Evaluation of machine learning methods for anomaly detection in combined heat and power plant

FREDRIK CARLS
Evaluation of machine learning methods for anomaly detection in combined heat and power plant

FREDRIK CARLS

Master in Computer Science and Machine Learning
Date: June 27, 2019
Supervisor: Somayeh Aghanavesi
Examiner: Erik Fransén
School of Electrical Engineering and Computer Science
Host company: Stockholm Exergi
Swedish title: Utvärdering av maskininlärnings-metoder för anomalidetektion i kraftvärmeverk
Abstract

In the hope to increase the detection rate of faults in combined heat and power plant boilers - thus lowering unplanned maintenance - three machine learning models are constructed and evaluated. The algorithms; k-Nearest Neighbor, One-Class Support Vector Machine, and Auto-encoder have a proven track record in research for anomaly detection, but are relatively unexplored for industrial applications such as this one due to the difficulty in collecting non-artificial labeled data in the field.

The baseline versions of the k-Nearest Neighbor and Auto-encoder performed very similarly. Nevertheless, the Auto-encoder was slightly better and reached an area under the precision-recall curve (AUPRC) of 0.966 and 0.615 on the training- and test period, respectively. However, no sufficiently good results were reached with the One-Class Support Vector Machine. The Auto-encoder was made more sophisticated to see how much performance could be increased. It was found that the AUPRC could be increased to 0.987 and 0.801 on the training- and test period, respectively. Additionally, the model was able to detect and generate one alarm for each incident period that occurred under the test period.

The conclusion is that ML can successfully be utilized to detect faults at an earlier stage and potentially circumvent otherwise costly unplanned maintenance. Nevertheless, there is still a lot of room for improvements in the model and the collection of the data.

Keywords:
Machine Learning, Anomaly detection, Fault detection, Health/condition monitoring, Sensor surveillance, PHM, CHP Plant Boilers, k-Nearest Neighbor, One-Class Support Vector Machine, Auto-encoder
**Sammanfattning**

I hopp om att öka identifieringsgraden av störningar i kraftvärmepannor - och därigenom minska oplanerat underhåll - konstrueras och evalueras tre maskininlärningsmodeller. Algoritmerna; k-Nearest Neighbor, One-Class Support Vector Machine, och Auto-encoder har bevisad framgång inom forskning av anomalidetektion, men är relativt outforskatade för industriella applikationer som denna på grund av svårigheten att samla in icke-artificiell uppmärkt data inom området.


Slutsatsen är att ML framgängsrikt kan användas för att identifiera störningar i ett tidigare skede och därigenom potentiellt kringgå in annat fall dyra oplanerade underhåll. Emellertid finns det fortfarande mycket utrymme för förbättringar av modellen samt inom insamlingen av data.

**Nyckelord:**
Maskininlärning, Anomalidetektion, Feldetektering, Tillståndsbevakning, Sensorövervakning, PHM, Kraftvärmeverkpannor, k-Nearest Neighbor, One-Class Support Vector Machine, Auto-encoder
Acknowledgements

First, I would like to thank Stockholm Exergi and especially Anders Karlsson and Leo Jakobsson for their excellent supervision and valuable knowledge in the field. Our discussions have genuinely been rewarding, and I have learned a lot these last couple of months, thanks to you.

Additionally, I would like to thank Somayeh Aghanavesi, my supervisor at KTH, for her guidance throughout the process and excellent feedback.

Further, I would like to thank the coworkers at Stockholm Exergi for making sure that I always felt welcome and providing me with what was required to carry out this work.

Finally, I would like to thank my wonderful family, friends and my fiancée Rebecka. Thank you for all the encouragement and support, not only for this thesis but also throughout the whole education and in life.

Sincerely,

Fredrik Carls
KTH Royal Institute of Technology, June 2019
## Contents

1 **Introduction** 1
   1.1 Research question ................................................. 2
   1.2 Objective ....................................................... 2
   1.3 Challenges ..................................................... 4
   1.4 Societal impact .................................................. 5

2 **Background** 7
   2.1 Introduction To Research Area .................................... 7
     2.1.1 Energy Systems ................................................ 7
     2.1.2 Machine Learning ............................................. 8
   2.2 Data ............................................................. 8
   2.3 Theory .......................................................... 9
     2.3.1 The Plant ..................................................... 9
     2.3.2 Anomaly Detection ........................................... 12
     2.3.3 Semi-supervised Learning ................................... 13
     2.3.4 k-Nearest Neighbor ......................................... 14
     2.3.5 Support Vector Machine ..................................... 16
     2.3.6 Artificial Neural Network ................................... 19
     2.3.7 Auto-Encoder ............................................... 21
   2.4 Related Work ................................................... 23

3 **Methodology** 26
   3.1 Data preprocessing ............................................... 26
   3.2 Models ........................................................ 27
     3.2.1 k-Nearest Neighbor ......................................... 29
     3.2.2 One-class Support Vector Machine ......................... 30
     3.2.3 Auto-encoder ................................................. 31
   3.3 Performance metrics .............................................. 32

4 **Results** 35
   4.1 k-Nearest Neighbor .............................................. 35
   4.2 One-class Support Vector Machine ................................ 40
   4.3 Auto-encoder ................................................... 43
   4.4 Improved version of AE .......................................... 48
   4.5 Summary ........................................................ 54
1 Introduction

The boilers are critical components of combined heat and power (CHP) plants based on waste incineration. Monitoring their health is essential in ensuring efficient plant operation and preventing costly unplanned maintenance. However, the performance (detection rate) of fielded anomaly detection solutions in this area are still inadequate. Many disturbances that could potentially have been counteracted remain unnoticed until it is too late, and the damage has already happened.

Stockholm Exergi (Stockholm Exergi 2019) acts within the field of energy production and distribution and is co-owned by Fortum and the city of Stockholm. The company aims to produce environmental and resource-efficient district heating and cooling as well as electricity. This thesis investigates the circulating fluidized bed boiler at Stockholm Exergi’s waste incineration CHP plant located in Högdalen, Sweden. Like any extensive system, the plant and its boilers are subject to haphazard faults as a result of equipment degradation by aging and mishandling. In such an occurrence, the boiler must be stopped, faults identified, and repairs conducted on faulty parts. The normal behavior of the system is that of stable operation and production, while anomalies are characterized by machinery faults. While some part of the system is faulty it usually affects one or several sensors attached to the boiler in such a way that the signal readings significantly deviate from their nominal contextual values. Left unattended, these initially relatively insignificant faults might escalate to a point in which unplanned maintenance of the system is inevitable. To prevent this, it is essential to work predictively. Ideally, it is possible to identify parts of the system working sub-optimally and perform maintenance before the situation worsens.

Today at Stockholm Exergi, lots of data is collected from sensors attached to the machinery. The data is fed to a control system in which the signals are monitored and used to alter operation. The control system automatically optimizes the operation of the boilers to maximize energy efficiency and minimize environmental impact. For anomaly detection, handcrafted knowledge-based (based on domain and engineering expertise) rules are used to raise alarms. In the case of an alarm, system engineers can alter operation. If the situation escalates, parts of or the whole boiler are shut down for a thorough investigation. To prevent these unplanned maintenances from happening, Stockholm Exergi works predictively through frequent in-plant inspections and data trend analysis. However, this manual analyzes of data are ambiguous, unreliable, and inconsistent. Additionally, the knowledge-based rules are both laborious in designing and developing and perform inadequately. By the time problems
are identified, they have sometimes been ongoing for a while, making them both difficult and costly to solve.

Digitalization, digital transformation, and similar buzz words all aim to increase productivity and efficiency through the use of technology and data. The key enabler to reach this goal is the ability to extract useful information from data. This is where machine learning (ML) comes into play. As this thesis investigates the potential for ML to aid in detecting faults of the aforementioned CHP plant’s boilers, the wider area within which the degree project will be carried out is that of ML.

1.1 Research question

The research question addressed in this thesis is:

*To what extent can machine learning be used to predict anomalies in a combined heat and power plant boiler, and which model is best suited for the task?*

In practice, the question is explored by evaluating three baseline (low complexity) versions of state-of-the-art ML models in the field: k-Nearest Neighbor, One-Class Support Vector Machine, and Auto-encoder. The evaluation investigates which model performs best in predicting meaningful anomalies from the boiler’s complex time series sensor data. In addition, the best performing baseline model is further explored to find to what extent it can be improved through sophisticated measures.

1.2 Objective

The broader goal of this thesis is to develop a support system that utilizes ML to achieve better predictability of system faults. In case the faults are identified at an earlier stage, precautions can be taken sooner rather than working reactively. Consequently, the support system has the potential of lowering the number of unplanned maintenances of the system by flagging for scheduled maintenance. Hence, the objective is to build ML models that utilize a boilers’ sensor data to detect meaningful anomalies and investigate what model is best suited for the task. Meaningful anomalies translate to data that indicates real problems in the system that requires attention. In the case of this thesis, anomalous data is associated with some malfunction, fault, or defect in the system.
The motivation behind the use of ML in combination with the control system is that many faults are novel to the system and thus difficult to identify through the use of knowledge-based constructed rules. In addition, rules (or rather thresholds) applied to individual sensors often fail as faults usually follow a combination of disturbances over several sensors (see Figure 1).

Figure 1: While it is difficult to identify outliers in $X$ or $Y$ individually, it becomes easier if they are viewed together in a scatter plot. Image taken from Flovik 2018

**The principal’s interest**

Of course, any stop in production is very costly. If there is some way to potentially reduce disturbances or at least schedule them to a time of low production to minimize losses, Stockholm Exergi is willing to invest time and resources towards that end. The resulting models might aid the planning of maintenance by identifying faults at an early stage and prevent them from developing severe problems. As such, unplanned maintenance of the system can be kept at a minimum.
1.3 Challenges

While building ML models for machine fault detection, one is faced with several challenges, most of which originates from the data. Most often - as is the case for Stockholm Exergi - the majority of data available is unlabeled as it is costly and often even impossible to label the data due to the uncertainty of true events. This means that learning cannot be guided. Instead, the model must find the underlying structures by itself. The difficulty in labeling faulty data arises from the lack of an intuitive way to classify a system operation snapshot of several thousand sensor readings as either normal or abnormal behavior. Additionally, since the data is a time series, there is no clear period for which the anomalies should be given. Compared to data less affected by the time dimension - like fraud detection systems - where each data point can be analyzed in isolation, it is difficult to decide when the anomaly starts and when it ends. For example, while an anomaly can be given by the time in which a disturbance occurs, it is difficult to decide at what point the anomaly started. The problem that caused the disturbance might have existed for a while before it was identified. The complexity of this challenge is further increased by the fact that there often exists a delay between disturbance discovery, maintenance and report.

Then there is the problem with the data itself and how to deal with its noise. Noisy data could be interpreted as abnormal due to the blurred boundary between normal and abnormal behavior. To complicate the problem further, the definition of abnormal or normal behavior may change over time. For example, the change can be a result of a drifting data mean, seasonality of data, or merely a change in equipment. Due to natural fluctuations of the data, it is often difficult to discern the health of the system at any given time.

Apart from the challenges related to the data, the difficulty of anomaly detection, in general, arises from the task of predicting the unknown. An anomaly is an event that in many cases is happening for the first time in the system. Thus it cannot be found in historical data. Specifically, for the investigated boiler, anomaly detection is technically challenging as the system is complex. The operating conditions are heavily dependent on many factors (e.g. waste properties and equipment aging).

As the consequences of system faults are severe, it is vital that the model manages to identify as many faults as possible. However - for the model to be useful in practice - it is essential to minimize the number of false alarms. The challenge is to find the best trade-off by setting a threshold on the data point anomaly level that generates
alarms.

Another challenge lies in what to do once an anomaly has been identified. As the ultimate goal of the tool is to reduce unplanned maintenance, the system operator needs more information than knowing that an anomaly is likely to happen. Since anomalies often are identified through a combination of signal values, there is no obvious way to connect an arbitrary anomaly with a specific part of the system. I.e. an anomaly detection model might be able to tell that something is wrong, but the difficulty lies in interpreting what caused the problem so that maintenance can be performed accordingly.

1.4 Societal impact

The societal impact of this thesis largely depends on the outcome of the experiments. There are four possible scenarios; 1) the model is not able to detect anomalies, 2) the model is too sensitive and catches too many anomalies, 3) the model catches too few anomalies, or 4) that the model can detect anomalies satisfactorily. If the model is not able to detect anomalies, the thesis has little impact as the company has little too loose from this endeavor (apart from invested time and resources), although hopefully, they have gained some valuable insights on their data and what does not work for them. Similarly, if the model is too sensitive, it has no use in practice as too much work is required to filter out all the false alarms. Instead, if the model’s detection rate is too low, it would probably mean that most of the alarms it generates represent real problems in the system. While such a model might help prevent some unplanned maintenances of the system, it also might induce a kind of false comfort for the operators as it will only catch some faults while missing most. The final scenario that the model can detect anomalies satisfactorily is, of course, the optimal. That way, the model might help in preventing some of the many unplanned maintenances occurring today. Nevertheless, it should still be considered a considerable risk to blindly rely on the model as it still might leave some faults undetected.

Many companies are in the same position as Stockholm Exergi today, they have lots of data but are not quite sure what to do with it. Not least does this work have the potential to benefit Stockholm Exergi, but also other actors in the energy field as well as those active in other areas that utilize sensor data in production. It will hopefully shed light on the potential of ML and the usability for the methods compared in this thesis for this particular area.

From a sustainability perspective, the outcome is quite similar. If the tools are
able to identify anomalies well and help circumvent unplanned maintenance, the environmental CHP plants will become more reliable and achieve increased energy efficiency.
2 Background

This section introduces the reader to the research area of the thesis, the theory behind the energy system under investigation as well as the selected algorithms, and some previous work in anomaly detection.

Glossary for the reader less accustomed to Machine Learning:

Data point . A single instance of data containing some features. It can be labeled or unlabeled.

Features . The value types contained in the data point. For example, age and length.

Label . The correct prediction of a data point. This could be a classification. For example, a given data point represents a normal behavior or an anomaly.

Dataset . A collection of data points. A public dataset is available to everyone.

Training . Feeding model with data points iteratively and optimizing model parameters.

2.1 Introduction To Research Area

In this section, the reader is familiarized with the field of research. A short presentation is provided on relevant areas and how they relate to this thesis.

2.1.1 Energy Systems

An energy system can be defined in many ways, depending on the perspective. From a structural view it can be defined like any system, a regularly interacting or interdependent group of items forming a unified whole (Definition of SYSTEM 2019). A process view defines an energy system as consisting of an integrated set of technical and economic activities operating within a complex societal framework (Hoffman and Wood 1976). An energy system defined from an engineering perspective is that
of a flow network that map components and the interfaces between them (Quelhas et al. 2007).

The United Nations Intergovernmental Panel on Climate Change (IPCC) has the following definition of an energy system (Pachauri et al. 2014):

"The energy system comprises all components related to the production, conversion, delivery, and use of energy."

In this thesis, the focus will solely be put on the production of energy. More specifically, the theory part will go in-depth about the technology surrounding the investigated boiler.

2.1.2 Machine Learning

What is learning? What is machine learning? These are philosophical questions of little concern to this thesis. Instead, the emphasis is put on the practical applications of ML. In short, ML can be described as a computer program that learns some pattern while performing some task. A more formal definition is provided by Mitchell 1997:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."

The theories behind the chosen ML methods are further explained under the theory part below.

2.2 Data

The data was supplied by sensors attached to a circulating fluidized bed (CFB) part of a CHP plant located in Högdalen. More than 2000 sensor signals were provided to train and evaluate the model that aims to discern the health of the system. A couple of examples of sensor measurements include temperature, pressure, rotational speed, PH-value, effect, position, flow, current, weight, frequency, and voltage. The data of each sensor signal was supplied as a CSV file containing more than fourteen months of historical data per minute sample rate. That is a total sample size of more than 600 000, where each sample contains about 2000 features - resulting in
approximately 1 200 000 000 individual data points. In addition, each sensor sample included an indication of the data collection quality.

2.3 Theory

This section will describe the energy systems under investigation and the concepts behind the selected ML models.

2.3.1 The Plant

This thesis investigates the industrial waste incineration based circulating fluidized bed boiler at Högdalen CHP plant (Högdalen - power plant in Sweden — Fortum 2019). The plant is located in the southern suburbs and is mainly based on waste incineration cogeneration, although bio-oil heat only boilers are available for peak production (Levihn and Nuur 2015). A brief description of the technology surrounding the boiler is provided in the following sections.

Figure 2: Högdalen plant
Combined Heat and Power

Combined Heat and Power (CHP), or Cogeneration, is the simultaneous generation and useful application of electricity and useful heat (Pachauri et al. 2014). Electricity is produced through high-pressure steam being pushed through a rotor inside a turbine. The steam, in turn, is created from heated boilers. As such, heat is formed regardless. The generation of useful heat means that the heat is harvested after passing through the turbine. In many parts of the world, the steam is cooled down in special cooling towers which results in significantly lower energy efficiency. In the CHP however, the steam continues its journey after passing through the turbine by colliding with a bunch of steel pipes in a heat exchanger. The heat is then distributed through the district heating network in Stockholm. The utilization of the generated heat yields a significant increase in fuel and energy efficiency. Meanwhile, it also lowers the amount of emissions ($CO_2$) released into the atmosphere compared to the separate production of electricity and heat. As is the case with Stockholm Exergi, heating the boiler with carbon-neutral fuels to the mix substantially reduces emissions. Unlike single-source power and heat production with natural gas or coal, CHP with biomass and waste in the fuel mix reduces $CO_2$ emissions by up to 40% (Combined heat and power - efficient power generation — Fortum 2019).

Circulating Fluidized Bed

Circulating Fluidized Bed (CFB) is a fuel combustion steam generation technology. Due to its low temperature burning process, CFB technology can utilize all types of carbon-neutral fuels like biomass and recycled waste to produce clean and economical power generation. A major advantage over the limitation of conventional combustion is the ability to burn a wide range of fuels (Basu 2015). In contrast to conventional coal technology, CFB does not require the fuel to be finely ground and dried before entering the furnace. Instead, the fuel is coarsely crushed and dropped into fuel chutes which leads to ports in the lower section of the CFBs furnace. Unlike conventional boilers that burn the fuel in a massive high-temperature flame, CFB technology utilizes circulating hot solids (e.g. sand) to cleanly and efficiently burn the fuel in a flameless combustion process. CFB uses re-circulation of particles carried with the combustion air to maximize fuel efficiency and reduce the consumption of bed material. The separation of gas and solids is done in the cyclone. CFBs low combustion temperature minimizes the formation of nitrogen oxide and allows the injection of limestone to capture acid gases as the fuel burns, thus lowering furnace emissions. Since the fuel’s ash does not melt, heat transfer services stay clean, allowing the hot solids to conduct their heat efficiently throughout the entire boiler.
while corrosions are minimized. An overview of the boiler structure is provided in Figure 3

![Illustration of typical CFB structure. Image taken from Basu 2015](image)

**Machine Control system**

A distributed control system (DCS) based on supervisory control and data acquisition (SCADA) technology is used to supervise plant operation. The control system is connected to the plant sensors and can control the flow of material through the use of programmable logic controllers (PLC), discrete proportional–integral–derivative (PID) controllers and control valves. It sends a setpoint to the controller which alters its associated valve to reach the desired setpoint. Data is collected and centralized from computerized autonomous controllers distributed throughout the system. Consequently, control functions are localized near the process plants while allowing for remote monitoring and supervision. Thus, the real-time control logic and calculations are performed locally connected to the field sensors and actuators. This structure provides reliability as a process failure only affects a part of one plant process, as opposed to a failure of the whole system in case all control is handled centralized. To detect alarms, the control system monitors whether certain alarm conditions are satisfied. Once detected, the system notifies the management of the alarm. The
alarm indications remain active until the alarm conditions have been cleared. The conditions can be explicit or implicit. Alarms are either generated through some formula based on several sensor values or through individual sensor values that lie outside their threshold limit values.

Common Faults
A primary concern of any plant is the development of corrosion. To prevent this, a certain temperature must be maintained in the boiler at all times. Another factor that can affect energy production and therefore the sensor data includes the variance of granularity and material type of industrial waste. From time to time, parts of the waste get stuck between the tubes feeding the CFB. Additionally, machinery components deteriorate with usage and age. Depending on the component criticality, parts of or the whole system must be stopped to perform maintenance or repairs. Finally, the data collection could even be affected by the plant operator. As they work in shifts, personal preferences to how the boilers should be optimized may alternate, thus influence the data.

2.3.2 Anomaly Detection
Anomaly detection can be described by the process of identifying events or observations that significantly differs from the majority of the data, thus raising suspicion. They are patterns in the data that do not conform to expected behavior. As such, anomalies represent some noise or outliers in the data. As these outliers deviate from the nominal data, they typically translate to some problems. For example, there might be a pipe leakage in the boiler, which causes some temperature, pressure, or other sensors to fluctuate. In that case, the model should catch this fluctuation as the system is behaving unusually and hopefully backtrack the anomaly to the cause and flag for repairs.

Anomalies can be classified into three broader categories; point, contextual, and collective outliers (Chandola, Banerjee, and Kumar 2009). A point outlier refers to an individual data instance and is considered as anomalous with respect to the rest of the data. For example, a point outlier could be an unusual amount spent during a transaction in a credit card fraud detection system.

On the other hand, if a data point is treated as abnormal in a specific context but not otherwise, it is considered a contextual outlier. For example, a two-meter-tall adult may be normal, but if viewed in context with age, a two-meter-tall child is
considered as an anomaly. The same can be said for temperature time-series. $-10^\circ C$ is not uncommon during the winter, but is considered as an anomaly if occurring during the summer (see Figure 4).

![Figure 4: Contextual anomaly. Figure adapted from Chandola, Banerjee, and Kumar 2009](image)

Lastly, if a collection of related data points together forms an anomaly with respect to the whole data set, it is considered a collective outlier (see Figure 5). For example, a certain data point value might not qualify as an anomaly by itself but if maintained over a long time by successive occurrences it might (Singh and Upadhyaya 2012).

![Figure 5: Collective anomaly.](image)

### 2.3.3 Semi-supervised Learning

While anomaly detection could be devised as a supervised learning problem, this is typically not feasible as very few or no labeled ground-truth examples of anomalous
data are available. Further, it is usually impossible or undesirable to collect them in practice as it may have an expensive and negative impact. Therefore, unsupervised approaches are commonly employed for anomaly detection tasks. However, unsupervised methods tend to draw some assumptions that make them less robust. For example, that infrequent behavior is anomalous (Vercruyssen et al. 2018). In practice, this assumption is not valid, as most realistic settings contain some normal behaviors that occur less frequently than other abnormal behaviors.

Semi-supervised learning falls between supervised learning and unsupervised learning, where supervised learning only uses labeled data, and unsupervised learning only uses unlabeled data. In comparison to unsupervised learning, a significant improvement in learning accuracy can be achieved by using a small amount of labeled data in conjunction with the unlabeled data to guide the learning. Meanwhile, the time and cost required for supervised learning can be reduced.

Semi-supervised anomaly detection techniques build a model depicting normal behavior by training the model on nominal data. After all, since the majority of a dataset by definition represents "normal" behavior, it is often not too difficult to find such data with high confidence. Since the model is trained to recognize the normal behavior of the system, it can decide whether an arbitrary data sample is an anomaly by calculating the likelihood of that test instance to be generated by the model. In practice, what is happening will differ depending on which method is used. For example, the model might have some cost associated with the reconstruction of a data sample. In that case, the reconstruction of a data sample characterized by an anomaly will be distorted and the cost high. Meanwhile, the reconstruction cost for nominal data will be lower since the model will be trained to minimize this cost.

2.3.4 k-Nearest Neighbor

k-Nearest Neighbor (k-NN) is among the simplest of all ML algorithms used for classification. The input consists of the k (user-defined value) closest training examples in a multidimensional feature space. The classification is decided by a majority vote by its neighbors, labeling it as the most common class in its vicinity. This makes the method sensitive to the local structure of the data (Phyu 2009).

To calculate the similarity between two points, some measure must be used. Usually, this measure is based on the distance between the points. The distance to a data point's neighbors can be calculated in a couple of different ways, the most commonly
used being the euclidean distance. The euclidean distance between two points \( (p \text{ and } q) \) equals the length of the line segment connecting them (Phyu 2009).

\[
d(p, q) = d(q, p) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}
\]  

(1)

Other distance metrics include Manhattan and Minkowski. In case the model is dealing with categorical variables, the Hamming distance must be used. Further, the distance metric can be learned with specialized algorithms like large margin nearest neighbor or neighborhood component analysis (Weinberger and Saul 2009). However, since this thesis only will use the euclidean distance, the other measurements are left for the reader to explore further.

A drawback of the method is that some configurations cannot handle skewness in data well. Skewness is the measure of asymmetry in the probability distribution around the mean value. This translates to the problem that a more frequent class tends to dominate the classification. They are more common in most neighborhoods due to their sheer large number of occurrences. A useful trick to overcome this problem is to assign weights proportional to the inverse of the distance to the neighbors so that its relative contribution towards the classification is affected by its distance to the data point (Roy and Madhvanath n.d.).

In k-NN, k is a hyperparameter. A hyperparameter is a parameter whose value is used to control the learning process. I.e. it is a parameter that is not learned. Hyperparameter optimization can be used to find a good value for k. In practice, this often translates to an exhaustive search through a manually specified subset of the hyperparameter space based on some performance metrics like cross-validation on the training set. The optimal value for k depends on the data. Typically, increasing k reduces the effect of noise but lessens the class boundary distinctions (Wang, Jia, and Liu 2015). If the classification problem is binary (two classes), it is best to choose an odd k to avoid tied votes.
Figure 6: The classification of the red point will differ depending on our value for k. Image adapted from José 2018

An example related to Figure 6 could be a prediction of an unknown drink where the yellow samples represent beer, and the purple samples represent wine. In this example, the first variable $X_1$ could refer to some measurement of the drink’s color, and the second variable $X_2$ could refer to the alcohol in the drink. The model has already been trained on all the samples but the red one. Choosing a $k$ value of 3 results in the model predicting that the drink is wine, whereas choosing $k = 6$ leads to a model predicting the unknown drink as a beer.

The accuracy of the k-NN algorithm is greatly affected by the feature selection and normalization. I.e. performance will suffer from noisy or irrelevant features or if the feature scale does not match its importance.

2.3.5 Support Vector Machine

Support Vector Machine (SVM) is an ML algorithm used for classifying data. By the use of labeled training vectors, SVM aims to find the optimal hyperplane that separates two classes such that the hyperplane can be used to classify future data points (Shmilovici 2010). The data points are predicted to belong to the class based
on which side of the hyperplane it falls. The hyperplane’s margin should be as wide as possible, meaning that the distance between the data points belonging to the two classes should be as large as possible. The plane that maximizes the margin is called the optimal decision boundary.

Kernel Trick

The original SVM could only perform linear classifications. However, by utilizing the kernel trick, the method can separate non-linearly separable classes. The only adjustment is to replace every dot product by a nonlinear kernel function. The same procedure is done but in a transformed feature space. So, while the classifier is a hyperplane in the transformed feature space, it can be nonlinear in the original space (Shmilovici 2010). Although using a higher dimensional feature space increases the generalization error, the algorithm still performs well given enough samples.
Figure 8: Non-linear decision boundary through creating hyperplane in a higher dimensional space transformation. Image taken from Shmilovici 2010

**One-Class SVM**

One-Class SVM (OCSVM) is an extension of the traditional SVM where training is solely based on nominal data (Schölkopf et al. 2001). This requires the standard formulation of SVM to be redefined. The basic idea is to assume the origin as our second class and try to find a separating hyperplane between the origin and our mapped samples. This is done by mapping the data into a higher dimensional feature space and envelop it in a hyper-sphere. For one-class SVM, maximizing the margin is comparable to minimizing the enclosed volume of the hyper-sphere containing the available class. The hyper-sphere strives to represent the class data distribution accurately (Chen, Zhou, and Huang 2001). Consequently, data points belonging to another class should end up outside the boundary in that feature space. However, the decision boundary should allow for as much margin as possible to avoid overfitting.
2.3.6 Artificial Neural Network

Artificial Neural Networks (ANNs) are a type of ML framework vaguely inspired by the brain. Once trained, an ANN model will input some features, modify the values through a series of layers before generating output. This output might be some classification. The layers between the input and output layers are called hidden layers and are made up out of nodes called neurons. The input layer is connected to the neurons of the first hidden layer by weights and biases. So, the feature values of the input data point are multiplied by the weight values connected to the neurons. Afterward, the associated bias is added to form a single number for each neuron. Some activation function inside the neuron then modifies this number before it passes on the value to the next layer. In the general architecture of a fully connected feed-forward artificial neural network, each neuron from layer $l$ is connected with the output of all the neurons from the previous layer $l-1$ and has its own set of weights and biases. This is also referred to as a Multilayer Perceptron (MLP) (Stegemann and Buenfeld 1999).
Forward Propagation

An ANN is trained by randomly initializing the weight matrix $W^l$. Similarly, the bias values are stored in a bias vector $b^l$, and the inputs for a certain layer are stored in vector $x^l$. The forward pass is given by

$$h^l = \sigma(W^l h^{l-1} + b^l)$$

where $\sigma$ is the activation function. The activation function can for example be $\sigma = max(0, x)$.

Upon reaching the final output layer, the model will have reached a prediction. Initially, the prediction will have a low performance as the weights are randomized. Essentially, what is being learned in this ML model are the optimal parameter values of weights and biases. They will iteratively be adjusted until the model accurately outputs the desired prediction based on training data.

Backward Propagation

Once a forward pass has been completed, the error generated by the difference between the desired outcome and the predicted output can be calculated with respect to some cost function. By minimizing this cost, the optimal parameter settings can be found. The parameters are adjusted to minimize the cost through gradient descent. The goal of gradient descent is to find a minimum of the cost by calculating the derivative of the cost function and move in the direction of the gradient.
By iteratively adjusting the parameters in the best direction, the performance of the model will increase on the training data. A hyperparameter called the learning rate sets the length of each iterative gradient step. Setting this value too low will make the learning very slow as each iteration will change the parameter values very little. However, setting it too large will probably mean that the minimum is missed and will cause the model to jump back and forth along the gradient. As the model only cares about the derivative, there is no guarantee that the model stabilizes at a global minimum. Instead, if the random initialization ends up close to a local minimum, the model is likely to get stuck there. However, the local minimum might be good enough.

![Gradient descent visualized. Depending on the initial parameter randomization, the model stabilizes in different cost minimums. Image taken from Stanford 2019](image)

**2.3.7 Auto-Encoder**

An Auto-Encoder (AE) is a type of feed-forward ANN that aims to learn data representation (or coding) unsupervised. Typically, the goal is to determine important structures of the data. By ignoring the noise, the dimensionality of the data can be reduced with minimal data loss. The AE consists of two phases; encoder and decoder. After encoding (reducing) the data, the AE decodes (reconstructs) the data and tries to generate the original representation of the initial input with minimal loss (Tan 2015). Architecturally, the AE is made up of an input layer, an output layer, and one or more hidden layers connecting them. As the model aims to reconstruct the input signal, the output layer is of the same size as the input layer.
To be more precise, the encoder is supplied with an input $x$ and performs a mapping

$$h = \gamma(Wx + b)$$

(3)

where $\gamma$ represents a non-linear activation function, $W$ the weights, and $b$ the bias. The decoder then tries to map the hidden representation $h$ back to the original representation of $x$.

$$z = \gamma(W'h + b')$$

(4)

The model parameters $\theta = [W, b, W', b']$ are optimized to minimize the reconstruction error between $z = f_\theta(x)$ and $x$. The reconstruction error can be given by the squared error, creating the optimization problem

$$\min_{\theta} \frac{1}{N} \sum_i^N (x_i - f_\theta(x_i))^2$$

(5)

where $x_i$ is the i-th sample.

---

**Denoising Auto-Encoder**

Denoising AE (DAE) works like a normal AE with the exception that the training data is corrupted in the encoding step. However, the goal is still to reconstruct the original non-corrupted training data in the decoding step of the algorithm. This way, the extracted features are more robust to input noise and will generate a better high-level representation of the data (Yan and Yu 2015). Consequently, the model
is better able to capture the dependence among the input variables. A couple of methods normally used to corrupt the input data include Gaussian noise, masking noise, and salt-and-pepper noise. For Gaussian noise, a random value with zero mean, and some variance is added to each sample. Masking noise means that some random features for each sample are forced to zero. Salt-and-pepper noise sets some random features of each sample to their minimum or maximum value according to a fair coin flip.

**Deep Auto-Encoder**

By adding more hidden layers, a deep AE is formed. This results in an ability to store complex contextual attributes from the data.

### 2.4 Related Work

Historically, anomaly detection was solely utilized to find and remove anomalies from the data, thus ensuring good quality of the data before feeding it to the models. However, over time, the interest grew for investigating the anomalies themselves, what they represent, and what caused them. Recently, anomaly detection has been extensively used and developed in a wide range of applications. Some fields include fraud detection in credit card and insurance industries, intrusion detection in the cyber-security industry, and as is the case for this thesis; fault detection in industrial analytics.

As this thesis is comparative, this section will highlight some successful applications from research with regards to the individual methods under investigation.

In a book series paper written by Eskin et al. 2002, a geometric framework for unsupervised anomaly detection for intrusion detection is presented. Anomalies are detected through finding which points lie within sparse regions of the feature space. The authors use three algorithms; cluster-based estimation, k-NN, and OCSVM. The cluster-based algorithm simply counts the number of points “near” each point in the feature space. The k-NN method was based on what they referred to as the k-NN score, which was determined by the sum of the distances to the k nearest neighbors of the data point. The OCSVM found anomalies by mapping the data to a new feature space and separating small regions with high data density. The experiments used two data sets; a set of network connection records (KDD Cup 1999 dataset) and a set of system call traces (Basic Security Module data portion of the
1999 DARPA Intrusion Detection Evaluation data). To evaluate their systems, the authors used two performance indicators: the detection rate and the false positive rate. Further, they plot ROC (Receiver Operating Characteristic) curves to visualize the relationship and trade-offs between the performance measurements. For the system call data, each of the algorithms performed perfectly for certain thresholds. For the network connections, the algorithms again perform similarly. However, the authors found that they were able to discover some type of attacks well while they were unable to detect others.

OCSVM is further investigated and enhanced in a conference paper written by Amer, Goldstein, and Abdennadher 2013. Their goal is to compute an anomaly score such that a larger score corresponds to significantly outlying points. The further a point lies from the decision boundary, the more likely it is to be an anomaly. In contrast to the binary label assigned by standard OCSVMs, this method allows a ranking of the outliers. However, the authors found that the outliers themselves were the main contributors to the shape of the decision boundary. To overcome this problem, two approaches are proposed to make the decision boundary less dependent on outliers: robust OCSVMs, and eta OCSVMs. The robust variant mainly modified the slack variables by making them proportional to the distance to the cluster centroid. Thus, points distant from the center have large slack variables and are dropped from the minimization objective. The eta approach uses an explicit outlier suppression mechanism achieved by introducing a variable $\eta$, which represents an estimate that a point is normal. This variable controls the portion of slack variables that are going to contribute to the minimization objective. In addition, a variable $\beta$ is introduced to control the maximum number of points that are allowed to be outlying. Data sets from the UCI ML repository were used for evaluation. In the experiments, the authors compare their proposed OCSVM approaches against standard nearest-neighbor clustering and statistical-based unsupervised anomaly detection algorithms. Additionally, they compare the standard OCSVM against their proposed improvements of the method. The area under the ROC curve (AUC) is used as a performance measure where the outlier threshold is varied. Further, they study the number of support vectors as they affect the computational time. The authors conclude by claiming that the proposed SVM based algorithm is well suited for unsupervised anomaly detection problems. In two out of four data sets, their algorithms even proved to be superior. When comparing the OCSVM algorithms with each other, the eta variant seems to be most promising and performs best with regards to AUC.

Inspired by the success of deep learning in other domains, Yan and Yu 2015 explore how unsupervised deep representation feature learning for anomaly detection
can benefit prognostics and health management (PHM) applications in general, and combustor anomaly detection applications in particular. They mean that early detection of abnormal behaviors and incipient faults is critical in ensuring gas turbines operate efficiently and in preventing costly unplanned maintenance. Further, they claim that the performance (detection rate) of anomaly detection solutions fielded is inadequate. In comparison to the traditional handcrafted knowledge-based rules for feature engineering (based on domain and engineering knowledge), they believe that knowledge-augmented data-driven approaches have the potential to not only be less laborious in their designing and development of their rules but also perform better for classification. Self-taught unsupervised deep learning approaches to feature learning can discover sophisticated underlying structures in the data. As such, the authors adopt an unsupervised feature learning scheme based on stacked denoising auto-encoders (SDAE). The SDAE uses denoising autoencoder (DAE), a variant of autoencoder (AE), as its shallow learning blocks. The denoising part of the AE works as a regularization by corrupting input during training. This way, the extracted features will constitute a better high-level representation. The features learned from the SDAE are then taken as the input to a separate neural network (NN) classifier called extreme learning machine (ELM), for anomaly detection. As the connections between input and hidden neurons in ELM are randomly generated and fixed, training the network becomes finding connections between hidden and output neurons. That is a linear least-squares problem whose solution can be directly generated by the generalized inverse of the hidden layer output matrix, which makes the training very fast. The data set used for demonstration purpose consists of several months per-minute sampled data from one turbine. The data is highly imbalanced between normal and abnormal classes. To demonstrate the effectiveness of their model, they compare the classification performance between using the learned features by the SDAE and by using knowledge-driven handcrafted features. The classifier is identical for the tests. To visualize the result, the authors used ROC curves as the classification performance measure for comparison and used cross-validation to obtain robust results. They found that the learned features were significantly better in representing the data than the handcrafted ones, thus performing better for classification.

From research, it was found that a lot of work had been made in the field of anomaly detection. However, most studies focused on intrusion detection applications. Less attention has been put towards fault detection for industrial applications and more specifically for CHP plant’s boilers based on waste incineration. In addition, most studies that do focus on fault detection for industrial applications use artificial data sets crafted for demonstration purposes. Consequently, it will be interesting to evaluate the models against real-world data.
3 Methodology

3.1 Data preprocessing

A complete description of the used dataset is provided in subsection 2.2.

As the data features vary in magnitude and variance, a lot of effort was made in coding the data parameters to optimize the data representation. Initially, a substantial amount of work was put into analyzing, visualizing and understanding the data. The large number of sensors available lead to an enormous amount of data. Some data is redundant or has little impact on system health, which makes it difficult to deal with.

To start with, a challenge lay in selecting features relevant for the specific boiler this thesis aims to model. As the boiler belongs to a plant connected to an even larger system, it is difficult to draw a line with respect to what sensors should be used to predict the boiler’ health. Since the boilers in the plant are connected, what happens in one boiler might affect another. Does this mean that the model depicting one boiler should include data provided by other boilers? While this question is relevant to investigate further, it remains unanswered for this thesis. The sensors selected for the models are more or less directly related to the boiler investigated.

The next step lay in reviewing the data quality. Sensors with a majority of low-quality sensor readings were completely removed. Similarly, sensors missing a lot of information were removed. However, for sensors missing a relatively low amount of values, the null values were replaced with the sensor values’ mean.

The preprocessed dataset was reduced to 1457 features representing the various sensor signals.

Despite being fed with relevant as well as relatively high-quality sensors, the models will still perform poorly. This is due to the model’s dependence on value scaling, which varies a lot between the sensors. Without normalizing the data, the models set relative importance to features depending on the magnitude of their values. To solve this problem, the features are normalized through scaling and standardizing each feature individually using a min-max scaler. The scaler subtracts the minimum value in the feature and then divides by the range (the difference between the maximum and minimum). The resulting sensor values all end up within a range of zero and one. However, this transformation does not affect the shape of the underlying distributions. Further, it does not reduce the importance of outliers in the data. Many
machine learning algorithms perform better and/or converge faster when features are on a relatively similar scale (Mohamad and Usman 2013).

To provide a fair evaluation of the models, they must be tested on previously unseen data. This way, the models’ generalizability is taken into account, essentially it is made certain that the models are not overfitting on the training data. To simulate a productionalization of the models, they are first trained over a period lasting ten months and then tested on the following four-month period. The sampling frequency is minute wise. The data from the testing period is provided in small batches, imitating a real production setting.

As stated in subsection 2.2, the total sample size (number of rows in dataset) is more than 600 000, specifically 610 560. Out of these, 437 760 samples belong to the training period, while 172 800 samples are in the test period.

For the models to be semi-supervised, they need to know what samples represent ”normal” behavior and train on those. These samples were identified through analyzing the plant operation fault logs from Stockholm Exergi’s internal maintenance and deviation report system Maximo. There are many types of faults reported in the system. However, as the ultimate goal of the anomaly detection models is to prevent unplanned maintenance of the system, the only relevant faults are those related to such events (e.g. interruption or failure). Fittingly, the fault logs contain this kind of information. Hereafter, such events are referred to as incidents. Intuitively, historical periods ”close” to any incident are viewed as non-normal. What is considered ”close” is non-trivial to decide. How long an incident lasts and how fast it is reported varies, the worst case being that it occurred on a Friday and was not reported until next Monday. With this in mind, the limit on what is considered close to an incident was set to three days. Consequently, samples further away than three days from any incidents are considered normal.

### 3.2 Models

The models are semi-supervised, and the aim is for the models to learn how the system normally operates. Should a sample of sensor signals differ significantly from normal behavior, the model will flag the sample as an anomaly. Should the difference remain or continue to grow, the model will raise an alarm as the behavior most likely is not a result of some noise but implies that something is acting strangely in the system.
Whether or not a sample belongs to the normal behavior is determined by some kind of similarity measure between the samples referred to as anomaly score. While the calculation of the anomaly score differs between the models, it is also the only thing that differs between the baseline models. However, one thing in common between the models’ anomaly score is that they are all non-binary. This means that the value does not simply tell us if the sample represents an anomaly or not, but returns a continuous value describing the degree in which the samples are anomalous. Consequently, the models can rank the generated anomalies depending on their likeliness to represent some critical problem in the system.

By setting a threshold on what level of anomaly score is considered to translate to an anomalous sample, the generated anomalies can be retrieved. Depending on what level is set for the threshold, the model will catch more or less alarms. There is a trade-off between the detection rate and false alarm rate. A suitable threshold can be found by using the average as well as the variance for the anomaly score on the normal training data. A sensitivity parameter is added to the threshold calculation to control the model’s sensitivity.

Once the sample anomalies have been isolated, another threshold is applied to control what anomalies generate an alarm. This threshold is related to how the anomalies are changing over time and samples. If an anomaly is occurring by itself with normal samples before and after, it probably indicates some noise and is treated as such. As a result, alarms are only generated if anomalies are occurring in successive order over time (i.e. collective anomalies).

To perform maintenance, it must first be determined what part of the boiler the alarm originated from. Hence, an analysis is made of the average individual feature contribution towards the anomaly score throughout the alarm. The individual feature contribution is compared to the average feature contribution, which yields an estimate on what components (identified by their sensors) were most likely to cause the system fault.

Three baseline models are constructed; k-Nearest Neighbor (k-NN), One-class Support Vector Machine (OCSVM), and Auto-encoder (AE). The best performing baseline algorithm will be further examined until it satisfactorily can be put into production. It should detect faults in the system prior to any critical break downs. This way, Stockholm Exergi can plan their maintenance and act accordingly. Optimally, there should be a couple of days between the identification of a fault and breakdown of a system component. Should they need to stop production for repairs, more time available means higher flexibility in choosing a time of low capacity, which is most
beneficial.

### 3.2.1 k-Nearest Neighbor

The logic behind the k-NN model for anomaly detection is based on the assumption that normal data points occur around a dense neighborhood and anomalies are far away. k-NN is often called a lazy learning technique and is used to classify data based on similarities in some distance metrics. The model for this task will use euclidean distance as a measure for data point similarity. Neighbors-based models use instance-based learning (or non-generalizing learning), meaning that it does not attempt to construct a general internal model, but simply stores instances of the training data.

In practice, the anomaly version of k-NN calculates all the distances from every single data sample to all samples stored in the model and calculates the mean of the distance to the k nearest points. This mean of distances used as the anomaly score for each sample. Intuitively, a low anomaly score means that the data sample lies in close vicinity to the data points in the model and therefore should not be considered to be an anomaly. By contrast, a high anomaly score translates to a data sample that is located in a more disperse region and should therefore be considered an anomaly. A threshold is set for the anomaly score, and data samples that have a higher anomaly score than the set threshold are considered to be anomalies.
Algorithm 1 k-Nearest Neighbor based anomaly detection algorithm

INPUT: Normal data set \(X\), Anomalous dataset \(A\), Number of neighbors \(k\), sensitivity \(s\).

OUTPUT: Mean distance to \(k\) nearest points (anomaly score) \(\overline{A_{k-dist}}\), anomalies.

Create model using \(k\)

\[ \phi_k \leftarrow \text{train model using the normal dataset } X \]

Calculate all euclidean distances between samples in \(X\) and the samples stored in \(\phi_k\) \((X)\):

\[ N = \text{len}(X) \]

\[ \text{for } i = 1 \text{ to } N \text{ do} \]

\[ \text{sample distances } = [ ] \]

\[ \text{for } j = 1 \text{ to } N \text{ do} \]

\[ \text{sample distances}[j] + \sqrt{\sum_{l=1}^{N} (X_{i,l} - X_{j,l})^2} \]

\[ \text{end for} \]

\[ X_{k-dist} \text{ is set to top } k \text{ from sample distances} \]

\[ \overline{X_{k-dist}} = \frac{1}{k} \sum_{j=1}^{k} (X_{k-dist}) \]

\[ \text{end for} \]

\[ \mu = E(\overline{X_{k-dist}}), \quad \sigma = Var(\overline{X_{k-dist}}) \]

\[ \text{anomaly threshold } = \mu + s \times \sigma \]

\[ \text{for } i = 1 \text{ to } N \text{ do} \]

\[ \text{sample distances } = [ ] \]

\[ \text{for } j = 1 \text{ to } N \text{ do} \]

\[ \text{sample distances}[j] + \sqrt{\sum_{l=1}^{N} (A_{i,l} - X_{j,l})^2} \]

\[ \text{end for} \]

\[ A_{k-dist} \text{ is set to top } k \text{ from sample distances} \]

\[ \overline{A_{k-dist}} = \frac{1}{k} \sum_{j=1}^{k} (A_{k-dist}) \]

\[ \text{if } \overline{A_{k-dist}} > \text{anomaly threshold } \text{ then} \]

\[ \overline{A_{k-dist}} \text{ is an anomaly} \]

\[ \text{end if} \]

\[ \text{end for} \]

\[ \text{end for} \]

3.2.2 One-class Support Vector Machine

OCSVMs uses a transformation function defined by a kernel to project the data into a higher-dimensional space and tries to learn a decision boundary (hyperplane) with maximum separation between the points and the origin. Once trained, the anomaly
score is given by the signed distance to the separating hyperplane. Points far away from the hyperplane on the positive side are considered to be anomalies.

**Algorithm 2** One-class Support Vector Machine based anomaly detection algorithm

**INPUT:** Normal data set $X$, Anomalous dataset $A$, sensitivity $s$.

**OUTPUT:** Signed distance to hyperplane (anomaly score) $A_{dist}$, anomalies.

Create model using hyperparameters

$\phi_\theta \leftarrow$ train model using the normal dataset $X$, separating hyperplane is obtained

for $i = 1$ to $N$ do

Calculate distance to hyperplane $X_{dist_i}$ by applying the same transformation function on the point $X_i$ and projecting it onto a support vector.

end for

$\mu = E(X_{dist})$, $\sigma = Var(X_{dist})$

anomaly threshold $= \mu + s \times \sigma$

for $i = 1$ to $N$ do

Calculate distance to hyperplane $A_{dist_i}$ by applying the same transformation function on the point $A_i$ and projecting it onto a support vector.

if $A_{dist_i} >$ anomaly threshold then

$A_i$ is an anomaly

end if

end for

### 3.2.3 Auto-encoder

The idea behind using AE for anomaly detection is to compress the sensor readings to a lower-dimensional representation, which captures the correlations and interactions between the various variables. By training the AE on data representing the systems normal behavior, it learns how to reconstruct such signals. As a result, when the model is fed with anomalous data, it will output an increased reconstruction error. This reconstruction error is the anomaly score. A threshold is set for the reconstruction error. When the error is greater than the threshold, the sample is considered to be an anomaly.
Algorithm 3 Autoencoder based anomaly detection algorithm

**INPUT:** Normal data set $X$, Anomalous dataset $x^i$  $i = 1, ..., N$, sensitivity $s$

**OUTPUT:** Reconstruction error (anomaly score) $||x - \hat{x}||$, anomalies

Create model $\phi_\theta$ with parameters $\theta$

$\phi_\theta \leftarrow$ train the model using the normal dataset $X$

$X_{ReErr} = ||X - \phi_\theta(X)||$, $\mu = E(X_{ReErr})$, $\sigma = Var(X_{ReErr})$

anomaly threshold $= \mu + s * \sigma$

for $i = 1$ to $N$ do

$x_{ReErr}(i) = ||x^i - \phi_\theta(x^i)||$

if $x_{ReErr}(i) >$ anomaly threshold then

$x^i$ is an anomaly

end if

end for

3.3 Performance metrics

While the receiver operating characteristic (ROC) curve is the most popular measure of classifier quality in general, Precision-Recall (PR) curve is more suited for the task of anomaly detection as the dataset usually is imbalanced (Davis and Goadrich 2006). This imbalance is caused by a high amount of negative class (normal samples) instances in relation to the amount of positive ones (anomalies). As the ROC curve uses the false positive rate as one of the metrics, its result can be misleading since the sheer number of true negatives will outweigh the number of false positives.

$$False Positive Rate = \frac{False Positives}{False Positives + True Negatives}$$

Consequently, the false positive rate will be small even though the number of false positives is relatively high in comparison to the number of true positives. Hence, the ROC curve would imply that the model is favorable while it is performing poorly.

PR curves, on the other hand, uses (as its name suggests) precision and recall as evaluation metrics. Precision is given by the ratio between true positives and false
positives, which yields a better picture of the models’ performances.

\[
\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{7}
\]

Similarly, recall (or sensitivity) is given by the ratio between true positives and false negatives.

\[
\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{8}
\]

![Figure 13: Precision and recall. Image adapted from Precision and recall 2019](image)

Essentially, the difference between ROC and PR curves is that PR curves are not interested in the true negatives. It is only concerned with the correct prediction of the minority class representing anomalies. For this thesis, the main focus will be just that — accurate prediction of anomalies. Consequently, the PR curve was chosen over ROC curves as a performance metric.

The PR curve plot is created by using precision as y-axis and recall as x-axis for different threshold values on the model. A good model is characterized by PR curves that bow toward the upper right corner of the plot ([1.0, 1.0]). To obtain a composite score that summarizes the curve appearance into one variable, some methods include
average precision score (APS) and area under the precision-recall curve (AUPRC).
APS summarizes the weighted increase in precision with each change in recall for the
thresholds in the PR curve while AUPRC summarizes the integral or an approxima-
tion of the area under the PR curve.

Figure 14: Precision-Recall curves. In this example, the model yielding the green curve is
favored over the blue. $AUPRC_{green} > AUPRC_{blue}$
4 Results

This section presents the results obtained from the baseline models and the more sophisticated version of the best performing model. Results from both the training period and the testing period are provided. Finally, a summary is provided to give an overview of the models’ performances.

To evaluate the models, their output is compared to the expected output, and each sample is classified as being either a false negative, true negative, true positive, or false positive depending on the outcome. For anomaly detection, a false negative means that an anomalous sample is incorrectly classified as being normal, true negative means that a non-anomalous sample is correctly classified as being normal, true positive means that an anomalous sample is correctly classified as an anomaly, false positive means that a non-anomalous sample is incorrectly classified as an anomaly. The classifications of the samples depend on the set threshold for the anomaly score, meaning that each threshold will result in a combination of precision and recall. Through varying the threshold, lots of combinations are created. These values are plotted to create a PR curve that yields a picture of the model’s performance independent of a chosen level for the threshold. In addition, AUPRC is used to provide a composite score for the models.

4.1 k-Nearest Neighbor

The only hyperparameter available for training is the value for $k$. Seeing as this is a baseline version of the model, only a moderate amount of effort was put into choosing a good value for that parameter. The optimal value for $k$ depends on the data. However, a rule of thumb is to choose $k$ to be the square root of the number of training points, which is used for this model (Duda, Hart, and Stork 2012).

Each sample generates an anomaly score based on the mean euclidean distance to the $k$ nearest points stored in the model. A threshold is required to decide what level of anomaly score a sample needs to be considered as an anomaly. A decent threshold can be found by reviewing the anomaly score for the normal training data.
Figure 15: k-NN anomaly score on normal training period data. Using a threshold of 7 seems suitable for this model as it would lead to most of the normal training samples being classified as normal.

Using the mean and variance from the normal data anomaly score in combination with some user-defined sensitivity variable, a threshold is obtained. To visualize the model performance on the training period - which includes both the normal training data and the previously unseen abnormal data within the period - the trained model is fed with it, and the threshold is applied.
In Figure 16, the blue line represents the anomaly score (mean euclidean distance to the k nearest points). Intuitively, (in general) the anomaly score seems to increase for expected anomalous samples. The black horizontal line represents the threshold, and the red dots on the blue line above the threshold means that the sample is considered as an anomaly. However, alarms are not generated for every anomaly. If the anomaly continues in successively order over time, it is more likely that it does not correspond to some noise, and thus, an alarm is raised for the associated samples. Alarms can be seen in the figure as vertical faded red bars. The faded yellow bars represent the true anomaly periods in the data. An overlap between the two represents a successful alarm while standalone yellow bars represent a missed anomaly and a standalone red bar represents a false alarm.

With a high threshold, the model achieves a high precision as most generated alarms will correspond to real anomalies. However, the recall will be low as the model is not catching a lot of alarms. By lowering the threshold, the precision will decrease while the recall is increased. Through varying the threshold, a PR curve is generated, which helps us evaluate the model performance on the training period independent on the set threshold.
The area under the precision-recall curve (AUPRC) value is used as a composite score to determine the model’s performance. The PR curve in Figure 17 achieved an AUPRC value of 0.948.

While the results up to this point yield insight into the model’s performance, they do not show us how the model generalizes as most of the data in the training period was used to train the model. To test the generalizability, the model is evaluated against a hold-out test period previously unseen by the model.
Figure 18: k-NN result on test period data

Figure 18 is interpreted the same way as Figure 16. Again, the threshold is varied to generate a PR curve.
The PR curve in Figure 19 achieved an AUPRC value of 0.611.

### 4.2 One-class Support Vector Machine

The hyperparameter settings available for the OCSVM model include our choice of the kernel as well as the parameters $\nu$ and $\gamma$. As for the kernel, the model uses a radial basis function (RBF). This decision was mostly based on the kernel’s popularity and proven track record for anomaly detection tasks (Fugate and Gattiker 2003). The first parameter $\nu$ corresponds to the fraction of expected outliers of the training data and was set to $\nu = 0.1$. The second parameter $\gamma$ controls the influence of each training sample, with low values lowering the individual influence. Lowering the sample influence translates to a smoothing of the contour lines, which more or less means an increase in the model’s generalizability. The behavior of the model is very sensitive to the gamma parameter (RBF SVM parameters — scikit-learn)
0.21.1 documentation (2019)). As $\gamma$ is related to the samples, it is logical to choose a value related to the number of samples in our training set. Therefore, a $\gamma$ value of $\gamma = \frac{1}{\text{Number of samples in training set}}$ was chosen for the model.

As with k-NN, a suitable threshold for the anomaly score can be found by looking at the anomaly score for normal training period data samples.

![Histogram of anomaly score on training data](image)

Figure 20: OCSVM anomaly score on normal training period data

The threshold obtained with Figure 20 is used to visualize the model performance on the training period.
Figure 21 is interpreted the same way as Figure 16. The blue line represents the anomaly score (signed distance to separating hyperplane), the black horizontal line represents the threshold, and the red dots on the blue line above the threshold means that the sample is considered as an anomaly. Generated alarms are seen in the figure as vertical faded red bars. The faded yellow bars represent the true anomaly periods in the data.

By varying the threshold, the precision and recall are altered, and a PR curve is generated.
The PR curve in Figure 22 achieved an AUPRC value of 0.481. Despite experiments with different hyperparameter settings, the model was not able to predict the system’s faults satisfactorily. Therefore, the model is discarded.

4.3 Auto-encoder

An AE can be set up in lots of different ways. While some structure is necessary (e.g. output layer must be the same size as input layer), most settings are optional. To start with, the model’s architecture must be decided upon. The chosen model uses three fully connected layers, one hidden layer for the encoding and decoding side each, and the middle layer that holds the coded data representation. The number of nodes in the layers is reduced until the middle layer is reached and is then increased in the same order. This architecture forces each layer to capture important patterns
in the previous layer. By reducing the dimension of the data, it is ensured that the model does not learn the identity function, which has zero reconstruction error but does not capture the structure of the data-generating distribution. The input data size is 1457, and the following layer sizes were set to 200 and 100, respectively. Consequently, the architecture of the model is $1457 \rightarrow 200 \rightarrow 100 \rightarrow 200 \rightarrow 1457$. The activation functions were all chosen to be an exponential linear unit (ELU). Further, the mean squared error was used as the loss function and adam as the optimizer.

Again, to find a suitable threshold for the anomaly score, an analysis is made of the value for the normal training period data samples.

![Histogram of Reconstruction error on "normal" training data](image)

**Figure 23**: AE anomaly score on normal training period data

The threshold obtained with Figure 23 is used to visualize the model performance on the training period.
Figure 24: AE result on training period data

Figure 24 is interpreted the same way as Figure 16. The blue line represents the anomaly score (model reconstruction error), the black horizontal line represents the threshold, and the red dots on the blue line above the threshold means that the sample is considered as an anomaly. Generated alarms are seen in the figure as vertical faded red bars. The faded yellow bars represent the true anomaly periods in the data.

By varying the threshold, the precision and recall are altered, and a PR curve is generated.
The PR curve in Figure 25 achieved an AUPRC value of 0.951.

To evaluate the model’s generalizability, a hold-out test period previously unseen by the model is used.
Figure 26: AE result on test period data

Figure 26 is interpreted in the same way as Figure 24. The threshold is varied to generate a PR curve.
The PR curve in Figure 27 achieved an AUPRC value of 0.637.

4.4 Improved version of AE

Seeing as the baseline AE model performed best with regards to the AUPRC value on both the training and test data periods, it was selected as the model to go forward with. Apart from hyperparameter adjustments, the major improvements were the introduction of noise in training data, a deeper architecture, and online retaining (or active learning) of the model.

DAE was implemented through corrupting the training data with Gaussian noise, using a variance of 10% of the sample values. After adding more hidden layers to the model, the final architecture of the model was $1457 \rightarrow 400 \rightarrow 200 \rightarrow 100 \rightarrow 200 \rightarrow 400 \rightarrow 1457$. In contrast to the baseline version of AE, the model is iteratively
retrained during the test period, simulating the use of active learning in production. After predicting anomalies for a small batch of samples, the model is trained on the normal samples in an equally large batch of samples from three days earlier. By then, any incidents should have been reported, thus separating normal samples from abnormal. Hence, while the model is still predicting previously unseen data, it is better able to cope with a change in the definition of normal data (e.g. caused by drifting data mean). Another change in the model is the use of the Rectified Linear Unit (RELU) as the activation function.

The thresholds and PR-curves are retrieved in the same manner as in subsection 4.3.

![Histogram of Reconstruction error on "normal" training data](image)

Figure 28: Improved AE anomaly score on normal training period data
Figure 29: Improved AE result on training period data
Figure 30: Improved AE Precision-Recall curve on training period data

The PR curve in Figure 30 achieved an AUPRC value of 0.987.
Figure 31: Improved AE result on test period data
Figure 32: Improved AE Precision-Recall curve on test period data

The PR curve in Figure 32 achieved an AUPRC value of 0.801.
4.5 Summary

<table>
<thead>
<tr>
<th>Period</th>
<th>Model</th>
<th>AUPRC</th>
<th>Training time</th>
<th>Evaluation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>k-NN</td>
<td>0.948</td>
<td>1m 34s</td>
<td>17h 45min 56s</td>
</tr>
<tr>
<td></td>
<td>OCSVM</td>
<td>0.481</td>
<td>8m 42s</td>
<td>1d 21h 14m</td>
</tr>
<tr>
<td></td>
<td>AE</td>
<td>0.951</td>
<td>3h 4m 15s</td>
<td>28s</td>
</tr>
<tr>
<td></td>
<td>Improved AE</td>
<td>0.987</td>
<td>3h 27m 40s</td>
<td>1m 7s</td>
</tr>
<tr>
<td>Testing</td>
<td>k-NN</td>
<td>0.611</td>
<td>-</td>
<td>15h 22min 14s</td>
</tr>
<tr>
<td></td>
<td>OCSVM</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>AE</td>
<td>0.637</td>
<td>-</td>
<td>13s</td>
</tr>
<tr>
<td></td>
<td>Improved AE</td>
<td>0.801</td>
<td>11h 19m 16s</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: The models AUPRC, training- and evaluation time over training and testing period
Figure 33: PR-curves over training and test period

55
This thesis aimed to find to what extent machine learning can be used to predict anomalies in a CHP plant boiler, and which model is best suited for the task. To answer this question, an evaluation was made of three baseline state-of-the-art ML models; k-Nearest Neighbor, One-Class Support Vector Machine, and Auto-encoder. To determine which model performed best, the area under the precision-recall curve (AUPRC) was used as the performance measure. Further and more detailed insight into the models' performances is provided by their respective generated anomaly scores over time as well as by the shape of their precision-recall curve on both the training- and test period. The results reveal that the AE (subsection 4.3) is the best performing baseline model as it has the highest AUPRC on both the training- and test period (see Table 1). As such, the second part of the research question has been satisfied and answered.

By further reviewing the results, it is noticeable that k-NN performs almost as well as the AE for both the training- and test period. Although, it can also be seen that k-NN is very slow at testing time. This is explained by the logic behind the model, where each testing point has to be compared to each training point. In other words, the computational complexity is $O(nd)$, where $n$ is the number of training points, and $d$ is the dimensionality of the data. Nonetheless, k-NN was implemented to work as a benchmark, or sanity check if you will. In contrast, the AE is much slower at training time but very fast for testing. Once the model parameters have been tuned, calculating the anomaly score for a sample is done in a matter of milliseconds. This is optimal for an anomaly detection system in production since the predictions need to be calculated online to discover faults as soon as they appear whereas training only needs to be performed once (unless the model is retrained). Hence, it would be cumbersome for a system to perform online predictions using a k-NN model. OCSVM seems to work similarly, whereas the evaluation time is even longer than that of k-NN.

From a model point of view, the comparison might not have been entirely fair despite (and due) to my willingness to compare them equally. Towards this end, the same training and test datasets were used to evaluate the models. In reality, the models benefit differently from feature extraction techniques and preprocessing methods. Hence, a better comparison might have been achieved by individualizing these steps for each model. This might have been a reason why the OCSVM (subsection 4.2) underperformed. Another reason might be that the dataset was too complex (due
to the many features) for the model to be able to create a good decision boundary. Additionally, the OCSVM is sensitive to contamination in the training set.

The improved, deeper, denoised, and retrained version of AE was able to detect and generate one alarm each for all four larger incidents that occurred during the testing period as seen in Figure 31. Each change of the model made it better. However, retraining using active learning had the largest impact. This is not surprising as a slight data drift can be seen in both Figure 18 and Figure 26. The deeper structure also had a large impact. However, the increased model complexity meant that denoising was required for the model to not find unwanted patterns in the noise. In comparison to the baseline version, the performance increased for both the training- and testing period. The model achieved an AUPRC of 0.987 and 0.801 on the training- and test respectively (see Table 1). These results imply that ML can be successfully be used to predict anomalies in the system, thus satisfying the first part of the research question.

Each incident in the test data lasted for six days, and the anomaly was detected through knowledge-based methods in the middle of those periods. Therefore, the constructed ML model would have been helpful in all cases where it was able to detect and generate an alarm for the anomalies before the middle of the incident periods. For example, it was clearly useful for incident number 3 and 4 of the test period (see Figure 31). After reviewing the results with domain experts, they concluded that the tool has the potential to be useful in practice as long as it can identify the incidents prior to the knowledge-based method and that there is some way to connect the anomalies with a certain part of the system. This way, they can act on the information and prevent the incident from escalating to unplanned maintenance. Not only was the model able to identify some incidents sooner than the knowledge-based method, but it was also able to hint at which component caused the incident through identifying sensors with a high contribution towards the anomaly score. This suggests that the model can have a practical use for the organization.

It is somewhat difficult to consider the effect of the performance metric choice as solely PR curves were used. The benefit with the PR curve is that it is only concerned with the correct prediction of the anomalies. In contrast, the ROC curve takes the true negatives into account, which can be misleading for anomaly detection tasks due to the imbalance between the normal and abnormal class. Had ROC curves been used instead, the result might have favored models that are performing worse with regards to the classification of anomaly samples. Then again, it would have been interesting to confirm this. Overall, I believe that the PR curves yielded a fair insight into the models’ performances.
Although these results are relatively good, it is fair to question why they are not better. One problem lies in the data. While the sensors used generally are of good quality and covers most of the system, it is far-fetched to claim that they describe the entire reality which determines the system’s health. Another problem lies in the assumption regarding anomalous periods. The information contained in the log report system is limited, erroneous, and scarce. Thus, generated alarms that do not coincide with expected anomalous periods (false positives) might actually, in some cases, represent true positives. For example, see the last alarm in Figure 31. Additionally, as parts of expected anomalous periods might correspond to normal operational periods, some false negatives could, in reality, be true negatives. For example, see the second alarm in Figure 31.

Apart from data-related issues, the model can be further improved through exhaustive experimentation with different architectures and hyperparameter settings. Indeed, by no means does this result represent an upper boundary of ML performance in the task of anomaly detection in CHP plant boilers.
6 Conclusions

Accurate detection of boiler abnormalities is essential in reducing unplanned maintenance of a CHP plant based on waste incineration. Fielded traditional rule-based anomaly detection solutions are performing inadequately. To improve the detection performance, this thesis aims to leverage state-of-the-art machine learning solutions. While ML has a long proven track record for other domains, little research has been made for industrial applications and more specifically for CHP plant boilers.

Out of the baseline models, the Autoencoder performed best in predicting faults for the boiler. The k-Nearest Neighbor also performed well, whereas the One-class Support Vector Machine was unable to reach satisfactory results. The improvements of the Autoencoder through making it deeper, introducing noise in training data, and using online retraining further increased the model’s performance.

The results suggest that ML can successfully be utilized to detect faults at an earlier stage and potentially circumvent otherwise costly unplanned maintenance.
7 Further studies

To start with, it would be interesting to put the suggested model into production and evaluate its performance over time. Moreover, additional model tuning and more thorough feature engineering are required to harvest the true potential of the model. To add more complexity to the model, an alternative to the deep autoencoder structure is that of stacked AE (SAE / SDAE if denoised), using shallow AEs as building blocks. In an SAE, the output of one AE is connected to the input of another. The deeper the network, the more vulnerable it becomes to randomness. By instead training several AEs greedily and individually, the result may be a more reliable model.

In addition to improving the investigated ML models, it would also be interesting to explore others. ML is a quickly developing field, and there are many state-of-the-art models that can be used. During the research studies, some that were touched upon (to name a few) were Isolation Forest (Goix n.d.), Local Outlier Factor (Goix n.d.; Amer, Goldstein, and Abdennadher 2013; Chandola, Banerjee, and Kumar 2009), Self-Organizing Maps (Chandola, Banerjee, and Kumar 2009), K-means Clustering (Chandola, Banerjee, and Kumar 2009; Yan and Yu 2015; Vercruyssen et al. 2018), Expectation Maximization (Chandola, Banerjee, and Kumar 2009), and Long Short-Term Memory networks (Malhotra et al. 2016). Additionally, several models can be used together in an ensemble method, much like the algorithm proposed by Yan and Yu 2015.

Lastly, it would also be interesting to apply the investigated models on other datasets. Through using different datasets with varying characteristics, the research is generalized. Interesting aspects are the degree of outliers, number of features, feature types, size, and patterns.
References


Phyu, Thair Nu (2009). “Survey of Classification Techniques in Data Mining”. In: *Hong Kong*, p. 5.


## Appendix

### A Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHM</td>
<td>Prognostics and Health Management</td>
</tr>
<tr>
<td>CHP</td>
<td>Combined Heat and Power</td>
</tr>
<tr>
<td>CFB</td>
<td>Circulating Fluidized Bed</td>
</tr>
<tr>
<td>DCS</td>
<td>Distributed Control System</td>
</tr>
<tr>
<td>SCADA</td>
<td>Supervisory Control And Data Acquisition</td>
</tr>
<tr>
<td>PLC</td>
<td>Programmable Logic Controllers</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional–Integral–Derivative controllers</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>k-NN</td>
<td>k-Nearest Neighbor</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>OCSVM</td>
<td>One-Class Support Vector Machine</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>MLP</td>
<td>Multilayer Perceptron</td>
</tr>
<tr>
<td>DNN</td>
<td>Deep Neural Network</td>
</tr>
<tr>
<td>AE</td>
<td>Auto-Encoder</td>
</tr>
<tr>
<td>DAE</td>
<td>Denoising Auto-Encoder</td>
</tr>
<tr>
<td>SAE</td>
<td>Stacked Auto-Encoder</td>
</tr>
<tr>
<td>SDAE</td>
<td>Stacked Denoising Auto-Encoder</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Function</td>
</tr>
<tr>
<td>ELU</td>
<td>Exponential Linear Unit</td>
</tr>
<tr>
<td>ROC</td>
<td>Receiver Operating Characteristic</td>
</tr>
<tr>
<td>PR</td>
<td>Precision-Recall curve</td>
</tr>
<tr>
<td>AUC</td>
<td>Area Under the Curve</td>
</tr>
<tr>
<td>AUPRC</td>
<td>Area Under the Precision-Recall Curve</td>
</tr>
<tr>
<td>APS</td>
<td>Average Precision Score</td>
</tr>
</tbody>
</table>