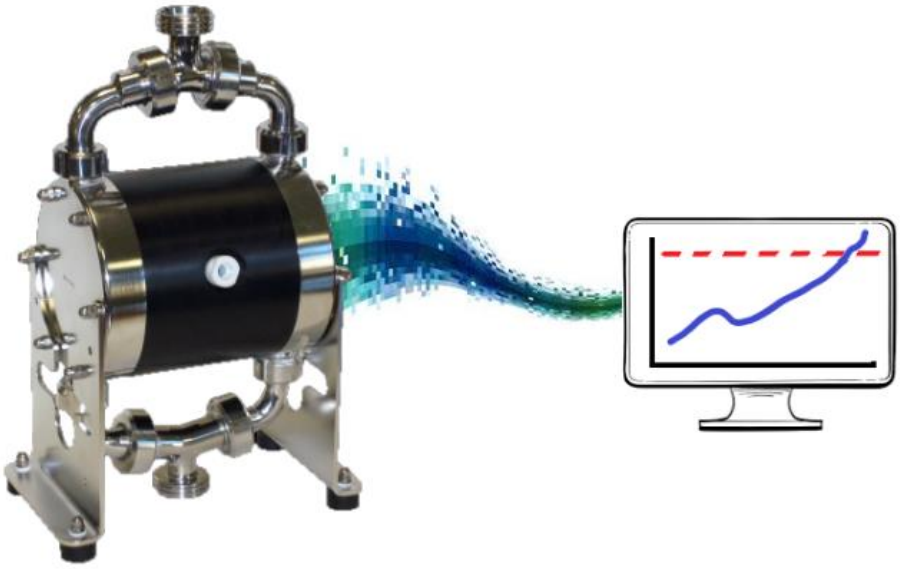
Predictive maintenance at Canon: Models to minimise downtime, a case study



Author : Freek Klabbers

Version : V2.3

Date : 12 June 2023

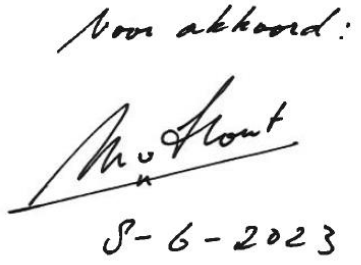
Education : Applied Physics, Fontys University of Applied Sciences

School of Natural Sciences

Predictive maintenance at Canon: Models to minimise downtime, a case study

Author	Freek Klabbers
Contact	Freek.klabbers@cpp.canon
Version	V2.3
Date	12 June 2023
Period	1st of February until the 30th of June
Location	Venlo
University	Fontys University of Applied Sciences
Department	Fontys Applied Natural Sciences
Study	Applied Physics
Mentor	Ir. Rick Walraven M.Ed.

Company	Canon Production Printing Venlo
Department	Technical Services
Sub-Department	Maintenance Engineering
Company mentors	Ing. Marco van Hout and Ing. Ton Driessen
Address	Van der Grintenstraat 10, 5914 HH Venlo



Samenvatting

Bij Canon Production Printing (CPP) in Venlo wordt onder andere water gebaseerde latexinkt geproduceerd. Deze productielijn moet de groeiende vraag bijhouden. Een van de methodes om dit aan te pakken is stilstand voorkomen. Om dit te doen wil CPP onderhoud voorspellen op basis van de data die in de fabriek gegenereerd wordt, dit heet *predictive maintenance*. Predictive maintenance zorgt ervoor dat periodiek onderhoud niet meer nodig is en voorkomt correctief onderhoud door van te voren te waarschuwen over een toekomstig falen. Om een eerste stap te zetten naar predictive maintenance is een membraanpomp als *pilot* gekozen.

Het doel van dit onderzoek is:
'Het ontwerpen van een *Data-Driven* predictive maintenance model voor de membraanpomp in de latexinkt productielijn bij CPP Venlo die de totale stilstand verminderd.'

Daartoe wordt eerst een literatuurstudie uitgevoerd om na te gaan welke modellen en aanpakken gebruikt worden voor predictive maintenance. Drie *machine learning* aanpakken zijn gekozen uit de literatuur en worden geïmplementeerd. Deze aanpakken zijn: een *auto-encoder anomaly detector*, een classificatiemodel gebaseerd op ruwe tijdreeks data en een classificatiemodel gebaseerd op statistische parameters verkregen uit tijdreeks data.

De faalconditie die de modellen proberen te voorspellen is de slijtage van het balventiel. Hiervan bevinden zich twee instanties in de dataset. De eerder gekozen machine learning modellen worden getraind op de faaldata van het balventiel. Voor de *supervised* classificatiemodellen zijn de targets de 20 productiecycli vóór falen. De modellen worden gevalideerd met 4-fold kruisvalidatie, waarbij de F1-score als prestatiemaatstaf wordt gekozen in verband met de klasse-onbalans.

De auto-encoder bleek slechts één van de twee faalmomenten te kunnen voorspellen. Het classificatiemodel op basis van ruwe tijdreeksgegevens kon niet één geval goed voorspellen. Het classificatiemodel op basis van statistische parameters is echter in staat om de 20 productiecycli voor het falen nauwkeurig te classificeren met een F1-score van 0,79 ± 0,04 na optimalisatie. Op de test dataset is de F1-score 0,86.

De andere twee benaderingen waren niet succesvol, elk om hun eigen redenen. De meest limiterende factor van de auto-encoder is het feit dat het slechter is in het voorspellen van bepaalde inktsoorten. Aangezien hetzelfde inkttype meerdere keren achter elkaar wordt aangemaakt, verslechteren de voorspellingen voor die periode. Het tijdreeksclassificatiemodel faalde omdat het door de klasse-onbalans niet in staat is de minderheidsklasse nauwkeurig te leren. Beide problemen zijn op te lossen als de modellen anders worden ingericht.

Als het classificatie model met statistische parameter wordt toegepast, bespaart dit op deze pomp naar schatting 37 uur aan stilstand per jaar. Hiermee is het doel van dit rapport bereikt.

De aanbeveling van dit rapport is om het classificatiemodel voor statistische parameters te implementeren in een systeem dat periodiek nieuwe gegevens krijgt, daarop voorspellingen doet en deze weergeeft. Ook zouden de twee onnauwkeurige aanpakken opnieuw geëvalueerd kunnen worden. Beide zijn ze een verbetering op het huidige model op sommige vlakken, zoals minder voorbewerking van de data.

Summary

At Canon Production Printing (CPP) in Venlo water-based latex ink is produced. The production line needs to keep up with the rising demand. One way to approach this is to increase factory uptime. To do this CPP wants to predict when maintenance is needed based on the data the production line produces, this is called predictive maintenance. Predictive maintenance results in less preventive maintenance and would decrease unplanned downtime by warning about imminent failures. An Air-Operated Double Diaphragm (AODD) pump is selected as a pilot to make the first step towards predictive maintenance.

The goal of this research:
'Building a data-driven predictive maintenance model for the AODD pump in the Latex-based ink production plant at CPP Venlo that will reduce downtime of the plant.'

First, a literature study is conducted to determine which models and approaches are commonly used for predictive maintenance. Three machine-learning approaches have been selected to explore further. These approaches are an auto-encoder anomaly detector, a classification model based on raw time-series data and a classification model based on statistical parameters extracted from those time series.

The failure mode that the models will try to predict is the wear of the ball valve; two of these events are available in the dataset. The earlier chosen machine learning models are trained on the ball valve failure data. The targets for the supervised classification models are the 20 production cycles before failure. The models are validated using 4-fold cross-validation; the performance metric chosen is the F1-score because of the class imbalance.

The auto-encoder turned out to be able to predict only one out of two of the failure moments. The classification model, based on raw time-series data, is not able to predict even one case right. The classification model based on statistical parameters is able to accurately classify the 20 batches before failure with a F1-score of 0.79 ± 0.04 after optimisation on the validation set. On the test data, the F1-score is 0.86.

The other two approaches failed to be successful each for their own reasons. The auto-encoders biggest issue is that it performs worse at predicting certain ink types. Since the same ink type tends to be created multiple times in a row, the auto-encoders predictions worsen for that period of time. The raw time-series classifier failed because the class imbalance left it unable to accurately learn the minority class. Both issues are solvable if more time were to be spent on them.

The classification model based on statistical parameters, when implemented, will save an estimated 37 hours of downtime per year for this pump. This accomplishes the goal of this rapport.

The recommendation of this research is to implement the statistical parameters classification model in a system that will periodically get new data, make predictions on it and display it. Without this implantation, the model will be of no use. Furthermore, the two failed approaches could be re-evaluated because they have some desirable traits like requiring less pre-processing of the data.

Preface

This document before you is my graduation thesis titled: *'Predictive maintenance at Canon: Models to minimise downtime, a case study'*. I wrote this thesis for my graduation internship from the study Applied Physics at Fontys University of Applied Sciences located in Eindhoven, the Netherlands. This research was conducted at Canon Production Printing in Venlo under the department of Technical Services as part of the Maintenance Engineering team from February until June 2023.

The author would first like to thank all of the Canon Production Printing personnel that have helped and made to feel welcome. Particular thanks go out to Ton Driessen and Marco van Hout for being my mentors throughout this process, without whom this would not have been possible.
I also would like to thank Rick Walraven for guiding me through the graduation track of Fontys.

The author has used Python 3 for a lot of the programming in this thesis. Python and its packages are all open-source projects built by a big community for free. The author would like to thank these communities for their excellent and time-saving work. In doing so these communities have enabled the author to conduct the research you see presented before you today.

The standard ISO 13372:2012(en) gives a common vocabulary for maintenance technicians in the field of condition monitoring and diagnostics systems [1]. Thus, these terms will be used throughout this report.

The writer was enabled by Canon Production Printing Netherlands B.V. to perform research that partly forms the basis for this report.
Canon Production Printing B.V. does not accept responsibility for the accuracy of the data, opinions, and conclusions mentioned in this report, which are entirely for the account of the writer.

Abbreviations

Table 1: Abbreviation List

Abbreviation	Meaning
AI	Artificial Intelligence
ANN	Artificial Neural Network
AU	Autoencoder
CBM	Condition Based Monitoring
CIP	Cleaning in Place
CPP	Canon Production Printing
Demi water	Demineralized water
FN	False Negative
FP	False Positive
IBC	Intermediate Bulk Container
MAE	Mean Average Error
ML	Machine Learning
ODBC	Open DataBase Connectivity
PCA	Principle Component Analysis
PdM	Predictive Maintenance
PLC	Programable Logic Control
PTFE	Polytetrafluoroethylene (Teflon)
SCADA	Supervisory control and data acquisition
SQL	Structured Query Language
TN	True Negative
TP	True Positive

Glossary

Table 2: Glossary for the report

Term	Meaning
Anomaly	Something that deviates from what is normal or expected.
Autoencoder (aka. AU)	A type of neural network that learns to reproduce its input through feature extraction.
Batch	The result of a single ink production cycle. Also used to refer to that production cycle specifically.
Class	A set or category of things. In machine learning a class is often used as the learning target.
Class imbalance	When a dataset has multiple classes where two or more classes have different sizes.
F1-score	A performance metric for classification on a scale of 0 to 1. The harmonic mean of the true positive rate and the positive predictive value.
Feature	A measurable property or characteristic a machine learning model can use to learn
Hyperparameter	A parameter that influences or changes the learning process of a machine learning model.
Machine learning (Aka. ML)	The practice where through algorithms and statistics computers learn patterns and decision-making from large amounts of data.
One-hot encoding	A technique to encode categorical data in a way that machine learning models can interpret it.
Supervised (Machine learning)	When a machine learning model gets the input and the answer. The models' task is then to learn what relationships of input data lead to the output data.
Target	Short for Target variable, the variable the supervised machine learning model needs to learn
Time series	A collection of data ordered based on the time recorded. Often sampled at regular intervals.

Contents

1 Introduction

Maintenance is a crucial part of a production process that is often overlooked. The traditional maintenance strategy was run to failure, this is when things are only repaired when they break. This is the most straightforward strategy but has many drawbacks, like downtime of equipment and high repair costs. This strategy is not viable for equipment that is not allowed to fail, for example, airplanes or elevators. Companies started implementing preventive maintenance, scheduling downtime for machinery to take it apart and assure everything is still working properly. This approach does have a few downsides as well. It is time-consuming, so the labour and planned downtime costs are still high. In some cases, preventive maintenance increases the chance of breakdowns occurring. This is called the Waddington Effect [2]. In these cases, preventive maintenance does more harm than good.

With the rise of Industrial internet of things, cheaper sensors, and more sophisticated process monitoring also comes access to more data. Companies often store this data in huge databases for later use. This data creates new opportunities in the field of maintenance namely, predictive maintenance (PdM). With this data, the condition of equipment can be monitored and breakdowns can be predicted based on past events. This technique prevents downtime by reducing unnecessary maintenance but does not have any of the downsides of preventive maintenance.

Canon Production Printing (CPP) has built a new production line for its latex-based inks. This state of the art production line contains a lot of monitoring and automatic data logging. The forecast is that the demand for this latex-based ink will rise greatly to the point that the factory might not be able to meet demand. It is crucial for CPP that this factory produces as much latex-based inks as possible to meet these demands. To achieve this the maintenance team wants to use the data from the plant to create a PdM Model. This model must decrease the downtime of the plant and thus result in more production.

For a proof of concept, it has been decided that an Air-Operated Double Diaphragm (AODD) pump will be selected as the starting point for PdM in the ink plant. This pump is crucial for the process, has experienced plenty of failures and thus caused a lot of downtime, and has a lot of sensors monitoring it. All of this makes it so that the AODD pump is the best candidate for a PdM approach.

The goal of this research is:
‘Building a data-driven predictive maintenance model for the AODD pump in the Latex-based ink production plant at CPP Venlo that will reduce downtime of the plant.’

To achieve this goal three main research questions are defined:

- What are the current failure modes of the AODD pump and what is their root cause?
- What models and approaches are most suitable for implementing a predictive maintenance system for the AODD pump?
- What model and approach performs best for predicting the failures of the AODD pump?

To accomplish these goals, in chapter 2 a literature study will be conducted on PdM. Some of the techniques used will be explained. Models suitable for PdM will be explored here too. Chapter 3 will describe the methods used to train the models and how to validate their performance. It will also give more information about the used dataset itself.

The dataset will be explored using exploratory data analysis. Based on these insights and the literature study a select few models will be applied on the data and tested. The results of all of this will be shown in chapter 4. In chapter 5 the accuracy of the results will be discussed. Additionally any limitations of the model or datasets will be mentioned. In chapter 6 a conclusion will be made based on the results. Recommendations for any future research can be found in chapter 7.

2 Theoretic Background

In this chapter all information needed to understand and interpret the results is described. First of all, an overview of available systems and sensor data will be made. Next, based on these resources, applicable predictive maintenance literature will be studied. From this research a select few models will be chosen and further described.

2.1 Latex ink production line

First, it is important to have knowledge of the factory, what system is used to log the data and what data is being logged. Without this, the literature study of PdM will not be able to focus on what strategies are most suitable for the available data.

The current production process can be described as follows:
First, the raw materials are dosed to a central mixing vessel. Here, the ingredients are mixed. The finished ink is filled into an Intermediate Bulk Container (IBC). The IBC is transported to the filling line. When a new colour ink needs to be created the whole system needs to be cleaned. For small regular cleaning demineralised water (demi water) is used; for bigger contaminations, the cleaning is done by a system called 'Cleaning In Place' (CIP). CIP first flushes all pipes with demi water, then with sodium hydroxide, followed by nitric acid and finally cleaned with demi water. These processes can be completed automatically by the Supervisory Control And Data Acquisition (SCADA) System [3].

The SCADA system is based on the Siemens PCS7. This system contains Siemens PLCs (Programmable Logic Control), which are connected to all the physical hardware. The PLCs can be read and controlled from a central engineering station. This station has its own software language where a complete recipe for a batch can be programmed. This way a recipe is programmed once, from there on operators can call this recipe without knowledge of precise quantities. The entire production line is automated this way, the only actions that are required from operators is making sure the raw resources are replenished once they are depleted.

To pilot PdM it is necessary to start with a singular piece of equipment. This equipment must meet the following requirements. First, it should have experienced a few failures in the past, if there have not been any failures yet it is harder to learn to predict them. Second, a piece of equipment should have sensors to measure and quantify its performance. Valves, for example, are hard to monitor because there is no physical quantity available that would indicate when it starts to deteriorate. Also, the equipment's cruciality for the process needs to be looked at, if the equipment can easily be replaced or doesn't shut down production then it is not worth monitoring since it is not cost efficient.
With these conditions in mind, the AODD pump at the bottom of the mixing vessel has been selected. When the mixing vessel is done mixing ink this pump is responsible for pumping the ink out of the mixing vessel. This pump is crucial, if the mixing vessel is not emptied it cannot start a new batch of ink. Since there is no backup system in place, this means pump failure will always result in downtime. Because of the pump's placement it is monitored with multiple sensors that could be indicators of deterioration.

This pump so far has caused a total of about 30 hours of downtime in a full year. Two of those failures were diaphragm ruptures, two other failures were caused by the check ball valves. Therefore is this critical pump with plenty of unforeseen failures is an ideal target for this PdM pilot.

2.1.1 Air-Operated Double Diaphragm pump

Like earlier stated, the AODD pump will be used for the PdM pilot. For this, it is useful to know the workings of the AODD pump so that deviating behaviour can be recognized. A schematic representation of an AODD pump is shown in Figure 1.

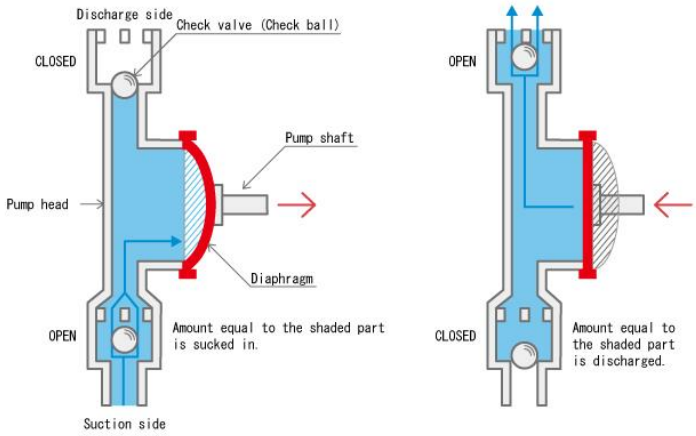


Figure 1: The schematics workings of an AODD pump [4].

The AODD pump is operated using air pressure. First, air pressure is used to pull the diaphragm (visualised in red) inwards creating suction. This suction pulls the liquid from the bottom past the check valve that works using a ball. The ball check valve will prevent the liquid from flowing back. The next step is to expand the diaphragm into the liquid, since the way down is blocked the only exit is up past another check valve. The diaphragm then subtracts again to suck in more liquid. This process is repeated to create a flow.
The AODD pump has two of the aforementioned chambers that operate in antiphase. When one diaphragm is expanding the other is receding and the other way around. The diaphragms are connected with a rod. This is how the air pressure manages to create suction, by pressing on the other side instead. The pump that is currently in use is the B40 BN-T-4-XVT from Almatec [5].

A current issue the AODD pump is facing is that the diaphragms keep failing quicker than they should according to the supplier. A diaphragm used under these conditions should last up to 2 years. The diaphragm currently tends to only last about 7 months or less. The supplier suspects wrongful use is the cause of this premature diaphragm breakdown. The AODD pump cannot handle a positive pressure on the suction side of the pump. It is suspected that during the cleaning procedure, a positive pressure is supplied. This claim can be investigated further by reviewing the data. If this turns out to be the case, changing the cleaning procedure could increase diaphragm lifespan and thus reduce the need for maintenance.

2.2 Predictive maintenance in literature

Next, it is useful to have a clear overview of maintenance strategies and what falls under PdM. There are multiple different maintenance strategies. The simplest form of maintenance is reactive maintenance. With reactive maintenance, maintenance occurs only when a system or component breaks down. This is a viable strategy for low-priority equipment like office chairs, but can’t be used for anything of high importance. Preventive Maintenance is designed to stop unplanned failures from occurring by regularly checking all the equipment and replacing parts when necessary. This however costs a considerable number of manhours and doesn’t completely eliminate unexpected

failures. In some cases too much preventive maintenance might increase the chance of unexpected breakdowns [2].

A new approach to maintenance has been thought up by using the data produced in the process. This new data-based prediction of failures is called PdM. The term PdM means in this context everything that uses historical data to make predictions of future required maintenance. Within PdM, a distinction is made between diagnostic and prognostic predictions. Diagnostic predictions are often achieved with Condition Based Monitoring (CBM) and can only predict the future failing of machinery, but not when it will fail. Prognostic predictions also predicts the moment of failure. This way the old equipment can be used as long as possible, saving wear on new equipment. The time the equipment can still be used before failure is called remaining useful life. CBM and remaining useful life estimators use data and possibly machine learning (ML) models to predict when a system is about to break down. These different maintenance strategies are visually represented in Figure 2.

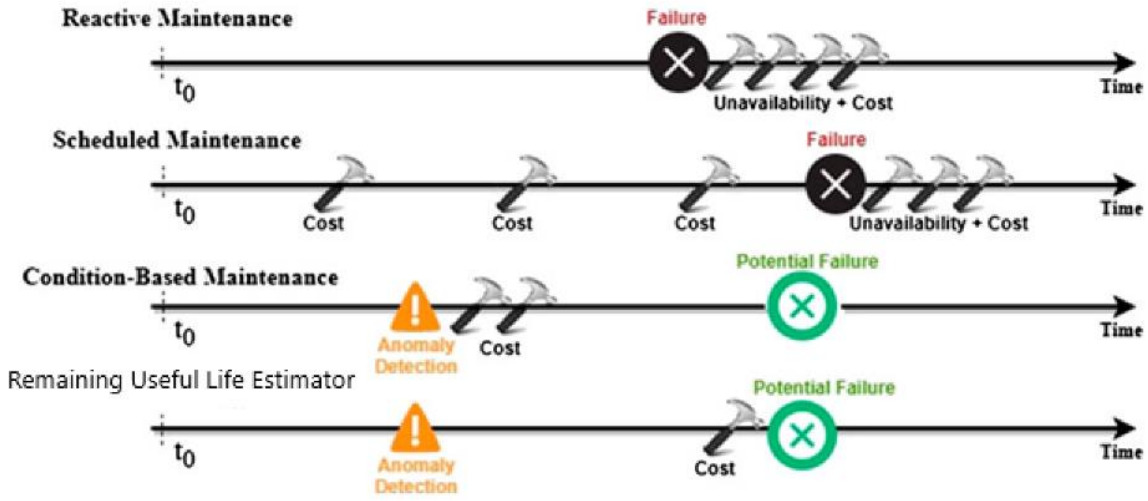


Figure 2: A graphical representation of different maintenance types [6].

Figure 2 states the failure is predicted by the use of anomaly detection, but this doesn't have to be the case. Other ML methods may also be used to predict such a failure. Important to note that in some more recent literature, remaining useful life estimation is sometimes called Predictive maintenance. In this definition CBM is not part of PdM but its own strategy. Since the earlier specified definition is what seems to be most used within CPP, this definition will be used for this report.

2.3 Model knowledge for predictive maintenance

This section will be divided into two subsections. First, the metrics to quantify the performance of models will be selected and explained. These performance metrics are universal for all models of the same kind. Next, some of the used models will be explained in more detail.

2.3.1 Model types

Now it is clear what PdM is, the commonly used techniques have been explored. There are a few different approaches that are used to predict failures, namely physical models, knowledge-based models, and data-driven models [7], [8]. Physical models predict the workings of a system based on

the workings of the underlying physical systems. For example, specific degradation laws and principles. Knowledge-based models predict systems by using expert knowledge and designing systems to mimic this. Examples of this are fuzzy logic and expert systems [9]. Data-Driven models have recently been thriving enabled by the advancements in the fields of ML and Artificial intelligence (AI). They use statistics, pattern recognition, AI and models based on ML.

Within the data-driven models there is a choice to be made regarding what approach to choose. This is generally split into the statistical approach and ML approach (and sometimes physical models). In recent literature ML models are more favoured. The field is still developing, this gives rise to plenty of recent papers on the topic. Furthermore, there are studies that show that ML-based models outperform statistical models [10].

ML can be divided into two categories, supervised and unsupervised ML. Supervised ML is when a model gets a lot of input and what outputs it should produce. Then the ML model needs to learn the relations between the input and the output.
Unsupervised ML does not have any labelled data, and thus learns with a *'reward algorithm'* [11]. For example, the photo app on most modern phones can automatically detect what faces are in a photo. It then makes albums of all photos where that person is in the photo. This is a clustering technique that works without labelled data and is an example of unsupervised learning [12].

So, the big difference between the two is the availability of labelled data. Depending on the task either method might be preferred, supervised learning is mostly used for regression or classification tasks. Unsupervised learning can be used for clustering or association tasks. Certain tasks like anomaly detection can be performed by both kinds of models [11].

These ML models regardless of how they work have a goal they want to achieve. In PdM the most used tasks are regression, classification, and anomaly detection. Regression tasks are tasks where a model predicts future values, for PdM that usually is the remaining useful life. The second kind of task is classification, where the model classifies failure or no failure. The third kind is anomaly detection, where a model classifes certain values as anomalies. Anomalies are significant deviations from a standard condition. The idea is that this deviation is caused by failure and/or degradation. Thus, anomalies should be an early warning sign of an imminent breakdown.

2.3.2 Performance metrics

To be able to quantify the predictions made by models and to compare them in a meaningful way metrics are needed.

Mean Absolute Error (MAE) is for regression tasks only. MAE takes the absolute difference between an observation and prediction, and averages that over all points. The formula for MAE is shown in formula (1) [13].

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} |x_i - \hat{x_i}| \qquad (1)$$

In this formula the variables are as followed:
MAE – Mean Absolute Error [-];
n – Number of samples [-];

i – Sample number, $i \epsilon (1, 2, 3, \dots n)$ [-];
x_i – Observation i [-];
$\hat{x}_i$ – Predicted value i [-].

MAE is a regression metric, but there are also other models. Binary classification models are models that don't have a numerical answer but just make a distinction. They can only answer yes or no. For PdM the question would be: 'Is maintenance required in the next *[timeframe]*? ' . The timeframe can then be chosen freely.

Classification also needs a way to compare between models. For this first some definitions must be established. A binary classification algorithm can only make 2 kinds of predictions. Based on if those predictions were correct 4 categories can be created which have been visually represented in Figure 3. A model can predict positively and when it is right it is called a True positive (TP), however when it predicts a positive and is wrong it's called a False Positive (FP). This same logic can be applied to negatives with True Negatives (TN) and False Negatives (FN). Figure 3 is called a confusion matrix and can quickly visualise how well a model performs [14].

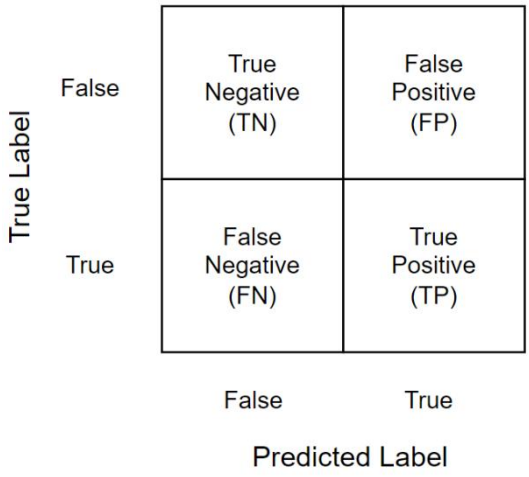


Figure 3: Confusion matrix for classification tasks.

The abbreviations in Figure 3 are generally replaced with the values of the algorithm. These values can then be used to describe some useful metrics. You could for example look at accuracy, which is the percentage of predictions did the model get right. Accuracy is described in Formula (2) [15].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

In this formula the variables are as follows:
$Accuracy$ – The percentage of correct classifications from 0 to 1 [-];
TP – The amount of true positive classifications [-];
TN – The amount of true negative classifications [-];
FP – The amount of false positive classifications [-];
FN – The amount of false negative classifications [-].

Accuracy seems like a fair metric. However, it has some problems. For example if the dataset has a class imbalance, meaning that the dataset has considerably more negatives or positives it describes the performance poorly. Say that a dataset has 99 negatives and 1 positive in its training set. If a model would only predict negative this algorithm would have an accuracy of 99%, despite not predicting a single positive. In this case positives are likely the class that needed to be predicted, so

the model fails in predicting the target class. Accuracy is dangerous because a model can appear to be doing well while it is not [15].

To combat this other metrics can be used, below Formulas (3-5) show a few of these metrics [16] [17].

$$Precision = \frac{TP}{TP+FP} \quad (3) \qquad Recall = \frac{TP}{TP+FN} \quad (4) \qquad F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall} \quad (5)$$

Formula (3) is for precision, it is a measure for what proportion of the positive predictions are correct. A precision of 0.5 means that half of the positive predictions are actually positive.
Recall (sometimes called True Positive Rate) in formula (4) is a measure of what proportion of real positives is predicted correctly. A recall of 0,7 means that it will correctly classify positives 70% of the time [16]. Precision and recall are used because they describe a system's ability for classifying positives, which is generally the goal of the classification algorithm. There are similar metrics for negative classifications, but those will not be discussed.
If both metrics are of equal importance, the F1-Score in formula (5) is used. The F1-Score is the harmonic mean of the precision and recall [17], [18]. This means that it will be high if both precision and recall will be high, low if they are both low and somewhere in between if one is high and the other is low. F1-Score is especially useful for unbalanced datasets where metrics like accuracy are not reliable.

2.3.3 Models used in predictive maintenance

The three basic ways ML is used for PdM are: anomaly detection, classification, and regression. Regression tasks are commonly used for Remaining Useful Life (RUL). RUL is generally more complicated than the other two approaches and often involves unsupervised ML. The main advantage of RUL predictions is more lead time for planning and saving wear on new equipment. Because of the low costs of the equipment studied RUL is not as valuable. The choice has thus been made to not select any RUL approaches.
From literature 4 approaches were chosen: a process-based model, an autoencoder anomaly detector, a time-series classifier and an aggregating time-series classifier. What these approaches entail and which specific models they use will be described in the upcoming section.

2.3.3.1 Process-based models

Process-based models have been defined as models that use process knowledge to analyse and predict the values of the process and seeing if these deviate. They include parameter estimation and observer-based fault diagnosis [19]. Observer-based fault diagnosis in particular tries to use previously observed behaviour to create a signal that can describe faulty events [20]. This is commonly done with residuals but can also be done using other methods.
The advantages of this approach are that it is not very computationally heavy, it is simple, and well explainable. With a bit of system process knowledge, a few approaches can be thought up and tested rapidly. However, if the process or some of the equipment changes these models might need to be redesigned completely. Its simplicity also means that it is hard to capture high complexity, it is limited by human knowledge, thinking, and acting. Ideally, a system would not think like a human so that it can see what humans cannot.

2.3.3.2 Neural Networks

Artificial Neural Networks (ANN) are at the basis of both the autoencoder and the time-series classifier. This section will describe how they work.
An ANN is a type of algorithm inspired by the working of the brain. An ANN works by having neurons that are connected. A group of neurons is called a layer, within that layer neurons don't have connections with each other only with neurons of the previous and following layer. A signal starts at the input layer and ends at the output layer with a possibility of multiple layers in between. The layers that are not the in- or output layers are called hidden layers. Figure 4 shows what layers within an ANN look like. An ANN with at least 2 hidden layers, so including in- and output layers a total of 4 layers is sometimes called a deep neural network or is referred to as deep learning. The connections between neurons have weights assigned to them. The ANN gets better at its task by gradually changing these weights until the performance improves. The weights can be seen as the 'importance' of a value or combination of values. Figure 4 shows a simple ANN with 3 layers. The thickness of the line from the hidden layer to the output layer represents the weight of the connection.

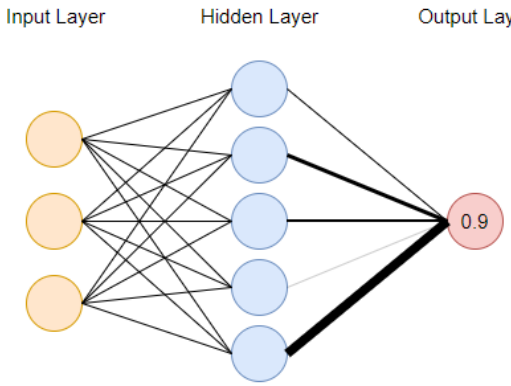


Figure 4: A simple Artificial Neural Network with 3 layers. The thickness of the lines represents the weights of the connections between neurons.

A system of linear functions added up will create a bigger linear function [21], however it is desirable for an ANN to learn non-linear behaviour. ANN have something called an 'activation function' in every neuron. This activation function can be active or inactive based on the input. These activation functions can be used as another parameter of ANN that can be experimented with. Some activation functions will perform better than others based on the application.

ANNs are used widely and have applications in regression tasks, anomaly detection, classification, and computer vision systems. ANNs have already been used directly for CBM and to predict RUL. They are also the basis for more complex ML models like Autoencoders [19], [22].

2.3.3.3 Autoencoders

An Autoencoder (AU) is an unsupervised deep learning ML algorithm generally used for dimensionality reduction, denoising of data and image processing [22]. AU's goal is to learn how to replicate the input values that are being fed into it. This is accomplished by using an ANN with its data as input and target. The layers of the ANN get smaller first and then grow back to their original size. The reduction in layers size makes it so that the network can't directly connect the input to the output. This way the AU learns the most important features of the input data. It does this with three

basic parts, the encoder, the code in the middle and the decoder. These three parts are represented in Figure 5. Because it only learns the most important features it can denoise data or reduce the dimensionality of the data by removing the decoder part.

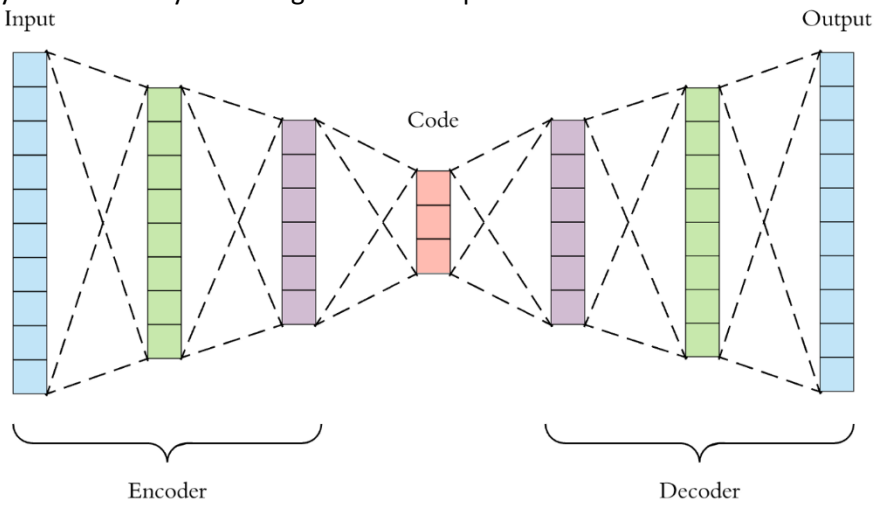


Figure 5: A representation of an autoencoder [22].

In recent literature AU have also been used in the field of PdM [23] [24]. These studies specifically chose Deep Convolutional AUs. These AUs use at least one convolutional layer and one deconvolving layers used in an ANN. The AU's learns the signal the machine produces, and if it strays too far from how it expects the signal to behave it can be classified as an anomaly. This way AU can predict failure in advance by recognising consistent deviating behaviour over a longer period of time.

2.3.3.4 XGBoost

XGBoost stands for eXtreme Gradient Boosting and is a supervised ML ensemble learning algorithm that can be used for classification and regression tasks. Ensemble learning means that it uses multiple learning algorithms to make its predictions, more specifically using a method called boosting. With boosting the predictions of a previous learning algorithm and its right and wrong classifications are fed into the next algorithm. This way algorithms further along the chain are better at learning the cases the earlier algorithms had wrong. For XGBoost the smaller learning algorithms it uses are decision tree [25]. Figure 6 is a simplified graphical representation of how XGBoost works.

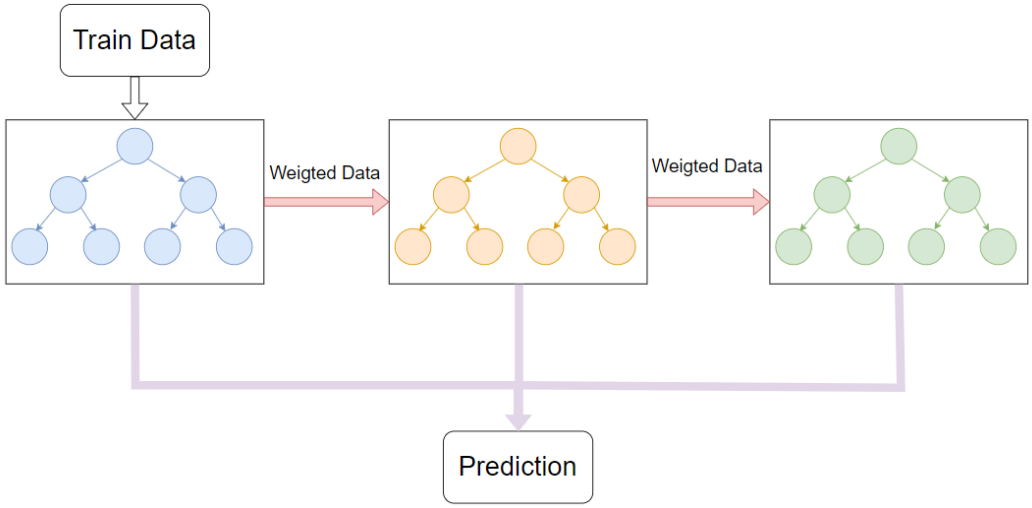


Figure 6: Graphical representation of how XGBoost functions.

2.4 Data preprocessing techniques

Not all models can learn from raw data. Sometimes it required to transform data or to derive insights from them so the ML algorithm can learn from this. The XGBoost algorithm from the previous section for example needs its data transformed.

In data analysis and ML *'feature engineering'* refers to the process of using observations and application specific knowledge to gain additional insights from the data. In ML this is commonly used to learn a relationship to a ML algorithm which it would probably never have discovered itself. These features generally do a better job of predicting the outcome than ordinary features because they contain background knowledge. Common feature engineering techniques include scaling, dividing (fractions), subtracting, aggregation (or grouping), clustering, imputation, categorial encoding and Principal Component Analysis (PCA) [26], [27]. These techniques will be discussed below.

Normalisation

To properly compare the things like flow, with typical values of 150 kg/min, and pressure, which stands at 3 bar, scaling will need to be applied. This is even required for some ML models. These models generally make very small steps and consider all value steps of equal importance. So a difference of 3 bar and 3 Kg/min would be considered just as important, even though one of them is not significant at all. There are multiple methods for scaling, here the standard score (or Z-score) is used [28]. The formula for the Z-score is given by Formula (6).

$$Z = \frac{x - \bar{x}}{\sigma} \qquad (6)$$

In this formula, the variables are as follows:
Z – the Z-score for x [-];
x – A value sampled from the population [-];
$\bar{x}$ – Average value of the population [-];
σ – Standard deviation of the population [-].

The Z-score in this case is just a measure of how many standard deviations a value is away from its mean value. Do not confuse this with a normal distribution, though this is lent from statistics the data is not normally distributed so this measure is useless in that context.

Time Aggregation

Another feature engineering technique that's used is time aggregation, more specifically aggregation of a time series. Time aggregation in this context means the grouping of all data points within a specified time period. This allows the calculation of multiple statistical parameters over this period, like mean, standard deviation, or maximum value [29]. For this dataset the choice has been made to aggregate over the period of an ink batch. This way every subset is sure to contain the same processes. Batch groups can be used instead of needing to work with a time series.

Principal Component Analysis

Feature engineering tends to create a lot of features. Though all features have the potential to help in making a prediction, too many features could reduce performance. This is also called 'the curse of dimensionality', in this context every feature adds a dimension [30]. To help reduce this influence, there are a few dimensionality reduction techniques. One that is often used is called Principal Component Analysis (PCA). PCA works by taking all the data and looking into multidimensional space to what dimension or combination of dimensions has the highest variance. Another way to think

about variance is signal. If data has no variance it contains no information about the goal and thus no signal. PCA starts with the strongest signal and then continues with the second strongest and so onward. What remains is a list of the strongest signals, this list is guaranteed to have fewer features than the input data. Another advantage of the PCA algorithm is that it only gives out uncorrelated data. This is useful in preventing multicollinearity, which is when multiple correlated predictors cause them to behave in an unstable way [31].

Weighted Training, over- and under-sampling

A big problem that is universal in PdM ML problems is class imbalance. In classification, a class is a category the result can fall into. So failure would be one class and no failure would be another class. The average machine is more often in working order than it breaks down. The amount of normal operation data will greatly outnumber the amount of failure data. The failure class will be considerably smaller than the normal operation class, this is a class imbalance. The class with considerably fewer data points is in this context referred to as the minority class. ML models tend to give each data point equal weight, this means the model won't get very good at recognising the minority class. What makes this worse is that for classification problems accuracy is often used as a metric. As briefly touched upon in section 2.3.2, accuracy gives a bad impression of the performance of the model because the minority class is the class that needs to be predicted well. Worse even is that some models use validation data in their training process to train the best model, if the model uses accuracy as the metric to monitor the performance it can make the model perform even worse.

Untreated class imbalance will likely lead to models that are unable to predict the desired class. To treat this problem a few approaches can be taken. These are:

- Over-sampling the minority class;
- Under-sampling the majority class;
- Weighted Training.

Under- and Over-sampling both work based on the same basic concept. That is, the ML model will evenly weigh both classes as long as they are evenly sized. Under-sampling does this by discarding datapoint from the majority class, thus making both classes equal in size. This is generally not preferred because this discards information.

Over-sampling aims to create more of the minority class to balance. In the best case this is done by taking more measurements of the minority class. In most cases this is not an option, in that case over-sampling can be done by duplicating some data points or creating new ones. Synthetic Minority Oversampling TEchnique (SMOTE) creates new data points by using the old ones [32]. In essence it creates new data points on the line between two existing data points at random. It does this as many times as is necessary until both classes are the same size.

Weighted Training works by adding more weight to certain outcomes. By default, misclassifying a majority or minority class are both weighted equally as 1. However, it is possible for certain algorithms to assign more severe weights to undesirable cases, like misclassifying a minority class. If the sum of the weights of both classes is equal, they will equally impact the way the model learns [33]. This should in turn improve the results.

One-hot encoding

Not all data can be expressed numerically, take nationality for instance. For simplicity's sake let say a participant is either American, Canadian, or English. Since these are categories it's hard to express them as numbers. However, it is data that adds something to a model, so it does need to be included and for ML it needs to be numerical (most of the time). It's possible to just give them an ID number,

but this doesn’t work well in certain ML models. This will inherently give them an order even though their nationality is equal. Besides, if someone were to have a double nationality the system would probably be unable to handle that. To encode this kind of categorical data One-hot encoding can be used. One-hot encoding simply creates a new Boolean column for every unique instance of a category [34]. The value of this column will be True if that row is part of the category the column encodes and False if it does not. This way categorial data can be fed into a ML model.

3 Methods

This chapter will describe the methods used, will go into further detail on the dataset and give more information on tools used.

3.1 Dataset

The dataset originates from the SCADA system that controls the latex-based ink plant. A part of the data recorded here is also stored in the production archive for analysis by the production engineers. The data used is fetched from the production archive. Data archiving started on the 10th of April 2022 at 12:00. March the 6th 2023, 12:00 has been chosen as the end-date for the dataset. The database stores an entry every minute, so over the course of 330 days this totals 475,260 rows of data. In these 330 days a total of 1119 batches have been completed.

The database consists of a total of 2947 unique sensor entries. The first and easiest selection step is only selecting the data from the Latex ink plant, this leaves 524 sensors. Since only the AODD pump has to be investigated, only that data has been selected. This leaves a total number of 55 sensors.

These 55 entries have all been kept in in the initial data-analysis. This is because 55 is a manageable amount, so there is no reason to slim down any further. Another reason is that it's hard to predict what data will turn out to be useful beforehand.

3.1.1 Failure modes

So far 4 noteworthy failures have occurred. Two of those failures were diaphragm failures, and two were caused by the one-way ball check. The pump's user manual [5] specifies a few possible causes for every type of problem, unsteady flow for example. However, this can also be reversed and used to look at what indicators every type of problem has. That way data analysis can start by looking at those variables first. An overview of these symptoms is shown in Table 3.

Table 3: An overview of the possible symptoms associated with a type of failure

Malfunction	Possible Symptoms	Possible root causes
Diaphragm rapture	- Pump operates unsteadily - Air within the liquid - Pump stops itself - Liquid leaks out of pump via muffler	- Pressure within system - Inadmissible vacuum - icing
Ball Valve worn	- Insufficient discharge pressure - Output decreases - Pump stops itself	- Input pressure too high - Suction pressure too low

This table gives a few hints what to look for. It's obvious for both failure types that the output pressure and flow and their shapes will be important in determining the pump condition. A diaphragm rapture might introduce air into the fluid and pipes, this might be something that could be detected using density measurements in the pipes. The mass flow meters used have density

measurement built in. These symptoms provide a baseline as to where to start to search for possible indicators.

For the possible root causes some equipment engineers have been in contact with the pump supplier. After this contact and looking at the user manual, its highly suspected that applying positive suction pressure is the cause of the premature diaphragm breakdowns. This does not happen during regular operation but probably during the cleaning process. During this process a double seat valve is opened, which allows CIP-fluid and demi-water to be pumped through the pipes. Both the CIP-fluid and demi-water are pumped into the system under pressure, which could cause the positive suction pressure that damages the AODD pump.

3.1.2 Target creation

If supervised learning were to be applied it would need a labelled dataset, meaning that for every datapoint the correct answer needs to be included. The correct answer is often called the target because this is what ML algorithms aim to learn. For PdM the questions are predefined. For classification tasks the question would be *'failure within x batches?'* and the answer would be true or false. For a classification tasks the target will thus be a column of Boolean values. This has been visually represented in Figure 7. This figure shows both how labeling is applied to the dataset, and what the predictions would precisely entail. Something to note is that this approach has multiple predictions for a single event. Thus, a single positive can be ignored relatively safely. If, however 3 positive arise in the span of 5 batches an opperator can be almost sure failure is imminent. This way the model does not need to be as accurate as other models.

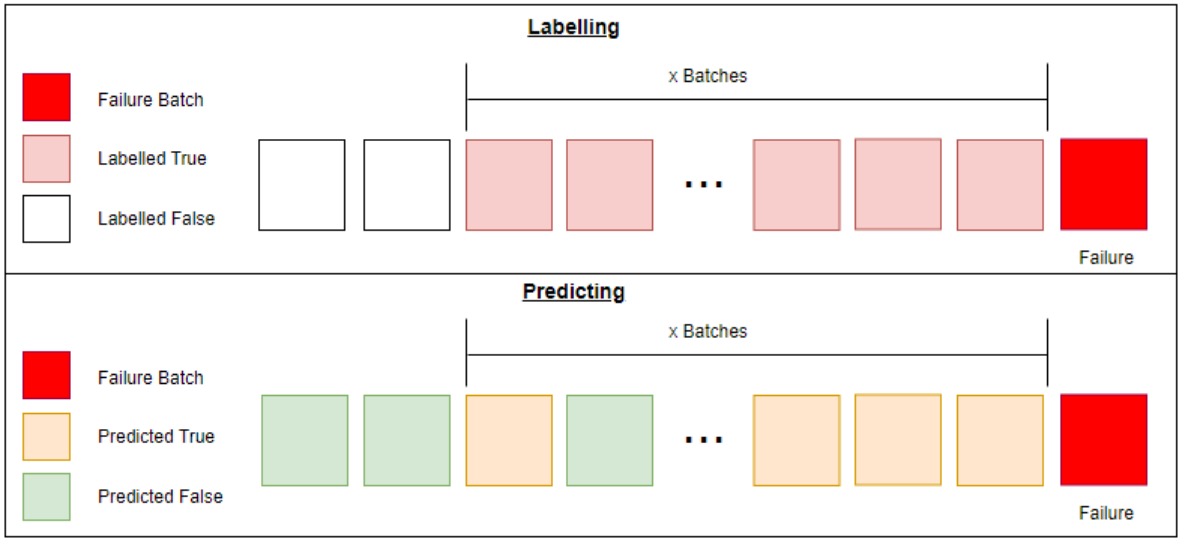


Figure 7: Target creation and goal for a classification algorithm.

The number of batches to look into the future is a parameter that can be tuned. If too little leading time is chosen the mechanics won't have enough time to react. However, if the leadup time is too long the accuracy of the models tends to worsen. For this use case 20 batches has been chosen. On a good 24-hour day about 10 batches can be produced, this would give 48 hour reaction time at least.

With the 2 ball check valve failures 40 batches are labelled as True. Since the dataset contains 1119 total batches this leaves 1079 batches labelled as false. This means about 3.6% of batches are labelled as true, this is a hefty class imbalance. This will need to be addressed when training the models.

3.2 Model validation & testing

It's important to validate how well a model generalises to new data. If the model only performs well on train data but not on new data its use is limited. Thus, the model needs to be validated using new data, meaning data that has not been used when the model was constructed. This new data can be used for two purposes, model optimisation and testing its performance. If the same data were to be used for this, you would optimise based on this data. In doing so, lose an unbiased estimator for model performance. Therefore, data that is not used for training must once again be split up into validation and test data.
Validation data will be used during model construction and other things like feature selection to see how certain actions affect the model performance. This, however, will give a biased view of the final performance. This is because during training the goal is to improve the performance of the validation data. To see if these performance improvements turned out to be universal and not just case depended, test data is required. Test data is used as the last step once model optimisation has been concluded. The model's performance on test data is the final and unbiased estimator of a model's performance in the field.

The easiest way to split the data into train data and validation/test data is called holdout validation. Holdout validation keeps a percentage aside so that it is not used during model building. Holdout validation does have a problem. If it chooses to keep a random percentage it could randomly choose data that makes the model perform better or worse. Holdout validation tends to have widely different outcomes based on the split chosen. Therefore, it is generally only used for the test set and not the validation set.

For the validation set k-fold cross-validation is applied. K-fold cross-validation avoids the problem of holdout validation by using all the data in the validation process. Figure 8 shows how k-fold cross-validation does this. The k stands for the amount of fold that need to be created. Every fold 100/k percent of data is kept out as validation data, a model is trained on the remaining 100-100/k % of data and validated using the validation data. This way all the data (except test data) is used as validation training sometimes, this gives the most reliable validation results [35].

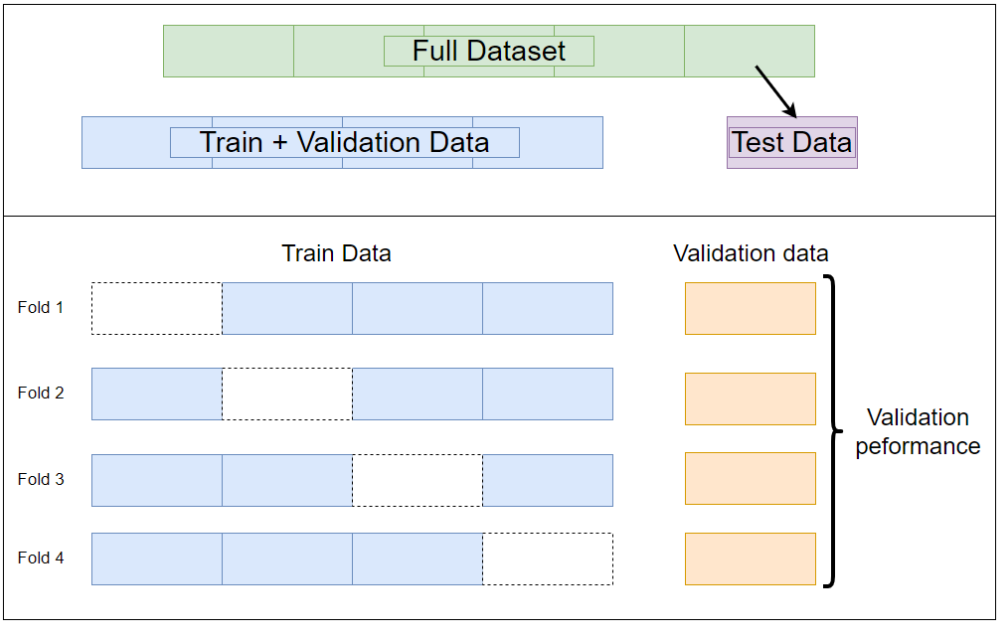


Figure 8: Visualization of k-fold cross-validation.

For this application, the chosen split is as follows: 10% test data, 90% train and validation data. On the remaining 90% 4-fold cross-validation will be applied. Specifically for the classification task this means that 4 true cases are left in the test data and 36 in the validation and train data. With 4-fold cross-validation that means every fold will have 9 true cases in its validation set and 27 in its training set.

Both the holdout and cross-validation make use of stratified sampling. This means that it will not pick data truly random. Stratified sampling will make sure it grabs the same percentage of every class, this way the number of true cases will not fluctuate between runs.

3.3 Experimentation

Based on the literature study the choice has been made to try out 3 approaches, these are:

- Autoencoder anomaly detection using time-series data;
- "Failure-within-20-batches" XGBoost Classifier using statistical batch data (aka. aggregated time series);
- "Failure-within-20-batches" ANN Classifier using raw time-series data.

These models will all be trained on the same dataset and will be pitted against each other to see what model performs best. Only the best performing model, using the F1-score, will be optimised. This is because F1-score is most fitting for imbalanced classification tasks. For this k-fold validation is applied 30 times.

All data is first cleaned the same way. The data is cleaned by filtering in such a way only process data remains. Then the data is split up into train and test data. Then all data is normalised using the Z-score with the mean and standard deviation acquired form the train set. The model specific steps are described below.

3.3.1 Autoencoder

For the AU the data is fed into a Convolutional ANN. The batches are fed into the AU in a random order, one by one. The AU learns to recreate its own input this way. Since it only gets given small time series it is unable to learn long-term behaviour. Any deviation from its learned behaviour is assumed to be caused by behaviour it couldn't have learned. This change in long-term behaviour is assumed to be caused by degradation of the equipment. Thus, an error is an indication of degradation. Bigger error means bigger deviation and thus more degradation has occurred. This is a long-term anomaly detection system. After the data is fitted the MAE can be determined for each batch. The exact structure of layers and settings can be found in Appendix A. The entire code for this model is displayed in Appendix C.

3.3.2 Aggregation Classifier

The batch-ordered time-series data needs to be aggregated. For this the 'tsfresh' Python package is used. Tsfresh automatically extracts over 75 unique feature types, many of which have different settings thus resulting in thousands of features. It does this for every channel, so this can be multiplied with the number of sensors available to get the total number of features. Since this is way too many features it also includes a way of only selecting relevant features. The result is a dataset with the best features for a model. To see all the features tsfresh extracts, see [36]. The ink type is also added as a feature through one-hot encoding. All features extracted and the importance of each

feature in the final model is shown in Appendix B. The code used for training the final model is displayed in Appendix D.

The resulting dataset that's used for the ML model is one big table with on every row a batch, and in every column a feature generated through tsfresh or one-hot encoding. This comes down to a total of 1119 rows and 367 columns as the final dataset.

The target for this approach has been described in section 3.1.2. This is a single Boolean column with a True or False value. Only 3.6% of the dataset is the minority class, thus over- or under-sampling techniques needs to be applied to train the model. In this case SMOTE will be used on the training data, this creates a training set of 50% minority class and 50% majority class. The test set will keep the old proportions since it needs to represent new data.

Like earlier stated this will be fit using the XGBoost model and validated using k-fold cross-validation. Here 4-fold cross-validation will be used because of the limited amount of minority class data.

3.3.3 Time-series Classifier

After the universal data cleaning has been concluded, the time series is fed into an ANN. This neural network gets raw time-series data from multiple channels as an input and only produces a single output. This output is a chance from 0 until 1 that the batch is part of a category. This ANN uses the same target as the Aggregation Classifier approach, so the dataset is just as skewed. Because raw time series are hard to over-sample and under-sampling throws away too much data weighted training is chosen to fix the class-imbalance problem. The model will be validated using 4-fold cross-validation. To see the full model and its settings used check out Appendix A. The full code used for training can be found in Appendix E.

3.3.4 Hyperparameter Optimisation

Every ML model has settings that influence the training process of the model, thus resulting in different models. These settings are often called hyperparameters. These hyperparameters have their own optimal value for which the model performs best, this optimal value is however case-dependent. Optimising these hyperparameters is often worth it since it is a decent performance increase for a relatively small effort.

Here optimisation will commence using "*GridSearchCV*" from the sklearn Python library [37]. GridSearchCV works by simply taking a list of every hyperparameter that needs optimizing and trying every possible combination. It does this by using 5-fold cross-validation at every point and saving the highest value. Hyperparameter optimisation will only be performed on the most promising model, based on the validation performance.

3.4 Programing tools

For every step, from data-acquisition to data visualization a programming language and environment has been used. Here the software tools and approaches chosen are listed.

The data used is stored on a historical SQL database. Within CPP a software program called *'AspenTech Process explorer'* can be used to get data from this database and visualise the data. For acquiring data the Open DataBase Connectivity (ODBC) drivers that got installed with AspenTech

were used. ODBC is designed so that any device could send commands to any database in a standardised way. These ODBC drivers can be used by the programming langue Python 32-bit with use of the *pyodbc* package. This way data could be programmatically acquired from the database and written to a .csv file. This way data only needed to be acquired once, and can be read from the .csv files afterwards. The script also eliminates the need for manual work, meaning that acquiring more or new data doesn't require a time investment. Acquiring data does however still have a waiting time.

4 Results

This chapter will display all results acquired throught the research.

4.1 Knowledge-based observations

First the flow right after the pump is inspected on a average day vs a day before failure to see if any deviating behaviour can be detected. For this the flow and pressure have been transformed using the Z-score. These results can be seen in Figure 9.

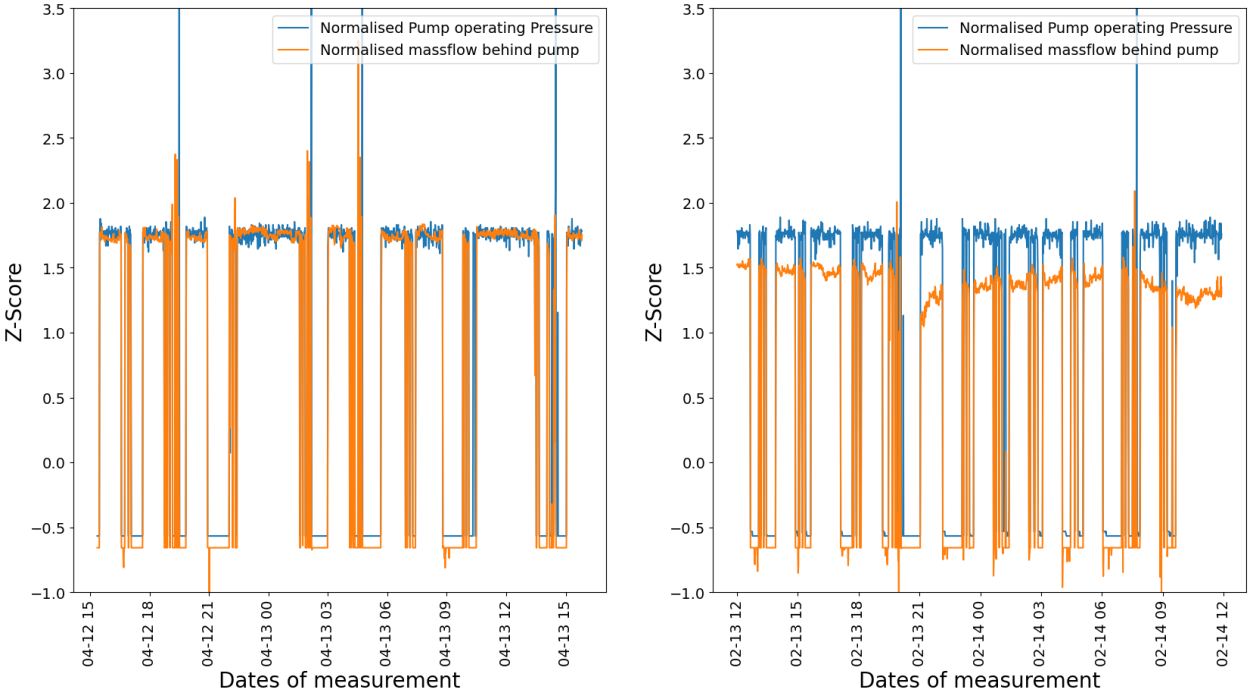


Figure 9: Pump normal operating profile vs a day before failure.

In Figure 9 the blue line is the transformed operating pressure of the pump and the orange line is the transformed output flow. In the left figure the flow and pressure are at the same level, indicating that they normally have the same kind of distribution. However, a day before failure, like shown on the right, the pattern has deviated from its normal setpoint. To further show that this is not a cherrypicked case Figure 10 shows the pump operating after it has been repaired. Once again orange is the transformed flow and blue is the transformed operating pressure. This indicates that after the pump has been disassembled and the diaphragm and ball valves have been replaced the pump operates as expected again.
A similar profile can be seen at the ball valve failure but not at the diaphragm failure. This means that this kind of output degradation is linked to ball valve wear but not diaphragm failure.

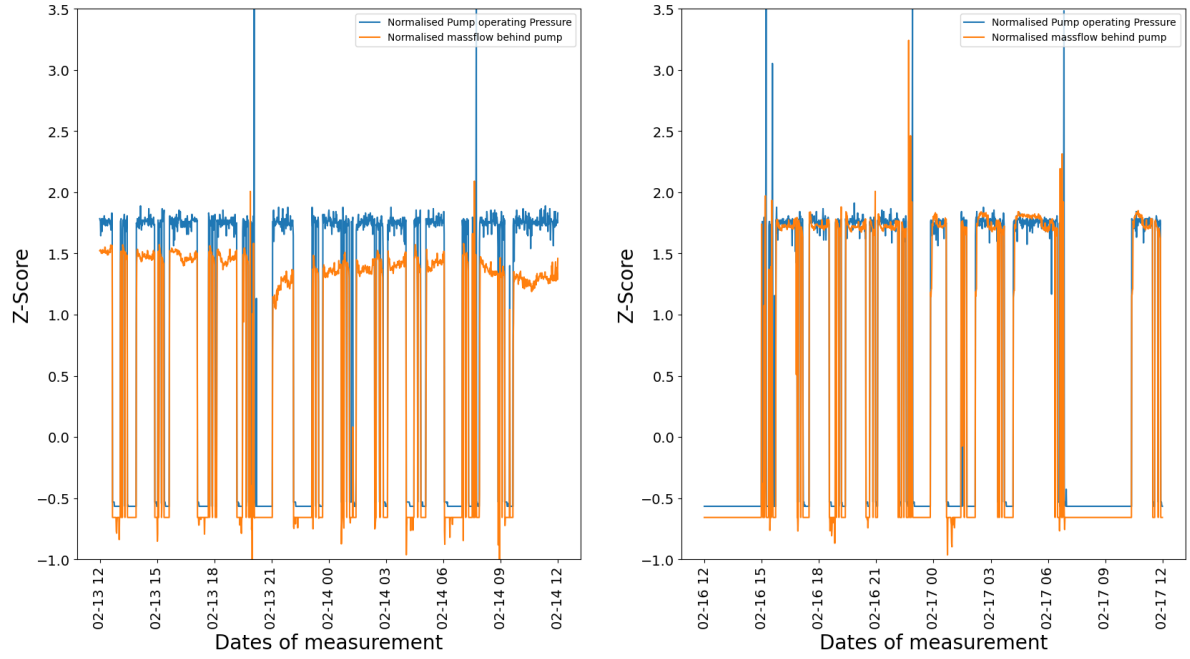


Figure 10: Flow profile behind the pump before and after repairs.

Within the equipment department the suspicion has existed that the cleaning process is damaging the diaphragms of the AODD pump. It would do this by applying a positive suction pressure. Figure 11 shows the driving pressure of the pump in blue, the pressure behind the pump in red and the mass flow in orange during the cleaning process. From this plot it can be concluded that the cleaning process does supply a positive suction pressure. First off at 2:25 there is a pressure peak behind the pump without the pump running, this means an external pressure source must exist. Second, before 2:10 the pressure behind the pump is higher than its driving pressure. Based on the working principle of the AODD pump, this is not possible even if the pump is building up pressure against a closed valve. The max pressure the pump could create would be its driving pressure, which in this case would be 2 bar.

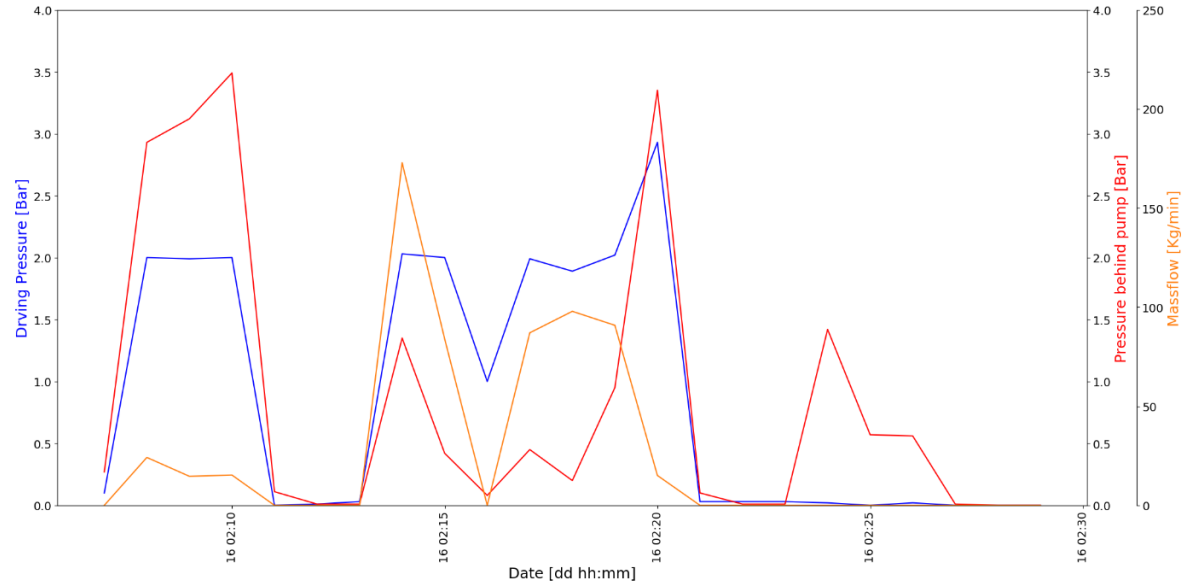


Figure 11: Figure showing pressure and flow during the cleaning process.

The flowmeter is direction sensitive, so if the external pressure source were to be behind the pump negative or no flow would be measured. Figure 11 shows positive flow with pressures the pump

could have never created itself. Therefore, the pressure source must be located before the pump. This proves that a positive inlet pressure is applied to the pump.

4.2 Model building

This sub-chapter will be divided into four parts, for each model respectively. Here, the models' performance on the validation set will be displayed.

4.2.1 Process-based model

Based on observations made, feature engineering commenced. The first approach is to look at a fraction of the pump driving pressure and output flow. From Figure 9 and Figure 10 it is derived that this fraction should be around 1, if however, the flow has decreased compared to driving pressure this would increase to above one. For analysis, extreme outliers have been removed. After this a moving average window of 12 hours has been applied, the result of this is shown in Figure 12. In this figure the blue line is the normalised fraction of the driving pressure of the pump divided by the output flow measured behind the pump. In red is a proposed alarm limit at the 95% quantile value and in yellow are 2 major historical errors shown. In the data two things immediately stand out, the periodic effect in the fraction data and the big slope in December 2022. The periodic effect is caused by the weekends, this is because in the weekends there is no production and thus no production data. The big peak in December is because the plant was offline for a few months for the installation of new systems, this data is not representative of the process and can thus be ignored. The peak in the middle of July is caused by a database fault and is thus not representative either. If these data impurities are ignored the proposed limit can predict one of these two errors at least the week beforehand.

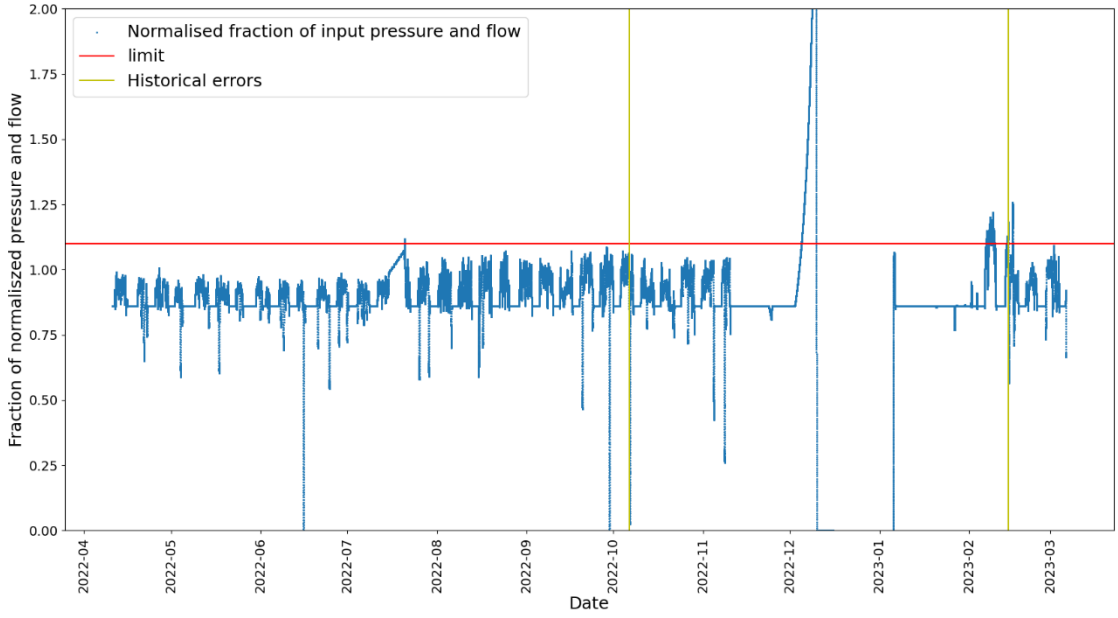


Figure 12: plot of normalized driving pressure divided by output flow.

Figure 13 shows the fraction as described in Figure 12, however it is averaged over an entire production cycle of one batch. This means that downtime like weekends, breakdowns and construction are no longer in the data. To smooth things out a rolling average window of 10 batches has been used. For both the errors, a rising fraction leads up to the failure.

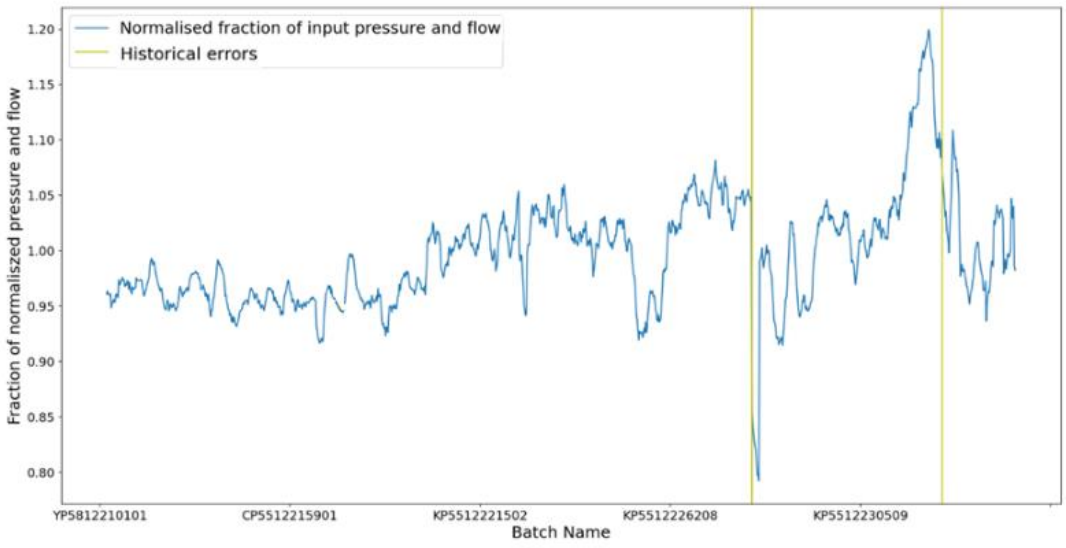


Figure 13: Normalized fraction of driving pressure and output flow, averaged over a batch.

This method has a few advantages. The biggest one is simplicity; this method is very intuitive in its predictions. This means that sceptics of a ML model are not looking at a black box. Secondly, it confirms and reinforces the operators' story who have already claimed that they can see failure coming in advance based on the flow behind the pump. Lastly, this can fraction could also be used as a feature in ML models. Meaning that if this is not sophisticated enough, it might be a strong feature that an ML model could build off.

4.2.2 Autoencoder

The AU, like previously stated learns to reproduce the input data. Figure 14 shows a single time series and how the AU reproduces it.

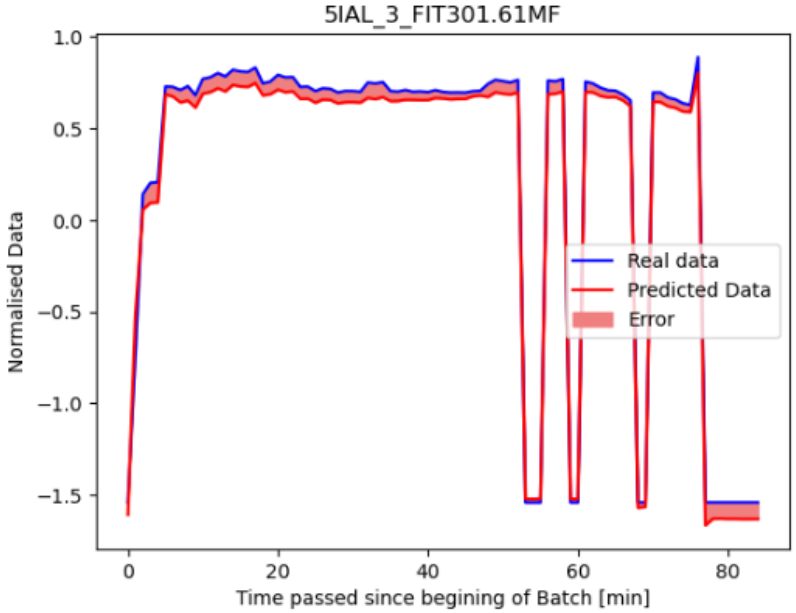


Figure 14: Plot of time-series data and the AU prediction.

The lighter red in Figure 14 is the difference between the predicted and the real value, also called the error. Now the MAE of all time series can be calculated and plotted to see if it is any use. Figure 15 shows this plot, the yellow lines are failure points.

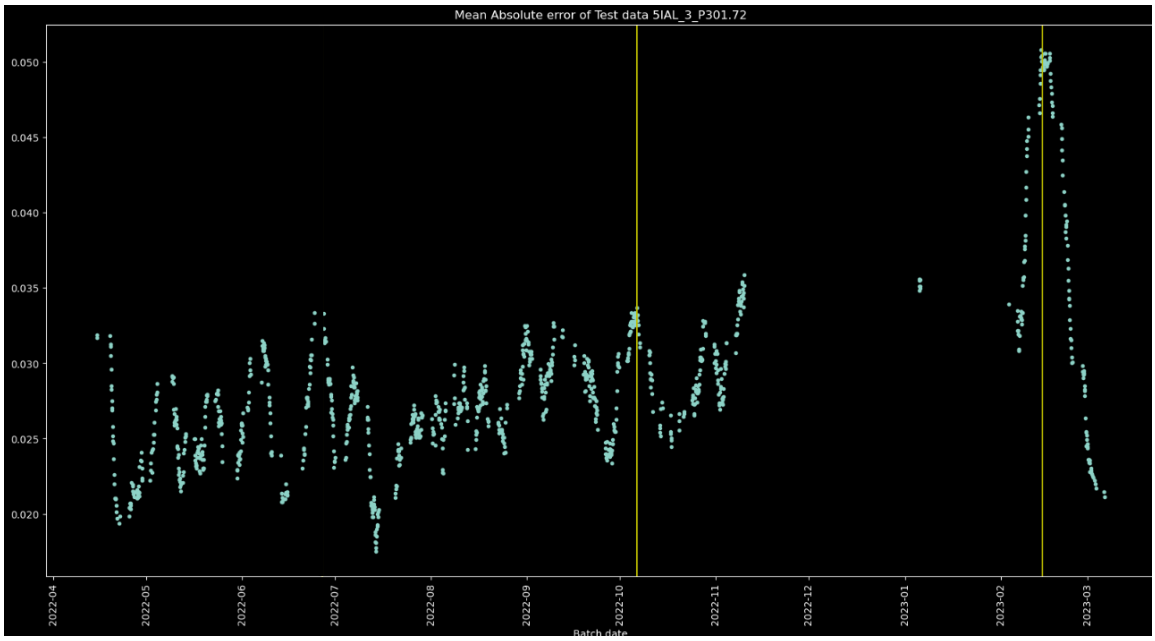


Figure 15: MAE error of pump signal over time. In this, the yellow lines are the failure moments recorded.

Figure 15 shows potential in its failure-predicting capabilities. The first failure however could have been missed. The second failure would have been predictable this way using a maximum value. If this method is only able to accurately identify one of two failures, it is not very reliable.

4.2.3 Aggregation Classifier

Before looking at the k-fold validation performance let's first look at the performance of a single run of the XGBoost algorithm. A confusion matrix like described in Figure 3 is displayed in Figure 16. The test result above this figure is the F1-score. The model is thus able to predict 7 out of 9 of the marked batches. It however has 3 false positives while doing so. 288 batches were predicted to be negative and turned out to be so.

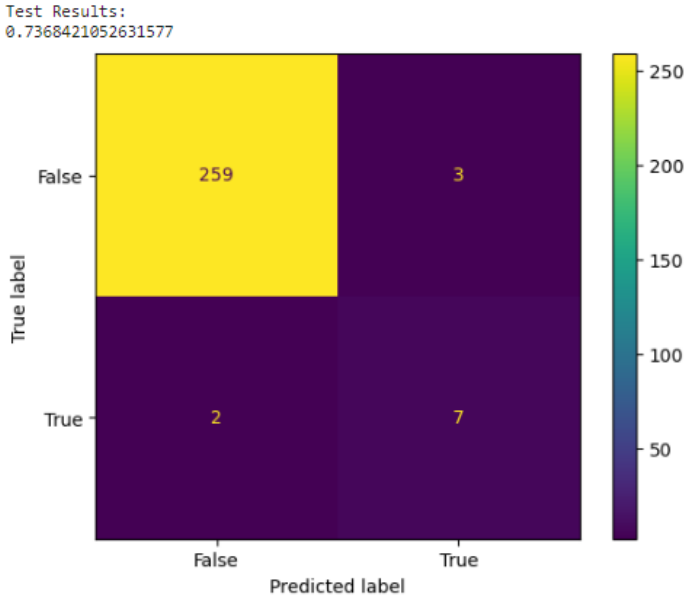


Figure 16: Confusion matrix of a singular run of the XGBoost algorithm.

Figure 16 shows a single run and ML tends to be probabilistic in nature, thus k-fold cross-validation will be applied. Because of the very limited number of positive classifications the choice has been made to use a 4-fold cross-validation. This way every test set contains 10 positives, and every training set contains 30 positives. This seems to be the right balance of having enough training data and not having too few test samples. An example of a run with 4-fold cross-validation is shown in Figure 17.

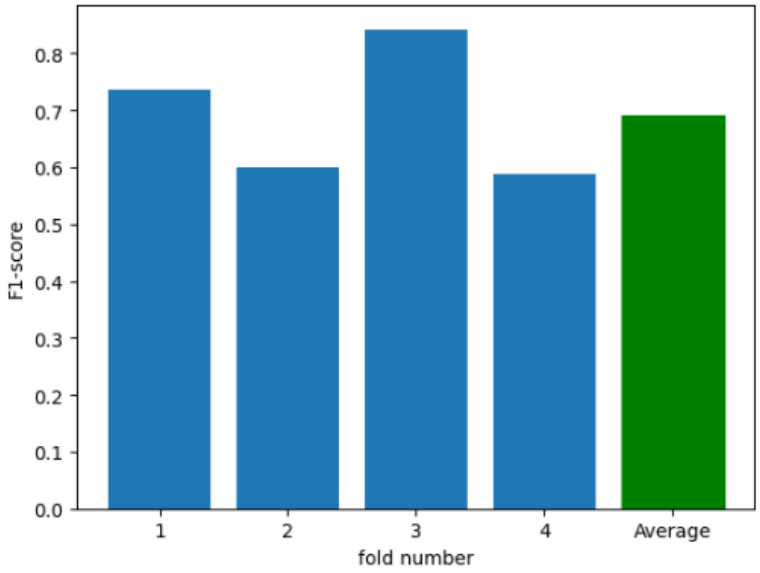


Figure 17: shows an average 4-fold cross-validation run of the XGBoost model.

To make this more reproducible the 4-fold cross-validation has been run 30 times. This resulted in an average F1-score of $\mu = 0{,}75$. The standard deviation is $\sigma = 0{,}04$ between the means of 4-fold run or $\sigma = 0{,}11$ for all singular runs combined.

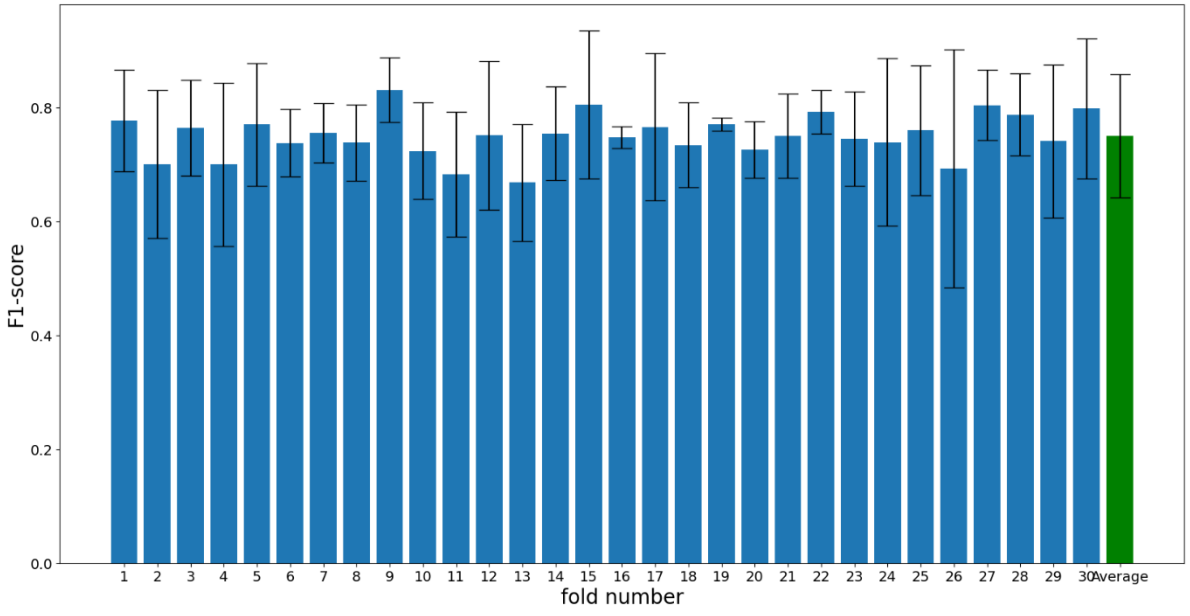


Figure 18: The distribution of 30 runs of 4-fold cross-validation including standard deviation.

4.2.4 Time-series Classifier

Figure 19 shows the confusion matrix of the time-series classifier. It shows that this model just always predicts *False*, it gets nothing right and ends up with a F1-score of 0. This model is having severe problems with the fact that the dataset is so heavily skewed.

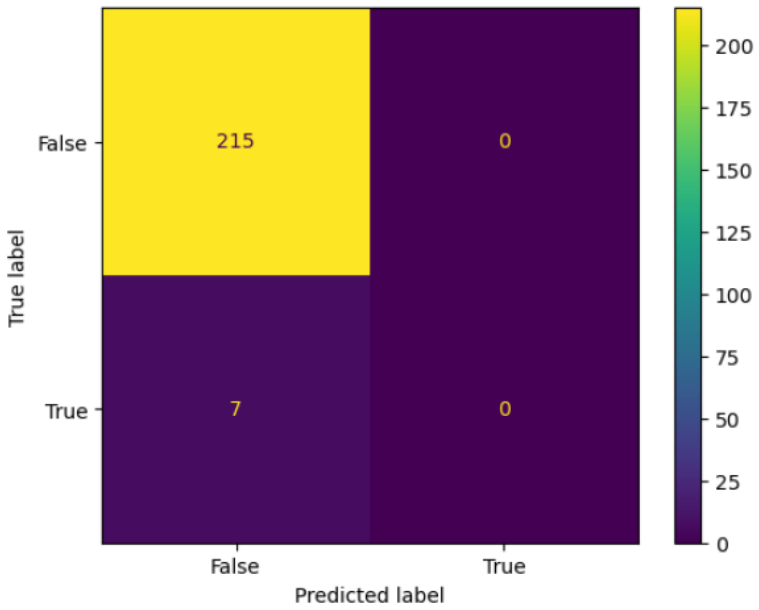


Figure 19: Confusion Matrix of the time-series classifier.

When k-fold cross-validation has been applied the result is consistently an F1-score of zero. This model has thus not been explored further.

4.3 Hyperparamter Optimisation

Since the Aggregation Classifier approach has performed best, the XGBoost Classifier model's hyperparameters are optimised. From literature the 4 most influential hyperparameters have been selected [38]. The exact meaning of these hyperparameters is not important but can be found in the XGBoost documentation [39]. The settings tested by GridSearchCV are shown in Table 4.

Table 4: Table containing four hyperparameters from XGBoost and their values that have been used by GridSearchCV.

Hyperparameter	Value's Tried
max_depth	3, 4, 5, 6, 7, 8, 10, 13
learning_rate	0.01, 0.05, 0.075, 0.1, 0.15
n_estimators	100, 300, 500, 700, 1000
Colsample_bytree	0.3, 0.4, 0.5, 0.6, 0.7, 0.8

The combinations shown in Table 4 give a total of 1200 different sets of settings. These settings were scored using the F1-score with 5-fold cross-validation.

The resulting optimal parameters were:

- max_depth: 3
- learning_rate: 0.075
- n_estimators: 500
- colsample_bytree: 0.8

With these hyperparameters the XGBoost model is scored again using 4-fold cross-validation 30 times to see if the performance increase is significant. The result is a mean F1-score of $\mu_2 = 0.79$ and a standard deviation of $\sigma_2 = 0.11$.

To test if this difference in mean is statistically significantly an one-sided independent sample t-test is used [40]. The null hypothesis and alternative hypothesis are:

$$H_0: \mu_2 \leq \mu_1 \quad | \quad H_1: \mu_2 > \mu_1$$

This means that the ground assumption is that μ_2 is smaller or equal to μ_1, meaning that the optimisation didn't not have make the model perform better. If the null hypothesis were to be rejected that would mean that μ_2 is significantly larger than μ_1.
With a sample size of $N = 120$ and a confidence interval of $\alpha = 0.05$ the t-test has been performed. The right-tailed p-value for this test is 0.003, this is smaller than 0.05 so the null hypothesis is rejected. This means that the optimised model Is significantly better than the unoptimized model.

A power analysis has been conducted to determine the minimal number of samples needed to have a power of 0.8. A power of 0.8 means there is a chance of 0.2 of a type II error. It is called a type II error when a false null hypothesis does not get rejected [41]. In other words, it gives an estimate of whether a certain effect could ever be accurately determined by a statistical test. The general consensus is that a power of 0.8 is sufficient, so the minimal number of samples to achieve that power has been determined using: $\mu_1 = 0.75,\ \mu_2 = 0.79,\ \sigma_1 = \sigma_2 = 0.11$ and $\beta = 0.8$. From this it has been determined that the minimal number of samples required per class is $N = 119$. 120 Samples were taken; thus the power is sufficient.

4.4 Test set peformance

After optimisation the test data has been used to give the final unbiased performance of the XGBoost model. The other 2 models did not perform well on the validation data, thus comparing them to the test data is of no use. Figure 20 shows the performance of the XGBoost model on the test dataset. This gives a final F1-score of 0.86, which is better than the validation performance is.

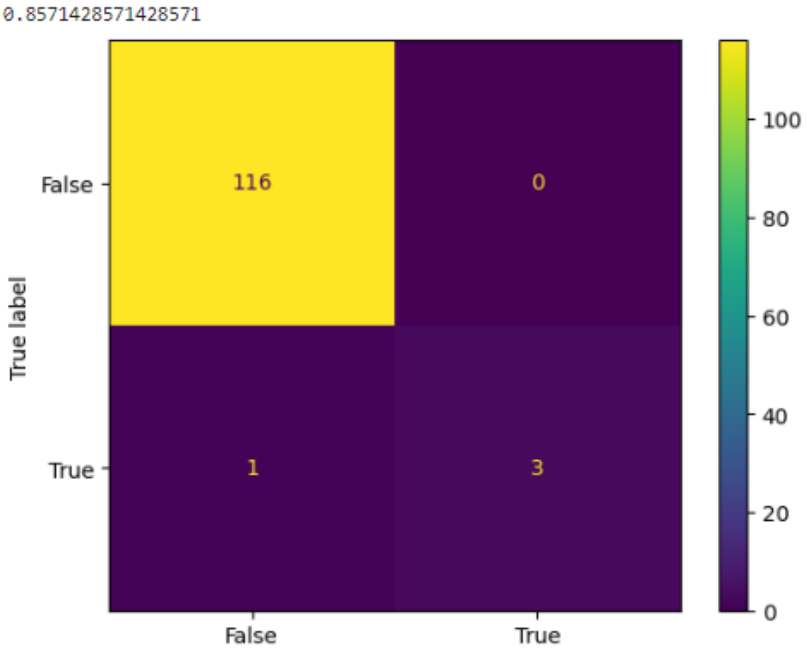


Figure 20: Test results of the XGBoost model.

5 Discussion

5.1 Knowledge-based observations

Based on the observations during the cleaning process it can be concluded that the pump is indeed supplied with a positive suction pressure. But why is this so bad for the pump? The pump can handle an operating air pressure of 7 bar [5], why would 3 bar inlet pressure damage it? The diaphragms in the pump are made from Polytetrafluoroethylene, commonly called PTFE or Teflon [42]. A different manufacturer of AODD pumps states that PTFE diaphragms can only handle a positive suction pressure of about 0.6 bar [43] [44]. This is because when the pump is supplied with higher pressures, water hammer is more likely to occur and does more damage.

Water hammer occurs when an incompressible flow is stopped abruptly. Since the flow is incompressible there is nowhere for the energy to go. Consequently, a shock wave is created within the liquid. This shockwave can bounce around in the pipes a few times causing damage. One source states that this pressure peak can be up to 4 times the input pressure [43] [44].

Because the check valves slam shut, they always create water hammer on at least one side of the valve. When the input pressure is too high these water hammer shocks will be more severe, thus damaging the pump. These high peak pressures are the root cause of the diaphragm's failure.

Preventing this can be done in one of two ways: First is to stop applying a positive suction pressure. This can be done by changing the cleaning process. The mixing vessel is currently cleaned with water which is immediately discarded into the waste line. If instead the pump could pump this up no positive pressure would be applied. Alternatively, the check valves that cause the water hammer could be changed or disabled. Different valves that don't slam shut wouldn't alter the workings of the pump but would lessen the effect of the water hammer. Disabling the ball valve during cleaning would work, this could be done with for example a manual override or magnets to lift the balls. The magnets are an optional addition to a different model of the current AODD pump; however this is intended for draining purposes. It is not known if the balls would stay in place with flow in the pipes.

Though originally one of the goals in the end the decision has been made to not build a model to predict the diaphragm failure, this is for a few reasons. First reason is that a diaphragm monitoring sensor is already in storage, this sensor works by using conductivity to detect any liquid leaks of the diaphragm. This is already a very reliable way to detect failure's and will likely cost less time. Second reason is that the pump manufacturer has provided a reinforced diaphragm that should be able to withstand the input pressures. This would mean that monitoring the diaphragm is not as critical anymore.

5.2 Models

This section will discuss each modelling approach individually to avoid confusion.

5.2.1 Process-based model

The process-based model faces a few problems. The first problem being that it does not account for the current ink type. A few different ink types result in a significantly higher output flow produced by

the AODD pump. Since the driving pressure of the pump is consistent this would lower the fraction that is used. So, when certain ink types are being produced this method would not be able to predict failures. Apart from this problem, the whole model is rather unsophisticated. The model is unable to look at anything else but flow and driving pressure. Since driving pressure is mostly constant anyway this model is about as good as placing an alarm when the flow gets under a certain threshold. Though this would likely already be an improvement on the current situation, this is not something that would fall under predictive maintenance and is not likely to predict any complex problems.

5.2.2 Autoencoder

The AU has shown its potential in the results but due to it not being able to predict one of the two failures it has been discarded as an option. The appeal of the AU is that in theory, it doesn't need any expert information about the system or data pre-processing, just a single strong condition-indicating signal.

There were a few problems with the AU. The biggest one is that it had no way of knowing what kind of ink is being produced. The AU model would have a higher error for these ink types. Since production tends to make a few batches of the same ink type at a time, it appeared as if a failure was coming. The aggregation classifier is able to account for this thanks to one-hot encoding, but the Autoencoder had no way to do this. An attempt at one-hot encoding has been made by feeding the Auto-encoder additional time series with 1's or 0's for its entire length. This approach only barely improved the performance and wasn't worth noting.

One way that could have fixed this problem is by making a separate AU for every ink type, or at least for all types with roughly the same behaviour. This way each model would learn the behaviour of its own ink type. This would however result in a rather complex model that might be hard to troubleshoot.

5.2.3 Aggregation Classifier

The aggregation classifier has been shown to work well and lead to an average F1-score of 0.75 on the validation set. It is however the most labour-intensive process of the 3 approaches. The other two approaches create their own features, but for this approach, the features need to be created manually. If this approach were to be scaled to more equipment this would end up costing more time.

Noteworthy is the fact that the model performed better on the test set than on the cross-validation. This is likely due to a random split that happened to be easy for the model to predict correctly. The reason holdout is not used for the validation data is because of its randomness, it is thus no surprise that the test set is going to have some randomness in it. The standard deviation of a non-cross-validation run on the validation set is 0.11, so 0.86 is exactly 1 standard deviation away from the mean. This is thus not very worrisome.

To prevent the randomness described the test set is sometimes curated before training begins. A few datapoints are then explicitly collected to be a good representation of a few different scenarios or cases. This has not been done in this case, thus the 4 cases it needed to predict might have been too easy which would mean that the test score won't be representative. For future research the recommendation is to curate the test set to be sure it can be trusted completely.

5.2.4 Time-series Classifier

The Time-series classifier has been shown to perform poorly in this use case. When examined further the model also doesn't seem to predict any of the train data correctly. If it cannot correctly classify its trainings data, the problem cannot be overfitting. This leaves only a few possible explanations for why the model performs so poorly. Either the model doesn't contain enough complexity to predict the target or the class imbalance confuses training.

The first possibility is too little complexity. If the model is unable to gain any information from the data to predict the target it would explain why the model performs so poorly. This lack of complexity can come from one of two places, the data or the model. Since the aggregation classifier model is performing well it can only be assumed that the data is not the problem. Theoretically, this model should be able to extract a lot of the same features that the model uses. Thus, if the problem were to be a lack of complexity it originates from the model itself. The model currently consists of a total of 6 layers of which 2 are the in- and output layers. This should generally be enough complexity. The class imbalance still seems to be the biggest problem for the model.

Weighted training like it has been applied does not seem to have been enough to solve the imbalance issue. Different weights show different behaviours, but none of it seems desirable. By the process of elimination, this seems to be the most likely cause. To fix this other imbalanced learning techniques could be considered. Under-sampling might be the easiest to implement in this case. Over-sampling time-series data is more difficult but has been done before [45].

This method does show great promise since it could work the same as the aggregation classifier without the need to manually create features. However, in its current state, it is not useable. Because of time restrictions further optimisation of this approach has been halted. However, with the right approach, it has been shown to work [19] [46].

5.3 Hyperparamter optimisation

GridSearchCV is a decent estimate of the best parameter, but it is only a guess. Though it uses 5-fold cross-validation, this value can still end up higher or lower due to sheer randomness. A second problem is that this method uses discrete values. The optimum value for n_estimators is 500, but the closest values are 300 and 700. It is possible that the ideal value could be somewhere between either 500 and 700 or 300 and 500. So, the value GridSearchCV gives is not the true optimal value but a combination of values that is closest.

A way to improve the hyperparameter optimisation process would be to make use of full factorial analysis, part of the discipline of Design of Experiments. There have been publications that show its effectiveness in tuning ML hyperparameters [47]. This has a few advantages, firstly is that it makes a model of the factors it uses. This model gives information on how strong the parameter influence is over its entire range. With GridSearchCV it is unknowable where the optimum is located, factorial analysis creates the opportunity to determine the optimum value. It also requires significantly less computation time than GridSearchCV because of this. GridSearchCV required 1200 models to be trained, the smallest factorial experiment would only need about 18 points though more would be preferred.
The second advantage is that this method would be able to pick out the individual contributions of a hyperparameter and their interaction contribution. This means that if a hyperparameter had no influence this method could detect it, this would once again save computation time.

Computation time is in this case not very important since 5-fold cross-validation training only costs about 20 seconds. However, the AU for example took around 10 minutes to train, so saving a few runs would mean saving a few hours in that case.

The question however remains whether it is worth the time and effort to set up a factorial experiment. GridSearchCV increased the average F1-score by 0.04, the question is how much could factorial analysis improve this.

5.4 General discussion

Now it is established that the aggregation classifier performs well, it's insightful to translate this back to the original goal. The goal is to reduce downtime of the plant, how much downtime could this model potentially prevent?

The time it took between the two most recent ball valve changes was 66 days. In those 66 days, production was running normally with little downtime, thus for this estimate, 66 days will be taken as the mean time until failure. Simply replacing the balls takes the mechanics about an hour. If one of the balls becomes too small and disappears into the pump it takes the mechanics about six hours. These repair times however are counted when the mechanics are on the premise, which is from 8 AM until 4 PM. Outside that 8-hour window other mechanics are on-call. But the chance that any of the on-call mechanics will be able to fix this specific issue is very small. In these cases, the mechanics have to wait until the mechanic with the right knowledge starts at 8 AM, from that moment it will still take them the full 6 hours.

If failure is evenly likely to occur regardless of the hour of day this would mean an average of about 5.5 hours of waiting time. In total this would then mean that every failure would, on average, take 11,5 hours. With failure occurring every 66 days this would mean 5.5 failures per year, resulting in a total of 64 hours of downtime per year. If everything were to be predicted correctly the downtime would only be 5.5 hours per year, assuming it could not be scheduled during other activities. Thus, the model would save an estimated 58.5 hours of downtime per year.

These saved downtime estimations might be a bit inflated by the fact that the 66 days mean time until failure is not actually until failure but until the previous replacement. The previous replacement was already predicted using data, thus it was too early. If assumed the degradation rate of the ball volume is constant, this would give about 91 days until failure instead of 66. If the degradation rate of the ball diameter is constant this would give 99 days between failures. To make a pessimistic estimation 99 days will be used to redo the calculations. With 99 days between failures, this PdM system would save 37 hours of downtime per year.

With plans to upgrade the ink plant to 4 mixing vessels each with the same AODD pump, 148 production hours would be saved. If expressed in batches this would mean about 59 additional batches of ink accounting for 118 000 litres could be produced in this time. The model would need to be tested before it can be used on new equipment though.

6 Conclusion

The goal for this research is as follows:
'Building a data-driven predictive maintenance model for the AODD pump in the Latex-based ink production plant at CPP Venlo that will reduce downtime of the plant.'

To achieve this goal three research questions have been created:

- What are the current failure modes of the AODD pump and what is their root cause?
- What models and approaches are most suitable for implementing a predictive maintenance system for the AODD pump?
- What model and approach performs best for predicting the failures of the AODD pump?

The AODD failure modes have been examined. From this, two structural problems came forth, premature diaphragm rupture and wear of the ball valves. The diaphragm rupture turned out to be caused by a positive inlet pressure of the AODD pump, which in turn produces high pressures in the pumping chambers shortening the diaphragm's lifespans. No models have been made of this failure mode because diaphragm monitoring tools are already on site to monitor its condition.
The wear of the ball valves seems to be unavoidable but easily predictable and has thus been chosen for the predictive maintenance approach.

To predict the ball valve wear, three machine learning modelling approaches were chosen from literature. These approaches were: an auto-encoder anomaly detector, a classification model based on raw time series, and a classification model based on statistical parameters extracted from those time series.
In the end the classification model based on statistical parameters from the time-series data ended up performing the best. The targets of the classification model are the 20 batches before each of the two historical failures. The XGBoost Classification model achieved an average F1-score of 0.79 ± 0.04 on 4-fold cross-validated 30 times after hyperparameter tuning on the validation set. On the test set the F1-score is 0.86.

An estimate has been made to see how much downtime this model would save for this pump based on repair times and frequency of failure. It is estimated that this model would prevent 37 hours of downtime every year, thus achieving the goal of this study.

7 Recommendations

This model will briefly discuss the recommended actions based on the research presented in this report.

7.1 Cleaning process:

It has been proven that the cleaning process is inadvertently damaging the AODD pump by supplying a positive suction pressure that is too high. The check ball valves create water hammer, which sends high-amplitude pressure waves through the pump and pipes. These water hammer effects are more severe because of the higher suction and thus reduce diaphragm lifespan. It is recommended to combat this problem by either preventing water hammer from occurring at positive suction pressures or by not supplying a positive suction pressure. This can be done in one of two ways:

- Replace check valves with 'non-slam' valves: If different valves are used that close slower and don't slam shut, the pressure wave from water hammer can be reduced or even mitigated. For this spring assisted valves can be used, these don't rely on gravity or flow reduction to start closing and thus don't slam shut.
- Change the cleaning process: Changing the cleaning process to apply the positive suction pressure would also solve the problem. This could be done by letting the pump create its own suction or by lowering the pressure to under 0.6 bar.

7.2 Model implementation

The next step for the predictive maintenance system would be to build an implementation for the model. The model currently works but for it to be useful it should make predictions on real-time factory data. A server should periodically get new factory data, clean the data, use the model to make predictions and then display the predictions somewhere the mechanics or maintenance engineers can access it. Within CPP there are similar servers on the premise. It would thus be recommended to make use of these servers. As long as the system is not up and running the 37 hours downtime reduction will not be achieved.

7.3 Further research

Though not successful, the other two modelling approaches both have their own advantages. Both approaches require less data pre-processing and process knowledge to construct making them easier to apply broadly. The AU could even make a remaining useful life prediction if build correctly. If more research were to be put into model building, then these approaches both show great promise. Both have been proven by previous research to be effective [23], [46].

The AU's biggest problem is its inability to account for the different ink types. To solve this issue two approaches could be taken. The first approach is to make a different AU for each ink type. This would make the results even more precise. The second option would be to find some way to encode it into the data or already pre-process it in. One way to do this is to only give the AU a 'corrected' flow rate based on the ink type.
The problem the time-series classifier is facing is mostly its inability to overcome the class imbalance. Weighted training has been applied but ended up not being enough to make the model learn

correctly. Other techniques to fix the class imbalance in time-series classification problems exist that have not yet been explored. Under-sampling the majority class or using more advanced techniques to over-sample the minority class. Other models that might perform better under class imbalance might also be explored.
Other models for this approach could also be explored in general. An ANN needs to be constructed very specifically to work, but some other algorithms can perform well out of the box. *Random Convolutional Kernel Transform* for example is one of these algorithms which has been shown to work faster and better than other commonly used algorithms [48]. The class imbalance would still need to be accounted for when training other models though.

References

[1] "ISO 13372:2012(en) Condition monitoring and diagnostics of machines — Vocabulary," ISO, 2012. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:13372:ed-2:v1:en. [Accessed 11 April 2023].

[2] M. Busch, "The Waddington Effect," *Sport Aviation,* pp. 98-101, March 2011.

[3] Sanimatic, "What is CIP Cleaning?," Sanimatic, 2023`. [Online]. Available: https://sanimatic.com/what-is-cip-cleaning/. [Accessed 27 2 2023].

[4] "Diaphram pump: Whats an diapgram pump?," Tacmina, [Online]. Available: https://www.tacmina.com/learn/basics/01.html. [Accessed 15 3 2023].

[5] Almatec, "Air-Operated Double Diapghagm pumps made of stainless steel Biocor series," Almatec, 11 2018. [Online]. Available: https://www.axflow.com/globalassets/catalog/tm/almatec-biocor-technical-manual-1309.pdf. [Accessed 16 3 2023].

[6] L. B. a. L. Kastanis, "Prognostics and Health Management of Industrial Assets: Current Progress and Road Ahead," *Frontiers,* vol. 3, 2020.

[7] Zscheuch, "Prognostic Model Development with Missing Labels," *Springer,* vol. 3, no. 61, pp. 327-343, 2019`.

[8] Zonta, "Predictive maintenance in the industry 4.0: a systematic literature review," *Computers & Industrial engineering,* no. 150, 2020.

[9] F. Logic, "Fuzzy Logic," Wikipedia, 30 january 2023. [Online]. Available: https://en.wikipedia.org/wiki/Fuzzy_logic. [Accessed 20 2 2023].

[10] B. e. all, "Forecasting fault events for predictive maintenace using data-driven techniques and ARMA modeling," *Computers & Industrial Engineering,* no. 115, pp. 41-53, 2018.

[11] J. Delua, "Supervised vs. Unsupervised Learning: What's the difference," IBM, 12 March 2021. [Online]. Available: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning. [Accessed 20 2 2023].

[12] Apple machine learning research, "Recognizing People in Photos Through Private On-Device Machine Learning," Apple, July 2021. [Online]. Available: https://machinelearning.apple.com/research/recognizing-people-photos. [Accessed 1 June 2023].

[13] S. Allwright, "RMSE vs MAE, whichs is the best regression metric," 31 July 2022. [Online]. Available: https://stephenallwright.com/rmse-vs-mae/. [Accessed 3 March 2023].

[14] S. Narkhede, "Understanding Confusion Matrix," Towards Data Science, 9 May 2018. [Online]. Available: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. [Accessed 3 March 2023].

[15] Developers Google, "Classification: Accuracy," Google, 18 july 2022. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/accuracy. [Accessed 3 March 2023].

[16] Google Developers, "Classification: Precision and Recall," Google, 18 July 2022. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall. [Accessed 3 March 2023].

[17] J. Korstanje, "The F1-score," Towards Data science, 31 August 2021. [Online]. Available: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6. [Accessed 3 march 2023].

[18] M. Bromberg, “Harmonic mean Defenitions: Formula and examples,” Investopedia, 8 November 2020. [Online]. Available: https://www.investopedia.com/terms/h/harmonicaverage.asp. [Accessed 3 March 2023].

[19] R. Aliyu, “Prognostic Health management of pumps using artificial intelligence in the oil and gas sector: a review,” *Applied Sciences,* vol. 12, 2022.

[20] R. Patton, “Observer-based fault detection and isolation: Robustness and applications,” *Pergamon,* vol. 5, no. 5, pp. 671-682, 1997.

[21] J. Brownlee, “Using Activation Functions in Neural Networks,” Machine Learning Mastery, 4 July 2020. [Online]. Available: https://machinelearningmastery.com/using-activation-functions-in-neural-networks/. [Accessed 17 May 2023].

[22] A. Derdat, “Applied Deep Learning - Part 3: Autoencoders,” Towards datascience, 3 October 2017. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798. [Accessed 3 March 2023].

[23] K. Fathi, “Predictive Maintenance: An Autoencoder Anomaly-Based Approach for a 3 DoF Delta Robot,” *MDPI Sensors,* no. 21, 2021.

[24] G. Hajgato, “PredMaX: Predictive maintenance with explainable deep conlolutional autoencoders,” *Advanced Engineering Informatics,* no. 54, 2022.

[25] “XGBoost,” Nividia, 2023. [Online]. Available: https://www.nvidia.com/en-us/glossary/data-science/xgboost/. [Accessed 3 May 2023].

[26] R. Holbrook, “Feature engineering,” Kaggle, [Online]. Available: https://www.kaggle.com/learn/feature-engineering. [Accessed 20 3 2023].

[27] PrjoectPro, “8 Feature Engineering Techniques for Machine Learning,” ProjectPro, 2 febuary 2023. [Online]. Available: https://www.projectpro.io/article/8-feature-engineering-techniques-for-machine-learning/423. [Accessed 20 3 2023].

[28] Microsoft, “Normalize Data component,” Microsoft, 11 04 2021. [Online]. Available: https://learn.microsoft.com/en-us/azure/machine-learning/component-reference/normalize-data. [Accessed 6 3 2023].

[29] IBM, “Time Aggregation,” IBM, 04 March 2021. [Online]. Available: https://www.ibm.com/docs/en/tnpm/1.4.5?topic=series-time-aggregation. [Accessed 23 March 2023].

[30] T. Yiu, “The Curse of Demensionality,” Towards Data Science, 20 july 2019. [Online]. Available: https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e. [Accessed 23 03 2023].

[31] T. Yui, “Understanding PCA (Principal Components Analysis),” Towards Data Science, 6 July 2019. [Online]. Available: https://towardsdatascience.com/understanding-pca-fae3e243731d. [Accessed 1 June 2023].

[32] K. W. B. L. H. W. K. N. V Chawlka, “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence Research,* vol. 16, pp. 321-357, 2002.

[33] Anna, “Why Weigth? The importance of Training on Balanced Datasets,” Towards Data Science, 28 01 2021. [Online]. Available: https://towardsdatascience.com/why-weight-the-importance-of-training-on-balanced-datasets-f1e54688e7df. [Accessed 8 5 2023].

[34] J. Brownlee, “Why One-Hot Encode Data in Machine Learning,” Machine Learning Mastery, 30 June 2020. [Online]. Available: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/. [Accessed 1 June 2023].

[35] J. Brownlee, “A Gentle introduction to k-fold cross-validation,” Machine Learning Mastery, 23 may 2018. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/. [Accessed 3 may 2023].

[36] “Overview on extracted features,” ReadTheDocs, 2023. [Online]. Available: https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html. [Accessed 8 5 2023].

[37] “sklearn.model_selection.GridSearchCV,” sklearn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. [Accessed 09 May 2023].

[38] A. Coelho, “Narrowing the Search: Which Hyperparameters Really Matter?,” data iku, 17 June 2020. [Online]. Available: https://blog.dataiku.com/narrowing-the-search-which-hyperparameters-really-matter. [Accessed 15 May 2023].

[39] xgboos developers, “CGBoost Parameters,” ReadTheDocs, 2022. [Online]. Available: https://xgboost.readthedocs.io/en/stable/parameter.html. [Accessed 22 May 2023].

[40] R. Bevans, “An Introduction to t Test | Definitions, Formula and Examples,” Scribbr, 31 January 2020. [Online]. Available: https://www.scribbr.com/statistics/t-test/#:~:text=What%20is%20a%20t%2Dtest,means%20is%20different%20from%20zero.. [Accessed 9 May 2023].

[41] A. B. e. al., “Hypothesis testing, type I and type II errors,” *Industrial Psychiatry Journal,* vol. 18, no. 2, pp. 127-131, 2009.

[42] “CHEBI:53251 - poly(tetrafluoroethylene),” ChEBI, 4 August 2014. [Online]. Available: https://www.ebi.ac.uk/chebi/searchId.do?chebiId=53251. [Accessed 27 3 2023].

[43] Rick, “Too High positive inlet pressure to the pump,” YTS pumps, [Online]. Available: https://yts-pumps.eu/too-high-positive-inlet-pressure-to-the-pump/. [Accessed 27 March 2023].

[44] Sentry Pulsation Dampners, “'J' model inlet stabilizer,” [Online]. Available: https://www.blacoh.com/downloads/literature/Inlet-Stabilizer.pdf. [Accessed 28 March 2023].

[45] T. Z. e. all, “Minority oversampling for imbalanced time series classification,” *Knowledge-Based Systems,* vol. 247, 2022.

[46] A. Guillame, “Time series classification for predictive maintenance on event logs,” November 2020. [Online]. Available: https://www.researchgate.net/publication/346143772_Time_series_classification_for_predictive_maintenance_on_event_logs. [Accessed 15 May 2023].

[47] G. A. Lujan-Moreno, “Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case-study,” *Expert Systems with Applications,* vol. 109, pp. 195-205, 2018.

[48] P. Jain, “Random Convolutional Kernel Transform,” [Online]. Available: https://piyush.dev/research-paper/time-series/classification/2020/07/24/Random-convolutional-kernel-transform.html. [Accessed 15 May 2023].

[49] B. Christiansen, “A Complete Guide To Prescriptive Maintenance,” Limble, 13 January 2022. [Online]. Available: https://limblecmms.com/blog/prescriptive-maintenance/. [Accessed 17 Maart 2023].

[50] D. Miklovic, “What comes after predictive maintenance?,” 17 March 2016. [Online]. Available: https://blog.lnsresearch.com/what-comes-after-predictive-maintenance. [Accessed 17 May 2023].

[51] M. C. e. al., “Overview on extracted features,” ReadTheDocs, 2023. [Online]. Available: https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html. [Accessed 22 May 2023].

[52] “eli5.xgboost,” ReadTheDocs, 2017. [Online]. Available: https://eli5.readthedocs.io/en/latest/autodocs/xgboost.html. [Accessed 22 Mei 2023].

[53] xgboost developers, “Python API Reference,” ReadTheDocs, 2022. [Online]. Available: https://xgboost.readthedocs.io/en/latest/python/python_api.html. [Accessed 22 May 2023].

Appendix A: Structure of the Artificial Neural Networks

Throughout the study a few types of Artificial Neural Networks (ANN) are used. ANN can be built up in many different ways, thus just calling them ANN is not very descriptive. This Appendix will show and explain how the ANN were build up and why. Both models were made using the Keras Python package from the sklearn library. The figures were created with the Keras '*model.summary()*' method.

A.1 Autoencoder ANN

The ANN constructed to function as a AU is shown in Figure 21. The model consists of 2 2D convolution layers and 2 2D de-convolution layers. Like a AU should, the layer size goes from 64 until 32 in the middle, it then goes back out again to 64 layers. The number 51 in the input shape stands for the amount of input channels, aka amount of different time series. This has been inflated to 51 thanks to the one-hot time-series encoding that has been applied. Two settings that are not represented in this figure but are important are the learning rate and the dropout rate. The dropout rate in this model is 0.1 or 10%, the learning rate is 0.0001. Every layer has '*ReLU*', meaning Rectified Linear Unit, as its activation function

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d (Conv2D)              (None, 51, 1, 64)         4160

 dropout (Dropout)            (None, 51, 1, 64)         0

 conv2d_1 (Conv2D)            (None, 51, 1, 32)         524320

 conv2d_transpose (Conv2DTra  (None, 51, 1, 32)         262176
 nspose)

 dropout_1 (Dropout)          (None, 51, 1, 32)         0

 conv2d_transpose_1 (Conv2DT  (None, 51, 1, 64)         131136
 ranspose)

 conv2d_transpose_2 (Conv2DT  (None, 51, 1, 1)          2305
 ranspose)

=================================================================
Total params: 924,097
Trainable params: 924,097
Non-trainable params: 0
_________________________________________________________________
```

Figure 21: The ANN that was used for the Autoencoder.

Figure 22 shows the settings that have used for the training of the AU. This is important to include because it shows things like the number of iterations (epochs), validation settings, early callback settings and the target. What is unique for the AU is that the target and the train data are the same. Note how everything is placed in a loop. This is because every batch is a different length. Normally the data would be fed in with a NumPy array, but NumPy doesn't allow differently sized rows. Thus, the data needs to be fed in batch by batch. This is sub-optimal for speed though, training takes somewhere between 15 and 20 minutes.

```
for row in Train_Batches:

    #Get 1 batch, make the data fit in the model
    X_batch=Train_Grouped.get_group(row)

    #fit the model with the fetched batch
    history = model.fit(
        X_batch,
        X_batch,
        epochs=50,
        verbose=0,
        validation_split=0.1,
        callbacks=[
            keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
        ],
    )
```

Figure 22: Shows the training process that is used by the AU.

A.2 Time-series Classification ANN

The ANN model used for the time-series classification is shown in Figure 23. This model consists of 1 input layer, then 3 1D-convolutional layers and in the end a pooling layer. Every convolutional layer has a normalisation layer right before it. The activation function that used is once again the ReLU function, except for the output layer which uses sigmoid. The learning rate is default at 0.01. This model also uses weighted training to try to fix the class imbalance. The class weights are: 0.51 for the majority class and 18.3 for the minority class.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 300, 25)]         0

conv1d (Conv1D)              (None, 300, 64)           4864

batch_normalization (BatchN  (None, 300, 64)           256
ormalization)

re_lu (ReLU)                 (None, 300, 64)           0

conv1d_1 (Conv1D)            (None, 300, 40)           7720

batch_normalization_1 (Batc  (None, 300, 40)           160
hNormalization)

re_lu_1 (ReLU)               (None, 300, 40)           0

conv1d_2 (Conv1D)            (None, 300, 20)           2420

batch_normalization_2 (Batc  (None, 300, 20)           80
hNormalization)

re_lu_2 (ReLU)               (None, 300, 20)           0

global_average_pooling1d (G  (None, 20)                0
lobalAveragePooling1D)

dense (Dense)                (None, 2)                 42

=================================================================
Total params: 15,542
Trainable params: 15,294
Non-trainable params: 248
_________________________________________________________________
```

Figure 23: The layers of the ANN model used for time-series classification.

Figure 24 shows the settings that have been used for the training process of the ANN time-series classifier. This training has 3 separate callback setting. *Model Checkpoint* being how the best model should be determined and saved, in this case based on sparse categorical cross entropy. The *REduceLROnPlateau* reduced the learning rate to 0.0001 when the performance while training is stagnating. A finer learning rate will be able to make smaller steps and thus get closer to the optimum value. It also includes settings for what metric to monitor etc. Important for this setting is that the patience is smaller than the patience of early stopping, otherwise this finer learning would never activate.
The loss function uses *sparse_categorical_crossentropy*. Cross-entropy is a concept from information theory and is a measure of how much bits of information would be needed to represent an event from one probability distribution instead of a different one. It is a loss function commonly used for classification tasks because classification commonly works by giving each label a chance. The target and the train data both can be represented by a probability distribution. Since we want these to align the cross-entropy should thus be minimised. The In this case sparse categorical means that it will primarily look at the minority class.

```
epochs = 500
batch_size = 32

callbacks = [
    keras.callbacks.ModelCheckpoint(
        "best_model.h5", save_best_only=True, monitor="sparse_categorical_crossentropy",
    ),
    keras.callbacks.ReduceLROnPlateau(
        monitor="sparse_categorical_crossentropy", factor=0.5, patience=20, min_lr=0.0001
    ),
    keras.callbacks.EarlyStopping(monitor="sparse_categorical_crossentropy", patience=50, verbose=1),
]

opt = keras.optimizers.Adam(learning_rate=0.001)
model.compile(
    optimizer=opt,
    loss="sparse_categorical_crossentropy",
    metrics=["sparse_categorical_accuracy"],

)
```

Figure 24: The training setting used for the ANN time-series classifier.

Appendix B: Extracted features and their model influence

This appendix will briefly discuss what feature importance exactly means. Then two feature importance techniques will be discussed and applied on the aggregation classifier (or statistical-time-series-data classifier). This appendix will also contain a table with all the features used in the ML model and their relative feature importance.

B.1: Theoretical background

Machine Learning (ML) is often regarded as a black box. Data goes in and an answer comes out. Whatever happens in between is sometimes considered unknowable but depending on the architecture that is not entirely true. For the statistical time-series classification model (or time-series aggregation model) the features and their relative importance can be determined. Two techniques (and their python package) have been used for this: SHAP and eli5.

Explain like I'm 5 (eli5) is a python package that is able to explain the predictions of ML classification models. It does this by using a concept that it calls permutation importance. The eli5 library grabs the model and its training data and determines its base performance. Then it grabs a random input feature column, shuffles the values randomly and checks the performance afterwards. If the model's performance was less, the feature must have been important. If the performance was about equal to before, the feature must not have been of any added value to the model. A feature gets a positive score if shuffling it reduced performance, a score of zero if shuffling didn't matter, and a negative score if the performance improved by shuffling that column.

The Shapley Additive exPlanations (SHAP) python package is also commonly used for ML explainability. This package is able to show the feature importance of any ML model, including things like image classification. The SHAP package also can show a user how a ML model came to each individual prediction. It shows which value a feature had and how important that was for that prediction specify. It works by borrowing a lot from game theory, the most important being the Shapley value.

So, eli5 can give a simple 'importance value' and SHAP give how an individual prediction is made based on its value's. This has three main uses: explainability, feature reduction and debugging.

Debugging in this case refers to problems in the code for the model. For example, if a ML-engineer forgot to remove the target variable from the training class the model would perform extremely well. Generally, not all columns are inspected by hand, especially when training data could contain thousands of columns. When consequently checking the feature importance a ML-engineer would quickly find what is causing its abnormally good performance.

Both techniques can also be used for feature reduction. The 'curse of dimensionality' is a ML concept that more features can worsen model performance since every value can turn into a statistical outlier [30]. Both the SHAP and eli5 package can help pinpoint which features add little or no value to the model. These features can then be removed for better model performance. Do note that in this rapport this approach has not been used, instead feature reduction was applied using the tsfresh package.

The third use for eli5 and SHAP is explainability. Banks for instance are legally required to give a reason for loan rejection in certain countries or states. If a ML model were to be used for this, SHAP would make it possible to extract the reasons. In healthcare doctors are sometimes sceptical of ML methods for diagnostics, by showing the doctors what the model uses to make predictions this can build trust. Sometimes features are mere symptoms of a problem, but they can also be the cause. If SHAP shows that smokers get the biggest positive predictions for lung cancer, you could target this directly. This way ML and SHAP can be used to stop a problem from occurring in the first place, this is in a way already the first step of Prescriptive Maintenance (PxM). Prescriptive Maintenance is the next step up from PdM, it doesn't just predict failures but also learns way to avoid the failures it predicts [49]. An example from Dan Miklovic:

"Let's say a piece of equipment is showing increasing bearing temperature. Predictive analytics looks at the temperature profile and tells you it is likely to fail in X amount of time. On the other hand, prescriptive analytics tells you that if you slow the equipment down by Y%, the time to failure can be doubled, putting you within the already scheduled maintenance window and revealing whether you can still meet planned production requirements." [50].

So in this example the high Shapley value for the bearing temperature would be the symptom which eventually would lead you to the root cause, rotation speed.

B.2: SHAP predictions

Section C.3 will display all features and their absolute importance using eli5. This section has been moved to the end because of readability issues caused by the 9-page long table. The eli5 table shows feature importance but not how the feature impacts a prediction. Does a high value of feature x increase or decrease the prediction value? Eli5 cannot make this distinction, but SHAP can. SHAP can even make this distinction on individual predictions, like shown in Figure 25. It shows that every prediction starts from a base value 0.46, and every feature adds or subtracts a little resulting in the end prediction. In this case the end prediction was rounded up to a 1, meaning a positive prediction.

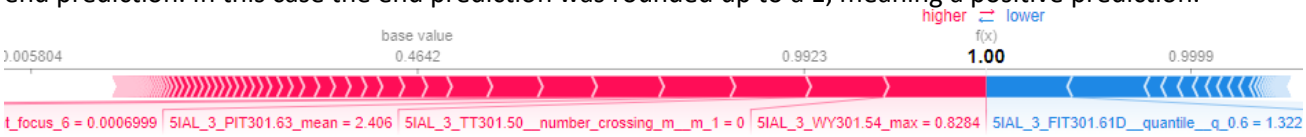


Figure 25: SHAP showing the additive and subtractive values caused by features of the XGBoost model.

Figure 25 of course still doesn't show the full picture of how each value affects a prediction, for this Figure 26 is more informative. Figure 26 displays the top 20 most important features, and what their respective values were at any prediction. The colour of the dot represents the relative value, high or low, of the feature. The position on the x-axis shows if that dot had a positive or negative impact on the prediction. In case of the top feature: *5IAL_3_TT301.50_number_crossing_m_m_1*, medium to high values tended to have a negative effect, whereas low values had a positive effect on failure predictions.

Note that the feature created called '*frac*' which is described in 4.2.1 is shown to be in the top 20 twice, which means that this is most likely a good feature. It also shows that the ML model 'looks' at the same things a human would, which is good for understanding and trust in the model.

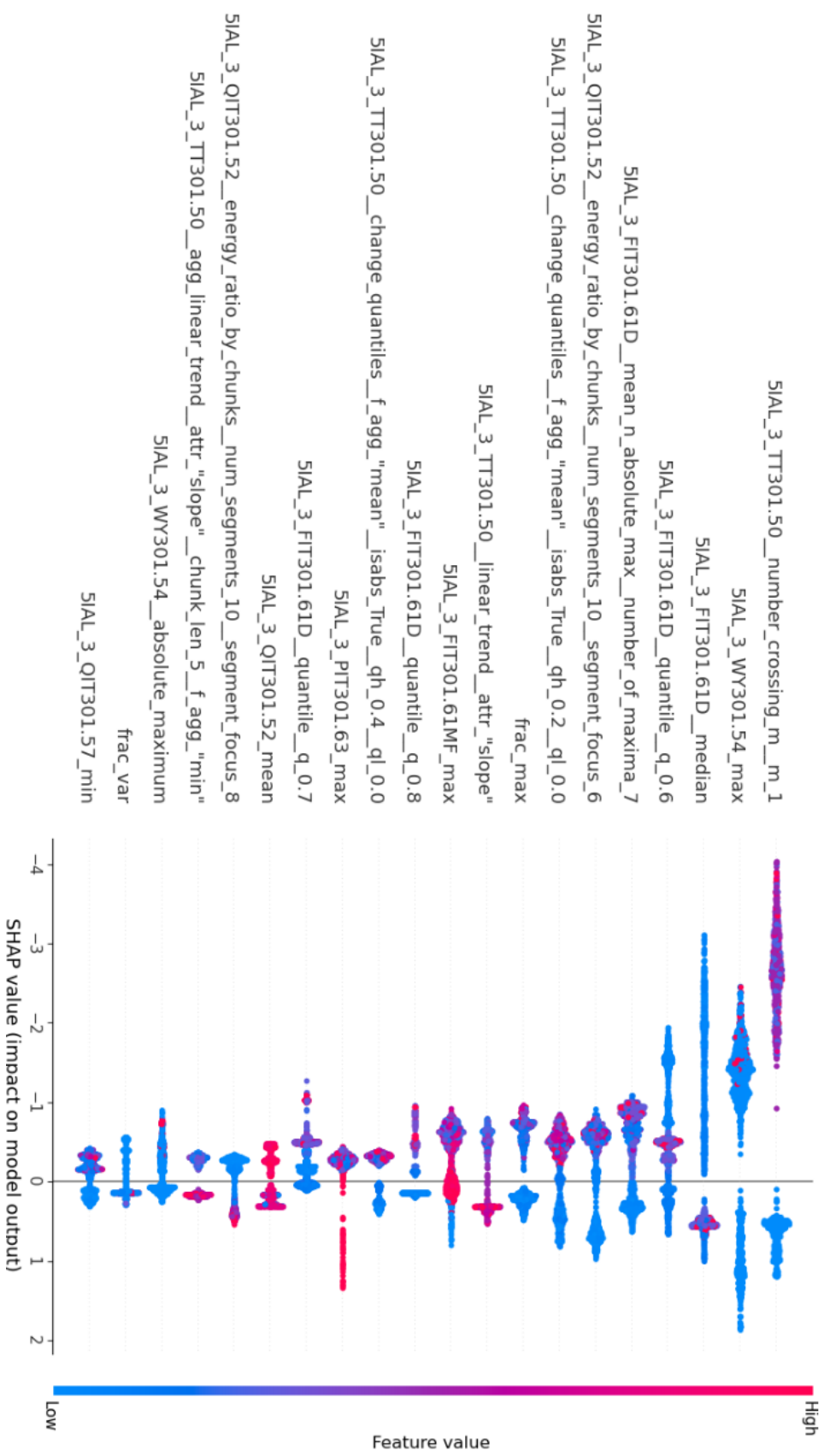


Figure 26: Most Important features visualised using the SHAP Package.

B.3 Feature importance:

In this section an overview of all features used for the XGBoost and their relative importance in the model are shown. These features are automatically generated and named based on both their name internally at CPP and the tsfresh package. The first part, structured as: *5IAL_3_XX301.XX*, is from CPP internal documentation, everything after that is automatically generated by tsfresh. To find out what these featuresnames entail, refer to tsfesh's documentation [51].

For how the feature importance is calculated for an XGBoost model, check out the documentation [52] [53]. Note that the 'weigth' column is constructed such that the sum of these weights is 1. Also be aware that the weigth is absolute, so a high feature value resulting in a negative prediction will have the same weight as a high feature value resulting in a positive prediction.

Table 5: Table including all the features used for the XGBoost model and their respective weights according to XGBoost.

Weight	Feature
0.1864	5IAL_3_TT301.50__number_crossing_m__m_1
0.1333	5IAL_3_TT301.50__ar_coefficient__coeff_0__k_10
0.059	5IAL_3_WY301.54__quantile__q_0.8
0.0493	5IAL_3_FIT301.61D__median
0.0479	5IAL_3_FIT301.61D__quantile__q_0.4
0.0391	5IAL_3_P301.70_min
0.0366	5IAL_3_WY301.54__maximum
0.0348	5IAL_3_WY301.54__absolute_maximum
0.0341	5IAL_3_PIT301.63__quantile__q_0.7
0.0225	5IAL_3_LIT301.54_sum
0.0219	5IAL_3_PIT301.63__maximum
0.0196	5IAL_3_WY301.54_max
0.0139	5IAL_3_TT301.50__agg_linear_trend__attr_"rvalue"__chunk_len_10__f_agg_"mean"
0.0133	5IAL_3_TT301.50_sum
0.0115	5IAL_3_FIT301.61VF_max
0.0113	5IAL_3_FIT301.61D__quantile__q_0.7
0.0112	5IAL_3_TT301.50__agg_linear_trend__attr_"rvalue"__chunk_len_5__f_agg_"mean"
0.0111	5IAL_3_PIT301.63__mean
0.0104	frac__cwt_coefficients__coeff_7__w_5__widths_(2, 5, 10, 20)
0.0102	5IAL_3_TT301.50_mean
0.01	5IAL_3_PIT301.63_mean
0.0094	frac_min
0.0091	5IAL_3_PIT 301.55_mean
0.0083	5IAL_3_TT301.50__linear_trend__attr_"rvalue"
0.0081	5IAL_3_FIT301.61MF_max
0.008	5IAL_3_QIT301.57_mean
0.007	5IAL_3_FIT301.61D_max
0.0066	5IAL_3_XPV301.53_mean
0.0065	5IAL_3_FIT301.61MF__agg_linear_trend__attr_"rvalue"__chunk_len_50__f_agg_"max"
0.0064	5IAL_3_PIT301.63__abs_energy

0.0062	5IAL_3_QIT301.57_max
0.0057	5IAL_3_PIT301.60_max
0.0056	5IAL_3_FIT301.61D__mean_n_absolute_max__number_of_maxima_7
0.0054	5IAL_3_QIT301.52_var
0.0049	5IAL_3_XPV301.54_mean
0.0047	5IAL_3_WY301.54_std
0.0045	5IAL_3_XPV301.54_max
0.0045	5IAL_3_PIT301.63_max
0.0043	5IAL_3_XPV301.53_max
0.0042	5IAL_3_QIT301.52__index_mass_quantile__q_0.2
0.0039	5IAL_3_FIT301.61MF_mean
0.0036	5IAL_3_PIT301.63_min
0.0034	5IAL_3_XPV301.13_sum
0.0033	5IAL_3_TT301.50__energy_ratio_by_chunks__num_segments_10__segment_focus_4
0.0032	5IAL_3_FIT301.61D__quantile__q_0.9
0.0032	5IAL_3_FIT301.61D__quantile__q_0.8
0.0031	5IAL_3_FIT301.61MF_sum
0.003	5IAL_3_PIT 301.55_std
0.0028	5IAL_3_PIT301.63__cwt_coefficients__coeff_8__w_2__widths_(2, 5, 10, 20)
0.0027	5IAL_3_TT301.50_max
0.0026	5IAL_3_LSL301.53_mean
0.0025	5IAL_3_FIT301.61D_sum
0.0024	5IAL_3_TT301.50__change_quantiles__f_agg_"mean"__isabs_True__qh_0.4__ql_0.0
0.0024	5IAL_3_QIT301.52__energy_ratio_by_chunks__num_segments_10__segment_focus_8
0.0023	5IAL_3_FIT301.61VF_min
0.0022	5IAL_3_QIT301.52_max
0.0022	5IAL_3_P301.72_max
0.002	5IAL_3_PIT301.63_sum
0.002	5IAL_3_TT301.50_min
0.0018	5IAL_3_PIT 301.55_max
0.0018	5IAL_3_LSL301.64_sum
0.0017	5IAL_3_PIT301.63__quantile__q_0.8
0.0017	5IAL_3_TT301.50__energy_ratio_by_chunks__num_segments_10__segment_focus_3
0.0017	5IAL_3_FIT301.61MF_min
0.0016	5IAL_3_WY301.54_min
0.0016	5IAL_3_XPV301.63_max
0.0015	5IAL_3_XPV301.54_std
0.0015	5IAL_3_WY301.54_mean
0.0014	5IAL_3_R301.71__number_cwt_peaks__n_5
0.0014	5IAL_3_PIT301.60_mean
0.0012	5IAL_3_TT301.50__variation_coefficient
0.0011	5IAL_3_PIT301.60_std
0.0011	5IAL_3_TT301.50__fft_coefficient__attr_"angle"__coeff_0
0.0011	5IAL_3_XPV301.13_min
0.0011	5IAL_3_FIT301.61MF_std

0.0009	frac_sum
0.0009	5IAL_3_PIT301.63__c3__lag_2
0.0008	5IAL_3_PIT301.63__quantile__q_0.9
0.0008	5IAL_3_PIT301.63__c3__lag_1
0.0008	5IAL_3_XPV301.53_std
0.0008	5IAL_3_LIT301.54_max
0.0008	5IAL_3_QIT301.52_mean
0.0007	5IAL_3_QIT301.52__c3__lag_1
0.0007	5IAL_3_TT301.50__energy_ratio_by_chunks__num_segments_10__segment_focus_0
0.0007	5IAL_3_FIT301.61D_min
0.0007	frac_var
0.0006	5IAL_3_XPV301.54_sum
0.0006	5IAL_3_PIT 301.55_sum
0.0005	5IAL_3_PIT301.63__agg_linear_trend__attr_"intercept"__chunk_len_50__f_agg_"mean"
0.0005	5IAL_3_TT301.50__change_quantiles__f_agg_"mean"__isabs_True__qh_0.2__ql_0.0
0.0005	5IAL_3_TT301.50__fft_coefficient__attr_"real"__coeff_1
0.0005	5IAL_3_LIT301.54_mean
0.0004	5IAL_3_TT301.50__linear_trend__attr_"slope"
0.0004	5IAL_3_FIT301.61D__quantile__q_0.6
0.0004	5IAL_3_P301.72_sum
0.0004	frac_max
0.0004	5IAL_3_QIT301.52__energy_ratio_by_chunks__num_segments_10__segment_focus_6
0.0004	5IAL_3_QIT301.57_min
0.0003	5IAL_3_QIT301.57_sum
0.0003	5IAL_3_QIT301.52_min
0.0003	5IAL_3_GSC301.44_sum
0.0003	5IAL_3_QIT301.52__energy_ratio_by_chunks__num_segments_10__segment_focus_7
0.0002	5IAL_3_TT301.50__index_mass_quantile__q_0.1
0.0002	5IAL_3_FIT301.61D_mean
0.0002	5IAL_3_QIT301.52_std
0.0002	5IAL_3_TT301.50__index_mass_quantile__q_0.2
0.0001	5IAL_3_PIT301.63__cwt_coefficients__coeff_12__w_5__widths_(2, 5, 10, 20)
0.0001	5IAL_3_TT301.50__agg_linear_trend__attr_"slope"__chunk_len_5__f_agg_"min"
0.0001	frac_mean
0.0001	5IAL_3_GSC301.44_mean
0.0001	5IAL_3_XPV301.13_max
0.0001	5IAL_3_XPV301.63_sum
0.0001	5IAL_3_XPV301.63_std
0	5IAL_3_P301.70_max
0	5IAL_3_R301.71_max
0	5IAL_3_PIT301.63__agg_linear_trend__attr_"intercept"__chunk_len_5__f_agg_"min"
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_9__w_10__widths_(2, 5, 10, 20)
0	5IAL_3_XPV301.05_max
0	5IAL_3_XPV301.06_max
0	5IAL_3_XPV301.09_max

0	5IAL_3_XPV301.22_max
0	5IAL_3_XPV301.35_max
0	5IAL_3_XPV301.08_max
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_13__w_5__widths_(2, 5, 10, 20)
0	5IAL_3_PIT301.63__median
0	5IAL_3_TT301.50__agg_linear_trend__attr_"rvalue"__chunk_len_5__f_agg_"min"
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_11__w_5__widths_(2, 5, 10, 20)
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_8__w_10__widths_(2, 5, 10, 20)
0	5IAL_3_XPV301.42_max
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_2__w_5__widths_(2, 5, 10, 20)
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_7__w_10__widths_(2, 5, 10, 20)
0	5IAL_3_PIT301.63__quantile__q_0.6
0	Batch_duration_x
0	frac_std
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_14__w_5__widths_(2, 5, 10, 20)
0	5IAL_3_XPV301.13_std
0	5IAL_3_LSH301.56_std
0	5IAL_3_XPV301.36_max
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_10__w_5__widths_(2, 5, 10, 20)
0	5IAL_3_XPV301.43_max
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_10__w_10__widths_(2, 5, 10, 20)
0	5IAL_3_XPV301.09_min
0	5IAL_3_XPV301.08_min
0	5IAL_3_XPV301.06_min
0	5IAL_3_XPV301.05_min
0	5IAL_3_P301.72_min
0	5IAL_3_R301.71_min
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_11__w_10__widths_(2, 5, 10, 20)
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_3__w_2__widths_(2, 5, 10, 20)
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_7__w_5__widths_(2, 5, 10, 20)
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_1__w_2__widths_(2, 5, 10, 20)
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_14__w_10__widths_(2, 5, 10, 20)
0	5IAL_3_PIT301.60_min
0	5IAL_3_PIT 301.55_min
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_9__w_5__widths_(2, 5, 10, 20)
0	5IAL_3_XPV301.46_max
0	5IAL_3_LSH301.56_max
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_13__w_10__widths_(2, 5, 10, 20)
0	5IAL_3_301.OCCUPIED_max
0	5IAL_3_LSLL301.69_max
0	5IAL_3_LSL301.68_max
0	5IAL_3_LSL301.64_max
0	5IAL_3_GSO301.44_max
0	5IAL_3_GSC301.44_max
0	5IAL_3_LSL301.53_max

0	5IAL_3_LSL301.51_max
0	5IAL_3_301.OCCUPIED_std
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_3__w_5__widths_(2, 5, 10, 20)
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_12__w_10__widths_(2, 5, 10, 20)
0	5IAL_3_LIT301.54_std
0	5IAL_3_LSL301.53_std
0	5IAL_3_LSLL301.69_std
0	5IAL_3_XPV301.35_mean
0	5IAL_3_301.OCCUPIED_mean
0	5IAL_3_LSLL301.69_mean
0	5IAL_3_LSL301.68_mean
0	5IAL_3_LSL301.64_mean
0	5IAL_3_GSO301.44_mean
0	5IAL_3_LSL301.51_mean
0	5IAL_3_XPV301.63_mean
0	5IAL_3_P301.72__agg_linear_trend__attr_"rvalue"__chunk_len_10__f_agg_"min"
0	5IAL_3_P301.72__agg_linear_trend__attr_"slope"__chunk_len_10__f_agg_"min"
0	5IAL_3_XPV301.46_mean
0	5IAL_3_XPV301.43_mean
0	5IAL_3_XPV301.42_mean
0	5IAL_3_XPV301.36_mean
0	5IAL_3_XPV301.22_mean
0	5IAL_3_LSL301.68_std
0	5IAL_3_XPV301.09_mean
0	5IAL_3_XPV301.08_mean
0	5IAL_3_XPV301.06_mean
0	5IAL_3_XPV301.05_mean
0	5IAL_3_P301.72_mean
0	5IAL_3_R301.71_mean
0	5IAL_3_P301.70_mean
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_14__w_20__widths_(2, 5, 10, 20)
0	5IAL_3_FIT301.61VF_mean
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_3__w_10__widths_(2, 5, 10, 20)
0	5IAL_3_FIT301.61D__matrix_profile__feature_"75"__threshold_0.98
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_13__w_20__widths_(2, 5, 10, 20)
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_12__w_20__widths_(2, 5, 10, 20)
0	5IAL_3_FIT301.61D__matrix_profile__feature_"max"__threshold_0.98
0	5IAL_3_LSH301.56_mean
0	5IAL_3_XPV301.13_mean
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_4__w_10__widths_(2, 5, 10, 20)
0	5IAL_3_LSL301.64_std
0	5IAL_3_GSO301.44_std
0	5IAL_3_GSC301.44_std
0	5IAL_3_XPV301.35_min
0	5IAL_3_LSL301.51_std

0	5IAL_3_PIT301.63__cwt_coefficients__coeff_1__w_5__widths_(2, 5, 10, 20)
0	5IAL_3_XPV301.46_std
0	5IAL_3_XPV301.43_std
0	5IAL_3_XPV301.42_std
0	5IAL_3_XPV301.36_std
0	5IAL_3_XPV301.35_std
0	5IAL_3_XPV301.22_std
0	5IAL_3_XPV301.09_std
0	5IAL_3_XPV301.08_std
0	5IAL_3_XPV301.06_std
0	5IAL_3_XPV301.05_std
0	5IAL_3_P301.72_std
0	5IAL_3_R301.71_std
0	5IAL_3_P301.70_std
0	5IAL_3_PIT301.63_std
0	5IAL_3_FIT301.61D_std
0	5IAL_3_FIT301.61VF_std
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_6__w_10__widths_(2, 5, 10, 20)
0	5IAL_3_TT301.50__agg_linear_trend__attr_"rvalue"__chunk_len_5__f_agg_"max"
0	5IAL_3_QIT301.57_std
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_5__w_10__widths_(2, 5, 10, 20)
0	5IAL_3_TT301.50_std
0	5IAL_3_XPV301.22_min
0	5IAL_3_XPV301.46_min
0	5IAL_3_XPV301.36_min
0	5IAL_3_LSL301.68_sum
0	CP55
0	CGD1
0	CGB2
0	CD1
0	CC1
0	CB2
0	Batch_duration_y
0	5IAL_3_PIT301.63__quantile__q_0.3
0	5IAL_3_WY301.54_sum
0	5IAL_3_PIT301.63__agg_linear_trend__attr_"intercept"__chunk_len_50__f_agg_"max"
0	5IAL_3_LSH301.56_sum
0	5IAL_3_PIT301.63__quantile__q_0.2
0	5IAL_3_301.OCCUPIED_sum
0	5IAL_3_LSLL301.69_sum
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_7__w_2__widths_(2, 5, 10, 20)
0	5IAL_3_R301.71_sum
0	5IAL_3_GSO301.44_sum
0	5IAL_3_LSL301.53_sum
0	5IAL_3_LSL301.51_sum

0	5IAL_3_PIT301.63__c3__lag_3
0	5IAL_3_XPV301.53_sum
0	5IAL_3_XPV301.46_sum
0	5IAL_3_XPV301.43_sum
0	5IAL_3_XPV301.42_sum
0	5IAL_3_XPV301.36_sum
0	5IAL_3_XPV301.35_sum
0	5IAL_3_XPV301.22_sum
0	5IAL_3_XPV301.09_sum
0	5IAL_3_XPV301.08_sum
0	5IAL_3_XPV301.06_sum
0	CP58
0	CP70
0	IU70
0	KB2
0	5IAL_3_PIT301.63__agg_linear_trend__attr_"intercept"__chunk_len_10__f_agg_"min"
0	5IAL_3_PIT301.63__agg_linear_trend__attr_"intercept"__chunk_len_5__f_agg_"mean"
0	5IAL_3_PIT301.63__agg_linear_trend__attr_"intercept"__chunk_len_5__f_agg_"max"
0	5IAL_3_PIT301.63__agg_linear_trend__attr_"intercept"__chunk_len_10__f_agg_"mean"
0	5IAL_3_PIT301.63__linear_trend__attr_"intercept"
0	5IAL_3_PIT301.63__agg_linear_trend__attr_"intercept"__chunk_len_50__f_agg_"min"
0	5IAL_3_XPV301.13__minimum
0	5IAL_3_PIT301.63__agg_linear_trend__attr_"intercept"__chunk_len_10__f_agg_"max"
0	5IAL_3_WY301.54__quantile__q_0.9
0	5IAL_3_WY301.54__mean_n_absolute_max__number_of_maxima_7
0	5IAL_3_PIT301.63__mean_n_absolute_max__number_of_maxima_7
0	5IAL_3_PIT301.63__absolute_maximum
0	YP70
0	YP58
0	YP55
0	YD1
0	YC1
0	YB2
0	MP70
0	MP58
0	MP55
0	MD1
0	MC1
0	MB2
0	KP70
0	KP58
0	KP55
0	KD1
0	KC1
0	5IAL_3_XPV301.05_sum

0	5IAL_3_P301.70_sum
0	5IAL_3_XPV301.42_min
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_8__w_5__widths_(2, 5, 10, 20)
0	5IAL_3_P301.72_var
0	5IAL_3_R301.71_var
0	5IAL_3_P301.70_var
0	5IAL_3_PIT301.63_var
0	5IAL_3_FIT301.61D_var
0	5IAL_3_FIT301.61VF_var
0	5IAL_3_FIT301.61MF_var
0	5IAL_3_PIT301.60_var
0	5IAL_3_QIT301.57_var
0	5IAL_3_PIT 301.55_var
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_6__w_5__widths_(2, 5, 10, 20)
0	5IAL_3_TT301.50_var
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_4__w_2__widths_(2, 5, 10, 20)
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_4__w_5__widths_(2, 5, 10, 20)
0	5IAL_3_LSH301.56_min
0	5IAL_3_PIT301.63__quantile__q_0.1
0	5IAL_3_LIT301.54_min
0	5IAL_3_301.OCCUPIED_min
0	5IAL_3_LSLL301.69_min
0	5IAL_3_LSL301.68_min
0	5IAL_3_LSL301.64_min
0	5IAL_3_GSO301.44_min
0	5IAL_3_GSC301.44_min
0	5IAL_3_LSL301.53_min
0	5IAL_3_LSL301.51_min
0	5IAL_3_XPV301.63_min
0	5IAL_3_XPV301.54_min
0	5IAL_3_XPV301.53_min
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_6__w_2__widths_(2, 5, 10, 20)
0	5IAL_3_XPV301.43_min
0	5IAL_3_XPV301.05_var
0	5IAL_3_XPV301.06_var
0	5IAL_3_XPV301.08_var
0	5IAL_3_XPV301.09_var
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_5__w_2__widths_(2, 5, 10, 20)
0	5IAL_3_FIT301.61VF_sum
0	5IAL_3_PIT301.63__quantile__q_0.4
0	5IAL_3_PIT301.60_sum
0	5IAL_3_PIT301.63__minimum
0	5IAL_3_QIT301.52_sum
0	5IAL_3_PIT301.63__root_mean_square
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_2__w_2__widths_(2, 5, 10, 20)

0	5IAL_3_WY301.54_var
0	5IAL_3_XPV301.13_var
0	5IAL_3_LSH301.56_var
0	5IAL_3_LIT301.54_var
0	5IAL_3_301.OCCUPIED_var
0	5IAL_3_LSLL301.69_var
0	5IAL_3_PIT301.63__cwt_coefficients__coeff_5__w_5__widths_(2, 5, 10, 20)
0	5IAL_3_LSL301.64_var
0	5IAL_3_GSO301.44_var
0	5IAL_3_GSC301.44_var
0	5IAL_3_LSL301.53_var
0	5IAL_3_LSL301.51_var
0	5IAL_3_XPV301.63_var
0	5IAL_3_XPV301.54_var
0	5IAL_3_XPV301.53_var
0	5IAL_3_XPV301.46_var
0	5IAL_3_XPV301.43_var
0	5IAL_3_XPV301.42_var
0	5IAL_3_XPV301.36_var
0	5IAL_3_XPV301.35_var
0	5IAL_3_XPV301.22_var
0	5IAL_3_LSL301.68_var

Appendix C: Raw Time-series Autoencoder

This appendix contains the entire final version of the Raw time-series autoencoder like it is called throughout the code. This code was written in Juypter notebook, advantage of this is that it also saves the output. This output is thus included. Part of the training process of the Autoencoder was removed because it took in too much space and was of very little value. The code is documented and commented throughout so It should be independently readable.

Appendix C.pdf

Appendix D: Aggregation classifier parameters notebook

This appendix contains the entire final version of the aggregation classifier as its calld throught the code. This code was written in Juypter notebook, advantage of this is that it also saves the output. This output is thus included.

Appendix D.pdf

Appendix E: Raw time-series classifier notebook

This appendix contains the entire final version of the aggregation classifier as its called throught the code. This code was written in Juypter notebook, advantage of this is that it also saves the output. This output is thus included.

Appendix E.pdf