Master's Thesis 2023 30 ECTS
Faculty of Science and Technology

Prediction of nitrogen purification in wastewater with Machine learning

Sebastian Tobias Becker
MSc Data Science

Abstract

Wastewater treatment plants are necessary for avoiding environmental pollution by humans. Last year the European Commission proposed a new directive with stricter requirements for wastewater treatment plants [1]. To meet the proposed regulatory changes regarding the allowed amount of pollution, many wastewater treatment plants need expensive facility upgrades. These upgrades may increase land use. Additionally, the taxpayers will most likely have to pay for the expenses related to meet the new requirements for the wastewater treatment plants. One possible solution for reducing the cost and land use could be to optimize the processes used today with new technology. This study will investigate if it is possible to use machine learning to predict the amount of nitrate contained in the wastewater after denitrification. For this purpose, historical data from two different denitrification processes from one wastewater treatment plant is utilized. The first process dosed methanol based on measurements of nitrate, oxygen, and flow before denitrification, while the second process dosed methanol based on measurements of nitrate, oxygen, and flow before denitrification and previous nitrate out measurements. The data were collected between 30.11.2022 and 05.01.2023. One statistical approach and two machine learning models were tested for predicting the amount of nitrate contained in the wastewater after denitrification. The statistical method is a seasonal autoregressive integrated moving average with exogenous variables (SARIMAX) and the machine learning approaches are the long short term memory (LSTM) and extreme gradient boosting (XGBoost) algorithms. For the first process all models showed similar results with SARIMAX as the best model with an MSE, RMSE and MAE of 0.15, 0.39 and 0.29 respectively. For the second process the SARIMAX model outperformed the LSTM and XGBoost with MSE,RMSE and MAE of 2.09, 1.45 and 1.24 respectively. Our research show that it is significantly easier to get good performing models for process one than two. We are presenting some aspects which should be further investigated to obtain a solution that is ready to be put into use.

Sammendrag

Renseanlegg er nødvendig for å unngå miljøforurensninger fra mennesker. I fjor foreslo EU-kommisjonen et nytt direktiv med strengere krav til renseanlegg [1]. For å møte de foreslåtte reguleringene angående tillatt mengde forurensning, trengs kostbare oppgraderinger av mange anlegg. Disse oppgraderingene kan føre til økt arealbruk. I tillegg vil mest sannsynlig skattebetalerne måtte betale for utgiftene knyttet til å oppfylle de nye kravene. En mulig løsning for å redusere kostnadene og arealbruk kan være å optimalisere dagens prosesser med ny teknologi. I denne oppgaven vil det undersøkes om det er mulig å bruke maskinlæring til å predikere mengden nitrat som er igjen i avløpsvannet etter denitrifikasjon. For dette formålet er historiske data fra to ulike denitrifikasjonsprosesser fra et renseanlegg benyttet. I den første prosessen er metanoldoseringen basert på målinger av nitrat, oksygen og strømning av avløpsvann før denitrifikasjon. I den andre prosessen doseres metanol basert på målinger av nitrat, oksygen, strømning av avløpsvann samt tidligere nitrat ut målinger. Dataen ble samlet inn mellom 30.11.2022 og 05.01.2023. En statistisk tilnærming og to maskinlæringsmodeller ble testet for å predikere mengden nitrat som er i avløpsvannet etter denitrifikasjon. Den statistiske metoden som ble brukt er en SARIMAX modell og maskinlæringsmetodene er LSTM og XGBoost. For den første prosessen viste alle modellene lignende resultater, hvor SARIMAX var den beste med en MSE, RMSE og MAE på henholdsvis 0.15, 0.39 og 0.29. For den andre prosessen var SARIMAX den klart beste modellen med en MSE, RMSE og MAE på henholdsvis 2.09, 1.45 og 1.24. Disse resultatene indikerer at det er lettere å lage gode modeller for den første prosessen enn den andre. Til slutt presenteres noen aspekter som bør undersøkes videre for å få en løsning som kan tas i bruk.

Acknowledgements

This thesis represents the final milestone of my master's degree at Norwegian University of Life Sciences (NMBU) and symbolizes the end of a significant chapter in my life. I have had an amazing time at NMBU.

I express my gratitude to my supervisor, Hans Ekkehard Plesser, for his guidance and insightful discussions.

I would also like to thank my supervisor at Sintef, Sølve Eidnes, for his valuable guidance and engaging discussions. Additionally, I am grateful to Sintef and Sølve Eidnes for providing me with the opportunity to work on this exciting problem statement.

Special thanks go to Veas, particularly Hilde Johansen and Anne-Kari Marsteng, for answering all my questions regarding wastewater treatment.

Lastly, I want to thank my girlfriend, friends, and family for their constant support.

Ås, 14.05.2023
Sebastian Tobias Becker

Contents

1 Introduction **9**

1.1 Context . . . 9

1.2 Thesis objective and motivation . . . 10

2 Theory **11**

2.1 Description of the wastewater treatment plant . . . 11

2.1.1 The biological treatment step . . . 12

2.1.2 Methanol dosage strategies . . . 13

2.2 Machine Learning . . . 13

2.2.1 Enabling computers to learn from data . . . 13

2.2.2 Learning strategies in ML . . . 13

2.3 Artificial neural network . . . 15

2.3.1 Single-layer neural network . . . 15

2.3.2 Multilayer artificial neural networks . . . 17

2.3.3 Recurrent neural network . . . 19

2.3.4 Long short-term memory . . . 21

2.4 Ensemble learning . . . 22

2.4.1 Extreme gradient boosting . . . 22

2.5 Time series forecasting . . . 24

2.5.1 Time series . . . 24

2.5.2 Stationarity and differencing . . . 24

2.5.3 Autoregressive . . . 25

2.5.4 Moving average . . . 25

2.5.5 Autoregressive integrated moving averge . . . 25

2.5.6 Seasonal autoregressive integrated moving averge with exogenous variables 26

2.6 Workflow of machine learning . . . 26

2.7 Overfitting vs underfitting . . . 28

2.8 Performance metrics for regression . . . 29

2.9 Evalution protocols for machine learning . . . 29

2.9.1 Hold out method . . . 30

2.9.2 K-fold cross validation . . . 30

3 Method **31**
3.1 The data sets 31
3.1.1 Features and target 31
3.2 Pre-processing and data analysis 32
3.3 Method Selection 33
3.4 Model implementation and development 34
3.4.1 SARIMAX 34
3.4.2 LSTM 35
3.4.3 XGBoost 36
3.5 Baseline 37
3.6 Software 37

4 Results **38**
4.1 Data exploration 38
4.2 Results on methanol dosage without feedback 41
4.2.1 XGBoost 41
4.2.2 LSTM 42
4.2.3 SARIMAX 43
4.2.4 Summary of the results when methanol was dosed without feedback . . 44
4.3 Results on methanol dosage with feedback 44
4.3.1 XGBoost 44
4.3.2 LSTM 46
4.3.3 SARIMAX 46
4.3.4 Summary of the results when methanol was dosed with feedback 47

5 Discussion **48**
5.1 The data sets 48
5.2 Differences between the process when methanol is dosed with and without feedback 49
5.3 Struggles with feedback 49

6 Conclusion and recommendations **51**
6.1 Conclusion 51
6.2 Recommendations 51

List of Figures

2.1 Building blocks of a wastewater treatment plant 12
2.2 Biological treatment in wastewater treatment plant 12
2.3 Adaline algorithm 15
2.4 Learning rate 17
2.5 Fully connected multilayer perceptron 18
2.6 Recurrent neural network 20
2.7 Long short term memory cell 21
2.8 Structure of a tree model 23
2.9 SARIMAX model 26
2.10 Under- and overfitting 28

3.1 Transforming a dataset to a 3D tensor 35

4.1 Visualization of nitrate out values 39
4.2 Pearson correlation 40
4.3 acf plot, both processes 41
4.4 XGBoost predictions, without feedback 41
4.5 Important features XGBoost, without feedback 42
4.6 LSTM predictions, without feedback 43
4.7 SARIMAX predictions, without feedback 44
4.8 XGBoost predictions, with feedback 45
4.9 Important features XGBoost, with feedback 45
4.10 LSTM predictions, with feedback 46
4.11 SARIMAX predictions, with feedback 47

List of Tables

3.1 Hyperparameters tested in *auto_arima* 35
3.2 Hyperparameters tested in *GridSearch* 36
3.3 Hyperparameters tested in *GridSearchCV* 37
3.4 Python libraries used 37

4.1 XGBoost parameters, without feedback 42
4.2 LSTM parameters, without feedback 43
4.3 Performance overview, without feedback 44
4.4 XGBoost parameters, with feedback 45
4.5 LSTM parameters, with feedback 46
4.6 Performance overview, with feedback 47

Chapter 1

Introduction

1.1 Context

In October 2022, the European Commission proposed a new directive for wastewater treatment [1]. Its primary aim is to regulate the emission of wastewater from municipalities in urban areas. The wastewater treatment directive primarily regulates the emission of organic material and nutrient salts from wastewater treatment plants, to prevent overfertilization in the ocean, rivers etc. [2]. The existing directive applies for wastewater treatment plants in densely populated areas with organic load of 2000 person equivalents (pe) to fresh water or 10 000 pe to sea water [2].

The proposed directive will impose stricter regulations on wastewater treatment plants in urban municipalities. According to Norwegian law, wastewater treatment plants in urban areas are subjected to a purification requirement of 70% of the amount of nitrate entering the treatment plant [3]. One of the proposed changes in the directive is to increase the nitrogen removal rate 85% [4].

According to the applicable directive, there are two mandatory steps for wastewater treatment plants. The first step is to remove material waste in the wastewater, this is referred to as the primary step [2]. The secondary step is the chemical or biological treatment of the wastewater for removing water soluble organic material [2]. The nitrogen removal is an essential part of the secondary step. Nitrogen removal can be obtained in several ways. One common method is to use biological filters where bacteria converts ammonia to nitrate by nitrification, and then nitrate to nitrogen gas, and this process is called denitrification.

Nitrogen is a primary nutrient for the growth of plants, but too high concentration of nitrogen can be harmful for nature [5]. Too high levels of nitrogen in water can lead to extensive growth of algae. The overgrowth in algae can lead to eutrophication. Eutrophication is overgrowth of plants in water and leads to less biodiversity [5]. Under normal conditions the algae serve as food for living organisms and strengthening the biodiversity. Furthermore, temperature plays a crucial role in the growth of algae. The production of algae is higher during summer compared to the winter [6]. One effect of global warming is the rise in water temperature. This again can lead to eutrophication if there is too much nitrogen in the water [6]. The arguments presented above are the main arguments for why it is important to remove the nitrogen from the wastewater before it is released back into rivers, lakes etc.

1.2 Thesis objective and motivation

Veas is the largest wastewater treatment plant in Norway and is owned by the municipalities Oslo, Bærum and Asker. The facility is located in Slemmestad and is responsible for treating the wastewater from the the owners and Nesodden municipality. Veas is a decisive contributor for keeping the Oslofjord clean. One of the measures to keep the Oslofjord clean is the denitrification of the wastewater. Veas adds methanol to the wastewater as a carbon-source for the denitrification process. The goal of Veas is to predict the amount of nitrate contained in the wastewater after denitrification under different conditions. This will enable them to test different strategies for adding methanol and see how it influences the denitrification.

The reason why Veas wants to predict the amount of nitrate contend in the wastewater after denitrification is to optimize their process. By optimizing their process, they are hoping to reduce the amount of methanol added to the denitrification process but still comply with the purification regulations. By lowering the consumption of methanol, Veas expects to lower their cost of operation. Another reason is to be better equipped for the future with population growth and increasing supply of wastewater to the treatment plant.

The first steps for making a model predicting the amount of nitrate after denitrification, is to investigate whether it is possible to predict the amount of nitrate after denitrification with use of machine learning. Therefore, the aim of this master thesis is to investigate what type of data and which machine learning algorithms are suitable for predicting the purification degree of nitrate. To do so this thesis adresses the following research questions:

1. Which models are suitable for predicting the amount of nitrate contained in the wastewater after denitrification?
2. Which features are important for predicting the amount of nitrate contained in the wastewater after denitrification?
3. Is it more difficult to predict the amount of nitrate contained in the wastewater after denitrification when the methanol is added with or without feedback?

Chapter 2

Theory

2.1 Description of the wastewater treatment plant

The process described in this section is specific to Veas and is therefor not generalizable to all wastewater treatment plants.

The water entering the sewerage system consists primarily of wastewater and storm water. Households and industry are the primary sources of wastewater while storm water comes from rain or melted snow. From now on, both wastewater and storm water will be denoted as wastewater. The main goal of a wastewater treatment plant is to remove nutrients and organic matter from the wastewater in accordance with governmental demands before releasing treated wastewater to the fjord.

Figure 2.1 shows the treatment steps of a wastewater treatment plant in chronological order. The primary goal of the grate is to remove physical waste, such as cotton buds, sanitary pads, wet wipes, plastic, and other waste. Heavy particles like sand and coffee grounds are removed in the sand trap. The wastewater is then transported and distributed in eight process lines, each of which consists of a chemical treatment step and a biological treatment step. In the chemical treatment step chemicals are added to cause the particles in the water to bind together, and the heavy particles sink to the bottom and form sludge. The sludge is pumped out, while the water is transported to the biological treatment step. The objective of the biological treatment is to remove nitrogen from the wastewater with help of bacteria. The biological treatment step is where methanol is added, and we will therefore elaborate more on the design of this treatment step.

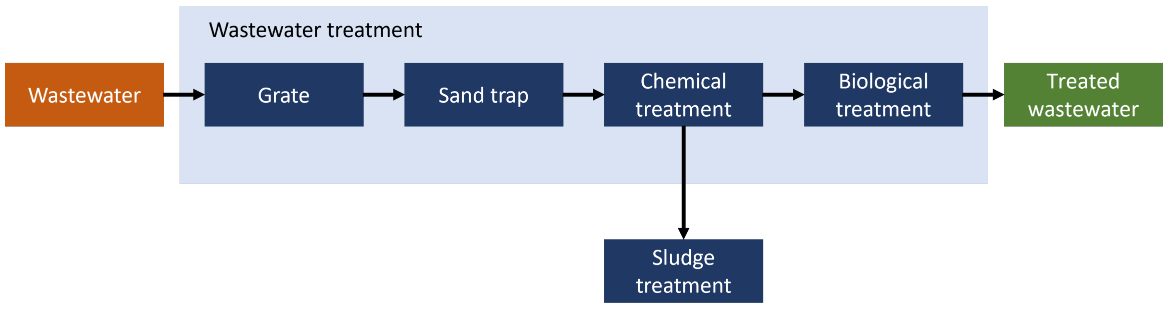


Figure 2.1: Schematic overview of the main building blocks of a wastewater treatment plant.

2.1.1 The biological treatment step

The biological treatment process is a two-step process with nitrification followed by denitrification. In the nitrification step bacteria are transferring ammonia into nitrate and in the denitrification step the bacteria transfer the nitrate into nitrogen gas. Figure 2.2 shows the flow of wastewater through the biological treatment step. Several real-time measurements are taken during the biological treatment. The measurements of most interest for the denitrification process are measurements of oxygen and nitrate concentrations. Methanol is added before the wastewater is pumped equally into four denitrification(DEN)-filters. The exception is when one filter is cleaned, which happens approximately once per day. Then the wastewater is pumped into the remaining filters. Methanol works as a energy source for the bacteria in the DEN-filters [7]. Real-time measurements of nitrate level is taken after the DEN-filters. The measurement is taken from a small tub which is filled with treated water from one of the four DEN-filters for fifteen minutes before another filter fills the tub for fifteen minutes. Fully treated water is then discharged into the Oslofjord.

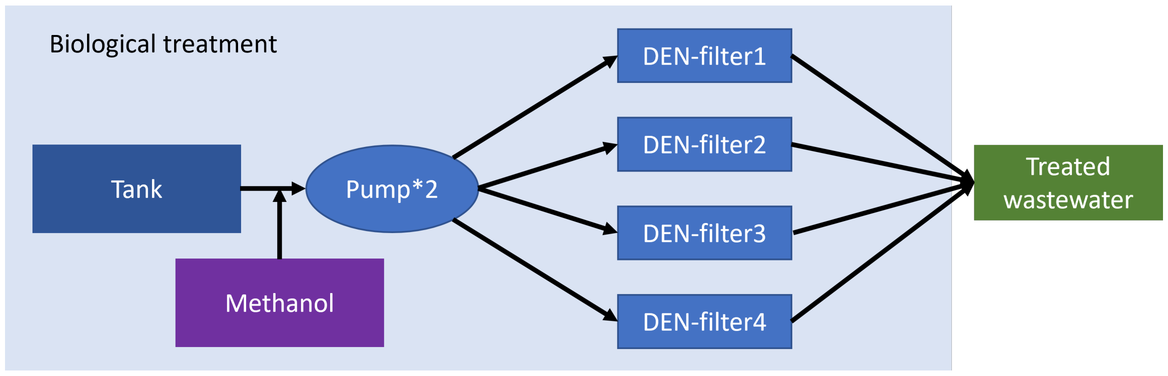


Figure 2.2: Schematic overview of the biological treatment in the wastewater treatment plant.

The denitrification step depends on several factors. Some of the most important factors are the temperature of the wastewater, available oxygen and methanol in the wastewater, amount of water going through the filters and the condition of each filter [7]. At regular intervals the DEN-filters are washed and this can affect the denitrification. For more information about the denitrification process, see [7, 8].

2.1.2 Methanol dosage strategies

In this thesis we will investigate two different approaches for how to dose methanol. The two different ways are:

1. **With feedback:** This is the most common way of adding methanol to the denitrification process. The dosage of methanol depends on four factors: nitrate and oxygen measurements in the tank before the denitrification, flow of wastewater, and nitrate measurement after denitrification. This method is called *with feedback* since previous nitrate measurement after denitrification has an influence on the methanol dosage.
2. **Without feedback:** This is an alternative dosage method. The difference is that the methanol dosage is solely dependent of nitrate and oxygen measurements in the tank before denitrification and flow of wastewater. Previous nitrate measurement after denitrification does not influence the methanol dosage, and therefore this method is called *without feedback*.

2.2 Machine Learning

2.2.1 Enabling computers to learn from data

Humans learn through a variety of methods, such as observation, trial and error, repetition, feedback, etc. Learning involves changes in the brain which lead to acquiring knowledge, experience, and behaviour. The human ability to learn gives us the possibility to retain information over time. In this way we are able to learn from the past and recall the information in the future. Machine learning tries to mimic both the human ability to learn and to recall learned information. In other words, machine learning is the science of enabling computers to learn from data. Arthur Samuel, who was one of the pioneers in Artificial intelligence (AI), defined machine learning as "The field of study that gives computers the ability to learn without being explicitly programmed" [9]. Through the development of machine learning(ML), several strategies on how to enable computers to learn have emerged.

2.2.2 Learning strategies in ML

The three learning strategies of ML are supervised learning, unsupervised learning, and reinforcement learning. The available data set and the problem domain determine which learning strategy should be utilized. Since the data set influences which learning strategy is the right one to utilize, one must first understand the different types of data sets. In the ML domain, data sets are either labelled or unlabelled. Unlabelled data sets are a collection of data samples without any form of labels. When dealing with unlabelled data the machine learning algorithm must learn these labels by itself, and this leads to a more demanding training process. Clustering is a common problem in which unlabelled data is grouped into clusters. A clustering algorithm could be utilised for grouping news articles based on their content into several clusters where all articles in one cluster have some similar attributes. Labelling data can be labour intensive and therefore cost consuming [10]. If each news article is assigned a specific label or category, then the data is labelled. Now that the difference between labelled an unlabelled data is established the next three paragraphs will elaborate on supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning

Supervised learning is a ML strategy in which an ML algorithm is trained using labelled data. The labelled data consists of features and one or several target variables. Each training sample in the training data consists of a set of features and a related target. During training, the supervised ML-algorithm learns to map features to targets by identifying patterns between features and targets. Once the model is trained it can make predictions on unseen data by using the learned pattern to predict the associated target to the unseen data. There are two main subcategories of supervised learning called classification and regression [11]. In classification we try to classify data samples to predefined discrete classes, while regression tries to predict a continuous value to a data sample. Supervised learning can be utilized in many applications such as image recognition, fraud detection, spam filters. Supervised learning is a fundamental technique in ML because of its ability to map features to targets and therefore it is an essential tool in a variety of fields [11].

Unsupervised learning

Unsupervised models are trained on unlabelled data. By using unsupervised learning we are able to extract hidden structure or patterns in our data without knowing the targets [12]. Two main methods of unsupervised learning are clustering and dimension reduction. Clustering involves dividing data into similar clusters without having any explicit information about the correct cluster association. The samples assigned to the same group have some degree of similarity and are dissimilar to samples in other groups. Dimension reduction is sometimes necessary when working with high dimensional data to limit the need for storage. It can also be necessary for compressing high dimensional data to a smaller subspace but still capturing most of the information contained in the data.

Reinforcement learning

In reinforcement learning the goal is to train an agent to interact with an environment with the aim of maximising the reward [12]. In other words, the agent learns to take actions that lead to the best possible outcome, based on the feedback from the environment. The agent receives either reward or punishment from the environment based on its actions. The agent must learn the optimal action at every stage through trial and error, it does not have access to the optimal policy, in contrast to supervised learning where the ground truth is known. Reinforcement learning is used in a wide range of applications as game playing, robotics, and autonomous driving.

2.3 Artificial neural network

Artificial neural networks (ANN) are a subcategory of machine learning, which has gained significant attention in years [11]. This section will start with explaining the feedforward network. This is done to build an understanding for the main building blocks of ANN. Once the basic concepts of ANNs are explained, the more complex recurrent neural networks (RNN), specifically long short-term memory (LSTM) will be explained. LSTM networks will be one of the primary methods used in this thesis.

2.3.1 Single-layer neural network

The foundations for ANNs were laid early in 1940s with the work of W. S. McCulloch and W. Pitts who published an article about the MccullockPits neuron [11]. Some years later F. Rosenblatt published the perceptron learning algorithm based on the work of McCulloch and Pitts [11]. Eventually in the beginning of the 1960s "Adaline" was introduced. The Adaline algorithm was an enhanced version of the perceptron. Figure 2.3 shows a schematic overview of the Adaline algorithm. The Adaline algorithm will be used for building the basic understanding for single-layer neural networks and later multi-layer neural networks.

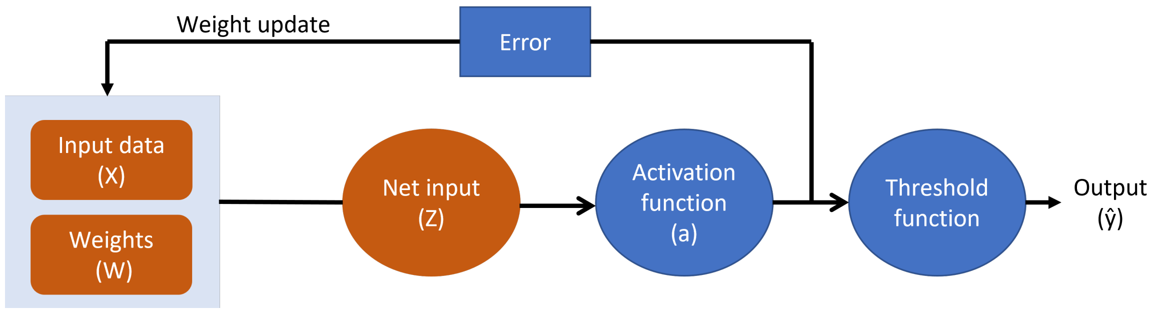


Figure 2.3: Schematic overview of the Adaline algorithm.

The Adaline algorithm is a supervised leaning algorithm. It is one of the simplest forms of a single layer ANN. The algorithm works by iteratively adjusting the weights based on a linear activation function $\phi(z)$ where z is the linear combination between the weights and feature values. The weights are initially set to small random values, and they are updated iteratively during the training phase using a gradient descent algorithm. The weights are updated during the training to minimize a cost function J which in the case of Adaline algorithm is the sum of squared errors between predicted output from the activation function and the actual output. The learning rate which controls the step size of the weighted updates is an important parameter to ensure good performance. The threshold function is used for making the final output prediction. All equations are obtained from [11, Ch. 2] and the Adaline learning rule can be summarized in the following steps:

1. Initially the weight vector(w) is initialized to small random numbers. Weights are the learnable parameters of the Adaline model.

2. The net input(z) is calculated, this is a linear combination of the weight vector w and the feature vector(x). The net input is defined as:

$$z = w_0x_0 + w_1x_1 + ... + w_mx_m, \tag{2.1}$$

   where w_0x_0 is called the bias unit and x_0 is by definition equal to 1.

3. In the case of Adaline the net input is transformed to an activation through the linear activation function $\phi(z)$, where $\phi(z) = z$. The output of the linear activation function is used for training the weights of the Adaline algorithm.

4. One essential step of the Adaline algorithm is to minimize the cost function, during training. Minimizing a cost function is a common step for supervised machine learning algorithms. The cost function for Adaline is defined as:

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_i \left(y^{(i)} - \phi\left(z^{(i)}\right)\right). \tag{2.2}$$

   The cost function of Adaline has two important properties. First, it is a convex function, and second it is differentiable. Because of these two properties one can use gradient descent to finding the weights minimizing the cost function. The main idea behind gradient decent is to take a step in the opposite direction of the gradient($\nabla J\left(\boldsymbol{w}\right)$) iteratively, until a global or local minimum is reached. The weight change is defined as:

$$\boldsymbol{w} := \boldsymbol{w} + \Delta\boldsymbol{w}, \tag{2.3}$$

   where $\Delta\boldsymbol{w}$ is the negative gradient multiplied with the learning rate (η) given by:

$$\Delta\boldsymbol{w} = -\eta\nabla\mathbf{J}(\boldsymbol{w}). \tag{2.4}$$

   To compute the gradient of the cost function, it is necessary to calculate the partial derivetive of the cost function with respect to each weight. The formula for updating weight w_j is defined as:

$$\Delta w_j = -\eta\frac{\partial J}{\partial w_j} = -\eta\sum_i \left(y^{(i)} - \phi\left(z^{(i)}\right)\right)x_j^{(i)}. \tag{2.5}$$

   In the case of Adaline all weights are updated simultaneously.

5. Step 2–4 is repeated until some criterion for stopping is meet. There are two main approaches for termination, the first one is to run the algorithm for a pre-defined number of epochs(number of training iterations). The second is to run until there is almost no change in the weights between two consecutive epochs.

6. Step 1-5 can be defined as the training process. The final output of the activation function

is passed to a threshold function. In the case of a binary classification the threshold function for Adaline is assigning the label one if the activation is greater than 0 and -1 otherwise.

The correct chose of learning rate is a crucial step in the Adaline algorithm. The learning rate impacts the convergence of the algorithm. An excessively large learning rate may result in overshooting the minimum. While a small learning rate may lead to slow convergence or being trapped in a local minimum. The different scenarios are visualised in Figure 2.4.

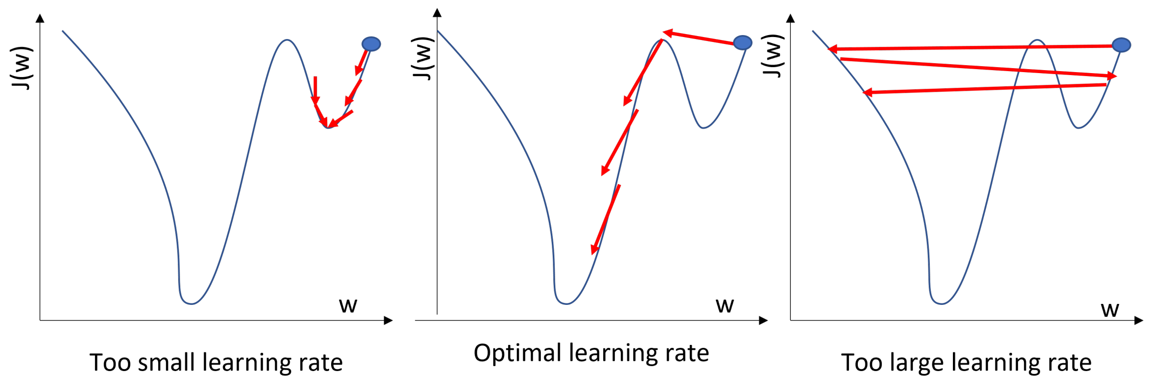


Figure 2.4: Shows how different learning rates influence the convergence of the model and where the blue dot is the first random initiation of the weights.

Feature scaling is the process of transforming features to the same scale. Machine learning algorithms utilizing gradient descent can benefit from feature scaling since the features are on the same scale. From Equation 2.5 we see that large values of $x_j^{(i)}$ have a greater impact on delta Δw_j than small values of $x_j^{(i)}$. Feature scaling ensures that all features contribute equal to Δw_j, and this may lead to faster convergence because the optimizer may need fewer steps to find the global minimum [11]. Standardization and normalization are two common feature scaling technique. Standardization normalise the features with mean equal to 0 and standard deviation of 1 while normalization scale the feature to the range (0, 1) [11]. Normalization is often called min-max-scaling.

2.3.2 Multilayer artificial neural networks

A multilayer artificial neural network contains more than one layer. In contrast, a single-layer ANN as the Adaline has only one connection between input and output layer. The three main building blocks of a multilayer artificial neural network is the input layer, hidden layer, and the output layer. These layers consist of multiple units. Multilayer neural networks can be divided into three main groups: fully connected neural networks, convolutional neural networks, and recurrent neural networks. The fully connected neural network and recurrent neural networks will be further elaborated here.

The fully connected neural network will be explained through a multilayer perceptron (MLP). An overview over the MLP is illustrated in Figure 2.5:

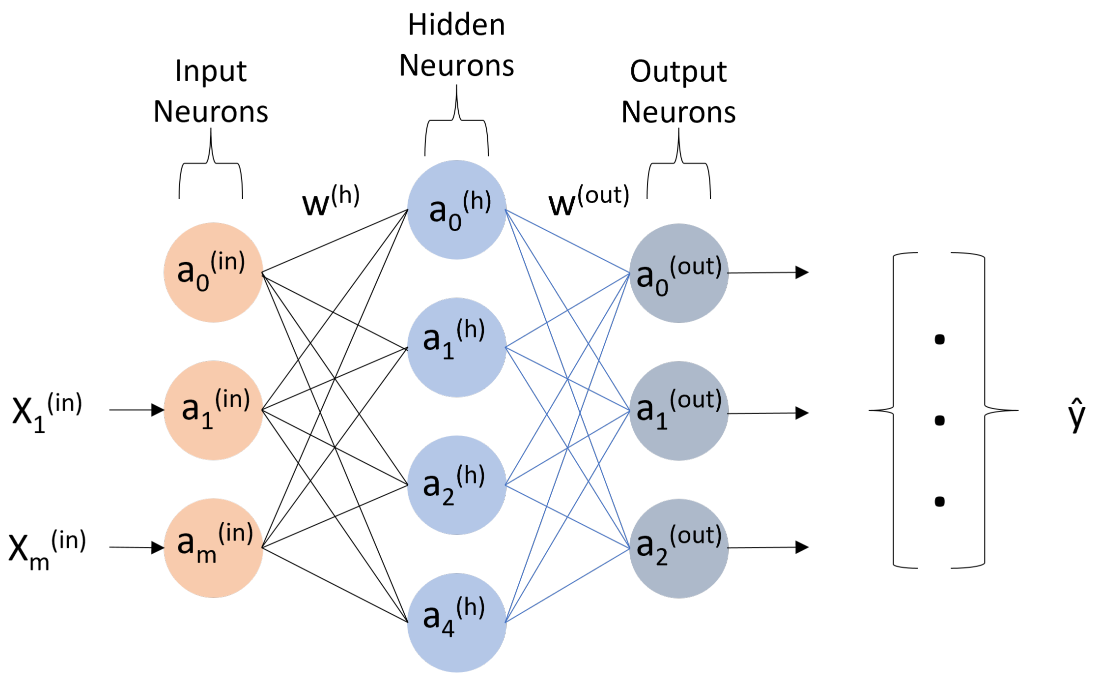


Figure 2.5: Schematic overview of a fully connected multilayer perceptron.

The MLP shown in Figure 2.5 consists of one input layer, one hidden layer and one output layer. Further it is an example of a fully connected artificial neural network because all nodes in the earlier layer are connected to all nodes in the following layer. If there is more than one hidden layer the model is called a deep neural network [11]. Additional layers and nodes will lead to more complex models. If there is too few nodes or hidden layers the model will not be complex enough. For obtaining the optimal number of nodes and layers one has to try many different combinations. There is no one-size-fits-all solution for the number of hidden layers and nodes. The MLP learning procedure can be summarized in three steps according to [11, Ch. 12]:

1. **Forward propagation:**
   Forward propagation is the process of passing the input data through a neural network to produce an output. The activation of each unit in one layer multiplied with the weight vector serves as the input for the next layer. For the MLP algorithem this can be written on matrix form as:

$$\boldsymbol{Z}^{(h)} = \boldsymbol{A}^{(in)}\boldsymbol{W}^{(h)} \tag{2.6}$$

$$\boldsymbol{A}^{(h)} = \phi\left(\boldsymbol{Z}^{(h)}\right), \tag{2.7}$$

   where $\boldsymbol{A}^{(in)}$ is an $n \times n$ matrix, m is the number of features and n is the number of samples. $\boldsymbol{W}^{(h)}$ is an $m \times d$ matrix where d is the number of units in the hidden layer.

This will result in the matrix $\boldsymbol{Z}^{(h)}$ with dimension $n \times d$. The superscripts (in) and (h) corresponds to input and the hidden layer.

Similarly for the output layer we can write:

$$\boldsymbol{Z}^{(out)} = \boldsymbol{A}^{(h)}\boldsymbol{W}^{(out)} \tag{2.8}$$

$$\boldsymbol{A}^{(out)} = \boldsymbol{\phi}\left(\boldsymbol{Z}^{(out)}\right), \tag{2.9}$$

where (out) correspond to the output layer.

2. **Error calculation:**
   Based on the output of the network and the true label the error is calculated. The error indicates the degree of accuracy or precision of the model on the input data. The goal is to minimize the error during the training, this is done through minimizing a cost function. There are many different cost functions, and the appropriate to use is dependent on the problem at hand.

3. **Backpropagate:**
   Backpropagation is an algorithm widely used for training the weights of a neural network. The backpropagation algorithm works by propagating the error backwards through the network, starting from the output layer and moving towards the input layer. Backpropagation utilizes the chain rule for effectively computing the partial derivatives of a complex function. In the case where gradient descent is used as an optimizing algorithm, the effectively computed partial derivatives come in handy for updating the weights of the model. Overall, backpropagation is an essential algorithm for training deep neural networks by effectively minimizing the error and enable the network to accurately predict outputs on new data.

2.3.3 Recurrent neural network

The theory and equations presented in this section is based on [11, Ch. 16].

Recurrent neural networks are designed for modelling sequential data. Sequential data is when a sequence of data points are not independent from each other. Recurrent neural networks have many of the same characteristics as the feedforward network, but the main difference is how information is propagated through the network. For example, in the case of a feedforward network with three layers, the information flows from the input layer to the hidden layer and from the hidden to the output layer. While in a recurrent neural network with three layers, the hidden layer gets two inputs. One from the input layer and the other from the hidden layer from the previous time step, as shown in the unfolded single layer RNN in Figure 2.6. The information from the previous time step enables the network to have memory of past time points.

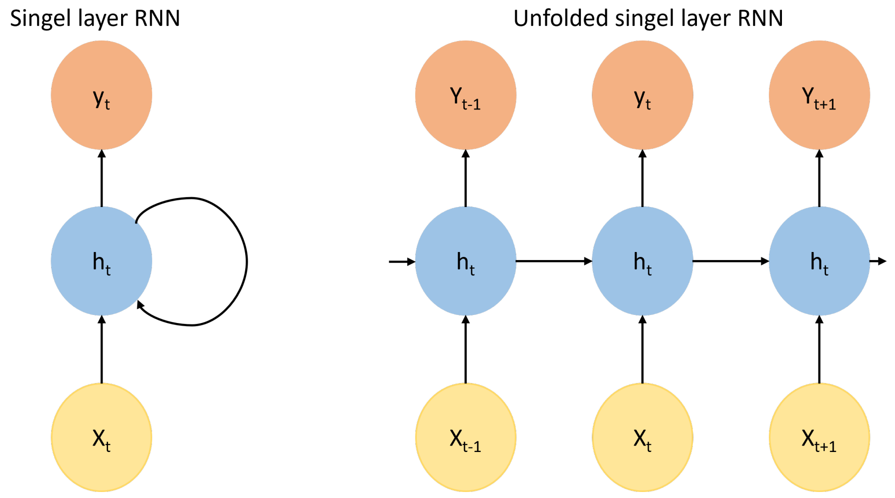


Figure 2.6: Schematic overview of recurrent neural network.

The weights do not depend on time and therefore they are shared across the time axis. The net input of the hidden layer depends on two weight matrices: one from the input, while the other comes from the activation, from the previous time step. The weight matrices are denoted $\boldsymbol{W}_{xh}$ and $\boldsymbol{W}_{hh}$. $\boldsymbol{W}_{xh}$ is the matrix between the input(x) and hidden layer(h) and $\boldsymbol{W}_{hh}$ is the weight matrix between time step t and $t-1$ from the same hidden layer. The activation of the hidden layer at time t is calculated as follows:

$$\boldsymbol{h}^{(t)} = \phi_h\left(\boldsymbol{z}_h^{(t)}\right) = \phi_h\left(\boldsymbol{W}_{xh}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hh}\boldsymbol{x}^{(t-1)} + \boldsymbol{b}_h\right), \tag{2.10}$$

where b_h is the bias from the hidden units and ϕ_h is the activation of the hidden layer.

The output activations of time stamp t, ($\boldsymbol{o}^{(t)}$) are given by:

$$\boldsymbol{o}^{(\boldsymbol{t})} = \phi_o\left(\boldsymbol{W}_{ho}\boldsymbol{h}^{(t)} + \boldsymbol{b}_o\right), \tag{2.11}$$

where $\boldsymbol{W}_{ho}$ is the weight matrix between the hidden layer and output layer and $\boldsymbol{b}_0$ is the bias from the output layer.

RNN uses backpropagation through time (BPTT) which is slightly different from the backpropagation used of feed forward neural networks. For more information about BPTT, see [11, Ch. 16]. One frequent problem with RNN or feedforward networks with many layers are vanishing or exploding gradients. Vanishing gradients occur when the gradients get smaller and smaller during the backpropagation. This leads to almost no change in the weights of the earlier layers, and the model will not converge correctly. Exploding gradients is the opposite where the gradients get bigger and bigger during the backpropagation. Long short-term memory (LSTM) is a model for sequence data which has been successful in overcoming the problems regarding the gradients.

2.3.4 Long short-term memory

The theory and equations presented in this section is based on [11, Ch. 16].

The main building block of a long short-term memory (LSTM) is a memory cell, which can be seen as a replacement of the hidden layer in an RNN. The memory cell allows the LSTM to handle sequence data with dependencies in the data and the structure of a LSTM is shown in Figure 2.7.

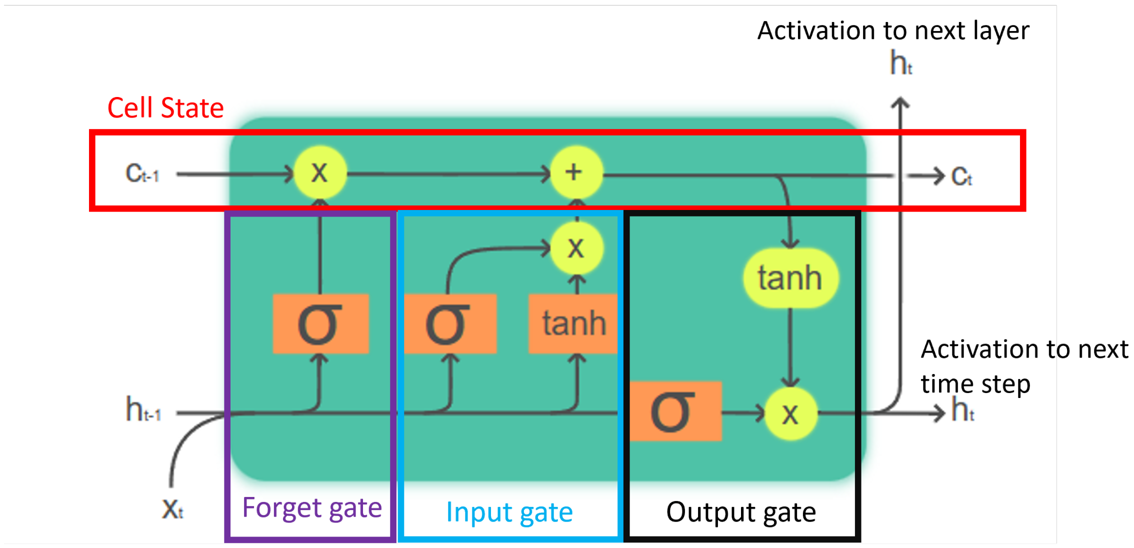


Figure 2.7: Schematic overview of a LSTM cell, see text for details. The visualization is licensed under the CC-BY license, by Guillaume Chevalier. The boxes in red, purple, turquoise, and black are changes. The original visualization can be found her: `https://github.com/guillaume-chevalier/Linear-Attention-Recurrent-Neural-Network/tree/master/inkscape_drawings`

The LSTM architecture has an additional flow of information called cell state (C) compared to the RNN in Sec. 2.3.3. A LSTM memory cell consists of three gates called forget gate(f_t), input gate(i_t) and output gate(o_t), marked in purple, turquoise and black, respectively, in Figure 2.7.

The forget gate affects which information to keep in the cell state, and is given by:

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_{xf}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hf}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_f). \tag{2.12}$$

The weight matrices $\boldsymbol{W}_{xf}$ and $\boldsymbol{W}_{hf}$ are the weights associated to the input ($\boldsymbol{x}^{(t)}$) and the activation from the previous time step ($\boldsymbol{h}^{(t-1)}$), respectively. The sigmoid function, σ, is responsible for deciding what information to keep. It uses the input, previous activation and bias ($\boldsymbol{b}_f$) to output a matrix with the same dimension as C_{t-1} with values between 0 and 1, where 1 represents remember and 0 represents forget. The weight matrices and bias vector are the trainable parts and are responsible for the behaviour of the forget gate.

The input gate and candidate value ($\tilde{C}_i$) determine what new information to add to the cell state, and are given by:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_{xi}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hi}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i) \tag{2.13}$$

$$\tilde{\boldsymbol{C}}_{\boldsymbol{i}} = \tanh(\boldsymbol{W}_{xc}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hc}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_c). \tag{2.14}$$

The sigmoid function in the input gate decides which values are updated, while the tanh function in the candidate value decides what value that should be added to the new cell state. The weight matrices and the bias are the trainable parts.

The cell state ($C_{(t)}$) is updated based on the forget gate, input gate and the previous cell state $C^{(t-1)}$. The update is given by:

$$\boldsymbol{C}^{(t)} = (\boldsymbol{C}^{(t-1)} \odot \boldsymbol{f}_t) \oplus (\boldsymbol{i}_t \odot \tilde{\boldsymbol{C}}_{\boldsymbol{i}}), \tag{2.15}$$

where $\odot$ denotes element-wise multiplication and $\oplus$ denotes element-wise addition.

The output gate is given by:

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_{xo}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{ho}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_o), \tag{2.16}$$

where the weight matrices and bias are the trainable parts. The output of the output gate is used for updating the hidden units. The update of the hidden units is given by:

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}_t \tanh\left(\boldsymbol{C}^{(t)}\right), \tag{2.17}$$

where the tanh function compresses the cell state values between -1 and 1 and when multiplied with the output gate we are left with the activation for the next time step.

2.4 Ensemble learning

Ensemble learning algorithms combine several algorithms to one model. The idea behind ensemble learning is that several algorithms combined is more generalizable than one single algorithm[11]. Majority voting is one of the least complex ensemble method, it simply predicts the label predicted by the most algorithms. For example, if six out of ten models predict x for a certain sample, then the ensemble model will predict x for that sample.

2.4.1 Extreme gradient boosting

Extreme gradient boosting (XGBoost) is a very popular and powerful ensemble learner. The main building block of the XGBoost algorithm is the gradient boosting decision tree algorithm. A decision tree has a tree structure with a root node, internal nodes and leaf nodes as shown in Figure 2.8, and it works by iteratively dividing the data into smaller chunks based on the feature that leads to the largest information gain [11]. A decision tree is iteratively dividing the nodes until it has reached the max depth of the tree, or a node consists of only a single sample.

For more information on decision trees, see [11, Ch. 3]. The gradient boosting algorithm can be summarized in the following steps [13].

1. Calculate the average of the target variable. This value will be used as a baseline prediction in the following steps.
2. Calculate the residual for every sample based on the average value from step one. The residual is calculated by subtracting the baseline prediction from the real target value.
3. Create a new decision tree with the aim of predicting the residuals.
4. Predict the target value based on the baseline prediction and all decision trees trained until now multiplied with their learning rate. The learning rate is introduced to reduce overfitting. In the first iteration the predicted value is calculated by adding the learning rate multiplied with the residual predicted by the decision tree to the baseline prediction.
5. Compute the new residual based on the predicted value in step four. The residual is calculated as in step 2.
6. Repeat step 3 to 5 until the desired number of decision trees are built. The desired number of decision trees is called number of estimators.
7. Make a final prediction based on all decision trees. The calculation of the final prediction is calculated by adding the prediction of all decision trees multiplied with the learning rate to the initial prediction.

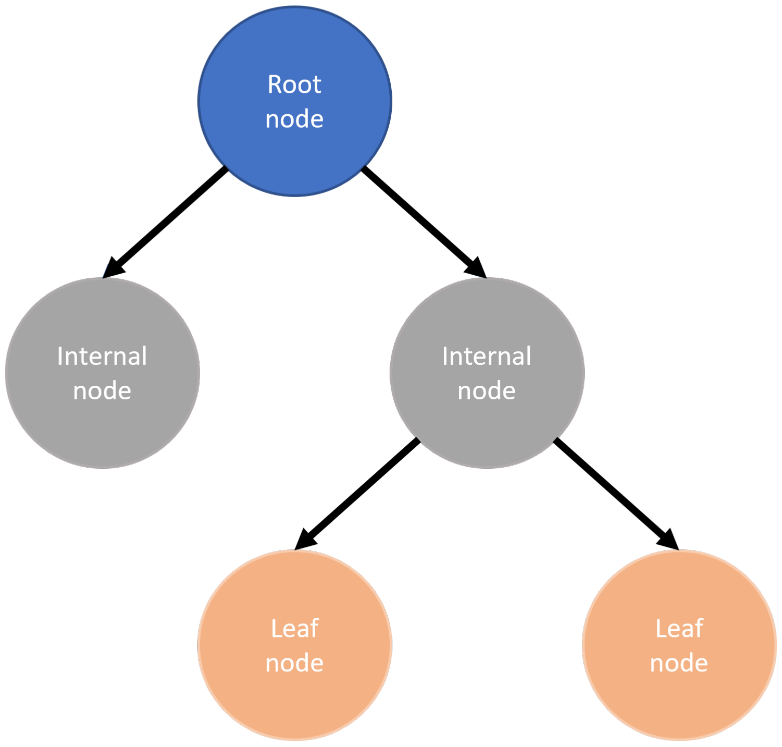


Figure 2.8: Schematic overview of the structure of a tree based model.

2.5 Time series forecasting

The theory and equations presented in this section is mainly obtained from [14, 15].

2.5.1 Time series

A time series is a set of numerical data in consecutive order along one axis, and this axis is referred to as time [14]. Any permutation of the order will change the information in the data. By default, time series data is not independent and identically distributed(i.i.d) along the time axis. For example, daily sea temperature measurements of two successive days are correlated across the year and daily measurements change between seasons and therefore are not i.i.d.

One parameter traced over time forms a univariate time series. Several parameters traced over the same time steps form a multivariate time series. Time series are either regular or irregular. Regular timeseries are defined as a timeseries with equal time difference between all pairs of consecutive values. This can be denoted as $\Delta t = t_2 - t_1 = t_{i+1} - t_i$ for all $i = 1, ..., t_{max-1}$. For irregular time series at least one $t_{i+1} - t_i \neq \Delta t$. Many models require regular time series by definition therefor transforming an irregular time series into a regular time is sometimes necessary.

Time series are often composed of the sum or product of three components. These components are trend, seasonality, and random noise [14]. The trend in the time series is a non-periodic function over time indicating a general direction of development of the time series. Seasonality is a periodic function describing the oscillation in a time series. Random noise is describing the time independent randomness in the time series.

2.5.2 Stationarity and differencing

Some models require stationary data. A stationary time series is defined as: "A stationary time series is one whose statistical properties do not depend on the time at which the series is observed" [14]. Consequently, a time series with trend or seasonality are nonstationary because the statistical properties will change with time. To transform a nonstationary time series to a stationary timeseries a widely used technique is differencing [15].

Differencing computes the difference between successive time points. This is often referred to as first differences. First differences are used for removing a linear trend. When we have a linear trend, we assume that:

$$y_t = y_{t-1} + c + \varepsilon_t, \tag{2.18}$$

where c is the average value of the change between two consecutive time points and ε_t is random noise for the current time point. By subtracting y_{t-1} from Equation 2.18 we are left with $y_t - y_{t-1} = c + \varepsilon_t$, and $y_t - y_{t-1}$ is the differenced value denoted y'_t. The differenced time series will result in a time series with length $T - 1$ where T is the length of the original time series. The reason for this is because y_1 have no previous value y_0.

Seasonal differencing is the differences for a given time period, such as a year, and the same period from the previous year. For example, one could subtract daily temperatures in the previous year from those of the current year. The goal of seasonal differencing is to remove seasonal variation, and can be denoted as:

$$y'_t = y_t - y_{t-m}, \tag{2.19}$$

where m is the seasonal length.

2.5.3 Autoregressive

Autoregressive (AR) models are widely used in time series forecasting where there is a relationship between past and feature values. This is due to the AR models using a linear combination of past target variables to forecast new target variables. An autoregressive model of order p can be written as:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \varepsilon_t, \tag{2.20}$$

where $y_{t-1}, ..., y_{t-p}$ are past values of the target variable, $\phi_1, ..., \phi_p$ are the model parameters and ε_t is random noise.

2.5.4 Moving average

Instead of using the past values of the target variable in a regression, an moving average (MA) model uses past errors in a regression like model. A MA model of order q can be written as:

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + ... + \theta_q \varepsilon_{t-q}, \tag{2.21}$$

where $\varepsilon_{t-1}, ..., \varepsilon_{t-p}$ are the past errors and $\phi_1, ..., \phi_p$ are the model parameters.

2.5.5 Autoregressive integrated moving averge

An autoregressive integrated moving averge (ARIMA) model is the combination of differencing with AR and a MA model. ARIMA models are often written as ARIMA(p, d, q) where p is the order of autoregressive part, d is the degree of first order differencing and q is the order of the moving average. The ARIMA model can be written as

$$\hat{y}_t = c + \phi_1 \hat{y}_{t-1} + ... + \phi_p \hat{y}_{t-p} + ... + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} + \varepsilon_t, \tag{2.22}$$

where the $\hat{y}$ values are the differenced values.

Finding the optimal values for p, d and q can be difficult. One common way of finding p and q is to use the autocorrelation function (acf) and partial autocorrelation function (pacf). Autocorrelation is the relationship between y_t and y_{t-k} for different values of k and partial autocorrelation is the relationship between y_t and y_{t-k} after removing the effect of $1, 2, \ldots, k-1$

[14]. It is also possible to use an auto ARIMA function to determine the values for p, d and q. The auto ARIMA function tries different values for p, d and q and returns the best combination. The drawback of this method is that it is computational expensive.

2.5.6 Seasonal autoregressive integrated moving averge with exogenous variables

Seasonal autoregressive integrated moving averge with exogenous variables (SARIMAX) is a extension of ARIMA models to incorporate exogenous variables and seasonal effects. The addition of exogenous variables and seasonal effects can lead to more powerful models [16]. The general notation for SARIMAX models is SARIMAX$(p, d, q)(P, D, Q, S)$, where (p, d, q) are the non-seasonal parameters that are similar to those of the ARIMA model. The additional seasonal parameters, (P, D, Q), represent the seasonal autoregression, differencing and moving average terms, respectively. The seasonal length, S, specifies the number of time units in a single season.

Exogenous variable are variables that affect the time series being modelled but are not themselves affected by the time series. For instance, in the context of predicting the corn production, the weather would be an exogenous variable because the weather effects the corn production, but corn production does not influence the weather. By incorporating exogenous variables into the model, we can potentially improve its forecasting accuracy and robustness [16].

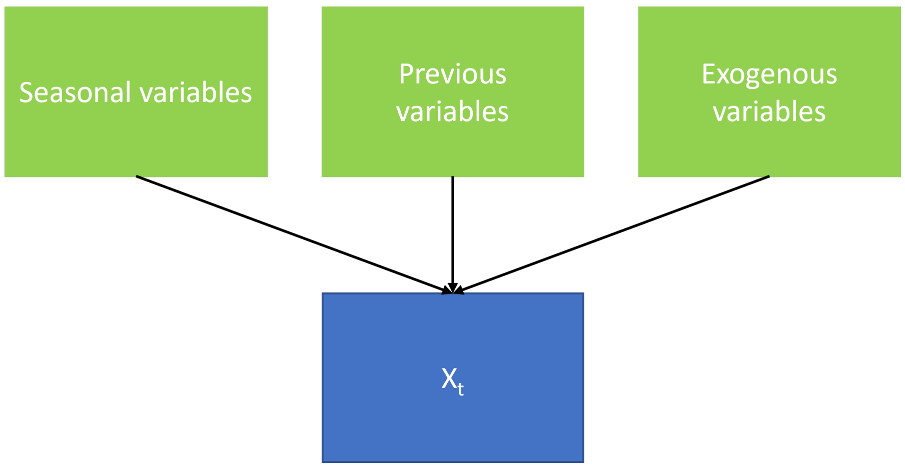


Figure 2.9: Schematic overview of the variables contributing to the forecast of an SARIMAX model.

2.6 Workflow of machine learning

Chollet and Francois [17] suggest an universal workflow for machine learning tasks. The universal workflow follows seven distinct steps:

1. **Define the problem and collect data**

   The initial step in applying machine learning is to precisely define the problem. Once the problem is defined it is important to identify a suitable ML method for solving the problem. Subsequently it is crucial to determine the type of input data that is required for training the model. It should be kept in mind that the model can only learn information contained within the data. In other words, if you have some features X and target Y it does not mean that X contains the information to predict Y. For instance, if you are trying to predict the corn production of this year based on the corn production from previous years. This will most likely lead to bad results since the corn production of recent years might not contain much predicative information about the corn production of this.

2. **Choose an evaluation metric**
   It is essential to define a common measurement of success to compare the performance of different ML models. The appropriate metric to measure success is highly related to the problem. For example, root mean squared error (RMS) or mean squared error (MSE) are metrics used for regression tasks.Further, it is important to establish a baseline. This baseline is used for determining whether a particular model is an improvement over the baseline.

3. **Decide on an evaluation protocol**
   Once the evaluation metric has been identified, it is essential to define the evaluation protocol. The holdout method, k-fold cross validation and nested cross validation are three common evaluation protocols [11]. The appropriate evaluation protocol is highly related to how much data is available. For instance, if there is a small data set, the k-fold cross validation or nested cross validation would be the appropriate choice of protocol. On the other hand, if there is plenty of data available the hold out method is appropriate.

4. **Prepare the data**
   For a machine learning model to be able to learn from data, the data must be organized in a way that the model can interpret. For instance, some models only work with numerical data and not categorical data, which require transforming the categorical data to numeric data. One-hot encoding is a suitable method for transforming categorical data to numerical data [11]. If working with deep neural networks, it is recommended to normalize or standardise the data as this can help the model to train [11]. Additionally, visualising the data during the data preparation step is recommended. Visualization can identify characteristics of the data and determine further data preparation steps. The steps mentioned above are just some of the most common data preparation steps, but the most important is to prepare the data for machine learning models to be able to learn for data.

5. **Develop a model**
   The objective of this step is to develop a model that outperforms the baseline. One essential step of developing a model is to identify the optimal activation function, loss function and hyperparameters for effectively solving the problem at hand.

6. **Scale up**
   After identifying models that outperforms the baseline, it is essential to evaluate the level of complexity of those models. Is it possible to obtain better results through additional layers? More hidden units? Or train for more epochs? These are important questions to ask when optimizing models.

7. **Regularize and tune our model**
   The final step involves regularizing and hyperparameter tuning. Regularizing is used for tackling overfitting while hyperparameter tuning is done for squeezing out the last improvements of the model.

2.7 Overfitting vs underfitting

In supervised machine learning we want models to be capable of predicting or classifying unseen data correctly. If a model is sufficiently good at predicting unseen data, we say that the model is generalizable. The main goal of machine learning is to produce a generalizable algorithm.

Data used for training machine learning models are known as training data and the data used for evaluating the model is known as test data. The ideal case is when there is little difference between the error of the training data and test data. When the model performs bad on both the training and test data, we say that the model is underfitting. On the other hand, when the model performance is sufficiently good on the training data and bad on the test data the model is overfitting. The different scenarios of underfitting, optimal fit and overfitting are illustrated in 2.10. Models suffering from overfitting have captured all details in the training data, which are too specific for the training data, and this leads to less generalizability. Underfitting fails to capture significant details of the training data and are therefore not generalizable.

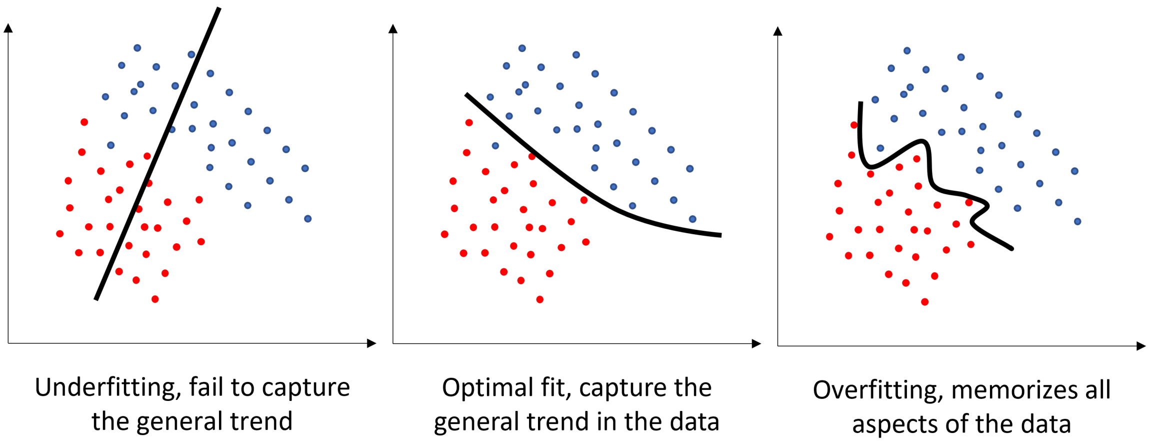

Underfitting, fail to capture the general trend
Optimal fit, capture the general trend in the data
Overfitting, memorizes all aspects of the data

Figure 2.10: Illustrates underfitting, optimal fitting and overfitting of machine learning models.

2.8 Performance metrics for regression

Regression refers to the problem of predicting a numerical value [18]. Performance metrices are used to evaluate the performance of models. In relation to regression the metric should be able to measure how far the prediction is from the real value. Mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE) and coefficient of determination R^2 are frequently used for regression tasks. These metrices will be further elaborated on below.

- MSE is the mean value of the sum of the squared estimate of errors (SSE).MSE is defined as:
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \tilde{y}^{(i)})^2, \tag{2.23}$$
where $y^{(i)}$ is the true value and $\tilde{y}^i$ is the predicted value. MSE is a unbonded metrics which means that MSE depends on the target value. This can easily be demonstrated by: $(10-15)^2 < (100-150)^2$.

- RMSE is the square root of MSE. The difference between RMSE and MSE is that RMSE will have the same unit as the target while MSE has units of the target squared. RMSE is defined as:
$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \tilde{y}^{(i)})^2}. \tag{2.24}$$

- MAE has the same property as RMSE regarding the error score having the same unit as the target. Unlike RMSE and MSE, MAE does not penalise larger errors more than smaller errors, but there is a linear relationship between small and large errors. This follows from the fact that there is no squaring of the error. MAE is defined as:
$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \tilde{y}^{(i)}\right|, \tag{2.25}$$
where y_i is the true value and $\tilde{y}^{(i)}$ is the predicted value.

2.9 Evalution protocols for machine learning

The theory presented in this section is mainly obtained from [11, Ch. 6]

To accurately assess the performance of machine learning algorithms, it is essential to evaluate the model on a data set that is distinct from the one used for training. Various techniques exist for assessing the effectiveness of these algorithms, with two common methods being the hold-out method and cross-validation. In the following parts we are going to further elaborate on these two methods.

2.9.1 Hold out method

The hold-out method, which is a commonly used approach for evaluating machine learning algorithms, involves partitioning the dataset into training and test sets. The training set is used for model fitting, while the test set is employed for evaluating the performance of the model. When dividing the data into training and test sets, it is essential to keep in mind that a larger test set may lead to a loss of crucial information, while an excessively small test set may not be representative [11]. Typical splits for the hold-out method include 90:10, 80:20, or 70:30, although the size of the split is highly dependent on the size of the original data. Furthermore, it is critical to note that when using the test data for model selection, it can lead to overfiting becuse of information lekage. A way to avoid this is to divide the data into training set, validation set, and test set. The training set is used for fitting the model while the validation set it used for model selection. Finally, the test set is used for evaluating the selected model since it is an unbiased data set that the model has not seen during the training and selection process.

2.9.2 K-fold cross validation

The k-fold cross validation algorithm works by dividing the training data into k folds. For k iterations, the model is trained on $k-1$ folds and tested on the remaining fold. This process is repeated k times, resulting in every sample being used for training and validation. The performance is averaged across the k iterations to obtain an overall estimate of the model performance.

K-fold cross-validation is less sensitive to the initial partition of the data compared to the hold out method. It is useful for finding the optimal hyperparameters. Once the optimal hyperparameters have been identified the final model is trained on the entire data with the selected hyperparameters and tested on a separate test data to get an indication of the overall performance of the model.

Chapter 3

Method

3.1 The data sets

The multivariate time series used in this study was obtained from Veas, a wastewater treatment company. They provided two distinct data sets which were selected to investigate the research questions established in Sec. 1.2. The data sets were collected in the same time span but from two different process halls with different strategies for adding methanol in the measured period. Data set one was collected from process hall one and the methanol dosage strategy called *with feedback* in Sec. 2.1.2 was utilized. Data set two was measured in process hall two and the methanol dosage strategy *without feedback* was utilized. From now on data set one will be donated as “process where methanol was dosed with feedback”, while data set two will be denoted as “process where methanol was dosed without feedback”.

Since the data sets were collected in the same time span, we assumed that all factors affecting the denitrification would be equal for the two process halls, except the strategy utilized for adding methanol. Consequently, it would be possible to conclude whether it is easier to predict the amount of nitrate after denitrification for one of the methanol dosage strategies.

3.1.1 Features and target

The two data sets consist of thirteen variables, with each distinct measurement representing hourly averages of the measured variables. The measurements were measured between 30.11.2022 and 05.01.2023 resulting in two distinct data sets of size 870 $\times$ 13. The variables were selected based on their assumed significant influence on the denitrification process, as determined from a review of relevant literature and input from a process engineer at Veas. These variables were selected:

- **Amount water:** Amount of water entering the wastewater treatment plant, measured in l/s.
- **Temp in:** Measurement of the temperature of the wastewater before the grate, measured

in Celsius.

- **Level:** Measurement of liquid level in DEN-tank, measured in mH2O.
- **Oxygen:** Measurement of oxygen level in the DEN-tank, measured in mg/l.
- **Nitrate in:** Measurement of Nitrate level in the DEN-tank, measured in mg/l.
- **Methanol set point:** Number between 0 and 100, that adjust the methanol addition if the current nitrate measurements are over or under given set points.
- **Methanol:** Measurement of methanol added to the wastewater measured in l/h.
- **Pressure pump:** Measurement of pressure on the pump, measured in bar.
- **RPM:** Average number of rounds per minute(RPM) of the two pumps.
- **Pressure filter:** Average pressure into the four DEN-filters, measured in bar.
- **Flow filter:** Average flow into the four DEN-filters, measured in l/s.
- **Nitrate out:** Measurement of Nitrate level in the waste water after denitrification, measured in mg/l.
- **Temp out:** Measurement of the temperature of the wastewater after denitrification, measured in Celsius.

The variables above are arranged in chronological order according to their respective time of measurement in the wastewater treatment plant. The objective of this thesis is to predict the Nitrate out based on the other variables in the data set. Despite temp out is being measured after Nitrate out we included it in our model because this variable is the closest temperature measurement of the wastewater in the denitrification filter. if the models are to be employed at Veas in the future, e.g. for decision support, it would be expedient to install sensors for measuring the temperature in the denitrification-tank.

3.2 Pre-processing and data analysis

The main goal of this step was to investigate the data and potentially discover important characteristics for the modelling process.

Since we were dealing with time series data the first thing we did was to investigate if there were any missing or duplicates in the data points. No missing data were found but there were some duplicated time stamps. All duplicates were removed from the data. Further we checked for global and contextual outliers. All outliers were sent to a process engineer at Veas for evaluating if the outliers should be considered as outliers or not. All outliers were measurements that could be expected, and therefore not removed.

The next step was to standardize the two data sets. For standardizing the data sets we used the *StandardScaler*[1] function from the preprocessing package from Sklearn. *StandardScaler* was used for getting the variables in the data on to the same scale. This can lead to faster convergence of machine learning algorithms as mentioned in Sec. 2.3.1. Further scaling the data made it possible to determine feature importance in the case of SARIMAX. If the data is scaled we can investigate the absolute value of the coefficients of the SARIMAX model, which corresponds to how much weight the model puts on the respective features when deciding the output.

After scaling the data, all variables in the data set were plotted in a line plot. This was done to investigate if there are some repeating pattern or correlation between variables in the data. The line plot was also used for investigating if the variables had constant mean and variance over time. To further investigate the correlation between variables we used the Pandas function *corr*[2] for obtaining the Pearson correlation. The Pearson correlation tells us if there is a linear relationship between two variables. The *plot_acf*[3] function from statsmodels was used for plotting the autocorrelation function. From the autocorrelation plot it is possible to determine if there is a seasonal pattern and if so the seasonal length.

For evaluating the performance of the models, we used the hold out method. All models were trained and tested on the same data. The training data contained measurements from between 30.11.2022 and 29.12.2022 and the test data contained measurements from between 30.12.2022 and 05.01.2023. We believe that maintaining the chronological order in the training and test sets would result in a more accurate assessment of the model performance. This is because if a model is deployed it is trained on past data and will be utilized to forecast future data.

3.3 Method Selection

SARIMAX, LSTM and XGBoost were selected to investigate which models are suitable for predicting the amount of nitrate contained in the wastewater after denitrification. Although SARIMAX is a forecast method and not a prediction method we chose to include this method because its forecast relies on both prior values of the predictor and exogenous variables. Therefore, if we provide two different exogenous inputs for the same forecast horizon, it will provide two different forecasts. Hence it is possible to use the SARIMAX model to predict future nitrate out values based on different exogenous variables and see how it will affect the denitrification process. In the following paragraphs will present why we chose to include SARIMAX, LSTM and XGBoost.

SARIMAX was included because it is a widely used forecasting method for modelling process with a seasonal component. We know that the nitrate has a daily variation and thus it contains

[1]Documentation for *StandardScaler*: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

[2]Documentation for *corr*: `https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html`

[3]Documentation for *plot_acf*: `https://www.statsmodels.org/dev/generated/statsmodels.graphics.tsaplots.plot_acf.html`

a daily seasonal component. Because SARIMAX is capable of incorporating daily fluctuations and handling exogenous variables we thought the model would be suitable for the modelling problem. Another reason for trying SARIMAX is that it is possible to see feature importance. If the data set is standardised then the absolute value of the coefficients tells how important a feature is [19].

LSTM is a powerful machine learning algorithm widely used for time series problems. The LSTM model was included in this paper because of its ability to deal with long term dependencies. The ability to handle long term dependencies is important because we assume there is some dependence between consecutive time steps in the data. Another advantage of LSTM is that it is more robust against the vanishing gradient problem than other RNN architectures.

Both SARIMAX and LSTM learn or incorporate information from previous time points. To investigate if this information is leading to any improvement in predicting the amount of nitrate contained in the wastewater after denitrification, we included XGBoost, which only depends on data points from each feature. XGBoost was also chosen because it is a powerful method for tabular data while being computational inexpensive. Another advantage of XGBoost is the built-in feature importance.

3.4 Model implementation and development

3.4.1 SARIMAX

The exogenous variables for the SARIMAX model were current values of all features and the endogenous variable was the nitrate out values.

The SARIMAX model was implemented in Python using the statsmodels module, a popular statistical modeling library that provides a range of statistical models, tests, and data visualization tools. To identify the optimal SARIMAX parameters, the *auto_arima*[4] function from the Pmdarima library was utilized. This library is specifically designed for time series analysis, providing a range of methods and tools to facilitate the modelling of time series data.

To determine the best SARIMAX parameters, several input parameters were specified, including the endogenous and exogenous training variables, as well as the start and end values of the autoregressive, moving average, seasonal autoregressive, and seasonal moving average components (p, q, P, Q). The values of p, q, P and Q tested for in the *auto_arima* are listed in Table 3.1. Additionally, the seasonality was set to true. The order of differencing and seasonal differencing is found by the *auto_arima* function by conducting several differencing tests. *auto_arima* is an exhaustive algorithm which tries every combination of the model parameters and returns the parameters optimising a given information criterion.

[4]Documentation for *auto_arima*: `https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html`

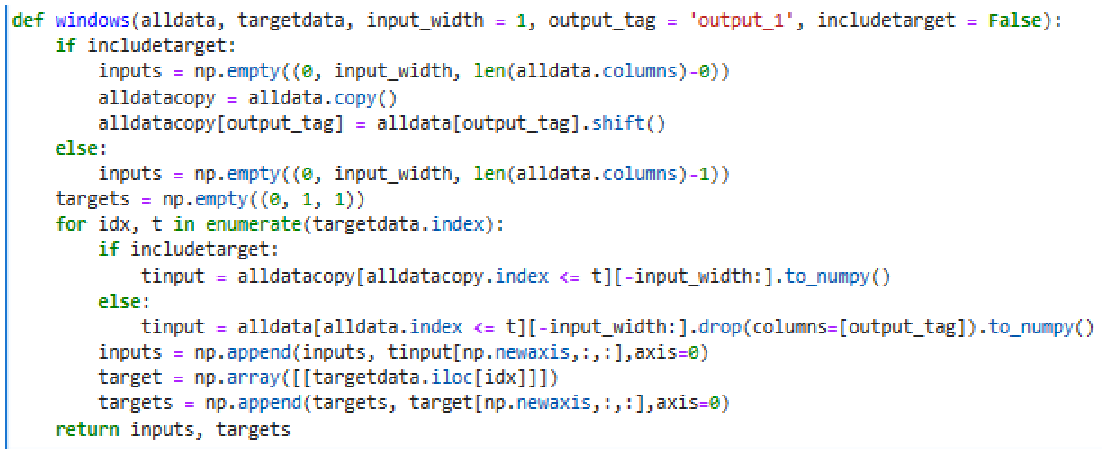
Parameter	Tried parameters
p	0, 1, 2, 3, 4, 5
q	0, 1, 2, 3, 4, 5
P	0, 1, 2, 3
Q	0, 1, 2, 3

Table 3.1: All parameters tested in the *auto_arima* for the SARIMAX.

One limitation of the *auto_arima* function is the possibility to overfitting, as the information criterion is based on the training set. To mitigate this risk, the top three parameter combinations were tested on a separate test set. The performance of each model was evaluated using standard metrics such as RMSE, MSE, and MAE, and a visual inspection was conducted. Further the performance of the SARIMAX model were compared to the performance of the LSTM and XGBoost.

3.4.2 LSTM

LSTM requires a 3D tensor with shape (batch, timesteps, features) as input. As the provided data was in the form of a data frame, we utilized the function presented in Figure 3.1 to convert it to the desired format. The transformation function returns two 3D tensors, one representing the training data and the other corresponding to the targets. It takes four parameters, namely, "alldat," which is the 2D data frame, "targetdata," which is the target, "input_width," which represents the lookback length, and "output_tag," which is the name of the target column. Additionally, the parameter "includetarget" is used to specify whether lagged values of the target should be included in the training data. In this case, we set "includetarget" to false, and since we assumed a daily fluctuation, "input_width" to 24. This lead to a 3D tensor with shape (846, 24, 12).



```
def windows(alldata, targetdata, input_width = 1, output_tag = 'output_1', includetarget = False):
    if includetarget:
        inputs = np.empty((0, input_width, len(alldata.columns)-0))
        alldatacopy = alldata.copy()
        alldatacopy[output_tag] = alldata[output_tag].shift()
    else:
        inputs = np.empty((0, input_width, len(alldata.columns)-1))
    targets = np.empty((0, 1, 1))
    for idx, t in enumerate(targetdata.index):
        if includetarget:
            tinput = alldatacopy[alldatacopy.index <= t][-input_width:].to_numpy()
        else:
            tinput = alldata[alldata.index <= t][-input_width:].drop(columns=[output_tag]).to_numpy()
        inputs = np.append(inputs, tinput[np.newaxis,:,:],axis=0)
        target = np.array([[targetdata.iloc[idx]]])
        targets = np.append(targets, target[np.newaxis,:,:],axis=0)
    return inputs, targets
```

Figure 3.1: Function for transforming a dataset to a 3D tensor. Code snippet obtained from Sølve Eidnes.

The LSTM model was implemented using the Keras deep learning API in Python, which offers a range of deep learning models, layers, optimizers, and other features. The LSTM model utilized in this study was a single-layer LSTM architecture with a single dense output layer containing a single unit.

To optimize the hyperparameters of the LSTM network, we used the *GridSearch*[5] tuner class from the Keras library. The *GridSearch* tuner is a computationally intensive algorithm that becomes more complex with the addition of more parameters. Because of the resource-intensive nature of exhaustive algorithms, we constrained the search space for hyperparameters to save computational power. Additionally, the purpose of this thesis was to identify the overall performance of various machine learning algorithms in predicting the amount of nitrate after denitrification, rather than maximizing model performance through hyperparameter tuning.

The hyperparameters considered in the *GridSearch* included the number of units, dropout rate, activation functions, and optimizer. The values of units, dropout rate, activation functions and optimizer tested for in the *GridSearch* are listed in Table 3.2. The *GridSearch* algorithm identified the combination of hyperparameters that produced the best performing model, based on a small validation set drawn from the training data. The hyperparameter combination that yielded the best performance was subsequently evaluated on the test set, using the same evaluation metrics as those employed for the SARIMAX model.

Parameter	**Tried parameters**
Units	2, 5, 10, 20, 30, 40, 50
Dropout rate	0.0, 0.1, 0.2, 0.3
Activation functions	relu and tanh
Optimizer	adam and RMSprop

Table 3.2: All parameters tested in the *GridSearch* for the LSTM.

3.4.3 XGBoost

Both the LSTM and SARIMAX model have information from past values. To give the XGBoost the same information we added lagged values as columns. Since there was a daily pattern in the data we added columns for the last twenty-three hours, resulting in a dataframe of shape 846×288.

XGBoost was implemented by using the *xgboost*[6] package for Python. The Scikitlearn wrapper XGBRegressor was used for implementing a XGBoost regressor. To identify the optimal parameters for the XGBoost the *GridSearchCV*[7] function from Scikitlearn was utilised. *GridSearchCV* is an exhaustive hyperparameter tuning algorithm.

The hyperparameters were booster, nestimators, max_depth and eta. The tested values in *GridSearchCV* are listed in Table 3.3. *GridSearchCV* identifies the combination of the best

[5]Documentation for *GridSearch*: `https://keras.io/api/keras_tuner/tuners/grid/`

[6]XGBoost package for Python: `https://xgboost.readthedocs.io/en/stable/python/python_intro.html`

[7]Documentation for *GridSearchCV*: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

performing hyperparameters based on a cross validation on the training data. The best performing hyperparameters was tested on the test set, and the performance of the model were evaluated on the same evaluation metrics used for the SARIMAX and LSTM models.

Parameter	**Tried parameters**
nestimators	2, 5, 7, 10, 15, 20, 25, 30, 50
max_depth	1, 2, 3, 4, 6, 8
eta	0.1, 0.3, 0.01

Table 3.3: All hyperparameters tested in the *GridSearchCV* for the XGBoost.

The XGBoost model has a built-in feature importance function. To visualize the feature importance the *plot_importance* function from XGBoost was used. By defining the importance parameter in the *plot_importance* function it is possible to determine how the importance is calculated. We used the weight importance: this is the number of times the feature appears in a tree.

3.5 Baseline

A simple baseline was created to verify the models in this thesis. The simple baseline used was to predict the average nitrate out value in the training set. For the models to be verified as success they should at least make predictions more accurate than the average nitrate out value from the training set.

3.6 Software

The experiment of this thesis was programmed in Python, and the most important libraries are listed in Table 3.4.

Library	**Version**
Python	3.10.9
Pandas	1.5.3
Statsmodels	0.13.5
XGBoost	1.7.5
NumPy	1.23.5
Scikit-learn	1.2.1
Keras	2.12.0
Tensorflow	2.12.0
Pmdarima	2.0.3
Seaborn	0.12.2
Matplotlib	3.7.0

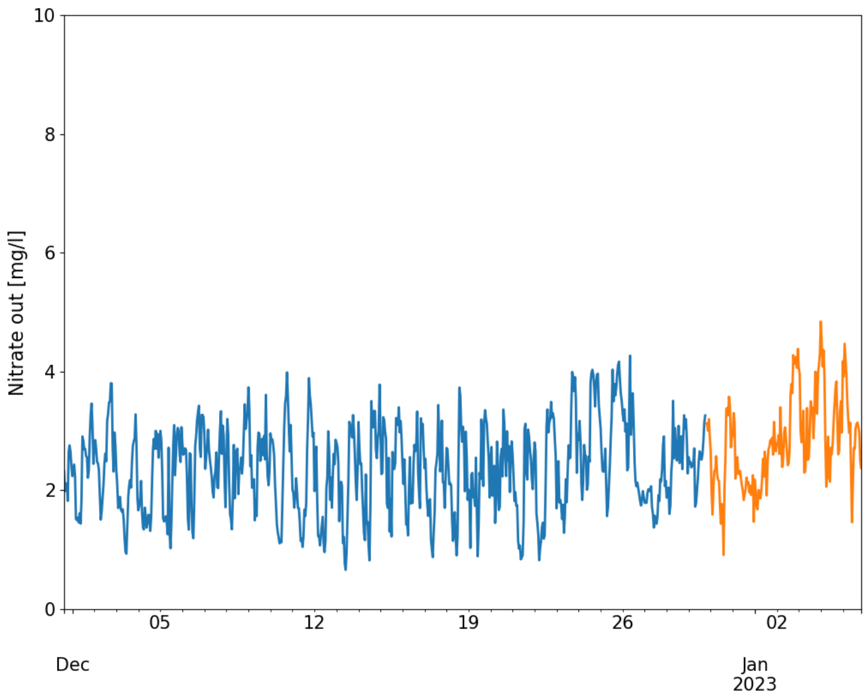
Table 3.4: Python libraries and the corresponding version used in the project.

Chapter 4

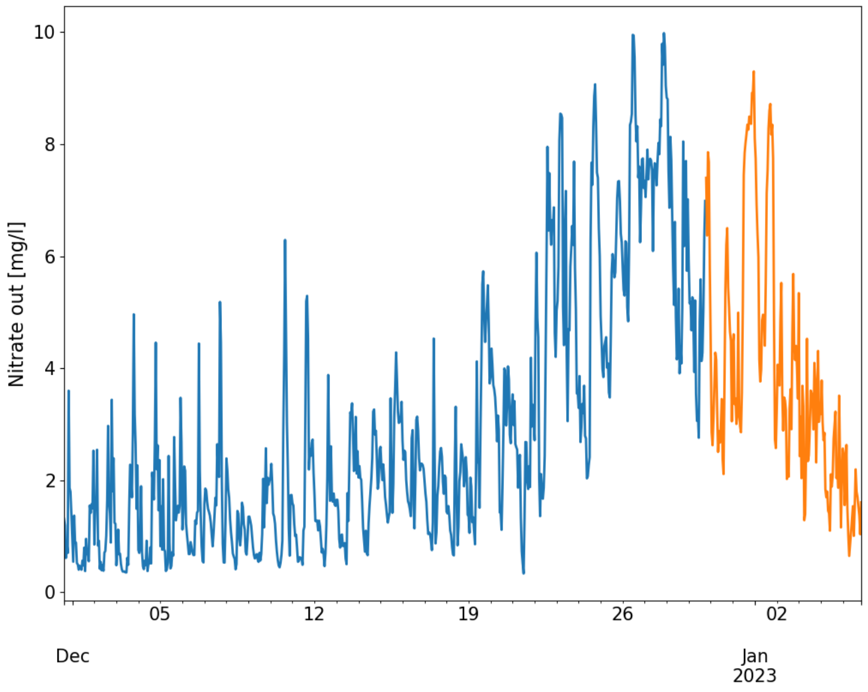
Results

4.1 Data exploration

In Sec. 3.1, we assumed that the factors affecting the denitrification process were similar for the two datasets except for the strategy utilized for adding methanol. A comparison of nitrate out values between the two datasets are shown in Figure 4.1. The figure reveals significant differences in nitrate out for the two processes. The nitrate out values for the process where methanol was dosed without feedback range between 0 and 5, whereas the process where methanol was dosed with feedback the nitrate out values range between 0 and 10. Moreover, in the data where methanol was dosed with feedback there is a noticeable increase in nitrate out values towards the end of the measurement period. This observation may suggest that the denitrification process may have been influenced by factors specific to each process or that there is a significant difference in the nitrate out values for the different strategies for dosing methanol.



(a)



(b)

Figure 4.1: Figure (a) shows nitrate out values where methanol was dosed without feedback and Figure (b) shows nitrate out values where methanol was dosed with feedback. The blue is the training period and the orange is the testing period.

Figure 4.2 shows the Pearson correlation between the variables for the two data sets with different methanol dosage strategy. We can observe that there are some different correlations between variables in the two distinct processes for adding methanol. In the process without feedback there is a correlation of 0.7 between nitrate out and methanol, while for the process with feedback there is a correlation of 0.1 for nitrate out and methanol. This can indicate that methanol is more important for predicting the nitrate out value for the process without feedback than for the process with feedback. There is also a difference in the correlation between nitrate in and out between the two processes. In the process without feedback nitrate in and out have a correlation of -0.7 while this correlation is 0.2 for the process with feedback. Furthermore, pressure pump, amount water, flow filter, pressure filter and rpm are highly correlated for both processes, with a correlation coefficient between 0.8 and 0.9. This is not unexpected since all these measures are related to the input flow of wastewater entering the wastewater treatment plant. Additionally, temp out and temp in are also found to be highly correlated features for both processes. For the process without feedback all variables besides methanol have a correlation coefficient between -0.25 and 0.3 with nitrate out. For the process with feedback we can obtain that nitrate out has a strong negative correlation with nitrate in, temp in, and temp out. In contrast, pressure pump, amount water, flow filter, pressure filter, and rpm are highly positive correlation with nitrate out. This can indicate that several variables are important for predicting nitrate out.

For the process without feedback, we can observe that methanol set point is not correlated with any other variable, this is as expected since the variable is set to 0 for the process without feedback because the methanol dosage is not dependent of nitrate out values. For the process with feedback methanol set point is either increased or decreased depending on if the nitrate out is over or under a given threshold value.

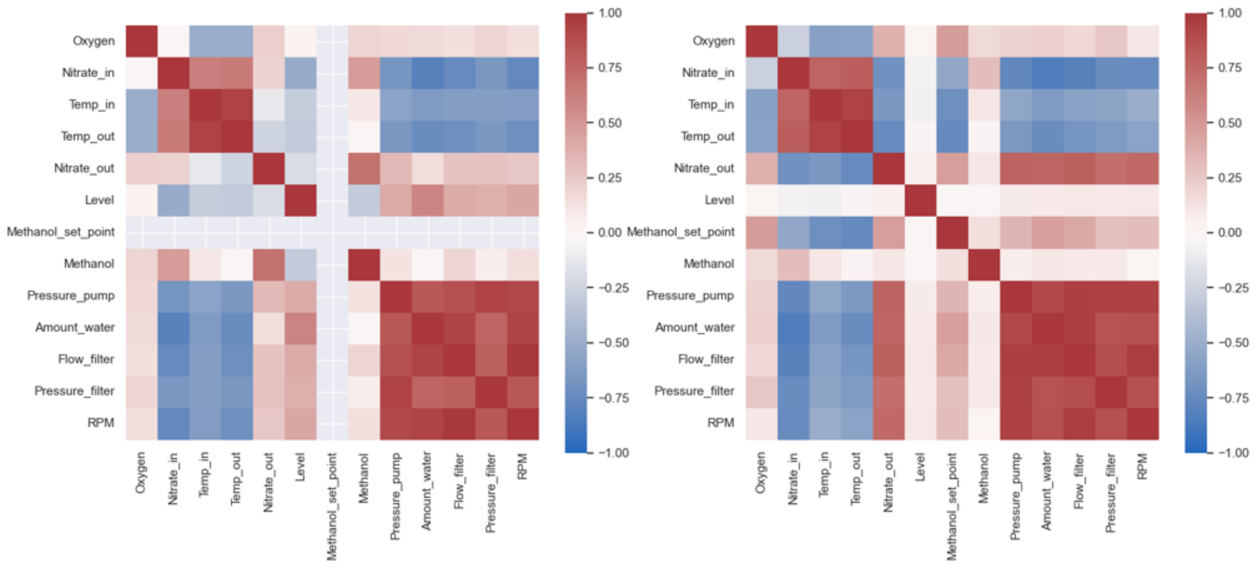
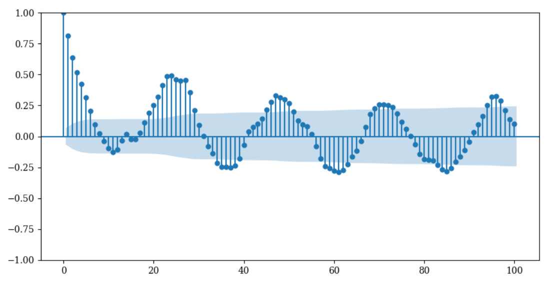
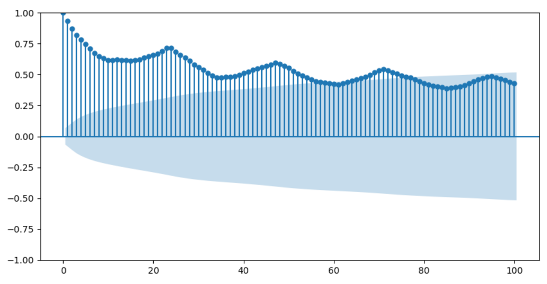


Figure 4.2: The Figure to the left shows the Pearson correlation between the features and targets for the process without feedback and the Figure to the right shows the Pearson correlation between the features and targets for the process with feedback.

Figure 4.3a and 4.3b show the autocorrelation for the first hundred lags for both the processes. We can observe a seasonal pattern in the nitrate out variable with a seasonal length of 24 hours because there are approximately 24 lags between each peak. This corresponds to the assumed daily variations in nitrate values entering the wastewater treatment plant.



(a)



(b)

Figure 4.3: Figure (a) shows acf for process when methanol is dosed without feedback, and Figure (b) shows acf for process with feedback.

4.2 Results on methanol dosage without feedback

4.2.1 XGBoost

The *GridSearchcv* for the XGBoost regressor, resulted in the parameters shown in Table 4.1. The objective function used was reg:squarederror. The performance of the XGBoost regressor was assessed using the test set, and the results are presented in Figure 4.4. The model manages to capture the overall trend, but it struggles to replicate the fluctuations in the data. Additionally, the predicted nitrate out values are one average lower than the actual values.

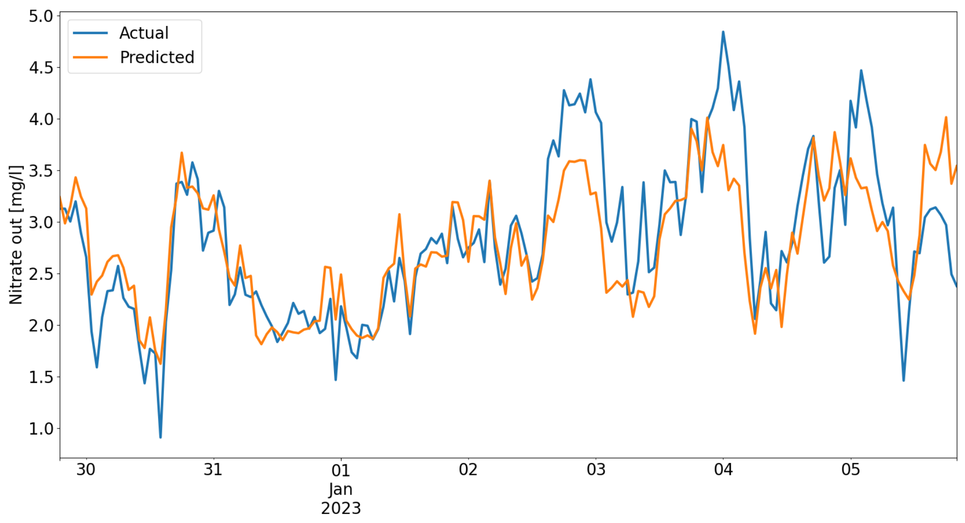


Figure 4.4: Predicted values generated by the XGBoost model compared to the actual values from the test period where methanol was dosed without feedback.

n_estimators	**max_depth**	**eta**
50	3	0.3

Table 4.1: The best parameter combination for XGBoost on methanol dosage without feedback.

Feature importance plot for XGBoost when methanol is dosed without feedback is represented in Figure 4.5. It reveals that methanol, nitrate in, and oxygen are the most significant features for the XGBoost model. This aligns with the theory presented in Sec. 2.1.1. Additionally, the analysis suggests that the model places less emphasis on lagged values. Notably, the six most important features include either current values or values from last time step.

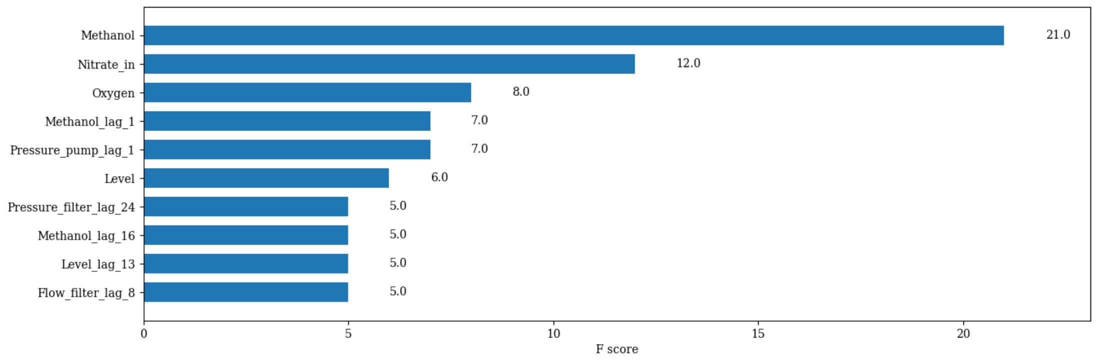


Figure 4.5: Ten most important features for XGBoost when methanol is dosed without feedback.

4.2.2 LSTM

The *GridSearch* resulted in the parameters presented in Table 4.2 and the loss function used was mean_squared_error. Figure 4.6 shows the predictions of the LSTM model on the test data compared to the actual values, and we can see that the model manages to catch the trend but struggles with the fluctuations. Moreover, the model tends to predict lower nitrate out values compared to the actual values.

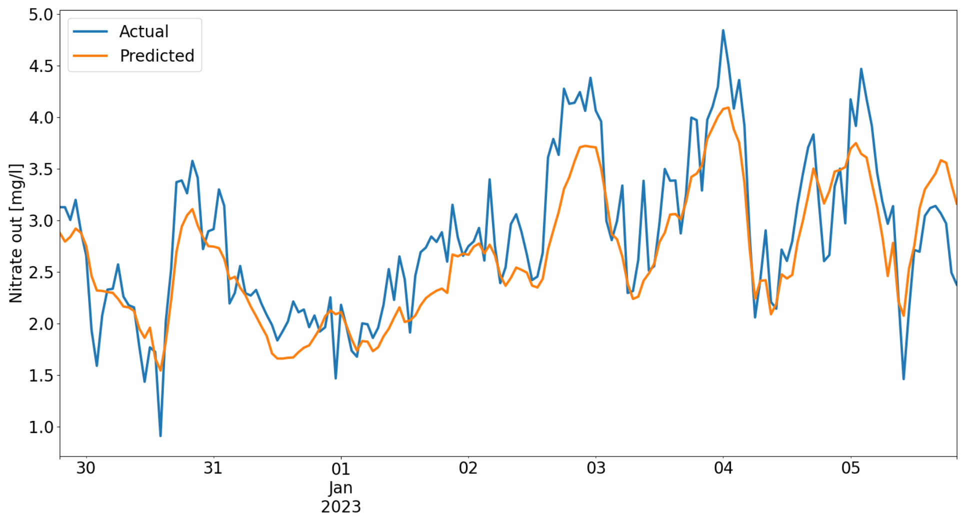


Figure 4.6: Predicted values generated by the LSTM model, compared to the actual values observed during the testing period when methanol was dosed without feedback.

units	**activation**	**dropout**	**optimizer**
30	relu	0.1	RMSprop

Table 4.2: The best parameter combination for LSTM on when methanol was dosed without feedback.

4.2.3 SARIMAX

From the *autoArima* function the optimal model parameters $(p, d, q)(P, D, Q, S)$ were equal to (1,0,1)(1,0,0,24). This indicates that the predictions for time step $t + 1$ is based on timestep t and t - 24h. The four most important exogenous features for the SARIMAX model were methanol, temp out, flow filter and pressure pump. The predicted values by the SARIMAX model compared to the actual values are represented in Figure 4.7, and it shows that the model catches the general trend and some of the fluctuations in the data.

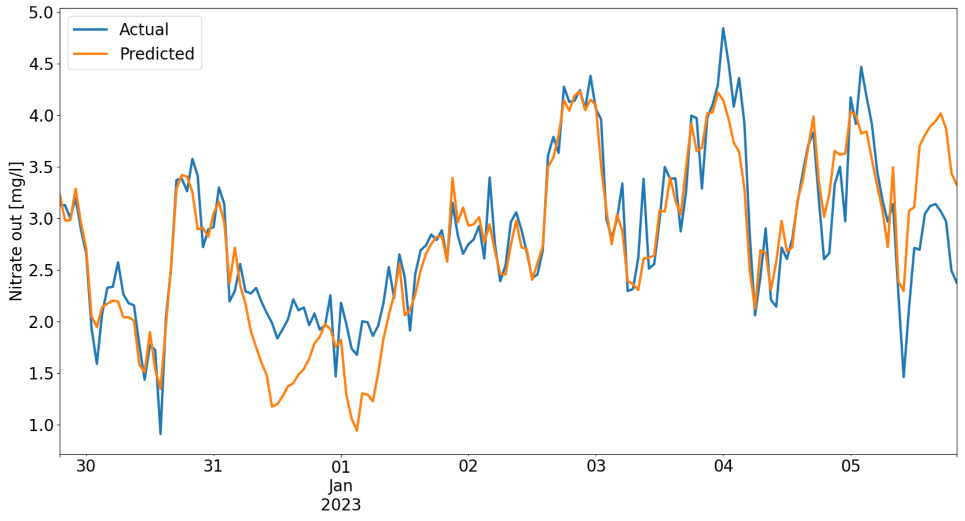


Figure 4.7: Predicted values generated by the SARIMAX model, compared to the actual values observed when methanol was dosed without feedback.

4.2.4 Summary of the results when methanol was dosed without feedback

Table 4.3 shows the performance of all models compared to the baseline. All models are quite equal in terms of performance but based on the metrics used, the SARIMAX model is the best performing model. The baseline is outperformed by all algorithms which indicates that it is possible to obtain reasonably good results on the process without feedback. Additionally, the examination of feature importance for both the XGBoost and SARIMAX models, along with the analysis of the model parameters of the SARIMAX models have indicated that lagged features are not significant in the prediction of nitrate out values when methanol is dosed without feedback.

	MSE	**RMSE**	**MAE**
XGBoost	0.20	0.45	0.35
LSTM	0.16	0.40	0.31
SARIMAX	0.15	0.39	0.29
Baseline	7.17	2.67	1.92

Table 4.3: Performance overview of the chosen models on the process when methanol was dosed without feedback.

4.3 Results on methanol dosage with feedback

4.3.1 XGBoost

The best parameter combination obtained from the *gridSearchCV* are shown in Table 4.4 and reg:squarederror was used as objective function. Figure 4.4 displays the predicted values for the test period. The model captures the general trend of the data during the first half of the test period, but the predictions deviate significantly from the actual values in the second half of the period. Specifically, the predicted values tend to be overestimated.

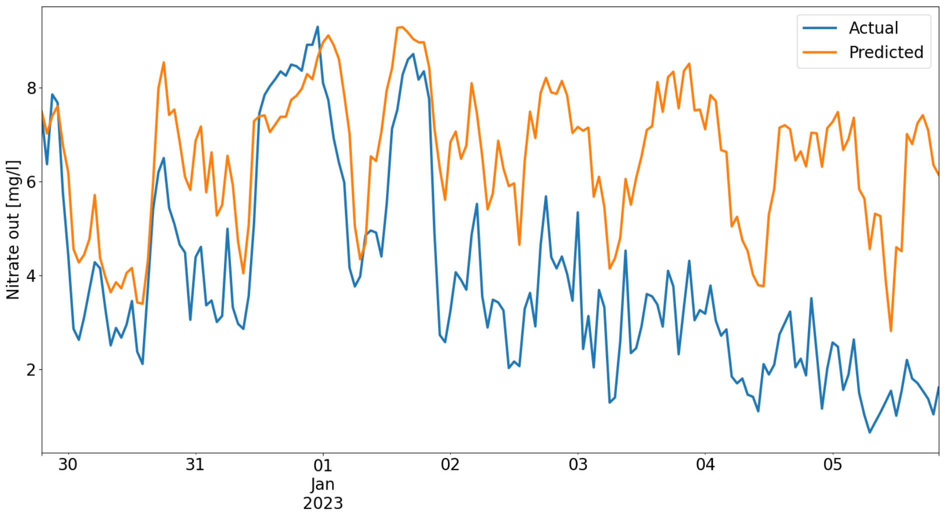


Figure 4.8: Predicted values generated by the XGBoost model compared to the actual values observed during the testing period when methanol was dosed with feedback.

n_estimators	max_depth	eta
30	4	0.3

Table 4.4: The best parameters for XGBoost on predicting nitrate out when methanol was dosed with feedback.

Figure 4.9 shows the feature importance for XGBoost on the process with feedback. Based on the feature analysis of the XGBoost model the three most important features are oxygen, methanol, and level of the current time point. Additionally, the XGBoost model trained on the process where methanol was dosed with feedback tends to give priority to the current or lag one values of the features. Seven of the top ten most important features are current values or values from last time step.

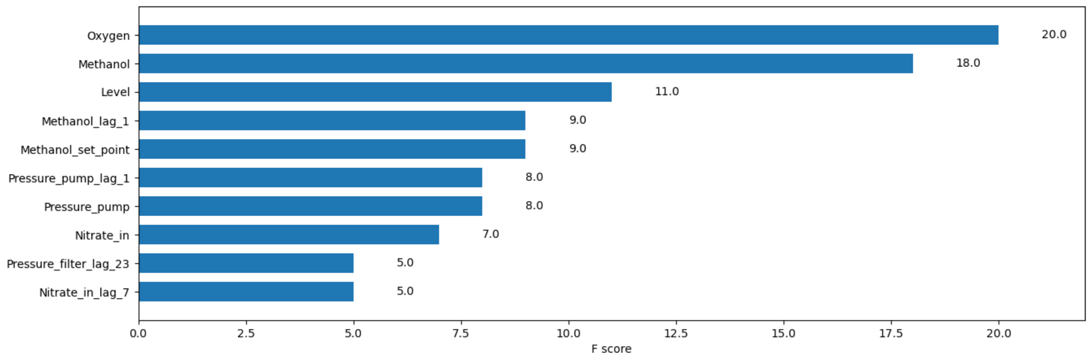


Figure 4.9: The ten most important features for XGBoost when methanol was dosed with feedback.

4.3.2 LSTM

In Figure 4.5 we see the predictions versus the actual value for nitrate out in the test period for the LSTM. The LSTM model effectively captures the overall trend in the first portion of the training data with some overshooting on some tops. The model appears to struggle with the second half of the test data. The predictions generated by the LSTM model has a lack of variability in comparison to the actual values. The optimal model parameters are shown in Table 4.5.

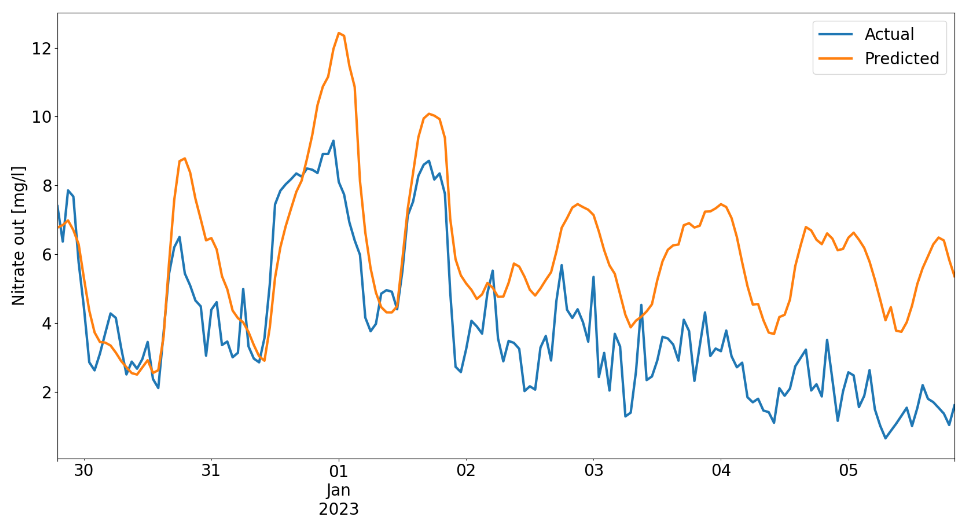


Figure 4.10: Predicted values generated by the LSTM model, compared to the actual values observed during the test period when methanol was dosed with feedback.

units	**activation**	**dropout**	**optimizer**
40	relu	0.1	adam

Table 4.5: The best parameter combination for LSTM when methanol was dosed with feedback.

4.3.3 SARIMAX

Figure 4.11 displays the predicted values for the test period for the SARIMAX model. As the other models, SARIMAX can predict the trend in the first half of the test set, but for the second part of the test set the model struggles to predict the nitrate out values. The optimal model parameters $(p, d, q)(P, D, Q, S)$ were (3,0,2)(1,0,0,24) for SARIMAX. This indicates that the model emphasizes previous values more than the SARIMAX model for the process where methanol was dosage without feedback. The most important exogenous variables for predicting nitrate out were RPM, flow filter, temp out, methanol and nitrate in. Both SARIMAX models emphasise methanol, temp out and flow filter for predicting nitrate out.

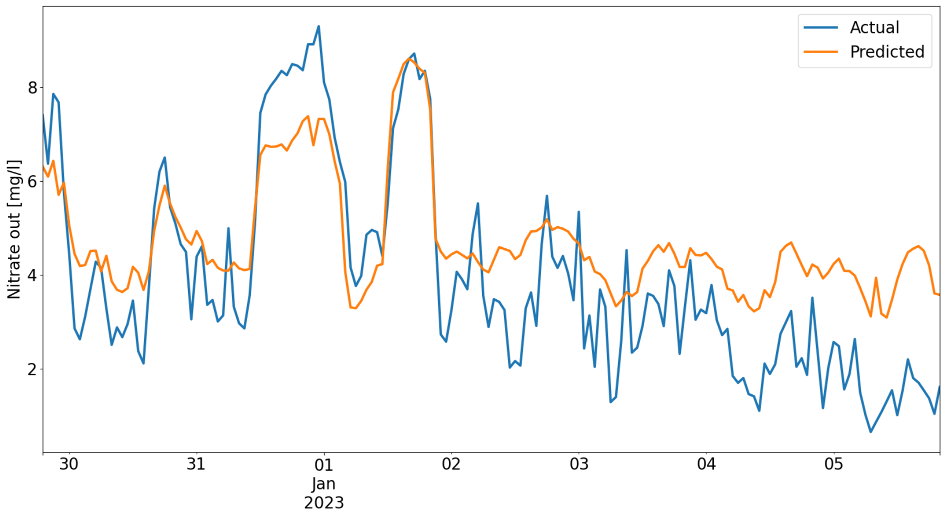


Figure 4.11: Predicted values generated by the SARIMAX model, compared to the actual values observed during the testing period when methanol was dosed with feedback.

4.3.4 Summary of the results when methanol was dosed with feedback

Table 4.6 shows the performance of the models trained on the process with feedback and the baseline. we can see that the SARIMAX model outperformance the XGBoost and LSTM, this observation corresponds to the visual inspection. SARIMAX is the only model performing better than the baseline. The XGBoost model do not emphasize lagged values and the SARIMAX model trained on the data with feedback emphasize to some extend lagged values more than the SARIMAX model trained on without feedback.

	MSE	**RMSE**	**MAE**
XGBoost	9.40	3.07	2.68
LSTM	6.94	2.63	2.24
SARIMAX	2.09	1.45	1.24
Baseline	5.77	2.40	1.71

Table 4.6: Performance overview of the best performing models when methanol was dosed with feedback.

Chapter 5

Discussion

5.1 The data sets

The datasets utilized to train the machine learning models in this thesis represent measurements taken during a relatively brief period during winter. Given that machine learning models only can learn information contained in the training data, it is reasonable to assume that the models developed in this study would be applicable to only a limited range of periods. This inference is based on the understanding that several of the features employed in this study exhibit natural variability over the course of a year. For instance, the wastewater temperature tends to be significantly higher during the summer season than during winter. It is reasonable to assume that the increased temperatures will influence the denitrification process, and therefore could the information learned during winter not be as applicable anymore. As a result, it is unlikely that the models developed in this thesis would generalize well to non-winter periods.

The two data sets utilized in this study have several correlated features, as described in Section 4.1. Highly correlated features can lead to less model interpretation [20]. This can be shown with a short example. If we have the equation $Y = X_0 + W_1X_1 + W_2X_2$, where the value of the coefficients (W) give us a sense of how important the coefficient is for predicting Y. In case one: $Y = 8$, $X_0 = 0$, $X_1 = 2$ and $X_2 = 4$, and in case two: $Y = 16$, $X_0 = 0$, $X_1 = 4$ and $X_2 = 8$. Then both $Y = X_0 + 4X_1 + 0X_2$ and $Y = X_0 + 0X_1 + 2X_2$ would satisfy the equation. In the first equation W_1 would have a higher impact on Y while in equation two W_2 would have a higher impact. This indicates that both combinations of coefficients can perfectly fit the data and that it is somewhat random which parameter the model prefers. Since we have many correlated features, it might be difficult to conclude which parameters are significant for predicting the amount of nitrate contained in the wastewater after denitrification.

Highly correlated data sets do not influence the model prediction in a negative way, but can lead to more computational expensive models because the models must take several parameters into account [20]. One possible way to reduce the dimension of the data without losing too much of the information in the data set is to use principal component analysis (PCA).

5.2 Differences between the process when methanol is dosed with and without feedback

From Table 4.3 and 4.6 we can observe that the models trained on the process where methanol is dosed without feedback yield better results than when methanol is dosed with feedback. In Section 3.1 we assumed that the features affecting the denitrification process would be equal for both process halls because the measurements are from the same time period. This assumption was not completely correct because the process halls in periods had significantly different input flows of wastewater. The difference in flow led to periods with greater input flow to the process hall where methanol was dosed with feedback. When there is high flow, the wastewater uses shorter time to go through the denitrification-filters. This leads to less denitrification and therefore higher nitrate out values. We assume this is the main reason for why the process with feedback had higher nitrate out values than the process without feedback. The differences in operation conditions could be one reason for the difference in prediction performance.

Another reason why there is a prediction difference between the two processes could be due to how the models are developed. The models are developed to predict the amount of nitrate contained in the wastewater after denitrification given some input variable. In other words, a function is trained for predicting an outcome based on some input without having some feedback. In the process without feedback, the methanol is dosed based on observable measurements before denitrification and thereby no feedback in the process. Hence it is possible for the models to learn a cause and effect between the features and the amount of nitrate after denitrification. On the other hand, when methanol is dosed based on previous nitrate out values, we have a feedback loop. Consequently, we are trying to learn a process with a feedback loop using a model that does not have a feedback loop. This makes it harder for the model to learn the cause and effect between the features and targets. The addition of feedback is most likely one reason for why it is easier to predict nitrate out values for the process without feedback.

5.3 Struggles with feedback

In Section 4.3 we observed that when methanol was dosed with feedback, all models struggled with predicting the nitrate out value after about halfway into the test data. This sudden drop in performance may be due to process specific changes that the model is not capable of predicting. One process-specific change we are aware of in this period was the increased supply of melted snow to the wastewater treatment plant. The increase in supply of melted snow might change the denitrification process in a way that the trained models do not have any information about, and therefore are we experiencing a drop in performance. One reason for this could be that the training data does not contain any information about increased supply of meltwater to the wastewater treatment plant. As mentioned earlier, the dosage strategy with feedback is a more complex strategy which makes it harder for the model to learn a cause and effect between the features and target. The increased supply of meltwater could lead to changes in the amount of methanol and therefore also change the cause and effect. On the other hand, the results on the first half of the test data when methanol was dosed with feedback showed promising results. In this period all models can predict the general trend in nitrate out values. This might indicate that it is possible to predict nitrate out values if the process is stable.

Chapter 6

Conclusion and recommendations

6.1 Conclusion

In this study we have investigated the possibility to predict the amount of nitrate after denitrification for two distinct processes at Veas. In the first process the methanol was added to the wastewater without feedback. For this process the SARIMAX was the best performing model, and we obtained a MSE, RMSE and MAE of 0.15, 0.39 and 0.29 respectively. The difference between the worst and best performing model was 0.05 for all measurements. All models outperformed the baseline for this process. This observation indicates that it is possible to predict the amount of nitrate after denitrification for the data tested for in this study.

In the second process the methanol was added with feedback. Process two is the normal dosing strategy for Veas and is therefore a more realistic process. We found that this process is much more difficult to make good predictions on. The main reason for this is assumed to be because of the feedback of prior values which makes it more difficult for the models to learn a cause and effect. The SARIMAX model was the best performing model on this process with an MSE, RMSE and MAE of 2.09, 1.45 and 1,24 respectively. The SARIMAX model was the only one outperforming the baseline.

The feature evaluation in this study has shown that the models emphasize recent features in the predictions. Adding information about the past does not necessarily lead to better performing models for this study. Both SARIMAX models and XGBoost models emphasized methanol values for predicting nitrate out.

6.2 Recommendations

One possible solution for overcoming the challenges associated with the more complex modelling task when methanol is added with feedback is to include more data during training. The variation in the data has shown to be difficult to make good models for, therefore we assume that more data will not necessarily lead to better performing models. Therefore, we suggest including more training data but first apply a clustering algorithm to the training data. The clusters identified by the algorithm would consist of similar samples and subsequently separate algorithms could be trained on each of the clusters. When making a prediction on a sample, the model trained on the cluster most similar to the sample is used for prediction. Alternatively, a combined prediction of all models where the models are weighted according to the similarity to the sample. The idea behind this approach is that several specialized machine learning algorithms are better than one big machine learning algorithm in predicting the amount of nitrate after denitrification.

Bibliography

[1] Proposal for a revised urban wastewater treatment directive, Oct 2022.

[2] Klima- og miljødepartementet. Revisjon av avløpsdirektivet. `https://www.regjeringen.no/no/sub/eos-notatbasen/notatene/2021/des/revisjon-av-avlopsdirektivet/id2966230/`, 2021. Accessed: 07.04.2023.

[3] Lovdata. Forskrift om begrensning av forurensning (forurensningsforskriften). `https://lovdata.no/dokument/SF/forskrift/2004-06-01-931/KAPITTEL_4#KAPITTEL_4`, 2007. Accessed: 03.04.2023.

[4] Norskvann. Forskrift om begrensning av forurensning (forurensningsforskriften). `https://norskvann.no/wp-content/uploads/V2_Gjennomgang-av-forslag-til-nytt-avlopsdirektiv-fra-EU.pdf`, 2022. Accessed: 03.04.2023.

[5] European commission. Protecting waters against pollution caused by nitrates from agricultural sources. `https://environment.ec.europa.eu/topics/water/nitrates_en`, 2021. Accessed: 07.04.2023.

[6] United states enviormental protection agency. Climate change and harmful algal blooms. `https://www.epa.gov/nutrientpollution/climate-change-and-harmful-algal-blooms`, 2022. Accessed: 07.04.2023.

[7] H Oedegaard. Oversikt over metoder for fjerning av nitrogen. *Vann*, 1980.

[8] Bruce E Rittmann, Joshua P Boltz, Doris Brockmann, Glen T Daigger, Eberhard Morgenroth, Kim Helleshøj Sørensen, Imre Takács, Mark Van Loosdrecht, and Peter A Vanrolleghem. A framework for good biofilm reactor modeling practice (gbrmp). *Water Science and Technology*, 77(5):1149–1164, 2018.

[9] Arthur L Samuel. Machine learning. *The Technology Review*, 62(1):42–45, 1959.

[10] Rob Toews. Synthetic data is about to transform artificial intelligence. `https://www.forbes.com/sites/robtoews/2022/06/12/synthetic-data-is-about-to-transform-artificial-intelligence/?sh=59f22d597523`, 2022. Accessed: 28.04.2023.

[11] Sebastian Raschka and Vahid Mirjalili. *Python Machine learning.* Packet publishing Ltd., 35 Living street, UK, 2019.

[12] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).*, 9:381–386, 2020.

[13] Cory Maklin. Gradient boosting decision tree algorithm explained. `https://towardsdatascience.com/machine-learning-part-18-boosting-algorithms-gradient-boosting-in-python-ef5ae6965be4`, 2019. Accessed: 04.05.2023.

[14] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[15] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. Springer, 2002.

[16] Brendan Artley. Time series forecasting with arima, sarima and sarimax. `https://towardsdatascience.com/time-series-forecasting-with-arima-sarima-and-sarimax-ee61099e78f6`, 2022. Accessed: 01.05.2023.

[17] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[18] Jason Brownlee. Regression metrics for machine learning. `https://machinelearningmastery.com/regression-metrics-for-machine-learning/`, 2021. Accessed: 18.04.2023.

[19] Alina Zhang. Feature importance in linear models: Four often neglected but crucial pitfalls. `https://towardsdatascience.com/feature-importance-in-linear-models-four-often-neglected-but-crucial-pitfalls-e5c513e45b18`, 2021. Accessed: 10.05.2023.

[20] Tarek Ghanoum. Why multicollinearity isn't an issue in machine learning. `https://towardsdatascience.com/why-multicollinearity-isnt-an-issue-in-machine-learning-5c9aa2f1a83a`, 2022. Accessed: 10.05.2023.