CHALMERS
UNIVERSITY OF TECHNOLOGY

Machine Learning Implementation at Mölndal Energi AB

Evaluating the Possibilities of Implementing Machine Learning within the Production System at Mölndal Energi AB

Master's thesis in System, Control and Mechatronics

LINA BRINK

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Machine Learning Implementation at Mölndal Energi AB

Evaluating the Possibilities of Implementing Machine Learning within the Production System at Mölndal Energi AB

LINA BRINK

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2026

Machine Learning Implementation at Mölndal Energi AB
Evaluating the Possibilities of Implementing Machine Learning within the Production System at Mölndal Energi AB
LINA BRINK

Supervisor: Filip Angwald, Mölndal Energi AB
Examiner: Petter Falkman, Electrical Engineering

Master's Thesis 2026
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Picture displaying Mölndal Energi AB.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Machine Learning Implementation at Mölndal Energi AB
Evaluating the Possibilities of Implementing Machine Learning within the Production System at Mölndal Energi AB
LINA BRINK
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis aimed to evaluate the possibilities of implementing machine learning within the production system at Mölndal Energi. The outcome of the thesis included an overview of potential applications and recommendations for implementation, along with two proof of concepts demonstrating simplified versions of potential applications. The aim was to support Mölndal Energi in improving certain areas of their production system and give the company a foundation for further development and increased competitiveness.

The literature review revealed predictive maintenance, demand forecasting, scheduling and energy storage as relevant applications. It was concluded that predictive maintenance would be the most relevant to implement due to the high potential of improvement. Mölndal Energi already use, both directly and indirectly through Göteborg Energi, external machine learning applications for both demand forecasting and scheduling. There are several other areas of interest that could be investigated further in future work, such as the usage of digital twins to simulate and debug systems, as well as inspection and fault detection using drones and augmented reality.

Two proof of concepts were developed using the programming language Python in a Jupyter Notebook environment. In the first proof of concept, a machine learning model was developed to predict the maximum load of a boiler based on operational data. It used an LGBM algorithm for quantile regression and the results were evaluated by comparing the predicted output with actual output for a number of random samples, which showed reasonable results. Yet, further testing would be required before deployment. A second proof of concept was developed to identify deviations within fuel data. Three regressor algorithms were compared (LGBM, XGB and RF) and results were evaluated using tables and plots of the deviations. Once again results seemed reasonable, however further testing would be required before operational usage. In future work, both proof of concepts could be improved by comparing more algorithms and fine-tuning the parameters further. The anomaly detection model could also be applied to similar areas by changing the dataset.

Keywords: Artificial Intelligence, Machine Learning, Optimization, Energy sector, Mölndal Energi AB.

Acknowledgements

I would like to express my deepest gratitude to Petter Falkman and Filip Angwald for all the guidance and encouragement I have recieved throughout this thesis. I am also grateful to all of the people at Mölndal Energi who have contributed with their experience and invaluable advice, as well as many others who have contributed with their knowledge through interviews and meetings. No one mentioned, no one forgotten.

Lina Brink, Gothenburg, May 2026

List of Acronyms

Below is a list of acronyms that have been used throughout this thesis listed in alphabetical order:

ABS	Absorption Refrigerator
ACM	Advanced Collection Manager
AI	Artificial Intelligence
ANN	Artificial Neural Networks
B1	Boiler 1
B2	Boiler 2
B3	Boiler 3
BFUS	Business for Utilities Suite
CHPP	Combined Heat and Power Plant
CNN	Convolutional Neural Network
FN	False Negative
FP	False Positive
GB	Gradient Boosting
LGBM	Light Gradient Boosting Machine
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
RF	Random Forest
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SDG	Sustainable Development Goals
SGD	Stochastic Gradient Descent
SVM	Support Vector Machines
TN	True Negative
TP	True Positive
XGB	eXtreme Gradient Boosting

Contents

List of Figures

List of Tables

1
Introduction

Based on a project proposal from Mölndal Energi AB emphasizing the need to evaluate potential applications of machine learning within the company, this thesis aimed to assess the possibility of using machine learning to optimize production at Mölndal Energi AB.

1.1 Background

The use of Artificial Intelligence (AI) and Machine Learning (ML) is rapidly increasing in all areas of society, from enhanced algorithms in social media to intelligent systems in factories. More and more companies discover how ML, a sub-category of AI, can be a useful tool, contributing to increased efficiency and accelerated development. To maintain competitiveness it is crucial for companies to stay updated on this technology.

Mölndal Energi AB is an energy distributor, providing district heating and cooling, as well as electrical power, to households and the public sector in Mölndal, Sweden. The energy is mainly produced at Riskulla cogeneration plant, and the company aims to be modern and sustainable. To meet future demands on capacity, flexibility and sustainability the company recently started a production optimization program called OptiPro. As part of the program, the company seeks to investigate how ML could contribute to more efficient operations. However, implementing new technologies such as ML tend to be challenging due to the complexity of the technology. While the range of possible applications is wide, understanding which algorithms are suitable for a specific situation can be difficult without relevant experience in the area. Identifying what data is required, along with the limitations of each approach, is essential for successful implementation.

1.2 Purpose

The purpose of this thesis is to evaluate the possibilities of implementing ML in order to optimize the production system at Mölndal Energi AB. The outcome of the thesis includes an overview of common applications within the energy sector and similar industries and recommended strategies for implementation, along with two proof of concepts demonstrating simplified versions of potential applications. The goal is to support Mölndal Energi AB in improving relevant areas of their production, such

as predictive maintenance, forecasting demand and scheduling production units. It also provides a foundation for further development and increased competitiveness.

1.3 Objective

The purpose of the thesis can be summarized into three main questions.

1. What potential applications of machine learning can be identified within the production system at Mölndal Energi AB, based on available data and operational needs?
2. What aspects of implementation do Mölndal Energi AB need to consider in order to implement machine learning in a safe and structured way?
3. How can selected applications be designed and implemented as simple proof of concepts?

1.4 Scope

The thesis focuses on evaluating the possibility of using ML at Mölndal Energi AB. It will be limited to applications within district heating and district cooling that can be developed using existing data or with little additional data. The focus will be on production, however a few applications related to distribution might be included. Two proof of concepts will be developed to demonstrate the usability of two selected solutions, but will not be adapted for full-scale deployment. Only minimal user interfaces will be created for demonstration purposes since the primary focus will be on the functionality and integration of the model.

1.5 Societal, ethical and ecological aspects

To ensure that the impact of the thesis is positive, a good objective is to align with relevant UN Sustainable Development Goals (SDGs). There are 17 goals in total, aimed at different problems; from ending poverty to creating sustainable cities and communities.

1.5.1 Societal aspects

The primary intention of the thesis is to support Mölndal Energi AB in improving the efficiency and reliability of the district heating and cooling production. This could potentially lead to lowering production costs, reducing resource consumption and enhancing energy security, which would contribute to SDG 7: Affordable and Clean Energy. The thesis could also lead the way for other companies in the industry to implement similar solutions, potentially amplifying its positive impact.

However, AI technologies can be misused or misinterpreted if applied carelessly, which could lead to a negative impact on society. Therefore, all methods and results should be presented transparently, with clear explanations of intended use and

potential risks. Sharing sensitive information or techniques that could be used in harmful applications should be avoided.

1.5.2 Ethical aspects

The primary ethical risk lies not in the thesis work itself, but in the potential misuse of its results and use with harmful intent. To minimize this risk, as described earlier, all sensitive information must be carefully reviewed and anonymized before being shared. This reduce the risk of data being exploited in ways that may lead to unintended or harmful consequences.

Another ethical risk stems from the "black-box" nature of many ML algorithms, particularly within deep learning. When a model lacks transparency, it can develop hidden biases or generate incorrect outputs without developers or users realizing it. To mitigate these risks, input data should be thoroughly examined to ensure that it is fairly distributed and representative. In addition, the output of the model must be critically reviewed to verify that it aligns with the expectations.

1.5.3 Ecological aspects

Mölndal Energi AB already aims at being a sustainable energy distributor, with a high share of renewable energy resources. Improving production efficiency can further reduce emissions, lower waste and enable better use of existing resources, contributing to SDG 13: Climate action. As with the societal aspects, this can serve as an example for other companies, increasing the positive ecological effect.

However, while increased production efficiency can reduce environmental impact, improved production capacity could also lead to a higher demand on resources. For instance, if biomass usage increases, it is essential that the deforestation do not exceed forest regrowth, or the long-term sustainability will be compromised. The thesis will therefore focus on solutions that increase output without increasing resource consumption.

2
Theory

An overview of fundamental machine learning concepts is presented to facilitate a better understanding of the results of the thesis.

2.1 Algorithms

Machine learning can be applied to a wide range of tasks, some of the most common displayed in Figure 2.1 [7].

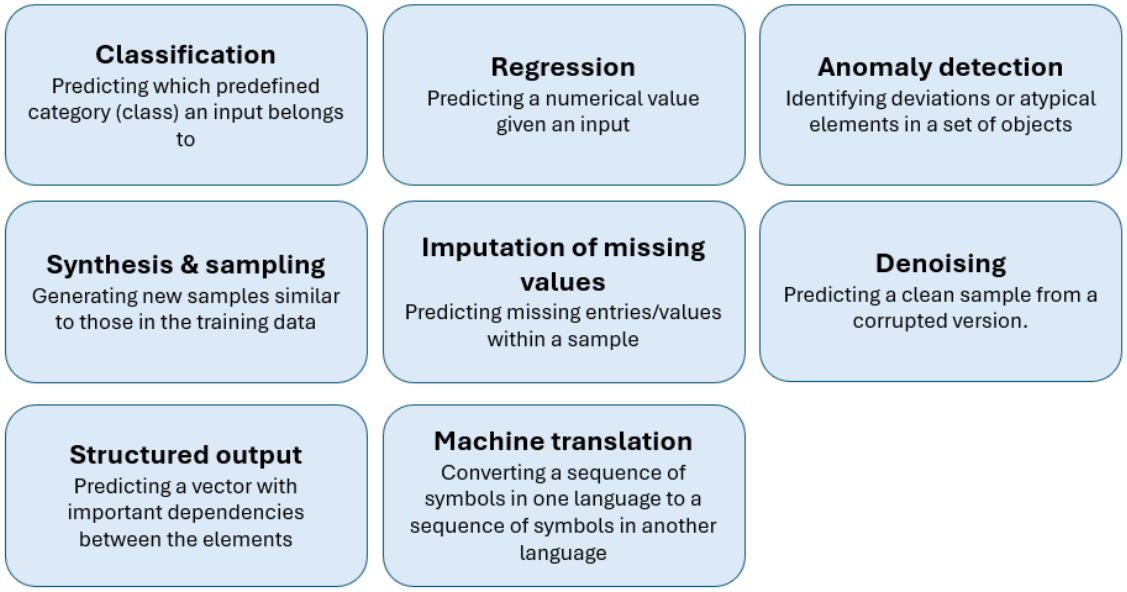


Figure 2.1: Common applications for machine learning.

Due to the many applications, it is reasonable to divide ML algorithms into different subcategories. Onwusinkwue et al. [17] discuss how ML algorithms can be divided into supervised, unsupervised and reinforcement learning. When choosing an ML algorithm, factors such as data characteristics and operational requirements must be considered.

2.1.1 Supervised Learning

Goodfellow et al. [7] explain that in supervised learning the algorithm is provided labelled data, meaning both the input x and the output y are defined. By training

the model on the labelled data, the aim is to teach the model how to predict the output y_{new} given unseen but similar values of the input x_{new}. The output could either be a class or a number. For instance, Support Vector Machines (SVM) are driven by a linear function to output a class identity based on the input. Other common supervised learning algorithms include linear regression and decision trees.

GeeksforGeeks [6] define linear regression as a method which constructs a relationship between two variables, with the target of computing the mean value of the response variable. In other words, the algorithm learns a function which, based on some input data, produce an output which is similar to the original output data. Another type of regression is quantile regression, which aims to estimate the quantile (percentile) value of the response value. This can be visualised by Figure 2.2. For instance, if the chosen percentage, called the alpha value, is set to 10%, the algorithm should predict a number which 10% of the output will be below. Similarily, if the alpha value is set to 90%, then 90% of the output values should be below the predicted number and only 10% above it.

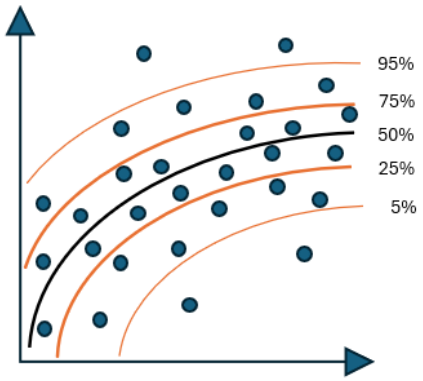


Figure 2.2: Graph showing the concept of quantile regression.

Decision trees are described by Goodfellow et al. [7] as a set of nodes associated with regions of the input space. Each region corresponds to one node and has its own parameters. Internal nodes split their region into subregions, one for each child node. Typically, each node maps all input points within its region to the same output. GeeksforGeeks [6] present Random Forest (RF) as a common algorithm that utilize decision trees to make predictions. It consists of multiple trees, each one looking at a different random part of the data and making predictions separately from the other trees. The algorithm can be used for both classification and regression. In classification the predicted class is typically chosen based on majority voting, while in regression the results are averaged to produce a final number.

Another common algorithm which uses decision trees, presented by GeeksforGeeks [6], is Gradient Boosting (GB). It combines a set of less accurate models to create one accurate model. Each model, which consists of multiple trees, is trained to correct errors made by previous models. This is done by computing the gradient of a loss function and training the new model to predict this gradient. A key

feature is shrinkage, which means the predictions are multiplied with the learning rate to scale the contribution of the model, which can reduce overfitting. Once all trees are trained, predictions are made by summing the contributions of all the trees.

GeeksforGeeks [6] further describe that eXtreme Gradient Boosting (XGB) is an optimized version of GB, particularly useful for regression. It uses GPU acceleration which makes it efficient on large datasets and it can handle missing data without imputations, however it is computationally intensive, consumes memory and requires parameter tuning. LightGBM (LGBM) is another optimized version of GB, also designed to handle large datasets. By supporting parallel and GPU training, it is optimized for memory efficiency and fast training. LGBM also allows quantile regression, making it useful for risk management and handling uncertainty, however it can sometimes be slower to train and more difficult to interpret.

2.1.2 Unsupervised Learning

In unsupervised learning, the labels are not given to the model, instead the algorithm must find correlations on its own. Clustering is presented by GeeksforGeeks [6] as a common subcategory of unsupervised learning. It works by grouping data into clusters of similar data points, and include algorithms such as k-means clustering and hierarchical clustering. Two other common subcategories are association rule learning and dimensionality reduction.

2.1.3 Reinforcement Learning

Balouji et al. [2] present reinforcement learning as another subcategory of ML. In reinforcement learning, the algorithm is rewarded or punished for different actions. The algorithm strives to maximize cumulative rewards and find an optimal solution by refining its decision-making policies through continuous interaction. It does not need labelled data, but require a well-defined cost function and rewards to learn what actions are encouraged. While reinforcement learning offers adaptability and dynamic decision-making, Hamdan et al. [8] note associated challenges include high computational requirements, long training times, and a need for well-defined reward structures.

2.1.4 Deep Learning & Neural Networks

Another subcategory of ML, discussed by Bello et al. [3], is deep learning and neural networks, also referred to as Artificial Neural Networks (ANNs). These algorithms excel at handling large volumes of complex and unstructured data, such as images, time-series data and sensor readings. Balouji et al. [2] explain how these algorithms consist of multiple layers of interconnected nodes, each layer in the network developed to find a certain type of features. A simplified version of an ANN can be viewed in Figure 2.3. By combining the features of each layer, the algorithm can classify and recognize advanced patterns without prior information about the features.

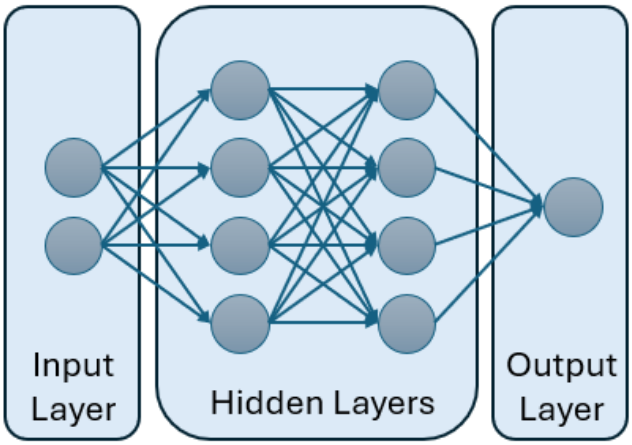


Figure 2.3: Simplified graph displaying an ANN consisting of two input nodes, two hidden layers with four nodes each, and an output layer with one node.

Goodfellow et al. [7] present a few common deep learning architectures, including deep feedforward neural networks, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Deep feedforward neural networks, also called Multi-Layer Perceptrons (MLPs), aims to approximate a function by mapping a set of input values to a set of output values using multiple simple functions in layers. CNNs use convolution, which is a special kind of linear operation, instead of general matrix multiplication in at least one of the layers. RNNs are specialized for processing a sequence of values by using parameter sharing across parts of the model. This makes it possible to extend and apply the model to examples of different forms and generalize across them. Gated RNNs, such as Long Short-Term Memory (LSTM), use gating mechanisms to stabilize gradients over time, allowing them to capture long-term dependencies in time series [7].

2.2 Model Development

The quality of an ML model highly depends on the data quality, model architecture and execution, as discussed by Mogren et al. [12]. If the implementation is poorly executed, this may result in incorrect outcomes. However, due to the nature of ML, the agent might still sound certain of the result, making errors difficult to discover. Therefore, it is of high importance to carefully evaluate both model and associated data, and to verify predictions to ensure model reliability.

2.2.1 Pre-processing

As mentioned earlier, ML models require high quality data, processed to fit the chosen algorithm. Wästberg et al. [32] discuss a few general rules, such as data being representative and standardized. This includes data having the same format throughout the entire dataset, and labels being unique and coherent. All relevant data types should be accounted for, with enough data of each type.

Mogren et al. [12] discuss how data should be processed when used for classification. In datasets used for classifications, all outcomes (classes) should be represented, often with a fair distribution to avoid biases. For instance, to detect faults using classification, there need to be data representing both faulty equipment and functioning equipment. If there is a bias in the dataset, which often is the case in datasets concerning breakdowns and faults, the data may be processed to make it more fairly distributed. This can be done through downsampling, meaning the data of the over-represented class is decreased. This is often done randomly to keep the distribution similar to the original distribution. However, downsampling may cause a loss of information in the data. To avoid this, one strategy is to downsample multiple times and use the average of the results. Another solution is to simulate data and create more sample data for the minor class.

In addition to ensuring data has a fair and representative distribution, Balouji et al. [2] explain that data is often split into two or three smaller datasets. These datasets are used for training, evaluation and possibly testing of the model. The evaluation dataset, which is often smaller than the training dataset, is used for checking that the model is not overfitted. Overfitting means it has learnt the dataset too well and is not generalizing enough when presented a new dataset, as shown in Figure 2.4. In other words, an overfitted model would perform well on the training dataset but not on the evaluation dataset.

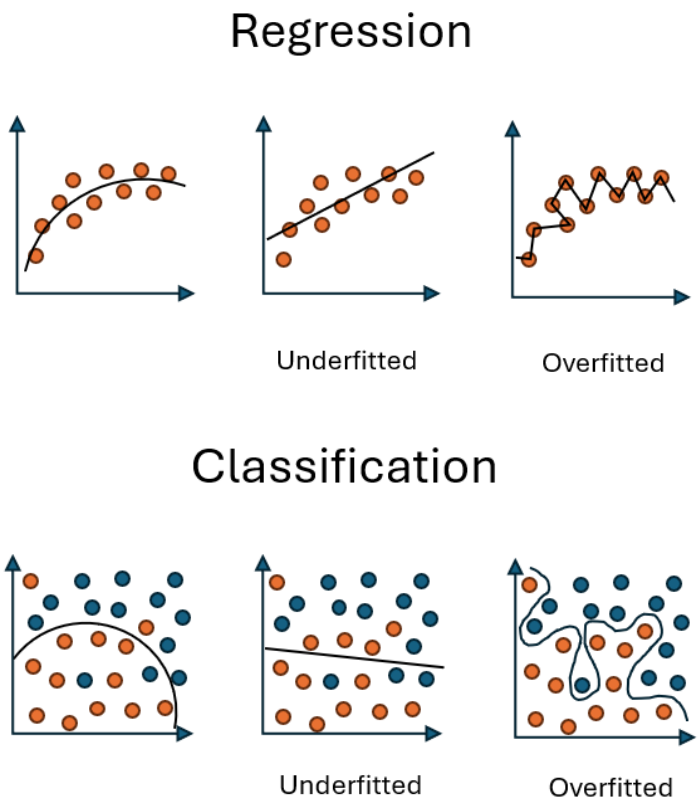


Figure 2.4: Graphs showing the concept of overfitting and underfitting in regression and classification.

2.2.2 Training

Typically, to build and train an ML model a dataset is needed, as well as an algorithm, a cost function and an optimization procedure, as discussed by Goodfellow et al. [7]. The choice of algorithm is based on the structure of the data and the purpose of the model. A commonly used cost function is the Maximum Likelihood, which uses cross-entropy between the training data and the model predictions. There are many different optimization algorithms, such as stochastic gradient descent (SGD), momentum, AdaGrad, RMSProp and Adam. Some algorithms require special-case optimizers, but a good start is often to use SGD or Adam.

Goodfellow et al. [7] further explain that in feedforward networks, activation functions are used in each hidden layer to compute the hidden layer values. The default recommendation is to use the rectified linear unit, defined by the activation function g(z) = max0, z. In the output layer, common output units are linear units, sigmoid units or softmax units. There are also parameters that control the algorithms behaviour called hyperparameters. Goodfellow et al. [7] note that the choice of hyperparameters depends on many factors, such as the relationship between parameters, training error, generalization error and computational resources. To fine-tune hyperparameters, common techniques include grid search or random search.

A final technique used in the development of ML models, described by GeeksforGeeks [6], is transfer learning. By training a model on a task with sufficient data, it could later be repurposed for another similar task where the available data is limited. This reduces the risk of overfitting.

These are just a few examples of the many strategies within ML model development and training.

2.2.3 Evaluation

How well an ML algorithm performs on unseen data determines how well it will work when deployed in the real world. To evaluate a classification model, GeeksforGeeks [6] present a popular strategy called the confusion matrix, displayed by Table 2.1. The matrix compares predictions made by the model with actual results by dividing it into four categories: true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

Table 2.1: Confusion matrix.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

These values can in turn be used to calculate other metrics such as accuracy, precision and recall. Accuracy (see Eq. 2.1) shows how many predictions were correct out of all predictions, precision (see Eq. 2.2) shows how many positive predictions were

actually positive, and recall (see Eq. 2.3) shows how many of the actual positives were correctly predicted.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.3}$$

To evaluate a regression model, GeeksforGeeks [6] presents the Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) as popular options. These metrics measure the difference between the predicted and actual values, with MAE being the average absolute difference (see Eq. 2.4), MSE being the average squared difference (see Eq. 2.5) and RMSE being the square root of the MSE (see Eq. 2.6).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.4}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.5}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.6}$$

Another common metric, especially for quantile regression, is called pinball loss (also known as quantile loss). Vermorel [31] explains that pinball loss estimates the accuracy of a quantile model and can be used to compare quantile models, where the model with the lowest pinball loss would be the most accurate. The function is defined by Eq. 2.7, where τ is the target quantile, y is the real value and z is the quantile prediction.

$$\text{pinball loss} = \begin{cases} (y - z)\tau & \text{if } y \geq z \\ (z - y)(1 - \tau) & \text{if } y < z \end{cases} \tag{2.7}$$

Although these metrics are common, Mbiydzenyuy et al. [11] points out that the symmetric nature of some metrics might cause problems when used for forecasting energy demand since they penalise overestimations and underestimations equally. From an energy provider's perspective there would be a significant difference between the two errors. Overestimation of the demand might lead to energy waste, but underestimation makes the network fail at providing sufficient heat to the customers and might cause collapse, which is not acceptable and must be taken into account.

2.3 Challenges

Löfblad et al. [10] observe that many Swedish energy companies are still in the early stages of AI integration, including the integration of ML. Although numerous projects and initiatives are underway, few have reached full-scale implementation. Danish [4] discuss the challenges of implementing AI, dividing them into four categories, as shown in Figure 2.5.

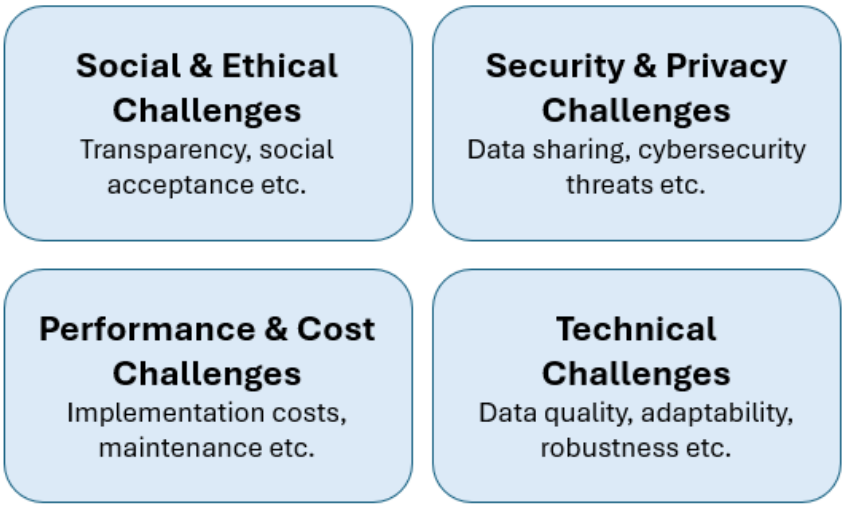


Figure 2.5: Four categories of challenges associated with implementing AI.

Although many of these challenges apply to all companies implementing ML, there is still no standardized approach to integration. As a result, each company must determine how to apply ML within their specific industry.

2.3.1 Social and Ethical Challenges

To elaborate on the issue of transparency, Löfblad et al. [9] highlight that many AI algorithms are so called "black boxes", making it difficult or even impossible for developers and users to fully understand how these algorithms operate. The authors also address concerns about AI replacing human labour, aggravating societal acceptance. However, they argue, such fears are often overstated and AI could also enable employees to take on more meaningful and engaging work by automating monotonous tasks.

Yamamoto et al. [34] notes that it can also be hard for personnel to set proper expectations on AI system due to a lack of knowledge about the abilities and limitations of the techniques. Meanwhile, data scientists developing the models might lack a proper understanding of the reality and the operations in which the model will be used. This complicates the integration.

2.3.2 Security and Privacy Challenges

Löfblad et al. [9] address concerns related to security and integrity. Increased connectivity and growing use of shared datasets heighten the vulnerability to cybersecurity threats, contributing to public hesitation toward adopting AI and sharing personal data. Yet it can be argued that these risks stem not from the AI techniques, but from insufficient cybersecurity measures. Consequently, exploring these technologies are still valid, provided security risks are recognized and managed with caution.

2.3.3 Performance and Cost Challenges

Onwusinkwue et al. [17] stress the importance of reliable data in achieving accurate predictions. This often require investments in sensors, data and infrastructure. Integrating AI into existing infrastructure and ensuring compatibility can be a complex and expensive process since many legacy systems were not originally designed for AI applications. These investments can pose financial challenges and should be minimized.

Changes within systems are common, especially when the systems are used for long periods of time. In an interview, Participant 5 [22] highlights the importance of developing models which can be updated and improved to account for future changes. This requires a process defining when the model needs to be updated and how to perform the update. One option is to allow the model to be updated automatically, however a human operator might still be needed to control the updated model before it is applied to any operational applications. The cost and time required to implement these solutions and maintaining them over time must therefore also be taken into account.

2.3.4 Technical Challenges

Using data for applications it was not originally designed for has its challenges. As mentioned by Participant 5 [22], one challenge is to account for changes over time. Within the energy sector, acquiring a full season would require a year of data, and predicting seasonal trends requires multiple years. During this time, the system might have changed, errors might have occurred and data might have been destroyed due to faults in software.

Another challenge presented by Participant 5 [22] is synchronization of data sources. Although having multiple data sources might improve the model performance, it can also decrease the data quality. To account for as many features as possible, it is common to use data from different sources. For instance, external weather data might be used together with internal production data. However, the format and temporal resolution of these sources might differ, making it difficult to find continuous data over long periods of time.

3
Methodology

The thesis consists of two main parts; a literature review and proof of concepts. The literature review provided a theoretical foundation for identifying relevant ML applications. The proof of concepts demonstrated practical examples of how selected applications could be implemented. There were also interviews to gain further information on certain topics.

3.1 Literature Review

The literature review was conducted to identify potential areas of interest and existing implementations of ML within the energy sector and related industries. The review initially explored the range of possible applications, followed by a deeper examination of technical requirements and limitations.

The first step of the literature review was to define appropriate keywords for searching information. To maximize the scope of findings, searches were performed in both English and Swedish. Initial keywords included *machine learning*, *energy sector*, *optimization* and *predictive maintenance*. As the review progressed, additional keywords were introduced to further investigate certain areas of interest identified during the review, such as *anomaly detection* and *demand forecasting*.

Sources were primarily gathered from Google Scholar and company websites. Throughout the process, the credibility of each source was critically evaluated to minimize the risk of incorporating inaccurate or biased information. For instance, the number of citations on Google Scholar was taken into account. The collected material included scientific articles, industry reports, technical documentation and other relevant publications.

All relevant literature was collected, organized and categorized to simplify usage and support efficient analysis in later phases of the thesis. Key findings were highlighted and summarized for each selected article, with proper citations to ensure traceability. These were eventually integrated into the final report.

3.2 Interviews

In addition to the literature review, interviews were held with experts to gain further information. These will be referred to as *Participant 1*, *Participant 2* and so on.

The interviews were performed after a basic understanding had been established, to gain clarifications of uncertainties, fill gaps in knowledge and provide practical perspectives on implementation challenges.

In total nine interviews were held. Seven of these were internal, with people working at Mölndal Energi AB, while the other two were with external experts. The majority of interviews were held online, using Teams, while the rest were held in person at Mölndal Energi AB. All interviews were recorded and transcribed. The anonymised transcripts can be provided by request.

A set of questions were prepared for each interview, aimed to clarify certain aspects relevant to the thesis. General opening questions were used to get a broad understanding of the topic before more targeted questions were asked about specific areas of interests. The findings were summarized and analysed before being anonymised and included in the report.

3.3 Proof of concept

Two proof of concepts were developed to demonstrate how selected applications could be implemented in practice. This showed feasibility and helped identify potential challenges that were not discovered in the literature review.

First, necessary data was collected and pre-processed. Pre-processing included exploration of data, selection of relevant variables and reductions required to prepare the dataset for the chosen method. The data was checked for gaps and lacking information, removing NaNs and controlling the format, to keep the datasets as consistent and contionuous as possible. For each model, the data was split into two sets, a training set and a testing/evaluation set.

Models were developed in a Jupyter Notebook environment to ensure transparency and readability. The code was written in Python due to it being a popular and fairly intuitive programming language, and detailed comments and markdown descriptions were included to support transparency, understanding and debugging. Candidate algorithms were evaluated according to their compatibility with the available data, technical requirements and their potential to generate meaningful results. One or multiple algorithms were chosen and parameters for each algorithm were fine-tuned using the scikit-learn function RandomizedSearchCV.

Once the models had been trained, systematic testing and evaluation was performed to verify that the code executed correctly and that outputs aligned with expected behaviour. Plots and tables were used to visualize the results and to facilitate analysis of the predictions.

4

Literature Review

The literature review is divided into three sections: information about Mölndal Energi AB, identified potential applications and recommendations for implementation.

4.1 Möndal Energi AB

Mölndal Energi AB is a Swedish energy distributor in Mölndal, providing district heating and cooling, as well as electrical power. An overview of the production system is presented to facilitate a better understanding of the results of the thesis. The data infrastructure is also discussed, as well as collaborations with Göteborg Energi.

4.1.1 Production

In an interview, Participant 3 [20] explained that Mölndal Energi AB has two boilers on-site to generate district heat, as well as a Combined Heat and Power Plant (CHPP) which generate both heat and power. There are also two off-site oil fuelled spare boilers in Lindome, used only when needed or tested due to their high production costs. Preisz et al. [27] explain that the boilers combust fuel in a furnace to produce heat which is used to heat up water. The water moves in a circulating system around the furnace and is then cooled at heat exchangers where the heat is transferred to the district heating water. The steam from the boiler in the CHPP can be led either through the turbine or a heat exchanger, or one after the other, depending on how much power and heat should be produced.

In the interview, Participant 3 [20] noted that the oldest boiler on-site, called Boiler 1 (B1), was built in 1983 and runs on biofuel, such as wood chips and wood waste. Göteborg Energi contributes financially to the maintenance costs and can in return buy heat generated by it. The second oldest boiler on-site, called Boiler 2 (B2), is oil fuelled and powered by fossil free biofuels. It is regarded as a spare boiler and only used when the other boilers cannot be used due to malfunctions or similar. In the CHPP, both heat and electric power is produced. The boiler in the CHPP, called Boiler 3 (B3), is run on recycled wood and bark and is the main boiler. It uses flue gas condensation, meaning the heat that is left in the flue gas after leaving the furnace is used to heat the return water from the district heating grid.

Regarding the district cooling, Participant 3 [20] notes that it is a fairly recent

expansion for Mölndal Energi AB, with a newly built facility next to Riskulla and one in the renovated underground facilities where the original district heating plant was located. The underground facility is called Berget and include compressor units, while the new facility consists of two compressor units and an Absorption Refrigerator (ABS). There is also an accumulator tank currently being built to store cold [20]. The compressor units work by compressing and condensing a liquid using electricity to generate cold. The ABS uses district heat to boil and condensate a saline solution, requiring less electrical power than the compressor units but using more heat instead [13].

4.1.2 Data Infrastructure

A strong data infrastructure is essential for implementing ML algorithms successfully. Onukwulu et al. [16] highlights collection, storage and processing of data as crucial aspects that the data architecture must be able to perform in a reliable and effective way, in order to make algorithms work optimally. Within Mölndal Energi AB, data is stored and used in multiple ways, as shown in Figure 4.1.

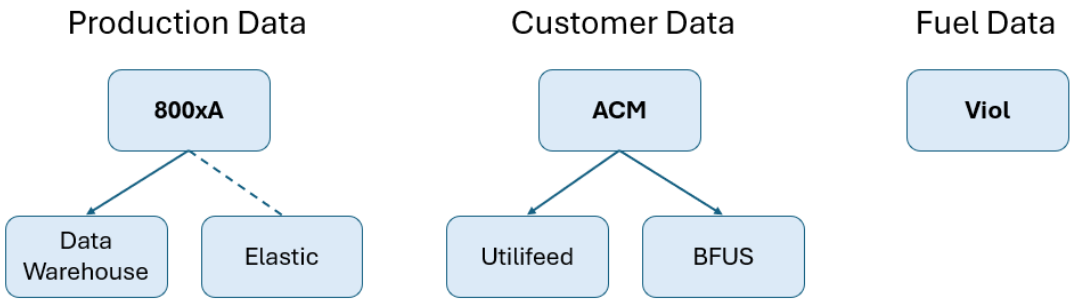


Figure 4.1: The ways in which different types of data is stored at Mölndal Energi AB.

In an interview, Participant 1 and Participant 2 [19] explains that data from the production units is stored as signals, in the form of time series data, in ABB's system 800xA. Currently, the Excel plug-in Smart Client is used for accessing historical data. Certain signals are extracted and sent to the data warehouse, which can be accessed through SQL. However, it is quite slow and hard to work with even for simple applications, which is why there is an ambition to extract all relevant data from 800xA to the system Elastic instead. This would facilitate computations and allow quicker services.

On the customer side, Participant 1 and Participant 2 [19] further explains that the district heat consumption for each customer is measured, including transferred energy, water volume, and temperatures to and from the facilities. Data is collected and stored in a system called Advanced Collection Manager (ACM). ACM sends the data to the AI-powered system Utilifeed, as well as the billing and metering system Business for Utilities Suite (BFUS). BFUS only store data used for billing purposes, which is processed to produce reasonable bills to the customers. This

includes approximations, settlements and final packaging. The data in BFUS also includes a reliability value which can be used as a moderator.

Lastly, fuel data is collected and stored in the system Viol. External data, such as weather data from SMHI, might also be collected for certain applications [19].

4.1.2.1 Utilifeed

Utilifeed [30] provide multiple tools for analysing and optimizing areas within the energy sector. Utilifeed Optimization combines ML and mathematical optimization models to provide forecasts and operational plans. For production they offer an ML algorithm called EnergyPredict which uses digital twins of the production system to model consumption patterns along with total demand. For distribution they optimize water temperature and pipe dimensions, providing continuous monitoring and highlighting substations in need of maintenance. In sales and marketing, Utilifeed Pricing can be seen as an advanced invoicing system. The system includes common district heating price models and uses ML to calculate what each customer would have been billed for each price model given certain circumstances.

Participant 1 [18] explain that Mölndal Energi AB use Utilifeed mainly in the operative work on the customer side. By collecting and analysing data from ACM, Utilifeed learns how each facility works and predicts the total consumption given certain cases based on weather conditions and time. The collected data include the amount of energy a customer receives, the volume of the circulating hot water in the network and what temperature the water has in and out of the system. It also collects weather data and what day of the week it is. Although Utilifeed does not collect data from production facilities at Mölndal Energi AB nor the measurements to the network of Göteborg Energi, Participant 1 [18] suggest these could be interesting additions.

4.1.3 Göteborg Energi

Mölndal Energi AB has an agreement with Göteborg Energi allowing them to transfer a limited amount of heat to or from Mölndal every hour [27]. Participant 3 [20] discuss how waste incineration and refineries in Gothenburg generate more heat than the customers of Göteborg Energi demand, resulting in waste heat. Instead of dissipating the heat into the ocean, Mölndal Energi AB can buy it and use it to decrease their own production. This is especially useful during the summer and leads to economic benefits for both companies.

Göteborg Energi also use Energy Optima 3 to produce prognoses that are shared with Mölndal Energi AB [20]. On their website, Energy Opticon [5] describe Energy Optima 3 as an Energy Management Service and software for forecasts and economic optimization for energy systems. It offers both short-term and long-term production optimization, validation of measurement data, load and price forecasts, maintenance optimization etc. The Forecast module use both statistical and AI methods and include load forecasts, electricity prices, weather forecasts etc.

4.2 Potential Applications

Löfblad et al. [10] highlights that ML can be applied to all parts of the energy sector: production, storage and distribution. In addition to saving time and reducing costs, optimizing these areas could also reduce waste.

4.2.1 Predictive Maintenance

Production relies on equipment and numerous components in need of regular maintenance. Traditionally, preventative maintenance is performed on a fixed schedule, resulting in components being serviced earlier than required or failing unexpectedly before their scheduled maintenance. Löfblad et al. [10] explains that the idea of predictive maintenance is to avoid breakdowns before they occur by predicting the condition of components and systems. This can reduce maintenance costs, increase operating times and decrease the risk of breakdowns. It can also give a better view of the production and how to optimize it.

Predictive maintenance can be divided into multiple subcategories. Bello et al. [3] present two important subcategories called *fault detection and diagnosis* and *remaining useful life estimation*. In fault detection and diagnosis, continuous monitoring of system performance is used to identify anomalies and deviations from normal operating conditions, as well as their root causes. Remaining useful life estimation involves predicting the remaining time before a component or system will require maintenance or replacement.

4.2.1.1 Current Method

Regarding maintenance of the production units, Participant 3 [20] explained that the aim at Mölndal Energi AB is to perform the majority of the maintenance during the summer when the production is stopped and to only perform routine inspections during the operating season. Yet sometimes operational disturbances occur during the operating season, requiring maintenance to be performed earlier. This leads to less time being spent on planning the summer maintenance, which in turn can lower the quality of the maintenance and increase the risk of even more failures during the operating season. Therefore it is important to minimize the need of maintenance during the operating season and to improve the maintenance during the summer period.

Participant 6 [23] divides the maintenance into preventive maintenance and remedial maintenance. In preventive maintenance, actions are based on maintenance plans from the manufacturer and scheduled on a regular basis to prevent failures. Inspections and tests are also used to analyse the remaining life of components. If certain spare parts are hard to find or have long delivery times, these might be changed earlier than necessary to prevent failures. Remedial maintenance on the other hand is based on error reports. Participant 3 [20] points out that there is a list of alarms which is used to check the status of the production facility. When certain measures

of emissions or components differ, operators are notified and associated processes or systems are checked. Whenever an error or fault is detected, Participant 6 [23] explains that it is reported and maintenance scheduled. In case the maintenance cannot be performed without stopping the facility, it is typically scheduled for the summer. Yet the work is prepared as much as possible, to allow maintenance to be performed in case the facility is stopped earlier for some reason.

The time a stop takes depend on the system in need of maintenance. For instance, Participant 3 [20] explain that stopping B3 requires 12 hours of pressure relief, followed by 48 hours of cooling before it can be accessed for maintenance. When starting it, 12 hours is required to increase the pressure. Although the duration of maintenance varies, the total time for a stop is rarely less than 80 hours. In comparison, stopping the turbine takes approximately 30 minutes and starting it can take up to 3 hours. Stopping the district cold production is almost instant.

4.2.1.2 Applications

In an interview, Participant 8 [25] discuss that current alarms acknowledge deviations based on set intervals but might miss oscillations and does not show the root cause of the deviation. Using ML to further analyse signals and find deviations that the alarms might miss could improve the maintenance. For instance, there have been recurrent problems with sintering in B3. Different solutions have been tried but the problem is not completely solved, meaning ML could potentially aid the process.

Another area where ML might improve the maintenance, suggested by Participant 3 [20], is leakages. Due to the high age of the boilers, the risk of leakages of the tube vessels is increasing. These can be difficult to detect at first, making leakages go unnoticed. However, by analysing temperatures and other relevant data these might be detected earlier. Participant 3 [20] also mentions that the filters in the flue gas condenser require regular cleaning, which requires the facility to be stopped. To allow better planning of when to perform this maintenance, the flow and pressure could be analysed, since a low flow or high pressure might indicate dirty filters.

Lastly, there is continuous online vibration monitoring of the turbine and fans, since abnormal speed could indicate something not functioning correctly. Participant 6 [23] suggests similar solutions could be applied to the conveyor belt, for instance monitoring the torque to determine if the amount of fuel is too high.

4.2.1.3 Algorithms

Bello et al. [3] suggests supervised learning algorithms such as SVM and RF could be trained on historical data to recognize operational patterns. When data deviates from these patterns, the model could flag it as a potential fault, allowing operators to act before any actual damage occurs. In addition, Participant 3 et al. [21] suggests classification algorithms could be trained on images to detect and classify what type of fault has occurred.

Onwusinkwue et al. [17] highlights a few common deep learning architectures in predictive maintenance applications, including CNNs, RNNs, MLPs and LSTMs. Hamdan et al. [8] explain that CNNs excel in image recognition and can be used to assess the visual condition of infrastructure and detect visual anomalies on equipment by analysing images from surveillance cameras or drones. Bello et al. [3] add that autoencoders could distinguish between normal operational fluctuations and actual faults, reducing false alarms and improving maintenance efficiency.

4.2.2 Demand Forecasting

In order to meet consumer demand while simultaneously keeping production costs to a minimum, accurate predictions of the demand are crucial. Yet, it is a complex task due to fluctuations and unpredictability. In an interview, Participant 5 [22] discussed how sending heated water through a network can take several hours until it has reached the end of the network. Therefore, it is crucial to predict the demand and send the heated water ahead of time. It is also common to increase the supply temperature when peaks are expected, to gain margins. However, each degree is expensive, so margins should be kept at a minimum.

When forecasting heat load, Salem et al. [28] suggest that weather occurrences are the main factor for short-term forecasting, while long-term forecasting is dominated by economic factors. While the heat demand is mainly affected by outdoor temperature, Participant 8 [25] describe district cooling as more "peaky" due to solar radiation.

4.2.2.1 Current Method

To forecast demand, Participant 8 [25] explains that Mölndal Energi AB use linear regressions on production data with one or multiple parameters. They also use Utilifeed, which provide demand load but not production load, and get forecasts by Optima through Göteborg Energi. These are optimized with the same methodology as Göteborg Energi use on their own network, typically resulting in good estimations.

Participant 8 [25] points out that there are three optimizations simultaneously running: daily, weekly and monthly. Operators are mainly interested in the current day and the next to make decisions, while ordering fuel requires predictions of a couple of weeks. Some aspects of operation may however require longer time considerations, hence the month-based optimization.

4.2.2.2 Algorithms

Aderibigbe et al. [1] compared multiple ML algorithms for forecasting energy demand. It was found that SVM and ANNs were robust in handling high-dimensional data and generalized well from training to unseen data, while LSTM algorithms demonstrated low error rates in load forecasting and were able to capture temporal dependencies in data, making it particularly suitable for time-series forecasting. Salem et al. [28] explain that the use of decision trees in RF and XGB allow these

algorithms to show how various factors contribute to energy demand.

Aderibigbe et al. [1] further discuss a study analysing two different scenarios, one excluding weekends and holidays and the other without exclusions. Several deep network algorithms were used to predict the demand. The study concluded that an MLP showed the least prediction error for the first scenario, while an RNN-based gated recurrent network was the most effective for the second scenario. This highlights the importance of choosing the right algorithm based on the dataset, forecasting horizon and requirements of the energy systems. Furthermore, Nguyen [14] note that researchers have found hybrid methods, which combine multiple approaches, to outperform single-method models.

4.2.3 Scheduling

Determining the optimal use of production units is a complex task. Mbiydzenyuy et al. [11] present long lead times from production to heat being available in the network, which can be multiple hours, as a significant challenge of scheduling. The aim is to minimise operation costs while ensuring heat delivery to customers. Optimal planning of energy production and storage requires predictive control. However, to predict the system performance in advance and adjust operation accordingly, predictions of demand and load (especially short-term) are necessary.

4.2.3.1 Current Method

The heat production at Mölndal Energi AB depends mainly on the boilers and the scheduling of these. Participant 8 [25] explains that the operators monitor and run the boilers and the network continuously, while also planning for the near future. During planning, they must take into account unexpected events and decisions that might have a large economical effect. For instance when stopping a boiler they need to consider every potential scenario in order to avoid unnecessary expenses. Currently, no ML is used for this, however they have regular meetings with Göteborg Energi each week to discuss the scheduling. Participant 3 et al. [21] highlight the need of tools which can improve forecasts and optimize schedules and strategies. These tools should take into account parameters such as economy to determine which strategy is preferred.

Participant 3 [20] discuss that due to the many subsystems, there are multiple production scenarios that can be used depending on the current conditions. During summers, no heat is generated on-site and all the required heat is bought from Göteborg Energi instead. When the waste heat from Göteborg Energi decreases at the end of the summer, Mölndal Energi AB begin by turning on B3. When B3 is run on maximum load, due to outdoor temperature being close to or below zero degrees, B1 might be turned on. Since turning on a boiler is expensive, it is preferable to start the boiler only when it can be run consistently for a period of time. Since B1 and B3 are the cheapest, those are always the first to be used, while B2 and the boilers in Lindome are only used if necessary. There is currently no system being used for storing district heat in Mölndal.

The district cold production scheduling will be changed once the accumulator is finished. Currently, the ABS is used during summers and additional compressor units are used when required. When the accumulator is finished, it could for instance be charged by the compressor units during low price periods and the ABS during high price periods [13]. Although there currently is a lack of data to use and no need to optimize that part of the scheduling yet, it could pose as an interesting application in the future.

4.2.3.2 Algorithms

Typically, in order to predict the output of a system, a reliable model of the network, including pipes, substations, customers and heat exchanging equipment would be required. Due to the complexity of the system, it can be difficult to model the system accurately. To solve this problem, Ntakolia et al. [15] investigated ML techniques as an alternative approach since they do not require detailed knowledge of the infrastructure and can provide control strategies based on available historical data. For instance, they discussed a study in which a genetic algorithm was used to determine the optimal operating schedules of subsystems, resulting in an increase in profit compared to a rule-based, priority order baseline strategy. Another option, proposed by Mbiydzenyuy et al. [11], would be using a reinforcement learning-based approach.

4.2.4 Energy Storage

The distribution network consists of a large system of pipes. To fully comprehend the heat supply capacity, Mbiydzenyuy et al. [11] explain that the amount of energy stored in the system needs to be taken into consideration. By temporarily increasing the temperature in the pipes, the network could be used as an advanced form of thermal energy storage and provide flexibility.

4.2.4.1 Current Method

Mölndal Energi AB are not using energy storage for district heating, but they are currently building an accumulator for storing cold. The accumulator tank will allow cold to be produced when the price is low and distributed when the price is high. When to store cold will be based on expected demand and expected production for the upcoming day, which in turn depends on various factors such as weather, wind, humidity, sunlight, temperature etc. [13].

4.2.4.2 Algorithms

In a study mentioned by Ntakolia et al. [15], ANNs were combined with a genetic algorithm optimizer to develop a model predictive control strategy for storing district cold. The method could effectively predict performance and determine the optimal control strategy of a district cooling system with ice storage. In another study, Stepanovic et al. [29] evaluated usage of reinforcement learning algorithms

due to the potential of dealing with the complex dynamics of the system and the adaptability to various operation scenarios without re-calculation. The developed model successfully approximated the complex dynamics of the network and resulted in fewer constraint violations compared with a nonlinear optimization method. Although the algorithm required a long training time and it proved difficult to ensure robust safety assurances, the study concluded that reinforcement learning was a promising option.

4.3 Recommendations for Implementation

The recommendations for implementation are divided into a suggested framework, advice for integrating ML with existing systems, and important aspects regarding data infrastructure.

4.3.1 Framework

Yamamoto et al. [33] has developed a framework for implementing AI models, consisting of the seven steps shown in Figure 4.2.

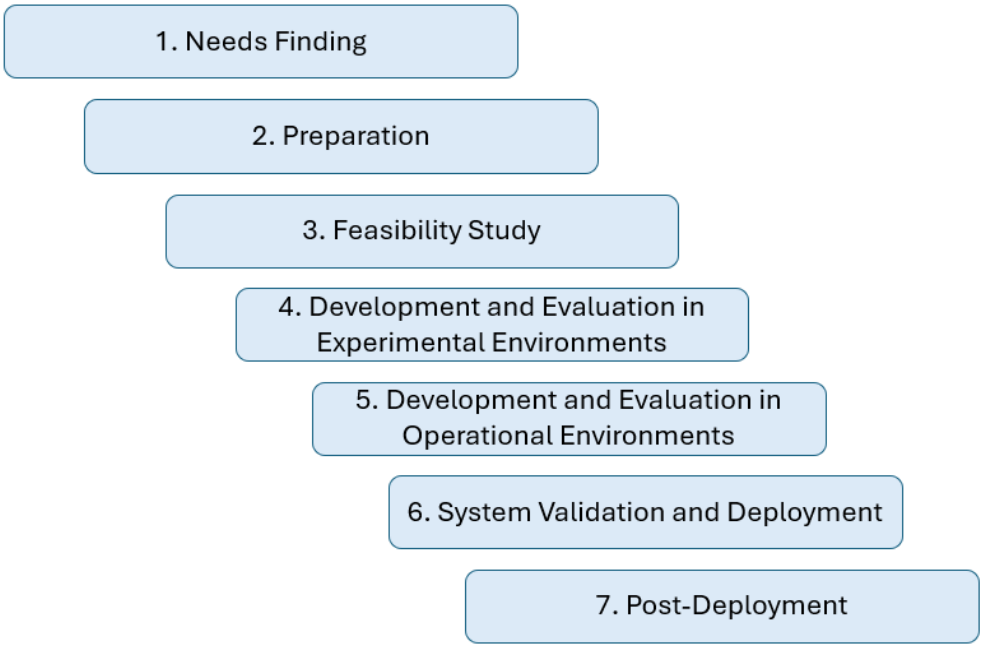


Figure 4.2: Framework developed by Yamamoto et al. [33] for implementing AI models, consisting of seven steps.

First, Yamamoto et al. [33] suggests stakeholders should be educated on relevant techniques to identify needs and use cases. Then, initial requirements should be specified by analysing relevant cases with focus on needs, data infrastructure, AI capabilities etc. Participant 7 [24] add that it is of high importance to involve operators and other end-users early in the development process to ensure developed

solutions meet their requirements. Yamamoto et al. [34] note that it is also important to ensure that operators know the probabilistic behaviour of AI systems since it often differs from more traditional equipment. Further, Yamamoto et al. [33] suggest initial prototypes and solution systems should be built to evaluate whether the use case has enough value for implementation, before the model is integrated with a limited environment. Finally, the model is validated and approved for deployment, but will still require regular maintenance and a plan for improvement. Yamamoto et al. [34] explain that having a symbiosis between operators and the AI system is beneficial. While the system assists operators in making more informed decisions, the operators could improve the performance by providing feedback to the system.

4.3.2 Integration with Existing Systems

To decrease risks associated with the implementation of ML, Onukwulu et al. [16] suggests companies should adopt an incremental approach to integration. By piloting ML applications in specific areas of the supply chain, the benefits of ML tools can be evaluated before they are fully integrated.

Onwusinkwue et al. [17] discuss dynamic optimization algorithms can continuously assess the state of a system and make instantaneous adjustments based on real-time data, such as adjusting operational parameters to account for fluctuations in demand. Ntakolia et al. [15] note that these intelligent control strategies need to be able to adapt to changes in the network status, cope with faults and be robust to stochastic uncertainties, while also being flexible and scalable. However, Participant 5 [22] point out that implementing solutions that monitor a system are often easier compared to operational solutions. Furthermore, it is often better to start with supporting components rather than critical components, since it is associated with less safety risks but can still increase the overall quality. Participant 5 [22] highlights that while many model predictions are not 100% certain, these can still aid the decision-making process and indicate systems in need of inspection.

4.3.3 Data Infrastructure

ML models require access to data, which Balouji et al. [2] suggest can be done either by implementing the models in the same system as where the data is stored, or by exporting data to an external system where the model is implemented. Exporting large amounts of data can prove to be difficult and time-consuming, however implementing models in a system not originally built for it can also be challenging. There is also the option of analysing data "on-the-edge", meaning algorithms are stored on hardware close to the sensor instead of in the cloud.

The amount of available data determines the prediction accuracy but also the required computational power. It is important to find a middle point with enough data but not too much. To save computational power, Participant 5 [22] suggests summertime data can be excluded since no heat is generated in summer. This also reduce the risk of the model generalizing too much, since summer data can pose as

noise. Participant 7 [24] discuss that most facilities within heat production are only operated during winter or on occasion, resulting in a limited amount of continuous operational data and a slow development rate. However since many energy companies are municipally owned and struggle with the same questions and problem, there is a large potential for collaborations. For instance, Mogren et al. [12] implemented a model using data from two different companies within the energy sector, which performed better than a similar model trained on data from only one company.

5

Proof of Concept

During the literature review, multiple areas were suggested for implementing ML within Mölndal Energi AB. Based on these, it was determined to develop two different proof of concepts. The first proof of concept estimates maximum boiler load for a set of parameters, while the second proof of concept detects anomalies in fuel data. The final code for both of these concepts can be found in Appendix B.

5.1 Boiler Load

In an interview, Participant 3 et al. [21] explained that during the winter, when the heat demand is high, the goal is to run the primary boiler on maximum load. The boiler is affected by other subsystems and components, each with their own limitations, and the load is determined by how close these other systems are to their limits. Due to the complexity of the system, it can be difficult to know when the boiler is running optimally or whether it could be increased just by looking at the values of other components. This results in operators running the boiler differently. For instance, one operator might decrease the boiler load to keep a certain safety margin, while another operator might try to push the limits more aggressively. By using ML to optimize the boiler load, Mölndal Energi AB could increase the efficiency and save money, as well as gain a better understanding of the system.

5.1.1 Data

To determine the optimal boiler load at a given moment, relevant paramteres must be defined and limitations of each subsystem should be taken into account. What parameters to use were based on operator experience and determined during the interview with Participant 3 et al. [21]. The chosen input parameters can be viewed in Table 5.1, along with their pre-defined constant limits.

Table 5.1: Chosen parameters and their limits.

Parameter	Description	Lower	Upper
Var1	Rotor	-150	
Var2	Turbine	0	
Var3	Flue gas fan		1340
Var4	Flue gas fan, motor bearing (temperature)		
Var5	Flue gas fan, drive side bearing (temperature)		
Var6-8	Primary flue gas (temperatures)		190
Var9	Flue gas analyzer (H20)	5	40
Var10	Flue gas analyzer (CO)		200
Var11	CO emission		
Var12	Moisture content in fuel		52
Var13	Bed temperature (difference)	-70	100
Var14	Bed temperature (average)	750	830
Out1	Boiler load		80

Since not all systems have defined limits, some limits were instead set to the maximum and minimum values found in the collected data. Minimum, maximum and mean values for each parameter can be found in Table 5.2.

Table 5.2: Minimum, maximum and mean values for each parameter.

Parameter	Description	Min	Max	Mean
Var1	Rotor	-998.5	205.3	11.8
Var2	Turbine	-8.1	23.5	10.9
Var3	Flue gas fan	0.0	1351.1	786.1
Var4	Motor bearing (temperature)	9.2	84.9	62.2
Var5	Drive side bearing (temperature)	9.9	74.7	58.5
Var6	Primary flue gas (temperatures)	11.9	198.2	151.0
Var7	Primary flue gas (temperatures)	12.0	190.7	146.2
Var8	Primary flue gas (temperatures)	11.6	197.5	154.4
Var9	Flue gas analyzer (H20)	0.0	39.8	21.4
Var10	Flue gas analyzer (CO)	$-1.97 \cdot 10^4$	$1.39 \cdot 10^{35}$	$4.95 \cdot 10^{30}$
Var11	CO emission	0.0	6293.1	106.6
Var12	Moisture content in fuel	20.0	80.0	51.1
Var13	Bed temperature (difference)	0.4	1194.0	87.2
Var14	Bed temperature (average)	9.6	974.8	714.9
Out1	Boiler load	11.0	80.0	56.1

In total, five years of operational data was collected from the production system 800xA using the Excel plug-in Smart Client. Since the systems change over time, the data collection was limited to five years to decrease the risk of system changes. Data from the summer period was excluded to improve computational efficiency and make the model focus on the operational period. This resulted in approximately 28000 samples, where each sample represent a time instant and the dataset has a resolution of one hour between samples. This means the samples might be

temporally correlated, however for simplicity the order has not been taken into account during the development of the proof of concept.

After loading the data, it was evaluated and pre-processed. Parameter values were plotted (figures can be found within the code in Appendix B) and parameters with inconsistent behaviour and missing values were removed, as well as rows containing NaNs. After the pre-processing approximately 27800 samples remained, which were randomly split into a training set and a test set. The ratio was 80% for the training set and 20% for the test set, resulting in approximately 22200 samples in the training set and 5600 in the test set.

5.1.2 Model

Supervised algorithms could be used since the data included boiler load, and due to the nature of the problem it was relevant to look at regression algorithms. To predict a probabilistic upper bound (the maximum load), quantile regression was a suitable option and LGBM was chosen. Four safety confidence levels were defined, referred to as risk levels, indicating the risk of systems overloading or malfunctioning if the recommended load is exceeded. To train the algorithm on different risk levels and show how the different risks affect the results, four different alpha values were used. A higher alpha value corresponds to a higher risk level, leading to the recommended load being closer to the maximum possible load. For instance, if the alpha value is set to 90%, then the model will predict a value which in 90% of similar situations the actual load stayed below that value. This can be used to adjust the desired safety margin when operating the boiler.

After the algorithm selection, parameters and model settings were automatically fine-tuned using RandomizedSearchCV. A set of reasonable values for each parameter was defined and the function determined a combination of the parameters that minimized a chosen cost function. For this purpose, *mean_pinball_loss* was used. The defined values for each parameter and the parameters chosen by the function for each alpha value can be viewed in Table 5.3.

Table 5.3: Set of values for fine-tuning parameters, along with the chosen values for each parameter and alpha value.

Parameter	**Values**	$\alpha = 0.8$	$\alpha = 0.85$	$\alpha = 0.9$	$\alpha = 0.95$
n_estimators	[300, 500, 800]	800	800	800	800
learning_rate	[0.01, 0.03, 0.05]	0.03	0.03	0.05	0.05
num_leaves	[15, 31, 50]	50	50	31	31
max_depth	[3, 5, 10]	10	10	10	10
min_child_samples	[3, 5, 10]	3	3	10	10
colsample_bytree	[0.7, 0.8, 0.9]	0.7	0.7	0.8	0.8
subsample	[0.7, 0.8, 0.9]	0.8	0.8	0.7	0.7
subsample_freq	[1, 5]	1	1	1	1

5.1.3 Evaluation

To evaluate the algorithm the metric pinball loss was used, resulting in Table 5.4. Based on the scale of the boiler load values (10-80 MW), the achieved pinball loss of 0.6-1.2 suggests the model is performing well on unseen data.

Table 5.4: Pinball loss.

Alpha	Pinball loss
$\alpha = 0.8$	0.666
$\alpha = 0.85$	0.776
$\alpha = 0.9$	0.924
$\alpha = 0.95$	1.189

Then, a number of random samples were selected and the predicted boiler loads for each alpha value as a function of the actual boiler load was plotted. This resulted in Figure 5.1. Darker dots indicate a higher alpha value and consequently a higher risk.

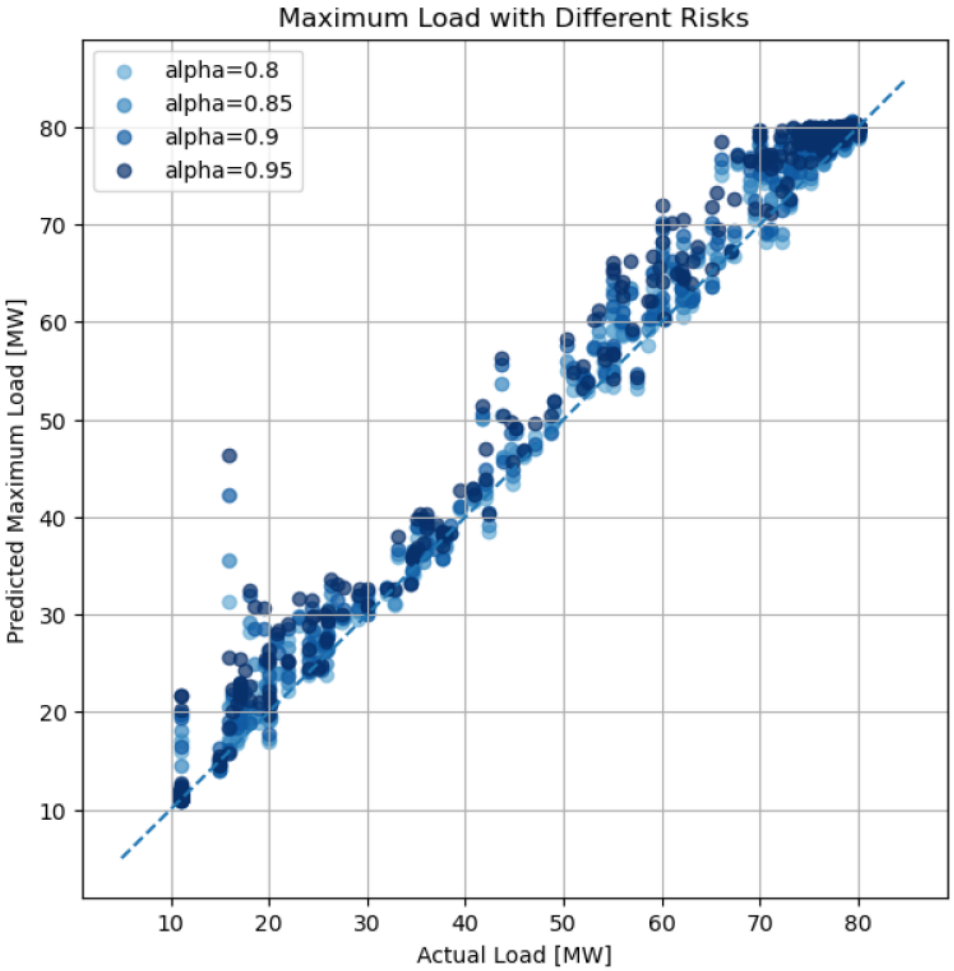


Figure 5.1: Plot displaying predictions with different alpha values as a function of actual boiler load.

Finally, a small number of samples and their predicted boiler load for each alpha value was plotted as a function of time. This resulted in Figure 5.2. Darker lines indicate a higher alpha value and consequently a higher risk.

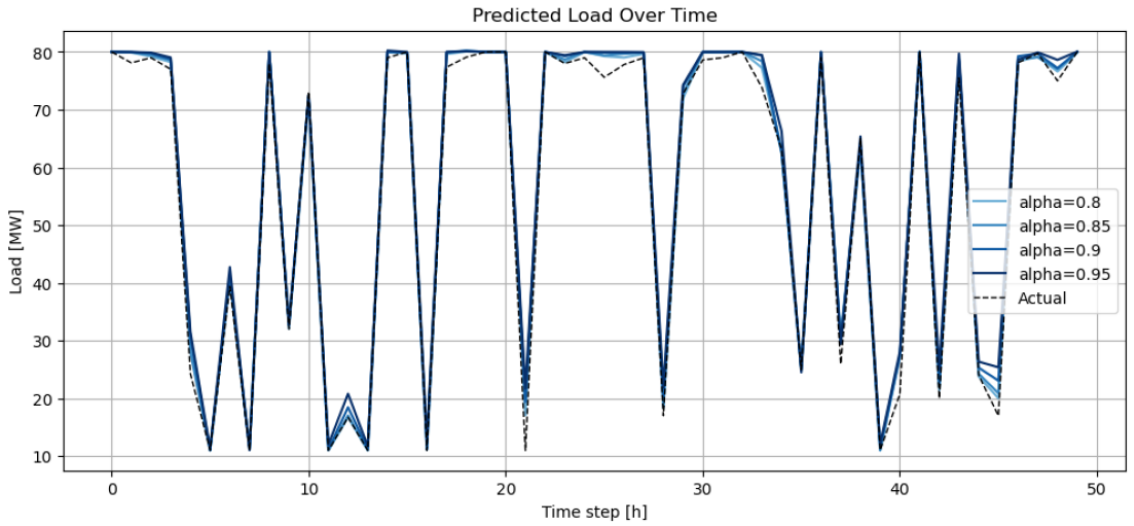


Figure 5.2: Plot displaying predictions with different alpha values as a function of time.

The predicted load is a bit higher for each alpha value since the safety margin is decreased. It can also be noted that a majority of predictions were higher than the actual load, which is to be expected since the boiler has not been run optimally. However, since the dataset did not include optimal boiler load, it is not certain that the recommended load is fully optimal either. The recommended load is just a probabilistic upper bound, not a definite limit. Although it is difficult to know whether the predicted maximum loads would work in an operational setting without further testing, they can still be considered more optimal than before and be seen as potential improvements.

5.2 Fuel Data

In an interview, Participant 9 et al. [26] discussed how Mölndal Energi AB collects large amounts of data in multiple areas within production, one of which is fuel data. Although there are alarms indicating when certain parameters exceed a specified interval, it can be difficult to detect deviations and anomalies within those intervals. It can also be challenging to find the root cause of an alarm, since alarms could be caused by a chain reaction of parameters affecting each other. This means a parameter whose deviation remained within the limits and has therefore not been detected might be the actual cause of the problem. Therefore, anomaly detection could be an important addition to statistical analytics. In this thesis it is applied to fuel data, however it could be applied to production data and other types of data as well.

5.2.1 Data

Due to recent changes in how the fuel data has been stored, only one year of data was collected for training and testing, consisting of approximately 1460 samples. This means seasonal trends could not be detected, however weekly and monthly trends might still be identified. The small amount of data increases the risk of noise affecting the result, however as a proof of concept the data was determined sufficient.

The collected data consisted of multiple parameters. Each sample corresponds to a delivery, and parameters include delivery name, time, amount of fuel etc. It is often preferable to include as many relevant parameters as possible, to allow the model to find patterns and relationships between these. Depending on what parameters are of interest, the range of input and output parameters could be changed. The chosen input and output parameters for the proof of concept were based on the interview with Participant 9 et al. [26] and can be viewed in Table 5.5.

Table 5.5: Chosen parameters and their type of usage.

Name	**Parameter**	**Meta/Input/Output**
meta1	Delivery name	Meta
meta2	Delivery date	Meta
var1	Full weight of delivery (with water)	Input
var2	Weight of dry matter (without water)	Input
var3	Energy content	Input
var4	Chipped volume	Input
var5	Solid volume	Input
out1	Percentage dry matter	Output
out2	Calorimetric value for dry matter	Output
out3	Percentage ash content	Output

After loading the data, it was evaluated and pre-processed. Minimum, maximum and mean values of each parameter were calculated, resulting in Table 5.6. Parameter values were plotted (figures can be found within the code in Appendix B) and rows containing NaNs were removed. After the pre-processing approximately 1380 samples remained.

Table 5.6: Minimum, maximum and mean values for each parameter.

Name	**Parameter**	**Min**	**Max**	**Mean**
var1	Full weight of delivery (with water)	0.0	53.6	37.2
var2	Weight of dry matter (without water)	0.1	41.1	19.7
var3	Energy content	0.0	196.3	90.9
var4	Chipped volume	0.1	156.0	123.7
var5	Solid volume	0.3	80.5	50.8
out1	Percentage dry matter	20.1	88.0	55.1
out2	Calorimetric value for dry matter	5.2	6.0	5.4
out3	Percentage ash content	1.0	16.2	2.8

Although samples were originally ordered in time, they were considered independent to each other. As in the first proof of concept, the data was randomly split into a training set and a test set with the ratio 80% for the training set and 20% for the test set. This resulted in approximately 1100 samples in the training set and 280 in the test set.

5.2.2 Model

Two ways to detect deviations in data are to either train a model to detect deviations directly, or to train a model to predict expected values and then calculate whether the actual values differ from the expected values. For this proof of concept the second alternative was used, meaning the model predicts the normal behaviour and anomalies are inferred after prediction, not learned directly.

Since the data included values for all relevant parameters and the objective was to predict a number, supervised regression algorithms could be used. Due to the many suitable algorithms for regression, three algorithms were compared: LGBM, XGB and RF. Once again, parameters were automatically fine-tuned using RandomizedSearchCV, now with the cost function *neg_mean_absolute_error*. The defined set of values for each parameter, as well as the parameters chosen by the function for each algorithm can be viewed in Tables 5.7-5.9.

Table 5.7: Set of values for fine-tuning parameters, along with the chosen values, for LGBM.

Parameter	**Values**	**Chosen**
estimator___n_estimators	[300, 600, 1000]	1000
estimator___learning_rate	[0.01, 0.03, 0.05]	0.03
estimator___num_leaves	[31, 63, 127]	127
estimator___subsample	[0.7, 0.9, 1.0]	1.0
estimator___colsample_bytree	[0.7, 0.9, 1.0]	1.0

Table 5.8: Set of values for fine-tuning parameters, along with the chosen values, for XGB.

Parameter	**Values**	**Chosen**
estimator___n_estimators	[300, 600, 1000]	600
estimator___learning_rate	[0.01, 0.03, 0.05]	0.05
estimator___max_depth	[3, 5, 7]	7
estimator___subsample	[0.6, 0.8, 1.0]	0.8
estimator___colsample_bytree	[0.6, 0.8, 1.0]	1.0

Table 5.9: Set of values for fine-tuning parameters, along with the chosen values, for RF.

Parameter	**Values**	**Chosen**
estimator___n_estimators	[200, 500, 1000]	200
estimator___max_depth	[5, 10, 20, None]	10
estimator___min_samples_leaf	[1, 3, 5]	1
estimator___max_features	["sqrt", "log2", None]	None

Once the model had predicted the expected values of each parameter, these were compared with actual values to find anomalies. For simplicity, a threshold was set to two standard deviations, however this could be changed depending on the aim and application. For each sample the residuals (the difference between the actual and predicted values) of each parameter were compared to the threshold, and samples with residuals exceeding the threshold were stored in tables. The samples with the most deviating parameters for each algorithm can be viewed in Tables 5.10-5.12.

Table 5.10: Samples with the most deviating parameters when using LGBM.

Sample	out1	out2	out3	Number of deviations
322	0.269	-0.109	-1.154	2
451	-0.857	0.072	0.783	2
1219	-0.943	-0.213	-1.239	2
1229	0.388	-0.084	-0.993	2
86	-1.495	0.084	0.947	2

Table 5.11: Samples with the most deviating parameters when using XGB.

Sample	out1	out2	out3	Number of deviations
752	1.958	-0.144	0.742	3
1221	2.217	-0.264	-0.837	3
49	4.451	0.122	-0.114	2
451	-0.159	0.083	0.853	2
1198	-3.980	0.013	0.773	2

Table 5.12: Samples with the most deviating parameters when using RF.

Sample	out1	out2	out3	Number of deviations
752	2.474	-0.137	0.804	3
1221	3.418	-0.289	-0.758	3
65	-2.188	0.001	-0.791	2
49	4.055	0.095	0.155	2
322	-0.027	-0.144	-1.512	2

5.2.3 Evaluation

The algorithms were first evaluated using the metrics MAE, MSE and RMSE, resulting in Tables 5.13- 5.15. Similar to pinball loss, the aim of these values is to be as small as possible.

Table 5.13: Metrics used for evaluation of LGBM.

Metric	out1	out2	out3
MAE	0.475	0.019	0.221
MSE	0.630	0.001	0.119
RMSE	0.794	0.034	0.345

Table 5.14: Metrics used for evaluation of XGB.

Metric	out1	out2	out3
MAE	0.426	0.019	0.211
MSE	0.532	0.001	0.121
RMSE	0.723	0.038	0.347

Table 5.15: Metrics used for evaluation of RF.

Metric	out1	out2	out3
MAE	0.512	0.025	0.213
MSE	0.709	0.002	0.115
RMSE	0.842	0.045	0.339

Since values for out1 ranges between 20.1-88.0, while the metrics ranges between 0.42-0.85, this shows all models were well trained for predicting the parameter. Both out2 and out3 has smaller ranges, with out2 ranging between 5.2-6.0 and out3 ranging between 1.0-16.2, meaning the values for the metrics should be smaller for these. For out2 the metrics range from 0.001-0.045 and for out3 the metrics range from 0.11-0.35. This indicates all models performed relatively well on all parameters. When comparing the models, RF generally performed worse than the other two models. XGB performed slightly better than LGBM on predicting out1, but with similar results on the other two parameters.

The predictions were also evaluated by plotting the predicted values as a function of the actual values, and the residuals as a function of the predictions. The plots for LGBM can be viewed in the Figures 5.3-5.5, while the rest are found in Appendix A.

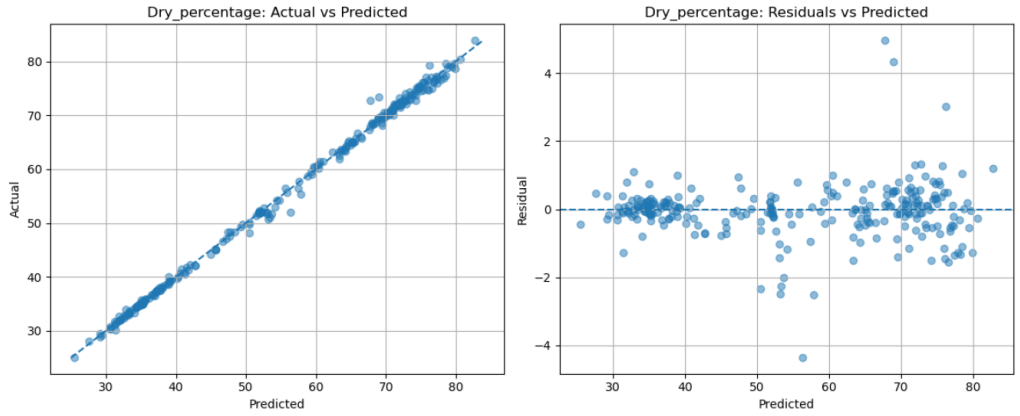


Figure 5.3: Plots for evaluating predictions of out1 using LGBM. The plot on the left shows predicted values as a function of actual values. The plot on the right shows the residuals for each prediction.

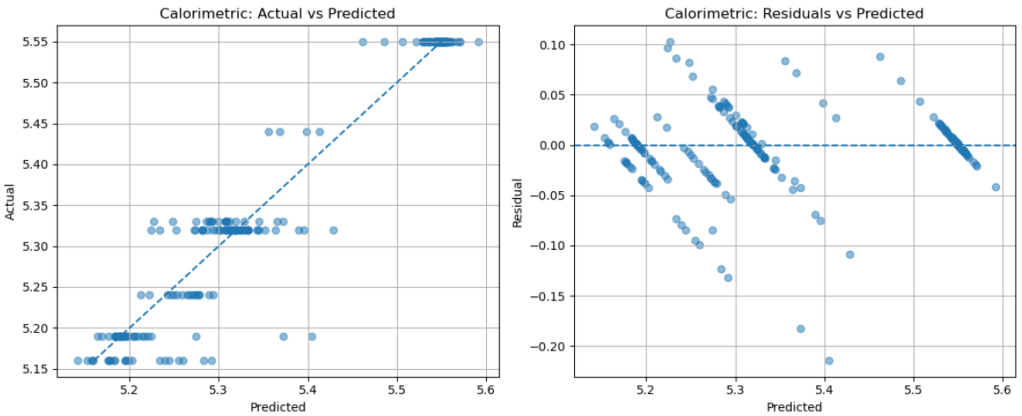


Figure 5.4: Plots for evaluating predictions of out2 using LGBM. The plot on the left shows predicted values as a function of actual values. The plot on the right shows the residuals for each prediction.

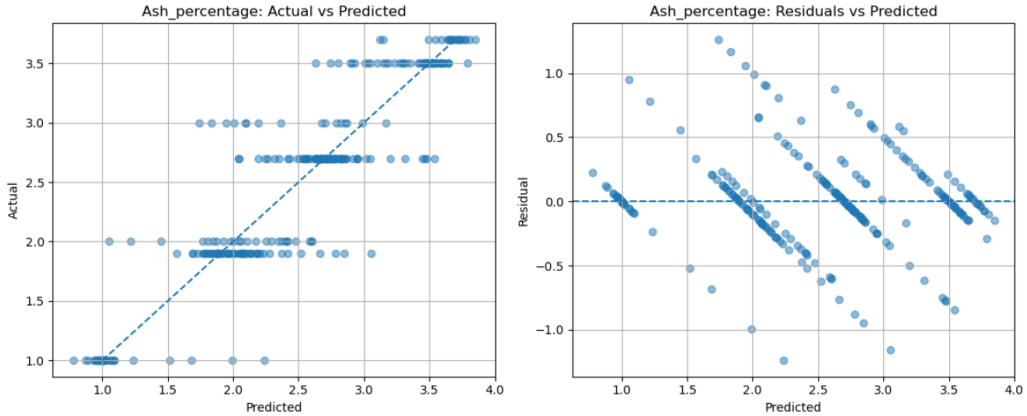


Figure 5.5: Plots for evaluating predictions of out3 using LGBM. The plot on the left shows predicted values as a function of actual values. The plot on the right shows the residuals for each prediction.

When comparing the plots for each parameter and each model, it can be concluded that the residuals for predictions of out1 increase a bit with the size of the prediction, while the residuals remained within similar ranges for all three models. For out2 and out3 the actual values were discrete, resulting in a special behaviour with dots creating rows. This is natural since the models predicts continuous values and has not learnt the discrete behaviour of the parameters. That is something which would need further development in order to improve the predictions before actual implementing.

Finally, the residuals for each parameter as a function of time was plotted, with anonymized names of each sample written on the X-axis. The resulting plots when using LGBM can be viewed in Figure 5.6, while the rest are found in Appendix A.

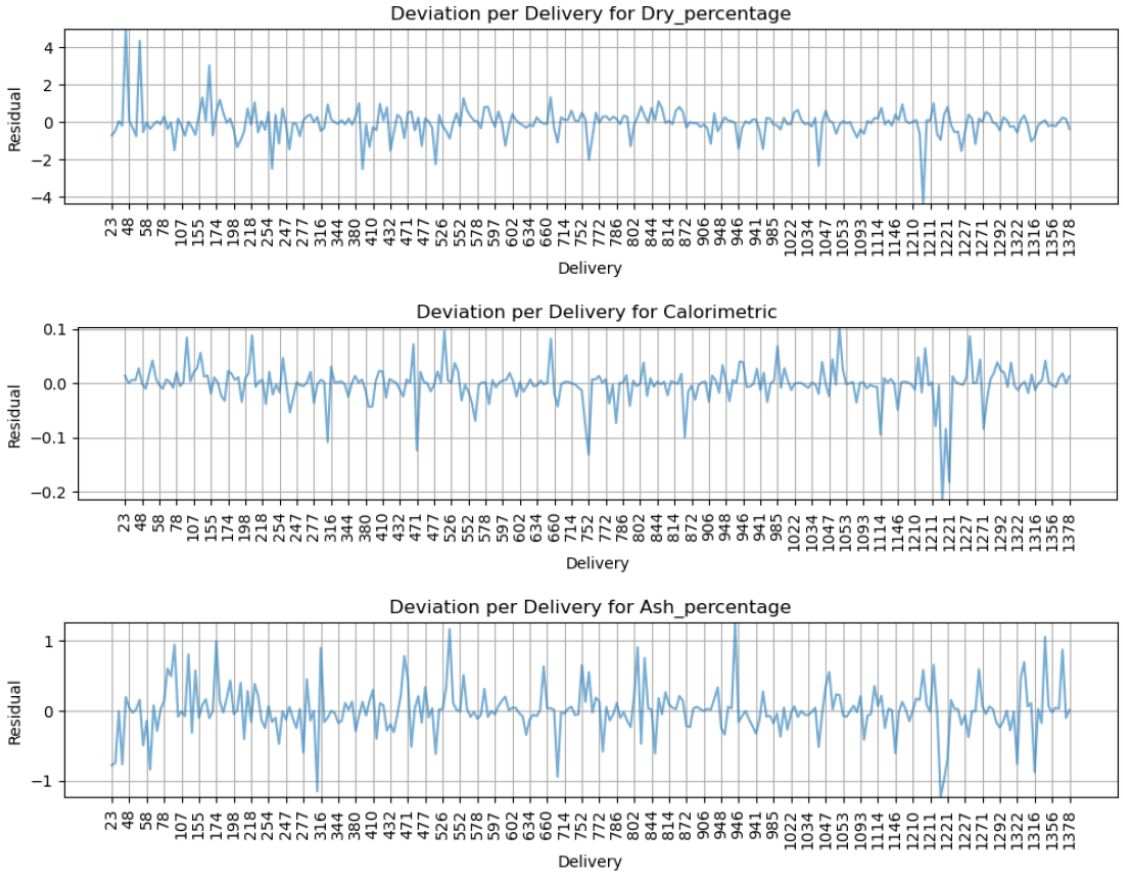


Figure 5.6: Plot displaying residuals for each parameter using LGBM.

When comparing the final plots for each model, it can be concluded that the models produce fairly similar results.

6
Discussion

The results of the thesis and the extent to which the thesis achieved its objectives is discussed, as well as the reliability of the results and potential areas for future work. The thesis aimed to evaluate the possibilities of implementing ML within the production system at Mölndal Energi AB. The outcome of the thesis included an overview of potential applications and important aspects of implementation, in addition to two proof of concepts demonstrating simplified versions of potential applications.

6.1 Research Questions

Each defined objective is discussed and the extent to which the thesis addressed them evaluated.

6.1.1 Question 1

What potential applications of machine learning can be identified within the production system at Mölndal Energi AB, based on available data and operational needs?

The literature review revealed multiple relevant applications for ML within the production system at Mölndal Energi AB, including predictive maintenance, demand forecasting, scheduling and energy storage. Mölndal Energi AB already use, both directly and indirectly through Göteborg Energi, external ML applications for demand forecasting and scheduling. Therefore, these areas would only benefit from internally developed solutions if these were better and cheaper. Predictive maintenance provided the most relevant applications and the highest potential for improvement. Regarding energy storage, it could be relevant to implement a solution for heat storage and it might be relevant to optimize the district cold storage once the accumulator is finished. Although the discovered applications could be investigated further, it is concluded that the research question has been addressed satisfactorily.

6.1.2 Question 2

What aspects of implementation do Mölndal Energi AB need to consider in order to implement machine learning in a safe and structured way?

The literature review revealed the importance of implementing ML incrementally,

with solutions that monitor a system being preferred initially compared to operational solutions. It was also suggested that operators and other end-users should be introduced to the development process early, and a framework for implementation was provided. The importance of sufficient data has been highlighted, along with suggestions of limiting summertime data and collaborating with other actors within the energy sector. Hence the research question is satisfactorily addressed, however a more detailed framework specifically adapted to Mölndal Energi AB could be developed and tested to identify additional aspects of importance.

6.1.3 Question 3

How can selected applications be designed and implemented as simple proof of concepts?

Two proof of concepts were developed, one to estimate the maximum boiler load and another to detect anomalies in fuel data. The boiler load model used five years of operational data from 800xA, while the fuel data model used only one year of data due to changes in data storage. An LGBM algorithm was used for quantile regression to predict boiler load, and three models (LGBM, XGB, RF) were compared for the fuel data case. Parameters were fine-tuned using RandomizedSearchCV in both cases. When evaluating the predictions using pinball loss for boiler load and MAE, MSE and RMSE for the fuel data, it was concluded that all models performed relatively well on unseen data. Plots showed reasonable results for both proof of concepts, however the anomaly detection models could be improved to account for the discrete behaviour of certain parameters. Although neither proof of concept was developed enough to be used operationally, they demonstrated feasibility and provided a good foundation for future implementations. Therefore, it can be concluded that the research question has been addressed.

6.2 Reliability

The results of the literature review are mainly based on reports obtained from Google Scholar, which were all cited and deemed reliable. All sources have been stated in the report for transparency. For practical information about how to implement certain ML models, the website GeeksforGeeks was used. Since it is not a scientific paper, it has not been reviewed in the same way and might include unreliable information. However, many of the proposed methods have been tested during the proof of concept with results that align with expectations. Information about Mölndal Energi AB is mainly based on interviews held with people within the company. There is a risk of information being incorrect due to the human factor, however all information has been reviewed after the interviews to check important statements. Since many of the mentioned applications were only proposed as potentially relevant, they have not been implemented and tested meaning they may not actually function in reality.

The proof of concepts were developed based on available data and obtained knowledge about ML. ChatGPT has been used to assist coding, for debugging purposes

and to suggest improvements of the code, however all decision making, assumptions and interpretations were made by the author alone. Code provided by ChatGPT might be incorrect, but thorough reviews, tests and evaluations have been performed to ensure all code is functioning properly and as expected.

6.3 Future Work

Specific areas of interest to investigate further include the usage of digital twins to simulate and debug systems. While requiring detailed models of the system, the possible benefits are plenty. Another interesting area is inspection and fault detection using drones and augmented reality. This could aid maintenance and help detect faulty components earlier. Large Language Models, similar to ChatGPT, could be trained on internal technical documentation to answer questions related to maintenance, generate instructions for service of components and similar, which could save both time and money. It would also be relevant to investigate how code generation through Claude Code or ChatGPT could be integrated into the development of ML tools, since these do not require profound programming experience.

The proof of concepts could be developed further by using larger datasets, evaluating more algorithms and fine-tuning parameters. For both proof of concepts, a first step would be to identify all relevant parameters and collect more data. The second proof of concept would require tuning of the threshold to ensure it suits the dataset and aim, and it could also be developed to account for weekly and monthly trends by including weekday data, or be applied to similar applications by changing the dataset. Physical tests would also be necessary before full deployment of either proof of concept, preferably with a limited environment initially.

Although ML has been proposed as a potential solution to certain applications, it is important to consider other solutions as well. For critical applications, more robust alternatives such as system control and model predictive control should also be reviewed and analysed since these might be more suitable. Simple statistical models, such as historical max and constant quantile, could also be evaluated to determine whether it is at all necessary to implement machine learning models or if simpler methods would be sufficient.

7
Conclusion

Conclusions from the literature review and the proof of concepts are summarized.

7.1 Literature Review

The conclusions of the literature review are divided into one section for each investigated application, in addition to one section regarding implementation aspects.

7.1.1 Predictive Maintenance

Predictive maintenance and fault detection are highly relevant implementations for ML within the production at Mölndal Energi AB. Current alarms acknowledge deviations but are based on set intervals and only alarms when a value is outside those intervals. They might miss oscillations and do not show how parameters affect each other or the root cause to the problem. Proposed areas for application are monitoring of the conveyor belt, scheduling cleaning of the flue gas condenser filters, detecting leakages in the tube vessels and analysing the problems with sintering in B3. Proposed algorithms include CNNs, RNNs, MLPs, LSTMs and autoencoders, as well as SVMs and RFs.

7.1.2 Demand Forecasting

To forecast demand Mölndal Energi AB use linear regressions on production data with one or multiple parameters. They also use systems like Utilifeed, which provide the demand load but not the production load, and get forecasts from Energy Optima 3 through Göteborg Energi. There are three optimizations simultaneously running; daily, weekly and monthly. Different algorithms have been suggested, with SVM and ANN being highlighted for their robustness and ability to generalize. LSTMs have demonstrated low error rates and proved suitable for time-series forecasting tasks.

7.1.3 Scheduling

At Mölndal Energi AB the most important decisions within heat production are how to schedule the boilers. Currently no AI is used for this, however they have regular meetings with Göteborg Energi to discuss the scheduling. Regarding the district cooling, the planning will change once the accumulator is finished, leaving it

an interesting application for the future. A study analysing a genetic algorithm for determining the optimal operating schedules of subsystems was mentioned, showing an increase in profit in comparison with a rule-based, priority order baseline strategy. Another option would be using a reinforcement learning-based approach.

7.1.4 Energy Storage

By over-heating the grid, it could be used as an accumulator, storing heat which can be used during peaks instead of having to produce more heat. Yet, Mölndal Energi AB is currently not using energy storage for neither district heat nor cold. Once the accumulator is finished, it will allow storing the cold so it can be produced when the price is low and distributed when the price is high. In one mentioned study, reinforcement learning algorithms have been applied to district heating systems to account for environment and modelling uncertainties. In another study, ANNs were combined with a genetic algorithm optimizer to develop a model predictive control strategy for storing district cold. The method could effectively predict performance and determine the optimal control strategy of thermal energy storage in a district cooling system with ice storage.

7.1.5 Implementation

During the literature review and interviews it was found that the implementation should be incremental. A framework with the steps Needs finding, Preparation, Feasibility study, Development and evaluation in experimental environments, Development and evaluation in operational environments, System validation and deployment, as well as Post-deployment was described. The importance of involving operators and other end-users early in the development process was highlighted, and it was noted that implementing solutions which monitor a system are often easier and more secure compared to operational solutions. It was mentioned that most facilities within heat production are only operating limited periods, resulting in a limited amount of continuous operational data and a slow development rate. However since many energy companies are municipally owned there is a large potential for collaboration which can aid the development.

7.2 Proof of Concept

Two proof of concepts were developed, one to estimate the maximum boiler load and another to detect anomalies in fuel data.

7.2.1 Boiler Load

The boiler load model used five years of operational data from 800xA, with the summer periods excluded to improve computational efficiency. An LGBM algorithm was used to allow for quantile regression with four different alphas. Parameters were fine-tuned using RandomizedSearchCV with the cost function *mean_pinball_loss*. When evaluating the predictions using pinball loss, results indicated the model performed

well on unseen data. A plot with predicted loads by actual loads and another plot with predicted loads over time showed most predictions were higher than the actual load, which was to be expected since the boiler has not been run optimally. For each alpha value, the predicted load was a bit higher which is also to be expected since the safety margin is decreased.

7.2.2 Fuel Data

The fuel data model used only one year of data, due to recent changes in data storage, meaning seasonal trends were not detected and the risk of noise affecting the results being relatively large. Three parameters were analysed for deviations and three algorithms for regression were compared: LGBM, XGB and RF. Parameters were fine-tuned using RandomizedSearchCV with the cost function *neg_mean_absolute_error*. When evaluating the predictions using MAE, MSE and RMSE, results showed all models performed well, with XGB and LGBM being superior depending on the parameter. Deviating parameters were summarized in tables, as well as plotted. Residuals were generally larger for larger predicted values of the first parameter, while discrete values made plots of the other two parameters consist of dots in rows. All models produced similar results.

Bibliography

[1] A. O. Aderibigbe, E. C. Ani, P. E. Ohenhen, N. C. Ohalete, and D. O. Daraojimba, "Enhancing energy efficiency with ai: A review of machine learning models in electricity demand forecasting," *Engineering Science & Technology Journal*, vol. 4, no. 6, pp. 341–356, 2023.

[2] E. Balouji, E. Berggren, K. Bäckström, I. Kärrman, and J. Rådemar, *Maskininlärning, elkvalitet och smarta elnät*, 2020.

[3] S. Bello, I. Wada, O. Ige, E. Chianumba, and S. Adebayo, "Ai-driven predictive maintenance and optimization of renewable energy systems for enhanced operational efficiency and longevity," *International Journal of Science and Research Archive*, vol. 13, no. 1, pp. 2823–2837, 2024.

[4] M. S. S. Danish, "Ai in energy: Overcoming unforeseen obstacles," *AI*, vol. 4, no. 2, pp. 406–425, 2023.

[5] Energy Opticon, *Energy optima 3*, `https://energyopticon.com/sv/energy-optima-3-sv/`, Accessed: 2026-03-05, 2026.

[6] GeeksforGeeks, *Geeksforgeeks*, `https://www.geeksforgeeks.org/`, Accessed: 2026-03-05, 2026.

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

[8] A. Hamdan, K. I. Ibekwe, V. I. Ilojianya, S. Sonko, E. A. Etukudoh, et al., "Ai in renewable energy: A review of predictive maintenance and energy optimization," *International Journal of Science and Research Archive*, vol. 11, no. 1, pp. 718–729, 2024.

[9] E. Löfblad, T. Unger, D. Holmström, M. Lewan, and S. Montin, *Digital utveckling och möjligheter för energisektorn*, 2018.

[10] E. Löfblad, T. Unger, D. Holmström, M. Lewan, and S. Montin, *Digital utveckling och möjligheter för energisektorn*, 2019.

[11] G. Mbiydzenyuy, S. Nowaczyk, H. Knutsson, D. Vanhoudt, J. Brage, and E. Calikus, "Opportunities for machine learning in district heating," *Applied Sciences*, vol. 11, no. 13, p. 6112, 2021.

[12] O. Mogren, O. Penttinen, M. Willbo, and K. Åkerlund, *Plattform för prediktivt underhåll*, 2023.

[13] Mölndal Energi AB, *Introduktionsutbildning riskulla kylanläggning*, Accessed: 2025-12-11, 2023.

[14] T. Nguyen, "Applications of artificial intelligence for demand forecasting," *Operations and Supply Chain Management: An International Journal*, vol. 16, no. 4, pp. 424–434, 2023.

[15] C. Ntakolia, A. Anagnostis, S. Moustakidis, and N. Karcanias, "Machine learning applied on the district heating and cooling sector: A review," *Energy Systems*, vol. 13, no. 1, pp. 1–30, 2022.
[16] E. C. Onukwulu, M. O. Agho, and N. L. Eyo-Udo, "Developing a framework for ai-driven optimization of supply chains in energy sector," *Global Journal of Advanced Research and Reviews*, vol. 1, no. 2, pp. 82–101, 2023.
[17] S. Onwusinkwue et al., "Artificial intelligence (ai) in renewable energy: A review of predictive maintenance and energy optimization," *World Journal of Advanced Research and Reviews*, vol. 21, no. 1, pp. 2487–2499, 2024.
[18] Participant 1, Interview, Date: 2026-02-23.
[19] Participant 1 and Participant 2, Interview, Date: 2026-02-09.
[20] Participant 3, Interview, Date: 2026-02-10.
[21] Participant 3 and Participant 4, Interview, Date: 2026-03-11.
[22] Participant 5, Interview, Date: 2026-02-11.
[23] Participant 6, Interview, Date: 2026-02-13.
[24] Participant 7, Interview, Date: 2026-02-23.
[25] Participant 8, Interview, Date: 2026-03-02.
[26] Participant 9 and Participant 10, Interview, Date: 2026-03-12.
[27] J. Preisz and D. Gustafsson, "Operation optimisation of district heating production," 2010.
[28] K. M. Salem, F. J. Rey-Martínez, A. Elgharib, and J. M. Rey-Hernández, "Energy demand forecasting scenarios for buildings using six ai models," *Applied Sciences*, vol. 15, no. 15, p. 8238, 2025.
[29] K. Stepanovic, J. Wu, R. Everhardt, and M. de Weerdt, "Unlocking the flexibility of district heating pipeline energy storage with reinforcement learning," *Energies*, vol. 15, no. 9, p. 3290, 2022.
[30] Utilifeed, *Utilifeed*, `https://www.utilifeed.com/sv`, Accessed: 2026-03-05, 2026.
[31] J. Vermorel, *Pinball loss function*, `https://www.lokad.com/pinball-loss-function-definition/`, Accessed: 2026-05-09, 2012.
[32] F. Wästberg, M. Hansson, and R. Edland, *Branschsamarbete för avancerad analys av värmedistribution och uppvärmningsbehov*, 2022.
[33] Y. Yamamoto, Á. Aranda-Muñoz, and K. Sandström, "A development framework for human work integrated ai systems in manufacturing," *Frontiers in Manufacturing Technology*, vol. 5, p. 1 601 903, 2025.
[34] Y. Yamamoto, A. A. Muñoz, and K. Sandström, "Practical aspects of designing a human-centred ai system in manufacturing," *Procedia Computer Science*, vol. 232, pp. 2626–2638, 2024.

A

Plots

A.1 Residuals of out1

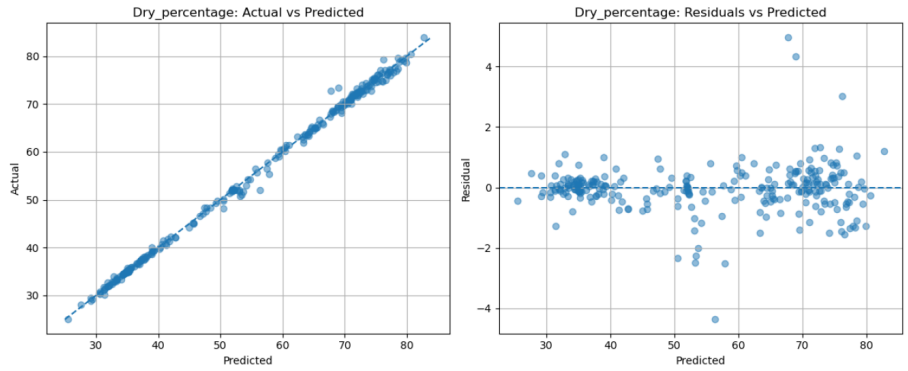


Figure A.1: Plots for evaluating predictions of out1 using LGBM. The plot on the left shows predicted values as a function of actual values. The plot on the right shows the residuals for each prediction.

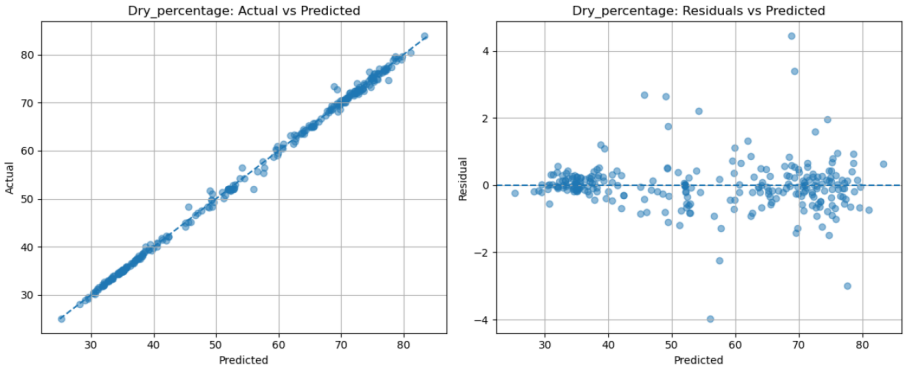


Figure A.2: Plots for evaluating predictions of out1 using XGB. The plot on the left shows predicted values as a function of actual values. The plot on the right shows the residuals for each prediction.

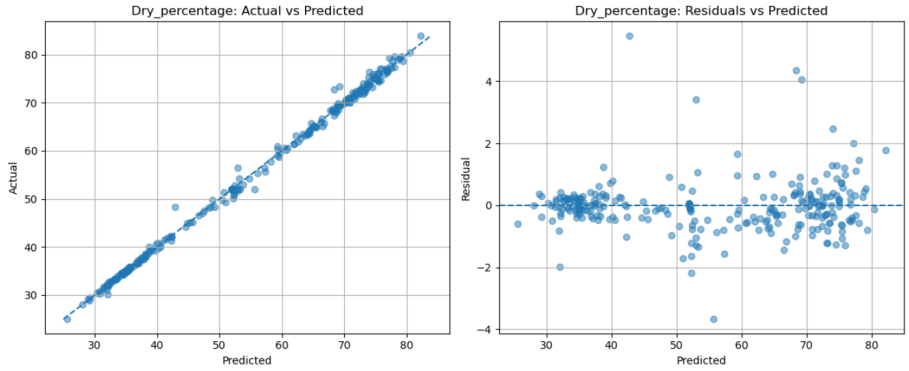


Figure A.3: Plots for evaluating predictions of out1 using RF. The plot on the left shows predicted values as a function of actual values. The plot on the right shows the residuals for each prediction.

A.2 Residuals of out2

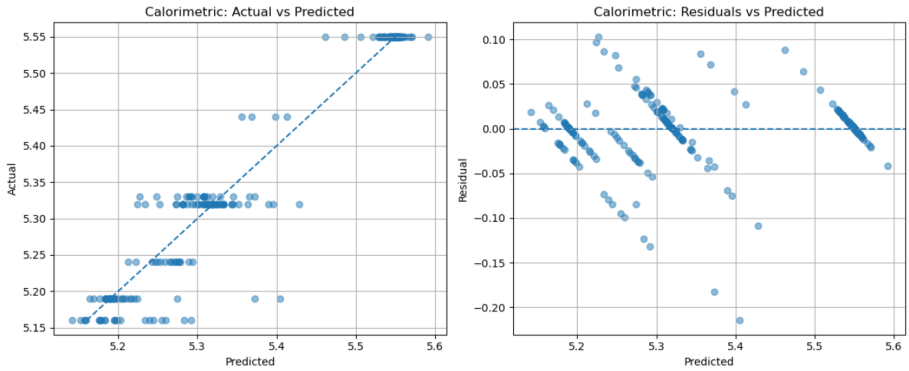


Figure A.4: Plots for evaluating predictions of out2 using LGBM. The plot on the left shows predicted values as a function of actual values. The plot on the right shows the residuals for each prediction.

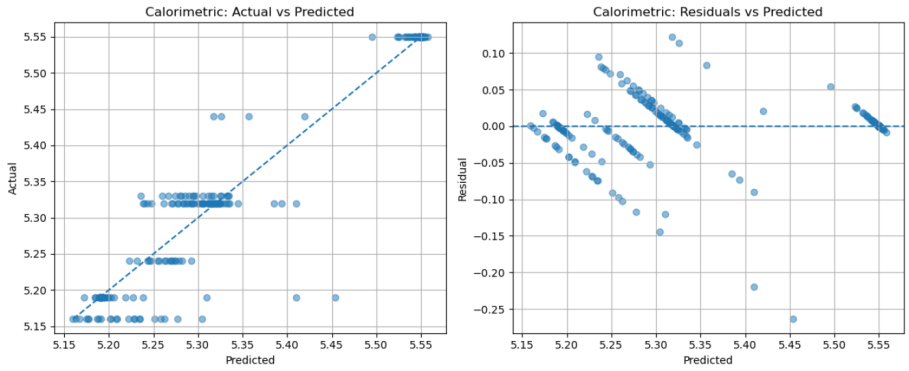


Figure A.5: Plots for evaluating predictions of out2 using XGB. The plot on the left shows predicted values as a function of actual values. The plot on the right shows the residuals for each prediction.

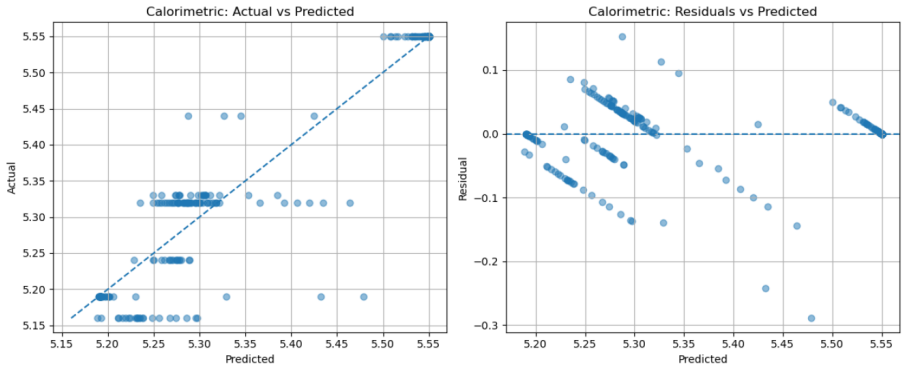


Figure A.6: Plots for evaluating predictions of out2 using RF. The plot on the left shows predicted values as a function of actual values. The plot on the right shows the residuals for each prediction.

A.3 Residuals of out3

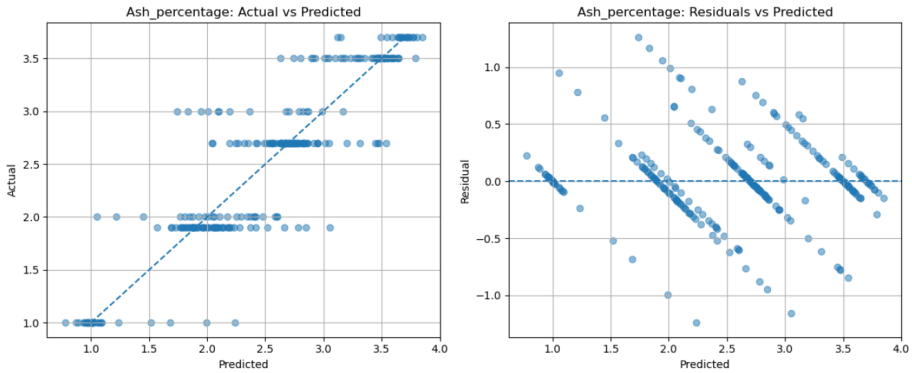


Figure A.7: Plots for evaluating predictions of out3 using LGBM. The plot on the left shows predicted values as a function of actual values. The plot on the right shows the residuals for each prediction.

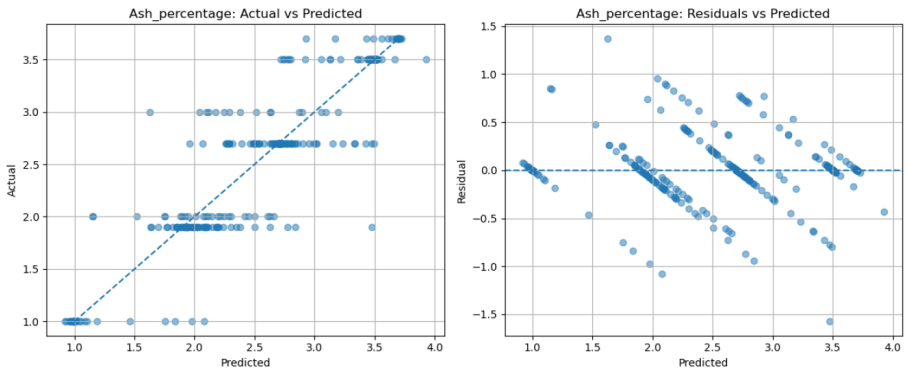


Figure A.8: Plots for evaluating predictions of out3 using XGB. The plot on the left shows predicted values as a function of actual values. The plot on the right shows the residuals for each prediction.

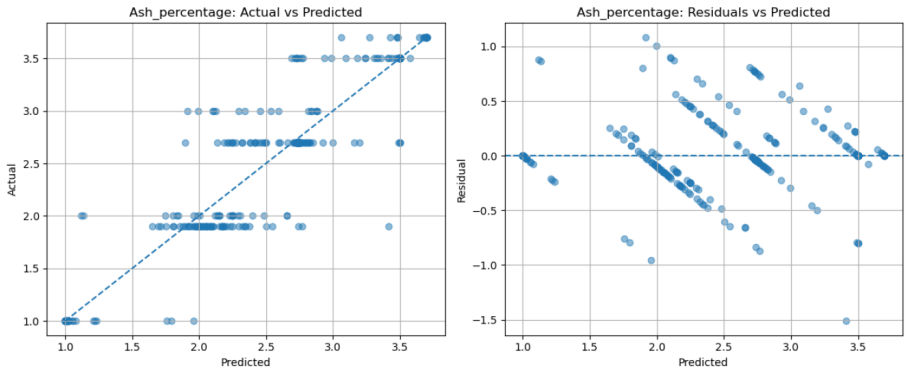


Figure A.9: Plots for evaluating predictions of out3 using RF. The plot on the left shows predicted values as a function of actual values. The plot on the right shows the residuals for each prediction.

A.4 Deviations over time

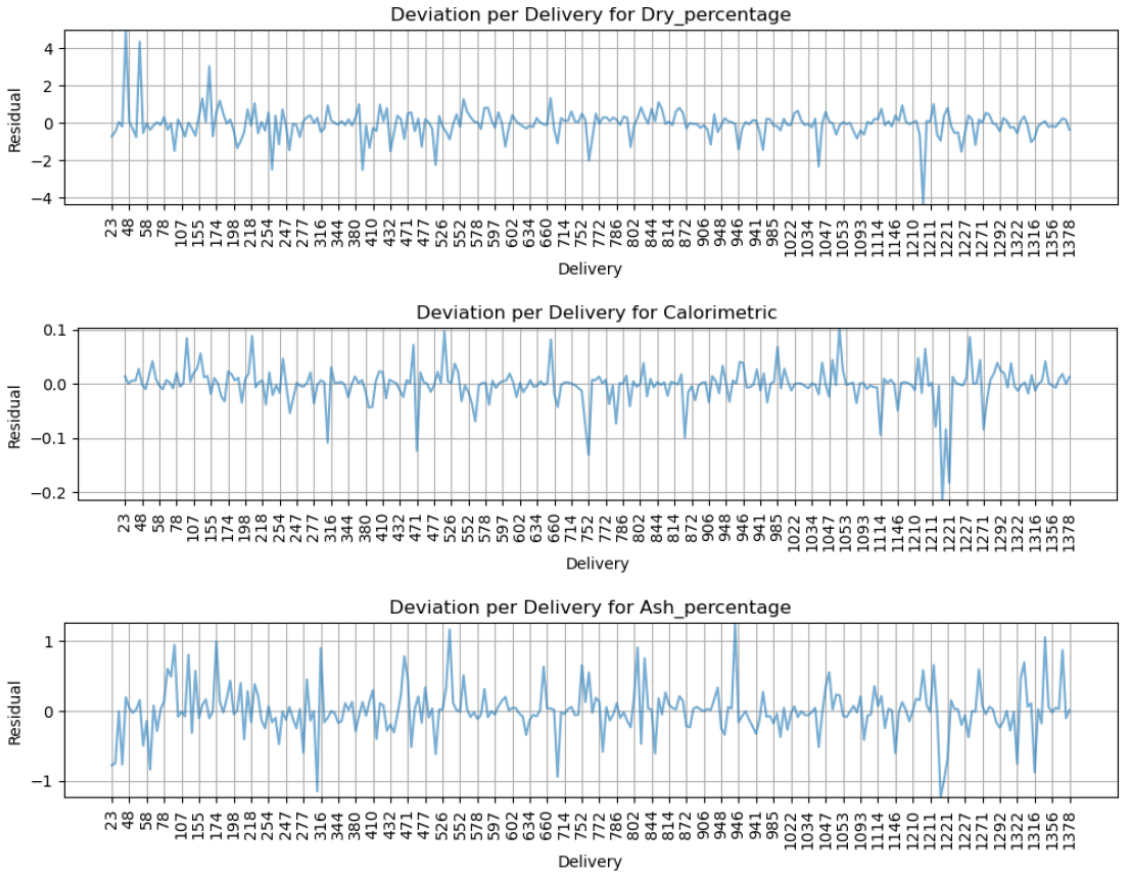


Figure A.10: Plots displaying residuals for each parameter using LGBM.

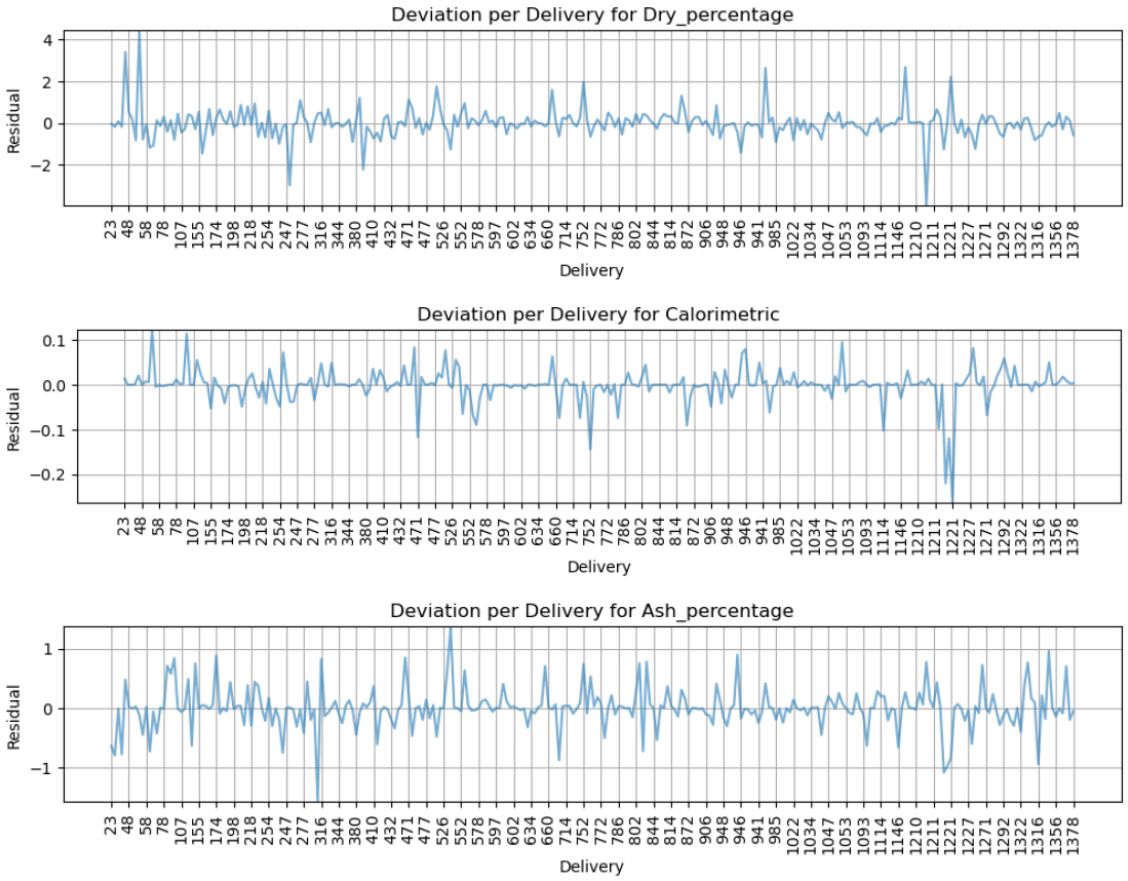


Figure A.11: Plots displaying residuals for each parameter using XGB.

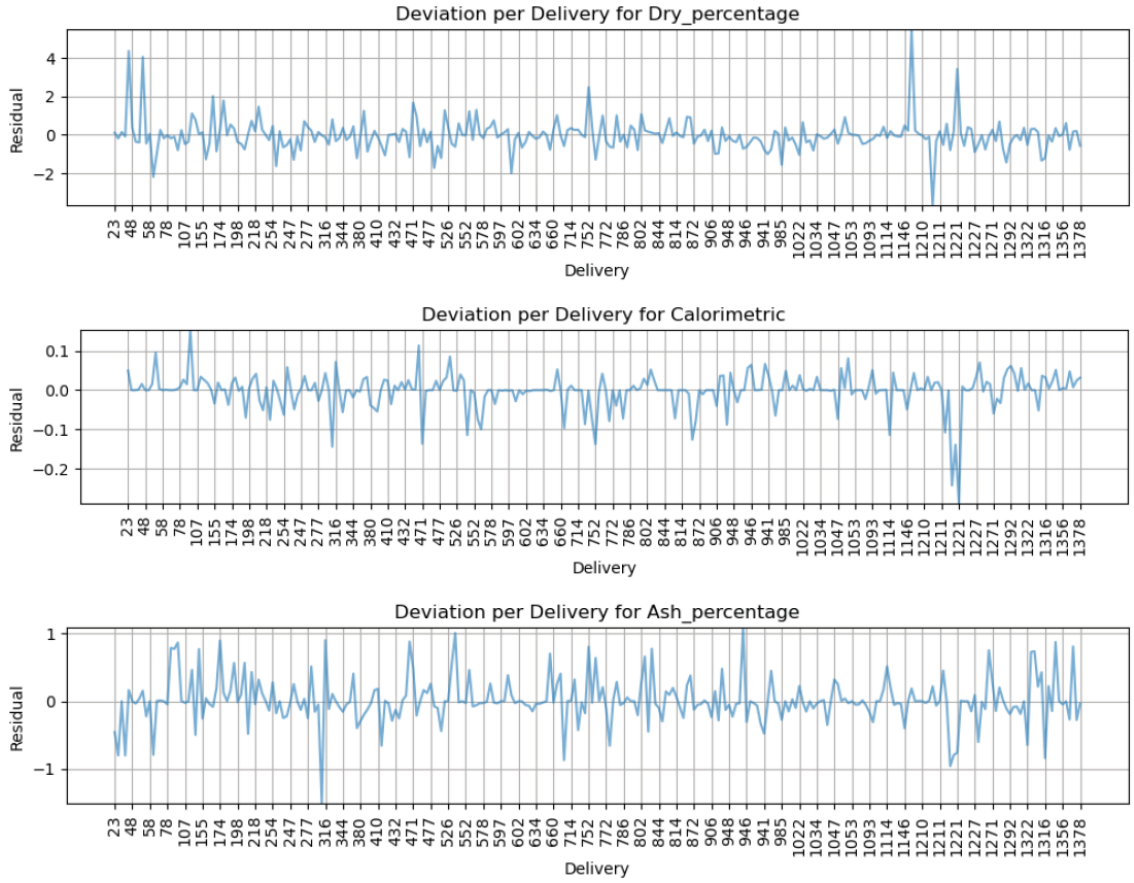


Figure A.12: Plots displaying residuals for each parameter using RF.

B

Jupyter Notebooks

B.1 Boiler Load

Boiler Load

Using machine learning to estimate maximum boiler load.

Import Libraries

Importing necessary libraries and modules.

In [1]:
```
# Useful libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Useful scikit-learn functions
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import make_scorer, mean_pinball_loss

# Machine learning model
from lightgbm import LGBMRegressor
```

Data

Load Data

Loading data as pandas DataFrames. Defining input parameters and output.

In [2]:
```
# Load data
df_25 = pd.read_csv("Data/2526.csv", sep=";", decimal=",")
df_24 = pd.read_csv("Data/2425.csv", sep=";", decimal=",")
df_23 = pd.read_csv("Data/2324.csv", sep=";", decimal=",")
df_22 = pd.read_csv("Data/2223.csv", sep=";", decimal=",")
df_21 = pd.read_csv("Data/2122.csv", sep=";", decimal=",")
df = pd.concat([df_25, df_24, df_23, df_22, df_21], ignore_index=True)

# Define parameters
var1 = "Rotor"
var2 = "Turbine"
var3 = "Fluegasfan"
var4 = "Motor_bear"
var5 = "Drive_bear"
var6 = "Prim_temp1"
var7 = "Prim_temp2"
var8 = "Prim_temp3"
var9 = "H2O"
var10 = "CO"          # Removed
var11 = "Emission"     # Removed
var12 = "Moisture"    # Removed
var13 = "Bedtemp_diff"
var14 = "Bedtemp_avg"

out1 = "Boilerload"

parameters = [var1, var2, var3, var4, var5, var6, var7, var8, var9, var10, var11, var12, var13, var14, out1]
```

Load Limits

Loading and defining limits for each parameter. Set to specified values or maximum/minimum of historical data.

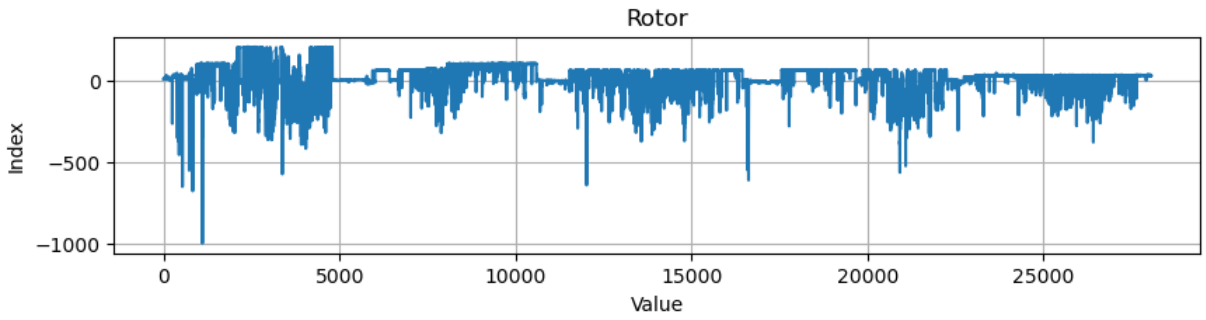
In [3]:
```
# Load limits
defined_limits = pd.read_csv("Data/limits.csv", sep=";", decimal=",")
low_limits = defined_limits[["Name", "L"]].set_index("Name")
high_limits = defined_limits[["Name", "H"]].set_index("Name")

# Define low limits
for par in low_limits.index.values:
    if pd.isna(low_limits.loc[par]).any():
        low_limits.loc[par] = df[par].min()

# Define high limits
for par in high_limits.index.values:
    if pd.isna(high_limits.loc[par]).any():
        high_limits.loc[par] = df[par].max()
```

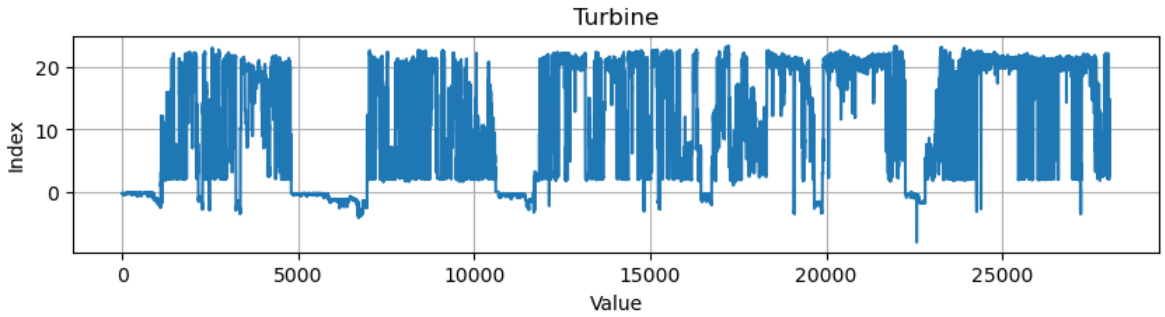
Pre-process Data

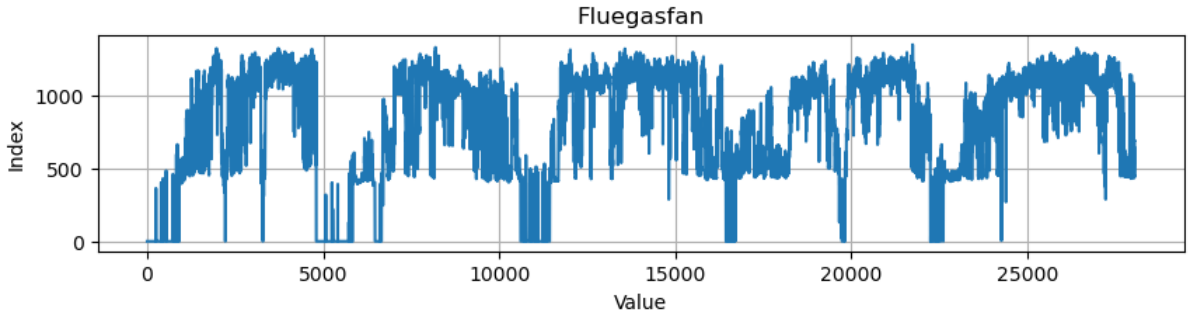
Plot Data

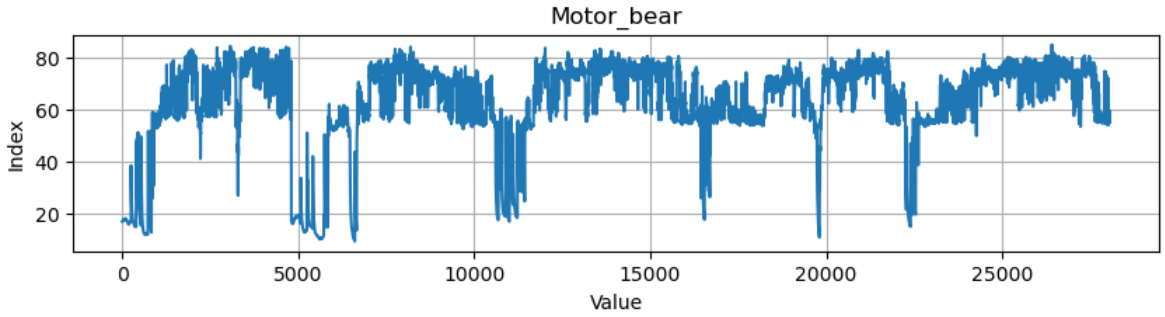
Plotting data for evaluation.

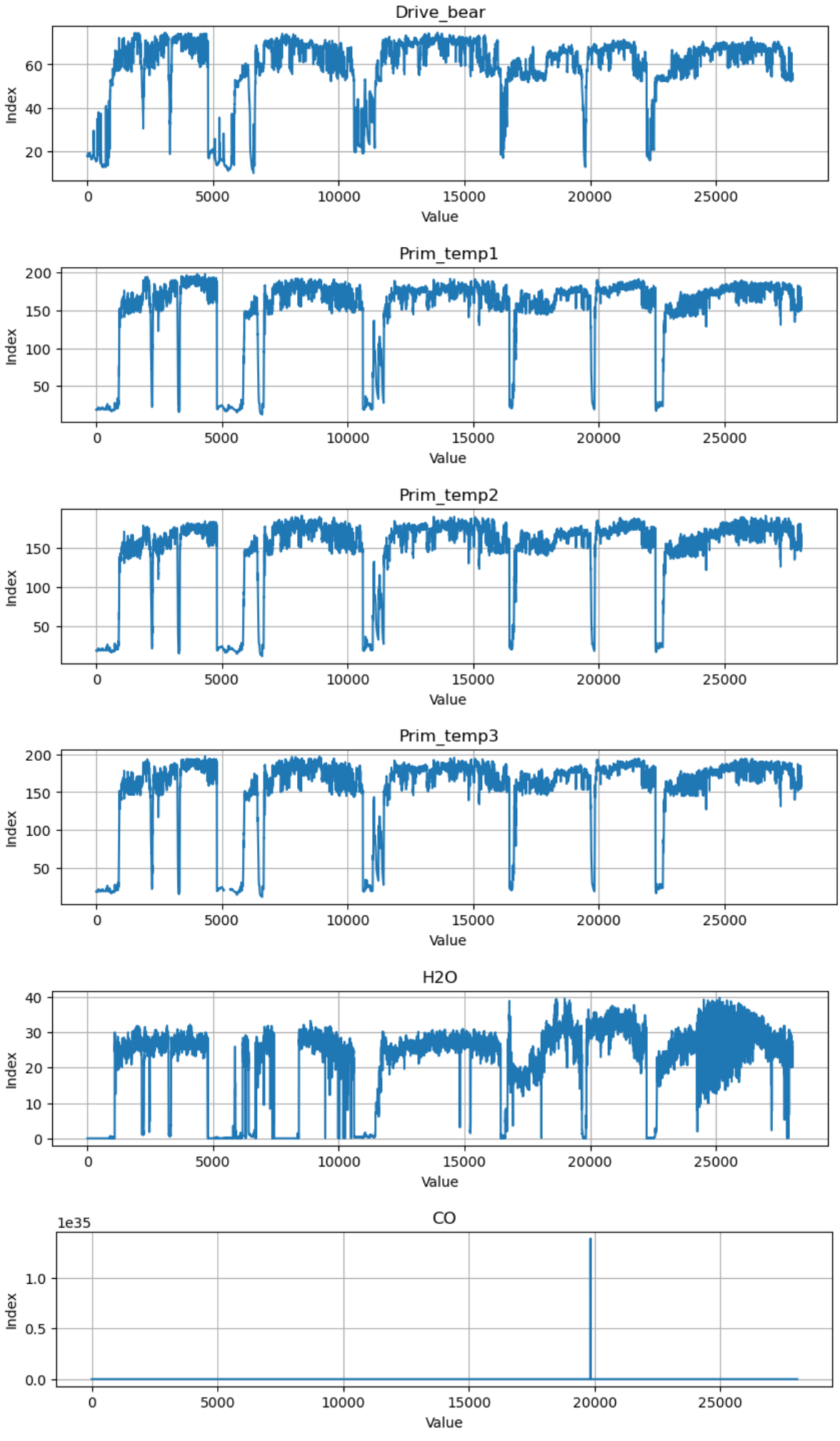
In [4]:
```
for par in parameters:
    plt.figure(figsize=(10, 2))
    plt.plot(df[par])
    plt.xlabel("Value")
    plt.ylabel("Index")
    plt.title(str(par))
    plt.grid(True)
    plt.show()
```

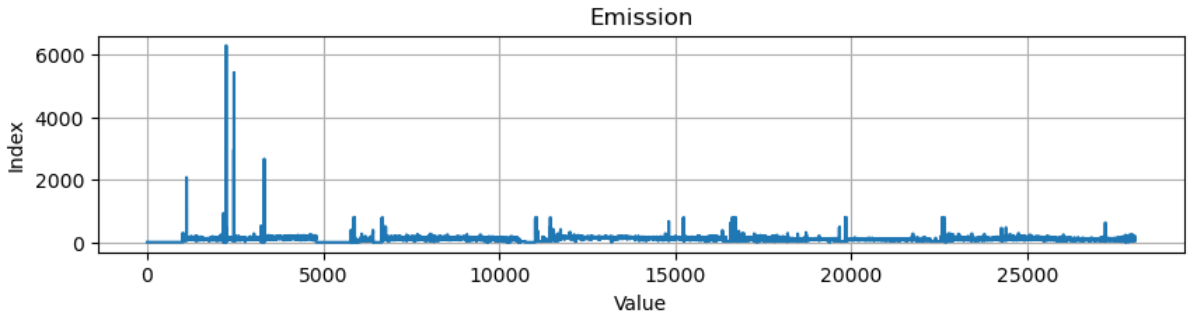


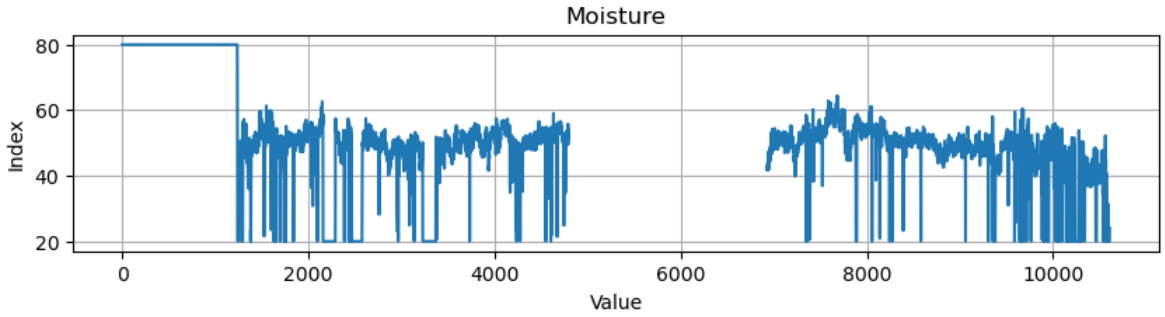


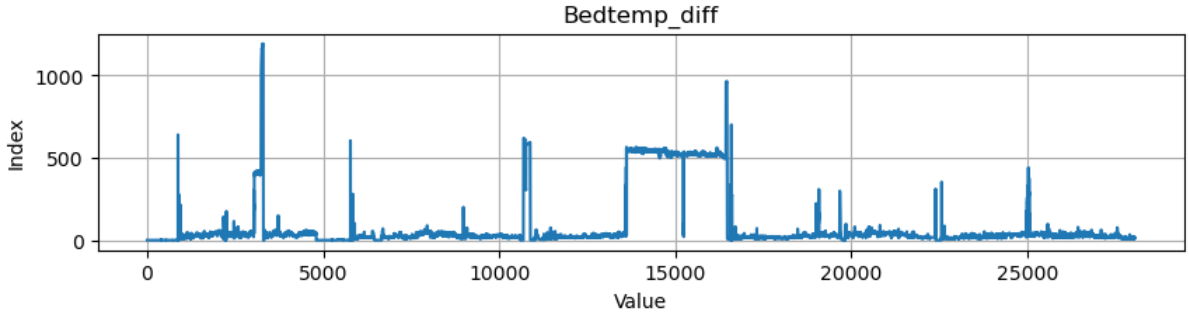


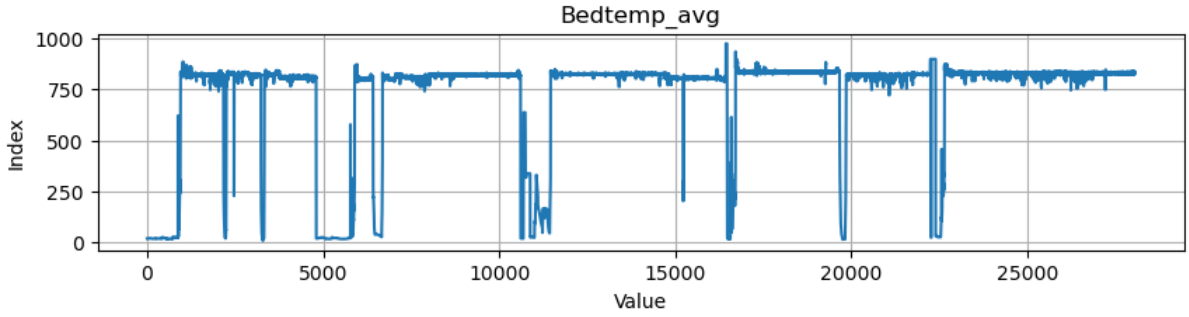


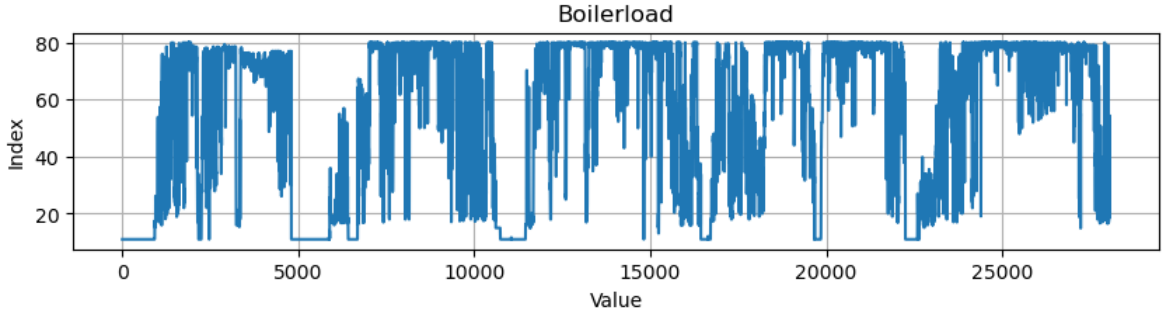

Drive_bear
Index
Value
Prim_temp1
Index
Value
Prim_temp2
Index
Value
Prim_temp3
Index
Value
H2O
Index
Value
CO
1e35
Index
Value











Print parameter stats

Printing min, max and mean values of each parameter.

In [5]:
```
par_df = pd.DataFrame(columns=["Parameter", "Min", "Max", "Mean"])
for par in parameters:
    par_min = np.min(df[par])
    par_max = np.max(df[par])
    par_mean = np.mean(df[par])
    par_df.loc[len(par_df)] = [par, par_min, par_max, par_mean]

print(par_df)
```

	Parameter	Min	Max	Mean
0	Rotor	-998.524000	2.052936e+02	1.183056e+01
1	Turbine	-8.113686	2.348363e+01	1.094310e+01
2	Fluegasfan	0.000000	1.351142e+03	7.861422e+02
3	Motor_bear	9.162680	8.493198e+01	6.222782e+01
4	Drive_bear	9.904990	7.466051e+01	5.849817e+01
5	Prim_temp1	11.910240	1.981535e+02	1.509895e+02
6	Prim_temp2	11.988360	1.906958e+02	1.462050e+02
7	Prim_temp3	11.571990	1.974578e+02	1.543712e+02
8	H2O	0.000000	3.975068e+01	2.136405e+01
9	CO	-19672.750000	1.390000e+35	4.954377e+30
10	Emission	0.000000	6.293111e+03	1.065682e+02
11	Moisture	20.000000	7.999995e+01	5.114754e+01
12	Bedtemp_diff	0.422847	1.194000e+03	8.723984e+01
13	Bedtemp_avg	9.554084	9.747814e+02	7.148543e+02
14	Boilerload	11.000000	8.000000e+01	5.606419e+01

Remove NaNs

Removing parameters with lacking data and rows with NaNs.

In [6]:
```
# Define inputs and outputs
inputs = [var1, var2, var3, var4, var5, var6, var7, var8, var9, var13, var14]
outputs = out1
df = df[inputs + [outputs]]

# Remove NaNs
print(f"Number of samples before pre-processing: {len(df)}")
df = df.dropna()
#df = df[df[outputs] >= 40]     # Can be used to remove low boiler loads
print(f"Number of samples after pre-processing: {len(df)}")
```

```
Number of samples before pre-processing: 28056
Number of samples after pre-processing: 27796
```

Split Data

Dividing the data into training and test sets. The ratio is set to 80% train set and 20% test set.

In [7]:
```
# Create training and test sets
X = df[inputs]
Y = df[outputs]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Print ratio
print(f"Total samples: {len(df)}")
print(f"Training: {len(X_train)}")
print(f"Testing: {len(X_test)}")
```

```
Total samples: 27796
Training: 22236
Testing: 5560
```

Model

Training

Defining and training a LGBM Regressor with multiple alpha values (the alpha value determines risk). Also using RandomizedSearchCV to find optimal parameters.

In [8]:
```
# Define alphas for tuning
alphas = [0.8, 0.85, 0.9, 0.95]
load_models = {}
for a in alphas:
    # Create model
    model = LGBMRegressor(
        objective='quantile',
        alpha=a,
        random_state=42,
        n_jobs=-1,
        verbosity=-1
    )

    # Define parameters for tuning
    param_dist = {
        "n_estimators": [300, 500, 800],
        "learning_rate": [0.01, 0.03, 0.05],
        "num_leaves": [15, 31, 50],
        "max_depth": [3, 5, 10],
        "min_child_samples": [3, 5, 10],
```

```
        "colsample_bytree": [0.7, 0.8, 0.9],
        "subsample": [0.7, 0.8, 0.9],
        "subsample_freq": [1, 5] }

    # Define scoring
    scorer = make_scorer(
        mean_pinball_loss,
        alpha=a,
        greater_is_better=False
    )

    # Find optimal parameters
    search = RandomizedSearchCV(
        model,
        param_dist,
        n_iter=20,
        scoring=scorer,
        cv=3,
        verbose=0,
        random_state=42,
        n_jobs=-1,
        error_score='raise'
    )

    # Train model and print chosen parameters
    search.fit(X_train, Y_train)
    model = search.best_estimator_
    load_models[a] = model
    print(search.best_params_)
```

```
{'subsample_freq': 1, 'subsample': 0.8, 'num_leaves': 50, 'n_estimators': 800, 'min_child_samples': 3, 'max_depth': 10, 'le
arning_rate': 0.03, 'colsample_bytree': 0.7}
{'subsample_freq': 1, 'subsample': 0.8, 'num_leaves': 50, 'n_estimators': 800, 'min_child_samples': 3, 'max_depth': 10, 'le
arning_rate': 0.03, 'colsample_bytree': 0.7}
{'subsample_freq': 1, 'subsample': 0.7, 'num_leaves': 31, 'n_estimators': 800, 'min_child_samples': 10, 'max_depth': 10, 'l
earning_rate': 0.05, 'colsample_bytree': 0.8}
{'subsample_freq': 1, 'subsample': 0.7, 'num_leaves': 31, 'n_estimators': 800, 'min_child_samples': 10, 'max_depth': 10, 'l
earning_rate': 0.05, 'colsample_bytree': 0.8}
```

Evaluation

Current State

Predicting the maximum load for a current state.

In [9]:
```
# Pick a current state
current_state = X_test.sample()

# Print parameters
print("\n--- PARAMETERS ---")
for param in inputs:
    val = current_state[param].values[0]
    low_limit = low_limits.loc[param,'L']
    high_limit = high_limits.loc[param,'H']
    print(f"{param:<30} {val:>6.1f}   (Low: {low_limit:>6.1f}, High: {high_limit:>6.1f})")

# Print boiler load
print("\n--- RESULTS ---")
print(f"Limits: Low: {low_limits.loc[outputs,'L']:.1f}, High: {high_limits.loc[outputs,'H']:.1f}")
print(f"Actual load: {Y_test.loc[current_state.index[0]]:.2f} MW")
for a in alphas:
    max_load = load_models[a].predict(current_state)[0]
    print(f"Recommended maximum load = {max_load:.2f} MW (a={a:.2f})")
```

```
--- PARAMETERS ---
Rotor                          60.8   (Low: -150.0, High:  205.3)
Turbine                         6.5   (Low:    0.0, High:   23.5)
Fluegasfan                    832.3   (Low:    0.0, High: 1340.0)
Motor_bear                     63.1   (Low:    9.2, High:   84.9)
Drive_bear                     62.7   (Low:    9.9, High:   74.7)
Prim_temp1                    175.0   (Low:   11.9, High:  190.0)
Prim_temp2                    168.3   (Low:   12.0, High:  190.0)
Prim_temp3                    176.7   (Low:   11.6, High:  190.0)
H2O                            26.0   (Low:    5.0, High:   40.0)
Bedtemp_diff                   20.2   (Low:  -70.0, High:  100.0)
Bedtemp_avg                   824.2   (Low:  750.0, High:  830.0)

--- RESULTS ---
Limits: Low: 11.0, High: 80.0
Actual load: 63.00 MW
Recommended maximum load = 63.08 MW (a=0.80)
Recommended maximum load = 62.98 MW (a=0.85)
Recommended maximum load = 64.54 MW (a=0.90)
Recommended maximum load = 67.03 MW (a=0.95)
```

Metrics

Evaluation using Pinball loss.

In [10]:
```python
# Calculate pinball loss for each alpha
for a in alphas:
    pinball = mean_pinball_loss(Y_test, load_models[a].predict(X_test))
    print(f"Pinball loss = {pinball:.3f} (alpha: {a})")
```

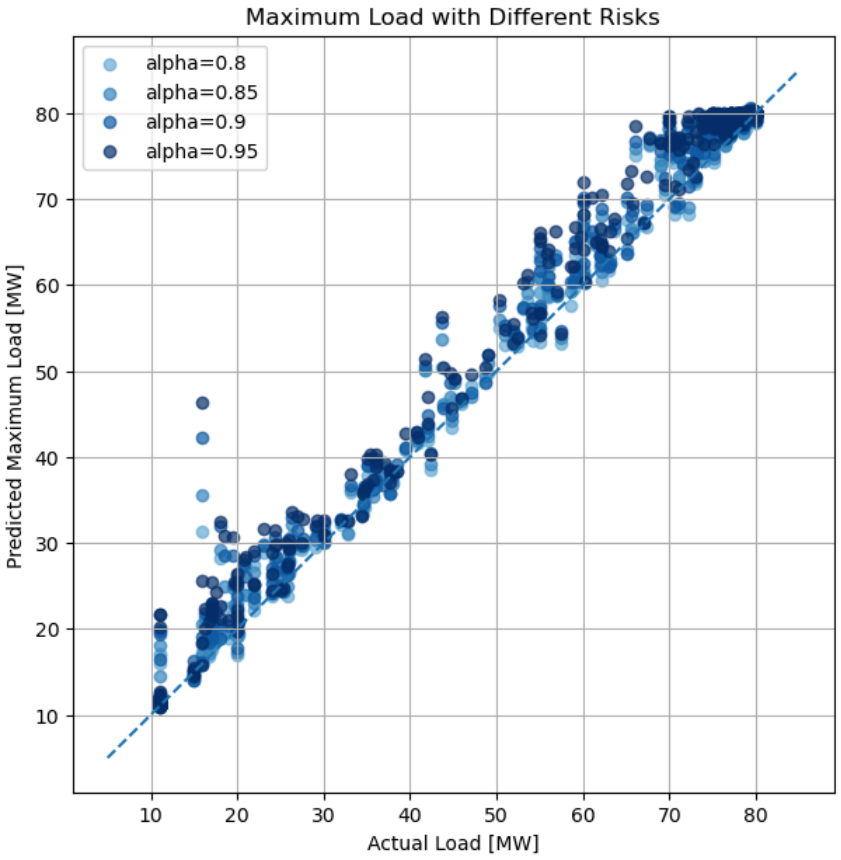
```
Pinball loss = 0.666 (alpha: 0.8)
Pinball loss = 0.776 (alpha: 0.85)
Pinball loss = 0.924 (alpha: 0.9)
Pinball loss = 1.189 (alpha: 0.95)
```

Plot

Predicted vs Actual Load

Plotting the predicted maximum load as a function of the actual current load for each alpha value.

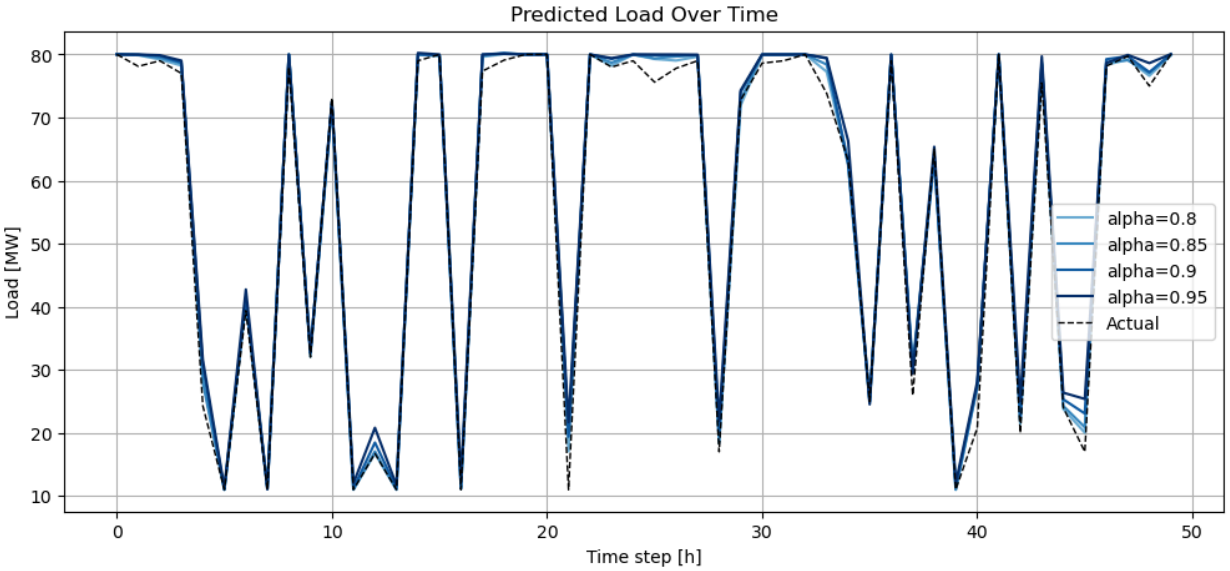
In [11]:
```python
# Choose number of samples and get actual boiler loads
n_points = 500
actuals = Y_test.iloc[:n_points]

# Predict loads for each alpha
alpha_results = {}
for a in alphas:
    max_preds = []
    for i in range(n_points):
        current_state = X_test.iloc[[i]]
        max_load = load_models[a].predict(current_state)[0]
        max_preds.append(max_load)
    alpha_results[a] = max_preds

# Create scatter plot with color depending on alphas
plt.figure(figsize=(7, 7))
cmap = plt.cm.Blues
colors = cmap(np.linspace(0.5, 1.0, len(alphas)))
for a, color in zip(alphas, colors):
    plt.scatter(actuals, alpha_results[a], alpha=0.7, color=color, label=f"alpha={a}")

# Set labels etc.
plt.plot([5, 85], [5, 85], linestyle='--')
plt.xlabel("Actual Load [MW]")
plt.ylabel("Predicted Maximum Load [MW]")
plt.title("Maximum Load with Different Risks")
plt.legend()
plt.grid(True)
plt.show()
```



Boiler Load over Time

Plotting the boiler load over time, with different alpha values.

In [12]:
```
# Generate predictions
n_points = 50
alpha_results = {a: [] for a in alphas}
for i in range(n_points):
    current_state = X_test.iloc[[i]]
    for a in alphas:
        max_load = load_models[a].predict(current_state)[0]
        alpha_results[a].append(max_load)

# Create time plot
plt.figure(figsize=(12, 5))
cmap = plt.cm.Blues
colors = cmap(np.linspace(0.5, 1.0, len(alphas)))
for a, color in zip(alphas, colors):
    plt.plot(alpha_results[a], color=color, label=f"alpha={a}")
plt.plot(Y_test.iloc[:n_points].values, color="black", linewidth=1, linestyle="--", label="Actual")

# Set labels etc.
plt.xlabel("Time step [h]")
plt.ylabel("Load [MW]")
plt.title("Predicted Load Over Time")
plt.legend()
plt.grid(True)
plt.show()
```


Predicted Load Over Time
Load [MW]
Time step [h]
alpha=0.8
alpha=0.85
alpha=0.9
alpha=0.95
Actual
80
70
60
50
40
30
20
10
0
10
20
30
40
50

In [ ]:

B.2 Fuel Data

Fuel Data

Machine learning model to detect deviations and anomalies in fuel data.

Import Libraries

Importing necessary libraries and modules.

In [1]:
```
# Useful libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Useful scikit-learn functions
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.multioutput import MultiOutputRegressor

# Machine learning models
from sklearn.ensemble import RandomForestRegressor
from lightgbm import LGBMRegressor
from xgboost import XGBRegressor
```

Data

Load Data

Loading data as pandas DataFrames and defining relevant parameters, as well as inputs and outputs. Pre-processing data to remove rows with NaNs.

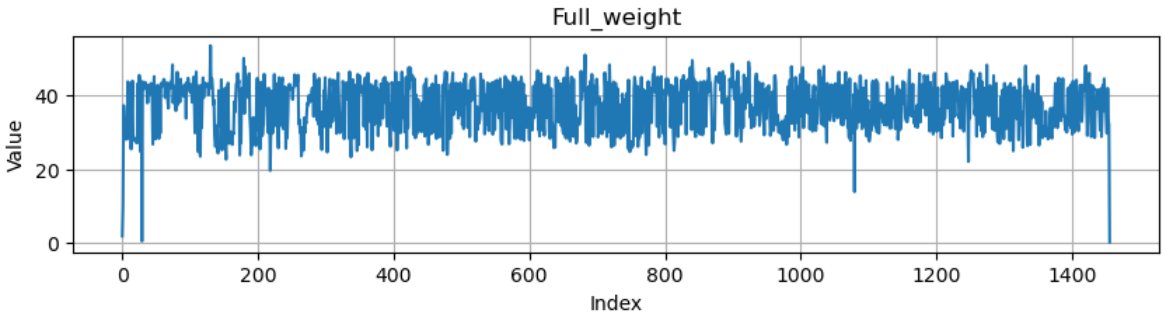
In [2]:
```
# Load data
df = pd.read_csv("Data/bransle.csv", sep=";", decimal=",")

# Define parameters
meta1 = "Delivery_name"
meta2 = "Delivery_date"

var1 = "Full_weight"
var2 = "Dry_weight"
var3 = "Energy_content"
var4 = "Chipped_volume"
var5 = "Solid_volume"

out1 = "Dry_percentage"
out2 = "Calorimetric"
out3 = "Ash_percentage"

# Define parameters
parameters = [var1, var2, var3, var4, var5, out1, out2, out3]
```

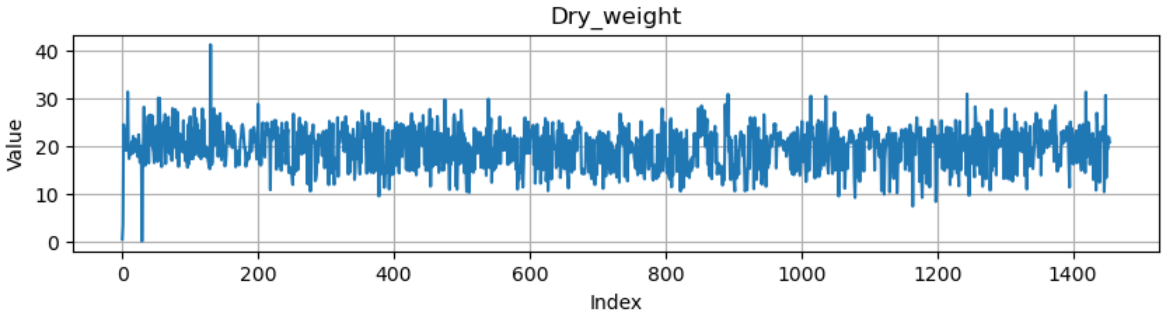
Pre-process Data

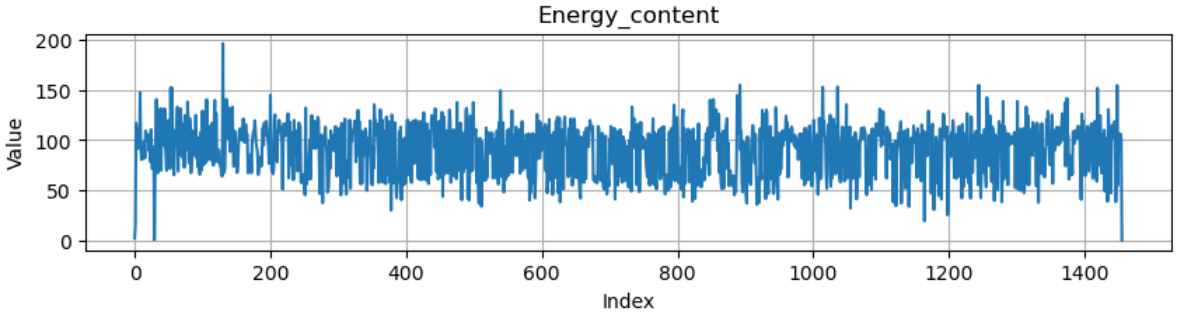
Plot Data

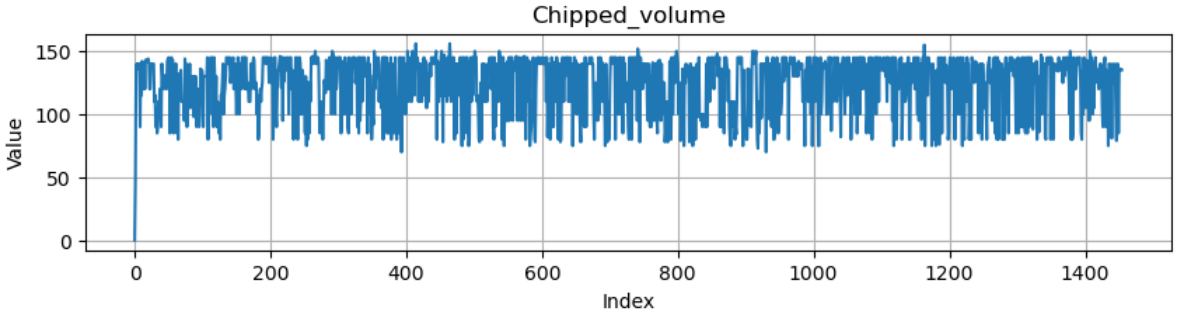
Plotting data for evaluation.

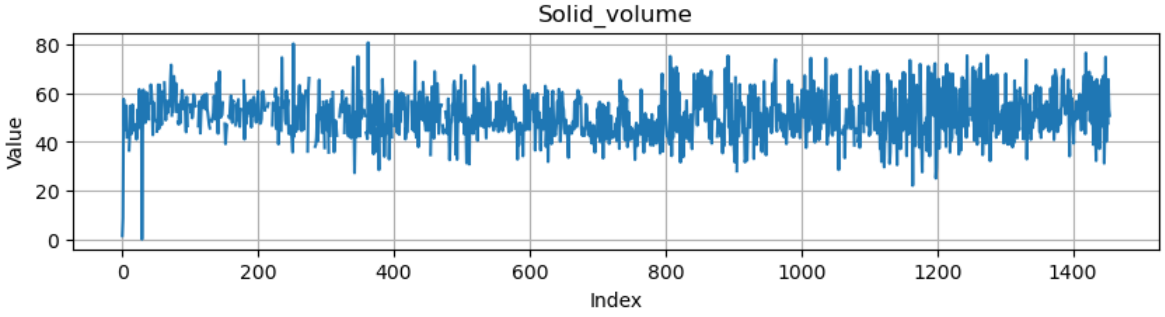
In [3]:
```
for par in parameters:
    plt.figure(figsize=(10, 2))
    plt.plot(df[par])
    plt.xlabel("Index")
    plt.ylabel("Value")
    plt.title(str(par))
    plt.grid(True)
    plt.show()
```

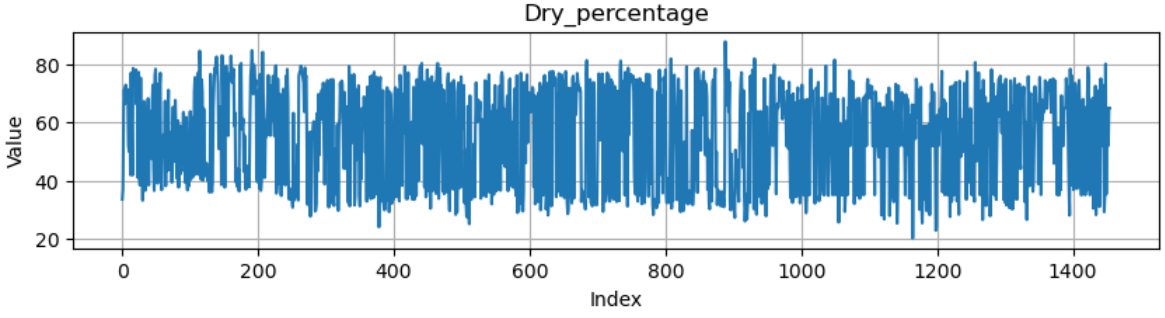

Full_weight
Value
Index

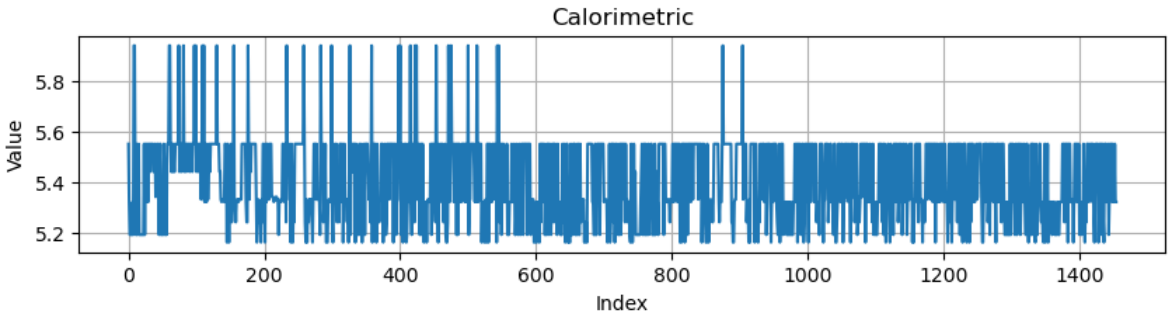

Dry_weight
Value
Index

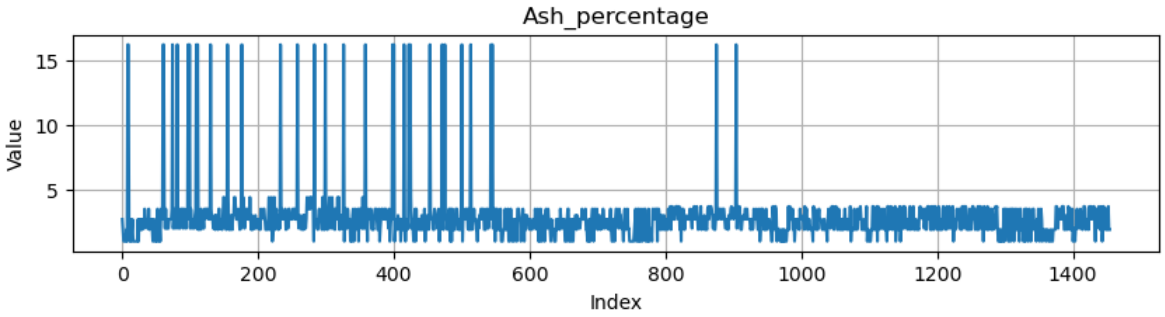

Energy_content
Value
Index


Chipped_volume
Value
Index


Solid_volume
Value
Index


Dry_percentage
Value
Index





Print parameter stats

Printing min, max and mean values of each parameter.

In [4]:
```
par_df = pd.DataFrame(columns=["Parameter", "Min", "Max", "Mean"])
for par in parameters:
    par_min = np.min(df[par])
    par_max = np.max(df[par])
    par_mean = np.mean(df[par])
    par_df.loc[len(par_df)] = [par, par_min, par_max, par_mean]

print(par_df)
```

```
       Parameter        Min      Max        Mean
0     Full_weight   0.000001   53.620   37.170371
1      Dry_weight   0.135000   41.134   19.673034
2  Energy_content   0.000004  196.290   90.903140
3  Chipped_volume   0.100000  156.000  123.725155
4    Solid_volume   0.330000   80.511   50.753725
5  Dry_percentage  20.053000   87.957   55.111141
6    Calorimetric   5.160000    5.940    5.404454
7  Ash_percentage   1.000000   16.200    2.843918
```

Remove NaNs

Removing parameters with lacking data and rows with NaNs. Defining inputs, outputs and meta data. Pre-processing by removing rows with NaNs.

In [5]:
```
# Define inputs, outputs and meta
metas = [meta1, meta2]
inputs = [var1, var2, var3, var4, var5]
outputs = [out1, out2, out3]
df = df[metas + inputs + outputs]

# Remove NaNs
print(f"Number of samples before pre-processing: {len(df)}")
df = df.dropna()
print(f"Number of samples after pre-processing: {len(df)}")
```

```
Number of samples before pre-processing: 1456
Number of samples after pre-processing: 1379
```

Split Data

Dividing data into training and test sets. The ratio is set to 80% train set and 20% test set.

In [6]:
```
# Create training and test sets
X = df[inputs]
Y = df[outputs]
meta = df[metas]
meta.loc[:, meta1] = [str(x) for x in range(len(meta))]
X_train, X_test, Y_train, Y_test, meta_train, meta_test = train_test_split(X, Y, meta, test_size=0.2, random_state=42)
```

```
# Print ratio
print(f"Total samples: {len(df)}")
print(f"Training: {len(X_train)}")
print(f"Testing: {len(X_test)}")
```

```
Total samples: 1379
Training: 1103
Testing: 276
```

Training Model

Defining and training a chosen model (LGBM, XGB or Random Forest). Also using RandomizedSearchCV to find optimal parameters.

In [7]:

```
# Select model
mod_sel = 1

# Create model and define parameters for tuning
if mod_sel == 1:
    base_model = LGBMRegressor(
         n_estimators=500,
         learning_rate=0.05,
         max_depth=-1,
         random_state=42,
         verbosity=-1
    )

    model = MultiOutputRegressor(base_model)
    param_dist = {
        "estimator__n_estimators": [300, 600, 1000],
        "estimator__learning_rate": [0.01, 0.03, 0.05],
        "estimator__num_leaves": [31, 63, 127],
        "estimator__subsample": [0.7, 0.9, 1.0],
        "estimator__colsample_bytree": [0.7, 0.9, 1.0]
    }

elif mod_sel == 2:
    base_model = XGBRegressor(
         n_estimators=400,
         learning_rate=0.05,
         max_depth=5,
         subsample=0.8,
         colsample_bytree=0.8,
         random_state=42
    )

    model = MultiOutputRegressor(base_model)
    param_dist = {
        "estimator__n_estimators": [300, 600, 1000],
        "estimator__learning_rate": [0.01, 0.03, 0.05],
        "estimator__max_depth": [3, 5, 7],
        "estimator__subsample": [0.6, 0.8, 1.0],
        "estimator__colsample_bytree": [0.6, 0.8, 1.0]
    }

else:
    base_model = RandomForestRegressor(
        n_estimators=300,
        max_depth=12,
        min_samples_leaf=5,
        random_state=42
    )

    model = MultiOutputRegressor(base_model)
    param_dist = {
        "estimator__n_estimators": [200, 500, 1000],
        "estimator__max_depth": [5, 10, 20, None],
        "estimator__min_samples_leaf": [1, 3, 5],
        "estimator__max_features": ["sqrt", "log2", None]
    }


# Find optimal parameters
search = RandomizedSearchCV(
    model,
    param_dist,
    n_iter=20,
    scoring="neg_mean_absolute_error",
    cv=3,
    verbose=0,
    random_state=42,
    n_jobs=-1,
    error_score='raise'
)
```

```
# Train model and print chosen parameters
search.fit(X_train, Y_train)
model = search.best_estimator_
print(search.best_params_)
```

```
{'estimator__subsample': 1.0, 'estimator__num_leaves': 127, 'estimator__n_estimators': 1000, 'estimator__learning_rate': 0.
03, 'estimator__colsample_bytree': 1.0}
```

Evaluation

Current State

Creating a random current state to test the model predictions.

In [8]:
```
# Pick a current state
current_state = X_test.sample()

# Print input parameters
print("\nCurrent state inputs:")
for param in inputs:
    val = current_state[param].values[0]
    print(f"{param}: {val:.1f}")

# Print actual output parameters
print("\nActual outputs:")
actual_out = Y_test.loc[current_state.index[0]]
for param in outputs:
    print(f"{param}: {actual_out[param]:.1f}")

# Print predicted output parameters
print("\nPredicted outputs:")
pred_out = model.predict(current_state)[0]
for i, param in enumerate(outputs):
    print(f"{param}: {pred_out[i]:.1f}")
```

```
Current state inputs:
Full_weight: 44.3
Dry_weight: 21.4
Energy_content: 98.7
Chipped_volume: 85.0
Solid_volume: 64.2

Actual outputs:
Dry_percentage: 48.3
Calorimetric: 5.5
Ash_percentage: 3.5

Predicted outputs:
Dry_percentage: 48.4
Calorimetric: 5.5
Ash_percentage: 3.4
```

Metrics

Evaluating the predictions using MAE, MSE and RMSE.

In [9]:
```
# Calculate MAE, MSE and RMSE
all_pred = model.predict(X_test)
for i, param in enumerate(outputs):
    y_true = Y_test[param].values
    y_pred = all_pred[:, i]

    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)

    print(f"\n{param}:")
    print(f"  MAE  = {mae:.3f}")
    print(f"  MSE  = {mse:.3f}")
    print(f"  RMSE = {rmse:.3f}")
```

```
Dry_percentage:
  MAE  = 0.475
  MSE  = 0.630
  RMSE = 0.794

Calorimetric:
  MAE  = 0.019
  MSE  = 0.001
  RMSE = 0.034

Ash_percentage:
  MAE  = 0.221
  MSE  = 0.119
  RMSE = 0.345
```

Printing

Deviations

Printing the samples with the largest deviations for each parameter.

In [10]:

```python
# Create total anomaly table
all_results = pd.DataFrame({
    "Delivery": meta_test[meta1].values,
    "Delivery Date": meta_test[meta2].values
})

# Create anomaly table for each parameter
deviation_flags = []
for i, param in enumerate(outputs):

    # Calculate residuals to determine anomaly
    y_true = Y_test[param].values
    y_pred = all_pred[:, i]
    residuals = y_true - y_pred
    anomalies = np.abs(residuals) > 2 * residuals.std()     # Threshold

    # Create anomaly table
    anomaly_df = pd.DataFrame({
        "Delivery": meta_test[meta1].values,
        "Delivery Date": meta_test[meta2].values,
        "Actual": y_true,
        "Predicted": y_pred,
        "Residual": residuals
    })
    anomaly_df = anomaly_df[anomalies]
    num_anomalies = np.sum(anomalies)
    deviation_flags.append(anomalies)
    all_results[f"{param}_residual"] = residuals

    # Print anomalies
    print(f"\n{param} (Anomalies: {num_anomalies}):")
    print(anomaly_df.sort_values(by="Residual", key=abs, ascending=False).head(3))
    anomaly_df.to_csv(f"{param}_anomalies.csv", index=False)


# Print total anomaly table
deviation_matrix = np.column_stack(deviation_flags)
all_results["# deviations"] = np.sum(deviation_matrix, axis=1)
print("\n\n---TOTAL---")
print(all_results.sort_values(by="# deviations", ascending=False).head(3))
all_results.to_csv(f"all_anomalies.csv", index=False)
```

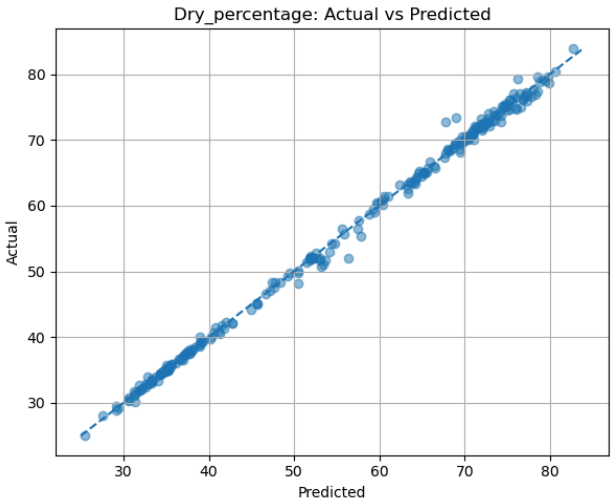
```
Dry_percentage (Anomalies: 9):
    Delivery Delivery Date  Actual  Predicted  Residual
43        51    2025-10-10  72.745  67.772257  4.972743
170     1198    2025-12-23  52.000  56.365350 -4.365350
52        49    2025-10-10  73.315  68.969876  4.345124

Calorimetric (Anomalies: 20):
    Delivery Delivery Date  Actual  Predicted  Residual
171     1219    2025-12-25    5.19   5.404619 -0.214619
244     1221    2025-12-25    5.19   5.372551 -0.182551
53       752    2025-11-24    5.16   5.291552 -0.131552

Ash_percentage (Anomalies: 22):
    Delivery Delivery Date  Actual  Predicted  Residual
267      937    2025-12-09     3.0   1.737846  1.262154
171     1219    2025-12-25     1.0   2.239006 -1.239006
130      532    2025-11-17     3.0   1.832793  1.167207


---TOTAL---
    Delivery Delivery Date  Dry_percentage_residual  Calorimetric_residual  \
225      493    2025-11-15                -2.252570               0.096381
71       322    2025-11-03                 0.268558              -0.108562
171     1219    2025-12-25                -0.942747              -0.214619

     Ash_percentage_residual  # deviations
225                -0.619282             2
71                 -1.153935             2
171                -1.239006             2
```

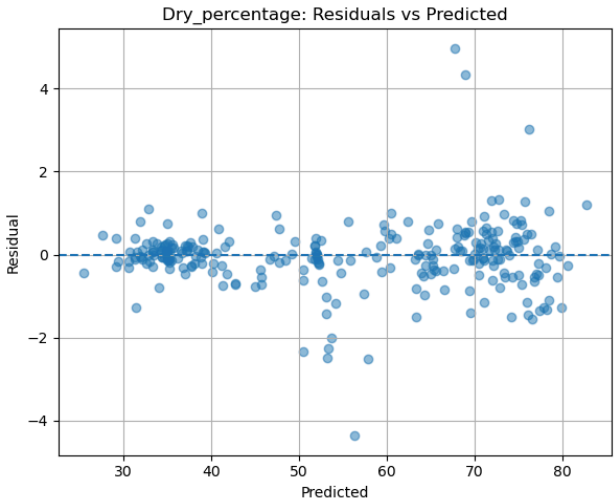
Plots

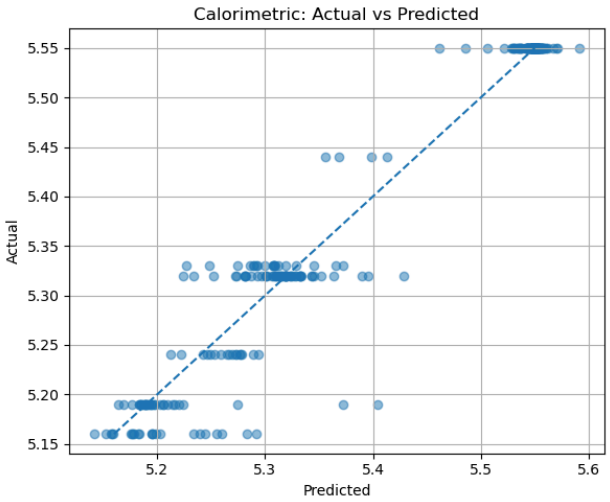
Predictions

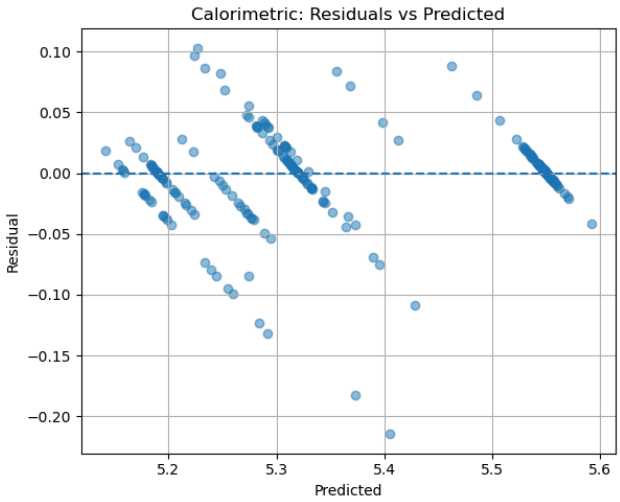
Plotting the predictions and their residuals for each parameter.

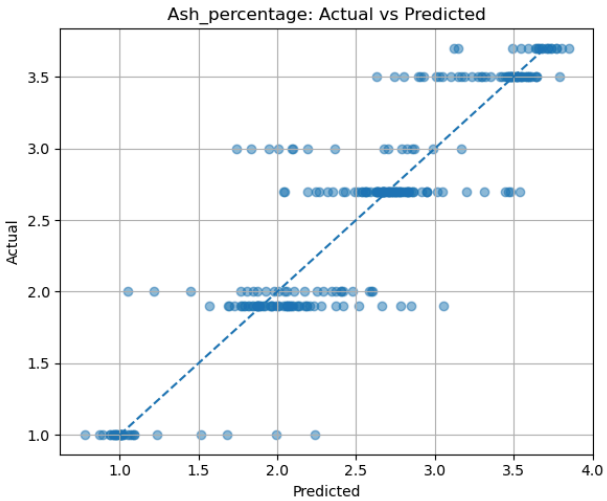
In [11]:
```python
# Create plots for each parameter
for i, param in enumerate(outputs):
    # Calculate residuals
    y_true = Y_test[param]
    y_pred = model.predict(X_test)[:, i]
    residuals = y_true - y_pred

    # Plot actual value vs predicted value
    fig, axes = plt.subplots(1, 2, figsize=(12, 5))
    axes[0].scatter(y_pred, y_true, alpha=0.5)
    axes[0].plot([y_true.min(), y_true.max()], [y_true.min(), y_true.max()], linestyle='--')
    axes[0].set_xlabel("Predicted")
    axes[0].set_ylabel("Actual")
    axes[0].set_title(f"{param}: Actual vs Predicted")
    axes[0].grid(True)

    # Plot residual vs predicted value
    axes[1].scatter(y_pred, residuals, alpha=0.5)
    axes[1].axhline(0, linestyle='--')
    axes[1].set_xlabel("Predicted")
    axes[1].set_ylabel("Residual")
    axes[1].set_title(f"{param}: Residuals vs Predicted")
    axes[1].grid(True)

    plt.tight_layout()
    plt.show()
```

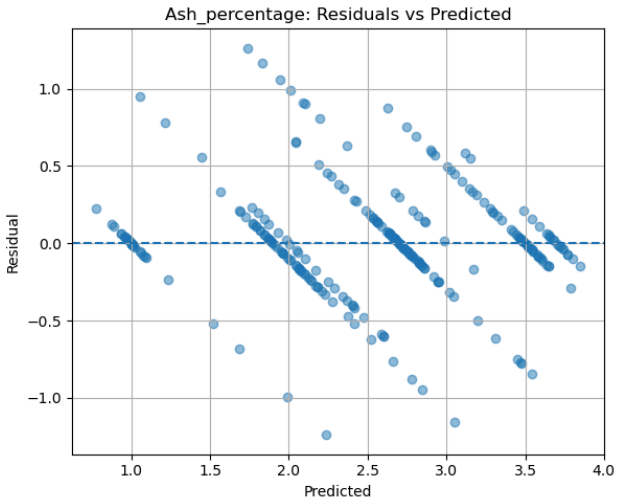










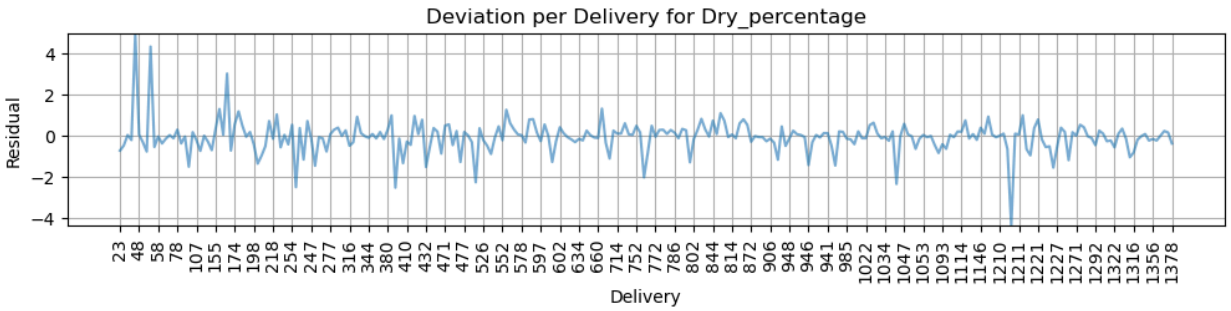


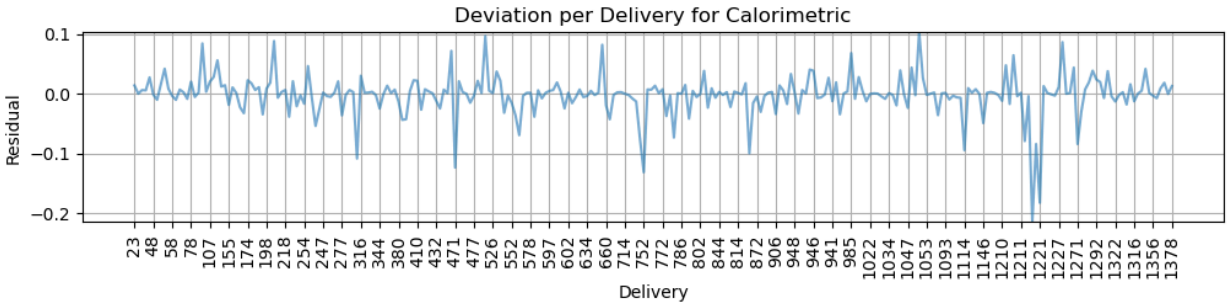
Deviation by date

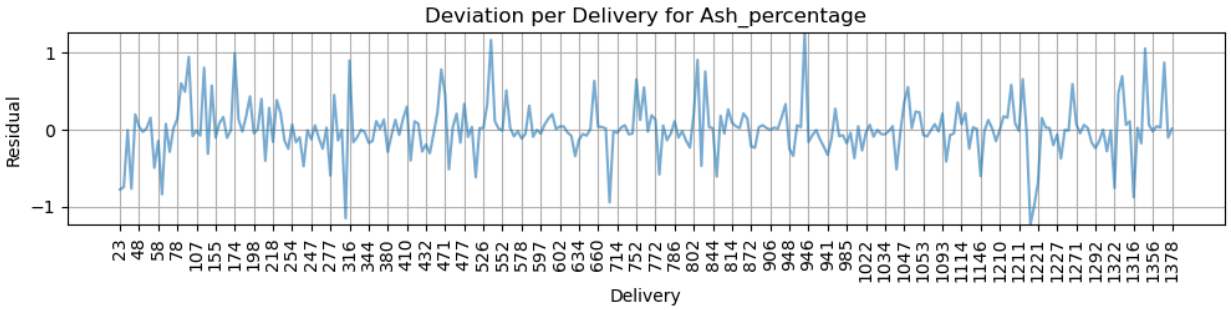
Plotting deviation for all parameters as a function of the delivery, sorted by date.

In [12]:
```
# Define number of samples and sort the dataset
num_samples = len(all_results)
all_results = all_results.sort_values(by="Delivery Date", ascending=True)

# Create plot
for param in outputs:
    plt.figure(figsize=(12, 2))

    # Plot deviations
    x_vals = all_results["Delivery"]
    y_vals = all_results[f"{param}_residual"]
    plt.plot(x_vals[:num_samples], y_vals[:num_samples], alpha=0.6)

    # Set plot settings
```

```
    plt.xticks(x_vals[::5], rotation=90)
    plt.xlabel("Delivery")
    plt.ylabel("Residual")
    plt.ylim(np.min(y_vals), np.max(y_vals))
    plt.title(f"Deviation per Delivery for {str(param)}")
    plt.grid(True)
    plt.show()
```







In [ ]:

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY