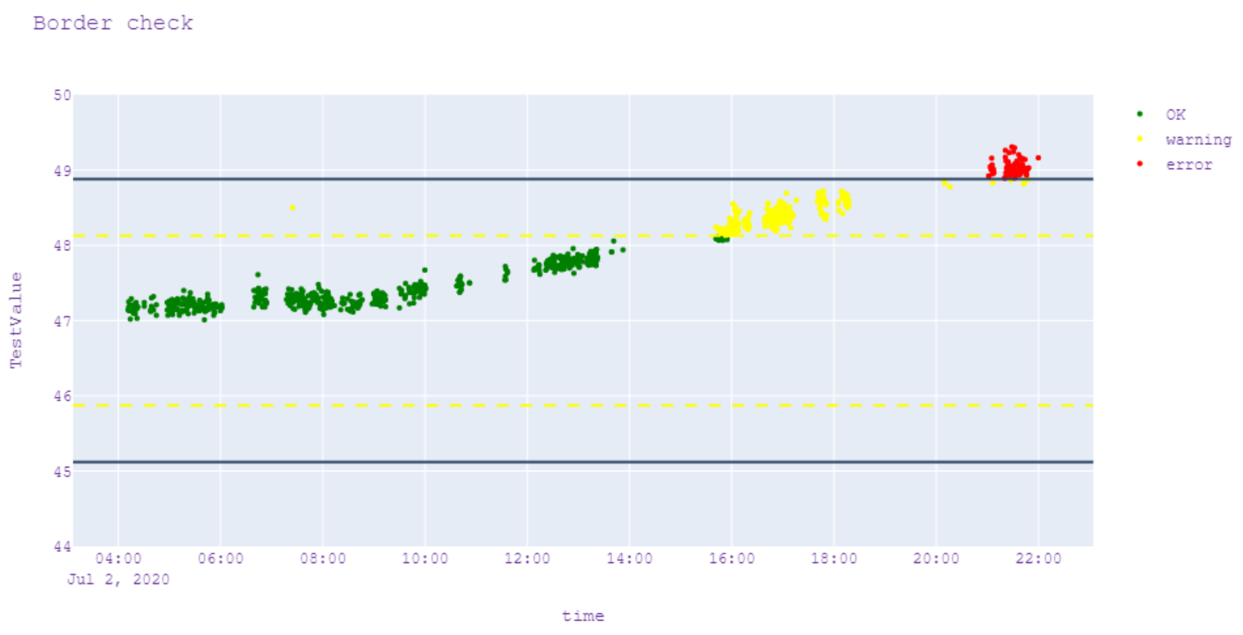


Figure 11: Border Check demonstration

The next approach - Exponential moving average (EMA) - is a running variable, which is the weighted average of previous measurements. It only issues warnings if many anomalous measurements are observed in sequence. The functionality is demonstrated in Figure 12. The light blue line represents the current value of the EMA. When it crosses the warning stage, a warning is issued and when the EMA crosses the upper limit the warnings become errors.

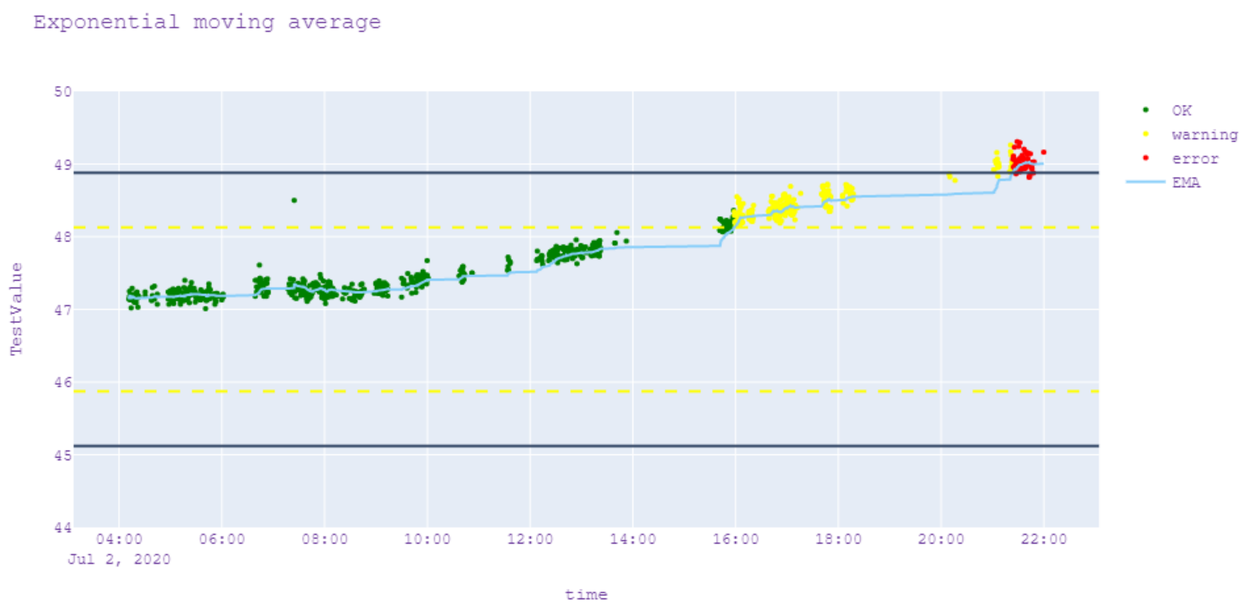


Figure 12: Exponential moving average algorithm

The third approach - Moving Average Convergence Divergence (MACD) - is aimed mostly at detecting consistent trends. In a trending scenario the difference between EMAs with different periods becomes large and stays large as long as a trend is present. Changing the two periods, we can adapt the MACD to the severity of the trend that we would like to detect. In Figure 13, an illustrative example is shown, where at some point a trend begins to emerge. The value of MACD soon becomes large, issuing warnings and Errors. As the data starts to level-off, the MACD comes back down indicating that no trend is present.

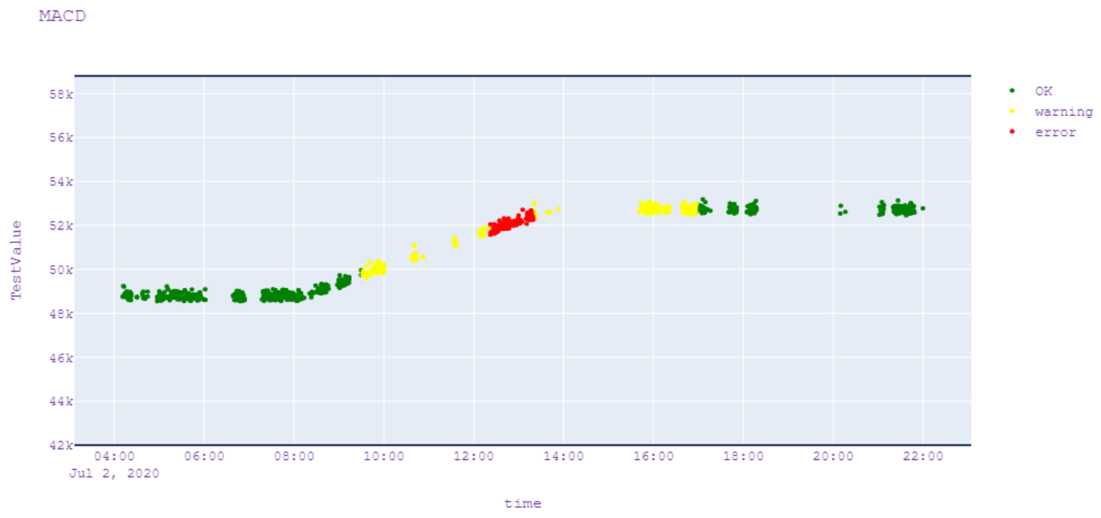


Figure 13: MACD algorithm

In the final algorithm – EMA Percentile, (Figure 14) the value of the running EMA is compared to a window of previous EMA values, to calculate the percentile of the latest EMA value. An error is issued if the percentile is either too high or too low e.g. above 98th or below 2nd. In principle, this algorithm is similar to the above described EMA, with a key difference of the varying threshold. In the pure EMA algorithm, the alarm thresholds are fixed, but in this approach, the threshold adapts to the ‘normal’ of the observed variable, since the e.g. 98th percentile of the latest N values will constantly change. This is advantageous as we do not need to input the threshold by hand. Emerging trends will always be labelled as anomalies, as long as the observed window is not too large.

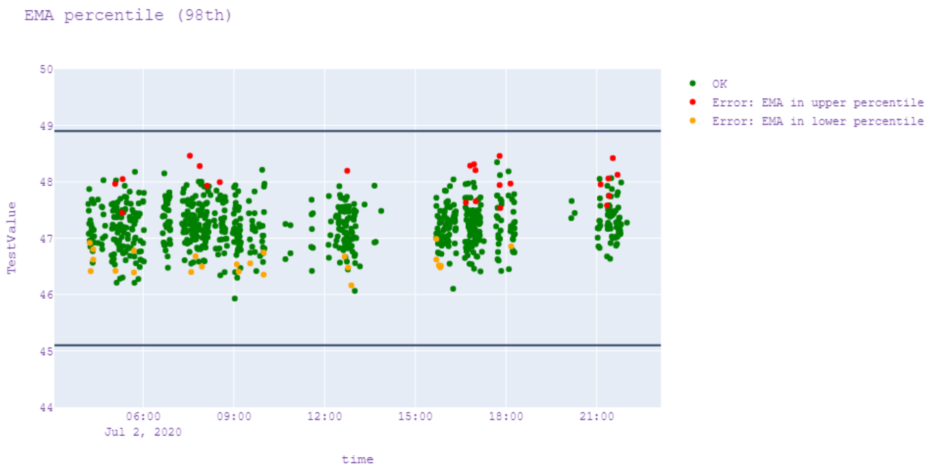


Figure 14: EMA percentile algorithm (window of 100 samples, 98th percentile)

Since the anomaly detection component offers the option of combining the results of different algorithms, the deployment should include a combination of the described algorithms. A warning will be issued if any of the chosen algorithms raises a warning or an error. Border check should be included as it is the simplest and easiest to understand. The more advanced algorithms like the EMA and MACD should also be used in order to spot emerging trends early and reliably.

3.3 Pipe Clogging

This section describes the pipe clogging scenario from the JEMS pilot. The goal is to predict an upcoming malfunction when the waste material clogs the pipe in the JEMS's Syndi waste processing plant. As deliverable FACTLOG D7.1 [2] describes, there are no labels for such historic events in the currently available data. The approach we decided to use therefore is using general anomaly detection to raise alerts to the plant operator, who then interprets the readings and interprets if they indicate an upcoming clogging event.

The data collected from sensors in chambers B100, B200 and B300 of the plant[2] was analysed with different anomaly detection algorithms with the objective to detect faulty behaviour in the production process. In general, two approaches were tested. In the first one we tested the anomaly detection algorithms on multidimensional feature vectors. In the second we only took the most important sensors from a chamber (temperature and pressure) and ran some more statistical anomaly detection algorithms on each data stream separately. Currently, all anomalies are highlighted, and it is up to the domain experts to interpret them.

3.3.1 Multidimensional feature vectors

Figure 15 shows the anomalies detected using the Isolation Forest algorithm (the inputs are the values from both sensors at one point in time) on the data from chamber B200. Yellow dots indicate 'normal' behaviour and purple dots are anomalies. On the x axis the index of the sample is shown. The measurements are spaced at 1-hour intervals. The problem with this data is that the machine was turned off and on multiple times during the time the data was collected. Since the machine was turned off more often the algorithm recognizes those states as "normal" and the operational states as anomalous. Therefore, no clear conclusion about the algorithm's effectiveness can be drawn from this data.

[2] For more details on the plant structure see deliverable D7.1 [7].

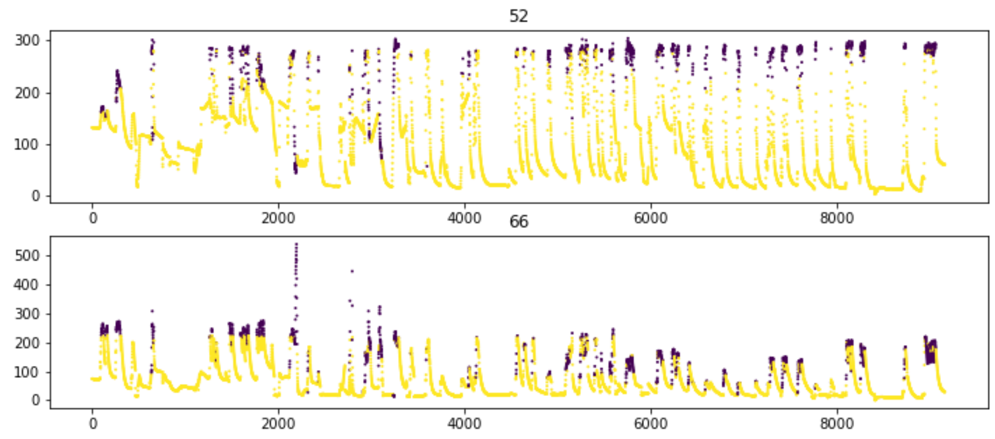


Figure 15: Top - sensor TIC 200 - 27 (temperature), bottom - sensor TIC 200-29 (temperature).

Chamber B300 was not functional during this time interval (it was started only 2 times in the beginning as it can be seen in Figure 16) however, its sensors still recorded data. The temperature detected is mainly just the temperature of the environment (the top graph). However, the pressure detected (bottom graph) should be slightly below zero and even though the chamber was not working it was still connected to the production process and so anomalies on this data can point to anomalies in the system. As can be seen when dealing with stable values the isolation forest algorithm nicely captures the outliers. Therefore, we can assume it would also work well if plugged into a stable data stream of a functional chamber.

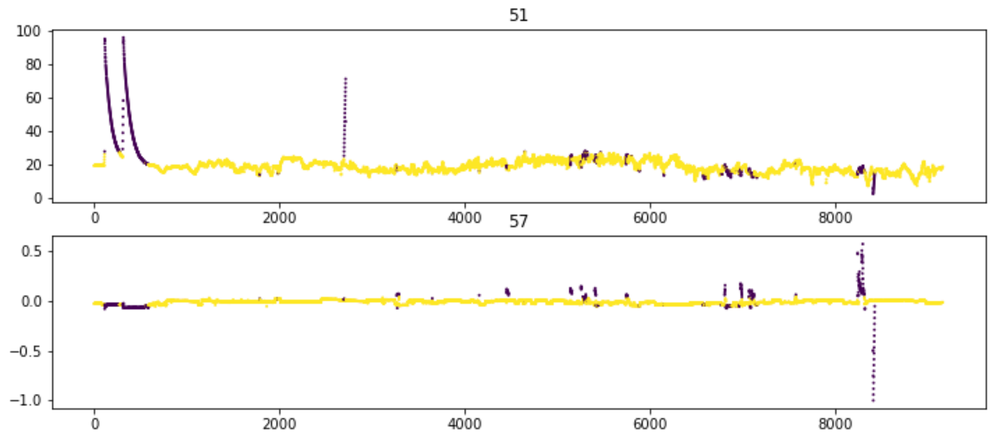


Figure 16: Top – sensor TIC 300-27 (temperature), bottom - sensor PIC 300-29 (pressure).

Figure 17 and Figure 18 illustrate the anomalies found on individual sensors one of temperature and one of pressure. In this case the GAN algorithm was used, which takes 10 consecutive measurements as the input. Blue lines are the variable value and the red and black lines are internal parameters of the algorithm – red dots indicate an anomaly. The pressure graph looks more promising since the obvious spikes are labelled as anomalies.

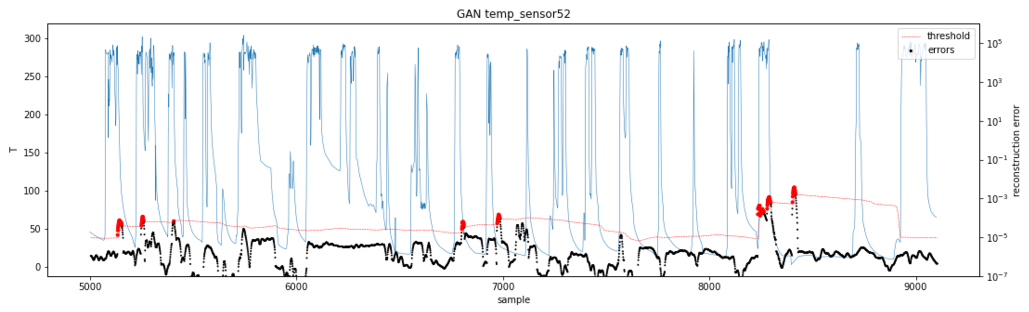


Figure 17: Anomalies on the single - variable analysis of the temperature sensor TIC 200-27.

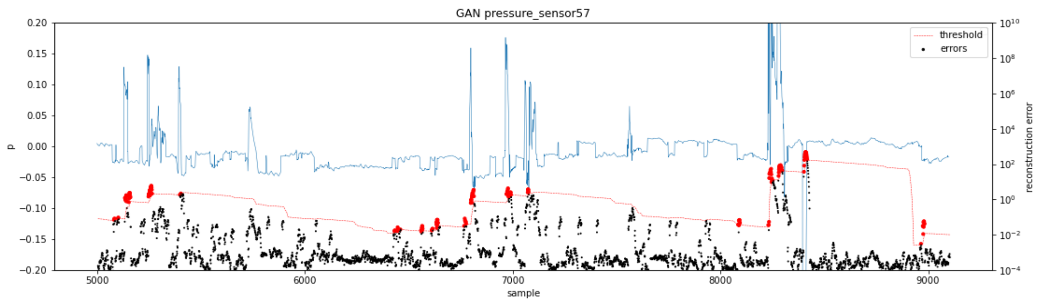


Figure 18: Anomalies on the single - variable analysis of the pressure sensor PIC 300 - 29.

3.3.2 One-dimensional feature vectors

The algorithm used to analyse the temperature and pressure inputs separately analyses last N samples and calculates mean and standard deviation. The interval from mean – X * stdev to mean + X * stdev (where stdev denotes standard deviation) is then considered normal and whatever falls out of that interval is anomalous. With X we can control the threshold for the anomalies.

On the graphs below, the x axis contains a timestamp in Unix timestamp format and on y axis is the observed value. The green dots on the graph represent normal samples, yellow colour means warning, and red is error. The blue dots are located in the beginning of the time series and mean undefined status (the algorithms need some samples to initialize).

Figure 19 and Figure 20 show behaviour in chamber B200 and as with isolation forest we cannot see any useful results.

Figure 21 and Figure 22 show temperature and pressure in B300 respectively. Again, it needs to be noted that this stage of the machine was not operational during this time. However, we can still see that the algorithm can detect anomalies (or issue warnings) on time intervals when the values are more stable. The down side of this approach is that when the data begins to deviate more often it no longer detects it as an anomaly.

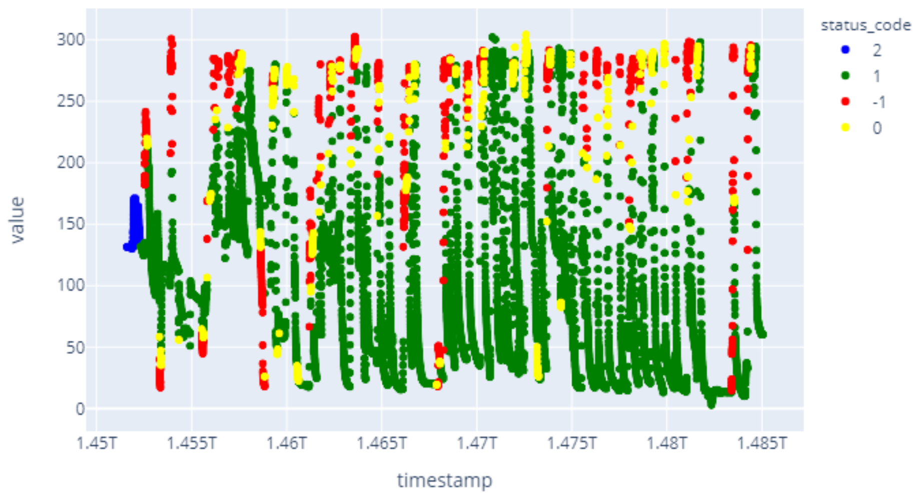


Figure 19: TIC_200_27 (Welford's algorithm with N=200, X=2)

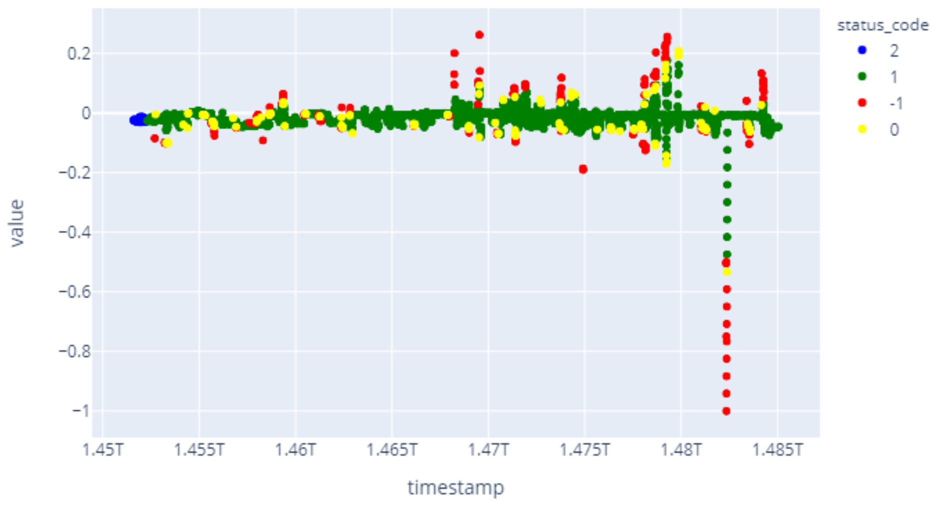


Figure 20: PIC_200_29 (Welford's algorithm with N=200, X=3)

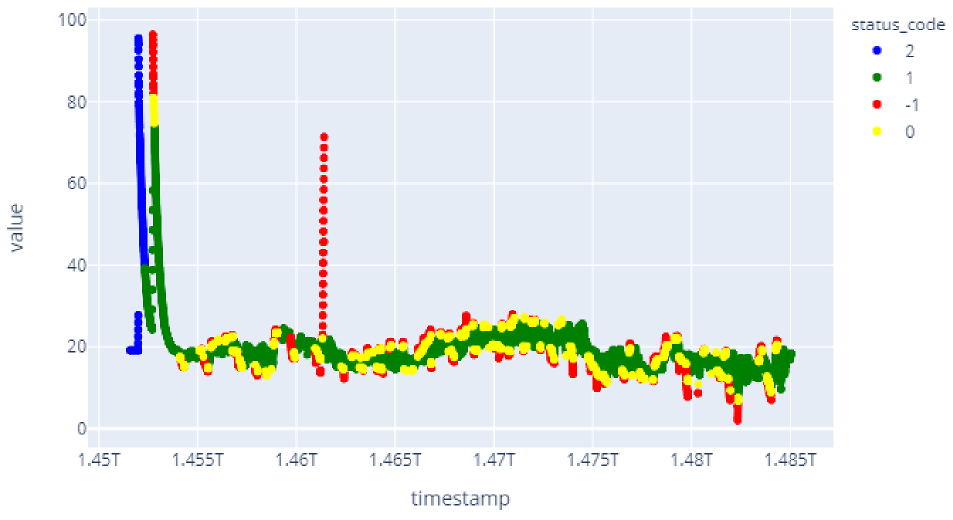


Figure 21: TIC_300_27 (Welford's algorithm N=200, X=2)

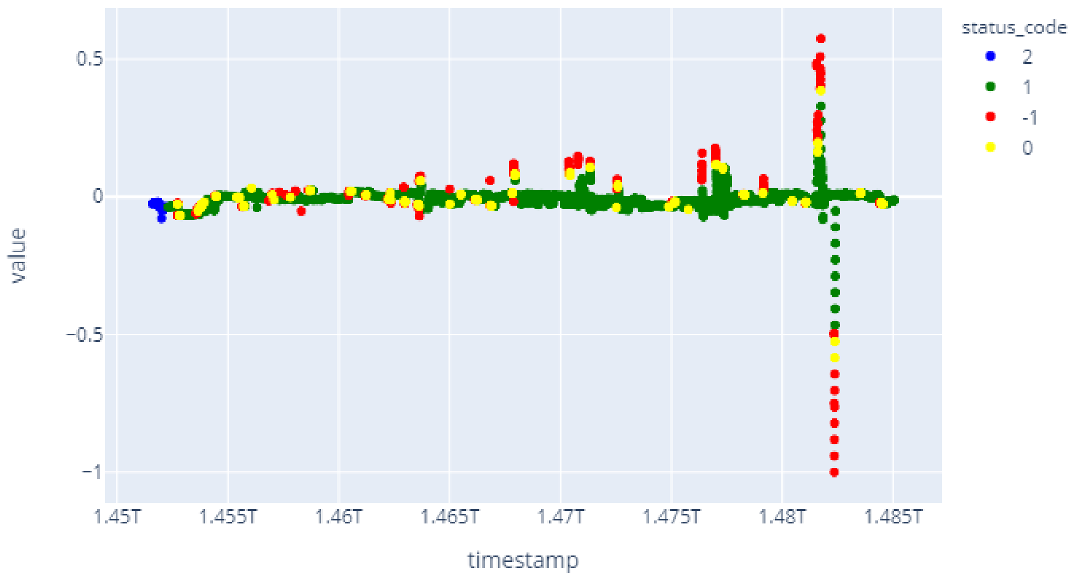


Figure 22: PIC_300_29 (Welford's algorithm N=200, X=3)

To gain the best insights from the different anomaly detection algorithms a combined approach seems best at the moment. Isolation forest and GAN are good at identifying value spikes, whereas Welford's algorithm can assist in keeping the waste processing process at optimal parameters.

3.4 StreamStory Application

StreamStory was used to model the state of the plant in the JEMS use-case. As already mentioned in Section 3.3 and described in greater detail in the deliverable D7.1 [2], the historic JEMS plant data does not have any labels of malfunctions or the plant state. An unsupervised approach was used to build a model of the process through StreamStory.

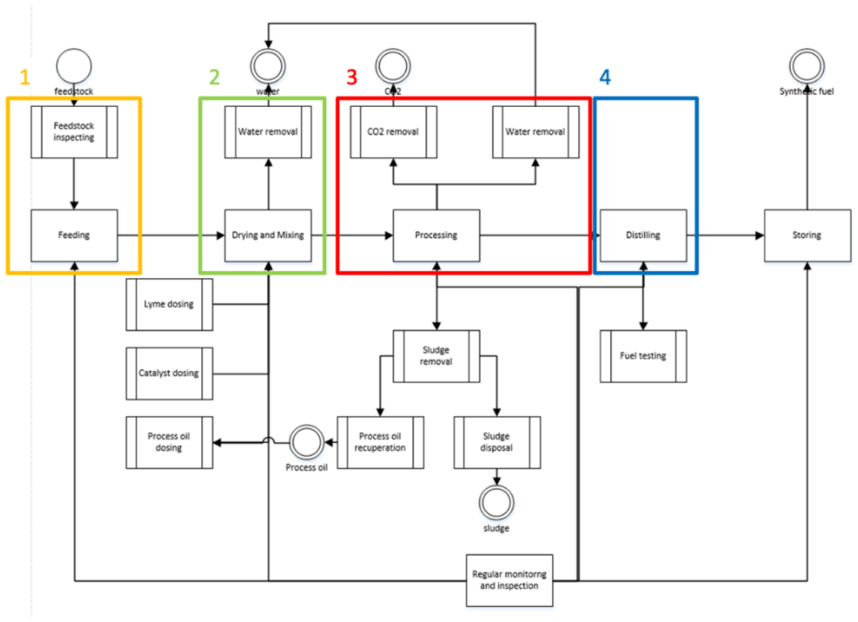


Figure 23: Stages of the JEMS pipeline: 1 - Feedstock inspection and Feeding, 2 - Drying and Mixing, 3 - Processing, 4 – Distilling

As the entire plant is too complex a system, we split the process into stages pictured in Figure 23, namely: 1 - Feedstock inspection and Feeding, 2 - Drying and Mixing, 3 - Processing, 4 - Distilling. Stage 1 does not have corresponding sensors, whereas stages 2, 3, and 4 correspond to chambers B100, B200, and B300. StreamStory was used on sensors from the three chambers to build models of these three stages.

Figure 24 below shows the model pf the chamber B100. After consulting the domain experts from JEMS, we interpreted that the nodes E, D and B correspond to normal operational cycles of the plant. E is when the plant is off, D corresponds to normal operation and B to peak-performance operation. Nodes A and C correspond to events which are very rare and can be considered anomalies. The models can also be more challenging to interpret as is the case of the model for chamber B200 in Figure 25. Here, both nodes, G and H, contain times when the plant is non-operational, but the plant valves were left in different idle states, which does not really carry any meaning. States C, D and E correspond to different levels of operation of the chamber (they also have different internal temperatures). The other nodes are different anomalies, one of which is due to noise in the data. The model for B300 is not useful as the chamber was not operational and the clustering mostly picked up noise.

Attempts were made to model the subsystems of the plant that span over multiple chambers. For example, Figure 26 shows the model of the water removal subsystem which is connected to chambers B100 and B200 (shown by the two boxes in the top-middle in Figure 23). Though not that much bigger than the others, the model is much harder to interpret and is so far a matter of ongoing work in collaboration with domain experts which are helping in refining the selection of sensors and modifying the method parameters to possibly obtain a more meaningful representation.

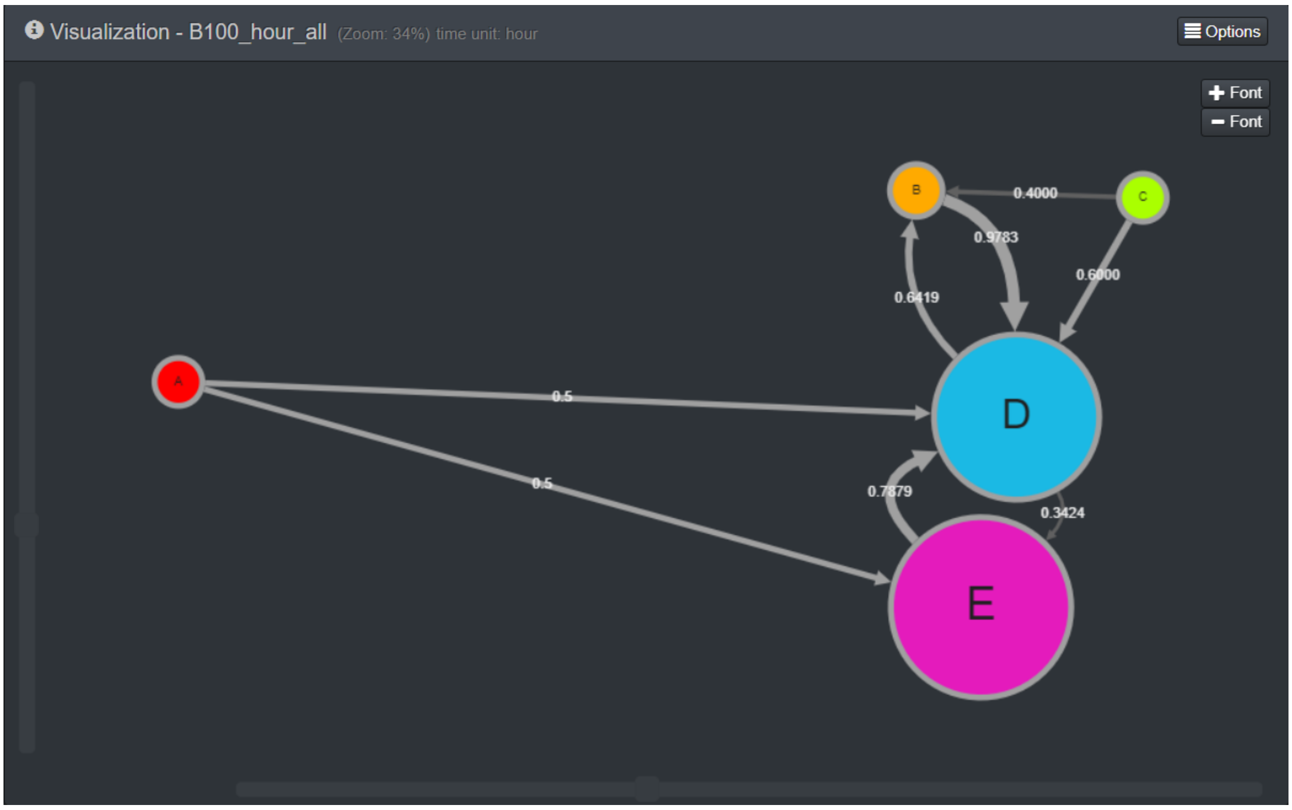


Figure 24: StreamStory model of the data from chamber B100.

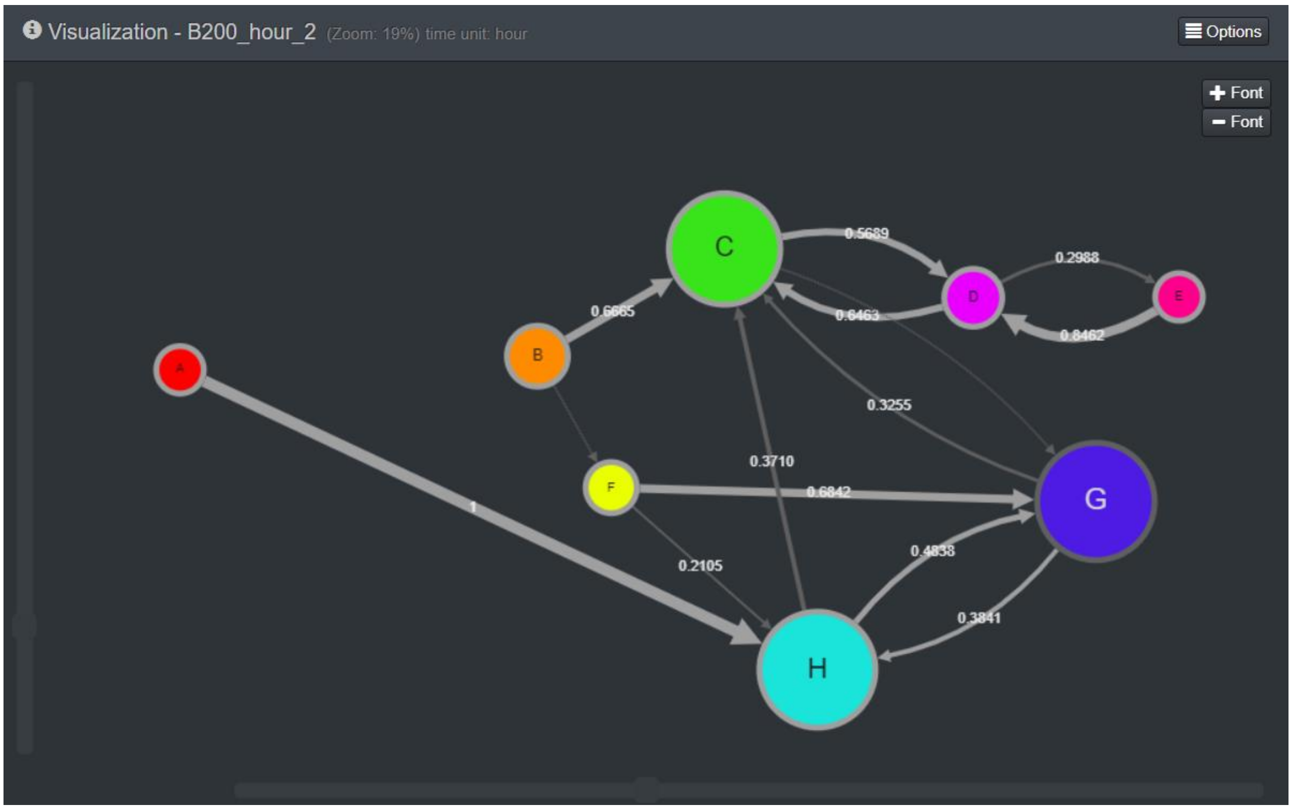


Figure 25: StreamStory model of the data from chamber B200.

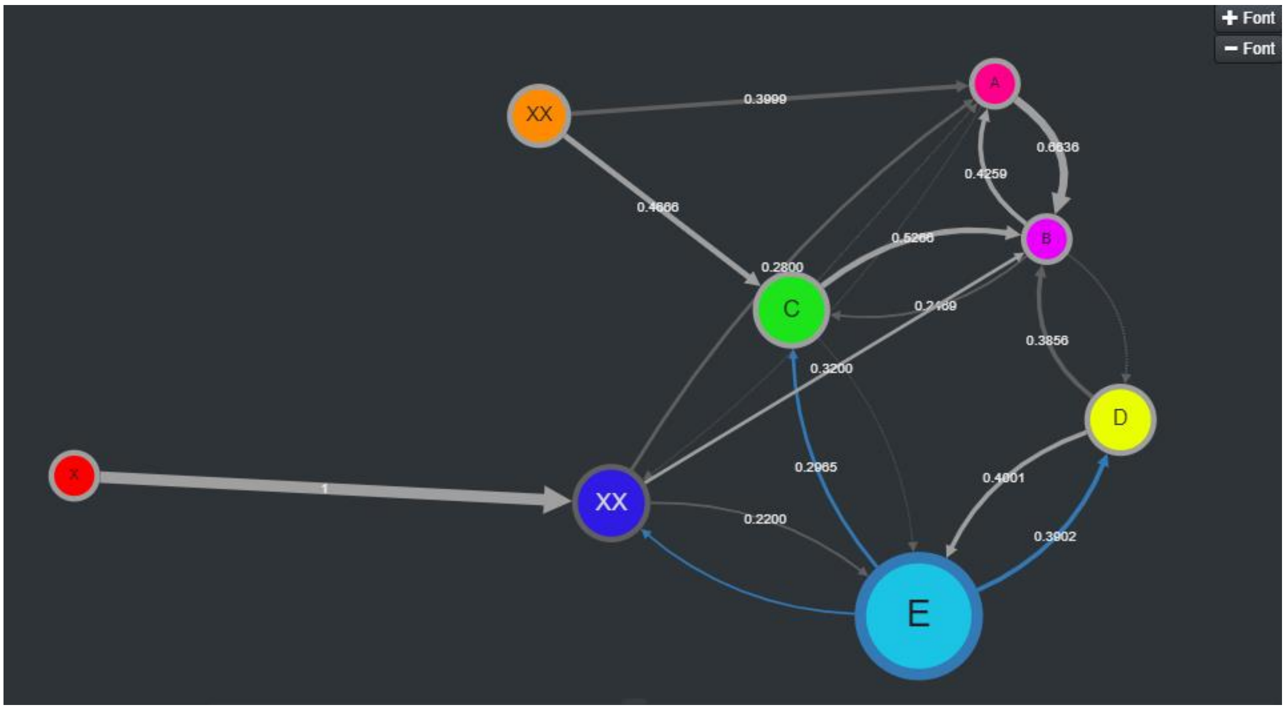
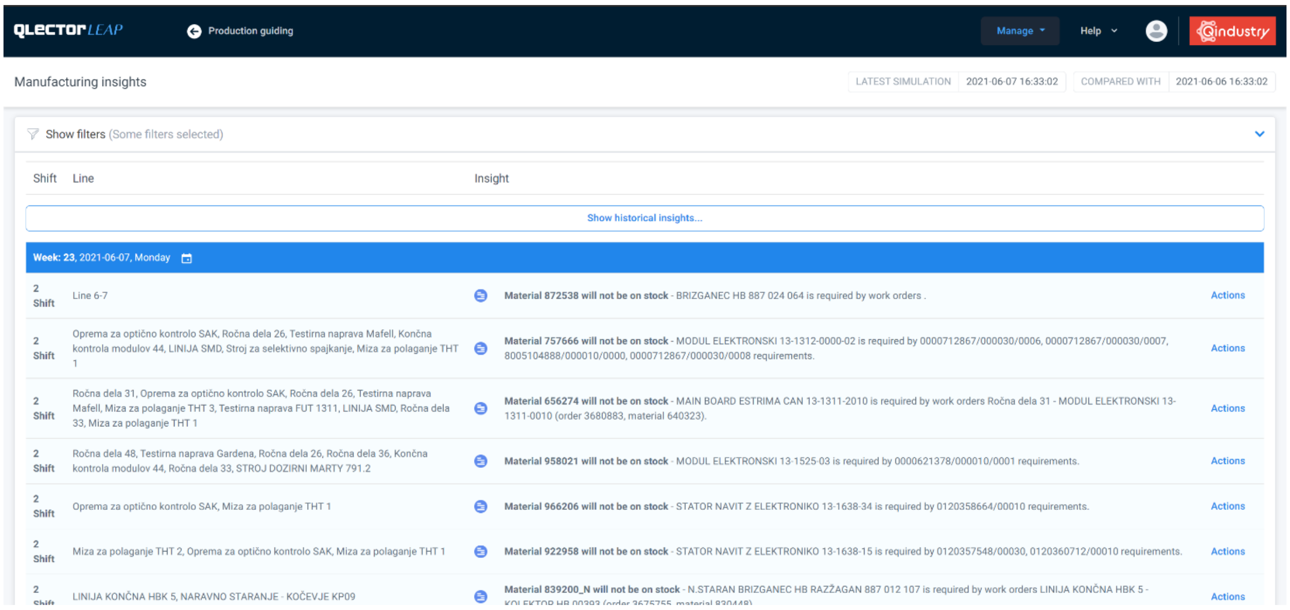


Figure 26:StreamStory model for the water removal subsystem.

3.5 Organisational Anomalies

Usually, the most important insights in practice are not (yet) based on the advanced ML methodologies, but rather on a comprehensive picture of the manufacturing process, which is encoded into a knowledge graph. By following the knowledge graph and matching it with the modelling capabilities in real time (see FACTLOG D2.3 [1]) one can extract extremely useful information out of the available data.


QLECTOR LEAP
Production guiding
Manage
Help
Qindustry
Show filters (Some filters selected)
Show historical insights...

Manufacturing insights

LATEST SIMULATION 2021-06-07 16:33:02 COMPARED WITH 2021-06-06 16:33:02

Week: 23, 2021-06-07, Monday

Shift	Line	Insight	
2 Shift	Line 6-7	**Material 872538 will not be on stock** - BRIZGANEC HB 887 024 064 is required by work orders .	Actions
2 Shift	Oprema za optično kontrolo SAK, Ročna dela 26, Testirna naprava Mafell, Končna kontrola modulov 44, LINIJA SMD, Stroj za selektivno spajkanje, Miza za polaganje THT 1	**Material 757666 will not be on stock** - MODUL ELEKTRONSKI 13-1312-0000-02 is required by 0000712867/000030/0006, 0000712867/000030/0007, 8005104888/000010/0000, 0000712867/000030/0008 requirements.	Actions
2 Shift	Ročna dela 31, Oprema za optično kontrolo SAK, Ročna dela 26, Testirna naprava Mafell, Miza za polaganje THT 3, Testirna naprava FUT 1311, LINIJA SMD, Ročna dela 33, Miza za polaganje THT 1	**Material 656274 will not be on stock** - MAIN BOARD ESTRIMA CAN 13-1311-2010 is required by work orders Ročna dela 31 - MODUL ELEKTRONSKI 13-1311-0010 (order 3680883, material 640323).	Actions
2 Shift	Ročna dela 48, Testirna naprava Gardena, Ročna dela 26, Ročna dela 36, Končna kontrola modulov 44, Ročna dela 33, STROJ DOZIRNI MARTY 791.2	**Material 958021 will not be on stock** - MODUL ELEKTRONSKI 13-1525-03 is required by 0000621378/000010/0001 requirements.	Actions
2 Shift	Oprema za optično kontrolo SAK, Miza za polaganje THT 1	**Material 966206 will not be on stock** - STATOR NAVIT Z ELEKTRONIKO 13-1638-34 is required by 0120358664/00010 requirements.	Actions
2 Shift	Miza za polaganje THT 2, Oprema za optično kontrolo SAK, Miza za polaganje THT 1	**Material 922958 will not be on stock** - STATOR NAVIT Z ELEKTRONIKO 13-1638-15 is required by 0120357548/00030, 0120360712/00010 requirements.	Actions
2 Shift	LINIJA KONČNA HBK 5, NARAVNO STARANJE - KOČEVJE KP09	**Material 839200_N will not be on stock** - N.STARAN BRIZGANEC HB RAZŽAGAN 887 012 107 is required by work orders LINIJA KONČNA HBK 5 - KOLEKTOR HB 00393 (order 3675755, material 830448)	Actions

Figure 27: Insights from QlectorLEAP.

Figure 27 depicts the insights screen from QlectorLEAP, where a particular subclass of the anomalies is shown. The anomalies here are mostly based on the material stocks, however, other organisational anomalies are being proactively spotted here, those include the needs for multiple simultaneous tool exchanges, delay due to the bad scheduling of the people, insights on order delays and similar.

All of these insights are based on the modelling outputs [1], where realistic simulations replace the normative values for a particular manufacturing process. By combining these results with the complex knowledge graph, which includes the hierarchy of the processes, bills of materials, hierarchy of production lines, machines, operations and others, insightful information can be presented to the planners and shop floor manages.

The power of combining the knowledge graph with usually quite simple (mostly linear) models in every step of the manufacturing process, has yielded amazing results and possesses surprisingly valuable information for the end users. A lesson learned here is, that when a process is complex it is much more important that you are able to model it in its entirety (even with simple methods). This comprises much more value than a partially modelled process with state-of-the-art methodologies, which are more complex, less robust and can only improve the KPIs by a very small margin, compared to the holistic approach.

4 Implementation

This section presents the implementation of the anomaly detection modules developed during FACTLOG.

4.1 Anomalous Event Detection

Anomaly detection component implements several algorithms for anomaly detection on time series. A time series is a sequence of data points (samples) that were collected during some time period. Anomaly detection techniques on this kind of data are different than the techniques for batch anomaly detection. Furthermore, time series are usually live data streams so the algorithms for handling such data need to be fast (real-time) and regularly updated for the newest data.

The anomaly detection component is therefore capable of taking in data streams, detecting anomalies and outputting results sample by sample.

Architecture:

The anomaly detection component is divided into subcomponents so that each covers a certain functionality in the pipeline. Abstract classes provide an interface which the implementations of the components must satisfy. This approach makes it easy to implement new components and combine them in the way suitable for a certain use case. The anomaly detection pipeline shown in Figure 28, consists of three main components (consumer, anomaly detection and output) and two optional components (normalization and visualization).

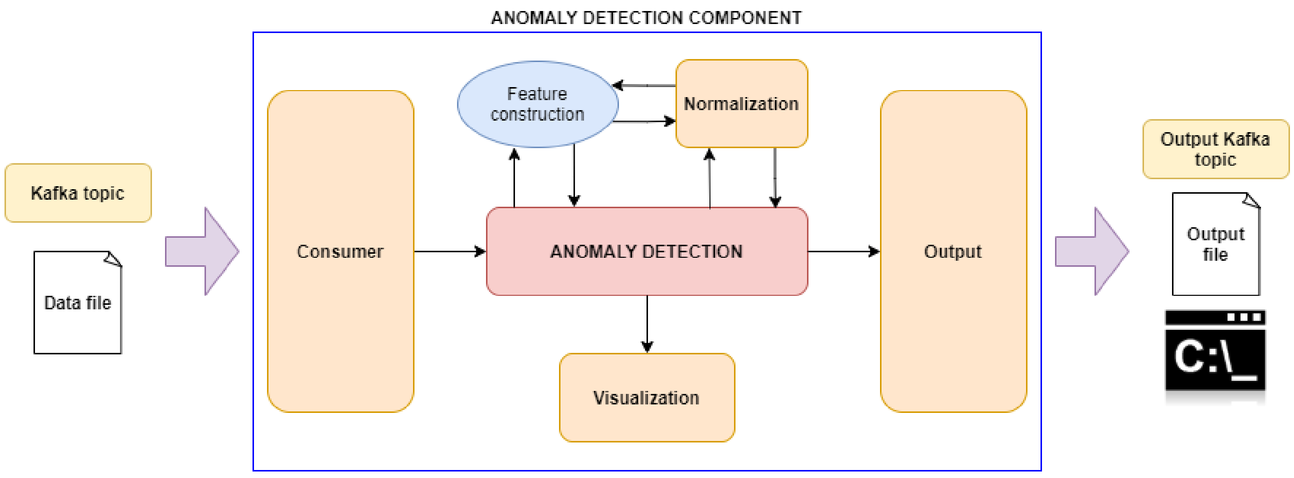


Figure 28: Anomaly detection architecture.

1. **Consumer component** reads the data from the source, extracts the required information and passes it to the anomaly detection component. The source can be one of the following:
    - Kafka topic,
    - File,
    - File and Kafka topic (the first part of the stream is saved in the file and then continues on the topic).

2. **Anomaly detection component** accepts the information from the consumer component, constructs additional features (if specified) and executes some anomaly detection algorithm on this sample (section 4.1 Methods provides more information about specific algorithms). Optionally this component also communicates with normalization and visualization components. Finally, it passes the result of the algorithm to the output component (or components if more than one is specified).

3. **Output component** accepts the result of the anomaly detection algorithm and exports it to the output media. Currently three different output components are implemented:
   - Kafka topic,
   - File,
   - Terminal (mainly for experiments and tests)

4. **Normalization component** is an optional component which “suggests” a normal value when an anomaly is detected. Currently two algorithms for producing “normal” value are implemented:
   - Average of last N samples,
   - Average of last N samples with a specified period (useful, for example, if we want to get an average of a sample from the same time of day for the last N days)

   This normalized value is then used also in feature construction for features like shifts and averages.

5. **Visualization component** was mainly intended for testing new algorithms and experimenting with new data. The data from anomaly detection component can be visualized in any way the user requires, however the two most common and general are:
   - Status points visualization: The first element of the feature vector (accepted by the anomaly detection) is plotted in a graph with the colour of the dot representing the status of the sample (white: OK, yellow: warning, red: error, blue: undefined) - Figure 29.

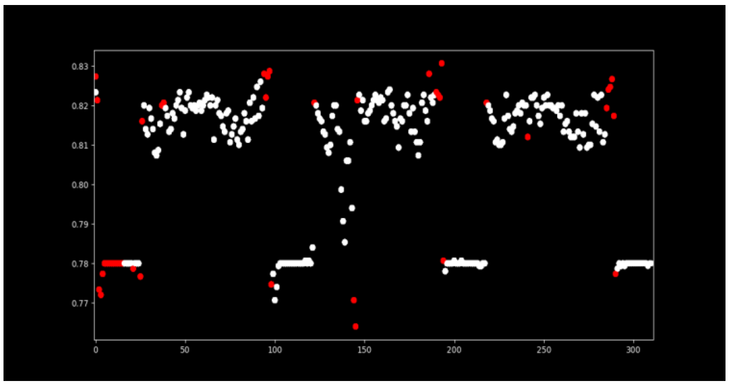


Figure 29: Status points visualization.

   - Histogram visualization: Again, the first element of the feature vector (accepted by the anomaly detection) is plotted in a histogram that shows the distribution of the data - Figure 30.

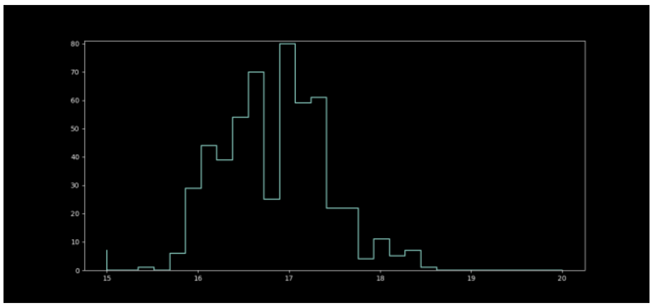


Figure 30: Histogram visualization.

4.1.1 Methods

In this section some of the implemented anomaly detection algorithms are described. Some of them were already briefly described in section 2. As mentioned, new algorithms can easily be added as long as they follow the interface defined in the abstract class.

1. **Border check:** Border check aims to issue a warning when a measurement is close to either the upper or lower bound. Multiple warning stages can be implemented, based on how close the measurement is to the bound. We define the mid - point of the tolerance range using the given upper limit (UL) and lower limit (LL).

$$MP = \frac{UL + LL}{2}$$

Warning stages are then defined at fixed percentage distances from the mid - point to either bound. For example, for UL = 1, LL = -1, a warning stage 0.8 would mean that a warning is issued if a measurement falls below -0.8 or above 0.8.

2. **Exponential moving average:** Exponential moving average (EMA) is a running variable, which is the weighted average of previous measurements, giving the more recent measurements a larger weight. The advantage of using running averages is to avoid false-positive errors. A single measurement might be considerably different than the expected value, but as long as a large number of such measurements is not observed it is still not problematic. The current EMA(t) is calculated in the following way:

$$EMA(t) = x * k + EMA(t-1) * (1-k),$$

where 0<k<=1. The exponential moving average takes this into account and only issues warnings if many anomalous measurements are observed. The principle for issuing warnings is similar to Border check, except that now we are comparing the value of the running exponential average to the warning stages.

3. **Moving Average Convergence Divergence:** Moving Average Convergence Divergence (MACD) is derived from the financial world, where it is used to detect trends in stocks. In our case, it is also advantageous to detect an emerging trend, even before the measurements are close to the limits. The MACD is simply a difference of two exponential moving averages with different periods.

$$MACD = EMA(short\ period) - EMA(long\ period)$$

A shorter period EMA will follow the data more closely, as the weights of recent measurements are much larger, while a longer period EMA will be slower to react to a trend shift as also much older measurements still contribute to its calculation. A large difference between the values of different period EMAs thus indicates that the data is in a trend. Changing the two periods, we can adapt the MACD to the severity of the trend that we would like to detect.

4. **Welford's algorithm:** Welford's algorithm is very similar to border check, with the difference that at every step the upper limit (UL) and lower limit (LL) are calculated from mean and standard deviation of previous data in the following way:

$$UL = mean + x * stdev$$

$$LL = mean - x * stdev$$

where x is a prespecified parameter. When calculating mean and standard deviation we consider two cases:

- mean and standard deviation are calculated from a fixed number last received samples
- mean and standard deviation are calculated from all previously received samples. This is done using Welford's algorithm for online mean and standard deviation calculation:

$$\overline{x_n} = \overline{x_{n-1}} + \frac{x_n - \overline{x_{n-1}}}{n}$$

$$s_n = s_{n-1} + (x_n - \overline{x_{n-1}}) * (x_n - \overline{x_{n-1}})$$

This algorithm can be very useful in cases where the data is normally distributed, but we do not know the parameters of the distribution or if the distribution can change.

5. **EMA Percentiles:** This algorithm is a combination of the concepts used in the EMA and Border check algorithms. An exponential moving average is calculated in the same way as before. The difference here is in the decision to label the data point an outlier or not. Each new EMA value is compared to the last N values (N is the size of the comparative window which is also pre-specified), to calculate which percentile the new value falls in. A limit percentile is pre-specified in order to label the EMA values which fall further from the mean as outliers. For example, if the 99th percentile is specified, an error will be raised if the latest EMA value is in the top or bottom 1% of the last N EMA values. The advantage of this method is easy setup, as we can use pretty much the same parameters for any data set. The disadvantage is, that by definition there will always be measurements labelled as outliers, even if all the data points are well within the desired range (there will always be a top and bottom percentile).

Additionally, we have tested state-of-the-art methodologies like Generative Adversarial Networks (GANs) and Isolation Forests. It is notable that in some use cases these more complex methods do not seem to give us additional value in comparison with simpler methods.

The presented methods, due to their simplicity are not able to recognise anomalies in terms of deviations from some complex behaviour (e.g. seasonality) but are rather aimed at detecting more localised anomalies like sudden jumps in the stream data or emerging trends, where the observed values slowly approach the given upper or lower limit. Multiple algorithms can be used in combination for example, one algorithm which is able to detect a trend in combination with an algorithm which detects sudden jumps. A warning can then be issued in case either of the chosen algorithms reports an anomaly.

4.1.2 API specification

The component's functionalities are meant to be used through the main.py script that initializes all of the components as it is specified in the configuration file. Then it also starts the execution of the consumer component.

4.1.2.1 Running the script

The script needs to be run in a Python environment with all the required packages installed. The command for executing the main.py script should follow the following structure:

```
python main.py [-h] [-c CONFIG] [--f]
```

Short	Long	Description
-h	--help	Show help
-c	--config	Name of the configuration file located in configuration folder
-f	--file	If this flag is used the program will read data from file specified in config file.
-fk	--filekafka	If this flag is used the program will read data from file specified in config file and then from the Kafka topic.

The only required flag is the configuration flag that specifies the location and file name of the configuration file. If neither –file or –filekafka flags are used a Kafka consumer is assumed.

4.1.2.2 Configuration file

In this section the structure of configuration file will be presented. Below we can see a general example of a such file. The consumer configuration section contains different fields depending on the type of consumer.

Kafka consumer:

- bootstrap_server
- auto_offset_reset
- enable_auto_commit
- group_id
- value_deserializer
- topics: (a list of topics that are consumed)

File consumer:

- file_name

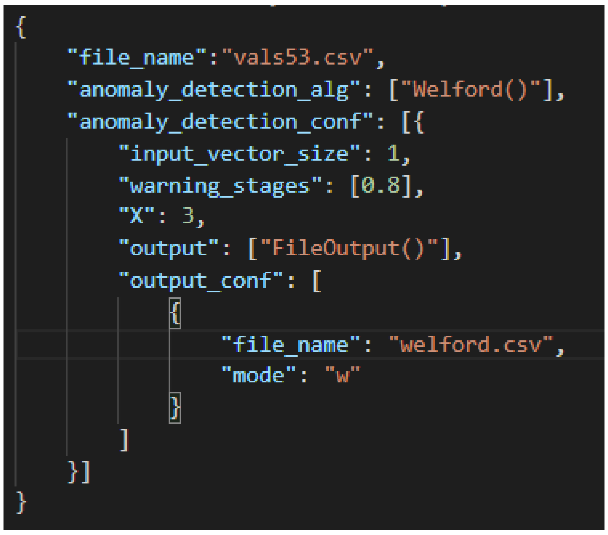
File Kafka consumer contains fields from both previous consumers.

```
{
    ...
    consumer configuration
    ...
    "anomaly_detection_alg": […],
    "anomaly_detection_conf": [{
        ...
        anomaly detection configuration
        ...
        "input_vector_size": ...,
        "averages": [...], # optional
        "shifts": [...], # optional
        "time_features": [...], # optional
        "normalization": …, # optional
        "normalization_conf": …, # optional
        "output": […],
        "output_conf": […],
        "visualization": …, # optional
        "visualization_conf": … # optional
      }]
}
```

The anomaly_detection_alg field contains a list of algorithm's names that each correspond to a Kafka topic. If the file consumer is used only the first algorithm will be used to analyze the file. anomaly_detection_conf field also contains a list of configurations for a specific anomaly detection algorithm specified in the previous filed. Apart from the algorithm-specific configurations all must contain the following fields:

- input_vector_size - the size of the input vector
- averages - optional parameter specifying additional average features to be constructed
- shifts - optional parameter specifying additional shift features to be constructed
- time_features - optional parameter specifying additional time features to be constructed
- normalization - optional parameter specifying the normalization component used
- normalization_conf - optional parameter specifying the configuration of the normalization component
- visualization - optional parameter specifying the visualization component used
- visualization_conf - optional parameter specifying the configuration of the visualization component used
- output - a list of output components (in case you want to output the result to different outputs)
- output_conf - a list of configurations for the output components

Figure 31 below shows an example configuration for the Welford's algorithm.

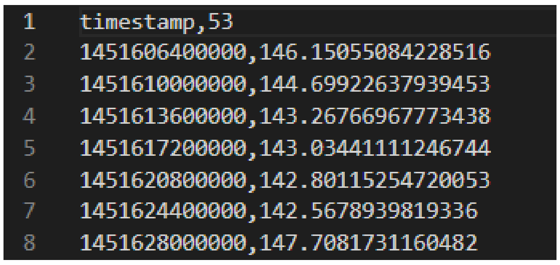


```
{
    "file_name":"vals53.csv",
    "anomaly_detection_alg": ["Welford()"],
    "anomaly_detection_conf": [{
        "input_vector_size": 1,
        "warning_stages": [0.8],
        "X": 3,
        "output": ["FileOutput()"],
        "output_conf": [
            {
                "file_name": "welford.csv",
                "mode": "w"
            }
        ]
    }]
}
```

Figure 31: Example configuration file for Welford's algorithm

4.1.2.3 Input data format

If the data is provided as a csv file it must contain a "timestamp" field for the time value in Unix timestamp format as shown in the example in Figure 32. All the other fields in the file will be included into the feature vector.



```
timestamp,53
1451606400000,146.15055084228516
1451610000000,144.69922637939453
1451613600000,143.26766967773438
1451617200000,143.03441111246744
1451620800000,142.80115254720053
1451624400000,142.5678939819336
1451628000000,147.7081731160482
```

Figure 32: Example of csv input (B100 sensor 53)

If the data is provided as a JSON file or through a Kafka topic it must be in the following format:

```
{
        "ftr_vector":  array (a feature vector) of values,
        "timestamp": timestamp as strings in the Unix timestamp format
}
```

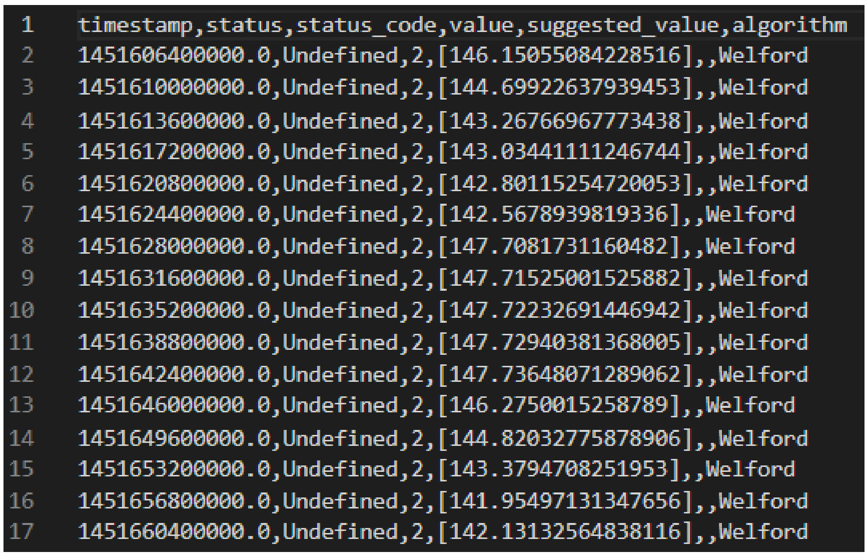
4.1.2.4 Output data format

The output data can be formatted in many different ways (see an example in Figure 33). In this section the format of the Kafka and file output component will be presented.

If the output file is in csv format the file will contain the following fields:

- timestamp - timestamp of the sample
- value - the sample (feature vector) analysed
- status_code - an integer holding information about anomalousness of the sample

- status - a string describing the status code
- suggested_value - an optional field that suggests "normal" value (for the anomalous samples)
- algorithm - the name of the anomaly detection algorithm used



```
timestamp,status,status_code,value,suggested_value,algorithm
1451606400000.0,Undefined,2,[146.15055084228516],,Welford
1451610000000.0,Undefined,2,[144.69922637939453],,Welford
1451613600000.0,Undefined,2,[143.26766967773438],,Welford
1451617200000.0,Undefined,2,[143.03441111246744],,Welford
1451620800000.0,Undefined,2,[142.80115254720053],,Welford
1451624400000.0,Undefined,2,[142.5678939819336],,Welford
1451628000000.0,Undefined,2,[147.7081731160482],,Welford
1451631600000.0,Undefined,2,[147.71525001525882],,Welford
1451635200000.0,Undefined,2,[147.72232691446942],,Welford
1451638800000.0,Undefined,2,[147.72940381368005],,Welford
1451642400000.0,Undefined,2,[147.73648071289062],,Welford
1451646000000.0,Undefined,2,[146.2750015258789],,Welford
1451649600000.0,Undefined,2,[144.82032775878906],,Welford
1451653200000.0,Undefined,2,[143.3794708251953],,Welford
1451656800000.0,Undefined,2,[141.95497131347656],,Welford
1451660400000.0,Undefined,2,[142.13132564838116],,Welford
```

Figure 33: Example of csv output (B100 sensor 53)

The JSON file or Kafka output contain the same fields only the whole output is in JSON format.

4.1.3 Deployment

The component (with Kafka consumer) can be started with a command "python main.py -c path_to_configuration_file.json" (optionally -f or -fk flags can be added for file or file-kafka consumer). All the other parameters are specified in the configuration file. To avoid running a separate instance of the anomaly detection component for every data stream, multiple input Kafka topics can be specified and for each one a separate anomaly detection algorithm as pictured in the diagram in Figure 34.

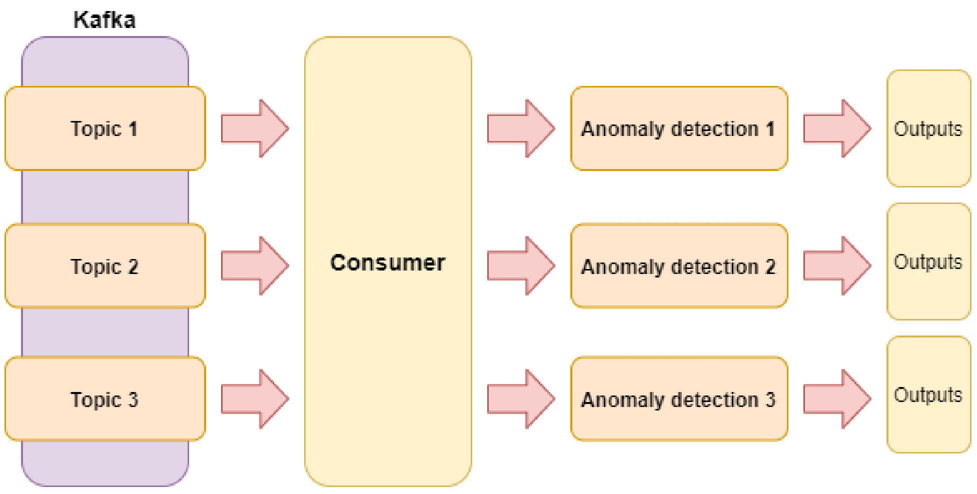


Figure 34: Multiple stream anomaly detection

5 Conclusions

As stated before, in the FACTLOG architecture anomaly detection methods play the role of the eyes and ears. They need to be able to capture the occurring or upcoming problems, so that the other components can act. This document presented the wide array of methods available for this purpose in the anomaly detection system and demonstrated their use in the use-cases. Minor extensions may be added as the use-case deployments reach maturity, but overall the system should be able to address the main anomaly detection needs in the project.

In the continuation of the project, deployment of these methods in the use-cases is to be finalised and the orchestration logic needs to be prepared that handles the alerts raised and shows them to human operators and/or hands them on to the other components. Calibration of the anomaly detection models is also key, so that the alerts raised are "just right" and don't miss critical events or flood the output with useless warnings.

References

[1] K. Kenda, J. Rupnik, M. Karlovčec, B. Fortuna, M. Grobelnik and G. Petkovšek, “D2.3 Holistic Model of Uncertainty and Causal Relations,” FACTLOG, Ljubljana, 2021.

[2] Y. Mourtos, G. Zois, S. Lounis, E. Zampou, G. Kasapidis, P. Repoussis, J. Lu, A. Košmerlj, K. Kalaboukas, M. Koukovini, A. Mousas, N. Dellas, E. Papagiannakopoulou, G. Lioudakis, P. Laras, K. Michalis, K. Helen, K. Kenda, J. Rožanec, G. Arampatzis, G. Tsinarakis, N. Sarantinoudis, P. Eirinakis and G. Koronakos, “D7.1 Installation and Initial Testing (Interim Version),” FACTLOG, 2021.

[3] A. Bhattacharya, “Effective Approaches for Time Series Anomaly Detection,” towardsdatascicence.com, 2021. [Online]. Available: https://towardsdatascience.com/effective-approaches-for-time-series-anomaly-detection-9485b40077f1. [Accessed 8 June 2021].

[4] K. Kenda and D. Mladenić, “Autonomous Sensor Data Clearning in Stream Mining Setting,” *Business Systems Research Journal,* vol. 9, no. 2, pp. 69-79, 2018.

[5] F. T. Liu, K. M. Ting and Z. H. Zhou, “Isolation Forest,” in *8th IEEE International Conference on Data Mining*, 2008.

[6] S. C. Tan, K. M. Ting and T. F. Liu, “Fast anomaly detection for streaming data,” *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence,* vol. 2, pp. 1511-1516, 2011.

[7] S. Hariri, M. Carrasco Kind and R. J. Brunner, “Extended Isolation Forest,” *IEEE Transactions on Knowledge and Data Engineering,* vol. 33, no. 4, pp. 1479-1489, 2018.

[8] L. Stopar, P. Skraba, M. Grobelnik and D. Mladenic, “StreamStory: Exploring Multivariate Time Series on Multiple Scales,” *IEEE Transactions on Visualization and Computer Graphics,* vol. 25, pp. 1788-1802, 2019.

[9] W. Jiang, Y. Hong, B. Zhou, X. He and C. Cheng, “A GAN-Based Anomaly Detection Approach for Imbalanced Industrial Time Series,” *IEEE Access,* vol. 7, pp. 143608-143619, 2019.