Condition Monitoring of Hydroelectric Power Plants

Asgeir Øen Åsnes

Master of Science in Cybernetics and Robotics

Submission date:	May 2018
Supervisor:	Lars Imsland, ITK
Co-supervisor:	Anders Willersrud, Hymatek Controls Beatriz Galindo-Prieto, ITK Frank Ove Westad, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Preface

This project thesis is written as a part of my Master's degree in Cybernetics and Robotics at NTNU Trondheim. The work is done during the spring of 2018. This thesis has four listed supervisors. Prof. Lars Imsland has been my main supervisor, we have met almost every other week during the thesis. Lars has been available for discussions about the case and methods used. Dr. Anders Willersrud has been my co-supervisor from Hymatek Controls. Anders has been available for discussion during the thesis as well. Mostly through Skype, but I was at one occasion in Oslo at Hymatek to discuss the different possible cases I had found in the dataset. Dr. B. Galindo-Prieto and Adj. Prof. Frank Westad were brought on-board as co-supervisors to help with missing data, regression analysis, and dimensionality reduction in one of the datasets. The data were analyzed by PLS regression and several variable selection methods. In addition we discussed possible techniques for advanced feature selection from a larger dataset. The results were not conclusive for the studied data set, and as it was not possible to include the variables in the anomaly detection due to the sampling of the data, it is not included. However, we have reasons to think they could work well in other data sets and we aim to include this as future work.

During this thesis I have been working on a dataset from 27 hydroelectric power plants provided by Hymatek. In addition they provided a log of recorded incidents and maintenance at the plants. I have been working on a Dell desktop provided by NTNU in Ubuntu, using the Anaconda distribution to manage my Python environment. Keras and Scikit-learn have been used as machine learning libraries. In my project thesis I looked into the possibility to classify the condition of the guide vanes of a Francis turbine. The code used for one class support vector machines is reused from my project thesis. The rest of the framework used in this thesis, has been written by me during the master thesis. This includes a library for reading and converting the dataset provided by Hymatek into a form that could be analyzed.

In this thesis three different anomaly detection techniques are tested on Pelton turbines. One class support vector machine was used in my project thesis. The two others are proposed by myself after searching the literature. I found the Pelton case when looking through the data and plant logs provided by Hymatek. It was chosen as a good use-case in cooperation with Hymatek and Lars. No literature has been provided, and all papers and articles used in this thesis are a result of my own literary research.

I want to thank all my supervisors for their help during my thesis, and Hymatek for providing such an extensive data set.

Trondheim, May 31, 2018,
Asgeir Øen Åsnes

Abstract

Much research can be found on condition monitoring for many industries. There are, however, very little research on the use of condition monitoring in hydroelectric power plants. A literature search investigates this field, and it is found that support vector machines and neural networks have been successfully used for condition monitoring of known failure modes. Most of the existing research involves methods that require both normal and abnormal data, and need abnormal data for all failure modes that one want to monitor. As most hydroelectric power plants are operating normally for the majority of their lifetime, it is not trivial to sample data from all known failure modes.

Anomaly detection is presented as it enables the detection of anomalies in the process data with only normal process data available. Three different methods for anomaly detection are presented, support vector machines, kernel density estimation and long short term neural networks. In addition, different techniques for feature and dimensionality reduction are presented. A data set containing data from 27 power plants is analyzed, and a reported incident with operational problems for the needles of a Pelton turbine is extracted for analysis. Data from two different plants and three different turbines are included in the analysis to investigate cross plant performance. An artificial error replicating the operational problems for the reported incident is created to verify how early the anomaly detection techniques can detect system degradation. In addition, two Pelton turbine start failures are included in the analysis to verify that the methods correctly detect abnormal system data.

All three methods are shown able to detect the operational problems for the Pelton needles. However, the one class support vector machine is shown to be very dependent on the choice of hyperparameters with regards to the training data. Kernel density estimation and long short term recurrent neural network show better performance and are more robust with regards to parameterization and training data. It is also shown that they can detect the early signs of the system degradation that can be seen leading up to the reported incident with the Pelton needles.

The work from this thesis and the work from Åsnes (2017) will be a major part of a paper written for HYDRO 2018 Progress through Partnerships. It will be presented in Gdansk, Poland 15 - 17 October 2018.

Samandrag

Mykje forskning finnast på tilstandsovervaking for mange industriar. Det er imidlertid svært lite forskning på bruk av tilstandsovervåking i vannkraftverk. Ei litteraturstudie undersøkjer dette feltet, og det blir funne at support vector machines og nevrale nettverk har vorte brukt vellukka for tilstandsovervaking av kjente feilmodi. Det meste av den eksisterande forskninga innebær metodar som krev både normale og unormale data, og treng difor data for alle feilmodi ein vil overvake. Då dei fleste vannkraftverk opererer utan feil i brorparten av deira levetid, er det ikkje trivielt å finne data som representerar alle kjende feilmodi.

Anomaliedeteksjon muliggjer deteksjon av uregelmessigheiter i prosessdataen, med kun normal prosessdata tilgjengeleg under trening. Tre ulike metodar for anomalitetsdeteksjon blir presentert, support vector machines, kernel density estimation og long short term recurrent neural netverk. I tillegg presenterast ulike teknikkar for variabel og dimensjonsreduksjon. Eit datasett som inneheld data frå 27 kraftverk blir analysert, og ein rapportert hendelse med operasjonsproblemer for nålene til en Pelton-turbin blir plukka ut for analyse. Data frå to forskjellige kraftverk og tre forskjellige turbinar inngår i analysa for å samanlikne metodane på kryss av kraftverk. Ein kunstig feil som replikerar operasjonsproblemet for den rapporterte hendelsen blir laga for å bekrefte kor tidleg anomalitetsdetekteringsteknikkane kan oppdage avvik i systemet. I tillegg inngår to startfeil frå Pelton turbinar i analysen, for å verifisere at metodane klarar å fange fleire kjende eksempel på unormal data.

Alle tre metodane viser seg i stand til å oppdage operasjonsproblemer for Pelton nålene. Support vector machine viser seg å være svært avhengig av valet av hyperparametre med hensyn til treningsdataene. Kernel density estimation og long short term recurrent neural networks viser betre ytelse og er meir robuste med hensyn til hyperparameterisering og treningsdata. Det blir også vist at dei kan oppdage dei tidlige teikna på systemavvik sett i forkant av den rapporterte hendelsen med Pelton nålene.

Arbeidet frå denne masteroppgåva og frå Åsnes (2017) vil være ein stor del av ein artikkel til HYDRO 2018, Progress through Partnerships som skal presenterast i Gdansk i Polen 15 - 17 oktober 2018.

Contents

Abbreviations

MI	=	Mutual Information
SVM	=	Support Vector Machine
OC SVM	=	One class Support Vector Machine
NN	=	Neural Network
RNN	=	Recurrent Neural Network
LSTM RNN	=	Long Short Term Recurrent Neural Network
PDF	=	Probability Density Function
RBF	=	Radial Basis Function
PCA	=	Principal Component Analysis
KDE	=	Kernel Density Estimation
SVD	=	Singular Value Decomposition

Chapter 1

Introduction

This is a master thesis written in cooperation between NTNU and Hymatek Controls. Hymatek Controls is located in Oslo and delivers equipment and control systems to hydroelectric power plants worldwide. Hymatek is increasing its focus on condition monitoring and anomaly detection and has acquired a large dataset containing data from a five year period for 27 hydroelectric power plants. The goal of this thesis is to find possible use cases for condition monitoring in the provided dataset, research possible techniques and implement some of them in the discovered use case. Since this is a new research area for Hymatek, this thesis will cover a wide range of aspects.

This chapter provides an introduction to the work that has been done, describing the motivation and problem description for the thesis. A literature review looks into state of the art techniques used for condition monitoring. Finally, the outline of the report is presented.

1.1 Motivation

There is an increasing need for stable renewable energy in Norway, Europe and worldwide. According to Statkraft (2009) 99% of the Norwegian power production comes from hydroelectric power plants. Norway has almost 50% of Europe's hydroelectric capacity. According to Statkraft (2009) hydroelectric power is one of the most stable and cleanest energy sources one can find. Hydroelectric power plants have a high investment cost, but according to Selak et al. (2014) operation and maintenance costs are as low as 2% of the cost of the initial investment, this makes it one of the cheapest energy sources available. As many European countries are transitioning into renewable energy sources, the need for stable and controllable power sources grows. Both wind and solar power production are dependent on weather conditions, and their production capacity can change quickly and unexpectedly. Norway has been called Europe's renewable battery due to its hydroelectric capacity, and the power export capacity is growing. By 2021 Norway and Great Britain will be connected through a new underwater power cable. This shows how Norway is becoming a part of a more complex energy market. As the export capacity grows, green energy from Norway can be supplied to reduce the consumption of fossil energy from for example coal from the European continent. This means that increased productivity and efficiency in hydroelectric power plants effectively can help make the green shift in the European power production happen.

One way to increase a plant's productivity is to ensure that it is operable at all times. However, this is not possible since one has to perform maintenance on all parts of a plant. Planned maintenance is however not the only reason for a power plant shutting down. Components can break down outside of their service interval, leading to unplanned downtime. This is of course not only a case for hydroelectric power plants but more or less any industry or factory. Unplanned downtime is expensive for the energy company that operates the plant, and depending on the time of year and the duration of the breakdown, the cost can have a significant impact on their operation. Finding a way to avoid these breakdowns, and replacing unplanned maintenance with planned maintenance, is something all energy companies find interesting. A hydroelectric power plant contains several large components such as turbines, generators, transformers, hydraulic systems, valves, pipes, and so on. As an addition comes all the equipment needed to operate the plant. Keeping spare parts for all components would require enormous storage capacity, and ties considerable assets due to the cost of the spare components. Many components do not need to be often replaced, hence the spare can be left unused in storage for many years, and may even never be replaced. This is not an optimal solution.

Another approach is to try to eradicate unexpected breakdowns. This means that one needs to detect that a component is decaying so that one can plan an overhaul or replacement before it breaks down. Depending on the situation and the component, if a component breaks down, there is a risk that it can take several others with it. If a bearing on a turbine shaft breaks down, there is a risks that the damage will spread to the bearing on the other side of the shaft, and even to the turbine and generator. Continuous analysis of the condition of the bearing could help avoid this issue. This is known as condition monitoring. In its most complete form, condition monitoring gives an insight into the condition of the monitored components, giving an estimate to the remaining lifetime of the component. Introducing condition monitoring could help reduce the number of breakdowns, as components can be replaced before breaking.

There are many good reasons for pursuing the topic of this thesis, on a personal level it is motivating to work with real-world data, to learn the difference between theoretical and practical engineering. Looking for solutions that will improve the operation of a hydroelectric power plant, which can give even more clean energy is also motivating. Additionally, the fact that possible solutions found for the specific cases analyzed most likely can be adapted to other industries is also very motivating.

1.2 Problem definition

Hymatek provided no exact problem definition, they provided a dataset containing process-data from 27 hydroelectric power plants for the period 2013 – 2017, and gave no restrictions on which plants and cases to investigate. A historical incident log was also provided for all plants, with varying level of detail. Based on this, the problem definition was split into three main parts. As this is a thesis built upon an unknown dataset with unknown quality, an important factor is to recognize what the data can be utilized for, and what improvements could/needed to be done to make the data analysis more extensive. Ideally, the result of the thesis will be a system for condition monitoring and anomaly detection for a real-world case found in the dataset. However, since Hymatek is still in the start-up phase of its condition monitoring research, all experiences from this thesis are valuable. The goal is that the thesis can be used as a basis for further condition monitoring case analysis. The three parts are as follows:

- Finding methods for anomaly detection. Finding methods for feature extraction and dimensionality reduction to be used in addition to the anomaly detection techniques.
- Data preparation and case extraction. The provided data is not ready for analysis, and one needs to extract datasets for each of the plants. Once the data is on a format that can be analyzed, the historical data from the plants and the datasets need to be analyzed to build a case for testing of the anomaly detection techniques.
- Analyzing the case using the techniques found. Emphasize on how the different techniques perform, and what can/needs to be done to improve the performance.

1.3 Limitations

Since the thesis span over many different research areas, only one case will be analyzed, even if it should be possible to find several more from such an extensive dataset. Also, only a subset of the suggested techniques will be chosen for analysis, spanning from easy to more complex. There might be algorithms not tested that could be useful for the given case. The techniques are chosen in cooperation with Hymatek.

Availability of computing power and memory also introduces constraints to the analysis. All analysis is performed on a desktop from Dell provided by NTNU. This means that no analysis will be performed using the full datasets, they will be reduced by either feature selection or dimensionality reduction.

Condition monitoring is a broad field, and creating a full-scale condition monitoring system is too comprehensive. The thesis will, therefore, focus on anomaly/novelty detection for the given case. This means that one will train different techniques on normal operation data, and verify how well the different techniques can detect the anomaly found for the specific case.

1.4 Hydroelectric power production

99% of the Norwegian power production comes from hydroelectric power plants. Hydroelectric power is clean, reliable and once built, it produces cheap energy. According to Statkraft (2009) no other types of power plants have longer expected lifetime and higher efficiency than hydroelectric power plants. Statkraft also states that countries with the highest increase in energy demand are also countries with the highest unused potential for hydroelectric power. This means that hydroelectric power production can play a significant role in the hunt for meeting the global energy demand. There are many old power plants today, so there are also possibilities for modernization and extensions of already existing plants.

The principal behind hydroelectric power is simple. It utilizes the energy in running water, converting it to electric energy through a turbine. According to Paish (2002), there are two main types of turbines, impulse and reaction. The difference between the two types is how they create the rotational force. The impulse turbine rotor runs in air, and the rotational force is created by a water jet lead onto the turbine blades. The reaction turbine is submerged in water, and the rotational force comes from a lift force created by the oncoming water flow. For the impulse turbine, kinetic energy generates the rotational force, for the reaction both kinetic and pressure energy generates it. When the water hits the turbine blades, the water pressure is reduced to atmospheric pressure, and all pressure energy is converted to kinetic energy. For the reaction turbine, which is submerged in water, the water pressure is still high, meaning that one needs a casing which is sealed from atmospheric pressure.

The two most common turbine types are Francis and Pelton. The type used is defined by the size of the flow and the head of the water. Whether the turbine will be producing power only at optimal flow, or if it will be used for power production over a broader range is also a criterion.

1.4.1 Pelton turbines

The Pelton turbine is the most common impulse turbine. Water is lead onto to the turbine through a series of needles or valves. The turbine has a set of buckets located

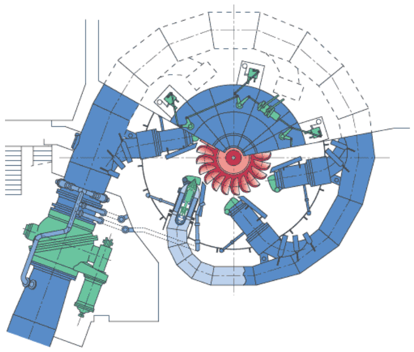

Figure 1.1: Cross section of a Pelton turbine with 6 needles. By Voith Siemens Hydro Power Generation.[a]

[a]Licensed under GFDL, Wikimedia Commons `https://commons.wikimedia.org/wiki/File:S_vs_pelton_schnitt_1_zoom.png`

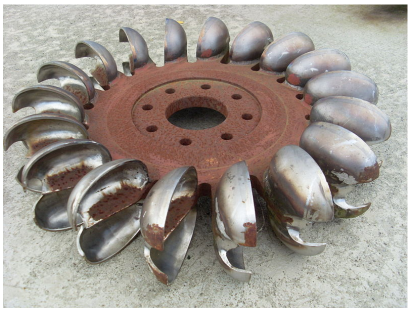

Figure 1.2: A Pelton turbine wheel, the split buckets are easily visible. By Zedh.

Licensed under CC-BY-SA, Wikimedia Commons. `https://commons.wikimedia.org/wiki/File:Pelton_400kW_roue_1.JPG`

on the turbine wheel which splits the water jet in two, where each half is deflected back and falls into a discharge channel, Paish (2002). A cross-section of a Pelton turbine is seen in Figure 1.1. As one can see the needles are divided equally around the turbine. The amount of produced power is controlled by the number of active needles, and by regulating their opening. Figure 1.2 shows how the buckets are designed. Where the water jets hit the buckets is easily verifiable by looking at the corroded parts of the buckets. The needles are controlled by a hydraulic system, which is controlled by the power plant control system. To ensure an even momentum on the turbine, all active needles should have the same opening.

1.4.2 Francis turbines

The Francis turbine is a reaction turbine. The water is lead onto the turbine through a set of guide vanes connected to the spiral casing, and lead out from the center of the turbine. Guide vanes control the amount of water lead to the turbine, and hence the amount of produced energy. A cross-section of a Francis turbine is seen in Figure 1.3.

All guide vanes have the same opening and are controlled through a ring mounted on the top of the turbine. Figure 1.4 shows a hydraulic actuator used to control the ring that operates the guide vanes. As for the Pelton turbine, the hydraulics is again controlled by the plant's control system. Åsnes (2017) tries to classify the condition of the guide vanes in a Francis turbine using different machine learning methods such as support vector machines (SVM), neural networks (NN) and regression.

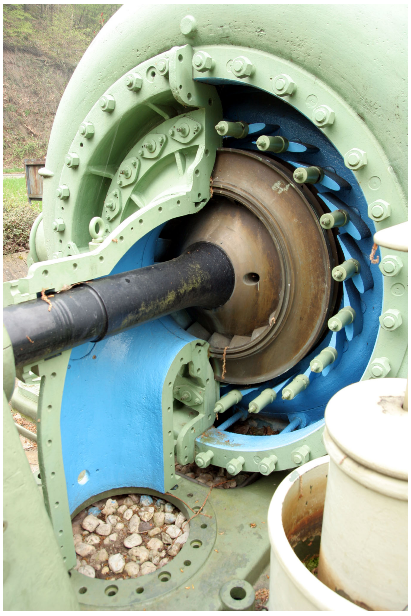

Figure 1.3: Cross section of a Francis turbine. The guide vanes are seen in almost closed position. By Armin Kübelbeck.[a]

[a]Licensed under CC-BY-SA, Wikimedia Commons, `https://commons.wikimedia.org/wiki/File:Fankel_Francisturbine_01.jpg`

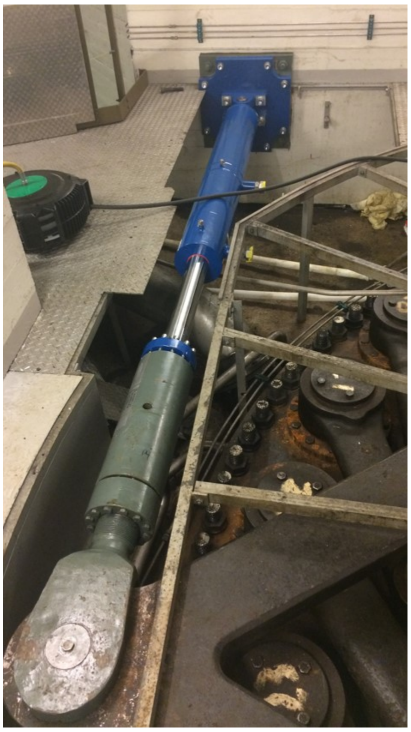

Figure 1.4: Hydraulic actuator for guide vane control. The actuator is mounted to a ring that controls the guide vane opening. Courtesy of Hymatek Controls

1.4.3 Other plant equipment

There are several other large components in a hydro-electrical power plant. Among them are;

- Generators
- Transformers
- Coolers
- Hydraulic systems

1.5 Condition monitoring for hydroelectric power plants

Selak et al. (2014) presents a complete condition monitoring and fault detection system for hydroelectric power plants, covering everything from data acquisition to fault detection methods. Cost reduction, increase in equipment availability and increased performance are presented as the benefits of introducing condition monitoring. They propose using condition monitoring in addition to preventive maintenance and claim that it is the most comprehensive maintenance scheme available. Data is transferred to a virtual diagnostics center, where a support vector machine (SVM) is used to diagnose the data. The proposed condition monitoring system is separated into six steps; data acquisition, data analysis and storage, data transferring, data selection, SVM training and SVM testing. It is proposed to sample data in four different ways to reduce the amount of data stored; periodically at a predefined sampling rate, when signal values exceed a threshold, during transients and sudden events.

Data that represents all known failure modes and normal data is needed to train a SVM to for fault diagnostics. The work is restricted only to include two variables at a time, as this enables scatterplotting variables against each other for visual confirmation of the normal operation. The training cases separate data into four different datasets; All input data, removing low power operations, removing high water flow operations and removing transients between operation regimes. Experts extracted 44 causalities that can be traced to a pair of process variables. Abnormal data is created as a complement to the normal data, by fitting an oversized boundary to the data from normal operation. As the SVM detector is only trained on two classes, data is only evaluated as normal or abnormal. It is claimed that two-dimensional models are sufficient to catch all known failure modes. The choice of using SVM is supported by Widodo and Yang (2007) that presents that SVM has been successfully used in the industry for several different use-cases.

Molina et al. (2000) introduces two neural network (NN) approaches to integrate into a decision system for hydroelectric power plant management. An expert system, an NN for acoustic prediction and an NN for predictive maintenance are proposed integrated into one system. It is argued that there are non-linear plant dependencies undetectable for a human expert, that the NN can find. As the expert system tries to mimic the response of a human operator, the system suffers from the same limitations as a human operator. Interpreting an expert system is however much more straightforward than interpreting an NN. Therefore a hybrid system is proposed, benefiting from the best of both worlds. The expert system and the NN for predictive maintenance depend on process data from the plant. The NN for acoustic prediction is fed with data recorded from sensors not used in the two other cases. 106 process variables are available for analysis, all are given as input to the NN, but to mimic the limitations of a human operator, the expert system is only considering 15 variables at a time. The NN for predictive maintenance needs data that represent both normal and abnormal data, for all known failure modes. A particular procedure combining expert knowledge and NNs are used to create data vectors with values for all 106 variables for the given errors created by the expert system. The error vectors are used to train the NN. Classes for the acoustic prediction network are created based on an experts classification of the normal and abnormal regime based on the plants generated power.

The proposed method was applied and tested on a plant in Zamora, Spain. It was found that the combination of the NNs and the expert system outperformed a human operator, but the human operator outperformed the methods individually. The NN for predictive maintenance gave the best individual results. The NN for acoustic predictions suffered from variability in the acoustic signals, but showed that it served as a good addition to the two other methods.

1.6 Report outline

Chapter 2 introduces different anomaly detection techniques, and methods for feature selection and dimensionallity reduction. Chapter 3 introduces the software and the libraries used in this thesis. Chapter 4 introduces the dataset and the cases analyzed. Chapter 5 walks the reader through the analysis before the discussion and conclusion are found in chapters 6 and 7.

Chapter 2

Theory

This chapter introduces three anomaly detection techniques that can be used for condition monitoring, and methods for feature extraction and dimensionality reduction to be used in addition to the anomaly detection techniques.

2.1 Anomaly detection

Pimentel et al. (2014) defines anomaly detection as the task of classifying test data that in some way differs from data used for training, hence this is one obvious approach to condition monitoring. Anomaly detection is like a one-sided test or one class classification, as one is not training on data that represents fault, only on data sampled from normal operation. As a model describing the normal operation of the system is learned, it is essential that one has samples of normal system behavior for all the possible states of operation. Anomaly detection avoids the issue with finding data that represents all possible failure modes. Such techniques are very valuable for industrial applications, where the need for condition monitoring is great, but identifying all possible failure modes and collecting data to represent them can be difficult and very costly. It is much easier to get measurements from a machine during normal operation, than during failure, and it is in many cases close to impossible to obtain as many samples of negative or faulty behavior as of normal behavior. Tarassenko et al. (2009) claims that modern high-integrity systems are so complex that they introduce many possible failure modes that are not very well captured by the instrumentation available. This can be verified by visiting a hydroelectric power plant. There is much instrumentation, and many alarms for different components, but all of these are linked to well-known failure modes. Anomaly detection introduces the possibility to detect unknown abnormalities, as it is designed to detect anything that deviate from normal system behaviour. It is, however, important to notice that anomaly detection methods are prone to suffer from a higher rate of false positives than when abnormal data is available for training Latecki et al. (n.d.). The performance of the anomaly detection is dependent on the choice of parameters, and it is essential to keep in mind whether it is false negatives or false positives that are of the highest concern

Since anomaly detection techniques are based on "normal" system behavior, they should, in theory, be able to detect any failure mode that can occur, if provided with the correct process information. Adding process signals that are not known to be connected to any of the known failure modes can be beneficial for such methods. This also makes sense when one does not know what to look for. One wants all possible information available. This leads to a significant remark, only getting a notification that an anomaly is observed, does not provide very much information. For these techniques to have any real value, one needs to be able to trace which component the anomaly comes from, and the magnitude of the anomaly. Once the anomaly detection techniques are trained, new data is given an anomaly score based on how similar the data is to the normal training data. This score can then be compared to a threshold, and the data will be marked as normal or abnormal. Using anomaly detection techniques avoids having to create data that represent the failure modes as seen in 1.5. This means that the methods are more straightforward to generalize and adapt to new uses.

Many of the techniques in the following sections are based on the suggestions from Pimentel et al. (2014) which provides an extensive review of novelty detection.

2.2 Reducing the input space

As the size of the input grows, the complexity and run-time of the techniques grow with it. It can become necessary to reduce the input space of the anomaly detection techniques to ensure reasonable run time. Interpreting and visualizing data of high dimensions is difficult. The feature size also introduces issues regarding memory and algorithm run-time, Guyon and Elisseeff (2003) and Dy and Brodley (2004). Reducing the complexity of the problem is heavily correlated with reducing the number of features. Feature selection techniques can also reveal unknown plant dynamics which can help to understand why some components fail.

"The problem is that not all features are important. Some of the features may be redundant, some may be irrelevant, and some can even misguide clustering results" from Dy and Brodley (2004) sums up one of the issues with datasets with many features. A concrete example of this is shown in Figure 2.1, her one can clearly see that X_2 does not provide any information on how the data points are clustered. Including this feature in a learning algorithm does not provide any information about the two classes found in the dataset, hence in the best case, the performance will be the same as if only X_1 was used. This is supported by Liu et al. (2010) that states that some features can be removed without lowering the performance of a learning algorithm.

There are two main techniques for reducing the input space, feature selection and dimensionality reduction. The former, try to remove the least informative and the redundant features from the feature set. The latter creates new features either as linear or non-linear combinations of the original feature set, hence one can remove the dimensions with little variance. Feature selection is again separated into a supervised and non-supervised selection. In the supervised case, one has a set of features and a target variable, and one want to keep the features that are related to the target variable. For unsupervised feature selection, there is no target variable to help remove uninformative or redundant features, and other factors must be used for selection.

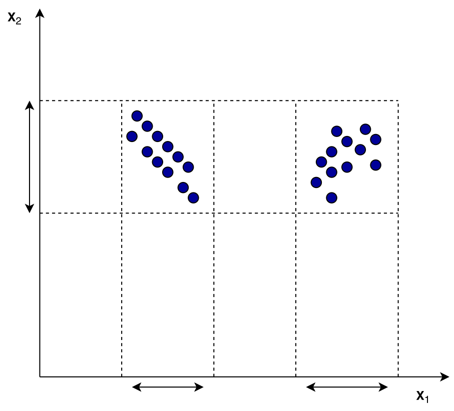

Figure 2.1: Example of a dataset with two features and two classes, only $\boldsymbol{x}_1$ is providing information about how to separate the classes.

2.2.1 Filter, wrapper and embedded methods

The different classes or types of methods for feature selection are split into three groups, filter, wrapper and embedded methods. This section will give a short introduction to each of them. According to Liu et al. (2010),m filter methods are methods that perform feature selection separate from the learning algorithm, and hence can be used no matter which learning algorithm is applied later. The wrapper methods need a predetermined learning algorithm. The features are then selected based upon the performance of the chosen algorithm. Finally, the embedded methods incorporate the selection of features in the training of the learning algorithms model. Since the filter methods are independent of the learning algorithm they are not biased with regard to the algorithm,Liu et al. (2010). This means that filter methods are to prefer when the learning algorithms are not known beforehand.

2.3 Supervised feature selection

In supervised feature selection one typically has a process variable or a target variable which represents behaviour one want to trace in other process variables. Tracing the target into new variables can yield information about what might be the cause of the behaviour, and the new variables can be used to detect unwanted behaviour earlier. An example is anomaly detection for a bearing. If a bearing is showing an increase in vibration and temperature, the variables can be used as targets for feature selection. The feature selection returns the variable which best describe the targets. These variables can then be used to analyze the cause. This section introduces two supervised feature selection methods.

2.3.1 Correlation and Mutual Information

Mutual information (MI), Kraskov et al. (2004) and Peng et al. (2005) explains how dependent or inversely how independent two variables are. It gives an understanding of how much knowing something about one variable reduces the uncertainty about the other. If two variables are completely independent, their corresponding MI = 0.

MI is defined as

$$MI(X,Y) = \int\int p(x,y)\log\frac{p(x,y)}{p_x(x)p_y(y)}, \tag{2.1}$$

where x and y are the two variables to compare, and $p_x(x)$ and $p_y(y)$ are their corresponding probability density functions. One benefit with MI is that it does not only find linear correlation between two variables. In other words, it finds dependencies between variables not necessarily shown in correlation. It serves as complementary technique to using co-variance or correlation for feature selection, Li (1990).

The features can also be selected using the above-mentioned correlation. The correlation between a feature and the target variable is found as

$$corr(X,Y) = \frac{cov(X,Y)}{\sigma_x\sigma_y} = \frac{E[(X-\mu_x)(Y-\mu_y)]}{\sigma_x\sigma_y}. \tag{2.2}$$

Here the features with highest correlation with the target variable will be chosen.

2.4 Unsupervised feature selection

Unsupervised feature selection is harder than supervised feature selection. The lack of a target variable removes the ability to easily interpret which variable dynamics that are important. Still, the need to reduce the feature set is prominent. Dy and Brodley (2004) describes the problem as follows, "The goal of feature selection for unsupervised learning is to find the smallest feature subset that best uncovers “interesting natural” groupings (clusters) from data according to the chosen criterion." The chosen criterion defines what is thought to be interesting natural groupings. There is no golden rule for defining this criterion, and a subset that is good for one purpose might not be relevant for others. Two possible methods for unsupervised feature selection follows.

2.4.1 Variance threshold

Variance threshold removes merely the features with variance below a set threshold. According to He et al. (2005) it is one of the most straightforward evaluation methods for unsupervised selection. One of the drawbacks of using variance to select features is that there might be many features with large variances that is non-informative with what one is looking for.

2.4.2 Laplacian score

Laplacian score for feature selection was proposed as an alternative to unsupervised feature selection by He et al. (2005). The method builds upon the assumption that features belonging to the same class or grouping can be found close together. The features are evaluated based on how well locality is preserved, which is found by the Laplacian score. The Laplacian score is calculated for each feature. A short introduction to the method is described below. More details is found in He et al. (2005).

Laplacian score is based on Laplacian Eigenmaps, which is an algorithm for dimensionality reduction. The algorithm creates a nearest neighbors graph with a node for each sample of each feature. Hence the dimension becomes m times the number of features. Two nodes share an edge if they are among the k-nearest neighbors to each other. Each edge is then given a weight based on the Gaussian radial basis function;

$$W_{i,j} = e^{-}\frac{||X_i - X_j{}^2||}{t}, \tag{2.3}$$

Then a graph Laplacian is computed which then again is used to calculate a Laplacian score. The main reason for choosing Laplacian score as one of the unsupervised feature selection methods is that it can compare and rate features against each other. This introduces a new dimension compared to the more naive variance threshold algorithm that only looks at one feature at a time.

2.5 Dimensionality reduction

One of the issues with feature selection is that the features that are removed may hold information that could help the learning algorithm. In dimensional reduction, new dimensions are created as a function of the features. This means that in some cases a reduced feature space of dimension n can hold the same amount of information as the original feature space of size m, $m > n$.

2.5.1 Principal component analysis

Principal component analysis (PCA) is a popular technique used for dimensional reduction. PCA is an orthogonal transformation that takes a set of possibly correlated variables, and transforms them into a set of non-correlated components, effectively reducing the dimensions of the data. These new dimensions are known as principal components. There are several different algorithms that can be used to calculate the principal components, one of them is singular value decomposition or SVD. SVD decompose the data into eigenvalues and eigenvectors. The larger an eigenvalue is, the better its corresponding eigenvector capture the variance of the data. All eigenvectors are orthogonal, this means that each vector introduces an entirely new dimension. In a high dimensional dataset with many features, there will most likely be features that are fairly similar. Figure 2.2 shows intuitively how the PCA algorithm works. Data samples

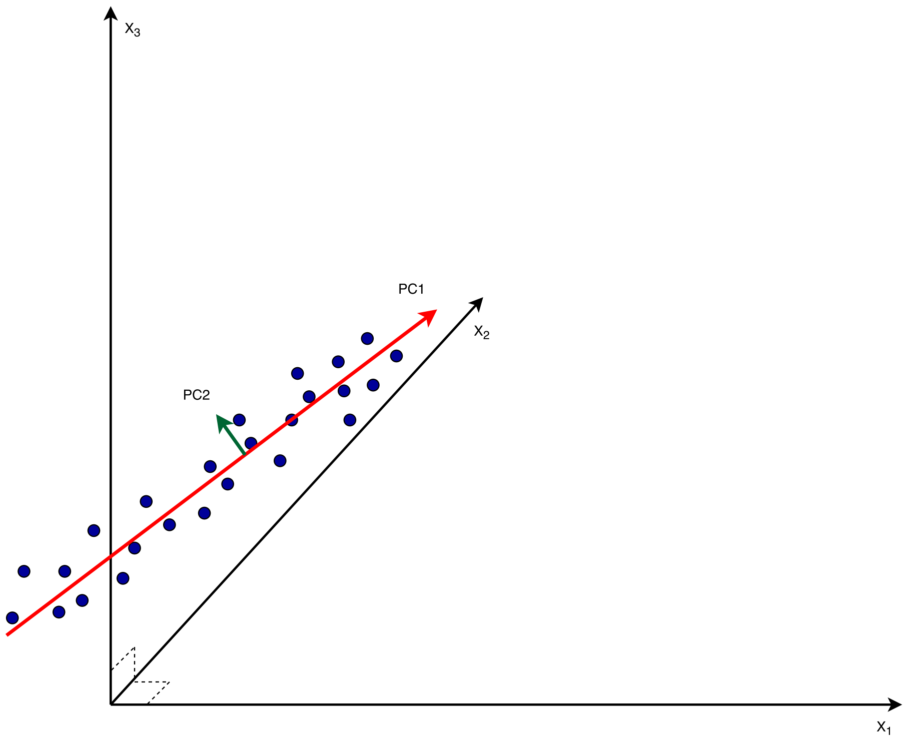

Figure 2.2: Illustration of principal component analysis. The two principal components are seen in red and green. Using only PC1 most of the variance in the data is still kept.

from a dataset with three features are shown in the plot. As the principal components in red and green shows, most of the variance in the samples can be represented using only one dimension. Adding the second dimension, only small residuals might be left. This illustrates how the PCA algorithm operates.

2.5.2 Kernel principal component analysis

Kernel PCA builds upon the PCA algorithm explained above, but it introduces a new trick. Where PCA is only able to create new features from linear combinations of the original feature set, kernel PCA uses the kernel trick to enable non-linear combinations as well. Take a two-dimensional dataset containing samples from two different classes. These classes are only linearly separable if the classes can be separated by a line. If one class surrounds the other class, there is no way to linearly separate the two classes in two dimensions. However, by extending the sample space into three dimensions, it might now be possible to separate the two classes with a hyperplane. Hence making a non-linear separation of the two classes possible, by still using linear methods. If the datasets are large, transforming all samples into a higher dimension can become computationally heavy. The kernel trick solves this problem without having to transform the samples. The kernel PCA extends the original PCA to a higher dimension using the kernel trick, enabling the extraction of nonlinear components.

By mapping the original data nonlinearly into a new feature space with $\phi(\boldsymbol{x}_1)$, and by performing PCA in this new feature space, one can get nonlinear principal components.

2.6 Anomaly detection techniques

In this section, three different techniques for anomaly detection is presented. SVM is a classifying algorithm that automatically classifies data as either normal or an anomaly. One Class Support Vector Machine (OC SVM) is presented as it only needs normal training data, removing the need for either sampling or artificially creating data for known failure modes. Kernel density estimation (KDE) is a method for estimating the probability density function or pdf of a set of variables. New data can then be evaluated on how well it fits the learned pdf. Long short term memory recurrent neural network is neural network designed for learning time series data.

2.6.1 One class support vector machine

The following section is a based on Åsnes (2017) and Hastie et al. (2001), SVM is presented before OC SVM, as it is an extension of SVM.

SVMs classifies data into classes by defining a separating hyperplane that best separates two different classes. Hence data is labeled based on which side of the hyperplane it lays. A SVM is capable of separating data both linearly and non-linearly. Non linear separation is achieved by using a kernel or similarity function. The kernel function describes how similar two feature vectors are by taking the inner product of the two samples in a higher dimensional space. The kernel function does this without having to transform the data into this dimension explicitly. Two commonly used kernels are the Gaussian kernel

$$K_g(\boldsymbol{x}^1, \boldsymbol{x}^2) = e^{\frac{\|\boldsymbol{x}^1 - \boldsymbol{x}^2\|^2}{2\sigma^2}}, \tag{2.4}$$

and the polynomial kernel

$$K_p(\boldsymbol{x}^1, \boldsymbol{x}^2) = (\boldsymbol{x}^{1^T}\boldsymbol{x}^2 + \alpha)^{\beta}. \tag{2.5}$$

These are just two of many examples. The Gaussian or radial basis function (RBF) kernel is a good first choice if a nonlinear boundary is needed to separate the classes, without knowing exactly what shape the boundary will take.

$$f(\boldsymbol{x}) = \beta_0 + \boldsymbol{\beta}^T\boldsymbol{x} = 0, \tag{2.6}$$

defines a hyperplane. For any two feature vectors $\boldsymbol{x}^{[1]}, \boldsymbol{x}^{[2]}$ that lie in or on the hyperplane where $\hat{x} = \boldsymbol{x}^{[1]} - \boldsymbol{x}^{[2]}$, gives $\boldsymbol{\beta}^T\hat{\boldsymbol{x}} = 0$ and hence, $\boldsymbol{\beta}$ is a scalar multiplication of the normal vector to the hyperplane. Where

$$\boldsymbol{\beta}^* = \frac{\boldsymbol{\beta}}{||\boldsymbol{\beta}||} \tag{2.7}$$

is the normal vector. Inserting any feature point $\boldsymbol{x}_0$ that lies on the hyperplane defined by Equation 2.6, yields $\boldsymbol{\beta}^T\boldsymbol{x}_0 = -\boldsymbol{\beta}_0$. By using $\boldsymbol{x}_0$ as any feature vector from origo to the hyperplane and the normal vector $\boldsymbol{\beta}^*$, the distance for any given feature vector $\boldsymbol{x}$

to the hyperplane is given by

$$\begin{aligned}\boldsymbol{\beta}^{*T}(\boldsymbol{x}-\boldsymbol{x}_0) =& \frac{\boldsymbol{\beta}^T}{||\boldsymbol{\beta}||}(\boldsymbol{x}-\boldsymbol{x}_0) \\ =& \frac{1}{||f'(\boldsymbol{x})||}f(\boldsymbol{x}). \end{aligned} \tag{2.8}$$

A decision rule

$$y^i(\boldsymbol{x}^{iT}\boldsymbol{\beta}+\beta_0) \geq 0 \tag{2.9}$$

can then be created using the hyperplane. Since there are two classes they are defined as $y_0 = 1$ and $y_1 = -1$. The decision rule tells which class a feature vector belongs to. Note that points belonging to the negative class should yield negative distance to the hyperplane, and hence the decision rule ensures that all points are classified correctly.

Now that an expression for the distance from a hyperplane to any given feature vector in the feature space is defined, one can look at how to find the optimal hyperplane to separate the two classes. The margin or the width between the two closest point from both classes can be defined as $M = \frac{2}{||\boldsymbol{\beta}||}$, and hence maximizing the distance can be formulated as minimizing

$$\begin{aligned} J(\boldsymbol{\beta}) =& \frac{1}{2}||\boldsymbol{\beta}||^2 \\ s.t \quad y^i(\boldsymbol{\beta}^T\boldsymbol{x}^i+\beta_0) \geq& 1 \quad \forall \ i. \end{aligned} \tag{2.10}$$

The constraint here ensures that every point is classified correctly. This can lead to a very complex boundary or hyperplane, in addition there is always a risk that data is labeled incorrectly. Therefore slack variables are added to enable some wrong classifications as seen in the updated minimization problem

$$\begin{aligned} J(\boldsymbol{\beta}) =& \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{N}\xi_i \\ s.t \quad \xi_i \geq 0, y^i(\boldsymbol{\beta}^T\boldsymbol{x}^i+\beta_0) \geq& 1-\xi_i \quad \forall \ i. \end{aligned} \tag{2.11}$$

In the case of OC SVM, Schölkopf et al. (2001), there is only one class available, so the problem is formulated slightly different,

$$\begin{aligned} \min \quad J(\boldsymbol{\beta}) =& \frac{1}{2}||\boldsymbol{\beta}||^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i - \rho \\ s.t \quad (\beta \cdot \phi(x_i) \geq& \rho - \xi_i \quad \forall \ i \\ \xi_i \geq& 0 \quad \forall \ i. \end{aligned} \tag{2.12}$$

The main difference is the parameter ν which serves as a upper fractional bound for accepted number of outliers, and as a lower bound for the fraction of support vectors used by the algorithm. ρ is simply an offset. OC SVM is typically trained on normal data, fitting the decision boundary close to the normal data pattern. Anomalies that deviate from the normal data will then be found on the other side of the separating hyperplane, and classified as an anomaly.

2.6.2 Kernel density estimation

KDE, creates an estimate of the probability density function or pdf of the data, Latecki et al. (n.d.). The data is then evaluated by how likely it is to belong to the estimated density function. This is a non-parametric technique meaning that the data is not assumed to take the shape of a given distribution. The probability density function is estimated using a set of kernels that are distributed across the data. The probability density is then estimated at each kernel location based on data that lie within a local neighborhood of the kernel. The density found at a point x within its neighboring n points is given by,

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n}K(\frac{x - x_i}{h}). \tag{2.13}$$

Here the $K()$ is the kernel function, and h is a tunable hyperparameter for variance known as the bandwidth. This parameter smooths the distribution, the larger the bandwidth the smoother the density estimation becomes. Hence a too small bandwidth will lead to overfitting, and a too large bandwidth will lead to underfitting.

The kernel function can take on many forms, where the Gaussian kernel,

$$K(x - x_i; h) = e^{-\frac{||x - x_i||^2}{2h^2}} \tag{2.14}$$

is the most commonly used. KDE is mentioned as a method for anomaly detection in Pimentel et al. (2014). The distribution of normal data is modeled by KDE, and new data is evaluated on how likely it is to belong to the learned distribution.

2.6.3 Neural networks

The introduction to NN is based on Åsnes (2017) and Hastie et al. (2001).

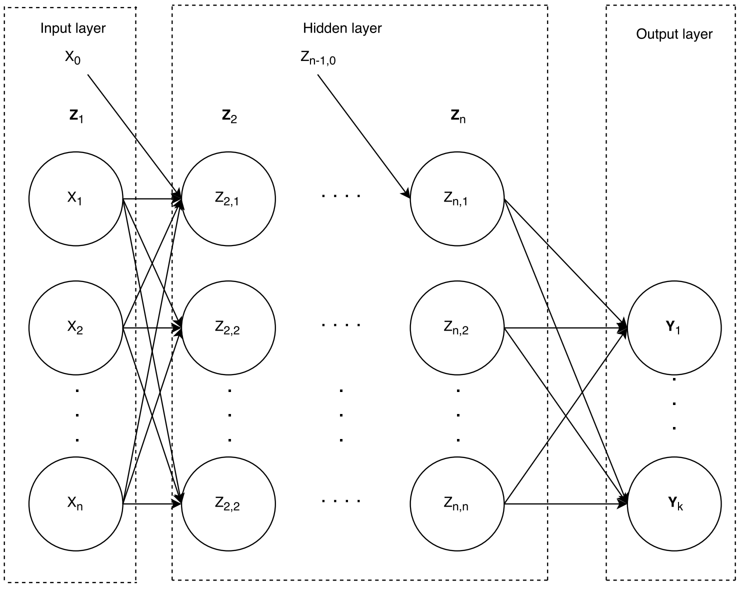

Figure 2.3: A graph visualizing a n-level deep neural network. All hidden layers are fully connected. The size of the input and output layer depends on input size, and if the NN is used for regression or classification.

An NN is an n-class classification or regression algorithm. For anomaly detection, an NN can be trained to recreate the input of normal data. If the network is trained with enough normal data, it should be able to learn a good representation of patterns in this data. Abnormal data will in some way deviate from normal data, and hence its patterns are not familiar to the network. This will then result in recreated data that deviate more than normal from the input. This is also known as auto-encoding. There are many types of NN specialized in solving specific tasks. Recurrent neural networks (RNN) are designed to learn trends seen over a period and are therefore a good option when working with time series. Within the recurrent NN, there is a type known as LSTM or long short-term memory, which has shown to outperform the standard RNN when it comes to long-horizon trends, Goodfellow et al. (2016). The following section introduces LSTM RNN by first introducing NN and RNN.

A visual representation of an NN is a graph with nodes and vertices as shown in Figure 2.3. As seen, an NN consists of three parts, an input layer, a hidden layer, and an output layer. The input layer is the feature vectors for the data samples, the hidden layer does all the computation, and the output layer makes the prediction. An NN with only one layer and a limited number of neurons can approximate any function, Hastie et al. (2001). This shows the ability an NN has for finding complex non-linear patterns.

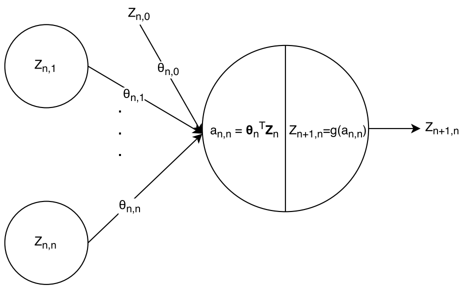

Figure 2.4: Visualization of a single neuron. The input is the output from the previous layer. The output is a scalar decided by the activation function $g(a)$

Each of the nodes marked $z_{n,n}$ seen in Figure 2.3 is known as a neuron. A single neuron is visualized in Figure 2.4. Each neuron does two things. First, it takes the dot product of the input vector and the weight vector,

$$a = \boldsymbol{\theta}_n^T \boldsymbol{Z}_n, \tag{2.15}$$

Then the output $Z_{n+1,n}$ is calculated

$$Z_{n+1,n} = g(a_{n,n}). \tag{2.16}$$

Here $g(a)$ is known as an activation function. There are many different options for activation functions, the most common are shown in Table 2.1. It is by using nonlinear activation functions, that a NN is enabled to approximate any function. If only linear functions are used, it can be verified that adding more layers does not but complicating a linear interpretation, Hastie et al. (2001). The algorithm learns by updating the weights θ in each neuron.

Function	
Identity	$g_k(a) = a$
Sigmoid	$g_k(a) = \frac{1}{1+e^{-a}}$
Softmax	$g_k(a) = \frac{e^{Tk}}{\sum_{i=1}^{K} e^{Ti}}$
Tanh	$g_k(a) = \frac{e^{a}-e^{-a}}{e^{a}+e^{-a}}$
Relu	$g_k(a) = max(a, 0)$

Table 2.1: Different activation functions

The final layer calculates the output $\hat{y}$. The number of elements in the final layer depends as mentioned on if it is classification or regression. For regression, the final layer has one element.For classification, it has as many elements as there are classes. For binary classification, the Sigmoid function is typically used. For k-class classification, the Softmax function is used, and for regression the identity function.

The weights of an NN are updated through forward and backward propagation. In forward propagation the input is fed through the mathematical operations in the network as seen in Figure 2.4 and 2.3 and a set of labels or a value is predicted. The cost function

$$E(\Theta) = \sum_{i=1}^{M} \frac{1}{2}(\hat{y} - y)^2,$$

expresses how well the NN is able to correctly predict its output. The NN weights or coefficients $\Theta = [\Theta_1 \ldots \Theta_n]$ are then updated through a step called backpropagation. The cost function is minimized as a function of the weights. Gradient descent or another optimization algorithm is used to calculate each of the different parameters in Θ's contribution to the cost. Θ is updated as

$$\begin{aligned} \Theta :=& \Theta - \alpha \nabla E(\Theta) \\ \Theta_{ij} =& \Theta_{ij} - \alpha \sum_{k=1}^{M} \frac{\partial E_k}{\partial \Theta_{ij}} \end{aligned} \tag{2.17}$$

Here α is the learning rate, and takes a value between [0,1]. The prediction errors are back-propagated to calculate the error in each neurons prediction. This prediction error is then used to update the corresponding weights for the neuron in the matrix Θ. A full mathematical proof can be found in Åsnes (2017)

The forward and backward-propagation steps are then iterated over until a given number of iterations are reached. A complete pass forward and backward of all training examples are known as one epoch. When the training set is large, it is often split into smaller batches to speed up calculations. The batch size explains how many samples are included in each forward and backward pass. Splitting the data into batches enables the algorithm to take advantage of parallelism, running the separate batches on separate CPUs or GPU's. How many epochs it takes for an NN to learn a good representation of the data can vary a lot. A technique called early stopping is introduced to ensure that the NN is stopped when the network no longer is showing sign of increased test predictions. Running an NN for too many epochs on a training set can cause the NN to overfit to the training data, giving poorer general performance than if stopped earlier.

A deep NN can learn more complex patterns than a shallow. However, the deeper the NN is, the more processing each iteration through it will take. Deep networks also increases the risk of overfitting. A good approach is to start with a shallow network, and increase the depth and width until you see that adding new layers, does not yield better performance. Other means to handle overfitting is regularization (making it costly to have large weights) and dropout. Dropout removes some edges between the neurons between each iteration, simplifying the network structure.

2.6.4 Recurrent neural network

Recurrent NN is a type of NN designed for sequential data. The RNN is specialized for processing values that belong to a sequence, as often is the case with time-series data. It is designed to scale to much longer sequences than what is practical for standard fully connected NN. The key is parameter sharing, which enables one to apply a model to different sequence lengths and generalize across them. The following quote from Goodfellow et al. (2016) explains this well. "For example, consider the two sentences “I went to Nepal in 2009” and “In 2009, I went to Nepal.” If we ask a machine learning model to read each sentence and extract the year in which the narrator went to Nepal, we would like it to recognize the year 2009 as the relevant piece of information, whether it appears in the sixth word or in the second word of the sentence. Suppose that we trained a feedforward network that processes sentences of fixed length. A traditional fully connected feedforward network would have separate parameters for each input feature, so it would need to learn all the rules of the language separately at each position in the sentence. By comparison, a recurrent NN shares the same weights across several time steps". Figure 2.5 shows a representation of a RNN. The RNN can be unfolded into a set of networks for a finite number of time steps as shown in the right of Figure 2.5, once the network is unfolded in this way the backpropagation can be computed, as explained in the above section. The internal state of the NN is updated as

$$c^{(t)} = f(c^{(t-1)}, x^t; \theta), \tag{2.18}$$

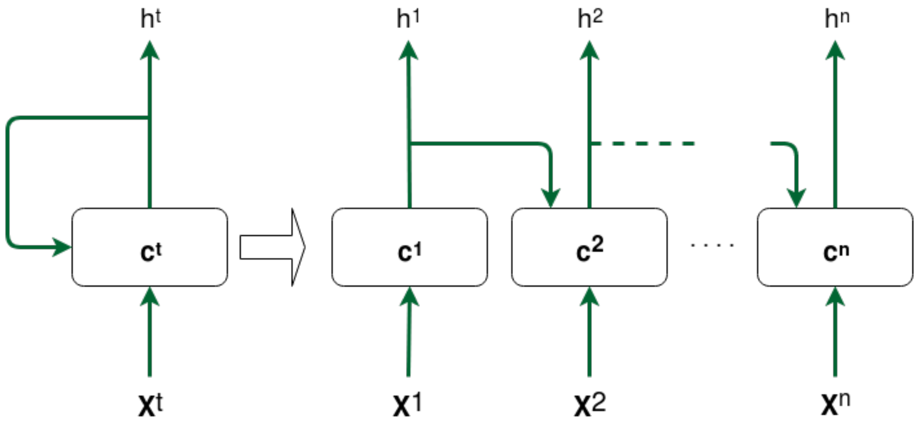

Figure 2.5: A recurrent neural network, shown in its compact form to the left, and in its unrolled form to the right.

here it becomes apparent that the current state is dependent on the previous state, which again is dependent on its predecessor, and so on. This clearly shows the recurrent pattern.

One of the pitfalls of the RNN is that as the depth of the network grows with the number of time steps. This introduces two problems known as vanishing and exploding gradient. This phenomenon is explained in detail in Pascanu et al. (n.d.). The short version is that as the number of time steps grows, the size of the network grows. This means that the number of computations in the backpropagation algorithm also grows. It can be shown that each of the n layers as seen in Figure 2.5 gets its parameters updated as a partial derivative of the cost function for its prediction, in addition to the partial derivative of the next network. The exploding or vanishing gradient problem can be found by examining how $\frac{\partial x_t}{\partial x_k}$ is calculated. The following derivation from Pascanu et al. (n.d.),

$$\begin{aligned}
\frac{\partial \mathcal{E}}{\partial \theta} &= \sum_{1 \leq k \leq t} \frac{\partial \mathcal{E}_t}{\partial \theta} \\
\frac{\partial \mathcal{E}_t}{=} &\sum_{1 \leq k \leq t} \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{x_t}} \frac{\partial \boldsymbol{x_t}}{\partial \boldsymbol{x_k}} \frac{\partial \boldsymbol{x_k}}{\partial \theta} \\
\frac{\partial \boldsymbol{x_t}}{\partial \boldsymbol{x_k}} &= \prod_{t \geq i \geq k} \frac{\partial \boldsymbol{x_i}}{\partial \boldsymbol{x_{i-1}}},
\end{aligned} \tag{2.19}$$

shows how the partial is calculated. As seen in Equation 2.19 the partial of $\boldsymbol{x}_t$ becomes a product of all internal states of the timesteps between t and k. Hence, if these values are ≥ 1 the gradient will grow, and if they are ≤ 1 they will decrease. As the number of timesteps grows, the gradient then runs the risk of either explode or vanish. A possible solution to this problem is presented in the next section.

2.6.5 Long short term memory recurrent neural networks

Long Short-Term Memory RNN or LSTM RNN is an extension of the RNN seen in subsection 2.6.4. The idea behind it is to create loops where the gradient can flow for long duration's, and hence solving the issue with vanishing and exploding gradients. LSTM was introduced in Hochreiter and Urgen Schmidhuber (1997), and has been found very successful in many different applications. Figure 2.6 shows the building blocks of the LSTM. The LSTM consist of four internal layers, forget layer, input gate layer, two tanh layers and the output layer. All of those layers are a separate NN. The previous cell state is fed in at the top of the LSTM. The previous state is kept based on the previous output $\boldsymbol{h}^{t-1}$ and the current input $\boldsymbol{x}^t$ which is fed through the forget layer. If none of the previous states is seen as relevant, the output will be a vector of zeros. If the previous state is seen as extremely important, it will output a vector of ones. Next, the information in the new input is analyzed. The tanh layer creates an estimation of the current LSTM state $\boldsymbol{c}^t$ based on $\boldsymbol{h}^{t-1}$ and $\boldsymbol{x}^t$. This value is then scaled by the output of a sigmoid layer $\boldsymbol{I}^t$ which decides which of the internal states to update. $\boldsymbol{c}^t$ is then computed as a function of the previous state and the new state

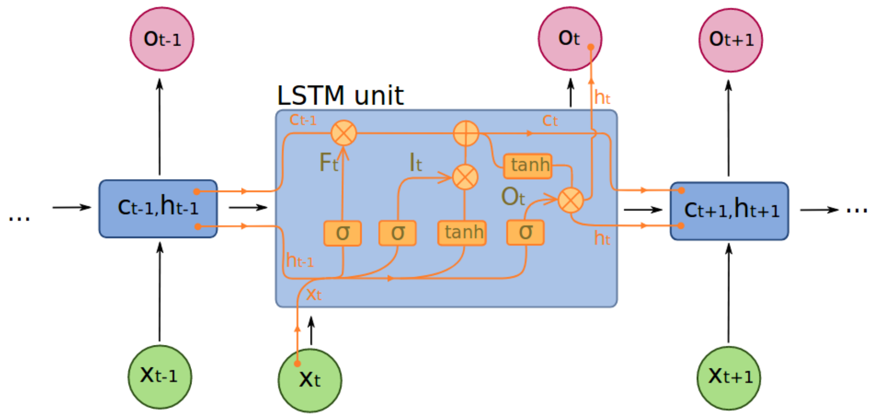

Figure 2.6: An LSTM RNN. By François Deloche [a]

[a]Lisenced under CC BY-SA 4.0, `https://commons.wikimedia.org/wiki/File:Long_Short-Term_Memory.svg`

estimation. The output is calculated by sending the current cell state through a tanh layer before a sigmoid layer decides which part of the cell state to output.

LSTM is proposed as a scheme for anomaly detection in Malhotra et al. (2016) and Malhotra et al. (n.d.). It is shown that the scheme is working for both periodic and aperiodic time series. LSTM RNN has been successfully used for handwritten text recognition and speech recognition. The paper proposes an encoder-decoder scheme designed for anomaly detection. A time series is encoded and then decoded back to reconstruct the input series. The reconstruction error, the error between the input and the reconstructed input is then used to identify anomalies. In the training phase, the system is only shown normal data. Hence it only learns to reconstruct the normal system behavior. The network should then not be able to reconstruct anomalous data as precisely as normal data. The paper concludes that LSTM networks can be a viable approach for anomaly detection. It also shows promise in detecting anomalies from unpredictable time-series.

Chapter 3

Implementation

This chapter introduces the system implementation and the software packages used during the thesis.

3.1 Python

All code is developed in Python 3.6 through Anaconda and Spyder. Anaconda is a Python distribution with focus on data science and machine learning. Anaconda simplifies installation and handling of packages and package dependencies. In addition, it allows exporting of the package environment, for automatic setup at new locations. Spyder is a Python IDE that has a similar layout to Matlab, where objects and variables can be accessed through a work space. This makes it a good alternative for data science.

A library for reading and preparing the data from the energy company is created from scratch. The library is meant to be used for further analysis at Hymatek. The code used for OC SVM is mostly reused from the project thesis. All other code is written during the master thesis. All methods mentioned in theory is implemented and ready for use. Libraries are created that makes it easy to analyze the data on the different methods. The next subsections are taken from Åsnes (2017)

3.2 Scikit-learn

Scikit-learn, Pedregosa et al. (2011) and *Scikit-learn 0.19.1* (n.d.) is an open source toolbox available for Python. It includes many different machine learning algorithms and the tools needed to preprocess and analyze data. The framework created by Scikit-learn is used throughout this thesis, as it includes implementations of OC SVM and KDE. It also has implementations of many other machine learning algorithms, if one wants to expand the analysis with new methods. In addition to the being a great machine learning library for Python, the website [1]also explains the main theory behind the different algorithms, and what the scikit-learn algorithms are built upon in an easy to find manner. This makes it a good choice when one want to understand what is making the algorithms work.

3.3 Keras

Scikit-learn does not include neural networks. Different libraries were considered before Keras was chosen for the NN analysis. Keras Chollet (2017) is a Python library that simplifies creation of NNs in Python. Instead of having its own implementation of NNs, it uses other NN Python libraries as backends. This means that you can run the analysis using, for example, Google's TensorFlow through Keras. At this moment Keras supports Google's TensorFlow, Theano developed by LISA Lab at Université de Montréal and CNTK developed by Microsoft. This means that by using Keras, only one line of code needs to be changed if one want to change between one of the options above. Developing code using one of the NN libraries directly would make switching between the different options more cumbersome. However, one important factor to keep in mind, is that Keras is not officially supported by any of the backend developers. This means that not all of the original functionality is guaranteed to be supported in Keras. Regardless, the fact that Keras is easy to use and its focus on enabling fast experimentation, are the main reasons for developing in Keras. Keras supports the many different types of NN, including the LSTM RNN used in this thesis. Keras also

[1]`http://scikit-learn.org/stable/`

enables the user to choose to run on either CPU or GPU, making it flexible and easy to optimize to the available hardware. More information about Keras can be found at [2].

3.3.0.1 Google TensorFlow

For this thesis, only the TensorFlow backend is used. TensorFlow Abadi et al. (2016) is developed by The Google Brain project, and it started as the DistBelif project back in 2011. Tensor flow is the second generation, built upon the experiences from DistBelif. Tensorflow has support for parallelism and can have many different devices collaborating on the same problem. As stated in Abadi et al. (2016), "a computation expressed using TensorFlow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards." This shows the flexibility TensorFlow introduces. More information about TensorFlow can be found at [3].

[2] `https://keras.io/`
[3] `https://www.tensorflow.org/`

Chapter 4

Analyzing the dataset and building a case

This chapter introduces the dataset and work done to enable information extraction and analysis of the provided data. The case for the analysis is also introduced.

4.1 Dataset

Hymatek controls provided a dataset from a Norwegian energy company, containing process information from 27 hydroelectric power plants logged from 2013 to mid 2017. The data is not pre-processed in any way, and come just as it is logged at the company. A big part of the thesis has been spent on exploring the data, finding out what is logged and what can be used. The data is split into five folders, one for each year. In each folder, several different files are stored. Table 4.1 shows the different file types that are found in the dataset. The file names indicated the frequency, sampling type and sampling duration of the data. The files are not separated by plants, only by date. Hence, a file containing data sampled every second for any given start and stop date contains data from all plants for that given period. In addition to the data files, a

Interval	Average	Max	Min	Actual
Daily	x	x	x	-
Hourly	x	x	x	-
Minutely	x	x	x	-
Secondly	-	-	-	x

Table 4.1: Table showing the available data for the different sampling frequencies.

meta-data file is provided where plant name and process signal for each tag can be found.

The total size of all files exceeded 90Gb. This means that one can't merely load all data into the computer memory, and work with it from there. The information needs to be extracted one plant at a time, for each of the different sampling rates and stored in a way that enabled fast and efficient loading.

4.1.1 Old and new data format

Regardless of sampling-rate and data type, the files are all in the same format. As seen in Table 4.2, each line holds a tag, a time of sampling and process value. Once a file is

Tag	Timestamp	Value
192390514	20170101000000000	0.897244155
192391514	20170101000000000	-0.549806237

Table 4.2: Example of the structure of the original datasets.

loaded the tag is replaced by the process-signal name found in the meta-data file. To enable interpretation and analysis of the data, it is decided that new datasets needs to be created on the format shown in Table 4.3.

Time stamp	**P. variable 1**	**P. variable 2**	**..**	**P. variable n**
time 1	NaN	2	..	NaN
time 2	3.00	NaN	..	0.00214
:	:	:	..	:
time n	1.00	NaN	..	0.814

Table 4.3: Example showing the structure of the data for each plant after reconstruction.

4.1.2 Overview of the datasets

Once the data is reconstructed as specified in subsection 4.1.1 one can look into what information it holds. The number of available process signals varies a lot, from above 250 to below 20, as can be seen in Figure 4.1. This was used as a first stage filtering to reduce the number of plants to examine. The plants with fewer than 30 process-signals are dropped from the analysis, reducing the number of plants in the dataset to 15.

After discarding the plants with few signals, the type of process-signals available is looked into. The more data sampled from different parts of a plant, the better. Having data from several parts and components of a plant can enable finding a link between unknown components and sensors. Figure 4.2 shows the different types of signals found in the datasets for the different plants. As can be seen, the temperature is one of the most common signals for all plants. There are also many signals of type "Def Type Måling", which is a common name that covers all signals not defined with a unique tag. Pressure, level, vibration, and so on are signals covered by this tag. Since all of the remaining plants has process-variables of different types, none are removed.

An issue arose when looking in detail at the datasets. Preferably, all process-signals should be sampled periodically and at the same time. This is however not the case, only at hourly resolution are the data matrices complete. At higher sampling rates there are a lot of missing data. At the highest sampling frequency (every second), there are many time-stamps where none of the process signals are sampled, and those that are sampled, often only samples one process-signal. This means that one either has to work with only hourly sample rate, work with incomplete datasets or reduce the

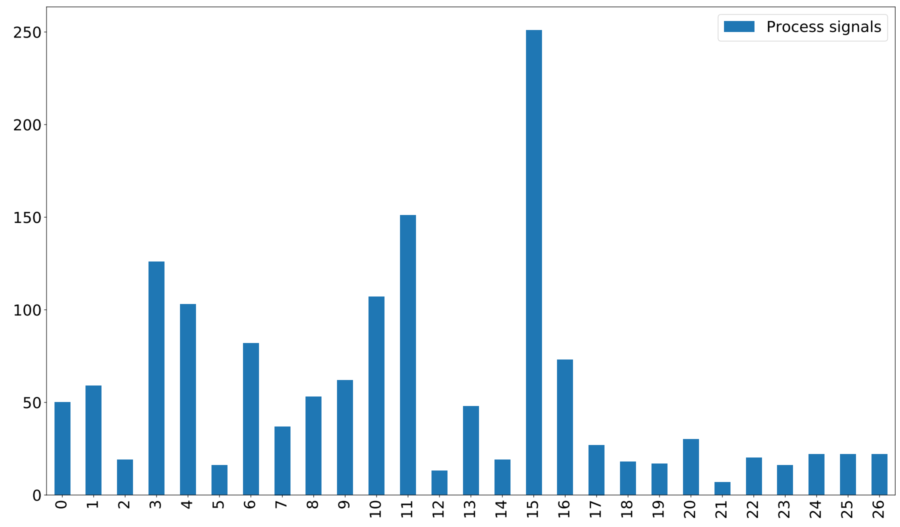

Figure 4.1: Overview of the number of process-signals or features for each of the 27 plants in the dataset.

number of process signals to investigate. Either the equipment used for sampling is not working as it should, or the varying sampling rate must be on purpose. It could be that a similar sampling scheme to the one seen in Selak et al. (2014) is used. This is however not possible to confirm from the data alone, and the energy company did not provide this information. This sheds light on the fact that having lots of data is not always a sign of quality. The data sampled in 2013 is poorer than the later years, and hence it is dropped from any further analysis.

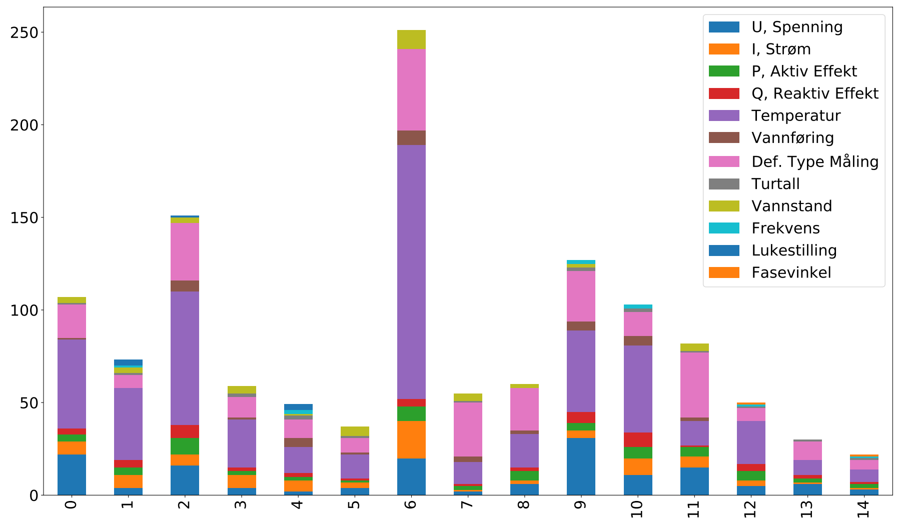

Figure 4.2: Figure showing the different process signals or features found at each of the 15 plants with more than 30 process-signals.

4.1.3 The historical log of the plants

Once an overview of the data was in place, the energy company was contacted, requesting an incident log from the remaining plants. Incident logs for the entire sampling period, for all of the 27 power plants, were received. The level of details in the logged incidents is very varying from plant to plant. Some plants have reported over 200 incidents, while others barely exceeded 20. Many of the incidents are also minor incidents like a broken light bulb or non-functioning tools. Finding interesting incidents was time-consuming. Several interesting incidents were found, but very few were reported to occur more than once. Having few incidents makes it hard to separate valuable information from noise. Only having a limited number of faults can easily lead to overfitting. There is always a risk that randomness can cause the correlations and variable dependencies found. This needs to be taken into consideration when working such cases.

In the log for one of the power plants, it was found an error with the needles for a Pelton turbine. The error reported that as the opening of needle four of turbine two decreased below 46%, it started to lag behind needle two. The plant has two turbines that both have four needles. A search through the 14 remaining power plants showed that there are two additional plants with Pelton turbines. These plants have however not reported any problems with the Pelton needles. This could then be a possible reference for normal operation. Also, it opens the possibility for testing how adaptable the anomaly detection methods are. The plant with the reported error is from now referenced as Plant 1.

4.2 Pelton needles case

The Pelton needle case was chosen for further analysis for several reasons. Firstly it is a critical part of the Pelton turbine, if the needles are not operating as they should it will affect the power produced by the turbine. The needle openings are a big part of a complex control system, that controls that the turbine holds constant speed. Finding a method that can give early warnings about incidents like the one above, means maintenance can be planned, and components overhauled before the system condition becomes too severe. Secondly, having data from different plants makes it possible to analyze how well the methods and techniques used to adapt to new data. This is an important factor, if one can develop a method that is transferable with little to no adaptation need between plants, the time for commissioning will be significantly reduced. In addition, this case fits well as an extension of Åsnes (2017), where the condition of the guide vanes for a Francis turbine was analyzed. Finally, quality and available process variables vary a lot from plant to plant. One of the most significant issues was that the majority of the process signals were not sampled at a constant frequency. The Pelton needles were found to be sampled in a manner that allowed the use of data from two separate plants.

4.2.1 Pelton needle incident 02.03.2017

As the incident reported that one needle was lagging behind another, it could indicate pairwise operated needles. This can also be justified from a practical perspective if the needles are placed opposite of each other they will give the turbine an even momentum as long as they lead the same amount of water onto the turbine. To confirm this, scatter-plots were created where each needle is plotted against the other. To enable this the needle process signals are pairwise extracted from the rest of the dataset, and all time-stamps where both needles are not sampled are removed. Figure 4.3 shows the scatter-plot of the pairwise operated needles for both turbines at Plant 1. As can be seen in all four plots, the majority of the data follows a linear dependency between the needles. However, several data-points deviate from this, especially for needle pair [2, 4] on turbine 2, which had the reported incident. The plot in the bottom right corner shows what appears to be operational problems. It is only needle [1,3] on turbine 1 seen in the upper left plot, which follows the linear pattern.

Figure 4.4 shows a scatter plot of the needles [2,4] for the last 150 days up to the reported incident. The sample color changes as days go by, as seen in the color bar. It becomes clear that the most deviating samples are from the days around the incident. There are very few dark samples that deviate from the linear pattern, indicating normal

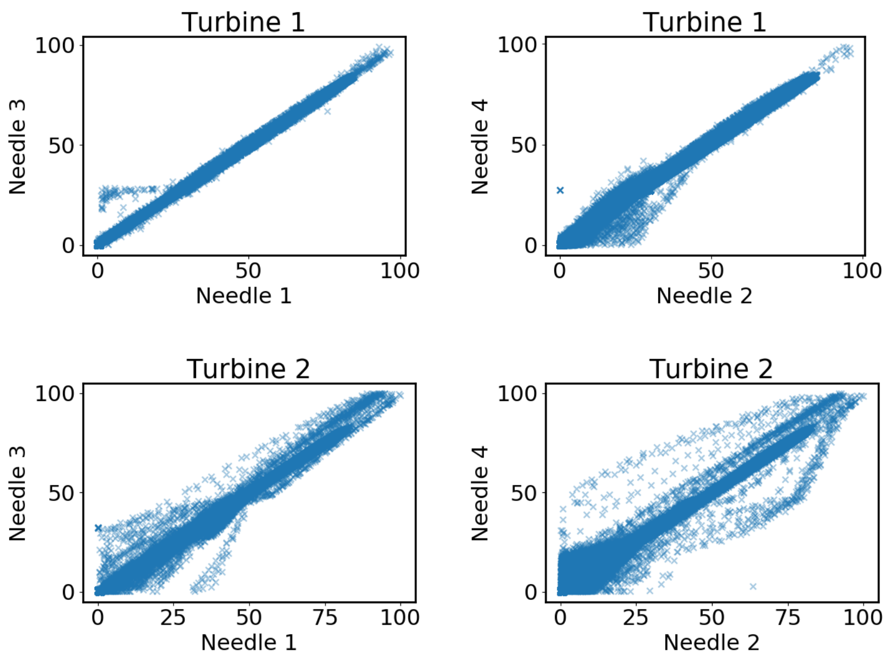

Figure 4.3: Needle openings for the pairwise operated needles for both turbines at Plant 1, scatterplotted in four sub-figures.

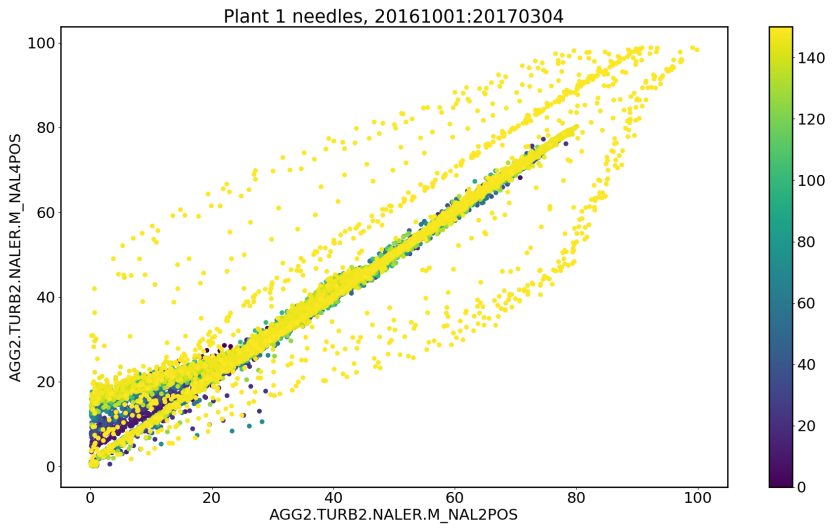

Figure 4.4: Scatterplot of needle [2,4] at Plant 1 turbine 2 of the 150 days leading up to the incident. The color of a sample indicates when it is sampled. The color bar to the right shows how the color change with time. Day 0 is 150 days before the incident. The opening of needle 2 is along the x-axis, and the opening of needle 4 is along the y-axis.

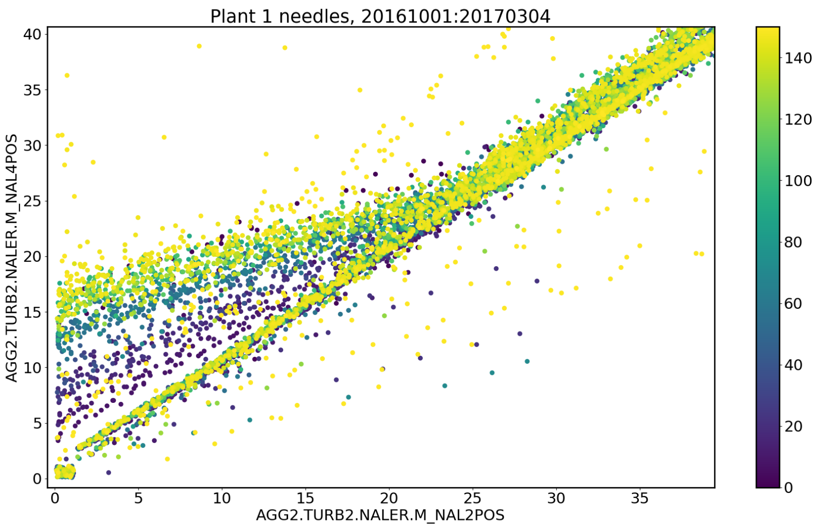

Figure 4.5: Plant 1 turbine 2 needle [2,4] scatterplot of the 150 days leading up to the incident zoomed in. How to interpret the figure can be seen in Figure 4.4.

system operation 150 days before the incident. Figure 4.5 shows the same plot zoomed in. Looking at the bottom left corner of the plot, one can see that needle 4 is lagging behind needle 2. It is also clear that the lag is worsening with time, as the color is shifting from dark to yellow. What happens the day of the incident is hard to explain, and the data takes on a completely new pattern. Unfortunately, it was not possible to get the full information from the energy company about what maintenance was performed to fix this problem. It is therefore assumed that the incident is a result of the system degradation seen leading up to 02.03.2017 and that this pattern can be used to detect similar system degradation before the performance becomes unacceptable.

To investigate how the turbine had performed earlier in the sampling period, Figure 4.6 was created. It shows the needles positions on top of each other, and the error between them. The reported incident 02.03.2017 can be seen in the plot. The claim made earlier that the system is gradually degrading can be supported by the growing error up to the incident. Interestingly, there also seem to have been some issues with

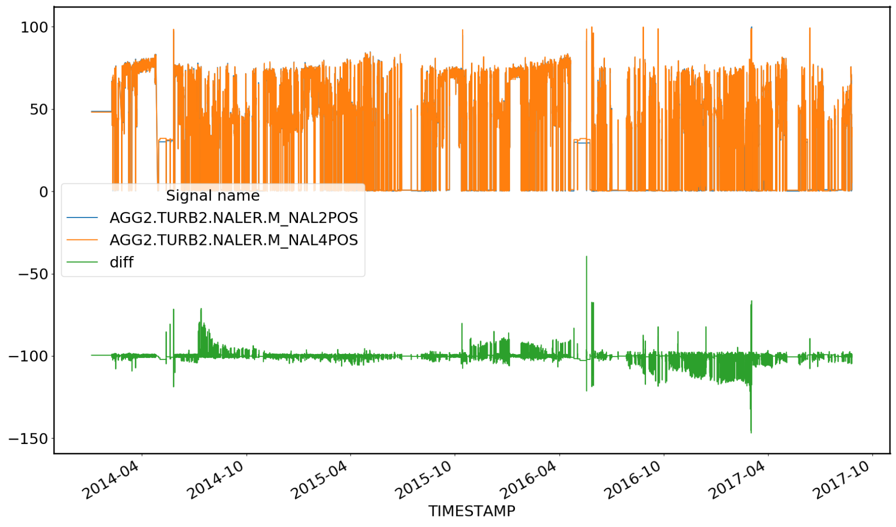

Figure 4.6: Plant 1 turbine 2 needle [2,4] plotted on top of each other in orange and blue at the top of the figure. With an offset of -100 the needle difference is seen in green. The dates are seen along the x-axis.

the operation in both 2014 and 2016. There is, however, no reported incidents in the plant log.

4.2.2 Other Pelton needle incidents

The log reported two other issues with the turbine needles at Plant 1, that is observable in a scatter plot. This is seen in Figure 4.7 and 4.8. Both are start-up failures due to problems with the needle operation.

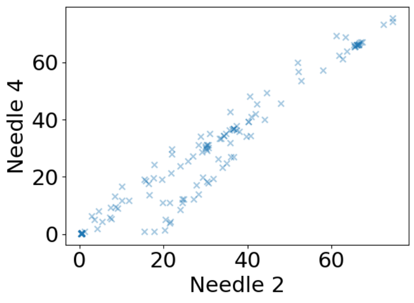

Figure 4.7: Plant 1 turbine 1 start failure, 10.12.2014. Pairwise operated needles are scatterplotted against each other.

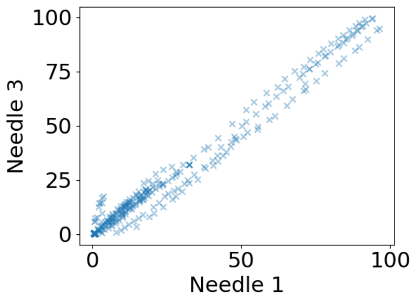

Figure 4.8: Plant 1 turbine 2 start failure, 25.08.2016. Pairwise operated needles are scatterplotted against each other.

4.2.3 Other plants with Pelton turbines

Further investigation showed that the second plant with Pelton turbines also has two turbines with pairwise operated needles. The third Pelton plant only has one turbine, and the needles are not pairwise operated. The turbine use from 1 to 5 needles depending on the produced power. Data from this plant could be used to generalize the anomaly detection methods to handle any combination of needles, but this will not be covered in this thesis. The needle operation for the Plant 2 is seen in Figure 4.9. One can see that turbine 1 has no sampling on needle 1, but the three other pairs are sampled correctly. Plant 2 further motivates that there is something wrong with the needles for plant 1. All needle pairs follow the linear pattern much better and have very few anomalies.

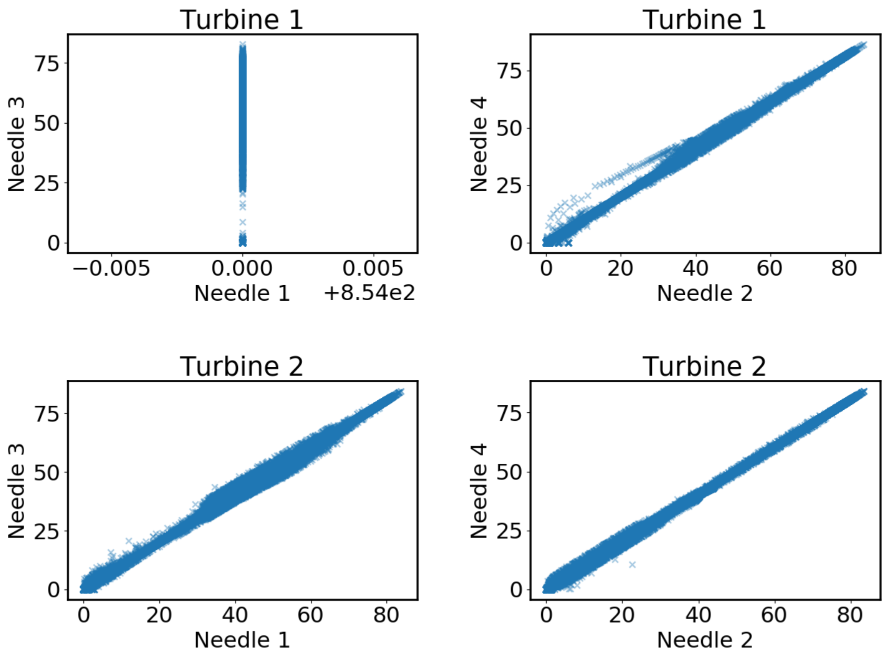

Figure 4.9: Needle openings for the pairwise operated needles for both turbines at Plant 2 are scatterplotted in four sub-figures.

4.2.4 Artificial needle error

Due to not having any incidents similar to the one seen at plant 1 turbine 2, an artificial error replicating the error is created. The primary motivation is to see when an error that is increasing over time is detectable. A subset of the data sampled at plant 2 is altered to replicate an error similar to the one seen for plant 1. By doing so, one knows exactly when the error starts occurring. This can then be used to evaluate how early different anomaly detection techniques detect anomalies.

The turbine needles are controlled through a hydraulic system. A hydraulic system can suffer from many problems which can cause operational problems, such as pressure drops due to both external and internal leakage. External leakage is often broken hoses or pipes. Internal leakages are within the pumps and actuators. Both types of leakages reduce the system pressure and can slow down the motion of the system. Oil contamination can also be an issue if filters are not maintained at a given interval, unfiltered oil can then lead to clogged components, which can lead to pressure drops. The leading theory of the problems seen at plant 1 is that the oil flow is restricted in one direction. As needle 4 closes, the larger the differential pressure caused by the water head being restricted becomes. As the differential pressure grows, closing the needle requires more force. This means that if the hydraulic system is suffering from reduced capacity in one direction, the motion of the needle will slow down as the needle close.

Figure 4.10 shows the colored scatter plot for the plant 2 data with and without the artificial error. As can be seen, it has a very similar pattern to what was seen at plant 1 in the days leading up to the reported incident. Figure 4.11, shows a close up on the lower openings of the needle. The deviation between the two needles increases with time, as seen in the case of plant 1. The artificial error is added to data sampled from needle pair [2,4] on turbine 2 at Plant 2. Data from 20161001 to 20170401 is used, and the artificial error is added from 20170201.

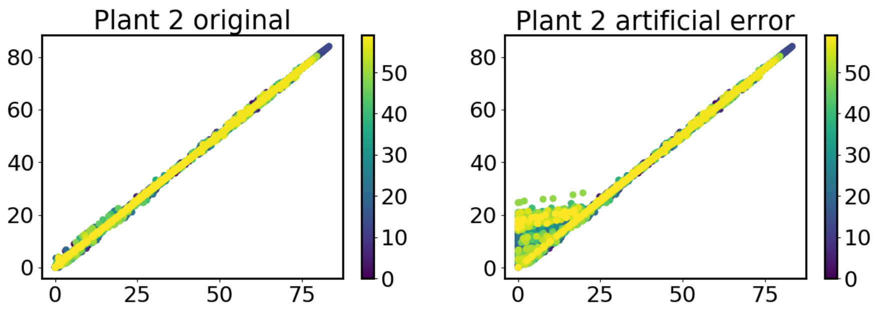

Figure 4.10: Colored scatterplots of the data before and after adding the artificial error. The original data is seen to the left. How to interpret the figure can be found in Figure 4.4.

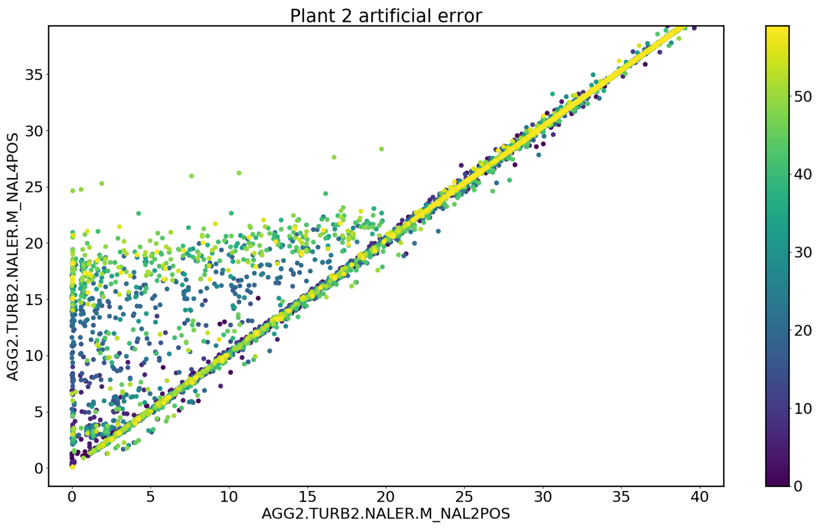

Figure 4.11: Colored scatterplot of the artificial error for the lower openings. How to interpret the figure is found in Figure 4.4.

Chapter 5

Results

This chapter contains the anomaly detection analysis performed on the data presented in 4.2. The goal is to investigate how well different methods detect anomalies when only being exposed to normal data during training. Three anomaly detection techniques are used, OC SVM, KDE and LSTM RNN. First, the different training sets and test cases are presented, then some prepossessing steps are introduced before the optimal hyperparameters of the different methods are presented. Finally, the performance of the different methods on the different cases is analyzed.

5.1 Training sets and test cases

Due to the aperiodic sampling of process signals, only the needle positions are sampled often enough to be included as features in the analysis. This means that the techniques for feature and dimensionality reduction presented in chapter 2 will not be used. The anomaly detection techniques are trained on three different training sets and evaluated on four test cases. The goal is to analyze how the methods evaluate the different cases when trained on different training sets.

The three different training sets are presented in Table 5.1. Training set 1 is from Plant 1 turbine 2 after the reported incident when the needle operation is back to

Set	Plant	Needle	Train	Validation	T. samples
1	1	2,4	20170304-20170701	20170701-20170825	9449
2	2	2,4	20160422-20160506	20160506-20161001	12870
3	2	2,4	20160101-20160506	20160506-20161001	84263

Table 5.1: Table over the three different training sets.

Case	Plant	Turbine	Needle pair	Data
1	1	2	2,4	20140101-20170304
2	2	2	2,4	20161001-20170401
3	1	1	2,4	20140801-20150201
4	1	2	1,3	20160601-20161101

Table 5.2: Table over the four different test cases.

normal. There is no golden rule for the ratio between testing and validation, but according to Kohavi (1995) somewhere around 1/3 of the data is common to use for validation. Only seven months of data is available from Plant 1 turbine 2 after the needles are fixed. Therefore five months are used as training data and little under two months for validation. This resulted in a training set of 9449 samples. The validation data is used to ensure that the methods are not overfitting on the training data. Training set 2 and 3 are taken from Plant 2 turbine 2, which has no reported incidents and normal system operation over the entire sample period. Set 2 is similar to set 1 in size, set 3 is 10 times larger. This is done to evaluate if the performance of the anomaly detection techniques changes when exposed to more or less data during training. The same data is used for validation for both sets. The validation data is larger simply because there is more data available. Data from 2016 was picked because it represented all operational modes well.

Table 5.2 shows the four test cases. The first case is all data from Plant 1 turbine 2 up until the incident, as the data after is used for training. As this is the needle pair with the recorded incident, it is interesting to see how the different methods evaluate the data, especially building up towards the incident. Case 2 is the artificial case introduced in 4.2.4 where an artificial error is added to normal data from plant 2. The artificial error is added from 20170201. Hence, all data from 2016 is normal plant data. The artificial case enables one to evaluate how early the methods detect anomalies when knowing exactly the start date of the system degradation. Case 3 and 4 are data surrounding the reported start failures introduced in 4.2.2. The start-up failures will help evaluate how well the methods recognizes known abnormal patterns in the data.

Training and evaluating the methods on data from different plants and turbines enables one to evaluate cross plant performance. This is very interesting, if it can be shown that one can simply deploy a pre-trained anomaly detection system at new plants with little to no alteration, this will greatly reduce the cost of installing such systems.

5.2 Prepossessing steps

The data needs to be preprocessed before it can be analyzed. First, all time steps where not both needles are sampled are removed. Secondly, Tarassenko et al. (2009) talks about the importance of removing steady-state samples from the dataset, to avoid biasing the data. The needles are constantly changing during operation, hence the only steady state is when the plant is not operating. Steady state is removed by removing all samples where the process signal values are close to zero (< 2) and unchanged since the last sample. Having a stationary time series is important. As Manuca and Savit (1996) explains, many of the common techniques for analyzing time series assume that the given series is stationary. A stationary time series is said to be independent of time, and don't have any correlations with seasonality. As the no operation data is removed from this dataset, the data is seen as stationary. The next step is to scale the data. For OC SVM and KDE, the data is scaled to zero mean and unit variance, for the LSTM RNN the data is scaled within a range of [0,1]. This was found to yield the optimal behavior for the different algorithms. To enable cross testing between plants, the transformations are calculated on plant 1 turbine 2 needles [2,4] and used for all datasets. Finally, the data is checked for outliers using scatterplots, as having anomalies in the training data could damage the systems ability to detect anomalies.

5.3 Hyperparameterization

The methods chosen for the anomaly detection have several different parameters or hyperparameters. To ensure that the methods perform optimally, it is necessary to search for the best combination of these parameters. To enable comparing the different cases analyzed, an optimal parameterization is found and used for all cases. The hyperparameter search is performed on the data sampled from plant 1 turbine 2 needle [2,4], after the reported incident.

5.3.1 One class support vector machine

The OC SVM has two tunable parameters. These are γ and ν. There is, however, no good way to automatically find the optimal parameterization for unlabeled data. By default, one has to inspect the one class SVM performance for different hyperparameterizations manually. It is attempted to create a custom cost function for automatic hyperparameterization. The cost function builds upon the idea that a good separating hyperplane or decision boundary lay close to the samples. The OC SVM implementation in scikit-learn stores the signed distance from each sample to the separating hyperplane. The distance can be used to evaluate the shape of the decision boundary.

$$\text{score} = 1 - \text{abs}(\text{std}(\boldsymbol{x})) - \text{abs}(\text{mean}(\boldsymbol{x})) - \frac{\text{outliers}}{\text{inliers}}100 \tag{5.1}$$

is proposed as cost function where $\boldsymbol{x}$ is the vector with the signed distances for each sample to the separating hyperplane. Several setups for a cost function were tried. The chosen performed well for the cases studied in this thesis. One cannot guarantee that it will perform adequately for different cases.

As explained in theory, ν is the upper boundary for the fraction of outliers. Since the data is assumed to be outlier free, it is natural to pick values for ν that lie around $\frac{1}{N}$ where N is the number of samples. γ is a measure of the complexity of the decision boundary and is therefore chosen to span a wide range of values. A grid search is performed over the values seen in the list below.

- $\gamma = [0.01, 0.05, 0.1, 0.2, 0.4, 0.8, 1, 2, 4, 8, 16]$
- $\nu = [\frac{1}{25N}, \frac{1}{10N}, \frac{1}{5N}, \frac{1}{2N}, \frac{1}{N}, \frac{2}{N}, \frac{5}{N}, \frac{10}{N}]$

Score	γ	ν
0.975	0.05	$\frac{1}{N}$
0.970	0.1	$\frac{1}{2N}$
0.961	0.1	$\frac{1}{N}$
0.960	0.05	$\frac{2}{N}$
0.957	0.2	$\frac{1}{N}$
-44.437	8	$\frac{1}{10N}$
-51.780	8	$\frac{1}{25N}$
-57.461	0.1	$\frac{1}{25N}$
-71.082	16	$\frac{1}{10N}$
-506.263	16	$\frac{1}{25N}$

Table 5.3: Table showing the five best and five worst scores for the gridsearch after optimal hyperparameters for the OC SVM. N = number of samples.

Table 5.3 shows the score for the five best and five worst hyperparameterizations. It can be seen that a too low ν or a too large γ yields the worst performance. Figure 5.1 shows the decision boundary for the classifier that scored the best with regards of the cost function defined in equation 5.1. All samples located inside the red boundary line, are classified as normal, and points outside as anomalies. The boundary covers all samples with a reasonable distance from the samples. As can be seen, most of the support vectors are located at the upper and lower range of the samples, showing the importance of having training data that span the full operating range of the machinery. A couple of samples in the mid area are also picked as support vectors. This gives a relatively close, but smooth boundary. The boundary for the worst parameterization is seen in Figure 5.2, it is easily verified that the boundary is to complex, due to a large γ

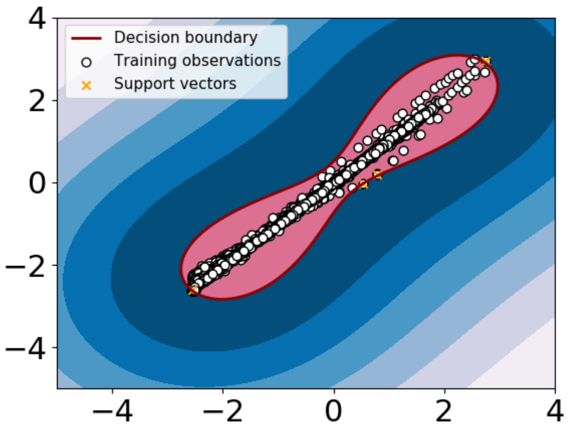

Figure 5.1: Decision boundary for the best scoring classifier. The scaled data is seen as white dots, x-axis is needle 2 y-axis needle 4. All data inside the red boundary are classified as normal, all data outside as anomalies. The support vectors are seen in orange crosses. The color outside the boundary indicates the distance to the separating hyperplane.

and a very small ν. The decision boundaries for the remaining classifiers seen in table 5.3, are found in appendix A. It is verified that all of the high scoring classifiers have a similar shape of the decision boundary. Note that the plots are produced using the scaled data.

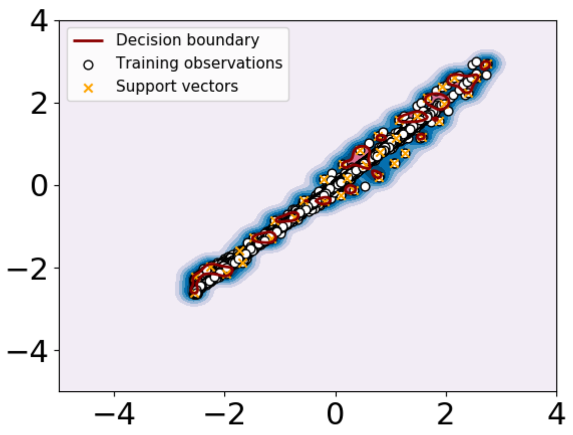

Figure 5.2: Decision boundary for the worst classifier. How to interpret the figure is found in Figure 5.1

5.3.2 Kernel density estimation

The optimal hyperparameters for KDE were found using the built-in grid search method in Scikit-learn. The different parameterizations are scored based on how well they replicate the distribution of the original data. As with OC SVM, KDE only has two parameters, bandwidth and kernel type. The bandwidth serves as a smoothing parameter, the higher the bandwidth, the smoother the density estimation will become. The KDE grid search was performed over a bandwidth from $[0, 1]$ at 0.1 steps, and with Gaussian, Tophat, Epanechnikov, Exponential, Linear and Cosine as kernels. The optimal parameter combination was found to be the Gaussian kernel with 0.1 as bandwidth.

5.3.3 Long short term memory recurrent neural network

A custom grid search for the optimal hyperparameterization for the LSTM network was created. As the network is trained to recreate the input, the Euclidean norm of the difference between predicted and real value is used to score the performance of the network. The models were trained on a training set and tested on a validation set. The different parameters are shown in the list below.

- Epochs = 100
- Batch size = [8, 32, 128, 1024]
- Neurons = [2, 4, 8, 16, 32, 64, 128, 256]
- Layers = [1]

A preliminary search found that increasing the layer depth introduced overfitting, and therefore only one layer was included in the grid search. The network consists of an LSTM layer with the number of neurons found in the grid search. In addition, there is a dense linear layer that recreates the input. The shape of the input is as required by Keras, transformed into a 3 dimensional tensor with dimension [num_samples,num_features,1]. Table 5.4 shows the score for the five best and five worst parameter combinations. The grid search is performed with a maximum number of epochs set to 300. Early stopping is used to avoid overfitting, and the optimal number of epochs for each parameterization is seen in the Table 5.4. Figure 5.3 and 5.4 shows the training history plots for the best and worst configuration of the . As can be seen, the worst configuration is stopped after five epochs due to an increasing error in the test prediction. This is typical for models with high variance, which are prone to model random noise in the training set. Looking into the parameters for this configuration one can see that the batch size is very small, meaning that the network weights are updated after only being exposed to a small number of samples, this seems to cause overfitting. The best configuration is not stopped before it reaches 134 epochs, and as can be seen, it performs equally on both training and test data. The history plots for the eight other configurations can be found in appendix B.

Score	**Epochs**	**Batch size**	**Neurons**	**layers**
$3.516e-03$	134	256	64	1
$3.521e-03$	288	512	16	1
$3.571e-03$	300	1000	64	1
$3.646e-03$	300	1000	128	1
$3.650e-03$	300	256	128	1
0.026	12	16	2	1
0.026	6	16	128	1
0.026	6	16	256	1
0.031	9	8	16	1
0.046	6	8	256	1

Table 5.4: Top five and bottom five scores after the grid search for optimal LSTM RNN structure and parameters, the lower score the better the performance.

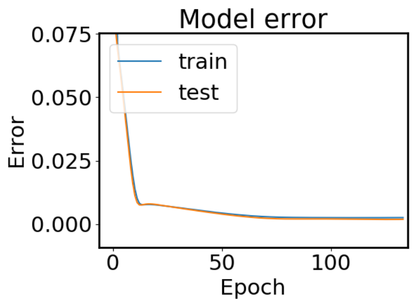

Figure 5.3: Training history for the best LSTM RNN configuration. Test and training prediction error is shown as a function of epochs.

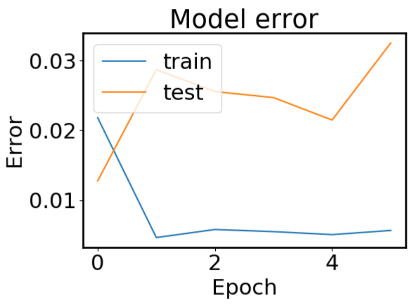

Figure 5.4: Training history for the worst LSTM RNN configuration. Test and training prediction error is shown as a function of epochs.

5.4 Anomaly detection

Of the three methods used in the analysis, only OC SVM is a classifier. KDE and LSTM RNN gives an anomaly score. Hence, they will be referenced as anomaly scorers. This means that to identify a sample as an anomaly, one needs to find a threshold for the anomaly score. This is not covered in the thesis, and a threshold evaluating 0.1% of the most extreme scores are used as an example.

5.4.1 Training set 1, test case 1 and 2

Figure 5.5 shows training set 1, and how the different methods evaluate the training data. The validation data can be found in Figure C.4. As can be seen, the anomaly scores for the KDE and LSTM RNN anomaly scorers are similar in shape but different in magnitude. One data point is standing out from the rest for all three methods, but the anomaly score is still very close to the average. The same point is also evaluated as an anomaly for the OC SVM classifier.

Figure 5.6 shows the anomaly scores for the three different methods on test case 1. The top window shows the opening of the two needles. As can be seen, it is hard to spot any deviation between the two needles, even for the data from the reported incident at the end of the plot. For both the KDE and the LSTM RNN anomaly scorers one can clearly see that the anomaly scores are increasing from September 2016. Several samples after January 2017 are located above the proposed threshold line, and a very high score is given for the data sampled during the incident. The anomaly score for both scorers is more than doubled from the score seen during the lowest periods. This shows that one can clearly identify that the scorers are detecting patterns in the data not seen during training, in the period leading up to the incident. This is verified by comparing the magnitude of the anomaly scores with the magnitude seen in the training set in figure 5.5. Interestingly, there are also several samples spread out over the entire sampling period that is given a high anomaly score. This can be because the training set does not cover all modes of operation, operational problems or problems with data sampling. This trend is also seen in Figure 4.6. The power plant log did not report any operational issues except for the one in March 2017, but from the authors

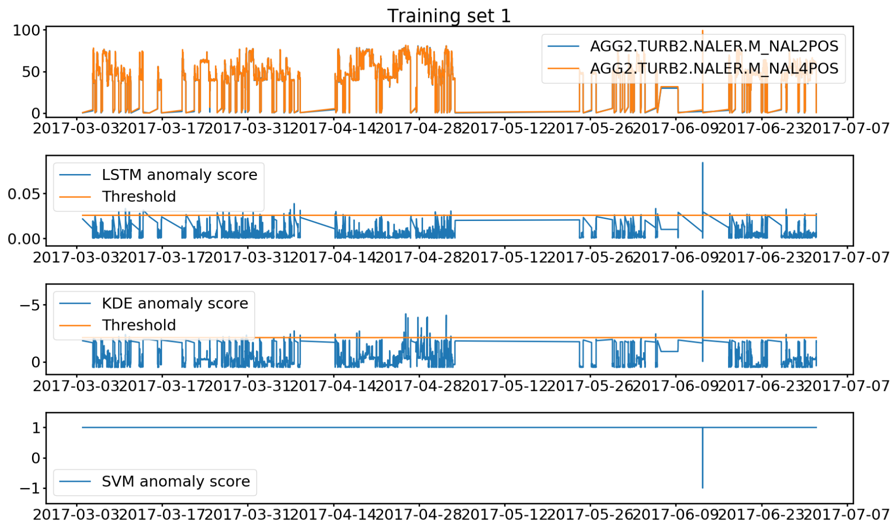

Figure 5.5: Training set 1 with the anomaly results for the different methods. The upper subplot shows the needle openings on top of each other. The second subplot shows how the anomaly scorer evaluates the data seen in the upper subplot, the higher the score, the more anomalous the data is. The third subplot shows the KDE scorer, the more negative the score is, the more anomalous the data is. The orange line in the two score plots indicates the 0.1% most extreme scores. The last subplot shows how the OC SVM classifies the needle openings, normal = 1 and anomalous = -1.

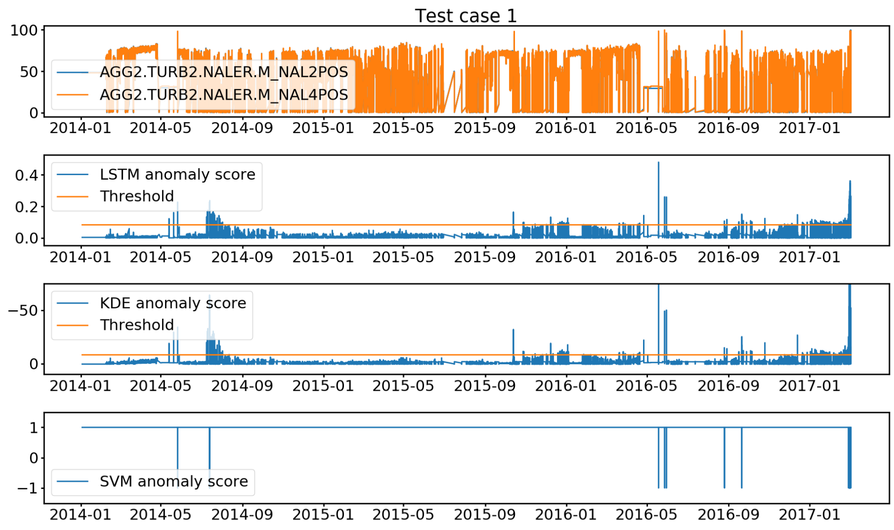

Figure 5.6: Anomaly results for the methods trained on training set 1 and evaluated on test case 1. For plot interpretation see Figure 5.5.

	KDE validation	**KDE case 1**	**LSTM validation**	**LSTM case 2**
min	-2.671	-275.66	1.659e-04	9.716e-05
max	0.491	0.491	3.032e-2	0.400
mean	-0.296	-0.728	3.318e-03	6.200e-3

Table 5.5: Table comparing the statistics for the KDE and LSTM RNN anomaly detectors on the validation and case 1 data for training case 1

experience from the industry, it is not uncommon that not all maintenance work is logged. The OC SVM classifier is not able to detect a trend towards the incident. As can be seen, it only evaluates a few anomalies in the period up until the reported incident. There are also signs of abnormalities earlier in the production data as seen for the anomaly scorers. Interestingly, one can see that the samples evaluated as anomalies for the one class SVM are all among the highest scoring samples for the two anomaly scorers.

Table 5.5 shows how the scorers evaluate the validation data compared to the data from test case 1. As can be seen, there is a large difference between the minimum value for the validation and case data for the KDE scorer. There is also a difference between the mean values. This shows that the test case 1 data takes on patterns not seen in the validation data. The same trend is seen in the statistics for the LSTM RNN scorer. In both cases, the most extreme observation between training and production is approximately 100 times larger.

Figure 5.7 shows the number and percentage of anomalies for the OC SVM classifier for test case 1. As can be seen, the anomaly percentage of the anomalies detected in 2014 and 2016 is much lower than for the reported incident in 2017. This indicates that separating what could be false positives from true positives, could be solved by using an anomaly percentage threshold.

Figure 5.9 shows the anomaly scores for test case 2. Note that test case 1 and 2 are from different plants. There is no sample data from January 2017, but since dates are used as indexes, January is still seen in the plot. The data from 2016 is normal data from plant 2, and the artificial error is gradually increasing from February to April in

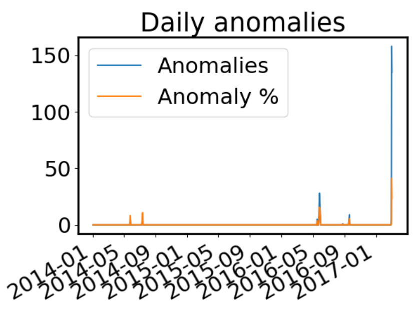

Figure 5.7: Daily classified anomalies for the OC SVM classifier on production data

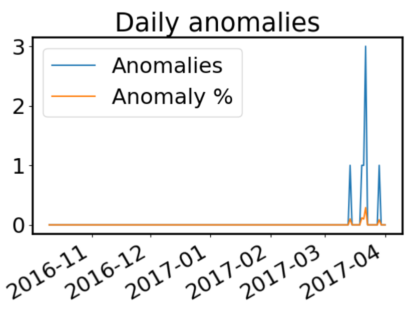

Figure 5.8: Daily classified anomalies for the OC SVM classifier on artificial data

2017. As can be seen, both the LSTM RNN and KDE scorers have growing anomaly scores from mid-February. Towards March, the anomaly scores are significantly higher than for the normal data. Looking at figure 5.6 one can see that the anomaly score for the artificial error is of the same magnitude as the score for test case 1 in the months before the incidents.

The OC SVM classifier is not picking up the trend as fast as, but it clearly detects that something is wrong towards the end of the sample period. This can be verified by looking at 5.8. Notice however how few samples that are classified as anomalous, and that the percentage is very low. In appendix C boundary plots for the SVM classifier and the KDE scorer and learning history plots for the LSTM RNN scorer can be found.

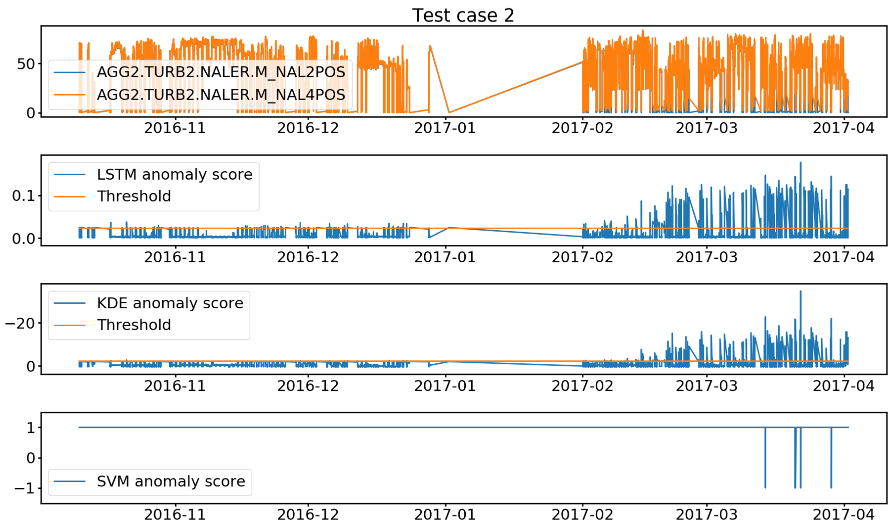

Figure 5.9: Anomaly results for the methods trained on training set 1 and evaluated on test case 2. For plot interpretation see Figure 5.5.

5.4.2 Training set 2, test case 1 and 2

Figure 5.10 shows training set 2, and how the different methods evaluate the training data. The validation data can be found in Figure D.3. As can be seen the anomaly scores for the KDE and LSTM RNN anomaly scorers are not as similar in shape as was seen for training set 1. They are also reduced in magnitude. The OC SVM evaluates all data points as normal.

Figure 5.11 shows the anomaly scores for test case 1, when training the methods on training set 2. Training set 1 and 2 are as mentioned approximately of the same size. The figure shows a very similar pattern for the two anomaly scorers to what was seen for training set 1. This could mean that both methods generalize well across plants, at least as long as the training sets are of similar size.

The biggest difference is seen in the OC SVM window, where some new samples are being classified as anomalies in 2015. This could show that OC SVM is more sensitive to the training data, as the data from 2015 seems to be the best data from this needle pair. However, looking at Figure 5.12 it becomes clear that the number of anomalies detected in 2015 is small, and that setting a threshold as mentioned in the previous section could classify them as false positives.

Table 5.6 shows the anomaly statistics for the validation set and test case 1. It is clear that both scorers have a higher minimum and maximum values than what was seen when the methods were trained on training set 1. They are, however, very similar in magnitude. The mean values for both scorers are also changed, which could indicate that the score is simply biased.

	KDE validation	KDE case 1	LSTM validation	LSTM case 1
min	-4.566	-296.370	2.336e-05	2.336e-05
max	1.291	1.2905	4.047e-2	0.498
mean	−0.518	-1.493	4.976e-04	8.116e-3

Table 5.6: Table comparing the statistics for the KDE and LSTM RNN anomaly detectors on the validation and production data

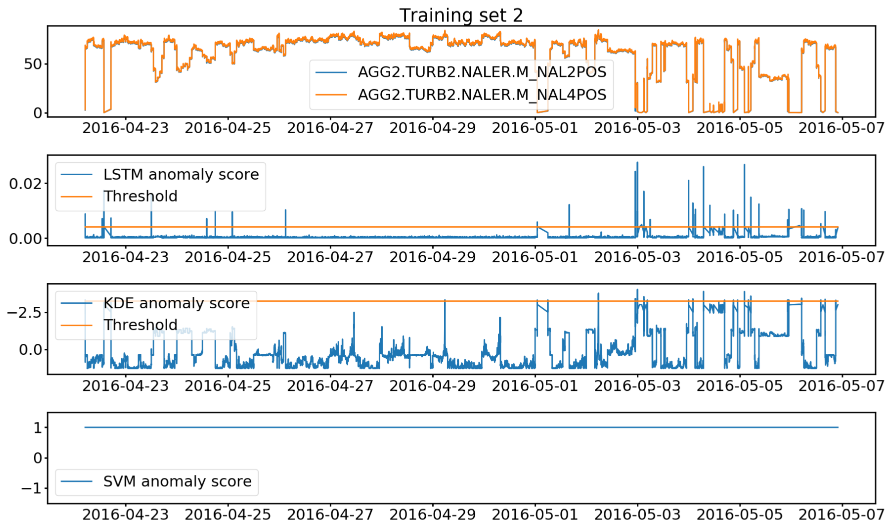

Figure 5.10: Anomaly results for the methods trained on training set 2 evaluated on training set 2. For plot interpretation see Figure 5.5.

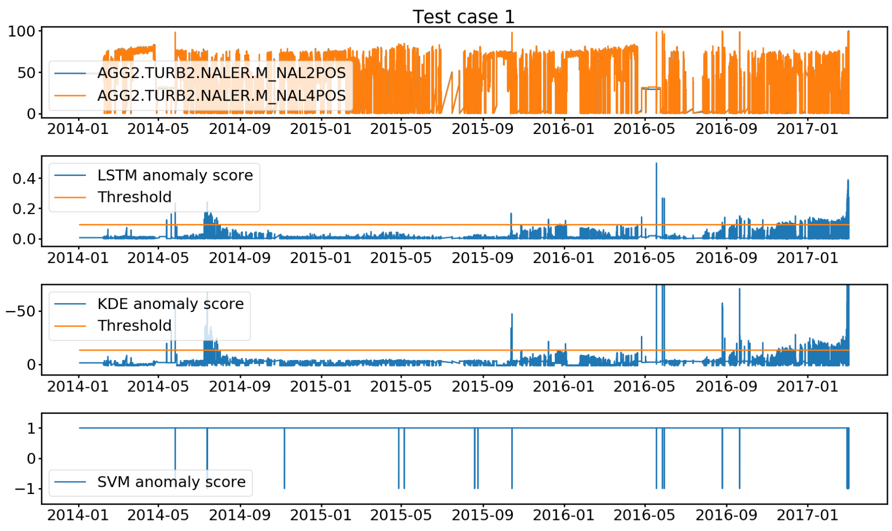

Figure 5.11: Anomaly results for the methods trained on training set 2 and evaluated on test case 1. For plot interpretation see Figure 5.5.

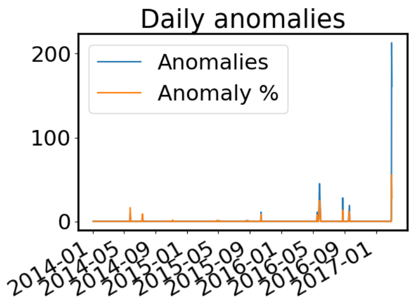

Figure 5.12: Anomaly statistics for OC SVM on production data

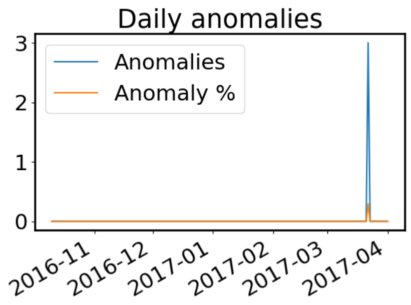

Figure 5.13: Anomaly statistics for OC SVM on artificial data

Figure 5.14 shows the anomaly scores for the artificial data in test case 2. As with test case 1, both scorers behave very similar to what was seen for training set

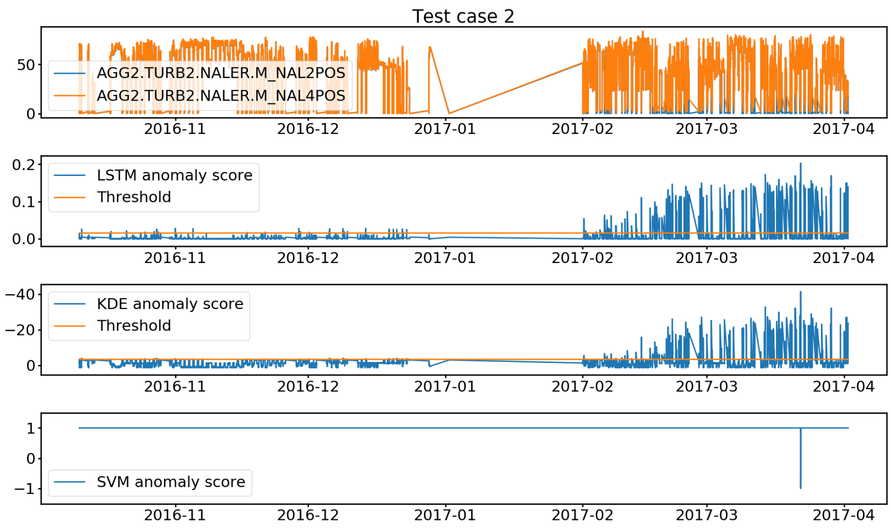

Figure 5.14: Anomaly results for the methods trained on training set 2 and evaluated on test case 2. For plot interpretation see Figure 5.5.

1. Looking at the LSTM RNN anomaly score in window two, one can verify that the anomaly score increases immediately after the error is applied. It can detect only small deviation from the normal operating pattern. The KDE scorer also seems to pick up the artificial error faster than for training set 1. As the artificial error is added to process data from the same needle pair that is used for the data in training set 2, it is sensible that it is possible to detect the anomalies earlier than for training set 1. Looking at figure 5.11 one can see that the anomaly score for the artificial error is of the same magnitude as the score for test case 1 in the months before the incidents.

The OC SVM detector is, however, having problems detecting the anomalies, and only detects the sample that the two other detectors identify as the most anomalous. One possible explanation to this is that the parameters found in the hyperparameterization for the OC SVM do not generalize very well to new data, as the hyperparameterization was performed on data from plant 1. Figure 5.13 shows that the classifier is only classifying three samples as anomalous.

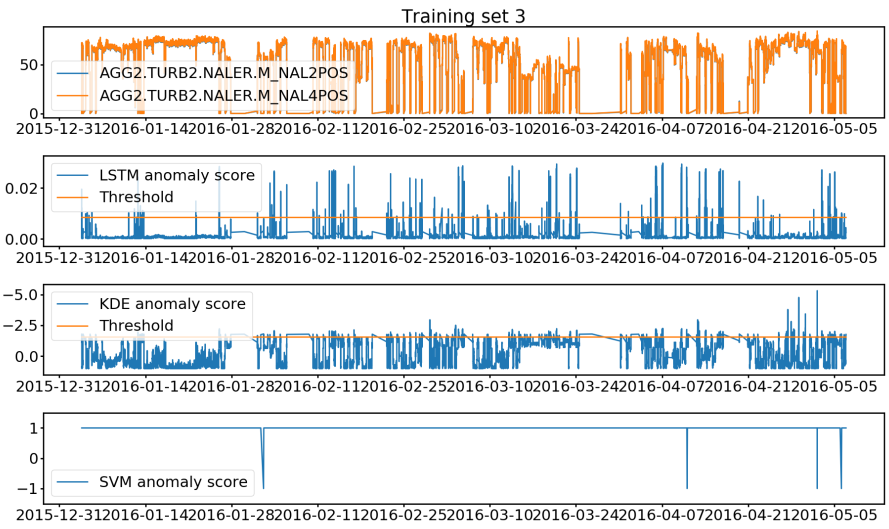

Figure 5.15: Anomaly results for the methods trained on training set 3 evaluated on training set 3. For plot interpretation see Figure 5.5.

5.4.3 Training set 3, test case 1 and 2

Figure 5.15 shows training set 3, and how the different methods evaluate the training data. The validation data can be found in Figure E.4. The magnitude of the anomaly scores for the two scorers is between the scores for training set 1 and 2. The OC SVM evaluates the most anomalies of all the training sets. As this training set is 10 times larger than the two other, this further strengthens the claim that the OC SVM hyperparameters are sensitive to the training data.

Figure 5.16 shows the anomaly scores and classifications for test case 1 when trained on training set 3. The scorers evaluate anomalies very similar to what was seen for training set 1 and 2. Both scorers detect a rising trend of abnormal data towards the reported incident in March 2017. The anomaly scores for the data from the day of the incident is very high for both scorers, as was seen in the two previous cases as well. The OC SVM classifier is classifying test case 1 very similar to what was seen for

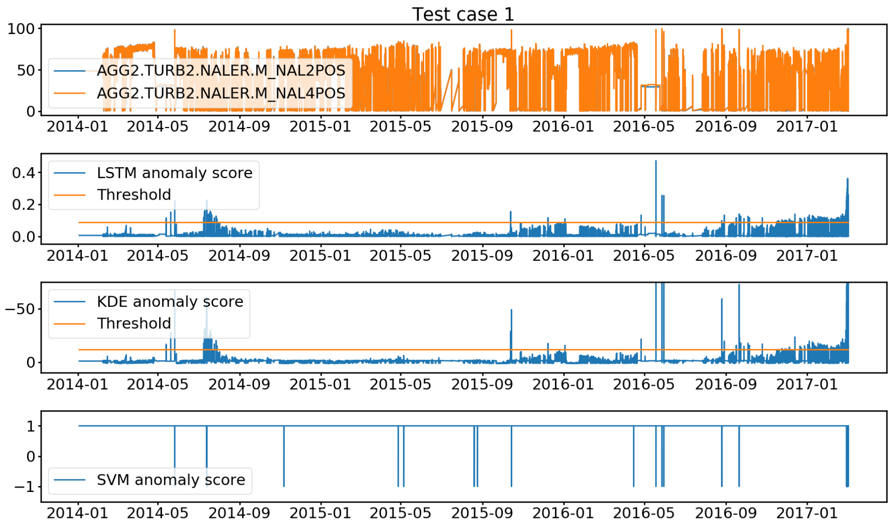

Figure 5.16: Anomaly results for the methods trained on training set 3 and evaluated on test case 1. For plot interpretation see Figure 5.5.

training set 2, as is verified by Figure 5.17. Adding more data from plant 2 did not fix the issue with what appears to be false positives in the data sampled in 2015.

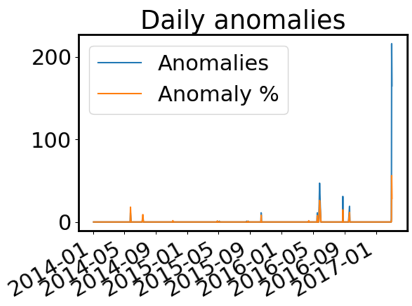

Figure 5.17: Anomaly statistics for OC SVM on production data

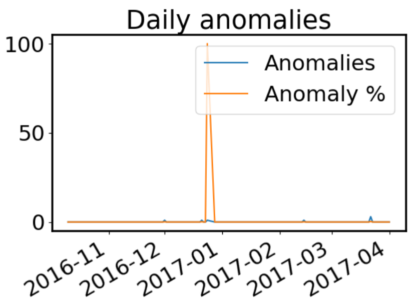

Figure 5.18: Anomaly statistics for OC SVM on artificial data

	KDE validation	KDE case 1	LSTM validation	LSTM case 1
min	-2.755	-277.804	2.559e-05	2.796e-05
max	0.999	0.9995	3.597e-2	0.471
mean	-0.366	-1.167	6.709e-04	7.651e-03

Table 5.7: Table comparing the statistics for the KDE and LSTM RNN anomaly detectors on the validation and production data

Table 5.7 show the statistics for the LSTM RNN and KDE detectors, the statistics are similar to what was seen in the previous training sets. Early stopping stopped the LSTM RNN scorer after only 20 epochs, where the two previous sets ran for 134. When running the LSTM RNN for 134 epochs, the performance of the LSTM RNN scorer was much worse than for the previous training sets. This could indicate that training set 2 holds the same information as training set 3, but as training set 3 has more samples, the algorithm does not need to iterate over the entire set as many times. Early stopping is used to avoid overfitting on the training data, and the poor performance when running for 134 epochs on set 3, is most likely due to overfitting. The fact that the two other methods appear to evaluate test case 1 similar for both training set 2 and 3, further strengthens this claim. If the added data in set 3 has a different distribution than the one in training set 2, this would have been caught by KDE and one class SVM.

Figure 5.19 shows the anomaly scores for test case 2. As for test case 1, one can verify that the scorers behave more or less equal to what was seen for training set 2. The OC SVM classifier is detecting anomalies in the normal test data, and it is only able to classify data from one day during the artificial error as anomalous. Figure 5.18 shows that for a day in the end of December, all samples are evaluated as anomalies. This is most likely due to only one sample being stored right after midnight. The fact that training set 3 yields the poorest performance for the OC SVM, further strengthens the assumption that the algorithm is sensitive to the combination of hyperparameters and training data. In appendix E boundary plots for the SVM classifier and the KDE scorer and learning history plots for the LSTM RNN scorer can be found.

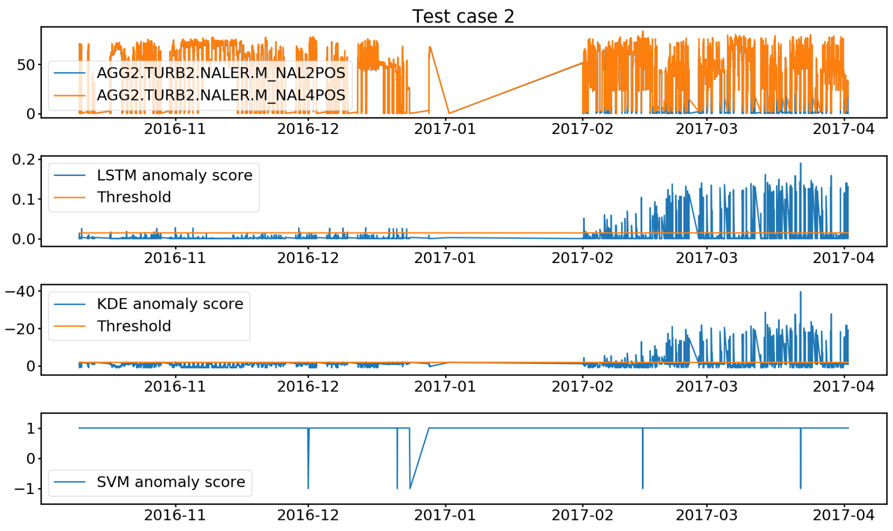

Figure 5.19: Anomaly results for the methods trained on training set 3 and evaluated on test case 2. For plot interpretation see Figure 5.5.

5.4.4 Comparing the methods for all training sets, on test case 1

To simplify comparing the performance of the different methods new plots are created that shows how the different methods evaluate the production data in test case 1, for all three training sets. Figure 5.20 show how test case 1 is evaluated by the three LSTM RNN scorers. It is clear that all the scorers perform very similar. This shows that the method works well for both large and smaller training sets. All scores indicate a growing anomaly trend towards the incident in 2017, but the scorers for training set 2 and 3 show an increasing anomaly trend earlier than the scorer for set 1. A theory for why this is happening is that the training data used in set 1 is sampled right after maintenance, while the training sets from plant 2 are sampled after years of operation. This means that minor deviations between the needles that could be normal after some time of use is seen as anomalous for the scorer trained on set 1 while this is seen as

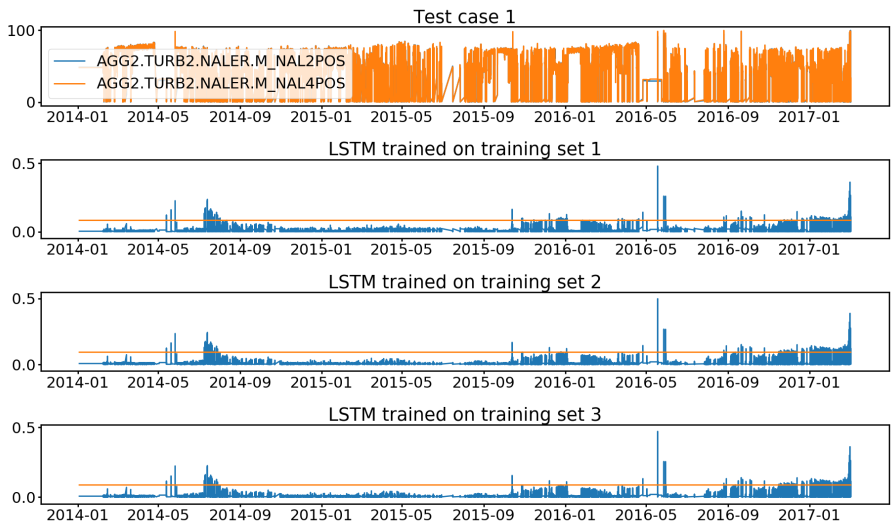

Figure 5.20: The anomaly score for the three LSTM RNN scorers tested on test case 1, is seen in the three lower subplots. The upper subplot shows the needle positions on top of each other. The orange line indicates the 0.1% most extreme scores.

normal when set 2 and 3 are used for training. However, deploying any of the three scorers at the plant could have warned about an increase in anomalous observations, that could have identified the anomalous trend before it is becoming too extreme.

Figure 5.21 shows how the different KDE scorers compare. All three scorers evaluate test case 1 very similar. The scorer from training set 2 gives the highest anomaly scores for the data leading up to the incident. As with the LSTM RNN scorers, set 2 and set 3 appears to perform slightly better than set 1. All three scorers are able to detect the growing number of abnormal data seen towards March 2017. Notice that the score is limited to -75 to enable the reader to see the smaller trends, as the extreme values go below -250, making interpreting the plot very hard.

Finally, Figure 5.22 show the three different OC SVM classifiers. As with the two previous methods, there is little difference between set 2 and 3. This is the only method that appears to perform best when trained on data from plant 1. However, there is

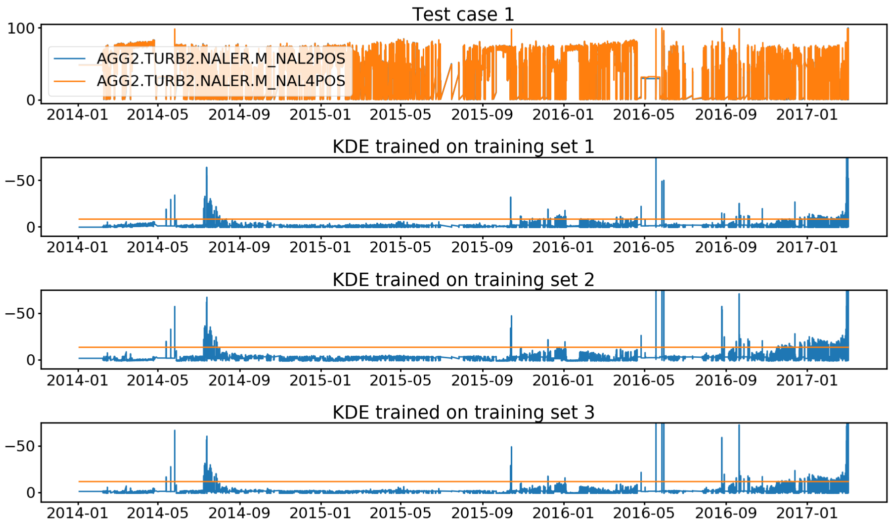

Figure 5.21: The anomaly score for the three KDE scorers tested on test case 1, for plot interpretation see Figure 5.20

very little difference in the classifications, and when using the percentage of daily anomalies, one can easily separate the data from the incident with the other. This method is harder to spot trends with, and with the current parameterization, it is hard to find a way this method can be used as an early warning system for the incident seen in March 2017. This makes sense, as this method predicts all data as either normal or anomalous, and hence include an extra step compared to KDE and LSTM RNN, which gives a score that the user need to interpret. Hence using this method would require less of the user, but more tuning and testing to ensure correct behavior.

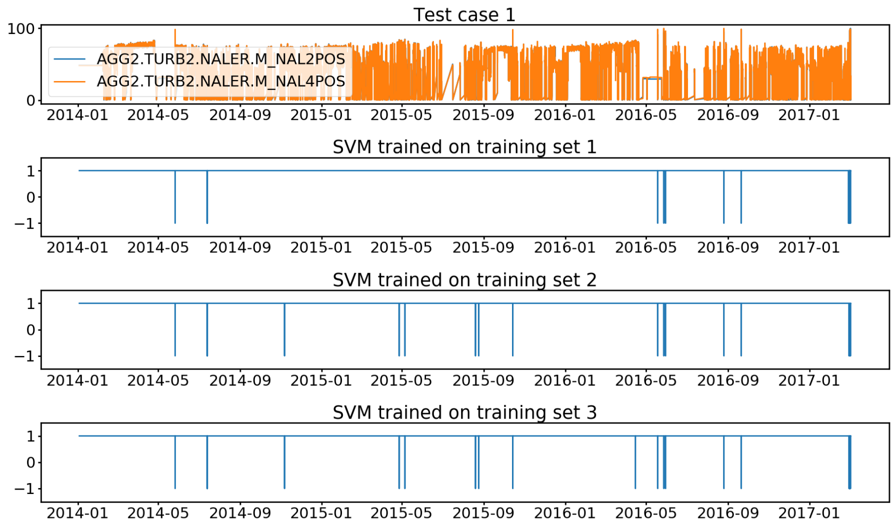

Figure 5.22: The anomaly classification for the three OC SVM classifiers tested on test case 1, for plot interpretation see Figure 5.20

5.4.5 Test case 3 and 4

As the production data from test case 1 only contained one reported anomaly, the methods are tested on two start failures in test case 3 and 4. The results for the methods trained on training set 2 is seen in Figure 5.23 and 5.24. In Figure 5.23 one can see that the reported start failure from 10.12.2014 is given a high anomaly score for both scorers. The OC SVM classifier is not able to detect the start failure, and also appears to classify data from November as an anomaly wrongly. Notice that the magnitude of the scores is lower than what was seen for the most extreme samples from test case 1, it is, however, very similar to the magnitude seen for the artificial error. In Figure 5.24, the start failure from case 4 is analyzed. The scorers evaluate the start failure to be more anomalous than the start failure in case 3, as the magnitude of the scores are doubled. This is reasonable when looking at Figure 4.7 and 4.8, where the scatterplot for turbine 1 clearly deviates more than the scatterplot for turbine 2. The OC SVM

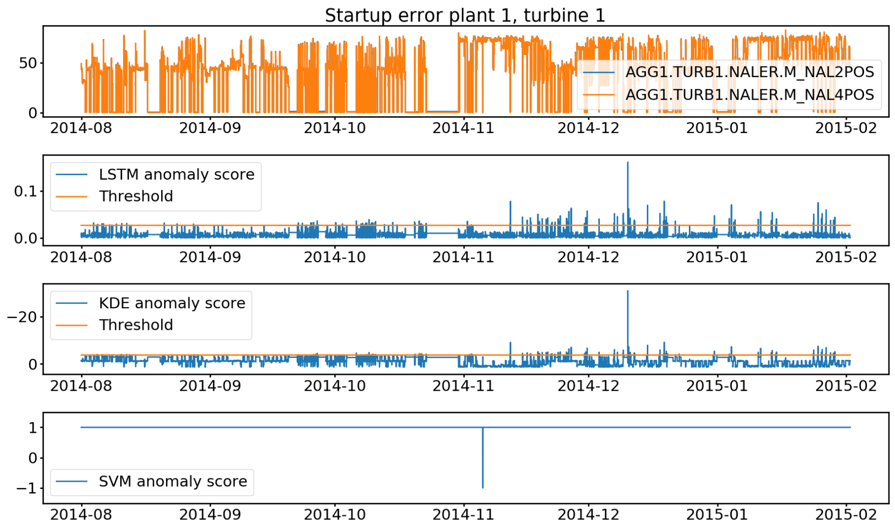

Figure 5.23: Anomaly results for the methods trained on training set 2 and evaluated on test case 3. How to interpret the figure can be seen in Figure 5.5.

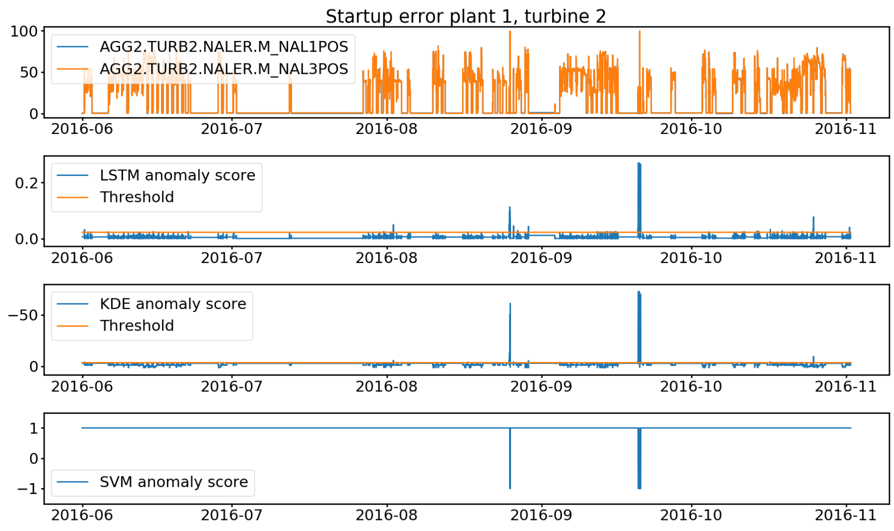

Figure 5.24: Anomaly results for the methods trained on training set 2 and evaluated on test case 4. How to interpret the figure can be seen in Figure 5.5.

classifier is also able to detect the start failure. The figures for the two other training sets can be found in appendix F. They are not included in this section, due to very similar performance as the one seen for training set 2.

Chapter 6

Discussion

6.1 Analyzing the data and building a case

Before the data analysis could start, the data needed to be reconstructed. As mentioned the data was far from ready for analysis as it was received. A Python library was created that automatically reads the files received by the energy company, and produces data with a structure that allows analysis. The library is created in a way that allows for easy use of new data in the same format. This shows one of several challenges when working with real-world data from the industry.

Several issues arose once starting to work with the data. A problem which most likely is relevant to most industries today is that having data is far from a guarantee of having good data. At first glance, the 90Gb of available data seemed like more than enough to find many interesting cases. However, as the work progressed, it became apparent that finding good cases for testing anomaly detection, was not straightforward. It became clear that very few variables were sampled simultaneously, at a higher sampling rate than once every hour. Due to this, much time was spent trying to find a case that did not need a higher sampling rate. Going through the plant logs, no good cases were found. It might be possible that the energy company is sampling data in a scheme similar to the one used in Selak et al. (2014), but this is very hard to reverse engineer from the data. Even if one chooses to sample data in such a scheme, it would be beneficial to sample more data at a higher and constant frequency, as it could open up for using techniques such as time-series forecasting.

Going through the logs from the power plants, the Pelton needle case was quickly picked out as interesting. As Åsnes (2017) focused on the guide vanes of Francis turbines, building a case around the Pelton needles would serve as a natural extension, where one of the methods used for the Francis case could be tested on a new case. The needles play a vital part in the power production, and increasing knowledge about their condition and the operational trend would be a great addition to the existing instrumentation. Having four different Pelton turbines from two different plants in the dataset would allow testing how much adaptation would be needed to make the different anomaly detection techniques work on new plants. Having turbines with no reported incidents also made it possible to verify if the techniques were prone to give false positives.

It would have been interesting to include more than just the needle variables in the analysis, but this was not possible due to the data. The methods used in the analysis can handle more variables. Hence the analysis can be extended if the data is available. The possibility of using interpolation to handle the missing data was also addressed, but it was quickly discarded as it would require a lot of plant and process insight to pick techniques that interpolated every process variable in a reasonable way.

As there only was one reported incident with the needles during operation, it was decided to create an artificial error that replicates the needle pattern seen for the days building up to the incident. Having the start date of the artificial error would give an insight into how quickly the anomaly detection techniques can catch such a trend. It could also give an indication of when one could expect that the original error started.

6.2 Choosing the methods

Several different anomaly detection methods and techniques were considered. As the Pelton case only had one reported incident, it made making a labeled data set hard. This opted for using methods that did not need labeled data, as one does not have to collect data from all the known failure modes, only from normal operation. As mentioned in the introduction, this opens the possibility also to detect unknown failure modes. This means that methods for labeled and unlabeled data does not exclude one another, and for a complete condition monitoring system using both approaches might yield the best solution. Three methods were chosen for the anomaly analysis. OC SVM was chosen because it showed promise in the case with the Francis turbines guide vanes. KDE was picked because it is a relatively simple method, which is fast to implement through Scikit-learn. The LSTM RNN algorithm was picked because it has shown promise when working with time series, and would serve well for comparison for the less complex other methods. Where OC SVM is a classifier, KDE and LSTM RNN are regressors. Hence the two latter does not classify the data as normal or abnormal. They produce a score that mirrors how anomalous the data is evaluated to be.

6.3 Hyperparameterization

The optimal set of hyperparameters were searched for through a grid search for the three methods. The parameters were found using data from Plant 1, and the same parameters were used for all the different training sets. This was done to see how well the methods would transfer between plants. If one needs to search for the optimal hyperparameters for each plant, training the methods will take much more time. KDE and LSTM RNN was found to adapt well to data from new plants. A more extensive search for the optimal set of hyperparameters could have been performed, especially for OC SVM and LSTM RNN, but as the goal of this thesis was to investigate possible anomaly detection methods, the parameterizations found yielded good enough results to give an indication on what the methods can be used for.

The custom cost function designed for the OC SVM seemed to work as intended, but can most likely be improved. As one gets more knowledge about how the data is structured, using different kernels than the Gaussian kernel can also help with the performance of the algorithm.

6.4 Training sets and test cases

Many different approaches could have been taken with the training sets and test cases. The main motivation for the chosen setup was that it enabled to compare performance across plants and training sizes. Many more cases could have been created, but as mentioned earlier this is an initial study, that can be used as starting ground for further testing. It was clear that the sampling rate was very different between the plants, as 14 days of sampling on plant 2 gave as many data points as almost five months at plant 1. This shows the importance of investigating the data. It was also important to ensure that the 14 days from plant 2 covered all operational modes for the plant. The final setup was three different training sets and four different test cases. The three training sets made it possible to evaluate how well the methods adapted to data from different plants and of different sizes. As it was chosen to use the optimal hyperparameterization found for data from plant 1, it also gave insight into how general the parameters were. The artificial test case made it possible to evaluate the performance of the production data seen in test case 1. The two start failure cases made it possible to evaluate the performance of more than one reported incident.

6.5 Performance of the methods

All methods showed promise in detecting the reported incident from Plant 1 found in test case 1. OC SVM performed best when trained on training set 1. This indicates that the method is more sensitive to the chosen hyperparameters and that the optimal hyperparameters change when the training set changes. It also became clear that the OC SVM did not catch the pattern building up to the reported incident in test case 1, and that it seems like a method more suited for shutdown alerts than for early warning. As the method is trained without labeled data, it is hard to find the optimal selection of hyperparameters. The custom cost function created tried to deal with this issue, but as can be seen from the performance, there could be a set of hyperparameters that would increase the performance of the OC SVM. When the method was trained on data from plant 2, it showed signs of giving false positives. This further strengthens the claim that the optimal hyperparameters change when the training set is changed.

KDE and LSTM RNN performed very similarly for all training sets on all test cases. This indicates that the two methods are more robust to changes in the training set. It is, however, important to notice that the LSTM RNN needed early stopping to yield a similar performance from training set 3. Complex NNs always run the risk of overfitting to the data. Both methods performed best when trained on training set 2. As mentioned in the analysis this can be because training set 2 is taken from plant 2 after years of operation, and it might then better represent normal needle deviation, not seen in the training set from plant 1 sampled right after maintenance. In addition, set 2 is approximately of the same size as set 1 which the hyperparameterization is performed on, where as set 3 is 10 times larger. This shows that even if KDE and LSTM RNN seem more robust to changing the training sets without changing the hyperparameterization, this still is a concern one needs to keep in mind. Both methods clearly evaluated the data from the reported incident in test case 1 as anomalous, and one could identify a growing trend in the anomaly score towards the incident.

The artificial error in test case 2 showed that both KDE and LSTM RNN could detect an error as the one seen in plant 1 in its early stages. For training set 2, the LSTM RNN showed an increase in the anomaly score immediately after the artificial error was added to the data. The KDE also found this pattern but did not detect it as fast. This means that it is likely that the pattern seen for the anomaly scores for test case 1 towards the reported incident, can be used as an early warning system for system degradation. One can also claim that it can be detected by a pre-trained method, as long as the same transformation is used for both training and test data.

Training case 3 and 4 showed that KDE and LSTM RNN clearly are able to separate anomalies from normal data. The two start failures only generated a few anomalous data samples, but the methods gave those samples a much higher anomaly score than the normal data surrounding the incidents. OC SVM was only able to detect one of the start failures. This was also given the highest score by the scorers. The two start failure cases show that the methods evaluate the data correctly, and strengthen the hypothesis that the pattern seen leading up to the incident in test case 1, is due to system degradation.

The LSTM RNN is the most complex method, and several different network structures could have been included. If data had been sampled at an even frequency, it could have benefited more from its capability to remember previous calculations. Dropout could have been included to reduce the risk of overfitting, and this could have given better performance for the more complex network structures. However, the structure used in this analysis shows that the method is capable of detecting minor changes in the normal operation pattern and that it can be used for early warning about system degradation.

It is important to remark that the same transformation was used for all data sets, regardless of plant origin. This is crucial to ensure that the data is transformed in the same way, if not one could not have trusted the results of the methods on data from different plants. It could have been beneficial to have more classes in the SVM. This could either be solved by getting labeled data and train the regular SVM. Another possible approach is to train several different OC SVM classifiers, with increasing complex decision boundary created by the separating hyperplane. Then the one can compare how the different classifiers evaluate the data and make it easier to spot trends earlier.

Overall KDE seems like the best choice for anomaly detection as presented in this thesis. As mentioned, LSTM RNN did show slightly better performance for some of the cases, but KDE is a more straightforward method with much fewer hyperparameters. This makes it faster to optimize, and from the results, it appears to be very robust to changes in both data size and source. The results clearly show that deploying a KDE anomaly scorer at a plant with a Pelton turbine will increase the knowledge about the condition of its needles. Merely providing a graph with the anomaly score history will give operators a good indication of the condition of the needles. The primary challenge is to define a threshold for what is an anomaly and what is normal. In the results, a line evaluating 0.1% of the data as anomalous was used. To apply this at a plant one would either need to determine a reasonable threshold through testing in cooperation with system experts or present the score to the operator, making them responsible for the evaluation. How fast the anomaly score changes should also be evaluated, as this can be an indication of how long the plant can normally be operated.

Chapter 7

Conclusion and future work

7.1 Conclusion

It this thesis anomaly detection for hydroelectric power plants have been investigated, attempting to detect anomalies related to the operation of the needles of Pelton turbines. Three different methods are used for the anomaly detection, one class support vector machines, kernel density estimation and long short term memory neural networks. The first is a classifier and the two latter are regressors. The methods are trained on three different training sets, and evaluated on four different test cases. All methods showed promise as potential anomaly detectors, where kernel density estimation and long short term memory recurrent neural network performed best. Kernel density estimation is found superior, as it performs almost as well as the neural network which is a much more complicated method. Kernel density estimation has fewer hyperparameters, making it both faster and more straightforward to optimize. In addition it was shown to be the method that generalized best to new data. One class support vector machine was found to have hyperparameters that are very dependent on the training data, and hence the hyperparameters needs to be tuned for every new training set. This makes the method more cumbersome to use than the two others. Kernel density estimation performed very well with the same parameterization for all training sets and test cases. The neural network also performed well, as long as early stopping was enabled. Early stopping avoids overfitting on the training data. The techniques showed promise when testing a pre-trained anomaly detector at data from a new plant. As long as the data transformation used during training was used on data from the new plant, all methods showed that they were able to detect anomalies.

The challenges of working with real industrial data became apparent during this thesis. The data was found to be sampled aperiodically and often with only one process variable at a time. This made building a case for anomaly detection more difficult than expected, and it resulted in only being able to include two process variables in the analysis. Ideally, many more variables would have been sampled, so that more plant information could have been included in the analysis. This could have created a more robust anomaly detection system, and the feature selection and dimensionality reduction techniques introduced in theory could have been tested.

The two regressors output continuous values and do not classify the data as normal or anomalous. A numeric value evaluating the 0.1% percent most extreme data was used as a threshold in the analysis. Before the two methods can be used as anomaly detectors a new way to set the threshold must be found. It is suggested that using both a threshold for magnitude and rate of change to give plant operators more information about the condition of the needles, but no exact solution is presented.

The results showed that system degradation could be detected without having data representing system degradation during training. As sampling data from system failures is tough to come by, this problem is often solved by creating artificial data that represents known failure modes. This does, however, introduce the risk of not detecting unknown failures, and is a non-trivial task that requires a lot of system expertise. By using methods that only requires data from normal system operation, this problem is avoided, and all data used during training can be sampled at the plant. Also, it has been shown that anomaly detectors can be trained on data from one plant, and correctly analyze data from another.

7.2 Future work

A natural next step would be to investigate how to find reasonable thresholds for the regressors. Having this in place would then make the methods ready for testing at a plant. More effort can also be spent on finding the optimal hyperparameters for the neural network, as there might exist parameterizations that yield better performance. Another natural next step would be to collect more data in a way that allows one to perform a similar analysis with more process variables from the plant.

How to best present the new information to plant operators should also be investigated. It is vital that the system is easy to interpret, and that the operators trust it. The work from this thesis and the work from Åsnes (2017) will be a major part of a paper written for HYDRO 2018 Progress through Partnerships that will be presented in Gdansk, Poland 15 - 17 October 2018.

Appendix A

SVM gridsearch

All decision boundary plots for the five best and five worst SVM configurations are found below, sorted by score. As can be seen, all five highest scoring configurations produce a very similar decision boundary.

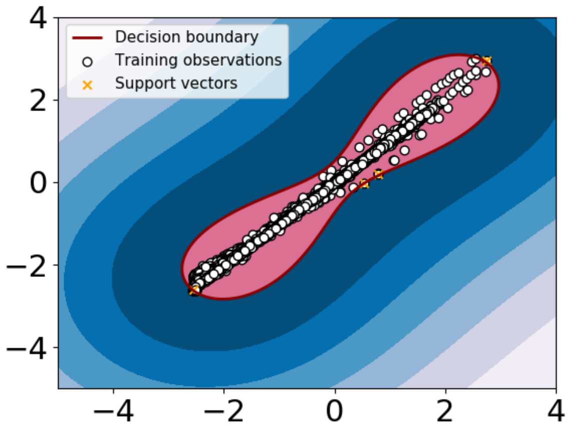

Figure A.1: Decision boundary for OC SVM with parameters $\gamma = 0.05$, $\nu = 1/N$. Score = 0.975. How to interpret the figure is found in Figure 5.1

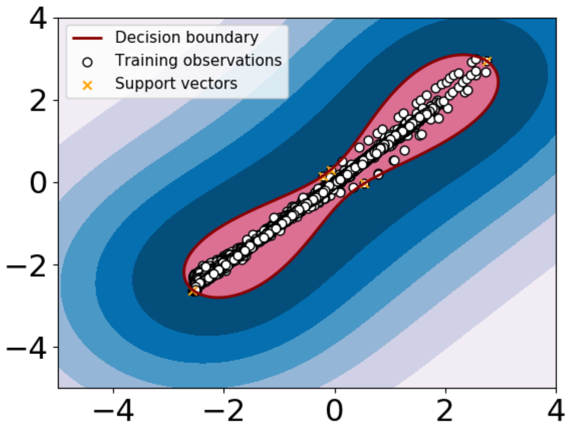

Figure A.2: Decision boundary for OC SVM with parameters $\gamma = 0.1$, $\nu = 1/2N$. Score = 0.970. How to interpret the figure is found in Figure 5.1

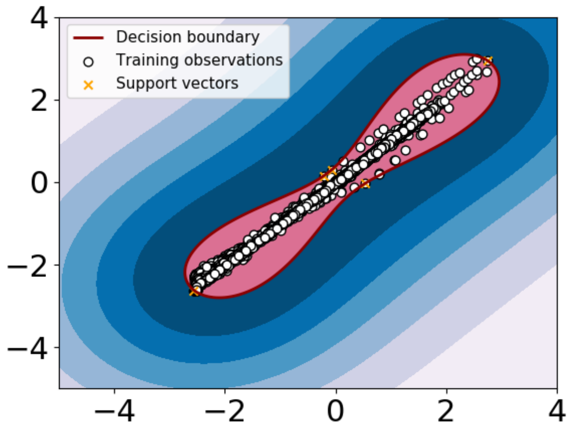

Figure A.3: Decision boundary for OC SVM with parameters $\gamma = 0.1$, $\nu = 1/N$. Score = 0.961.How to interpret the figure is found in Figure 5.1

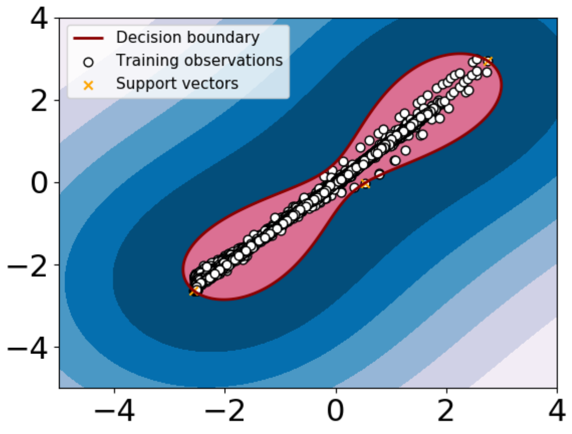

Figure A.4: Decision boundary for OC SVM with parameters $\gamma = 0.05$, $\nu = 2/N$. Score = 0.960. How to interpret the figure is found in Figure 5.1

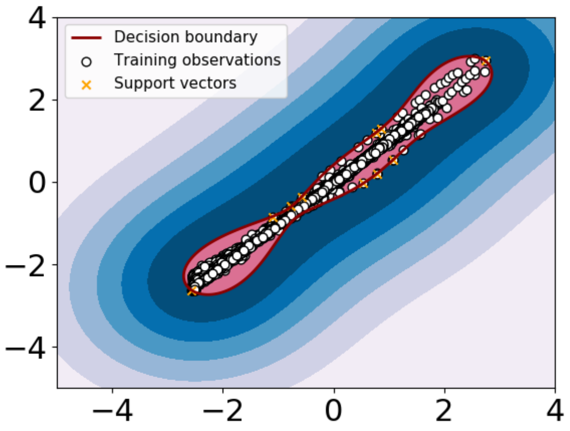

Figure A.5: Decision boundary for OC SVM with parameters $\gamma = 0.2$, $\nu = 1/N$. Score $= 0.957$. How to interpret the figure is found in Figure 5.1

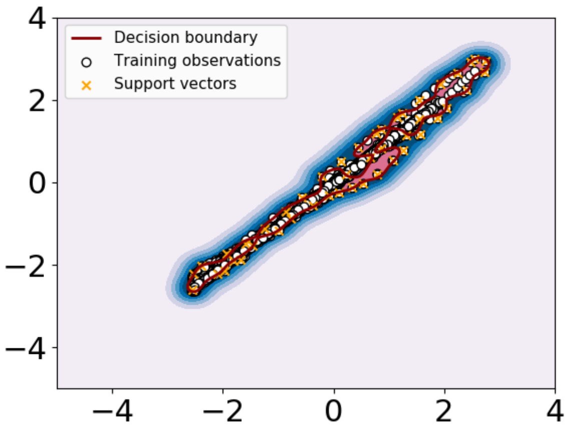

Figure A.6: Decision boundary for OC SVM with parameters $\gamma = 8$, $\nu = 1/10N5$. Score $= -44.437$. How to interpret the figure is found in Figure 5.1

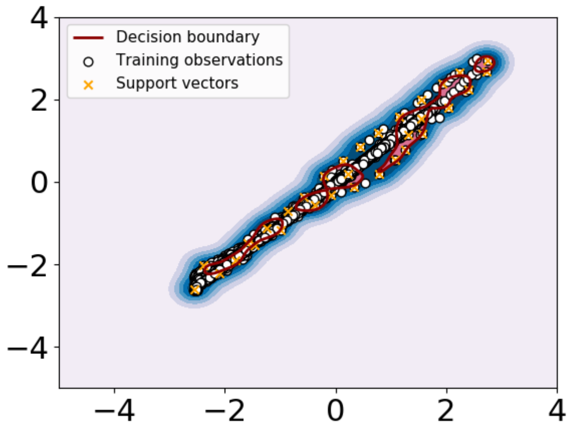

Figure A.7: Decision boundary for OC SVM with parameters $\gamma = 8$, $\nu = 1/25N$. Score = −51.780. How to interpret the figure is found in Figure 5.1

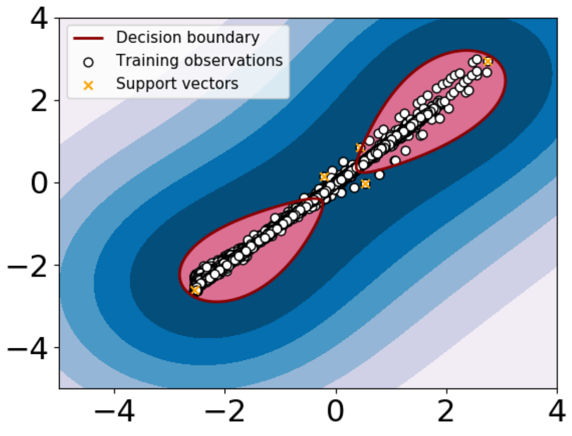

Figure A.8: Decision boundary for OC SVM with parameters $\gamma = 0.1$, $\nu = 1/25N$. Score = −57.461. How to interpret the figure is found in Figure 5.1

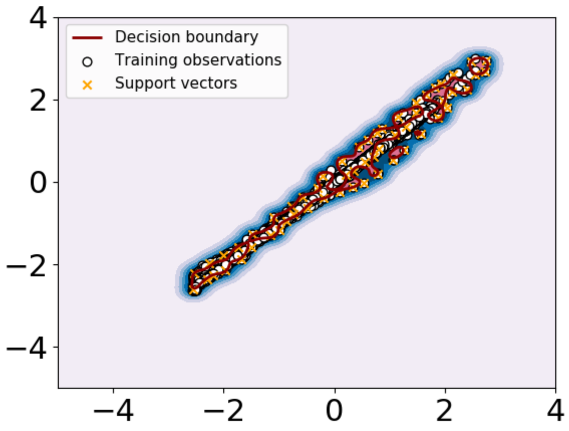

Figure A.9: Decision boundary for OC SVM with parameters $\gamma = 16$, $\nu = 1/10N$. Score $= -71.082$. How to interpret the figure is found in Figure 5.1

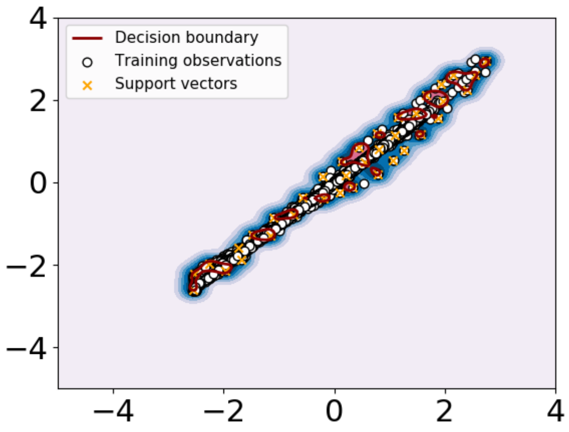

Figure A.10: Decision boundary for OC SVM with parameters $\gamma = 16$, $\nu = 1/25N$. Score $= -506.263$. How to interpret the figure is found in Figure 5.1

Appendix B

LSTM gridsearch

The model history for the training of the five best and five worst LSTM configurations are plotted below, from best to worst. As can be seen the worst configurations are stopped after only a few epochs due to lack of improvement of the predictions of the network.

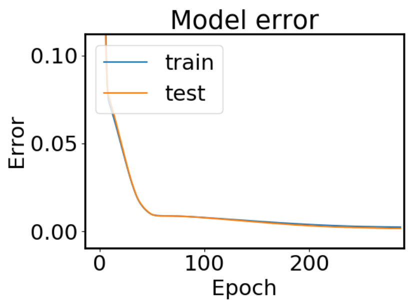

Figure B.1: Training history for the second best LSTM configuration. Test and training prediction error is shown as a function of epochs.

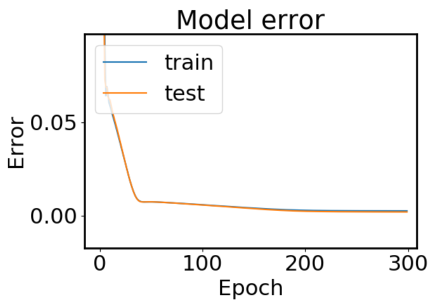

Figure B.2: Training history for the third best LSTM configuration. Test and training prediction error is shown as a function of epochs.

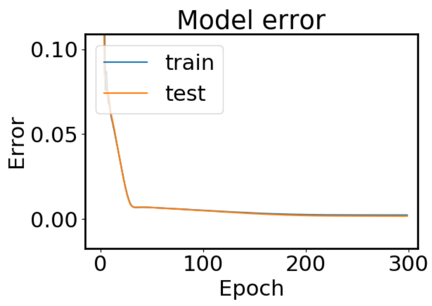

Figure B.3: Training history for the fourth best LSTM configuration. Test and training prediction error is shown as a function of epochs.

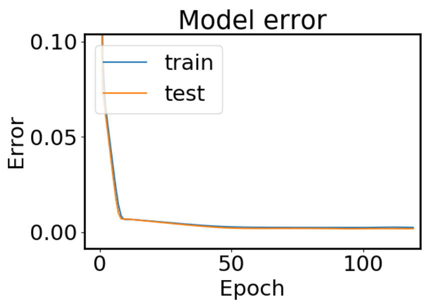

Figure B.4: Training history for the fifth best LSTM configuration. Test and training prediction error is shown as a function of epochs.

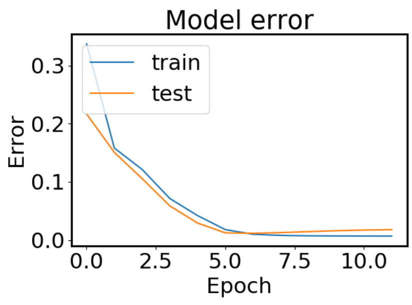

Figure B.5: Training history for the fifth worst LSTM configuration. Test and training prediction error is shown as a function of epochs.

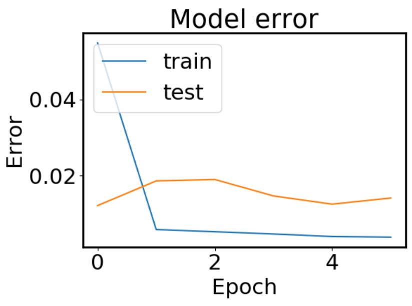

Figure B.6: Training history for the fourth worst LSTM configuration. Test and training prediction error is shown as a function of epochs.

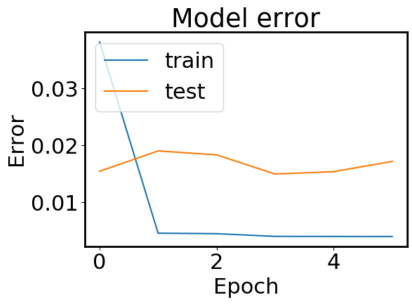

Figure B.7: Training history for the third worst LSTM configuration. Test and training prediction error is shown as a function of epochs.

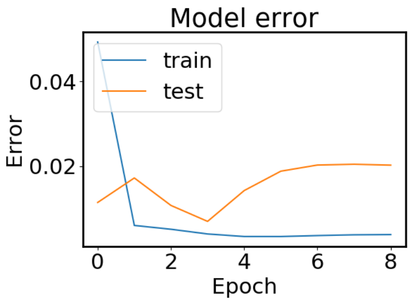

Figure B.8: Training history for the second worst LSTM configuration. Test and training prediction error is shown as a function of epochs.

Appendix C

Training case 1

Additional plots from the results of training case 1 is presented below. The plots are discussed in their caption.

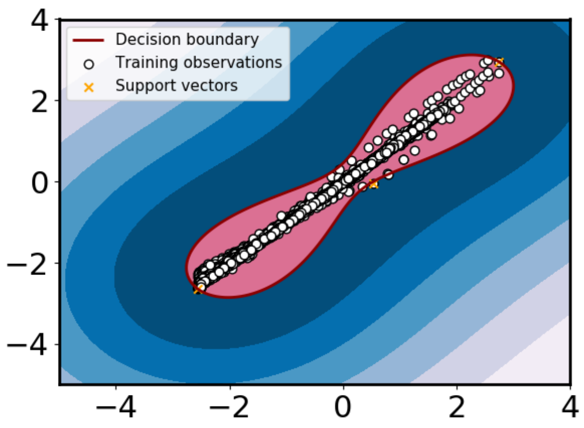

Figure C.1: Decision boundary for the OC SVM classifier. Support vectors and training observations are plotted. All samples located outside the red boundary are classified as anomalies. The color indicates the distance to the separating hyperplane.

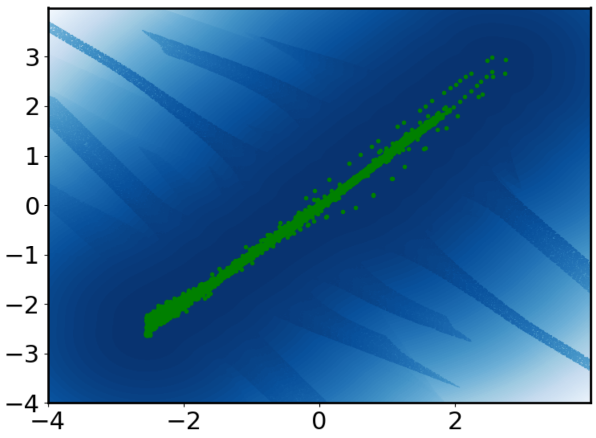

Figure C.2: The KDE anomaly scorers probability score plotted as contours in the space of the scaled training data. Training samples are seen in green. As the grid value becomes more and more anomalous the color becomes lighter.

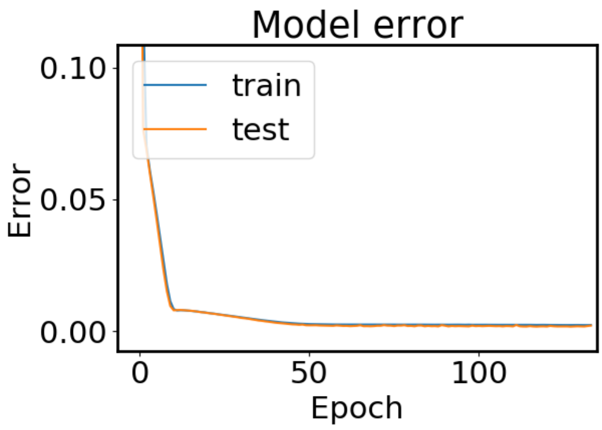

Figure C.3: Model error for the LTSM network as the number of epochs increase.

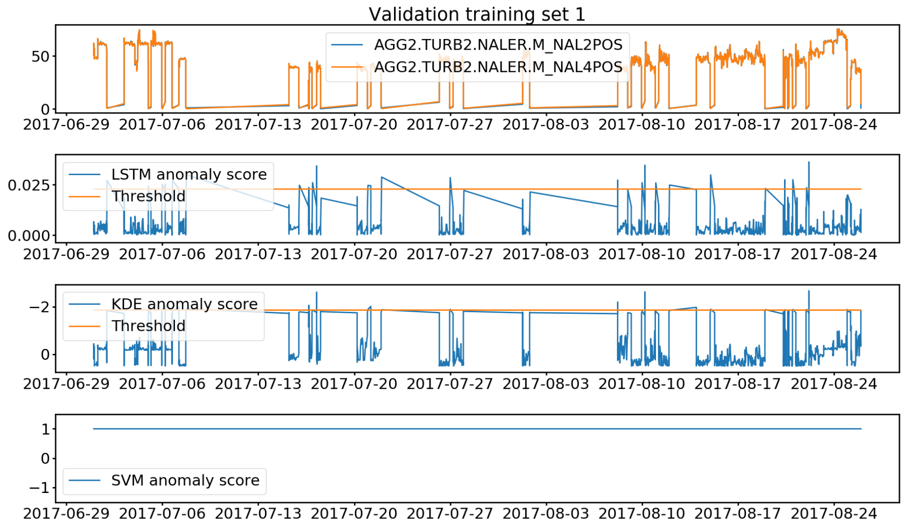

Figure C.4: Anomaly results for the methods trained on training set 1 evaluated on validation set 1. How to interpret the figure can be seen in Figure 5.5. As can be seen the anomaly scores for the KDE and LSTM anomaly scorers are much lower than for case 1 and 2 seen in the results, indicating that the scorers correctly evaluates the validation set as normal. The OC SVM classifier correctly classifies all samples as normal.

Appendix D

Training case 2

Additional plots from the results of training set 2 are presented below. The plots are discussed in their caption.

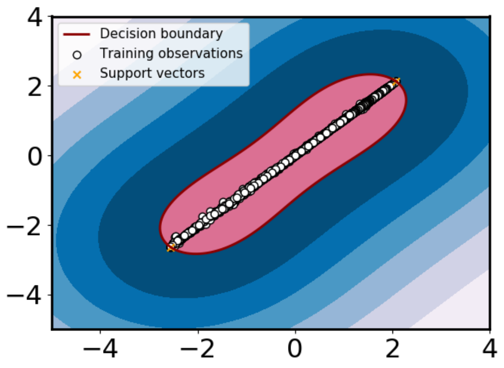

Figure D.1: Decision boundary for the one class SVM classifier. Support vectors and training observations are plotted. All samples located outside the red boundary are classified as anomalies. The decision boundary is smoother than for training set 1, but it has very similar shape. The color indicates the distance to the separating hyperplane.

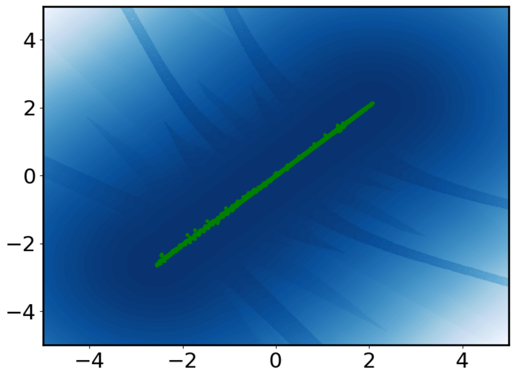

Figure D.2: The KDE anomaly scorers probability score plotted as contours in the space of the scaled training data. Training samples are seen in green. As the grid value becomes more and more anomalous the color becomes lighter.

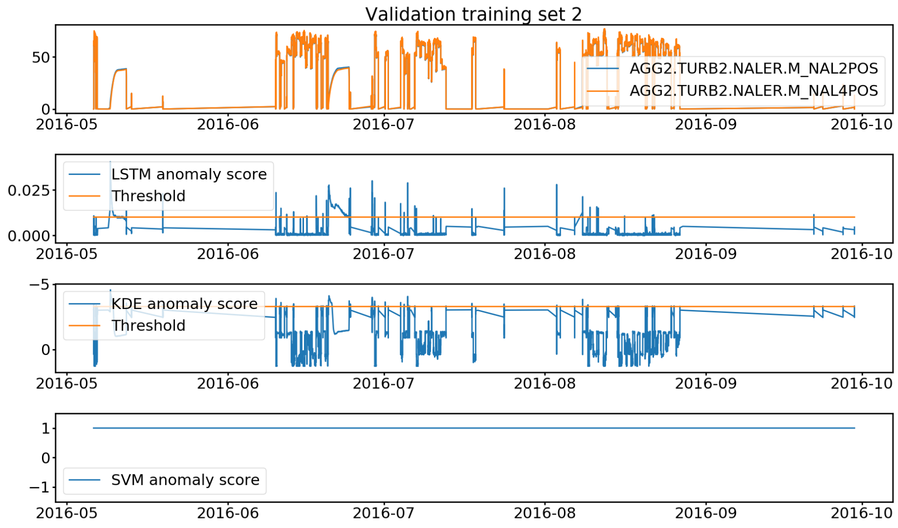

Figure D.3: Anomaly results for the methods trained on training set 2 evaluated on its validation set. How to interpret the figure can be seen in Figure 5.5. As can be seen the anomaly scores for the KDE and LSTM anomaly scorers are much lower than for the production and artificial data seen in the analysis. One datapoint is classified as an anomaly by the one class SVM classifier, it is also given a higher anomaly score by the scorers.

Appendix E

Training case 3

Additional plots from the results of training set 3 is presented below. The plots are discussed in their caption.

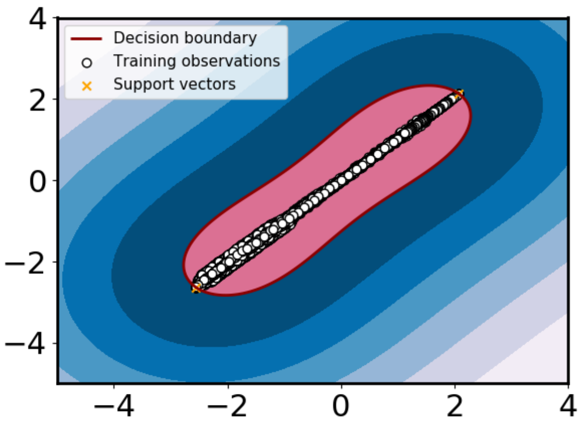

Figure E.1: Decision boundary for the one class SVM classifier. Support vectors and training observations are plotted. All samples located outside the red boundary are classified as anomalies. The decision boundary is very similar to the one seen for training set 2. The color indicates the distance to the separating hyperplane.

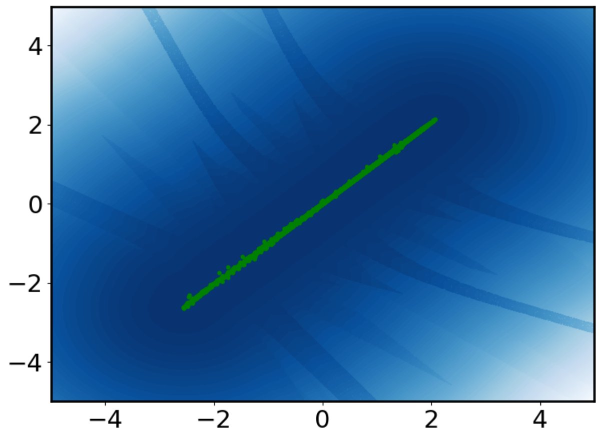

Figure E.2: The KDE anomaly scorers probability score plotted as contours in the space of the scaled training data. Training samples are seen in green. As the grid value becomes more and more anomalous the color becomes lighter.

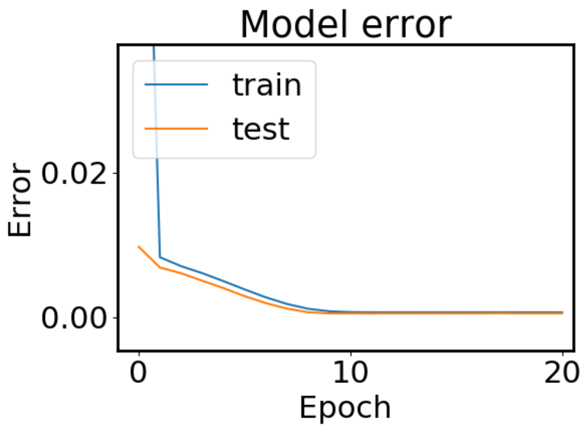

Figure E.3: Model error for the LTSM network as the number of epochs increase.

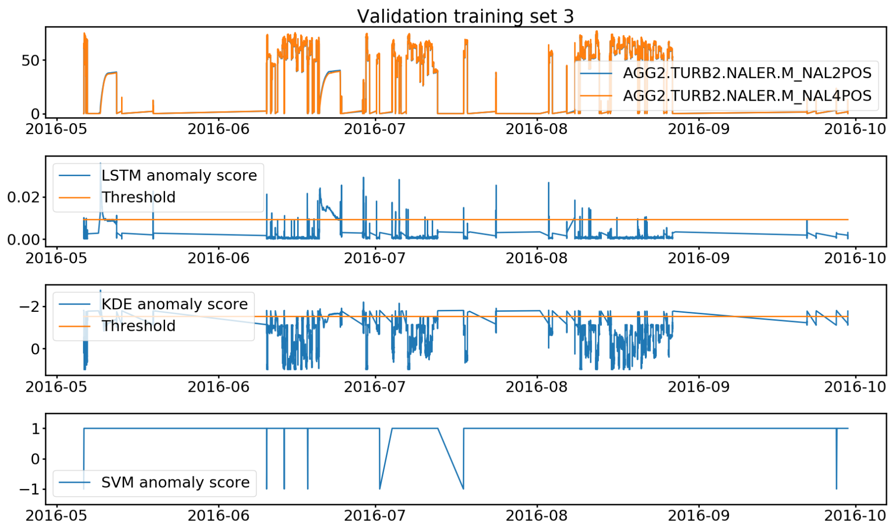

Figure E.4: Anomaly results for the methods trained on training set 3 evaluated on its validation set. How to interpret the figure can be seen in Figure 5.5. As can be seen the anomaly scores for the KDE and LSTM anomaly scorers are similar to the training set. The anomaly score is still much lower than for test case 1 and 2. Several datapoints are evaluated as anomalies by the OC SVM classifier. This can as mentioned be because the hyperparameters of the OC SVM is tuned for another dataset.

Appendix F

Test case 3 and 4

Here the figures for the anomaly score for test case 3 and 4 when training set 1 and 3 are used for training the methods.

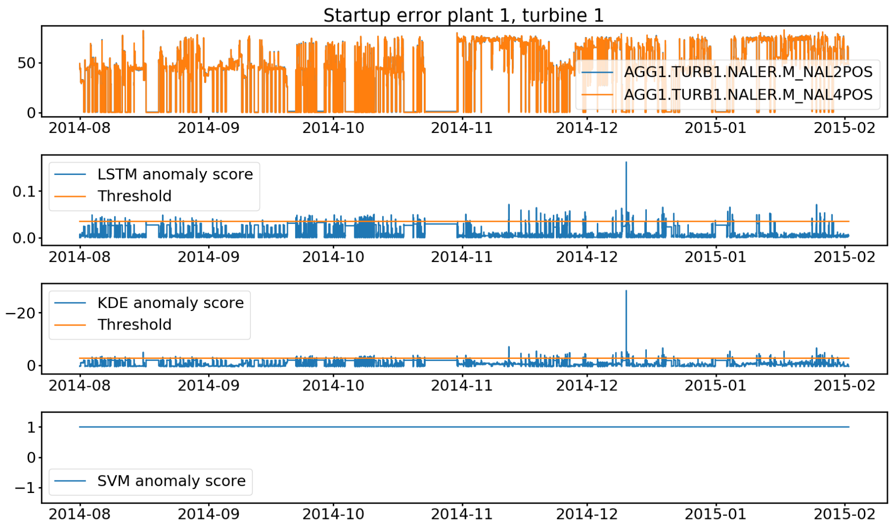

Figure F.1: Anomaly results for the methods trained on training set 1 evaluated on test case 3. How to interpret the figure can be seen in Figure 5.5. The results are very similar to the ones seen for training set 2.

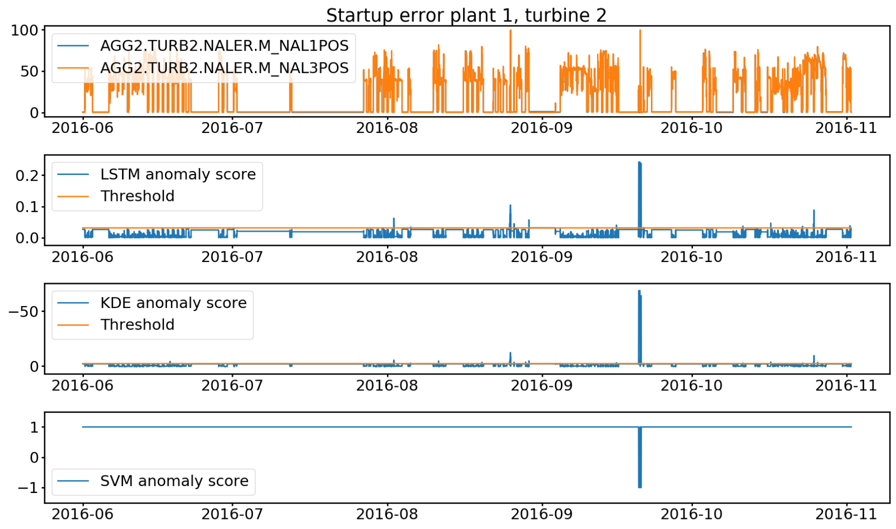

Figure F.2: Anomaly results for the methods trained on training set 1 evaluated on test case 4. How to interpret the figure can be seen in Figure 5.5. The results are very similar to the ones seen for training set 2.

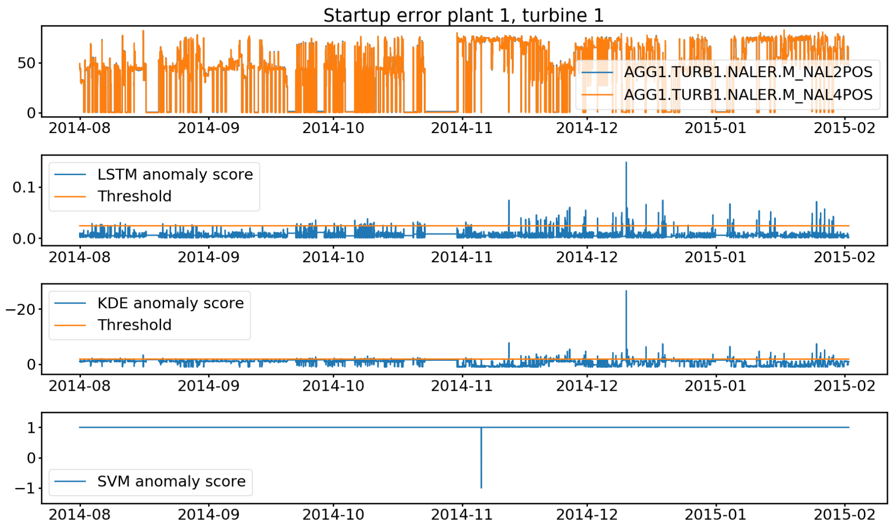

Figure F.3: Anomaly results for the methods trained on training set 3 evaluated on test case 3. How to interpret the figure can be seen in Figure 5.5. The results are very similar to the ones seen for training set 2.

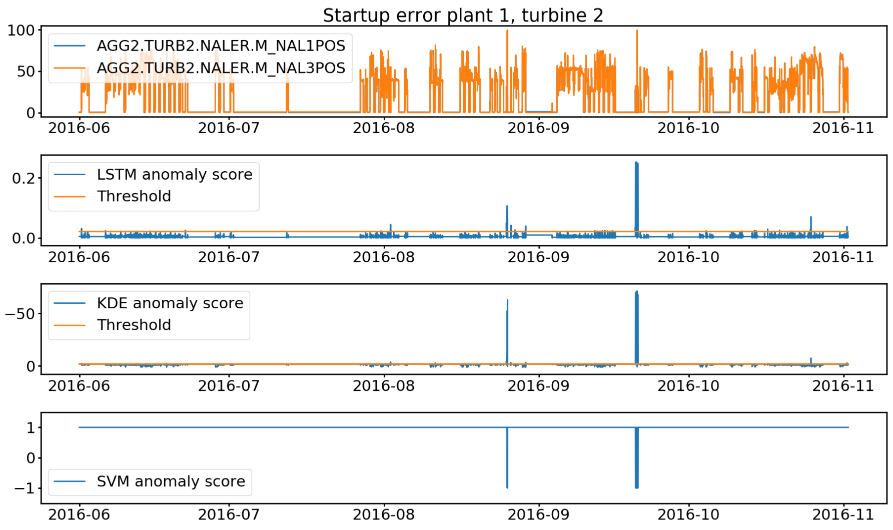

Figure F.4: Anomaly results for the methods trained on training set 3 evaluated on test case 3. How to interpret the figure can be seen in Figure 5.5. The results are very similar to the ones seen for training set 2.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.
URL: *https://arxiv.org/pdf/1603.04467.pdf http://arxiv.org/abs/1603.04467*

Chollet, F. (2017). Keras, commit 04e0a10, `https://github.com/fchollet/keras`.

Dy, J. G. and Brodley, C. E. (2004). Feature Selection for Unsupervised Learning, *Journal of Machine Learning Research* **5**: 845–889.
URL: *http://www.jmlr.org/papers/volume5/dy04a/dy04a.pdf*

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

Guyon, I. and Elisseeff, A. (2003). An Introduction to Variable and Feature Selection, *Journal of Machine Learning Research (JMLR)* **3**(3): 1157–1182.
URL: *http://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf*

Hastie, T., Tibshirani, R. and Friedman, J. (2001). The Elements of Statistical Learning, *The Mathematical Intelligencer* **27**(2): 83–85.

URL: *https://web.stanford.edu/%7B %7Dhastie/Papers/ESLII.pdf http://www.springerlink.com/index/D7X7KX6772HQ2135.pdf%255Cnhttp://www-stat.stanford.edu/%7B %7Dtibs/book/preface.ps*

He, X., Cai, D. and Niyogi, P. (2005). Laplacian Score for Feature Selection, *Advances in Neural Information Processing Systems 18* pp. 507–514.
URL: *http://papers.nips.cc/paper/2909-laplacian-score-for-feature-selection.pdf*

Hochreiter, S. and Urgen Schmidhuber, J. (1997). LONG SHORT-TERM MEMORY, *Neural Computation* **9**(8): 1735–1780.
URL: *http://www7.informatik.tu-muenchen.de/ hochreit http://www.idsia.ch/ juergen http://www7.informatik.tu-muenchen.de/ hochreit%5Cnhttp://www.idsia.ch/ juergen*

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, *Appears in the International Joint Conference on Articial Intelligence (IJCAI)*, pp. 1–7.
URL: *http://robotics.stanford.edu/ ronnyk*

Kraskov, A., Stögbauer, H. and Grassberger, P. (2004). Estimating mutual information, *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics* **69**(6): 16.
URL: *https://journals.aps.org/pre/pdf/10.1103/PhysRevE.69.066138*

Latecki, L. J., Lazarevic, A. and Pokrajac, D. (n.d.). Outlier Detection with Kernel Density Functions.
URL: *https://cis.temple.edu/ latecki/Papers/mldm07.pdf*

Li, W. (1990). Mutual information versus correlation functions, *J. Stat. Phys.* **60**(5): 823–837.
URL: *https://link.springer.com/content/pdf/10.1007%2FBF01025996.pdf papers2://publication/uuid/805C6936-1BA8-4632-B6C6-5BB2CDB979C3*

Liu, H. N. U. o. S., Motoda, H. O. U., Setiono, R. and Zhao, Z. (2010). Feature Selection : An Ever Evolving Frontier in Data Mining, *Journal of Machine Learning Research: Workshop and Conference Proceedings 10: The Fourth Workshop on Feature Selection*

in Data Mining **10**: 4–13.
URL: *http://proceedings.mlr.press/v10/liu10b/liu10b.pdf*

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P. and Shroff, G. (2016). LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection.
URL: *https://arxiv.org/pdf/1607.00148.pdf http://arxiv.org/abs/1607.00148*

Malhotra, P., Vig, L., Shroff, G. and Agarwal, P. (n.d.). Long Short Term Memory Networks for Anomaly Detection in Time Series.
URL: *https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2015-56.pdf*

Manuca, R. and Savit, R. (1996). Stationarity and nonstationarity in time series analysis, *Physica D: Nonlinear Phenomena* **99**(2-3): 134–161.
URL: *http://linkinghub.elsevier.com/retrieve/pii/S016727899600139X*

Molina, J. M., Isasi, P., Berlanga, A. and Sanchis, A. (2000). Hydroelectric power plant management relying on neural networks and expert system integration, *Engineering Applications of Artificial Intelligence* **13**(3): 357–369.
URL: *https://ac.els-cdn.com/S0952197600000099/1-s2.0-S0952197600000099-main.pdf?_tid=4e916db7-3e66-4e1f-9d9d-9151ab9ca503&acdnat=1526287990_d9e9a9d5b113ada40a1f34c61dd1e37b*

Paish, O. (2002). Small hydro power: Technology and current status, *Renewable and Sustainable Energy Reviews* **6**(6): 537–556.
URL: *www.elsevier.com/locate/rser*

Pascanu, R., Mikolov, T. and Bengio, Y. (n.d.). On the difficulty of training recurrent neural networks.
URL: *http://proceedings.mlr.press/v28/pascanu13.pdf*

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* **12**: 2825–2830.

Peng, H., Long, F. and Ding, C. (2005). Feature selection based on mutual information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy, *IEEE Trans. on Pattern Analysis and Machine Intelligence* **27**(8): 1226–1238.

Pimentel, M. A., Clifton, D. A., Clifton, L. and Tarassenko, L. (2014). A review of novelty detection.
URL: *https://www.sciencedirect.com/science/article/pii/S016516841300515X?via%3Dihub*

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J. and Williamson, R. C. (2001). Estimating the Support of a High-Dimensional Distribution, *Neural Computation* **13**(7): 1443–1471.
URL: *http://www.mitpressjournals.org/doi/10.1162/089976601750264965*

Scikit-learn 0.19.1 (n.d.). `http://scikit-learn.org/stable/install.html`.

Selak, L., Butala, P. and Sluga, A. (2014). Condition monitoring and fault diagnostics for hydropower plants, *Computers in Industry* **65**(6): 924–936.
URL: *https://ac.els-cdn.com/S0166361514000402/1-s2.0-S0166361514000402-main.pdf?_tid=8e73ce27-8ead-4ac5-adee-2ac822165479&acdnat=1525945025_d5a75c0ad338e97d87f85430579d1b1f*

Statkraft (2009). Vannkraft.
URL: *https://www.statkraft.no/globalassets/old-contains-the-old-folder-structure/documents/no/vannkraft-09-no_tcm10-4585.pdf*

Tarassenko, L., Clifton, D. A., Bannister, P. R., King, S. and King, D. (2009). Novelty Detection, *Encyclopaedia of Structural Health Monitoring* pp. 653–675.
URL: *https://pdfs.semanticscholar.org/a65d/7c7137968e291f0732b12110e485276cbe11.pdf*

Widodo, A. and Yang, B.-S. (2007). Support vector machine in machine condition monitoring and fault diagnosis, *Mechanical Systems and Signal Processing* **21**(6): 2560–2574.
URL: *https://ac.els-cdn.com/S0888327007000027/1-s2.0-S0888327007000027-main.pdf?_tid=2aa3df2c-ee0b-4e63-a335-*

2052850aeeea&acdnat=1526288040_dd85dd11edbdba4dad94202201df58bb
http://linkinghub.elsevier.com/retrieve/pii/S0888327007000027

Åsnes, A. (2017). Machine learning for condition monitoring in hydroelectric power plants, project thesis fall 2017.