Changepoint Influenced Anomaly Detection

University of Glasgow
Tamas Suli
2022

Abstract

The current project discovered and demonstrated the practical usability of two machine learning algorithms in an industrial setting. The recently implemented Stochastic Gradient Descent based One-Class Support Vector Machine of Scikit-Learn Python library was assessed on real multivariate industrial data. In addition, a univariate nonparametric changepoint detection algorithm – implemented in the *changepoint.np* R package – was applied on a univariate time series data. Both applications represent relevant approaches in real life setting.

The time series data used for the assessment was taken from 'The Open Industrial Data Project' accessible by the *openindustrialdata.com* platform. It is an open-source project that makes available a real, operational industrial data generated by sensors of a single compressor. The compressor operates on the Aker BP's Valhall oil platform in the North Sea. Its data has been collected since 2013 by 363 sensors. The current paper utilizes a smaller subset of the data that represents approximately 6 months of operation from June to December in 2021.

The thesis contributes to knowledge by answering two research questions while analysing the data mentioned. It is aimed to explore if the *changepoint.np* R package can support the recognition of anomalies in the output variable time series as proposed in the literature (Amiri & Allahyari, 2012, p. 674; Arellano et al., 2021, p. 3). Anomaly is defined for this as short term changes of the mean or variance of the time series. It is also examined if the recently implemented Stochastic Gradient Descent (SGD) based One-Class Support Vector Machine (OCSVM) is suitable for the analysis of large amount of sensor data in a common desktop environment. This is to see the contrast between applying SGD based OCSVM and the standard OCSVM (Schölkopf et al., 1999), since the latter requires large IT infrastructure with some servers to handle the kind of data mentioned (Lai et al., 2021).

The assessment was carried out by completing an end-to-end analytics workflow to evaluate the results and experiences gained considering runtimes of algorithms and predictive power of models measuring it by F1 score. It was confirmed that *changepoint.np* R package can support anomaly detection workflows by labelling an output variable data series. Though, it has a limitation in terms of runtime, because the current data set was stretching its capabilities. It was also confirmed that the SGD based OCSVM algorithm of Scikit-Learn library is highly capable of processing a data set size as the current one, and it could create a model with high predictive power. Furthermore, it indicated its ability to process a five to ten times greater data set conveniently by a common desktop environment. The common desktop environment was represented by a computer having an Intel Core i5 CPU with 8 cores and 16GB memory.

Content

1 Introduction

Understanding the behaviour of industrial processes have been important for engineers, business leaders, regulatory agencies, and other stakeholders. Their expectations are not the same. However, all those can be fulfilled usually if the processes operate as planned, and do not deviate. Thus, it has been an equally long endeavour to identify when and why industrial processes do not meet such expectations (Smith, 2021).

For this, time series process data is an important information source. It was used often when attempting to recognize process deviations. An early – but very impactful – publication on this topic was written by Shewhart (1930). His univariate approach was called 'Statistical Process Control' (SPC) and utilized the mean and variance of process data. His method has been evolving since then. It became a continuously maintained formal industrial standard in many engineering fields. Nowadays, it is applied to control the more risky process and product characteristics when health & safety or legal regulations are in concern (Bolivar, 2005; *ISO 7870-1:2019*, n.d.). Often, these applications involve small proportions of the data by periodical samples taken or extracted from the process (Sousa et al., 2017, p. 1220).

Therefore, the more and more accessible sensor-based measurement solutions with potentially high data frequency represent significant changes in industrial practices. What it provides is often called Big Data (Thudumu et al., 2020). Such data has special characteristics, and it is often multivariate. This new information source generated new challenges. For instance, the special time series data required new analytical methods, the amount of information resulted long computational time, but so large data sets often did not fit into a common computer's memory.

The new challenges have been addressed. Currently, efficient algorithms are available to handle high amount of multivariate data. Many industrial objectives can be fulfilled without the need for cluster computers. Though, such high performing infrastructure can fulfil the most demanding current and future needs. All in all, the latest tools and methods enable a lot of new opportunities while completing deviation or – otherwise – anomaly detection.

All aspects of the current short review are relevant in this thesis that aims to demonstrate the power of some analytical tools and methods published recently. They are applied on the process data of a single compressor operating automatically on Aker BP's Valhall oil platform in the North Sea (*Open Industrial Data*, n.d.). Its data – collected by hundreds of sensors – is shared from 2013 onwards by the Open Industrial Data project "*to accelerate innovation within data-heavy fields. This includes predictive maintenance, condition monitoring, and advanced visualization techniques* …".

The thesis has a structure as follows. After the Introduction, the Literature Review gives insights and critical review of the previous research. The Data chapter provides a detailed description of the industrial process and its data. It also explains the relevant engineering background information necessary for building suitable models and justifies the data subset identified. The Methods chapter reviews the main analytical methods in detail covering their mathematical and practical aspects. The Results chapter introduces all analytical outcomes realized during any steps of the analytical workflow. It explains how steps were done and what decisions were made. In the end, the Discussion chapter answers the research questions, describes the conclusions, and draws attention on the limitations of the assessment.

The choice of data source for the thesis supports to fulfil a university requirement. The data analysed must be found nowhere else considering publications or the internet. Data from Open Industrial Data appears in only one publication as per Google Scholar using the key word 'openindustrialdata.com'. That is also a thesis targeting very different analytical objectives, and it was published in 2019 (Reiten & Kleiven, 2019). Therefore, the data used in here from 2021 does meet the university requirements. It never was analysed in the literature.

The Open Industrial Data project offers a perfect opportunity from other viewpoints, too. Other industries' multivariate time series data analysis for an automatic equipment or process could follow a similar analytical workflow. Examples of domains could be processing, automotive, aerospace, chemical, or pharmaceutical. This data is not confidential, though. Instead, it is shared with the public together with technical details and meaningful support.

2 Literature Review

The literature review has sections that represent important themes. Some of the conclusions regarding what approach to prefer in this paper is shared in the Methods chapter. Considering algorithms in particular, aspects of applications and practicalities are discussed only in this chapter. Detailed introductions of the important algorithms are available in the Methods chapter.

2.1 Anomaly Detection

Past, Present, and Future

Alimohammadi and Nancy Chen (2022, p. 2) give definitions of anomalies as outliers first in the form of "single observations that stand out compared to surrounding points", then as "collection of points that are anomalous to the entire dataset", and finally as "contextual outliers … abnormal only in a specific context". These definitions are supported by a number of publications. Availability of these and other publications on anomaly detection, though, have not been distributed evenly over time. Nunes (2021, p. 9) – reviewing anomalous sound detection research – found no publication earlier than 2014, but saw exponential growth since then. Lei et al. (2020, p. 3) found the number of relevant articles for their machine fault diagnostic topic till 2017 altogether equal with the number of articles from 2017 to 2019. Nti et al. (2021, p. 1) worked on a little more general approach of artificial intelligence (AI) in engineering and manufacturing. They also found recent increase in count of publications. These systematic reviews have different acceptance criteria. Still, they commonly reveal a trend of growing interest on this research field. They also clarify that some of the specific research fields – like anomalous sound detection – does not really have a past, but they seem to have an interesting future.

Future seems to bring not only new fields of applications. Challenges (Nti et al., 2021, p. 2) may come because of diversity of AI techniques in engineering and manufacturing that may hinder the utilization of AI. New practices are being developed for the more established areas (Lei et al., 2020, p. 21). The so-called transfer learning can be one of the concepts. The problem it handles is that the result of analytics can be expected to be reused from one or multiple diagnosis tasks to other related but different ones. IT infrastructure as computing resources is changing to become on-demand cloud services (Nti et al., 2021, p. 16). This accelerates GPU based AI algorithm implementations. An example of this can be the Rapids AI cuML library (Mitra et al., 2020, p. 7).

Algorithms

Lei et al. (2020, p. 10) give the summary of algorithms used in anomaly detections as Artificial Neural Network (ANN), many different Support Vector Machine (SVM) implementations, k Nearest Neighbours (KNN), and Probabilistic Graphical Model (PGM). They emphasize the tendency of moving towards ANN. Nassif et al. (2021, p. 78664) maintain a richer list, but that reflects to a 20-year long history. They add the cluster, and One-Class SVM algorithms as some of the few top performing algorithms. Nti et al. (2021, p. 12) reenforce

ANN, and SVM. They also include Decision Tree (DT), and Random Forest (RF). They report cases where SVM is superior to ANN significantly. Autoencoder (AE), Convolutional Neural Network (CNN) – a kind of an ANN -, and very recent transfer learning approaches are also highlighted (Nunes, 2021, p. 10). 1D and 2D CNN models are repeated by Mitra et al. (2020, p. 2). Meng et al. (2020, p. 2367) add PCA to the list while concluding, though, that supervised methods are more useful in a technical environment as the typical aim is to predict a target parameter or class. They have a less usual finding with the Gaussian Process (GP) used within a low frequency operation. Overall, Meng et al. admit that CNN is the most popular algorithm and SVM can be a great – sometimes better – alternative. They also notify that not only CNN, but also SVM is recommended for image or acoustic emission-based problems.

Synthesizing the above, there are specific applications and algorithms, and there are more general ones. The view of Lei et al. (2020, p. 10) has been confirmed that SVM is particularly efficient in health state recognition of machines with rolling elements. However, it may not be efficient for the amount of data sensors can generate (Mitra et al., 2020, p. 6). In that case, CNN is considered as a better choice.

Data and Metrics

The data analysed while anomaly detection is seriously imbalanced (Lei et al., 2020, p. 20). It means the number of normal data points are a lot more than the number of anomalous data points (Alimohammadi & Nancy Chen, 2022, p. 2). It is advisable to form a suitable strategy for the analysis of an imbalanced data set (Wallace & Dahabreh, 2012, p. 695).

It is a common assumption that data is labelled for an analysis. However, such assumption is not realistic in engineering scenarios because of technical reasons. First, machines typically work in a good condition, and anomalies occur occasionally. Consequently, the machines' process data reflect the same. Second, it is a significant cost to arrange any labelling procedure. There are unsupervised methods that can help in situations like this (Alimohammadi & Nancy Chen, 2022, p. 3).

Suitable metric is advised by many sources. Alimohammadi and Nancy Chen (2022, p. 4) recommend accuracy, precision, recall, F1 score, Cohen's Kappa, and confusion matrix. Mitra et al. (2020, p. 5) mention Matthews Correlation Coefficient (MCC) in addition. Though Teh et al. (2020, p. 45) highlights MCC is not popular as it is used in one paper of 57 selected articles in their systematic review. On the contrary, Nunes (2021, p. 9) reports about 75% of studies she reviews using AUC-ROC and F1 score. Also, Meng et al. (2020, p. 2371) consider accuracy as not appropriate for imbalanced data.

Even if the analyst applies a suitable metric, it cannot guarantee the good outcome. Data quality plays an even more important role in that. Sensor data error may be present in the form of bias, drift, noise, constant value, stuck-at-zero, missing values, duplicates and uncertainty. There are three possible process areas where it can occur: perception phase when the sensor measures, processing phase when the data flows from the sensor to the application using it, application phase when the data gets stored and used (H. Y. Teh et al., 2020, p. 11).

Correction of sensor data errors is challenging, but it is not impossible. Bias and drift often require sensor calibration. Constant value and stuck-at-zero errors are typical signs of dead sensors, and so maintenance is required. Missing value error type may be corrected by imputing synthetic data. There are multiple ways for that. For instance, the MICE R package (Wu et al., 2022, p. 10672), and the cubic spline interpolation method using R or Python may be useful for that (Haslwanter, 2021, pp. 118–119).

2.2 Changepoint Detection

Engineering sciences motivated the research on changepoint detection methods (Ramsay & Chenouri, 2021, p. 1). Since then, very different other domains have also been

applying these methods to their problems such as natural sciences and financial management to name a few.

According to the literature, a changepoint itself can be described as "moments of abrupt change in the behaviour of a time series" (Burg & Williams, 2020, p. 1). Detecting changepoints results three important types of outputs: the number of changepoints, the changepoints locations, and the segments' statistical parameters (Wilms et al., 2021, p. 9). In industry, large proportion of equipment have constant operating conditions over time. In such cases, industrial processes can be considered non-anomalous if their measured output values have constant means and variances (*ISO 7870-1:2019*, n.d.).

On high level, changepoint detection methods may be categorized as online versus offline, univariate versus multivariate, or model-based versus nonparametric (Burg & Williams, 2020, p. 3). Offline methods often outperform the online methods. Some of the model-based solutions can support autocorrelated data analysis, too, which is very useful for sensor data. Amongst the offline frequentist approaches, Binary Segmentation, Pruned Exact Linear Time, At Most One Change, and Segment Neighborhoods methods are the more recommended ones. Implementation in Python is available for some, but the R statistical language may have more to offer.

2.3 Consequences of Big Data

The five important characteristics of Big Data are value, veracity, variety, velocity, and volume (Moyne & Iskandar, 2017, p. 3). Value is meant as benefit of data analysis; veracity is related to accuracy of data; variety is about different types and kinds of data; velocity indicates measurement frequency, volume is size of the data. These are known as the 5Vs.

Volume of data may be considered from multiple viewpoints. In case of Big Data, high volume often results high dimensionality. This means high number of features, and as it increases it can result data sparsity. It is also called 'curse of dimensionality' when data points are scattered and get very isolated. It may result greater type I error rate. For instance, classification, distance and density based anomaly detection algorithms are a lot less effective with such data (Thudumu et al., 2020, p. 11).

Reducing dimensionality is a possible option. PCA is one of the methods that can be used for this. Nevertheless, the amount of data makes it difficult to use PCA because of its computational cost. The standard PCA uses all the data at the same time. Incremental PCA is different in how data is utilized. Observations are involved into the calculation one by one (Lippi & Ceccarelli, 2019, p. 475). Other methods address different problems. The high volume of data makes it more probable that outliers are present. Standard PCA is sensitive for outliers. The Robust PCA (Fan et al., 2018, p. 4), can solve issues like that. Regardless of all its strengths, PCA sometimes worsens models (Meng et al., 2020, p. 2375). Other dimensionality reduction methods, or direct feature selection may help in such cases.

High volume data also means high number of data points. Many standard algorithms do not scale well, because their runtime increase in a non-linear higher order way. Solutions are available for faster computation, for instance, Stochastic Gradient Descent (SGD) is one of the well-known methods. Bottou advises (2012, p. 4) to use it if the model training time is a bottleneck. Algorithms optimized to run on GPU processors speed up computation in a different way, they enable parallel computations. The Rgtsvm R package (Wang et al., 2017), and Rapid AI cuML Python library (Mitra et al., 2020, p. 7) could be good examples of those. Finally, there are distributed data processing engines developed for this purpose. Spark (Thudumu et al., 2020, p. 21), Dask (*Scikit-Learn & Joblib with Dask*, n.d.), and Terality (*What Is Terality?*, n.d.) are potential alternatives.

2.4 Research Questions

As introduced earlier, time series analytics methods often utilize mean and variance of data to identify condition change of processes and equipment in operations. In this paper, a recently published computationally efficient nonparametric changepoint detection method is utilized for this purpose. As proposed in the literature (Amiri & Allahyari, 2012, p. 674; Arellano et al., 2021, p. 3), it is explored whether such method can provide valuable support for the analysis. Therefore, the first research question is:

RQ1: Can the nonparametric changepoint detection method of Haynes et al. (2017) recognize anomalies within the univariate output process data of the Valhall oil platform's compressor?

Lai et al. (2021) shared their benchmarking results of applying a number of anomaly detection methods on synthetic and real data sets. One of the best performing methods was the standard OCSVM (Schölkopf et al., 1999). Their real data sets contained records up till 284,000 data points with 79 dimensions. The smallest outlier ratio in the data sets was 0.173%. Their computational resources contained 3 servers with 48 CPUs in each that had memory ranging from 64GB to 188GB. On the other hand, the Scikit Learn Team's SGD based OCSVM implementation aims to accelerate the runtime of the standard OCSVM algorithm within a common desktop environment (*Scikit-Learn Release Note v1.0.0*, 2021). Both publications happened in the same year. This raises the second research question:

RQ2: Is it realistic to expect that industrial professionals using desktop environments with common computing power can efficiently utilize OCSVM models on data sets with 200,000 or more multivariate records?

3 Data

This chapter introduces where the data comes from, what interesting or important properties it has, and how the time interval of the process data was identified for the analysis. The measures have an interesting general feature: the data platform uses non-SI symbols for some units, i.e. pressure symbol is barg and degree Celsius symbol is degC.

3.1 The Process and its Output Variable

According to a process overview (*Open Industrial Data,* n.d.), the first stage compressor (23-KA-9101) is an electrically-driven, fixed-speed centrifugal compressor. It receives gas from the separators through the upstream valves (23-ESDV-92501A/B) at 3 barg pressure. Before reaching the compressor, the gas is cooled in the first stage suction cooler (23-HA-9103) which is a heat exchanger. The cooled gas flows first through the suction throttle valve (23-PV-92538) then into the first stage suction scrubber (23-VG-9101). The latter is to remove liquid droplets. Then the gas is compressed to approximately 12 barg. Eventually, the compressed gas flows through a pair of discharge coolers (23-HA-9114/9115) and leaves the process via the downstream valves (23-ESDV-92551A/B). The anti-surge valve (23-FV-92543) supports the control of the process (Figure 3.1). Gas flows from *right to left* except for the anti-surge valve. There the gas flow from *left to right*.

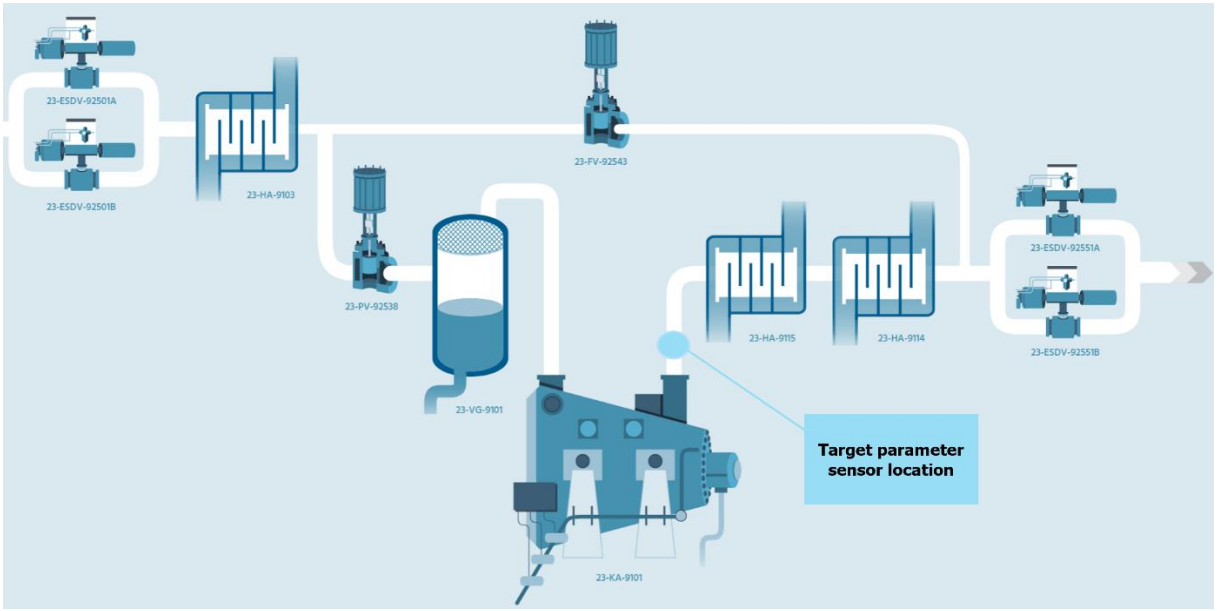


Figure 3.1: Graphical description of the process and its main elements: upstream valves 23-ESDV-92501A/B, suction cooler 23-HA-9103, suction throttle valve 23-PV-92538, suction scrubber 23-VG-9101, compressor 23-KA-9101, discharge coolers 23-HA-9114/9115, downstream valves 23-ESDV-92551A/B, anti-surge valve 23-FV-92543, and the location of target parameter sensor (Open Industrial Data, n.d.).

The output parameter is the Compressor Discharge Mass Flow (CDM Flow), and its measurement unit is kg/h. This is one of the three parameters that are measured when the compressed gas leaves the compressor itself. The other two is the Compressor Discharge Temperature with measurement unit degC, and the Compressor Discharge Pressure with measurement unit barg. These parameters are important because they fit into the Ideal Gas Law

$$\mathrm{PV} = \mathrm{nRT}$$

where n is the number of moles, R is the universal gas constant, T is the temperature, P is the pressure, and V is the volume. Using the equation differently confirms that CDM Flow shall be constant unless the operating conditions change as

$$\frac{\mathrm{PV}}{T} = \mathrm{nR}$$

and this conclusion is shared by researchers (Perez, 2022, p. 209). The equations indicate as well that the three parameters are dependent on each other. Therefore, it is important to clarify that while CDM Flow is used as an output variable, the other two variables mentioned above are excluded completely from the analysis.

The API – through which the data is accessible using Python – is rich in functionalities. Data can be downloaded in row format and many other aggregated or pre-processed format, too. The first task was to find the right time interval that provides suitable data based on the output variable timer series. Row data was partitioned while downloading from 2015 onwards. One chunk covered 100 days of operation, and data was saved in binary format to keep files sizes low (Haslwanter, 2021, p. 47). Looking at the number of records in the chunks, a pattern can be seen (Figure 3.2 – Left). The charts indicate that the measurement system did not remain the same over time. Less data per time unit is collected in the last period where the typical time gap between time stamps is 10 seconds instead of 1 second prior to that (Figure 3.2 – Right).

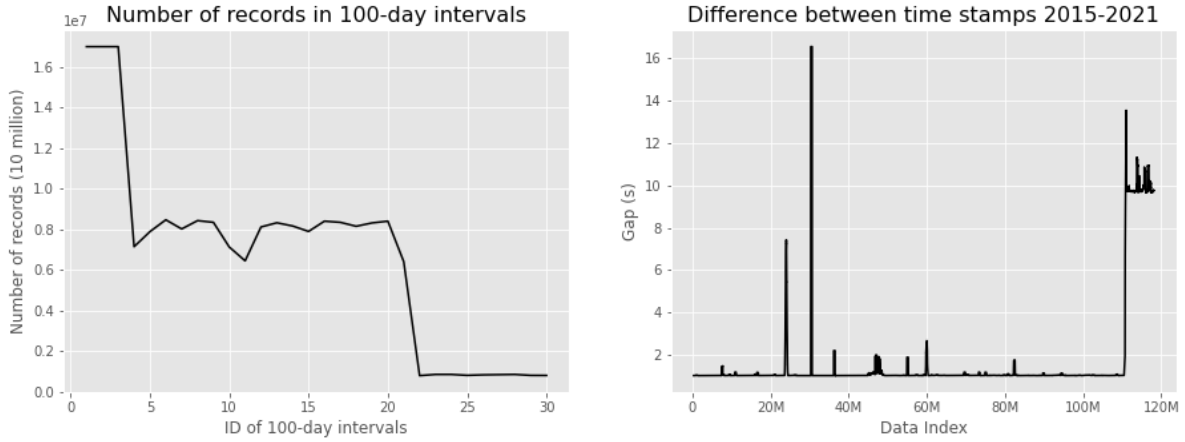


Figure 3.2: Basic charts of the target variable; on the left, the number of records can be seen within each 100-day long time series chunk with starting values of around 16 million and closing values of roughly 80 thousand; on the right, time difference between time stamps in seconds can be seen confirming a sudden large increase approximately for the last 5 million values.

This is normal to see changes in technology during a longer period as per the author's experience. Continuous improvement as an industrial principle is important, and it is taken seriously in matured organizations. It is interesting, however, that collecting lower number of data is the improvement in this case. No technical information is available to justify the assumption, though. Figure 3.3 shows the complete time series raw data with approximately 120 million data points.

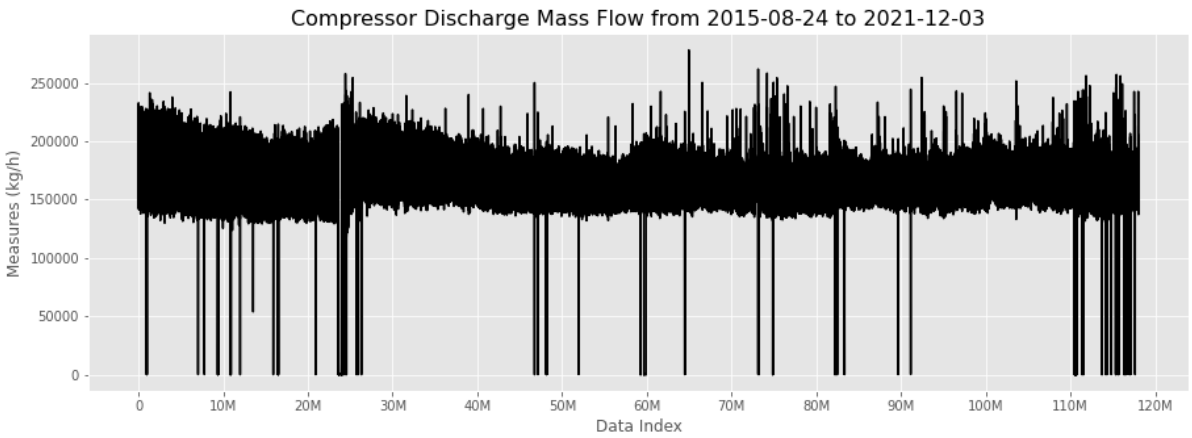


Figure 3.3: Time series of Compressor Discharge Mass Flow (CDM Flow) from 2015 to 2021; changes of the process output variable are visually recognizable by more and less out of range values and changing range over time; zero flow rates indicate the process stopped working many times; the time series consists of almost 120 million data points.

Considering what RQ2 requires to get it answered, a subset of the initial raw data would be suitable to keep. Oriented by the pattern highlighted above, a visual time series review resulted an arbitrary choice. Around 290,000 data points from 15 June 2021 onwards were taken further for pre-processing. This corresponds to the data index 116612968 and above in the initial raw data. Values in this subset are not in row format, but in a 1-minute mean aggregation format (Figure 3.4).

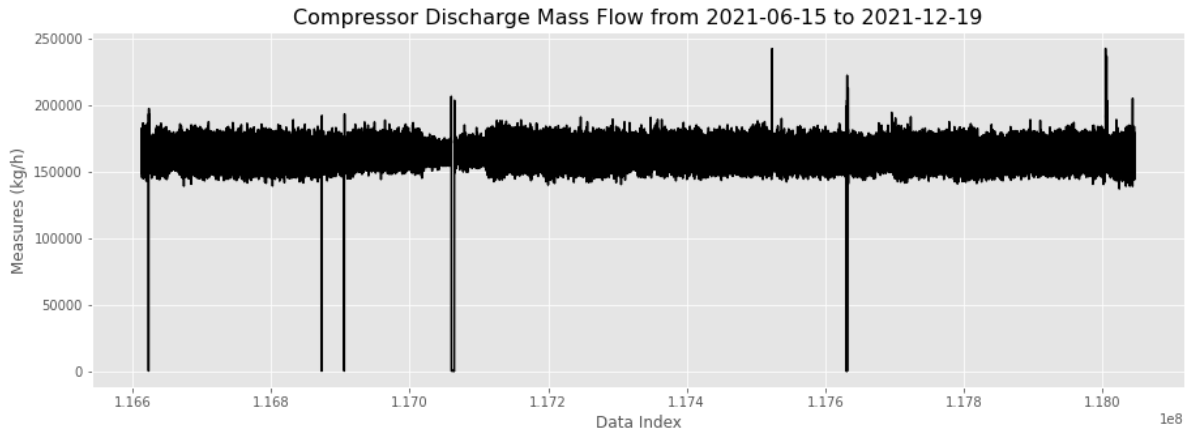


Figure 3.4: The time series chart shows the output variable data within the time interval identified; the time series consists of around 290,000 values; data points are in 1-minute mean aggregation format.

3.2 The Input Variables

Analysing the technical details of the process, the variables measured physically earlier than the target variable are considered to be part of the input variables' set. Initially, it meant 146 variables (Appendix A). Reviewing all revealed that 44 of them had no data at all for the period analysed. Many of the remaining variables had constant value that could be either 0 or other value with 0 variations. All these were excluded including 9 more after visual evaluation. 41 input variables remained in the final set to take further for pre-processing. The values in these series are also in 1-minute mean aggregation format to be aligned with the target variable.

Final input variables and some descriptive information

Code	ID	Data Count	Mean	STD	Min	Max
X1	pi:160627	128812	0.00834	0.18825	-0.10374	5.95834
X2	pi:160031	50744	4.25747	0.81871	0.0000	15.0000
X3	pi:160030	285722	36.88313	6.73907	15.47186	100.0000
X4	pi:160182	292175	97028.21579	15375.94054	0.0000	279032.56264
X5	pi:160261	107767	63.42946	1.72646	24.77438	68.76824
X6	pi:160068	287202	9147.71698	654.03	2.9304	11877.55817
X7	pi:160500	285548	32.04069	8.41145	0.0000	98.24716
X8	pi:160696	294238	3.10469	0.35272	0.0000	8.70928
X9	pi:160882	294809	43.29709	4.59261	7.27743	50.29299
X10	pi:160629	242228	0.25277	0.03047	0.00185	1.17302
X11	pi:160220	291243	9117.9395	401.11518	-34.86487	12679.02977
X12	pi:160520	294459	39.6407	2.47298	11.53804	52.34468
X13	pi:160140	173198	92.18307	8.30527	29.4796	99.97558
X14	pi:160628	291721	0.26268	0.04214	0.0000	0.6927
X15	pi:160995	293747	0.63658	0.01402	0.0000	2.91939
X16	pi:160697	293350	3.19359	0.2716	0.30273	8.71883
X17	pi:160499	294821	40.63106	2.51685	10.33913	53.93948
X18	pi:160141	274634	36.69462	7.13199	15.2614	99.90232
X19	pi:160262	49691	67.73344	2.7937	24.92425	71.37734
X20	pi:160883	290800	46.71633	4.15123	8.86453	55.17048

Code	ID	Data Count	Mean	STD	Min	Max
X21	pi:160519	294704	0.00834	0.18825	-0.10374	5.95834
X22	pi:160884	283912	39.22229	2.65194	8.45567	52.63821
X23	pi:160254	68031	33.80613	1.88522	14.19621	46.79024
X24	pi:160236	126349	4559.92633	448.05394	0.0000	47218.30417
X25	pi:160237	126827	535.54865	32.84149	0.0000	2616.11745
X26	pi:160701	292805	535.74039	32.81469	0.0000	2598.42401
X27	pi:160263	63883	2.70344	0.20068	-0.01339	6.64392
X28	pi:160183	289346	65.64662	2.39523	24.69937	69.19506
X29	pi:160698	294410	4.13759	1.098	0.00115	11.53224
X30	pi:160631	291569	2.88161	0.23774	0.01105	6.70062
X31	pi:160781	294815	107.91313	5.52482	0.43409	184.91662
X32	pi:160224	290859	39.37287	3.54944	3.95357	50.14939
X33	pi:160246	29417	8146.87371	260.67616	6909.92796	11506.15804
X34	pi:160267	291579	438.94035	1.92848	435.0000	441.0000
X35	pi:160700	292699	9245.73742	397.4178	9.24281	11957.60597
X36	pi:160104	288060	2.70792	0.1964	0.0000	6.65742
X37	pi:160235	127109	2.70917	0.23409	0.0000	6.65291
X38	pi:160110	283352	535.24242	32.73171	0.0000	2607.5827
X39	pi:160630	291809	33.64041	2.19362	13.88034	47.02292
X40	pi:160231	196535	114.62858	6.75805	0.0000	197.5552
X41	pi:160699	292767	9161.36018	443.24914	0.0000	11615.01361

4 Methods

The method chapter describes the more important tools and techniques that were used during the analysis.

4.1 Analytics Workflow and Data Pre-Processing

The analytics workflow followed in this paper is outlined in Figure 4.1. It is an iterative process since there are loops turning back to earlier steps. For instance, having collected and

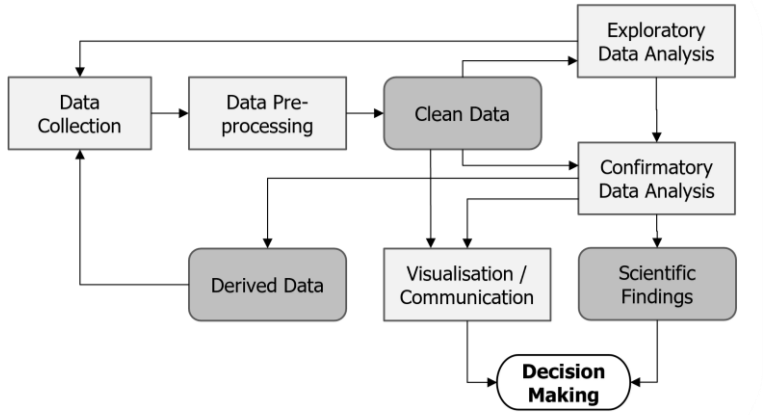


Figure 4.1: Data analytics workflow adapted from Ma et al. (2017, p. 2).

cleaned the data, the exploratory data analysis (EDA) may result the conclusion to collect more data. The same is applicable for the confirmatory data analysis.

However, while pre-processing the entire data set i.e. prior to any train-test split and confirmatory analysis, recognizing missing data and finding the right approach to handle them was important. The column of Data Count in the previous chapter's final input variable list confirms that it was a real issue. One of the interesting items is variable X33 which has 29417 data values whereas it could have above 290,000. It is shown in Results chapter that such sensors have no major issues, though. They just collect data with one tenth frequency compared to some others. Evaluating randomness in the appearance of missing data helps to decide if synthetic data imputation may happen or not, and if it can, then how. Wu et al. describe this as a suitable approach even for greater missing gaps (2022, p. 10681). This may happen by contrasting the frequency of missing values in sections of a time series with the mean or variance of those sections. Such analysis may involve non-linear correlation methods like for instance Kendall's rank correlation. In this case, the correlation coefficient is calculated as

$$\tau(X, Y) = \frac{\{(\text{number of concordant pairs}) - (\text{number of discordant pairs})\}}{0.5n(n-1)}$$

where concordant pair is when $x_i > x_j$ and $y_i > y_j$, or $x_i < x_j$ and $y_i < y_j$; pairs are discordant when $x_i < x_j$ and $y_i > y_j$, or $x_i > x_j$ and $y_i < y_j$. If $x_i = x_j$ or $y_i = y_j$, the pair is neither concordant nor discordant (de Siqueira Santos et al., 2014, p. 3). τ may take values between -1 and +1. The lower value indicates perfect monotonically decreasing relationship, and the other one indicates perfect monotonically increasing relationship.

Cubic spline interpolation may be used for missing data imputation in case of offline sensor data analysis (Haslwanter, 2021, pp. 118–119) when all data points represent past events. The spline function with fixed knot K and fixed degree d can be written as

$$f(X) = \sum_{k=1}^{K+d+1} \beta_k B_k(X)$$

where B_k represent some basis functions and the β_k are the spline coefficients. Utilizing this for the cubic spline function, it can be written as

$$f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 (X - \tau_1)^3 + \beta_5 (X - \tau_2)^3 + \beta_6 (X - \tau_3)^3,$$

where τ_1, τ_2, and τ_3 are the knots (Perperoglou et al., 2019, pp. 4–5). This function then may be used to interpolate having at least four data points. Two data points – x_{i+1}, and x_{i+3} as in Figure 4.2 – would be the ones missing a third one x_{i+2} in between that needs replacement with synthetic data. Another two points – x_i, and x_{i+4} – shall be from the neighbouring points.

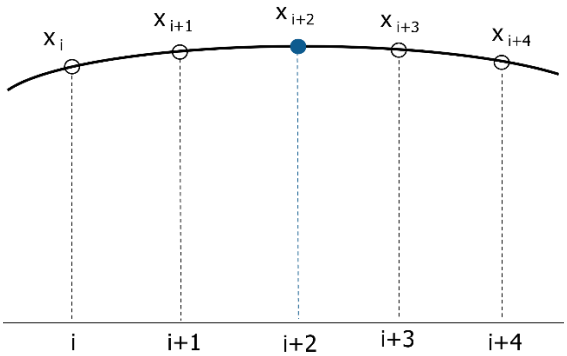


Figure 4.2: Data points on a cubic spline interpolation (Usman & Ramdhani, 2019, p. 30)

The R statistical programming language and its spline() function was used to achieve this (Torres-Reyna, 2014) while pre-processing the entire data set.

4.2 Output Variable Labelling

It has been discussed that it is a common industrial practice not to have output variable data labels available. The compressor analysed in this thesis is no different. Therefore, it is an important task to identify the anomalies and label the data accordingly before modelling.

The compressor analysed operates with fixed speed based on the technical background information. Thus, it may be assumed it provides a constant mean and variance of Compressor Discharge Mass Flow (CDM Flow) rate. Visual data review has confirmed it. Anomaly may then be defined for the current analysis: any single or consecutive data is an anomaly which represent a change while running operation in the CDM Flow rate compared to the reference. The reference value is to be derived from the data. One potentially identifiable example for a deviation is shown in Figure 4.3. Considering this, the definition focuses on the correct process characteristics to evaluate.

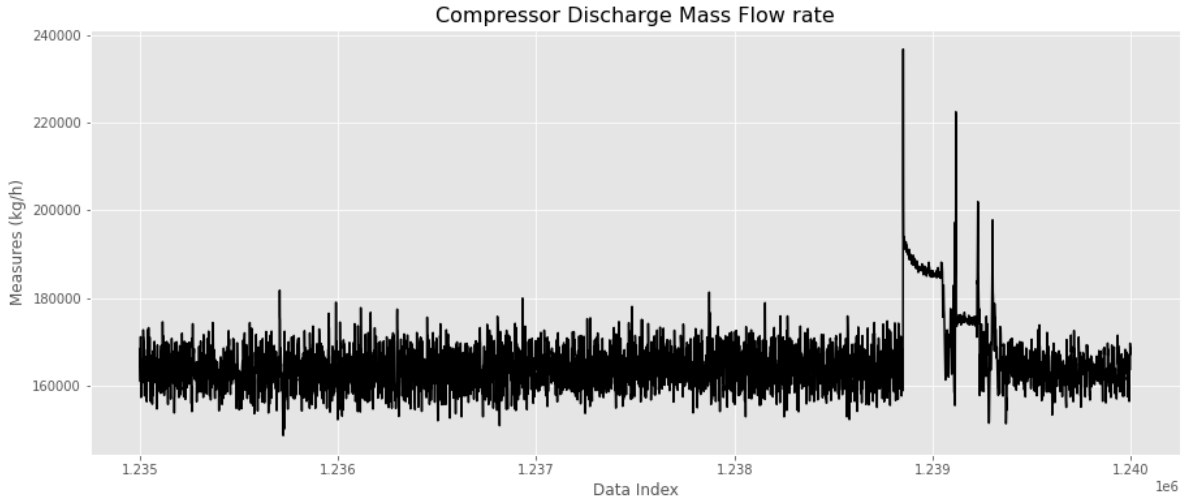


Figure 4.3: The image shows a potentially identifiable anomaly

Univariate changepoint detection methods are designed to support such aim. Several methods were reviewed during the Literature Review. A PELT based algorithm is chosen of those to support the work. PELT is one of the recommended frequentist methods and it is very accessible. Several journal articles explain all the theoretical and practical aspects of the method. Further on, it is trained in virtual sessions for the NHS in the UK. Their recordings are freely available online (*Introduction to Changepoint Analysis by Rebecca Killick*, n.d.; *Further Changepoint Analysis by Rebecca Killick*, n.d.).

Killick et al. describe that "changepoint analysis can, loosely speaking, be considered to be the identification of points within a data set where the statistical properties change" (2012, p. 4). Consequently, changepoints break the time series into segments, and the minimum segment length can be one data point. Killick et al. argue that a model may be applied for the data in a segment using minus the maximum of the resulting log-likelihood to calculate a cost for the segment. Then the cost of the segmentation can be calculated as sum of each segment's cost. Their method applies a penalty that depends on the number of segments found while optimizing the cost function and finding its global minimum (2012, p. 10). The penalty helps to avoid over- and underfitting. When the global minimum is found, the set of true changepoints is found, too. The approach assumes that the data in a segment is normally distributed, and a segment length is independent of the segment's model parameters (2012, p. 11). It is implemented as an R package (Killick & Eckley, 2014).

However, industrial sensor data is typically autocorrelated. This is true for the current compressor data as well. It is confirmed in the Results chapter in detail. Therefore, a nonparametric version of the PELT method is utilized in this thesis. It maps the empirical

distribution of the data to its empirical quantiles, hence its name ED-PELT. Haynes et al. explain (2017, p. 1296) that it can be achieved with a discrete approximation of sum of $K<<n$ terms; n is the length of data. The cost function is as follows

$$C_K(x_{u:v}) = \frac{-2c}{K} \sum_{k=1}^{K} L_{np}(x_{u:v}|t_k)$$

where c = – log (2n – 1) and each L_{np} term is a log-likelihood of a subset of $x_{u:v}$ where the subset is defined by t_k. t_k is an empirical quantile of the data calculated as

$$(1 + (2n-1)\exp\{\frac{c}{K}(2k-1)\})^{-1}$$

which – when used in the algorithm – maps unevenly spaced data to intervals of equal probabilities, thus giving more weight to the tail of the distribution. The cost function is utilized in the objective function:

$$Q_{PELT}(x_{1:v}|\xi_n) = \min_{u<v}(Q_{PELT}(x_{1:u}) + C_K(x_{u+1:v}) + \xi_n) .$$

In this, the PELT algorithm controls the potential new u changepoints by excluding those that never can be optimal locations. This is the feature of PELT that makes it more efficient computationally than many other algorithms. Finally, ξ_n is a single penalty value on which the optimization process depends. Since only one penalty value is included, only one changepoint set can be delivered. However, this may not be the true changepoint set.

Haynes et al. (2014, p. 7) propose a method – called Changepoints over a Range Of PenaltieS (CROPS) – to solve this problem by evaluating a range of penalty values. It results a changepoint set for each penalty value. Eventually, the sets can be compared to each other by their costs versus number of changepoints, and a chart based graphical evaluation method drives the conclusion. Further practical details are provided in the Results chapter. The nonparametric PELT and the CROPS methods are implemented within the *changepoint.np* R library, and this is used in the current analysis.

4.3 Multivariate Modelling

As outlined in the Research Questions subsection, the main modelling method is the SGD based OCSVM method. Its implementation by the Scikit-Learn development team was released in September 2021 (*Scikit-Learn Release Note v1.0.0*, 2021). First the non-linear binary SVM method is introduced, and then the one-class version is derived from that. In the end, the way how the SGD optimization algorithm supports the method is explained.

Non-Linear Binary SVM

Amongst others, Kutz and Brunton explain the non-linear binary SVM clearly (2019, pp. 182–185) based on the research of Cortes and Vapnik (1995).

The SVM algorithm is designed to find a hyperplane that separates the two classes of binary data from each other. The classification function is

$$f(x) = sign(w\,\phi(X) + b) \qquad \text{and} \quad y_j \in \{\pm 1\}$$

where $\phi(X)$ is the non-linear observable, w and b are constants, and w and b parameterize the hyperplane. The function produces a binary output vector. Two types of loss functions may be applied to provide the score for the evaluation. One is

$$l(y_j,\ \bar{y}_j) = \begin{cases} 0\ if\ data\ is\ correctly\ labeled \\ +1\ if\ data\ is\ incorectly\ labeled \end{cases}'$$

and the training error calculation is the sum of loss functions. The so-called Hinge loss could be another one that measures the confidence in the classification result in the form of the distance from the hyperplane. The Hinge loss is

$$H(z_i) = \max(0, 1 - z_i) \text{ and}$$
$$z_i = y_i w^T x_i\ .$$

In addition to minimizing the loss function, the width around the hyperplane with no data points inside is to be maximized. The area is called margin and it represents the distance between the two groups of data points. Putting this all together leads to an objective function

$$\min_w \frac{1}{2}\|w\|^2 + C\ \sum_{i=1}^{m} \max(0, 1 - y_i w^T x_i)$$
$$\text{subject to} \quad \min_j |x_j \cdot w| = 1$$

where C is a parameter set to manually control the number of misclassifications during the training process, and w is the margin.

Non-Linear One-Class SVM

The standard one-class SVM uses a slightly different approach. It separates the data points from the origin and maximizes the distance from this hyperplane to the origin. Schölkopf et al. (1999) published the method. The objective function is similar to the one above:

$$\min_w \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{m} \max(0, 1 - y_i w^T x_i) - \rho$$
$$\text{subject to} \quad \min_j |x_j \cdot w| = 1.$$

In this formula, ν may be finetuned manually by the user. It has two types of impacts on the results. It controls the proportion of the outliers that are in the training sample but considered out-of-class. It also controls the number of Support Vectors that are the data points utilized in defining the hyperplane. The parameters of the hyperplane are w and ρ.

Calculation with w is expensive. Having high number of features in the data may lead to the curse of dimensionality. This is when data become sparse due to the high volume. The kernel trick solves this problem. Parameter w is turned into a different form

$$w = \sum_{j=1}^{m} \alpha_j\ \Phi(x_j).$$

With this, the objective function becomes

$$\min_w \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{m} \alpha_i \mathrm{K}(x, x_i) - \rho$$
$$\text{subject to} \quad \min_j |x_j w| = 1$$

and $K(x, x_i)$ is the kernel function. The kernel function enables to plug-in many different types of functions. One of the common functions is the radial basis function (RBF):

$$K(x_j, x) = \exp\left(-\gamma \|x_j - x\|^2\right).$$

It is included into the implementation of the SGD based SVM as well. Interestingly, this is the only kernel function it offers for use at the time of writing the paper.

Optimization with Stochastic Gradient Descent

This is a general introduction how SGD algorithm can support optimization of a loss function that is applicable for the SGD based SVM implementation released by the Scikit-Learn Development Team. It is general because explicit description of the implementation could not be found. Nevertheless, several indications suggest that the theoretical foundations of Bottou (2012) was utilized for the implementation.

Ideally, loss functions are convex and have global minimum points where the slope – otherwise the gradient – of a function is zero. SGD is designed to find this global minimum or a point very near to that. This happens by following an iterative process in which only one randomly chosen data point is used for calculation during an iteration. The optimization algorithm applied is

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t)$$

where *Q(z, w)* is the loss function with z feature value set and w weight vector; ∇ indicates the partial derivatives of the loss function dimensions, and so the gradient vector; γ is the learning rate.

The greatest advantage of SGD is its speed when training a model. Other algorithms may be more precise, but they are usually a lot slower. There are some practicalities, though, that are useful to take into account. There is a risk that the algorithm gets stuck in a local minima, thus it cannot find the global minima. This risk may be significantly reduced by shuffling the training data. Normalizing the inputs is critical to support convergence. Multicollinearity may also undermine the SGD performance. To prevent that, dimensionality reduction may happen in many form, PCA or SVM-ANOVA could be two of those. Follow-up and evaluation of training process is advisable by focusing on the training cost metric and the validation error.

4.4 Dimensionality Reduction

Dimensionality reduction may be achieved in different ways depending on certain circumstances. Its main objective is to reduce noise and exclude redundant information from the model. This may result a smaller data set that is faster to process computationally. This also supports the performances of certain algorithms. SVM is known to be sensitive to noise (Zhang et al., 2016, p. 2), and SGD has difficulty with it, too (Montavon et al., 2012, p. 17).

SVM-ANOVA

The SVM-ANOVA method used in this paper has been implemented by the Scikit-Learn Development Team. It is a simulation algorithm that takes all possible subsets of the input variables from smallest to largest, runs the training algorithm with them, and compares the different models to each other. The best subset of input variables can be identified. It is demonstrated by the Scikit-Learn Team (*SVM-Anova*, n.d.) how the well know Iris data set – combined with 36 random variables – reveals the usefulness of the method. Figure 4.4 shows the final output of the demonstration that describes in a self-explanatory way how decision can be made based on that. The highest accuracy percentile is the best subset. The best model can be built with the variables from that. The rest of the variables can be dropped out. The same approach is followed in the thesis when this technique is applied.

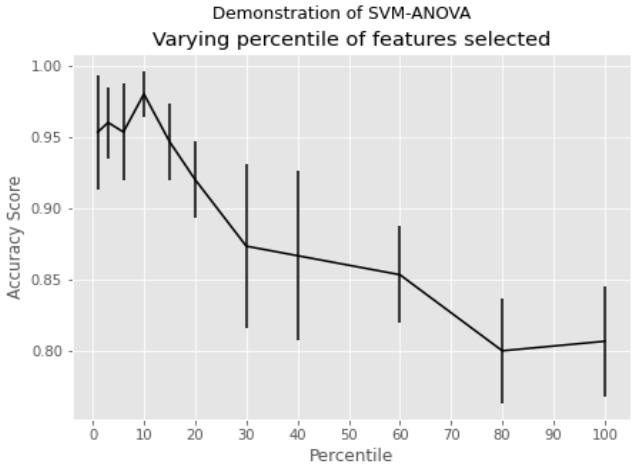


Figure 4.4: Output graph of SVM-ANOVA demonstration using the Iris data set's 4 variables with 36 additional random variables; the output correctly identifies that a 10% feature subset – the 4 non-random variables – gives the best accuracy; the variables of the best subset can be extracted (SVM-Anova, n.d.).

Incremental PCA

The review of incremental PCA algorithm is based on the publication of Lippi and Ceccarelli (2019, pp. 473–475). Their method may include improvements compared to some existing open-source implementations for instance the one used in this analysis. Regardless, their concept is built on the foundation of preceding research results, and their approach is particularly motivated by industrial sensor data. Therefore, it is a suitable basis for a high-level method description.

Starting it with the standard PCA, it is designed to be an off-line analytical method. Its procedure's first step is to create a batch of all data it has to work with. This takes the form of a matrix where x_i is a row vector representing all sensors' measurements at a point in time, and helps to calculate mean and standard deviations for all m number of variables:

$$X_n = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \bar{x}_{n(j)} = \frac{1}{n}\sum_{i=1}^{n} X_{n(j)}, \quad \sigma_{n(j)} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left[X_{n(ij)} - \bar{x}_{n(j)}\right]^2}.$$

This then continues with defining the standardized matrix for the data as

$$Z_n = \begin{bmatrix} x_1 - \bar{x}_n \\ x_2 - \bar{x}_n \\ \vdots \\ x_n - \bar{x}_n \end{bmatrix} {\Sigma_n}^{-1} \quad \text{and} \quad \Sigma_n \equiv diag(\sigma_n) \ .$$

The covariance matrix Q_n for data matrix X_n can then be defined:

$$Q_n = \frac{1}{n-1} Z_n^T Z_n \quad \text{and therefore}$$

$$Q_n = {C_n}^{-1} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} C_n$$

through a standard diagonalization. The positive eigenvalues λ_i are in descending order: $\lambda_i > \lambda_{i+1}$. C_n is the transformation matrix. Its rows are the principal components, and λ_i represents the variance relevant for the i^{th} principal component.

The incremental algorithm aims to take one step further – repeatedly. This is to include one more observation x_{n+1} to calculate Q_{n+1} utilizing Q_n. This may enable either real time analytics or big data analytics with small memory footprint and low computational cost. Inclusion of x_{n+1} starts with summing up the first n observables and their squares:

$$a_{n(j)} = \sum_{i=1}^{n} X_{n(ij)} \text{ , } b_{n(j)} = \sum_{i=1}^{n} X_{n(ij)}^2.$$

The starting mean and standard deviation can be calculated with these since $\bar{x}_n = \frac{a_n}{n}$ and $(n-1)\sigma_n^2 = b_n - na_n^2$. The current mean and standard deviation can also be obtained. These are eventually utilized in the incremental algorithm as

$$Z_{n+1} = \begin{bmatrix} Z_n \Sigma_n + \Delta \\ y \end{bmatrix} {\Sigma_{n+1}}^{-1}$$

where $y = x_{n+1} - \bar{x}_{n+1}$ is a row vector and Δ is a $n \times m$ matrix with n row vector of $\delta = \bar{x}_n - \bar{x}_{n+1}$. From above, it is known that

$$Q_{n+1} = \frac{1}{n}(Z_{n+1}^T Z_{n+1}) \text{ and therefore}$$

$$Q_{n+1} = \frac{1}{n}\left[{\Sigma_{n+1}}^{-1} \Sigma_n Q_n \Sigma_n {\Sigma_{n+1}}^{-1} + n{\Sigma_{n+1}}^{-1} \delta^T \delta {\Sigma_{n+1}}^{-1} + z^T z\right]$$

where $z = y{\Sigma_{n+1}}^{-1}$, and it is assumed that columns of matrix Z_n have zero mean and the columns of matrix Δ have same number.

When using incremental PCA in practice, the same technicalities are applicable as for the batch version. Either the number of principal components or the percentage of variance can be set to define the final output, and it is sensitive to outliers.

5 Result

The chapter's structure follows the analytical workflow. First, data pre-processing results are introduced. Then the output variable labelling univariate process is explained. The next step is the review of the multivariate model development process starting with a baseline model and continuing with step-by-step improvements on that. Finally, some computational cost benchmarking is shared. All analytical tasks are done in a common desktop environment having an Intel Core i5 CPU with 8 cores and 16GB memory.

5.1 Data Pre-Processing

The scope has been defined for the analysis. Data is gathered from 06 June 2021 to 19 December 2021 with one-minute mean aggregation. Rationale has been provided for the time period, but the one-minute aggregation remained unjustified. It actually ensures a synchronized data set. In this way, a suitable observation is available with data in all input variables at each time stamp. This is typically not possible with raw data since sensors may have a little different rhythm for making measurements. Gathering with such aggregated setting is supported by the data platform's API with built-in functions.

First, the output variable is examined and processed. Missing data values were found by the data review. Understanding the nature of the issue was deepened by checking the distribution of the gap size between measurements if there was at least one missing value between them. Figure 5.1 – Top chart provides details, and it can be seen the maximum gap is 9 data points, but the majority is 1 data point. Whether to fill the gap by synthetic data imputation or not should depend on a more detailed analysis. The first question is if the appearances of the gaps are random. The data series was split into 100 equal sections, and a chart (Figure 5.1 Bottom-Left) was prepared to show the number of missing values in all segments along with another chart to show the mean of sensor measurements in those segments (Figure 5.1 Bottom-Right).

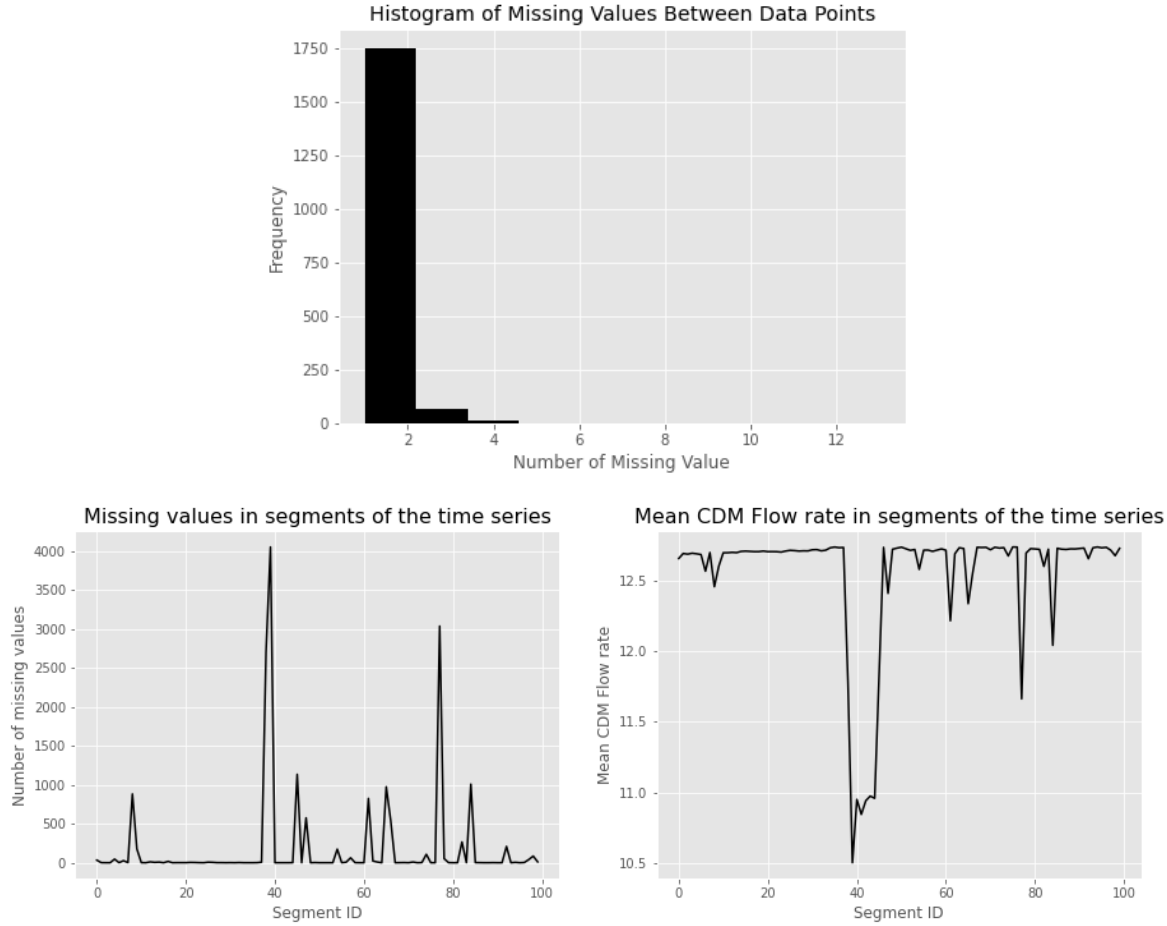


Figure 5.1: **Top** chart shows the number of missing values between consecutive data points if there is any; **Bottom-Left** chart shows the number of missing values in equally sized data segments; the **Bottom-Right** chart shows the mean CDM Flow rate in the same segments that are used on the left side.

The conclusion – based on Figure 5.1 Bottom-Left and Bottom-Right – is that the number of missing values and the segment based mean CDM Flow rate may correlate. The current data as well as another segmentation with 1,000 segments were analysed first with nonparametric methods. In addition, variation of mean CDM Flow rate was also gathered. Eventually, the data with greater segmentation was normalized with Log transformation to allow Pearson correlation as well. Each possible outcome confirmed the correlation, and the most revealing was the correlation of missing values in segments versus variation of mean CDM Flow rate both in 1,000 segments; the Spearman correlation coefficient was 0.73 and the Pearson correlation coefficient was 0.805.

The conclusion was that missing data carries information. However, filling it in would enable a lot of other variable values, and information of the missing value would not get lost since the output variable series has that information. The stats::spline() R function was used

to apply the cubic spline interpolation method for generating synthetic data points instead of the missing ones.

The input variables have been filtered for all possible issues to keep only the healthy sensors' measurement series. Those were similarly checked to see if they had missing values and what structure they had for those. A summary table was prepared to evaluate it. A great part of the missing values happened with no more than 9 missing values between two measurements. There were only a very few occasions when a gap with over 30 missing values could be found during two measurements. The rest had their largest gaps from 9 to 30 missing values in them. Based on Wu et al. (2022, p. 10678), the gaps were not considered to be so large gaps that need special attention. The same stats::spline() R function was used to fill all gaps by cubic spline interpolation again (Haslwanter, 2021, pp. 118–119).

The last step in the data pre-processing was to remove non-operational periods as advised by Letzgus (2020, p. 1381). During those periods, the CDM Flow rate goes down to zero, and so, the compressor is not operating. It may or may not be intentional. But it certainly gives inconsistency to the data that undermines the aim to find anomalies while running operation. The sections excluded were identified by the CDM Flow rate going down to zero with arbitrarily estimated interval around it. The interval starts with a slow-down period and finishes with a speed-up period.

5.2 Output Variable Labelling

It has been clarified that the output variable labelling happens with the *changepoint.np* R package which is particularly designed to identify changepoints in nonparametric time series data. The *cpt.np* function was applied to find the changepoints after some experiments. The experiments confirmed the likeliness of very long runtime on the current size of data. Therefore, an alternative approach was chosen. One tenth of the Y data was included. It was arranged so that mean of each 10 consecutive data points was calculated, and the data series was recreated by the series of mean values.

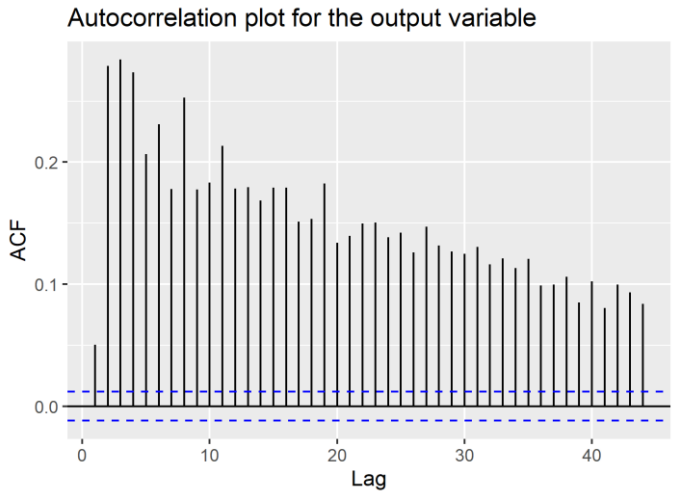


Figure 5.2: Autocorrelation plot for the recreated output variable

The sensor data is typically autocorrelated, and even if the recreated data series is an aggregated data, it still has this characteristic. This is confirmed by the autocorrelation plot (Figure 5.2). All lags are out of the confidence interval represented by the blue dashed line.

In this way, the runtime was only a little more than three hours. The penalty value had a large range with 10 as minimum and 200 as maximum. The number of quantiles has a default value 10, but Haynes et al. (2017, p. 1301) advise to use *4*log(n)* where *n* is the

length of the data series. Therefore *4*log(n)* was used as *nquantiles* parameter. This parameter setting represent a balance between accuracy and runtime since larger parameter value results better accuracy, but longer runtime. Minimum segment length was 1.

The outcome of the optimization was evaluated by the CROPS method as outlined already in the Method chapter. This method contrasts the penalty values and the number of changepoints found by applying the different penalty values. This is made transparent by a purposefully designed plot called "elbow" plot. It is evaluated manually which is subjective. During the evaluation, the analyst tries to find the point until the test statistic keeps dropping intensively. When its decrease slows down it indicates false positives are taken in not reducing the cost function intensively anymore. 44 changepoints were found as highlighted by the vertical blue line on Figure 5.3.

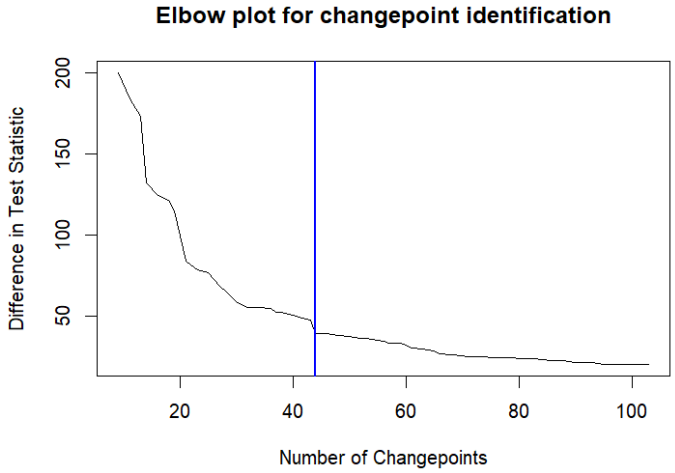


Figure 5.3: The elbow plot to identify the number of changepoints

The algorithm does not consider confidence intervals around the changepoints. Therefore, the changepoints can be directly utilized to identify all segments in the data. However, the current setting includes aggregation of data prepared for modelling. One data point used for changepoint detection represents ten real data points. Thus, a changepoint represents ten real data points as well, and any of them can be the single real changepoint. It is not going to be further finetuned which one is the real changepoint. Instead, as segments start with changepoints, the 10 data points will be the first 10 elements of a segment.

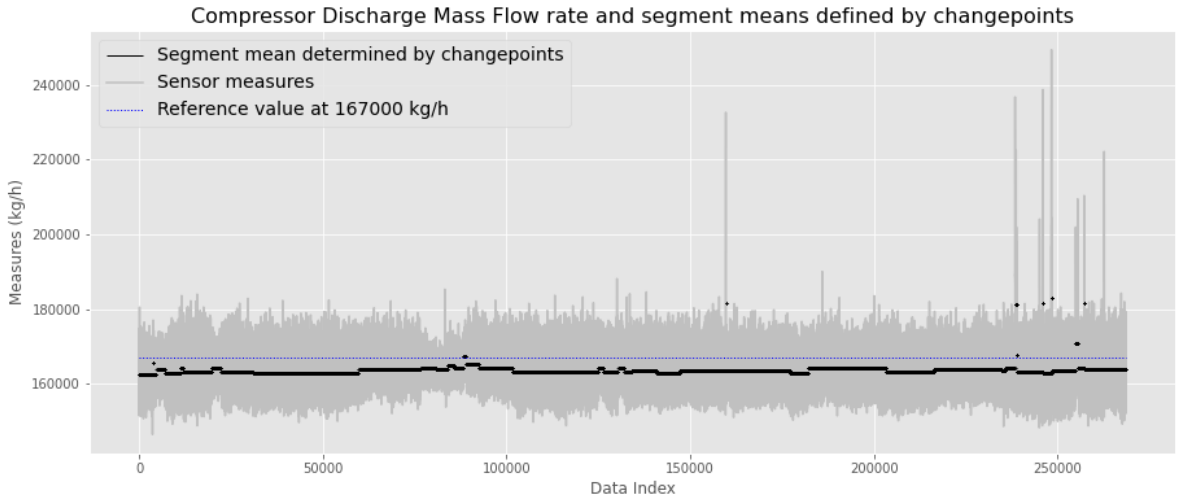


Figure 5.4: The image shows the sensor data values indicated by the grey colour; black colour highlights the segments and their mean values that was identified by the changepoint detection process; the blue reference line indicates the arbitrarily chosen limit that differentiates the anomalous and non-anomalous segments.

Segment ID	Count of Data Points	Segment Mean	Y Label
1	3810	162764	
2	210	165825	
3	720	162652	
4	2220	163873	
5	4330	162794	
6	760	164286	
7	7750	163235	
8	2570	164470	
9	8540	163448	
10	28890	163038	
11	16940	163932	
12	4120	164448	
13	3040	163920	
14	1850	164969	
15	2620	164273	
16	710	167493	anomalous
17	3530	165214	
18	8950	164464	
19	23310	163358	
20	1530	164333	
21	4010	163321	
22	1740	164426	
23	2090	163188	
24	6730	163590	
25	6140	163083	
26	12610	163803	
27	30	181663	anomalous
28	17400	163660	
29	4900	163071	
30	21350	164155	
31	12930	163400	
32	18470	164074	
33	1120	163631	
34	2530	164283	
35	390	181164	anomalous
36	120	167770	anomalous
37	7070	163357	
38	60	181819	anomalous
39	2270	163062	
40	90	182945	anomalous
41	6510	163474	
42	530	170993	anomalous
43	1770	164195	
44	100	181733	anomalous
45	11240	163856	

Table 5.1: Segment list with labels if a segment is anomalous

Based on the univariate analysis of the output variable, the data can be labelled. The reference value for that is arbitrarily chosen. It is 167,000 kg/h, and so, any segment having a mean value above the reference value is considered anomalous with all its data points. The reference value is identified by manually choosing a value above which only short-term data series can be found with some descriptive characteristics away from the common characteristics values of the rest of the data. The descriptive characteristics is the segment mean in this case. This is described graphically on Figure 5.4, and Table 5.1 gives the list of segments identified. The method to define the reference value in this way falls into the category of "manual selection of representative operational patterns" (Letzgus, 2020, p. 1376). It is a domain knowledge-based filtering approach to ensure healthy training data for modelling.

The final step before modelling is to prepare the training, validation, and test set. The data is a time series data, and splitting such data happens using time windows (Hyndman & Athanasopoulos, 2018) which is a non-random approach. This helps recognizing short term temporary patterns. The split is based on segments identified during change point detection process. The training set consists of segments from 1 till 30; the validation set consists of segments from 31 to 36; the test set consists of the segments from 37 and above.

5.3 Multivariate Modelling

The purpose of providing details of the model development results is to answer the research questions. It should utilize some of the outcomes of the exploratory data analyses as Figure 4.1 highlights. Learning from that is incorporated into the analytics workflow when it is required. The data set is imbalanced having 2030 anomalous and 264339 non-anomalous data points. Therefore, the workflow applies methods particularly suitable for such data. This includes the metrics. Model hyperparameter tuning happens based on F1 score and the final model is evaluation using confusion matrices as Fernández et al. (2018, pp. 51-52) and Lai et al. (2021, p. 8) argue.

The final special characteristics of this workflow is that while training happens on the training data set as usual, the hyperparameter tuning is based on the validation set (Mack et al., 2014, p. 8780). This is because one-class algorithm is used for modelling.

The input variable data is normalized and standardized due to the different units and scales present in the data. It ensures the algorithms can provide correct outputs.

5.3.1 The Baseline Model

The aim with a baseline model in practice is to have a simple model to which other more complex ones can be compared (K. Teh et al., 2020, p. 2). Its current purpose is also to provide a first impression on the data set, and to support working out a well-tested proof of concept for model training by its shorter runtime.

The analytics workflow follows certain steps in all three modelling subsections. The model training happens by iterations. In each iteration, the algorithm runs through a range of hyperparameters for ν in the form of a grid search and finds the best of those by the best F1 score. The difference between earlier and later iterations is the number of digits of accuracy of the hyperparameter range used. The last iteration is identified by having applied the highest digits of accuracy of hyperparameter range with no F1 score improvement achieved compared to the previous one. This step by step approach to refine parameter space for the grid search with OCSVM is proposed by Mack (2014) which is a Github location for Mack et al. (2014).

The best hyperparameter by the last iteration is used to construct the so-called learning curve. This is to see if the amount of training data – considering the model complexity – can provide a good model with limited bias and variance. Another indicator – called validation curve – is also constructed. It utilizes all best hyperparameters of the iterations to notify if the model is overfit or not (Raschka et al., 2022, pp. 180–185).

The current baseline model utilizes all training records and all 41 variables, but nothing else. As introduced in Data chapter, sensors of input variables are placed physically before the compressor, and the target variable is placed after it. Four iterations were run with SGDOneClassSVM function, please see Table 5.1 for details. The first iteration used ν range and *learning rate* range, too. The best *learning rate* was used in the rest of the iterations.

	Param. Range	**No of fits**	**Nu**	**Learning Rate**	**F1 score**	**Runtime (s)**
1	nu: [0.00, 1] l. rate: [10^{-3}, 10^{-6}]	404	0.45	10^{-5}	0.995985	338
2	nu: [0.44, 0.46]	2001	0.446	10^{-5}	0.997040	348
3	nu: [0.445, 0.447]	2001	0.44648	10^{-5}	0.997168	356
4	nu: [0.4454, 0.4456]	2001	0.445536	10^{-5}	0.997168	340

Table 5.1: Iterations of baseline modelling; the best result is given by the third iteration

The learning curve is shown by Figure 5.5 - Left. Since only the training data with non-anomalous data is used for that, the accuracy metric type is applied. It confirms large bias and variance.

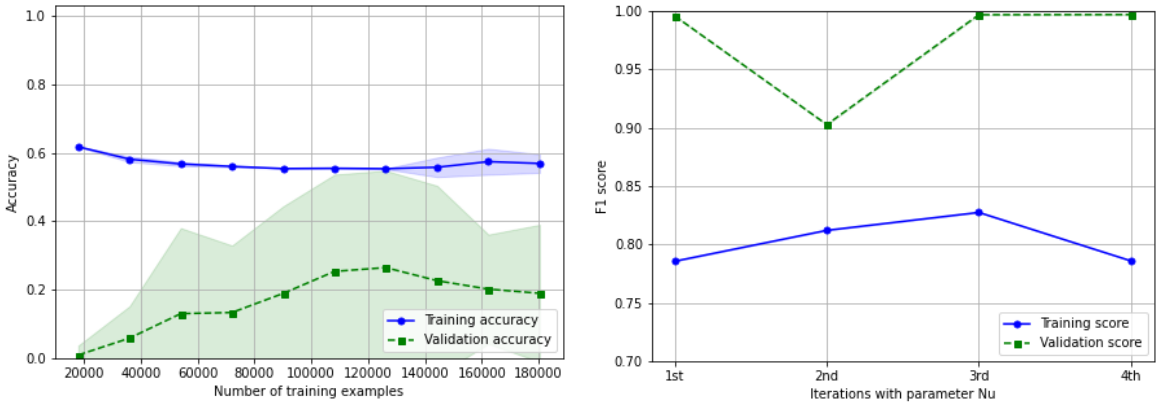


Figure 5.5: Left - Learning curve of the baseline model; Right - Validation curve of the baseline model

The validation curve is shown by Figure 5.5 – Right. This also confirms that the fourth model is overfitted. The overfit is indicated by the declining training score while the validation score is not declining.

5.3.2 The Model Including Pairwise-Interactions and Feature Selection

The exploratory data analysis revealed multicollinearity amongst the variables (Appendix B). This is important because of multiple reasons. Adding interaction terms to the model may improve that. On the other hand, SGD algorithm handle multicollinearity with great difficulty. In addition, including interactions would involve noise, too, which is a harm for the SVM algorithm. Nevertheless, adding pairwise interaction terms to the model results ($0.5*41+0.5*41^2$) features, so 861 all in all. Table 5.2 gives the iterations' details.

	Param. Range	**No of fits**	**Nu**	**Learning Rate**	**F1 score**	**Runtime (s)**
1	nu: [0.00, 1]	1001	0.006	10^{-5}	0.993579	1919
2	nu: [0.005, 0.007]	2001	0.00693	10^{-5}	0.995836	2586
3	nu: [0.00692, 0.00694]	2001	0.00693	10^{-5}	0.995836	2839

Table 5.2: Iterations of model including pairwise-interactions; the best result is given by second iteration

The learning curve in Figure 5.6 – Left confirms a huge improvement if it is compared to Figure 5.5 – Left. It highlights a strong model and enough training data. The validation curve is not

that informative showing the F1 scores of two iterations only (Figure 5.6 – Right). Nevertheless, it indicates an improved model, too.

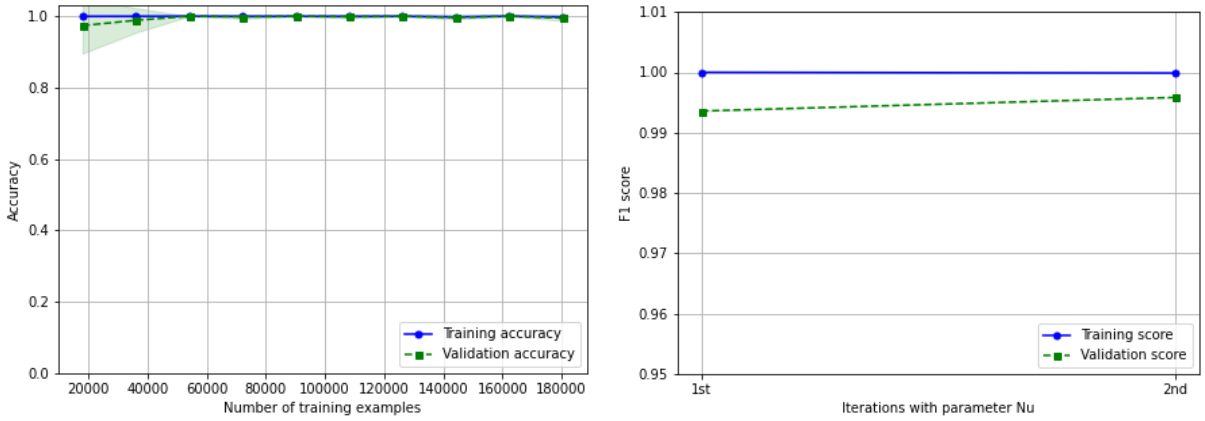


Figure 5.6: Left – Learning curve of the model with pairwise interactions; Right – Validation curve of the model with pairwise interactions

Since the noise and redundant information in the model have been increased, SVM-ANOVA feature selection method was applied to improve the model further. The feature selection method generates subsets of the full variable set and evaluate the performances of the models based on those subsets. The evaluation is made by 10-fold cross-validation. It tries to find the way to drop a significant number of features out of the model while improving the model capabilities. Figure 5.9 describes the result graphically. It indicates that the best model has more than 800 features included still. Thus, it is not considered as a successful attempt to improve the model.

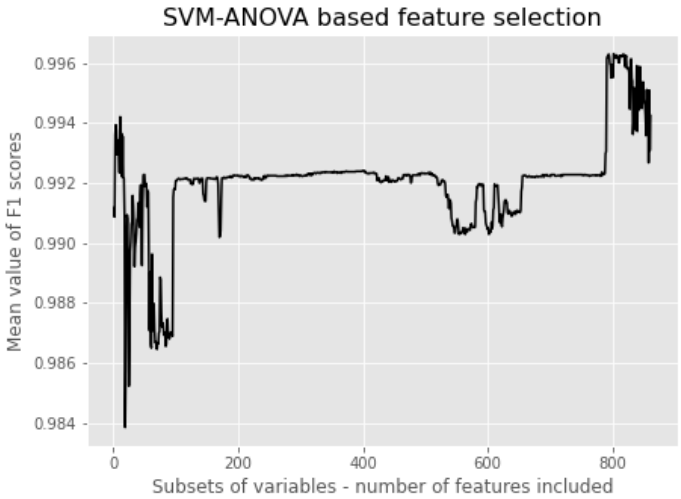


Figure 5.7: Mean F1 scores of variable subsets as per SVM-ANOVA

5.3.3 The Model Including Pairwise-Interactions and Dimensionality Reduction

The next step in the workflow was to arrange a different order of actions. First, noise and multicollinearity are excluded and then model is created using the remaining information. IncrementalPCA function of Scikit-Learn is applied as dimensionality reduction method on the data with pairwise interactions. 29 principal components gave the highest model validation score in the end. Any higher number of principal components resulted no convergence while model training. Iteration results can be seen in Table 5.3.

	Param. Range	No of fits	Nu	L. Rate	F1 score	Runtime (s)
1	nu: [0.00, 1]	1001	0.179	10^{-3}	0.998187	707
2	nu: [0.17, 0.19]	2001	0.18288000000000001	10^{-3}	0.998329	274
3	nu: [0.181, 0.183]	2001	0.181141	10^{-3}	0.998801	290
4	nu: [0.1810, 0.1812]	2001	0.1810486	10^{-3}	0.999601	282
5	nu: [0.18103, 0.18105]	2001	0.18104859999999998	10^{-3}	0.999601	284

Table 5.3: Iterations of model including pairwise-interactions with 29 principal components; the best result is given by second iteration

The learning and validation curve in Figure 5.8 show the best result overall. The highest validation score is reached by the fourth iteration considering any earlier results without overfitting.

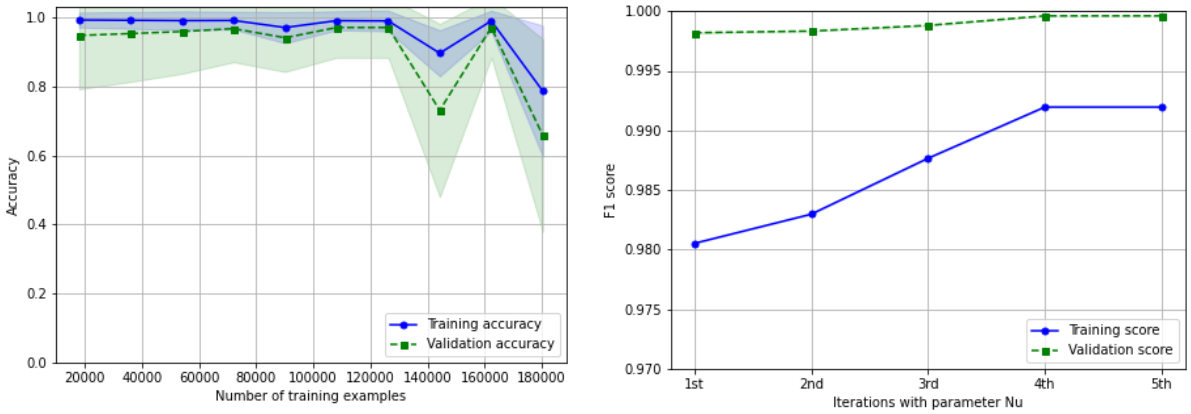


Figure 5.8: Left – Learning curve of the model with pairwise-interactions and 29 principal components; Right – Validation curve of the model with pairwise-interactions and 29 principal components

As discussed earlier, SVM is not a method that gives an output with probability values included. Regardless, it is possible to review the threshold value of the classification which is 0.5 in a normal probability-based case. The review is simple since all possible threshold values are tried and the one with the best predictive capabilities is kept. In this case, -0.004 is the best threshold value. It results 0.999786 F1 score with the validation set, and 0.983343 with the test set. Using the equation of (2*recall/(1+recall)) naïve random model F1 score, it gives 0.9927 and 0.9867 for the validation and test set respectively. Based on the test set result, the model is biased. Confirming the same, the relevant confusion matrices can be seen in Figure 5.9.

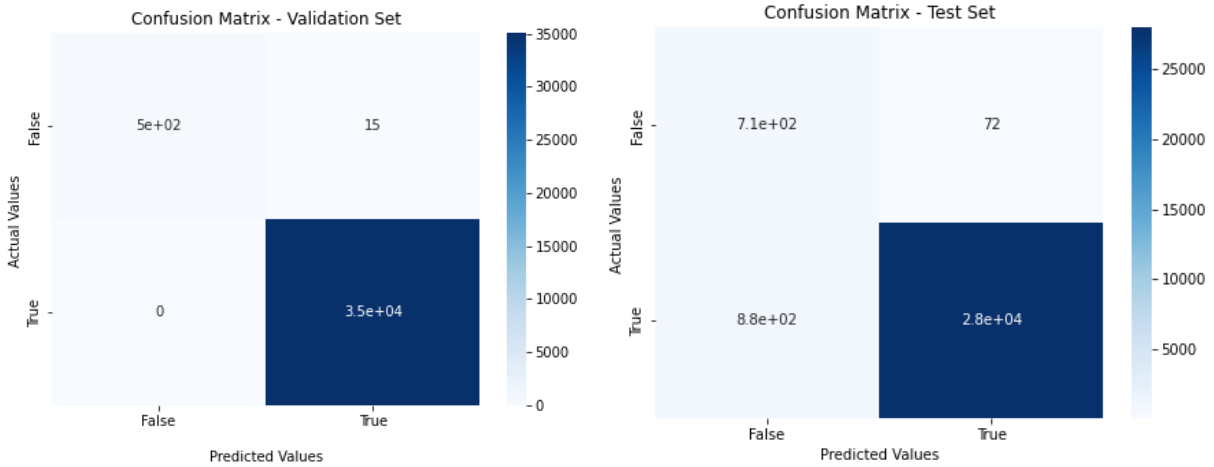


Figure 5.9: Left – Confusion matrix with validation set; Right – Confusion matrix with test set.

5.4 Benchmarking

The last analytical subsection provides information about a very practical aspect of using any algorithm. It is the runtime. In this case, it is measured how long it takes to run only one fit. Five Scikit-Learn implementations are compared to each other from this viewpoint. These algorithms are the SGD based One-Class SVM, Isolation Forest, Local Outlier Factor, Elliptic Envelope, and the standard One-Class SVM. These are particularly designed for anomaly detection. The last data set with 200429 records and 29 principal components is used for training the algorithms. Results can be seen in Figure 5.10.

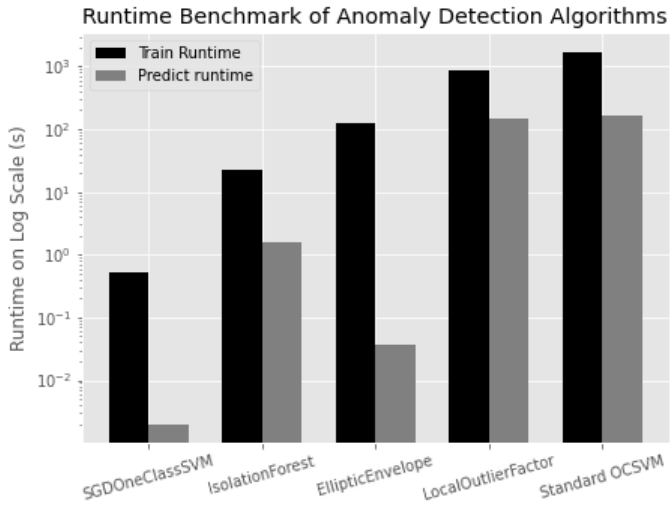


Figure 5.10: Runtime benchmark of anomaly detection algorithms

The results confirm that the SGD based One-Class SVM implementation is faster than all other algorithms considering both training and predicting runtime. Regarding runtime for training, it is more than 3,000 times faster compared to the standard OCSVM, and it is 42 times faster than the Isolation Forest which is the nearest one to it.

6 Discussion

Two research questions were raised at the beginning of the paper. The first one aims to clarify if the univariate nonparametric changepoint detection method of Haynes et al. (2017) can recognize anomalies efficiently in the current data set. Three main findings may drive the answer to the question. While applying the algorithm, some compromise had to be made due to the long runtime of the algorithm with this amount of data. The compromise meant aggregation of output variable data for the sake of shorter, practically feasible runtime which was still over three hours. The compromise did not result a weak model considering the validation results after modelling. However, the test result – with its 0.983343 F1 score – leaves room for interpretation. One potential interpretation may be based on the false negatives and false positives the model predicts. In the current situation, if the false negatives are more important to pay attention on then the current model may provide an acceptable result that could be used in practice. If the false positives are more important to keep on low level, then the current best model needs further enhancements applying potentially a different approach for output variable data labelling.

The second research question sought clarification on the potential advantages of the SGD based One-Class SVM function. It is confirmed it is currently the fastest of all anomaly detection methods within the Scikit-Learn library. Furthermore, it can comfortably handle the current data set. In fact, it is assumable it can conveniently handle five to ten times more data.

Therefore, it is realistically a great tool for an industrial professional when only a common desktop environment is available for doing an analysis similar to the current one.

6.1 Limitations and Future Directions

Multiple ways can be followed to complete such an analysis this thesis presents. What may remain more consistent though is the industrial environment and the challenges it gives to any analyst. Labelled industrial sensor data will probably be a less realistic expectation in the future, too. Though there may be promising approaches already to support such aims. Nevertheless, such uncertainty impacts the result of data analytics and may need continuous attention in the future. This is not only because of the limitation of the industry. It may be because of practical reasons. Any process can be divided into smaller ones, and the smaller ones may become the subject of individual analytics. Thus, any well-prepared overall data label could become a partial or no solution for subprocess oriented tasks. This means the output variable labelling methods and their characteristics shall remain an important topic for the analysts as well as for the researcher community.

Similarly, the amount of data continues to keep growing in industry. The methods that can handle the amount of data with real capability to get the valuable information out of it remains at least as important as today. Since the users have very different environments the low-cost accessibility of the methods is key for being able to take advantage of those for wider user base. This is a strength that are available for the practitioners already as confirmed by this paper as a common desktop environment was sufficient to use. However, this is only one data set which allows very limited generalization. Moreover, it comes with no technical verification of the findings, and this means its conclusions have limited real value.

Still, such case studies may be important partly to draw attention on interesting use cases, partly to provide reproducible workflows that can be followed by others. It may mean a form of a practical manual how to use a certain algorithm. It is known not all algorithms have good documentations. Future research could support those use cases by providing a larger selection of algorithms with capability to handle big data adapting faster optimization algorithms.

As a last remark, this paper leaves some room for improvements or interpretations. A number of additional tools and techniques could be used at every step of the work. Just to name a very few: some different changepoint sets could be compared to each other, feature engineering could be used during data pre-processing, Kernel PCA and independent component analysis could be applied during modelling as the data analysed is non-normally distributed, and model calibration could support finalizing the model in the end.

6.2 Conclusion

In the present paper, two algorithms were assessed if they can conveniently support an analyst when working with real industrial big data set. The changepoint detection algorithm was able identify segments in the output data, and those segments could be utilized efficiently to recognize manually selected patterns in the process data. The runtime of the algorithm forced to alter the standard approach by aggregating the data, though. The time constrained applied during the analysis may not be applicable in a real-life scenario. Therefore, the conclusion is that the changepoint algorithm applied can support the practitioners in operational environments. However, the size of the current data set may represent a close-to-the-maximum number of records. The SGD based One-Class SVM algorithm was able to generate a state-of-the-art predictive model. Nevertheless, it was dependent on the labelling process. Its runtime was suitable to the task. Moreover, due to its linearly scaling runtime, this algorithm may be easily suitable for five to ten times more records to analyse still in a common desktop environment. The best model gained is biased that may be taken into use in some specific operational circumstances, but it would require improvements in some others.

7 Appendices

7.1 Appendix A

The list below contains all potential input variables that are technically available. The variable code columns indicate the code that correspond to the same in the final input variable list of 3.2 The Input Variables subsection.

Variable Name	Variable ID	Variable Code
VAL_23-PDT-92501:X.Value	pi:160627	X1
VAL_23-KA-9101-M01_OC_low stage_NOC3low:VALUE	pi:160252	
VAL_23_FIC_92543_02:Z.X.Value	pi:160031	X2
VAL_23-LY-92529_SILch0_SC0_TYSP:VALUE	pi:160574	
VAL_23-KA-9101-M01_Motor_Winding_Temp_Trip:VALUE	pi:160244	
VAL_23-KA-9101-M01_TCS_Protection_Trip:VALUE	pi:160259	
VAL_23-LY-92529_Setpoints_SP_waterHi:VALUE	pi:160566	
VAL_23_FIC_92543_01:Z.X.Value	pi:160030	X3
VAL_23-FI-92512A:X.Value	pi:160179	
VAL_23-TIC-92504:Control Module:BX	pi:160775	
VAL_23-LY-92529_Outputs_sandLevel:VALUE	pi:160560	
VAL_23-FT-92512:X.Value	pi:160182	X4
VAL_23-KA-9101-M01_Winding_Temperature_U1:VALUE	pi:160261	X5
VAL_23_KA_9101_M01_62C:Z.X.Value	pi:160068	X6
VAL_23-LIC-92521:Z.Y.Value	pi:160500	X7
VAL_23-LY-92529_DensityBands_02foamLL:VALUE	pi:160545	
VAL_23-KA-9101-M01_Thermal_overload_TOL3dev:VALUE	pi:160260	
VAL_23-FE-9122-H01-F:Z.Y.Value	pi:160171	
VAL_23-PT-92504:X.Value	pi:160696	X8
VAL_23-PC-9109-M01-39:X.Value	pi:160613	
VAL_23-KA-9101-M01_OC_high stage_NOC3high:VALUE	pi:160248	
VAL_23-LY-92529_NON-SILStatus_Heartbeat:VALUE	pi:160555	
VAL_23-LY-92529_Setpoints_SP_foamHi:VALUE	pi:160563	
VAL_23-LY-92529_Setpoints_SP_oilHi:VALUE	pi:160565	
VAL_23-FE-9122-39:X.Value	pi:160170	
VAL_23-LY-92529_Outputs_emulsionLevel:VALUE	pi:160556	
VAL_23-FI-92512B:X.Value	pi:160180	
VAL_23-LY-92529_SILch0_SC0_TX1SP:VALUE	pi:160568	
VAL_23-KA-9101-M01_OC_high stage_negative_seq_NPShigh:VALUE	pi:160247	
VAL_23-LT-92529-02:X.Value	pi:160521	
VAL_23-LY-92529_Alarms_PNTA:VALUE	pi:160542	
VAL_23-KA-9101-M01-EL:XS.CoreStatus.XGL.Signal.Value	pi:160270	
VAL_23-LV-92521B:Y.Value	pi:160537	
VAL_23-LY-92529_SILch0_SC0_TX1PSP:VALUE	pi:160567	
VAL_23-LY-92529_DensityBands_01gasLL:VALUE	pi:160543	
VAL_23-LY-92529_DensityBands_02foamUL:VALUE	pi:160546	
VAL_23-LIC-92521:Control Module:BA	pi:160492	
VAL_23-TT-92502:X.Value	pi:160882	X9
VAL_23-KA-9101-M01:HSI.StatusMotorOn	pi:160229	

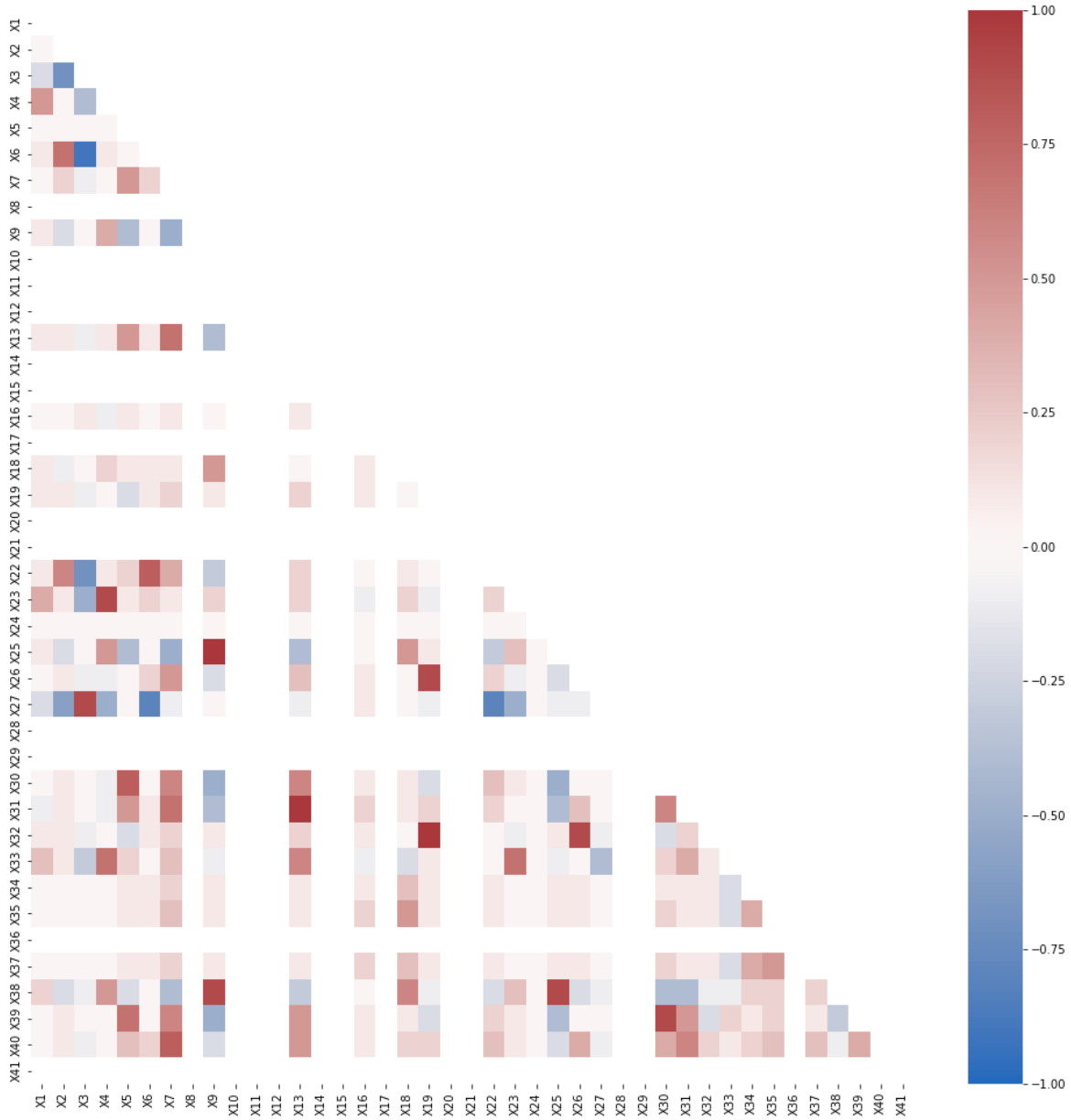
Variable Name	Variable ID	Variable Code
VAL_23-LIC-92521:Control Module:BX	pi:160493	
VAL_23-KA-9101-M01-EL:XS.CoreStatus.XGH.Signal.Value	pi:160269	
VAL_23-LY-92529_SILch0_SC0_TXSP:VALUE	pi:160570	
VAL_23-LY-92529_DensityBands_06sandUL:VALUE	pi:160554	
VAL_23-LY-92529_SILch0_SC0_TXPSP:VALUE	pi:160569	
VAL_23-PDT-92522:X.Value	pi:160629	X10
VAL_23-KA-9101-M01_E_stop_active:VALUE	pi:160238	
VAL_23-PC-9109-M01-EL:XS.MeasuredValues.CurrentL2.Value.Value	pi:160614	
VAL_23-YZSL-92526-F:X.Value	pi:161061	
VAL_23-KA-9101_ASP:VALUE	pi:160220	X11
VAL_23-LT-92523:Z.X2.Value	pi:160520	X12
VAL_23-FE-9106-H01-EL:XS.MeasuredValues.CurrentL2.Value.Value	pi:160156	
VAL_23-YZSH-92526-F:X.Value	pi:161056	
VAL_23-LV-92521A:Y.Value	pi:160536	
VAL_23_ZT_92538:Z.X.Value	pi:160140	X13
VAL_23-PDT-92502:X.Value	pi:160628	X14
VAL_23-VA-9110-M01-EL:XS.MeasuredValues.CurrentL2.Value.Value	pi:160995	X15
VAL_23-PT-92512:X.Value	pi:160697	X16
VAL_23-LY-92529_DensityBands_05waterUL:VALUE	pi:160552	
VAL_23-KA-9101-M01_Selector_switch_local_remote:VALUE	pi:160257	
VAL_23-KA-9101-M01-EL:XS.CoreStatus.XZ.Signal.Value	pi:160272	
VAL_23-LY-92529_Outputs_oilLevel:VALUE	pi:160559	
VAL_23-LIC-92521:Z.X.Value	pi:160499	X17
VAL_23_ZT_92543:Z.X.Value	pi:160141	X18
VAL_23-LY-92529_Outputs_foamLevel:VALUE	pi:160557	
VAL_23-LY-92529_DensityBands_03oilUL:VALUE	pi:160548	
VAL_23-TIC-92504:Control Module:YR	pi:160779	
VAL_23-TIC-92504:Control Module:PD	pi:160776	
VAL_23-TIC-92504:Control Module:PI	pi:160777	
VAL_23-TIC-92504:Z.Y.Value	pi:160782	
VAL_23-PC-9109-M01:Z.Y.Value	pi:160612	
VAL_23_KA_9101_M01_60A:Z.X.Value	pi:160067	
VAL_23-KA-9101-M01_Winding_Temperature_V1:VALUE	pi:160262	X19
VAL_23-KA-9101-M01_Spring_Charging_Alarm:VALUE	pi:160258	
VAL_23-KA-9101-M01_Motor_startup_supervision_MOTstart:VALUE	pi:160243	
VAL_23-TT-92512:X.Value	pi:160883	X20
VAL_23-LY-92529_DensityBands_03oilLL:VALUE	pi:160547	
VAL_23-LT-92523:Z.X1.Value	pi:160519	X21
VAL_23-LIC-92521:Control Module:PP	pi:160496	
VAL_23-LY-92529_DensityBands_04emulsionLL:VALUE	pi:160549	
VAL_23-TT-92533:X.Value	pi:160884	X22
VAL_23-LY-92529_DensityBands_06sandLL:VALUE	pi:160553	
VAL_23-KA-9101-M01_Reactive_power_kVAr:VALUE	pi:160254	X23
VAL_23-KA-9101-M01_Current_L2:VALUE	pi:160236	X24
VAL_23-LIC-92521:Control Module:YR	pi:160497	
VAL_23-KA-9101-M01_Current_L3:VALUE	pi:160237	X25
VAL_23-PT-92535:X.Value	pi:160701	X26

Variable Name	Variable ID	Variable Code
VAL_23-LY-92529_DensityBands_04emulsionUL:VALUE	pi:160550	
VAL_23-KA-9101-M01_OC_low_stage_residual_NEF1low:VALUE	pi:160253	
VAL_23-KA-9101-M01_MCB_failure-Motor_Tripping:VALUE	pi:160242	
VAL_23-LY-92529_Outputs_gasLevel:VALUE	pi:160558	
VAL_23-KA-9101-M01_Winding_Temperature_W1:VALUE	pi:160263	X27
VAL_23-FT-92521:X.Value	pi:160183	X28
VAL_23-LIC-92521:Control Module:PI	pi:160495	
VAL_23-KA-9101-M01-EL:XS.CoreStatus.Xservice.Signal.Value	pi:160271	
VAL_23-KA-9101-M01-39B:X.Value	pi:160266	
VAL_23-TIC-92504:Control Module:BA	pi:160774	
VAL_23-VA-9110-M01:Z.Y.Value	pi:160993	
VAL_23-KA-9101-M01_Internal_Relay_Fault:VALUE	pi:160241	
VAL_23-KA-9101-M01-EL:XS.Alarms.Alarm1.Signal.Value	pi:160268	
VAL_23-GK-9107B-M01-EL:XS.MeasuredValues.CurrentL2.Value.Value	pi:160208	
VAL_23-LY-92529_DensityBands_01gasUL:VALUE	pi:160544	
VAL_23-PT-92521:X.Value	pi:160698	X29
VAL_23-KA-9101-M01-36A:X.Value	pi:160264	
VAL_23-FE-9122-H01-F-EL:XS.MeasuredValues.CurrentL2.Value.Value	pi:160172	
VAL_23-TIC-92504:Control Module:PP	pi:160778	
VAL_23-PDT-92534:X.Value	pi:160631	X30
VAL_23-LY-92529_SILch0_SC0_TYPSP:VALUE	pi:160573	
VAL_23-LY-92529_Outputs_waterLevel:VALUE	pi:160561	
VAL_23-TIC-92504:Z.X.Value	pi:160781	X31
VAL_23-KA-9101_ESP:VALUE	pi:160224	X32
VAL_23-KA-9101-M01_OC_instantaneous_residual_NEF1inst:VALUE	pi:160250	
VAL_23-GK-9107B-M01:Z.Y.Value	pi:160207	
VAL_23-KA-9101-M01_CB_Mechanical_Fault:VALUE	pi:160232	
VAL_23-KA-9101-M01_OC_instantaneous_NOC3inst:VALUE	pi:160249	
VAL_23-KA-9101-M01_CB_multiple_plug_out:VALUE	pi:160233	
VAL_23-GK-9107A-M01:Z.Y.Value	pi:160205	
VAL_23-KA-9101-M01_Number_of_CB_Operations:VALUE	pi:160246	X33
VAL_23-KA-9101-M01_Interlock_active:VALUE	pi:160240	
VAL_23-KA-9101-M01:Z.Y.Value	pi:160230	
VAL_23-LY-92529_Setpoints_SP_gasHi:VALUE	pi:160564	
VAL_23-LY-92529_Setpoints_SP_emulsionHi:VALUE	pi:160562	
VAL_23-KA-9101-M01_Motor_Winding_Temp_Warning:VALUE	pi:160245	
VAL_23-KA-9101-M01_OC_low stage_negative_seq_NPSlow:VALUE	pi:160251	
VAL_23-LY-92529_SILch0_SC0_WATCHDOG:VALUE	pi:160575	
VAL_23-KA-9101-M01_REM543_in_test_mode:VALUE	pi:160255	
VAL_23-KA-9101-M01-62B:X.Value	pi:160267	X34
VAL_23-GK-9107A-M01-EL:XS.MeasuredValues.CurrentL2.Value.Value	pi:160206	
VAL_23-PT-92532:X.Value	pi:160700	X35
VAL_23_PT_92531:Z.X.Value	pi:160104	X36
VAL_23-LT-92529-03:X.Value	pi:160522	
VAL_23-LY-92529_SILch0_SC0_TY1SP:VALUE	pi:160572	
VAL_23-KA-9101-M01_Current_L1:VALUE	pi:160235	X37
VAL_23-LIC-92521:Control Module:PD	pi:160494	

Variable Name	Variable ID	Variable Code
VAL_23-KA-9101-M01_CB_travel_alarm:VALUE	pi:160234	
VAL_23_TT_92532:Z.X.Value	pi:160110	X38
VAL_23-VA-9110-M01-39:X.Value	pi:160994	
VAL_23-PDT-92530:X.Value	pi:160630	X39
VAL_23-KA-9101-M01_Active_power_kW:VALUE	pi:160231	X40
VAL_23-FE-9106-H01:Z.Y.Value	pi:160155	
VAL_23-KA-9101-M01_Earth_current:VALUE	pi:160239	
VAL_23-KA-9101-M01-EL:XS.MeasuredValues.CurrentL2.Value.Value	pi:160273	
VAL_23-LY-92529_SILch0_SC0_TY1PSP:VALUE	pi:160571	
VAL_23-PT-92523:X.Value	pi:160699	X41
VAL_23-KA-9101-M01-39A:X.Value	pi:160265	
VAL_23-LY-92529_DensityBands_05waterLL:VALUE	pi:160551	
VAL_23-KA-9101-M01_Running_hours:VALUE	pi:160256	

7.2 Appendix B

Correlation matrix of input variables created while exploratory data analysis



References

Alimohammadi, H., & Nancy Chen, S. (2022). Performance evaluation of outlier detection techniques in production timeseries: A systematic review and meta-analysis. *Expert Systems with Applications*, *191*, 116371. https://doi.org/10.1016/j.eswa.2021.116371

Amiri, A., & Allahyari, S. (2012). Change Point Estimation Methods for Control Chart Postsignal Diagnostics: A Literature Review. *Quality and Reliability Engineering International*, *28*(7), 673–685. https://doi.org/10.1002/qre.1266

Arellano, G. M., Nguyen, T., Hinton, C., & Ratchev, S. (2021, April 7). *Process Monitoring in a Flexible Manufacturing Cell through Data Analytics and a Visualisation Tool*. https://doi.org/10.21203/rs.3.rs-299328/v1

Bolivar, I. (2005). *AIAG – Statistical Process Control (SPC) 2nd Edition*. https://www.academia.edu/7829906/AIAG_Statistical_Process_Control_SPC_2nd_Edition

Bottou, L. (2012). Stochastic Gradient Descent Tricks. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural Networks: Tricks of the Trade* (Vol. 7700, pp. 421–436). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_25

Burg, G. J. J. van den, & Williams, C. K. I. (2020). An Evaluation of Change Point Detection Algorithms. *ArXiv:2003.06222 [Cs, Stat]*. http://arxiv.org/abs/2003.06222

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

de Siqueira Santos, S., Takahashi, D. Y., Nakata, A., & Fujita, A. (2014). A comparative study of statistical methods used to identify dependencies between gene expression signals. *Briefings in Bioinformatics*, *15*(6), 906–918. https://doi.org/10.1093/bib/bbt051

Fan, J., Sun, Q., Zhou, W.-X., & Zhu, Z. (2018). Principal component analysis for big data. *ArXiv:1801.01602 [Stat]*. http://arxiv.org/abs/1801.01602

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). *Learning from Imbalanced Data Sets*. Springer International Publishing. https://doi.org/10.1007/978-3-319-98074-4

Further Changepoint Analysis by Rebecca Killick. (n.d.). Retrieved 10 January 2022, from https://www.youtube.com/watch?v=GP5O3_iko1M

Haslwanter, T. (2021). *Hands-on Signal Analysis with Python: An Introduction*. Springer International Publishing. https://doi.org/10.1007/978-3-030-57903-6

Haynes, K., Eckley, I. A., & Fearnhead, P. (2014). Efficient penalty search for multiple changepoint problems. *ArXiv:1412.3617 [Stat]*. http://arxiv.org/abs/1412.3617

Haynes, K., Fearnhead, P., & Eckley, I. A. (2017). A computationally efficient nonparametric approach for changepoint detection. *Statistics and Computing*, *27*(5), 1293–1305. https://doi.org/10.1007/s11222-016-9687-5

Hyndman, R. J., & Athanasopoulos, G. (2018). 3.4 Evaluating forecast accuracy. In *Forecasting: Principles and Practice (2nd ed)*. https://otexts.com/fpp2/accuracy.html

Introduction to Changepoint Analysis by Rebecca Killick. (n.d.). Retrieved 10 January 2022, from https://www.youtube.com/watch?v=WelmlZK5G2Y

ISO 7870-1:2019 Control charts—Part 1: General guidelines. (n.d.). ISO. Retrieved 13 March 2022, from https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/96/69639.html

Killick, R., & Eckley, I. A. (2014). changepoint: An R Package for Changepoint Analysis. *Journal of Statistical Software*, *58*, 1–19. https://doi.org/10.18637/jss.v058.i03

Killick, R., Fearnhead, P., & Eckley, I. A. (2012). Optimal Detection of Changepoints With a Linear Computational Cost. *Journal of the American Statistical Association*, *107*(500), 1590–1598. https://doi.org/10.1080/01621459.2012.737745

Kutz, J. N., & Brunton, S. L. (Eds.). (2019). Clustering and Classification. In *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (pp. 154–194). Cambridge University Press; Cambridge Core. https://doi.org/10.1017/9781108380690.006

Lai, K.-H., Zha, D., Xu, J., Zhao, Y., Wang, G., & Hu, X. (2021, June 8). *Revisiting Time Series Outlier Detection: Definitions and Benchmarks*. Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1). https://openreview.net/forum?id=r8IvOsnHchr

Lei, Y., Yang, B., Jiang, X., Jia, F., Li, N., & Nandi, A. K. (2020). Applications of machine learning to machine fault diagnosis: A review and roadmap. *Mechanical Systems and Signal Processing*, *138*, 106587. https://doi.org/10.1016/j.ymssp.2019.106587

Letzgus, S. (2020). Change-point detection in wind turbine SCADA data for robust condition monitoring with normal behaviour models. *Wind Energy Science*, *5*(4), 1375–1397. https://doi.org/10.5194/wes-5-1375-2020

Lippi, V., & Ceccarelli, G. (2019). Incremental Principal Component Analysis: Exact Implementation and Continuity Corrections: *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics*, 473–480. https://doi.org/10.5220/0007743604730480

Ma, X., Hummer, D., Golden, J. J., Fox, P. A., Hazen, R. M., Morrison, S. M., Downs, R. T., Madhikarmi, B. L., Wang, C., & Meyer, M. B. (2017). Using Visual Exploratory Data Analysis to Facilitate Collaboration and Hypothesis Generation in Cross-Disciplinary Research. *ISPRS International Journal of Geo-Information*, *6*(11), 368. https://doi.org/10.3390/ijgi6110368

Mack, B. (2014). *One-class classification in R with the oneClass package* [Jupyter Notebook]. https://github.com/benmack/oneClass/blob/master/notebooks/oneClassIntro.ipynb (Original work published 2014)

Mack, B., Roscher, R., & Waske, B. (2014). Can I Trust My One-Class Classification? *Remote Sensing*, *6*(9), 8779–8802. https://doi.org/10.3390/rs6098779

Meng, L., McWilliams, B., Jarosinski, W., Park, H.-Y., Jung, Y.-G., Lee, J., & Zhang, J. (2020). Machine Learning in Additive Manufacturing: A Review. *JOM*, *72*(6), 2363–2377. https://doi.org/10.1007/s11837-020-04155-y

Mitra, P., Akhiyarov, D., Araya-Polo, M., & Byrd, D. (2020). Machine Learning-based Anomaly Detection with Magnetic Data. *Preprints*. https://doi.org/10.20944/preprints202012.0092.v1

Montavon, G., Orr, G. B., & Müller, K.-R. (Eds.). (2012). *Neural Networks: Tricks of the Trade: Second Edition* (Vol. 7700). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35289-8

Moyne, J., & Iskandar, J. (2017). Big Data Analytics for Smart Manufacturing: Case Studies in Semiconductor Manufacturing. *Processes*, *5*(3), 39. https://doi.org/10.3390/pr5030039

Nassif, A. B., Talib, M. A., Nasir, Q., & Dakalbab, F. M. (2021). Machine Learning for Anomaly Detection: A Systematic Review. *IEEE Access*, *9*, 78658–78700. https://doi.org/10.1109/ACCESS.2021.3083060

Nti, I. K., Adekoya, A. F., Weyori, B. A., & Nyarko-Boateng, O. (2021). Applications of artificial intelligence in engineering and manufacturing: A systematic review. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-021-01771-6

Nunes, E. C. (2021). *Anomalous Sound Detection with Machine Learning: A Systematic Review* (arXiv:2102.07820). arXiv. http://arxiv.org/abs/2102.07820

Open Industrial Data. (n.d.). Retrieved 18 June 2022, from https://openindustrialdata.com/

Open Industrial Data—Overview. (n.d.). Retrieved 3 July 2022, from https://openindustrialdata.com/media/1065/open-industrial-data-cognite-akerbp.pdf

Perez, R. X. (2022). Selecting the Best Type of Compressor for Your Application. In *Design, Modeling and Reliability in Rotating Machinery* (pp. 195–214). John Wiley & Sons, Ltd. https://doi.org/10.1002/9781119631699.ch7

Perperoglou, A., Sauerbrei, W., Abrahamowicz, M., & Schmid, M. (2019). A review of spline function procedures in R. *BMC Medical Research Methodology*, *19*(1), 46. https://doi.org/10.1186/s12874-019-0666-3

Ramsay, K., & Chenouri, S. (2021). Robust multiple change-point detection for multivariate variability using data depth. *ArXiv:2011.09558 [Stat]*. http://arxiv.org/abs/2011.09558

Raschka, S., Liu, Y. (Hayden), Mirjalili, V., Dzhulgakov, D., & ProQuest (Firm). (2022). *Machine learning with Pytorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Packt Publishing Ltd. https://go.exlibris.link/0rHThRwL

Reiten, H., & Kleiven, T. (2019). *Unsupervised Anomaly Detection on Streaming Data in the Petroleum Industry Using Deep Neural Networks*. https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2639893/no.ntnu:inspera:2525188.pdf?sequence=1

Schölkopf, B., Williamson, R. C., Smola, A., Shawe-Taylor, J., & Platt, J. (1999). Support vector method for novelty detection. *Advances in Neural Information Processing Systems*, *12*.

Scikit-Learn & Joblib with Dask. (n.d.). Retrieved 4 April 2022, from https://ml.dask.org/joblib.html

Scikit-Learn Release Note v1.0.0. (2021, September). Scikit-Learn. https://scikit-learn/stable/whats_new/v1.0.html

Shewhart, W. A. (1930). Economic quality control of manufactured product. *The Bell System Technical Journal*, *9*(2), 364–389. https://doi.org/10.1002/j.1538-7305.1930.tb00373.x

Smith, R. (2021). W. Edwards Deming, an Influential Statistician. *Research-Technology Management*, *64*(5), 58–60. https://doi.org/10.1080/08956308.2021.1949850

Sousa, S., Rodrigues, N., & Nunes, E. (2017). Application of SPC and Quality Tools for Process Improvement. *Procedia Manufacturing*, *11*, 1215–1222. https://doi.org/10.1016/j.promfg.2017.07.247

SVM-Anova: SVM with univariate feature selection. (n.d.). Scikit-Learn. Retrieved 18 April 2022, from https://scikit-learn.org/stable/auto_examples/svm/plot_svm_anova.html?highlight=svm+anova

Teh, H. Y., Kempa-Liehr, A. W., & Wang, K. I.-K. (2020). Sensor data quality: A systematic review. *Journal of Big Data*, *7*(1), 11. https://doi.org/10.1186/s40537-020-0285-1

Teh, K., Armitage, P., Tesfaye, S., Selvarajah, D., & Wilkinson, I. D. (2020). Imbalanced learning: Improving classification of diabetic neuropathy from magnetic resonance imaging. *PLOS ONE*, *15*(12), e0243907. https://doi.org/10.1371/journal.pone.0243907

Thudumu, S., Branch, P., Jin, J., & Singh, J. (2020). A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, *7*(1), 42. https://doi.org/10.1186/s40537-020-00320-x

Torres-Reyna, O. (2014). *Cubic interpolation using R*. Https://Dss.Princeton.Edu/Training/.

Usman, K., & Ramdhani, M. (2019). Comparison of Classical Interpolation Methods and Compressive Sensing for Missing Data Reconstruction. *2019 IEEE International Conference on Signals and Systems (ICSigSys)*, 29–33. https://doi.org/10.1109/ICSIGSYS.2019.8811057

Wallace, B. C., & Dahabreh, I. J. (2012). Class Probability Estimates are Unreliable for Imbalanced Data (and How to Fix Them). *2012 IEEE 12th International Conference on Data Mining*, 695–704. https://doi.org/10.1109/ICDM.2012.115

Wang, Z., Chu, T., Choate, L. A., & Danko, C. G. (2017). *Rgtsvm: Support Vector Machines on a GPU in R*. 6.

What is Terality? (n.d.). Retrieved 4 February 2022, from https://docs.terality.com/

Wilms, I., Killick, R., & Matteson, D. S. (2021). Graphical Influence Diagnostics for Changepoint Models. *Journal of Computational and Graphical Statistics*, *0*(ja), 1–21. https://doi.org/10.1080/10618600.2021.2000873

Wu, R., Hamshaw, S. D., Yang, L., Kincaid, D. W., Etheridge, R., & Ghasemkhani, A. (2022). Data Imputation for Multivariate Time Series Sensor Data With Large Gaps of Missing

Data. *IEEE Sensors Journal*, *22*(11), 10671–10683. https://doi.org/10.1109/JSEN.2022.3166643

Zhang, X., Wu, Y., Wang, L., & Li, R. (2016). Variable Selection for Support Vector Machines in Moderately High Dimensions. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, *78*(1), 53–76. https://doi.org/10.1111/rssb.12100