Norwegian University
of Life Sciences

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

Preprocessing of Industrial Time Series for Soft Sensor Development

Astrid Hæve Sedal
Data Science

Preface

This thesis is my final work after five years at the Norwegian University of Life Sciences at Ås, Norway, and marks the end of a Master of science in data science. The thesis was written from January until May 2023 and has challenged both my programming and academic skills. I am grateful for the experience and knowledge I have gained during the writing process.

I want to thank my supervisor at NMBU, Kristian Hovde Liland, for his valuable guidance and correspondence. I must also thank my co-supervisor at Nofima, Ingrid Måge, for her guidance vital to my progress. In addition, thanks to Marco Cattaldo for providing process understanding and important inputs on the progress.

To Stig Sander, I am forever grateful for your encouragement, positivity, and inspiration. Lastly, a special thanks to my family and friends, especially my dearest mother, for always being my biggest support.

Astrid Hæve Sedal
Ås, May 14, 2023

Abstract

With increasing population growth, the demand for food and proteins rises. The need for protein from poultry is projected to increase by 17.8% by 2030, and poultry production in Norway has never been higher. Nortura at Hæland/Norway processes chicken and is left with a significant amount of rest raw materials after production. To deal with waste management, Bioco, a bio-refining company located next to Nortura, has utilized the rest raw materials from Nortura. Through enzymatic hydrolysis, Bioco can transform the rest raw materials into high-quality end products while reducing waste and contributing to sustainable food production. However, the process line is prone to variation as the composition of the rest raw materials varies greatly. The sensors along the process line have gathered a vast amount of complex time series data that must be preprocessed to develop soft sensors.

This thesis presents multiple preprocessing steps, focusing on data filtering, feature engineering, and time series smoothing. Data filtering removed irrelevant spectra and process data, while feature engineering aimed to reduce the dimensions of the near-infrared data and created new features to enhance model performance. Smoothing eliminated noise in the time series data and revealed underlying patterns. Four smoothing methods were performed with a great range of window sizes. The methods were mean, median, linearly, and exponentially weighted smoothing. Finally, the preprocessed data were utilized as input data in two soft sensors, also termed machine learning models, to compare the impact of preprocessing on the soft sensors.

The application of mean and median smoothing methods resulted in smoothed time series, which had varying impacts on the performance of the machine learning models. While mean smoothing did not surpass the baseline model at lower window sizes, it demonstrated better results with larger window sizes. On the other hand, median smoothing led to more reliable predictions that performed well overall. In comparison, the linearly weighted mean generated the smoothest time series among the four methods and produced the most accurate predictions for larger window sizes. Exponential smoothing, however, yielded similar time series across different window sizes and did not significantly smooth the time series and may be related to the decay parameter used. Regarding the performance of the machine learning models, XGBRegressor predicted noisier time series than the RandomForestRegressor.

There are multiple remaining challenges regarding the preprocessing methods. The near-infrared data should be preprocessed in the spectral domain to correct multiplicative scattering effects, and different train/test splits should be performed, including cross-validation. Adding a smaller lagged feature of the target data to reveal temporal relationships is also recommended.

Due to the varied rest raw materials at Bioco, the process line sensors generate complex time series data with noise. Data filtering, feature engineering, and smoothing improved data quality. The results from the soft sensors highlighted the importance of choosing the right smoothing method and window size as a preprocessing step. Median smoothing with larger window sizes resulted in the most robust predictions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The production of poultry in Norway has never been higher and has increased 173% from 42 206 tons in 2001 to 115 494 tons in 2021 [1]. Rest raw materials from the chicken industry answer to around 25-30 000 tons yearly [2]. According to Norilia, the rest raw materials within the meat and egg industry account for 35% of the biomass [3] whereas Nofima put forward that as much as 51% of the chicken is categorized as the rest raw materials [4]. This gives rise to making the industry more sustainable regarding using available resources and fully utilizing the whole animal.

According to the Food and Agriculture Organization of the United Nations (FAO), population growth is expected to result in an increase in protein demand, and protein from poultry is projected to increase by 17.8% within 2030. Poultry is a more attractive protein source in low-income and high-income countries due to lower prices than other types of meat. Besides, poultry is seen as a healthier option regarding the high protein/low-fat ratio it holds [5]. Furthermore, in a high-income country such as Norway, consumers have a higher standard regarding food presentation, and subsequently, food that does not meet these standards is discarded [6]. Additionally, products made from the rest raw materials will meet higher demands for protein and hold several health advances such as reduced inflammation and cholesterol levels, thus making it more attractive for a broader range of people [7].
Not only is the use of rest raw materials beneficial when meeting higher protein demands, but it also covers specific sustainable development goals put forward by the United Nations. Sustainable development goal 12 is about ensuring sustainable consumption and production patterns, as humans today have significantly higher consumption than what is considered sustainable. Target 12.3 and 12.5 both aim toward the production line; thus, the utilization of the rest raw materials from the chicken industry is of great interest. The sustainability goals are formulated as follows [8].

> **SDG 12.3** By 2030, halve per capita global food waste at the retail and consumer levels and reduce food losses along production and supply chains, including post-harvest losses.
>
> **SDG 12.5** By 2030, substantially reduce waste generation through prevention, reduction, recycling, and reuse.

Bioco, a biorefining company in Hærland/Norway, uses the rest raw materials from chicken and turkey and turns them into high-quality protein products using a continuous hydrolysis process. The process uses hydrolysis enzyme to break down the rest raw materials into high-value products for both human consumption and pet food. In cooperation with Nofima at Ås/Norway, Bioco aims at new product innovation through process control and smart sensors. The work started in a project called SmartBio [9] and continues in SFI Digifoods. As a result, large

amounts of sensor data are being collected to gain insight into the process.

The rise of new and available technology in the later years has led to the term Industry 4.0. The goal of Industry 4.0 is to enhance efficiency by utilizing technology such as smart sensors. Smart sensors generate loads of data that can be used to improve product quality and efficiency and further make digital objects, namely digital twins. By collecting and modeling this kind of data, one can develop artificial intelligence that supports decision-making, regulates the process line, or predicts maintenance before the breakdown occurs. However, to use the data generated, the data must be preprocessed and analyzed.

Data preprocessing is a vital step in data analysis, where the goal is to acquire an in-depth understanding of the data. The task at hand may be considerable, and it is said that data preprocessing answers for 80% of the data science projects [10]. Furthermore, the preprocessing of time series may differ from other types of data as the sequential order of the data holds relevant itself. Thus the preprocessing must be handled carefully to retain that kind of information. Through different preprocessing techniques, the aim is to reveal underlying patterns and gain valuable insight into the time series.

1.1 Motivation

There is great potential in collecting and analyzing data from the process line to understand the process variations and monitor the process. By optimizing the use of the rest raw materials, the quality of the end products may be more consistent. Small adjustments and improvements in the processes can significantly increase profit for Bioco. However, the gathered data are not of desired quality as they are prone to considerable variance and deviation. This is due to the variance in the rest raw materials throughout the process and sensor variance. Sensor data are vulnerable to noise and outliers and often need preprocessing for the benefits of machine learning modeling. The data must undergo time-consuming preprocessing steps to derive insight from this data.

1.2 Objectives

This thesis investigates different sensor data collected over a period of eight weeks with the object of analyzing well-known preprocessing methods of time series and the effects they have on machine learning models.
The objectives can be summarized as the following research questions.

- What kind of preprocessing steps are needed for industrial time series?
- How do different smoothing methods and window sizes affect the predictions of the pressures in the hydrolysis pipe?

Chapter 2

Theory

2.1 Rest raw materials

Rest raw material, also referred to as by-products or co-products, can be defined as materials left after the slaughtering, such as bones, skin, and carcasses [11]. Co-products can be used for human consumption, whereas by-products are not and are further heavily regulated. Moreover, the rest raw materials can be divided into three categories based on the risk to the public health or the health of animals. The categories range from one to three, where one is the highest risk and three is the lowest risk. The categories are listed as follows.

1. The rest raw materials are at high risk for diseases and thus only intended for disposal and neither human nor animal consumption.
2. The rest raw materials have a high risk for contamination and are not intended for animal consumption but can be used for bio-gas or land-filling.
3. The rest raw materials are of low risk and can be processed into another end product for human or animal consumption.

The rest raw materials used at Bioco are in category three, implying that the by-products are regulated for human consumption. To retrieve the high-quality protein products from the rest raw materials, a method named *enzymatic protein hydrolysis* can be used to extract proteins from poultry. Enzymatic hydrolysis breaks the protein molecules into smaller components through a reaction with water and the use of enzymes. In contrast to chemical hydrolysis, using enzymes is a milder and more controllable process in addition to preserving the nutritional aspect too [12].

2.2 Near infrared spectroscopy

Sir William Herschel's discovery of near-infrared light in 1800 was a fortuitous moment in the history of science. Herschel attempted to identify which colors of visible light carried the most heat and observed that the temperature increased as he moved from violet to red on the spectrum. However, he discovered that the highest temperature was not found in the red part of the spectrum, but beyond it, in what we now refer to as the near-infrared region. This groundbreaking discovery paved the way for numerous scientific applications of near-infrared light, including remote sensing, astronomy, and medical imaging [13].

The near-infrared light range spans from 800 nm to 2500 nm in the electromagnetic spectrum. Near-infrared spectroscopy involves illuminating a substance with near-infrared light and then

capturing and analyzing the alterations in the emitted light. The majority of molecules in a substance have vibrations with frequencies situated within the near-infrared area of the electromagnetic spectrum. These vibrations occur due to competing attractive and repulsive forces between the positive atom core and negative electrons. When light with the same frequency as these vibrations is shone through a substance, the light gets absorbed and the frequency of the light decreases. This leads to distinctive infrared spectra with sharp peaks that can detect chemical constituents such as proteins, fats, or water content [14].

There are mainly three different ways to light a sample, and they are chosen depending on the compounds to be measured. The three different methods are *diffuse reflection*, *transmission mode*, and *transflection mode*. Transmission mode is when the light is shone directly at the sample where some wavelengths are absorbed whereas most of the wavelengths are transmitted to the sensor. Transflection mode is similar to transmission mode but differs in that sense that there is a mirror placed behind the sample where the shone light will reflect back through the sample and onto the sensor. Further, reflection mode has the sensor placed above the sample and the light emitted is reflected back onto the sensor above [15]. The three different modes are illustrated in Figure 2.1.

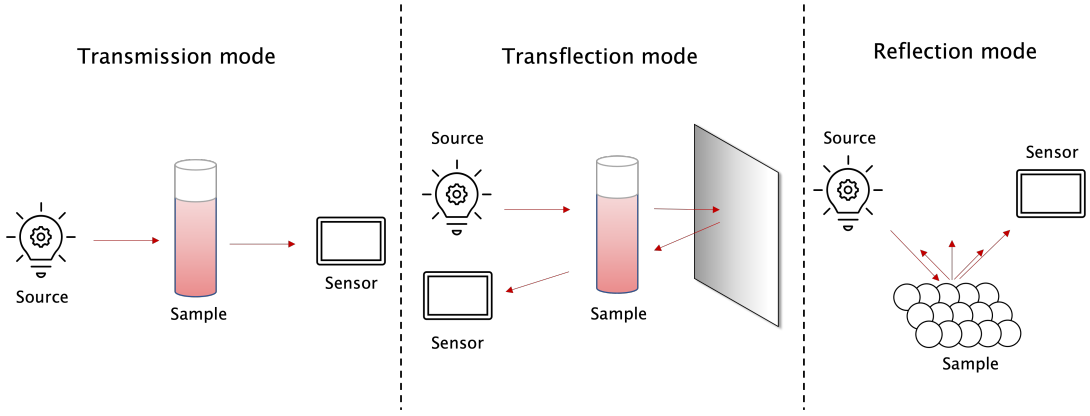


Figure 2.1: Figure illustrating near-infrared measure techniques. There are mainly three different methods to light a sample. Transmission, transflection, and reflection mode.

Furthermore, interactance measurement is a kind of image spectroscopy where the light is transmitted into the sample, and the sensor does not measure the direct reflected light but rather the light traversing the sample. Interactance imaging is shown to be suitable for non-contact and high-speed online measurements of food [16].

2.3 Time series

Sequential data are a type of data where the order of the sequence matter. In contrast to other types of data used in machine learning, sequential data are not independent and identically distributed (I.I.D.), meaning the data are not statistically independent and not identically distributed. Time series is a special type of sequential data where the axis along the data is referred to as *time* [17]. A time series is a collection of values retained from measurements over time, resulting in data in chronological and temporal order. The *index* of a time series is symbolized in Equation 2.1

$$T = \{t_{min}, t_{min} + 1, \ldots, t_{max-1}, t_{max}\} \tag{2.1}$$

where $t = 1$ is the first observed value, and $t = T$ is the last observed value. On the one hand,

one often talks about a *univariate* time series which only depends on the past values of itself and can be defined as Equation 2.2 [18]. Here, x_t is a vector of values at

$$x_t = (x_1, \ldots, x_T) \text{ for } x_t \in \mathbb{R} \tag{2.2}$$

On the other hand, a time series can be multivariate, meaning that the time series has several parameters collected over time, measured with the same time steps. In Equation 2.3, multivariate time series is defined.

$$\mathbf{x_t} = (x_t^{(1)}, \ldots, x_t^{(n)}) \in \mathbb{R}^n \tag{2.3}$$

Time series can be further decomposed into three components and described as sum (Equation 2.4) or product (Equation 2.5) of *trend*, *seasonality* and *noise*.

$$x_t = T_t + S_t + E_t \tag{2.4}$$

$$x_t = T_t * S_t * E_t \tag{2.5}$$

The trend component is a non-periodic parameter that indicates a systematic variation over time. Seasonality, however, is a periodic parameter that implies a recurring behavior over time. The last component, noise, is an error term depicting random noise in the data.
These components can further be used in exploratory analysis, detect anomalies, and handle missing values in a data set [19]

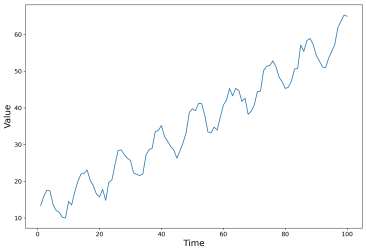
2.3.1 Autocorrelation

Autocorrelation is often used to reveal underlying patterns in a time series by investigating the time series with a lagged version of itself and the linear relationship between them. The autocorrelation coefficient r_k is defined in Equation 2.6 where T is the time series length. These autocorrelation coefficients compose the *autocorrelation function*, also known as ACF.

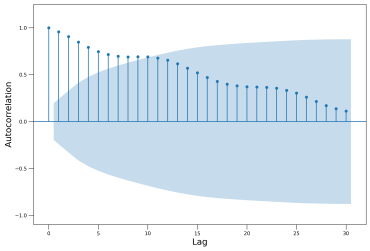
$$r_k = \frac{\sum_{t=k+1}^{T}(x_t - \overline{x})(x_{t-k} - \overline{x})}{\sum_{t=1}^{T}(x_t - \overline{x})^2} \tag{2.6}$$

If x_t and x_{t-1} are correlated, then x_{t-1} and x_{t-2} are also correlated. Further, if x_t and x_{t-2} are also correlated, it might be because they both are connected to x_{t-1}, instead of holding any new information. This problem is what *partial autocorrelation* addresses. Partial autocorrelation measures the linear relationship between x_t and x_{t-k}, but in contrast to autocorrelation, the information retained by other times lags, $1, 2, \ldots, k-1$, is removed.

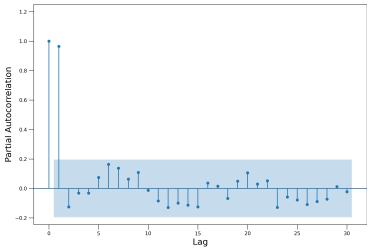
From an autocorrelation plot, it is possible to detect both seasonality and trend. Seasonality can be detected as there will be spikes at the seasonal lags. If there is a trend in the time series, the autocorrelations will have higher values as the value of time points close in time are also close in value. In Figure 2.2, there is a time series with a positive trend as well as seasonality seen in Figure 2.2a. These components can also be seen in the autocorrelation plot in Figure 2.2b, where autocorrelation coefficients are high in value, and there are spikes at lags corresponding with the seasonality. In Figure 2.2c, the first partial autocorrelation is equal to the first autocorrelation, and in this example data, the second partial autocorrelation shows a high correlation. Further, the other lags are within the 95% confidence interval demonstrated by a blue ribbon [20].



(a) Time series with trend and seasonality



(b) Corresponding autocorrelation plot



(c) Corresponding partial autocorrelation plot

Figure 2.2: Examples of autocorrelation and partial autocorrelation in times series.

The ACF and PACF plots can be guidelines when selecting appropriate forecasting models, such as an autoregressive (AR) or moving average (MA) model. AR models use a linear combination of past values to predict future values, whereas MA models use a linear combination of past error terms to forecast new values. The mathematics behind these models will not be covered in this thesis. However, the concept of using ACF and PACF plots for model selection will be explained as the plots provide information about the correlation between values.
If the ACF plot is exponentially decaying and the PACF plot shows only one significant spike at lag 1, the order of the AR model can be set to one, thus yielding AR(1). Generally, if the ACF is exponentially decaying and PACF zero shows p significant spike after lag q, it yields AR(p) model. Moreover, if the PACF plot is exponentially decaying and the ACF plot only shows one significant spike at lag 1, an MA(1) model would be appropriate. Subsequently, if the ACF plot shows zero significant lags after lag p and the PACF plot is exponentially decaying, the MA(q) is an appropriate model [20].

2.3.2 Baseline method

Baseline methods are simple yet highly efficient methods used to forecast time series and are used to benchmark more complex models such as machine learning models. The following subsections will present four common baseline methods within time series forecasting.

Average method

A simple baseline method for forecasting is the *average method*. The average method is based only on historical data and forecasts future values as the average of this historical data. The average method is symbolized in Equation 2.7 where $y_{T+h|T}$ means that the forecast of y_{T+h} is based on the data $(y_1 + \cdots + y_T)$. T is the time series length.

$$\begin{aligned} \hat{y}_{T+h|T} &= \overline{y} \\ &= \frac{(y_1 + \cdots + y_T)}{T} \end{aligned} \tag{2.7}$$

Naïve method

In contrast to the average method, the naïve method uses only one past value to forecast future values. This method uses the last observed value, y_T, and sets all the future values equal to this value as seen in Equation 2.8.

$$\hat{y}_{T+h|T} = y_T \tag{2.8}$$

Seasonal naïve method

The seasonal naïve method is relatively similar to the naïve method. The difference is that the seasonal naïve method considers the seasonal pattern when estimating future values. The method takes the last observed value from the last period to forecast future values. In Equation 2.9 the method is symbolized where p is the seasonal period and $k = \lfloor \frac{h-1}{p} \rfloor$.

$$\hat{y}_{T+h|T} = y_{T+h-p(k+1)} \tag{2.9}$$

Drift method

Similar to the naïve method, the drift method uses the last observed value. However, the drift method also considers the trend by adding the average drift. Drift can be defined as the amount of change over time [21]. The drift method is shown in Equation 2.10

$$\begin{aligned} \hat{y}_{T+h|T} &= y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) \\ &= y_T + h\left(\frac{y_T - y_1}{T-1}\right) \end{aligned} \tag{2.10}$$

2.4 Preprocessing of time series

Data analysis aims to find knowledge in data and to use this knowledge in decision-making or problem-solving processes. Data may arise from several sources, and with the increase in technology, the volume of available data also arises. However, one major challenge when working with real-world data is that the data are often imperfect, containing various sources of errors, noise, and missing values [22]. Preparing the data is undoubtedly the most time-consuming part of working with data [23].
Data preprocessing can be defined as the action taken before the actual data analysis, which aims to enhance the data quality. A. Famili et al. present in [22] three main reasons for data preprocessing.

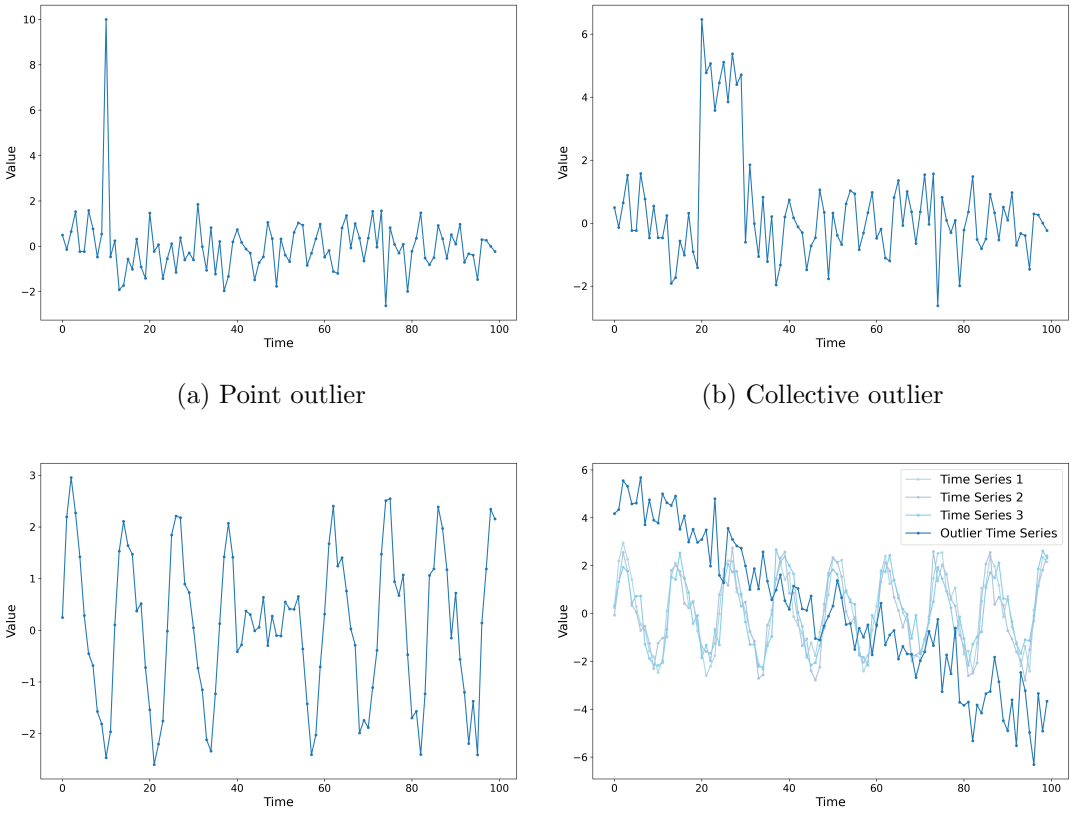
- Data preprocessing may solve problems that prevent further analysis.
- Data preprocessing may gain an understanding of the data that enhances the upcoming analysis.
- Data preprocessing may yield a more meaningful knowledge of the given data.

2.4.1 Outlier detection

Regarding time series, there are several definitions of the term outliers. Generally, an outlier can be said to be an observation that differs from the rest as it is generated by another mechanism. Furthermore, two types of terms are often used to describe these kinds of abnormal observations. *Outliers* and *anomalies*. The former is frequently used to describe extreme values that differ from the expected values and can be said to be unwanted data. In contrast, the latter aims more towards events of interest and often several data points.

Outliers can be sorted into three categories such as *point outliers*, *collective outliers*, and *contextual outliers* [24]. Point outliers are specific points in the timeline that deviate from local neighbor points or in a global context. Figure 2.3a shows how a point outlier can be an extreme value compared to the other values. Collective outliers can also be local or global point outliers

but differ from single point outliers because they are sequential points that deviate from the remaining data points, as seen in Figure 2.3b. Contextual outliers may be more challenging to discover as the data points may range within the global values. However, when compared to the local neighbors or, in other words, when considering the context, the values are outliers. This can be seen in Figure 2.3c. Additionally, a whole time series may be an outlier in multivariate time series as the time series differ from the behavior of the other time series. The blue time series in Figure 2.3d follow a different pattern than the other time series and is, therefore, an outlier in itself [24].



(a) Point outlier

(b) Collective outlier

(c) Contextual outlier

(d) Time series outlier

Figure 2.3: Illustrations of different types of outliers.

Z-score

A basic outlier detection is the statistical Z-score. The Z-score of a data point is the number of standard deviations it falls below or above the mean of the data set [25].

$$Z = \frac{x_t - \mu}{\sigma} \tag{2.11}$$

where x_t is the data point at time t, μ is the mean of the data set and σ is standard deviation of the data set. $Z = 0$ means that the data point is equivalent to the mean, μ. Furthermore, a data point is said to be an outlier if the Z-score is greater than the given threshold τ. This threshold is often set to $\tau = 3$, and an outlier can then be defined as the following:

$$Z > \tau \tag{2.12}$$

As the Z-score detects outliers globally, the Z-score will not capture contextual outliers such as in Figure 2.3c. Even collective outliers, as in Figure 2.3b, may be challenging to detect as these

outliers influence the mean. Furthermore, if the time series has a trend, meaning a slope, the range of values makes Z-score insufficient to detect outliers. An example of this can be seen in Figure 2.4.

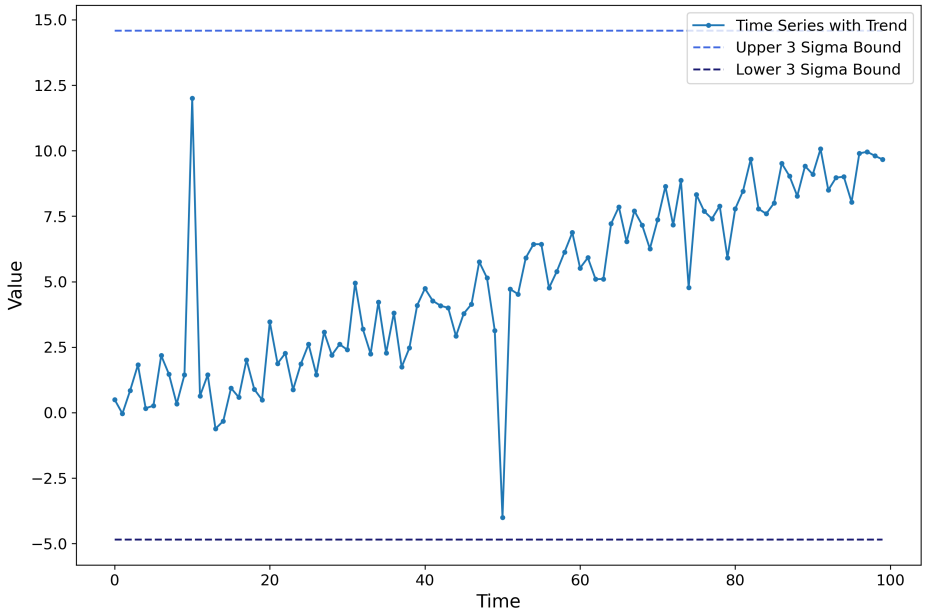


Figure 2.4: Illustration of how outliers in time series may be difficult to address. When the time series has a trend, the Z-score will not be able to detect point outliers as they are within the 3σ bounds.

2.4.2 Smoothing

Rolling mean

Rolling mean, also called moving average, is a simple smoothing technique that reduces noise in a time series and reveals underlying trends and patterns. The term "rolling" in the context of time series refers to how the mean is calculated as it is computed by sequentially "rolling" a window over the data, with the window size remaining constant and moving forward by a fixed number of observations at each step. As the window moves over the time series, the value x_t is replaced with the average of itself and a certain number of local neighbors given by the window size. The concept is based on the assumption that observations close in time are also likely to be close in value. By averaging the observation nearby in time, it is possible to obtain a reasonable estimate of the underlying trend [26].
The estimation can be written as in Equation 2.13.

$$\begin{aligned} \hat{T}_t &= \frac{1}{k} \sum_{j=-l}^{l} x_{t+j} \\ &= \frac{x_{t-l} + x_{t-l+1} + \cdots + x_t + \cdots + x_{t+l-1} + x_{t+l}}{2l+1} \end{aligned} \tag{2.13}$$

Here, $l = \frac{k-1}{2}$ and k is the window size. When this kind of smoothing expects an odd number as input, making it *centered*. A greater k makes the time series smoother as it includes more terms

in Equation 2.13. Subsequently, time points at the beginning and the end will be excluded equal to size l, meaning that this approach is not suitable to analyze or forecast recent time points [26]. The exclusion of time points may also result in loss of information.

Rolling median

The rolling median technique shares many similarities with the rolling mean approach, with the primary difference being that it calculates the median value over a specified window, as opposed to the arithmetic mean. Rolling median is a more robust method as it reduces the impact of outliers and other extreme values.
Rolling median can be denoted Equation 2.14.

$$\hat{T}_t = Median(x_{t-l}, \ldots, x_{t+l}) \text{ for } t \in \{l+1, \ldots, t_{max} - l\} \tag{2.14}$$

Weighted rolling mean

The weighted rolling mean is a more advanced method compared to the simple rolling average as it introduces weights meaning that some observations carry greater weights than others when calculating the arithmetic mean. Weighted rolling mean yields a much smoother time series as the weights slowly increase and decrease [18].
Weighted rolling mean can be written as Equation 2.15.

$$\hat{T}_t = \sum_{j=-l}^{l} a_j x_{t+j} \tag{2.15}$$

where $l = \frac{k-1}{2}$ and the weights are stored in a_j.
An important notation is that the weights are symmetric, that is, $a_j = a_{-j}$, and that they sum to one such as $\sum_{j=-l}^{l} a_j = 1$.
The weights used in exponentially weighted smoothing can be calculated as defined in Equation 2.16.

$$a(k) = \exp^{-|k-l|/\tau} \tag{2.16}$$

Here, τ is the decay parameter. A smaller τ represents greater weight to values close in time.
Equal to exponentially weighted smoothing, linearly weighted smoothing also applies greater weight to more recent data points in a time series. The difference is how the weights are linearly decaying instead of exponentially decaying, and the linear weights are symbolized in Equation 2.17.

$$a(k) = \frac{2 \times l}{k+1} \tag{2.17}$$

2.5 Machine Learning

2.5.1 Background

Artificial intelligence is about making machines think and act like humans to enable problem-solving. The idea of making a machine think like a human has excited for hundreds of years, and already in 1950, Alan Turning proposed the question *"Can machines think?"* [27]. Since then, the technology has thrived enormously with a wide range of applications. Machine learning is a subfield of artificial intelligence and aims to discover knowledge from structured and unstructured data using self-learning algorithms. In the late 90s, machine learning became hugely popular, with the focus shifting from classical programming, where rules were applied

to retrieve answers, to giving a machine learning system the answer and training it to learn the rules.
Further, machine learning can be divided into three subsections. *Supervised learning*, *unsupervised learning* and *reinforcement learning*.

Supervised learning

Supervised learning is a method that aims to learn from labeled data and further enable predictions on new, unseen data. The term "supervised" indicates how the training data is accompanied by their corresponding output signals or labels. A model is trained to find an accurate representation of the relationship between the input value and the following output value. A simplified overview of supervised learning can be seen in Figure 2.5. Regression and classification are subcategories within supervised learning.

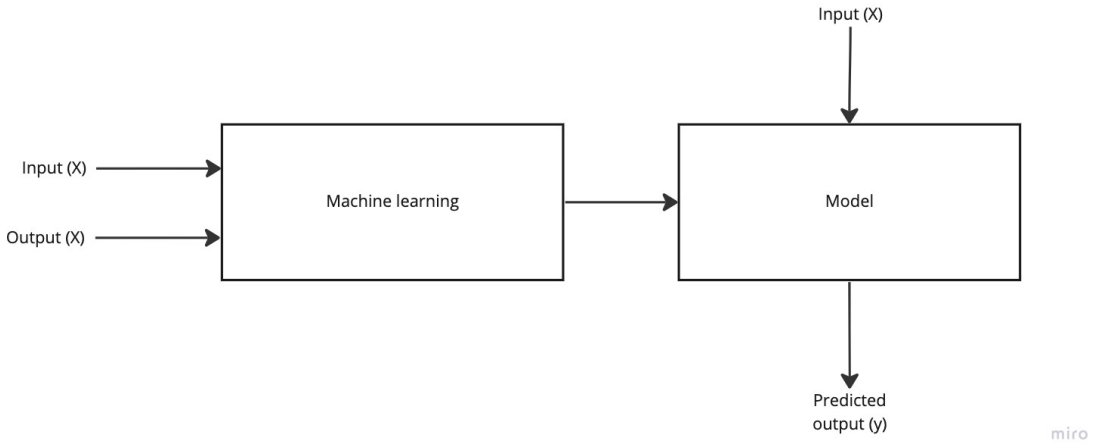


Figure 2.5: A figure showing how supervised machine learning is based on input and output values, and then new, unseen data is used to predict future values.

Unsupervised learning

Unsupervised learning is a powerful technique that can be used to analyze data where the output value is not known in advance. Unlike supervised learning, unsupervised learning does not rely on labeled data. Instead, it deals with unlabeled data of unknown structure [23]. In such scenarios, the primary goal is to uncover the underlying patterns or clusters within the data without human intervention. Clustering is a subcategory of unsupervised learning; its object is to discover the relationship between input values and group them according to similarities. On the other hand, unsupervised learning can also be used to reduce the curse of dimensionality by reducing the number of features. Together, these methods offer powerful tools for uncovering patterns and relationships within complex data sets without the need for explicit labels or prior knowledge [17].

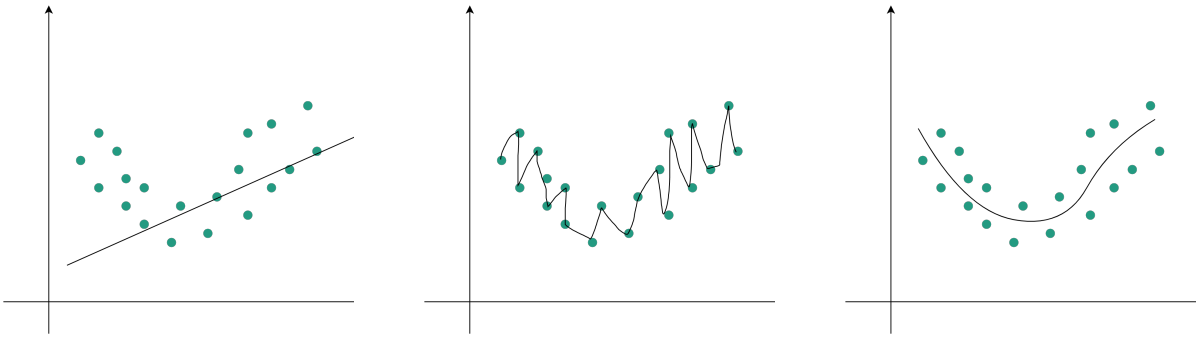
Reinforcement learning

Reinforcement learning is similar to supervised learning but differs because the algorithms do not use sample data. Reinforcement machine learning aims to train an agent to make decisions through interactions with an environment. During the learning process, the agent receives feedback through rewards or punishments based on its decisions. As a result of this exploratory trial and error approach, the agent learns actions that maximize the reward feedback [17].

2.5.2 Underfitting and overfitting

An important step in preprocessing data is to split the raw data into a training set and a test set. The purpose of this is to train and optimize the machine learning model on the training data and further for it to generalize well to new data using the test data [17].
Suppose the machine learning model is training on the training set for too long. In that case, the model might be able to learn all the details in the data, subsequently unable to predict unseen data as the learning is not generalized. The model can be evaluated during the training using a validation set to prevent this. The validation set will then indicate how well the training data is generalized. When the model is done training, the model performance can be evaluated using the test set [17].

If a machine learning model is optimized on the training data and the generalization is ignored, the machine learning model may suffer from *overfitting* seen in Figure 2.6b. Here, the machine learning model has learned all the details in the training data and is not generalized to perform well on the test data. In contrast to overfitting, *underfitting* results from when the machine learning model has not learned enough to predict unseen data. Figure 2.6a shows an example of underfitted data. Figure 2.6c shows an example of well-fitted data.



(a) Underfitting (b) Overfitting (c) Well fitting

Figure 2.6: Figures illustrate underfitting, overfitting, and well-fitting models.

Classic validation

Two common techniques are often used to evaluate the model to avoid overfitting and generalizing the machine learning model. *Holdout cross-validation* and *k-fold cross-validation*. Holdout cross-validation splits the raw data into a training set and a test set and furthers the training set into a training set and a validation set. The validation set is then used for model selection after hyperparameter tuning. The concept of hyperparameter tuning is to find the best set of parameters in a machine learning model and can be done through grid-search, where a set of parameters is tested. The model with the best performance on the validation set is chosen. Then the model performance is evaluated on the test set. The procedure is seen in Figure 2.7. This technique is sensitive to imbalanced data and how the data splits [28].

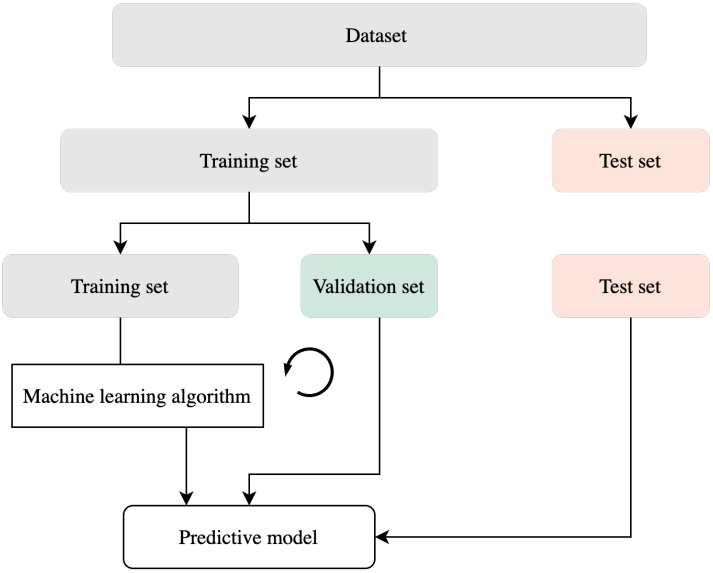


Figure 2.7: Illustration of holdout cross-validation. The rounded arrow illustrates the tuning of parameters where the validation set is used to select best model, the predictive model. The test set is held outside until the final evaluation.

K-fold cross-validation is a more robust technique that uses a similar approach as the holdout method, but the method is repeated k times. The training set is randomly split into k number of folds where $k-1$ folds are used as the training set and the last set as a validation set. This is done k times. By evaluating the model performance using k-fold cross-validation, the problems with imbalanced data and data splitting are addressed and are therefore said to be a more reliable technique [28]. Figure 2.8 illustrates a 4-fold cross-validation and shows how the training data is divided into folds and repeated four times. An estimate of the performance metric is computed as the mean of the estimations from the iterations. Given the 4-fold cross-validation, the estimation would be $E = \frac{1}{4}\sum_{i=1}^{4} E_i$.

Figure 2.8: Illustration of a 4-fold cross-validation. The green boxes represent the test folds, whereas the gray boxes are the training folds. Illustration inspired by [17].

Time series cross-validation

Cross-validation in time series differs from the cross-validation methods presented above. Not only do the classical methods assume the data to be independent and identically distributed, but the random splitting would result in wrongful autocorrelation between training and test sets [28]. Time series data can use a variant of the k-fold cross-validation where the difference is that given the k fold as the training set, the test set must be $k + 1$ fold. In other words, the training set must contain observations prior to the test set in order to obtain the temporal order. After the first fold, each k-fold is added to the first fold and is demonstrated in Figure 2.9. Like the classic k-fold, time series k-fold estimates the average performance estimation.

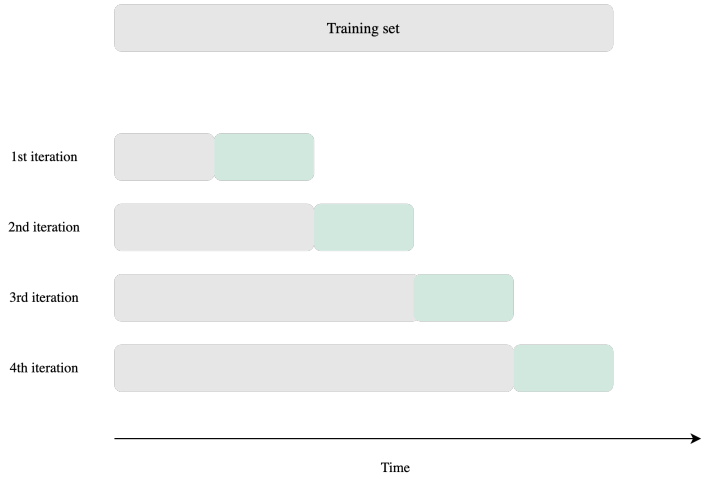


Figure 2.9: Illustration of cross-validation of time series. Since the temporal order must be considered, cross-validation of time series data must be performed on future data. Illustration inspired by [17].

2.5.3 Regression

Regression analysis is a subfield of supervised learning where the goal is to predict target variables on a continuous scale rather than classifying targets. Regression models aim to find a relationship between variables to predict the target. In the simplest form, a univariate regression model can be defined as in Equation 2.18 where y is the target variable, x is the explanatory variable, and w_0 and w_1 are the intercept and regression coefficient of the explanatory variable [17].

$$y = w_0 + w_1 x \tag{2.18}$$

When having multiple explanatory variables, Equation 2.18 can be extended to Equation 2.19, which defines multiple linear regression.

$$y = w_0 x_0 + w_1 x_1 + ... + w_m x_m = \sum_{i=0}^{m} w_i x_i = w^T x \tag{2.19}$$

The regression models above are linear, but what if the data is non-linear? If the data is non-linear, then decision trees are much more fitted than linear regression.

2.5.4 Decision trees

A popular supervised learning algorithm is decision trees where the goal is to use simple decision rules reasoned from the features to predict the value of a target [29]. The decision tree algorithm

is non-parametric, implying that the number of parameters is not fixed but grows with the size of the training data. Additionally, the algorithm does not need an assumption about the distribution of the data [17].

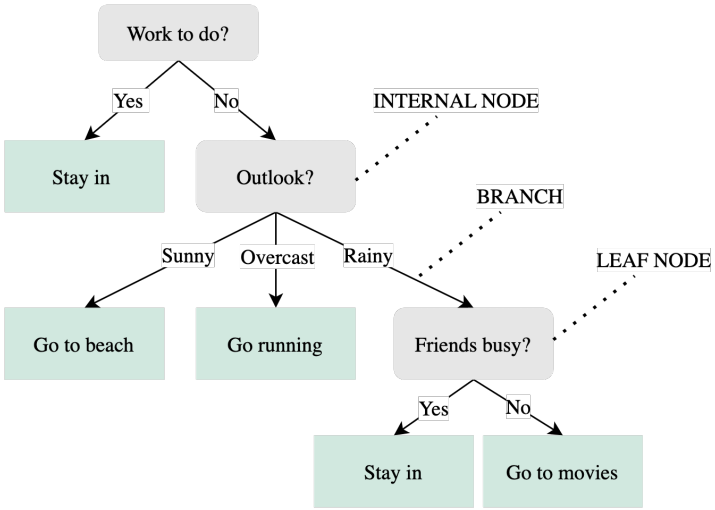


Figure 2.10: Illustration of how decision trees work. The model breaks down data by asking a series of questions.

Figure 2.10 is a simplified example of how the decision tree algorithm works based on asking a series of questions about what activity to do in a day. This is how the algorithm also works with real numbers. The model splits the data based on the feature that has the largest information gain (IG) defined in Equation 2.20 and briefly, it is the difference between the impurity I of the parent node D_p and the sum of the impurities of the child nodes $I(D_j)$. f is the feature to perform the split.

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j) \tag{2.20}$$

An impurity measure for continuous variables is to use the MSE as seen in Equation 2.21. Here, N_t is the number of training examples at node t, and D_t is the training subset at node t. Further, $y^{(i)}$ is the true target value, and $\hat{y}_t$ is the predicted target value. $\hat{y}_t$ is calculated as the sample mean at node t as in Equation 2.22.

$$I(t) = MSE(t) = \frac{1}{N_t} \sum_{t \in D_t} (y^{(i)} - \hat{y}_t^2) \tag{2.21}$$

$$\hat{y}_t = \frac{1}{N_t} \sum_{i \in D_t} y^{(I)} \tag{2.22}$$

Further, this splitting of data is an iterative process that continues until each node is pure, indicating that each node belongs to the same class. To avoid overfitting, one can prune the tree by defining a maximal depth of the decision tree resulting in the tree not being too complex. Decision trees are said to be a greedy algorithm as it does not use global optimizers but rather local ones.

2.5.5 Random forest regression

Random forest is a decision tree-based algorithm that is an ensemble of decision trees. In other words, the random forest algorithm uses several decision trees in order to gain a better

generalization performance than each decision tree alone would do [17]. The random forest algorithm is less prone to outliers and only needs one parameter, the number of trees.
Figure 2.11 illustrates the concept of random forest. After multiple decision trees have predicted values, the final prediction is obtained by either taking the average of the predicted values or major voting, meaning the most frequented value.

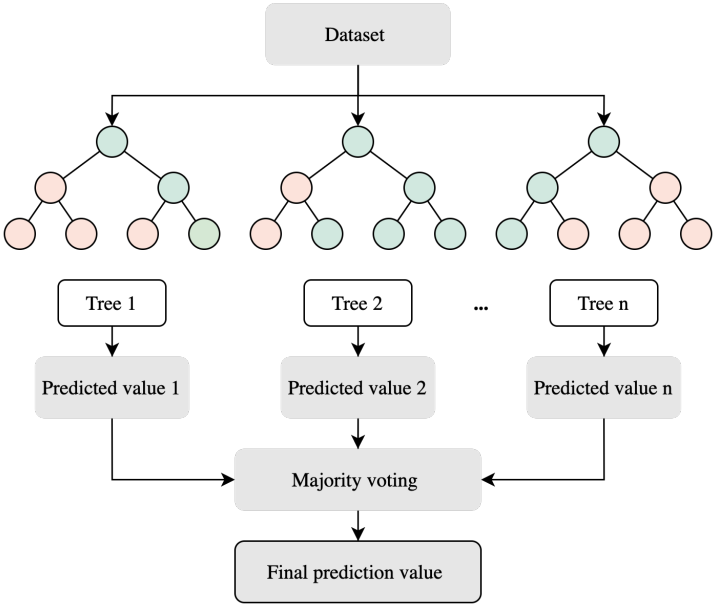


Figure 2.11: Illustration of how random forest model works. The model uses an ensemble of decision trees and aggregates the predictions based on major voting or averaging.

2.5.6 Extreme Gradient Boosting regression

Extreme Gradient boosting, also known as *XGBoost*, is an ensemble method widely used in machine learning.
XGBoost uses boosting, which is the concept of an algorithm that uses a random subset from the training data to minimize bias and variance. Drawing a random subset is done without replacement to train a *weak learner*. A weak learner performs barely better than random guessing. The purpose of using weak learners is to subsequently learn from the errors to improve the performance of an ensemble. The error is the difference between the predicted value and the true value. The next step is to draw a new random subset without replacement but also add the error from the previous weak learner and then train a second weak learner. Then the algorithm finds a subset where the two preceding weak learners disagree and train a third weak learner. Finally, the algorithm combines the three weak learns by major voting [17]. The core idea of gradient boosting is to minimize the errors of the previous weak learner through a gradient-based optimization of the loss function. The loss function indicates how well the model performs by calculating the difference between predicted and actual values. To reduce overfitting, the XGBoost uses a regularisation term. The purpose of a regularisation term is to penalize too complex models, thus preventing overfitting.
The combination of gradient boosting and regularization makes the XGBoost model capable of fast and accurate prediction of large data sets.

2.5.7 Principal Component Analysis

As a result of the ever-increasing amount of data, the ability to compress data has become a critical aspect of the machine learning field. The sheer volume of data generated daily makes it necessary to have efficient storage and analysis mechanisms in place. Data compression is a technique that enables us to reduce the size of data files, making them easier to store and process.

The Principal Component Analysis (PCA) is an unsupervised machine learning method that primarily aims to reduce dimensionality while storing most relevant information in the data. PCA aims to find the main direction of variance in data where $\mathbf{X} = P\mathbf{Y}$. Given a $p \times N$ matrix $\mathbf{X}$ of p variables and N samples, PCA aims to find the orthogonal $p \times p$ matrix that determines a change of variable and variables $y_1, ..., y_p$ that are uncorrelated and arranged in order of decreasing variance [30].

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 \mathbf{u}_2 \dots \mathbf{u}_p \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} \tag{2.23}$$

In Equation 2.23, the components of PCA are symbolized where $\mathbf{X}$ are decomposed into weight coefficients P, also called loadings, and the columns of $\mathbf{Y}$ represent the scores of each observation along the principal components. Scores can be used to explore relationships between observations and reveal patterns or groups in the data. Loadings, however, explore relationships between features and can be used to investigate the influence of each feature on the principal components [31].

Singular Value Decomposition (SVD) is a method for performing PCA and is symbolized in 2.24 where $U\Sigma$ equals to scores and V^T equals to loadings.

$$X = U\Sigma V^T \tag{2.24}$$

Briefly, PCA can be explained as follow. Firstly the covariance matrix Σ is computed and shown as 3×3 matrix in Equation 2.25.

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_2^2 & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_3^2 \end{bmatrix} \tag{2.25}$$

The covariance between two features is calculated as symbolized in Equation 2.26.

$$\sigma_{jk} = \frac{1}{n} \sum_{i=1}^{n} (x_j^{(i)} - \mu_j)(x_k^{(i)} - \mu_k) \tag{2.26}$$

Moreover, the covariance matrix is decomposed into eigenvectors and eigenvalues and further ranked by the highest corresponding eigenvectors.

2.5.8 Metrics

Pearson's r

When exploring the data, looking into the relationship between variables and how they are correlated is essential. This can be done with the Pearson product-moment correlation coefficient, also known as Pearson's r, defined in Equation 2.27. The correlation coefficient r measures the linear dependence between the variables pairwise and are stored in a square correlation matrix.

$$r = \frac{\sum_{i=1}^{n}[(x^{(i)} - \mu_x)(y^{(i)} - \mu_y)]}{\sqrt{\sum_{i=1}^{n}(x^{(i)} - \mu_x)^2}\sqrt{\sum_{i=1}^{n}(y^{(i)} - \mu_y)^2}} = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \tag{2.27}$$

Equation 2.27 show how Pearson's correlation coefficient is calculated between to features x and y. μ is the mean of the corresponding feature whereas σ_{xy} is covariance between x and y. Further, σ_x and σ_y are the corresponding standard deviations of the features.
The correlation coefficient ranges from -1 to 1 where $r = 1$ indicates perfect positive correlation, $r = 0$ means no correlation, whereas $r = -1$ results in a negative correlation.

Mean squared error

Mean squared error (MSE) is valuable when comparing several regression models. Equation 2.28 shows how the MSE is calculated by taking the average sum of the squared error (SSE).

$$MSE = \frac{1}{n}\sum_{n}^{i=1}(y^{(i)} - \hat{y}^{(i)})^2 \tag{2.28}$$

Furthermore, MSE can be extended to root mean squared error (RMSE) seen in Equation 2.29. RMSE measures, similar to MSE, the predicted errors and can be said to measure the variation in the distribution [32]. Subsequently, the lower the RMSE value is, the better the model is.

$$RMSE = \sqrt{\frac{1}{n}\sum_{n}^{i=1}(y^{(i)} - \hat{y}^{(i)})^2} \tag{2.29}$$

Coefficient of determination

The mean square error is prone to misinterpretation and depends on the data set and feature scaling. Thus, the coefficient of determination (R^2) may be a better choice [17]. R^2 can be considered a standardized MSE as it calculated the fraction of response variance in the machine learning model and is derived in Equation 2.30. The total sum of squares (SST) calculates the variance of the target $Var(y)$ while the sum of squared errors (SSE) calculates the sum of the squared difference between the predicted value and target value.

$$\begin{aligned} R^2 &= 1 - \frac{SSE}{SST} \\ &= \frac{\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2}{\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \mu_y)^2} \\ &= 1 - \frac{MSE}{Var(y)} \end{aligned} \tag{2.30}$$

When evaluating training data, a $R^2 = 1$ is desired as this indicates a $MSE = 0$ and that the machine learning model fits perfectly to the data [17].

2.5.9 Soft sensors

Soft sensors are data-driven models that can predict challenging hard-to-measure features in a process line based on easy-to-measure features. Here, the models are trained on historical data collected from the process line. Further, soft sensors may be deployed and use the input data stream to predict the upcoming process [33]. The predicted values may enhance process control as the predicted values and actual values can be compared to detect deviations or undesired patterns.

Chapter 3

Materials

This chapter gives an insight into the process at Bioco and the data gathered at the process line.

3.1 Bioco

Bioco is located next to Norturas slaughterhouse, and the rest raw material from Noturas production is transferred directly to Bioco through pipes. From here, the rest raw materials are ground into smaller pieces looking more like a farce, and mixed with water in a mixer. Here, the secret of the process is introduced, the enzymes. The enzymes in this process split the protein molecules into even smaller pieces. Further, the rest raw materials are transported through a massive piping system resulting in high-quality products in the form of oils and protein/mineral flour [34]. This piping system, in contrast to tanks, is unique at Bioco and gives them several benefits but also some obstacles. Pipes are prone to clogging if the materials have high viscosity, consequently leading to the pipes needing cleaning and hence flushed with water. This results in stops in production. From this, the mass is transported in long pipes before further processing, where the materials are made into the end product.

The process at Bioco is continuous rather than batch-wise process. This is an important aspect of the process as it is more stable in terms of the quality of the end product and also easy to change parameters such as temperatures. A simplified overview of the process can be seen in Figure 3.1

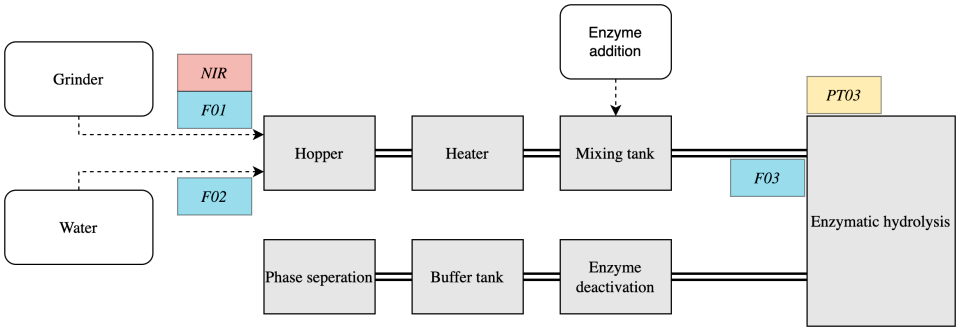


Figure 3.1: Simplified overview of the process at Bioco. The red box illustrates the placement of the near-infrared sensor, whereas the blue show the sensors of different flow rates. The yellow box depicts the pressure sensor at the beginning of the hydrolysis pipe.

3.2 Description of the data

The data from Bioco may reveal key information about the process and consequently the end product. Therefore, the presentation of the data may be of limited information due to a non-disclosure agreement.

The process data were collected during the autumn of 2022 using a SIMATIC WinCC Runtime Advanced (Siemens, Munich, Germany). The data were then extracted from the database and stored as CSV files.
The near-infrared data were collected using a Perten DA 7440 (PerkinElmer, Waltman, MA, USA) near-infrared inline sensor. The sensor was utilized to measure the second overtone region (950-1600 nm) of the rest raw materials stream with a 5 nm resolution. The sensor was positioned 25 cm from the rest raw materials stream, which was discharged from the grinder. Furthermore, a spectrum was saved every 0.5 seconds to gain valuable insight.

For more accessible storage and analysis, the files were divided into representative weeks for both the NIR data and the process data. Table 3.1 presents the file names and their shape. It is worth noticing that the NIR data are missing from week 46.
The fat data are predicted from the near-infrared spectra and contain the fat content in each sample.

Table 3.1: The table shows the data files, their content, the shape of the files, and the sampling frequency in sample(s) per minute.

Name	Content	Shape (n x p)	Sample frequency (*per minute*)
BC22NirWeek44.csv	NIR spectra from week 44	592 432 x 142	120
BC22NirWeek45.csv	NIR spectra from week 45	646 105 x 142	120
BC22NirWeek47.csv	NIR spectra from week 47	752 621 x 142	120
BC22NirWeek48.csv	NIR spectra from week 48	665 286 x 142	120
BC22NirWeek49.csv	NIR spectra from week 49	234 002 x 142	120
BC22NirWeek50.csv	NIR spectra from week 50	688 573 x 142	120
BC22fatW44.csv	Fat from NIR spectra from week 44	592 432 x 2	120
BC22fatW45.csv	Fat from NIR spectra from week 45	646 105 x 2	120
BC22fatW47.csv	Fat from NIR spectra from week 47	752 621 x 2	120
BC22fatW48.csv	Fat from NIR spectra from week 48	665 286 x 2	120
BC22fatW49.csv	Fat from NIR spectra from week 49	234 002 x 2	120
BC22fatWk50.csv	Fat from NIR spectra from week 50	688 573 x 2	120
ProcessDataWeek44.csv	Process data from week 44	8640 x 49	1
ProcessDataWeek45.csv	Process data from week 45	8640 x 49	1
ProcessDataWeek46.csv	Process data from week 46	8640 x 49	1
ProcessDataWeek47.csv	Process data from week 47	8640 x 49	1
ProcessDataWeek48.csv	Process data from week 48	8640 x 45	1
ProcessDataWeek49.csv	Process data from week 49	8640 x 45	1
ProcessDataWeek50.csv	Process data from week 50	8640 x 45	1

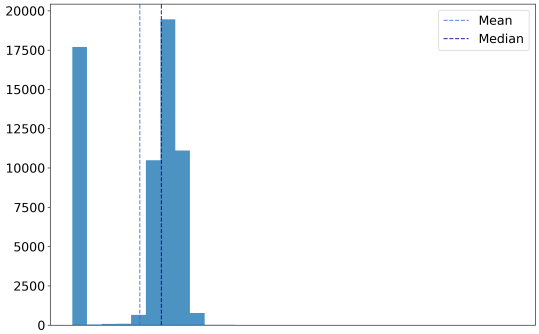
The process data contained many features, and not all of them are of interest in this thesis. Table 3.2 shows selected features in the process data and a simple explanation of the content. The wavelengths of the near-infrared spectra are also included.

Table 3.2: Table showing the features in the raw data set with a short description of what they are.

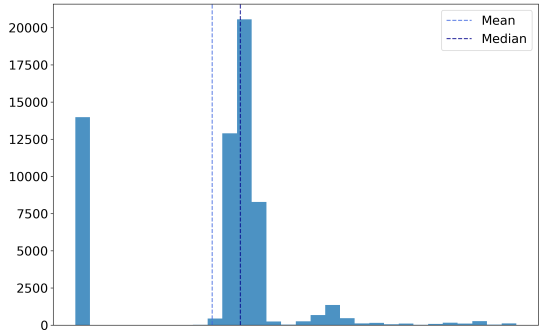
feature	Content
Time	Time stamps
F01	Flow rate rest raw materials
F02	Flow rate water
F03	Flow rate after rest raw materials and water are mixed in the grinder
PT03	Pressure at the beginning of hydrolysis
Fat	Fat content predicted from near-infrared data
950 - 1650	Wavelengths in the near-infrared data where each column represents a wavelength

3.3 Data exploration

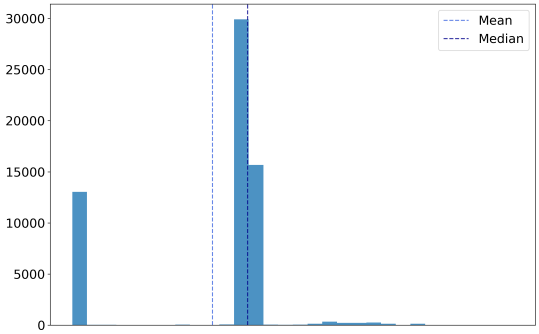
A vital step when working with data is to investigate the raw data to get an overview and gain valuable insight. In Figure 3.2, the histograms of the process features are plotted and show the frequency of values. It is observable that all the plots show a sizeable amount of data points at the lower range of the x-axes which also deviates from the rest of the histograms. This indicates that there are some outliers that must be investigated. In addition, the means and medians of each histogram reveal that there are values that heavily influence the mean towards the lower range of the x-axis. The medians, however, are more centered around the highest counted values. Likewise, Figure 3.2b and Figure 3.2c show values at the upper range of the x-axes that should be explored further to decide whether they are unwanted or of interest. The same tendency can be seen in Figure 3.2d of the pressure feature where there are some higher values at the tail of the distribution. Feature F01 in Figure 3.2a, however, only seems affected by the lower values.



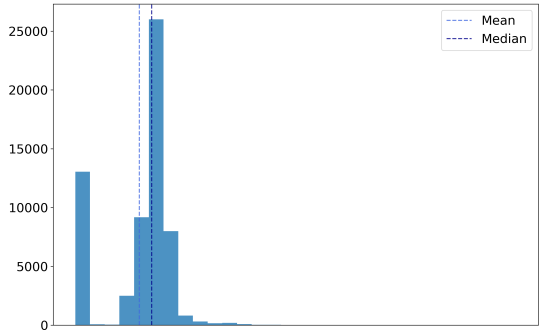
(a) Distribution of feature F01.



(b) Distribution of feature F02.



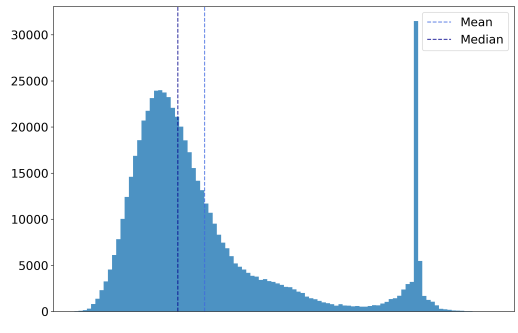
(c) Distribution of feature F03.



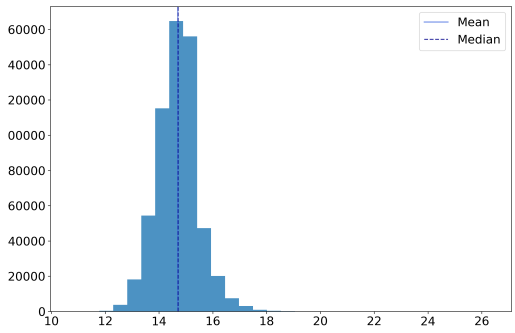
(d) Distribution of feature PT03.

Figure 3.2: Distribution of features from the process data. The distributions present multiple values deviating and must be further explored.

When exploring the near-infrared data, it has been chosen one specific wavelength to further consider. In Figure 3.3a, the distribution at wavelength 1215 nm is plotted, and the histogram exposes a large amount of data points deviating from the rest at the upper x-axis. Also here the mean is influenced by these higher values and is shifted more towards the right. On the other hand, the feature Fat in Figure 3.3b seems to have a symmetrical distribution as the mean and median are approximately indistinguishable.



(a) Distribution of wavelength at 1215 nm in week 44.

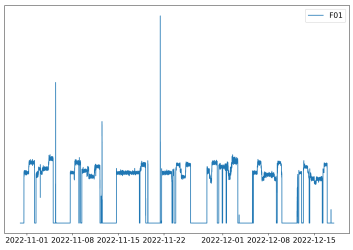


(b) Distribution of feature Fat, the predicted fat from NIR spectra.

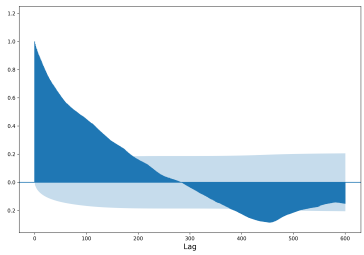
Figure 3.3: Distribution of features. The distribution of the wavelength indicates unwanted data.

3.3.1 Autocorrelation

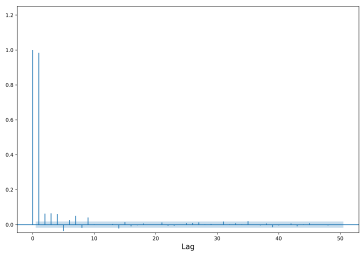
Autocorrelation is, as mentioned in Chapter 2, the correlation between a time series and a lagged version of the time series itself. Both Figure 3.4 and Figure 3.5 present a deeper insight into the process data from a time perspective. Given the data being of a relatively high time resolution, the data have been downsampled to one sample every five minutes to make it easier to visualize and interpret. The size of the downsampling rate was chosen based on comparison to the original data set. Figure 3.4a presents the downsampled feature F01, which is the flow rates of the rest raw materials, whereas Figure 3.5a presents the downsampled feature F02, which is the flow rates of the water. The figures demonstrate the vast variance in the time series. In both Figure 3.4b and 3.5b, the ACF plots indicate a high correlation between the time points close in time. Furthermore, the partial autocorrelation plots in Figure 3.4c and 3.5c emphasize the correlation of values close in time. Additionally, the PACF plots indicate significant correlations within the first hour as there are spikes above the 95% confidence intervals.



(a) Time series of F01.

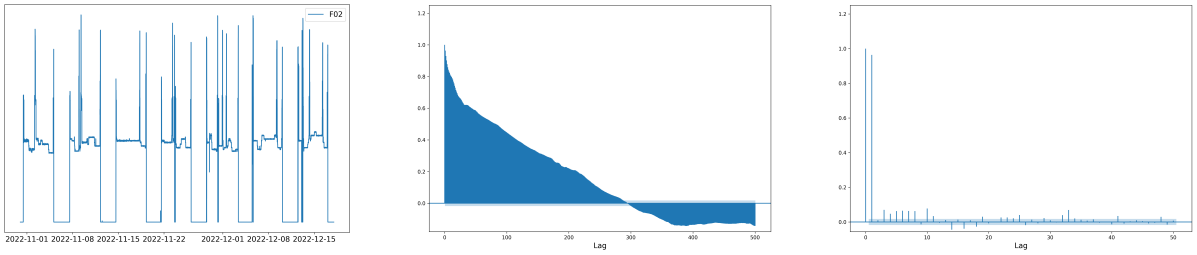


(b) ACF plot of F01.



(c) PACF plot of F01.

Figure 3.4: Autocorrelation and partial autocorrelation plots of the flow rate in feature F01. The feature has been downsampled to one sample per 5 minutes.



(a) Time series of feature F02. (b) ACF plot of F02. (c) PACF plot of F02.

Figure 3.5: Autocorrelation and partial autocorrelation plots of the flow rate in feature F02. The feature has been downsampled to one sample per 5 minutes.

3.3.2 Bad spectra

The near-infrared data were collected in relatively high time resolution, resulting in the large data size. For computational and resource efficiency, the next descriptions are based on data collected in the first week, week 44. The exploratory analysis is quite similar for all the weeks; therefore, the exploration explained is simplified by using only data from week 44. As mentioned, it is important to look into simple statistics, and the histograms earlier indicated that there are values from unwanted data. Hence, both the mean and the median of the wavelengths have been plotted in Figure 3.6. In addition, a ribbon showing the 25% and 75% percentiles of the wavelengths is added. By adding this ribbon, it is possible to visualize where most of the data are and therefore gain an impression of unwanted data. This can be seen already when investigating the mean of the data. Here, it is observable that the mean of earlier wavelengths is in the upper region of the percentiles, implying that there are some spectra with higher values in this region, thus influencing the mean. The median, which mathematically is more robust to outliers, has a pattern more aligned with the center of the ribbon band, seen in Figure 3.6.

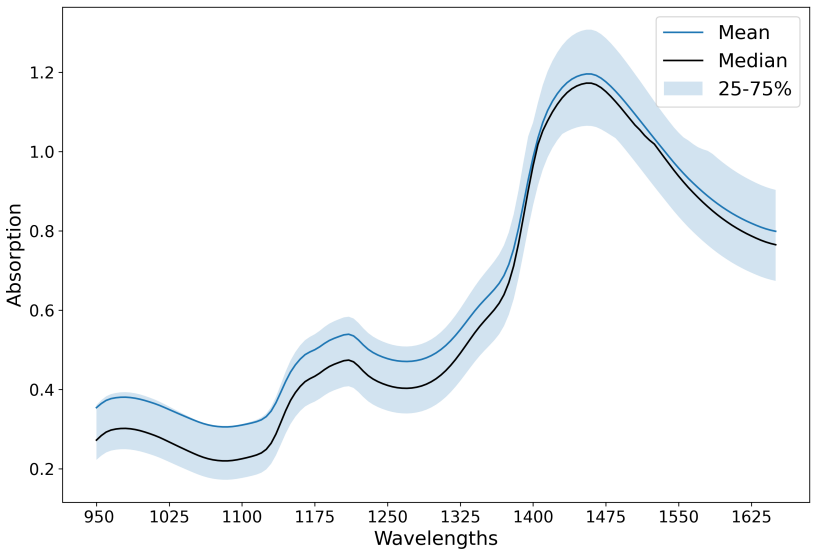
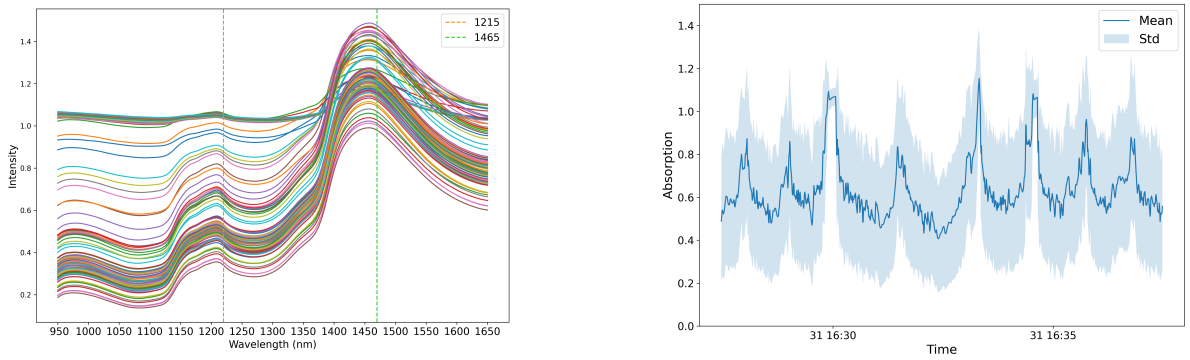


Figure 3.6: The figure presents the mean and median of the original near-infrared data from week 44. The blue ribbon indicates the 25/75% percentiles. The mean seems to be heavily influenced by higher values as the mean is aligned with the upper region of the ribbon.

When exploring the wavelengths further, it is visible how some spectra differ from others. Figure 3.7a show how these spectra follow a different pattern of a more straight line. Thus it can be said that the near-infrared sensor has sampled something other than the rest raw materials. From domain knowledge, it is highly plausible that these spectra are steel.



(a) Spectra plotted.

(b) Spectra versus Time.

Figure 3.7: The figures present the spectra in the spectral domain and time domain to illustrate the appearance of bad spectra. Some spectra seem to follow a different pattern than the other spectra

Furthermore, the fact that the data contain spectra of steel is underlined when plotting the near-infrared data against the timeline as in Figure 3.7b. A seasonal pattern can be identified. This is due to how the grinder discharges the rest raw materials in pulses. Thus it can be said that the near-infrared data contain bad spectra, and unwanted data, that need to be handled.

3.3.3 Time range

When exploring time series data, it is important to notice the time steps. The time steps can be *regular* or *irregular*. When time steps are regular, the distance between each point is equal, in other words, equidistant. This is not the case for the data in this analysis. In Figure 3.8, the y-axis is the timeline, and the x-axis is the different features in the process data. Here, the data have been resampled to have one sample each minute. Thus Figure 3.8 reveals irregular time steps by illustrating missing values at time points not included in the original data sets. These missing time points can be identified as weekends.

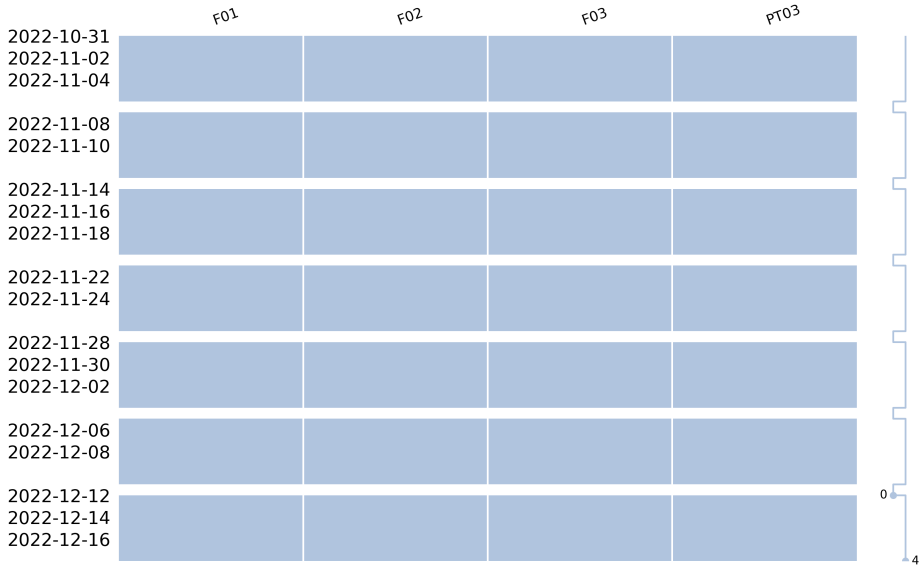


Figure 3.8: Figure illustrates regular time steps of the process data. The white area depicts where there are missing values, and these areas can be identified as the weekends.

Chapter 4

Method

In this chapter, the different methods are presented. The methods used in this thesis were not developed through a straightforward process, but rather through an iterative one to explore different techniques and gradually improve the methods.

Firstly, the methods of data filtering and feature engineering will be elaborated. Then smoothing will be presented and finally the machine learning models.

4.1 Data Filtering and Feature Engineering

A time-consuming, but important preprocessing step, is the *feature engineering* part where the task is to better prepare the data for modeling by extracting the most useful information in the data. The process involves tasks like transformations, extraction, and selection. In this thesis, the different data were in need of different feature engineering tasks due to the nature of the data. The next sections will go into further depth on the methods used.

4.1.1 Near-infrared data

The near-infrared data was collected using an online sensor placed right after the rest raw materials were ground into smaller pieces. From the data presented in Chapter 3, especially in Figure 3.7a, one can see that the online sensor measured more than only the rest raw materials as there are several spectra that deviated from the expected pattern. These spectra were removed based on percentiles. Firstly, two wavelengths that measured the lower and upper range of absorption values were chosen and a column was added to the data set which contained the ratio between the values of these two wavelengths. These were wavelengths at 1215 nm and 1465 nm.

$$col_{ratio} = \frac{col_{1215}}{col_{1465}} \tag{4.1}$$

Secondly, the 25% and 75% percentiles were calculated on the ratio column and bad spectra were defined as spectra that were outside these percentiles. If the ratio was outside the percentiles, it was removed from the data set as it indicated that the pattern of the spectrum deviated. The index of bad spectra was stored in order to correct the fat data. This approach was chosen based on domain knowledge and how the pattern of the unwanted spectrum would have a small ratio as the two wavelengths would have values that were not shifted as much as the desired spectra.

Moreover, the fat data contained fat predicted on these bad spectra. To address this, the index of the bad spectra was used to remove the corresponding fat data as the near-infrared data and

fat data had the same index.

The near-infrared data were as mentioned in Chapter 3 of greater dimensions, making it more complex to analyze. To address the curse of dimensionality, the near-infrared data were transformed through Principal Component Analysis (PCA). This is a method for data compression which is explained thoroughly in Chapter 2. In order to retain the same information obtained by the principal components, the data were merged into one data set and further, the principal components were projected onto the weekly data sets. This approach was chosen with regard to the volume of the data and resource limitations. The principal components were calculated using the machine learning package *scikit-learn* and *PCA* function where the only parameter given was the number of principal components, $n\_components = 5$. This function uses Singular Value Decomposition in order to decompress the data [35]. The scores of the five principal components were stored in a DataFrame for further analysis.

In Figure 4.1, the workflow of near-infrared data is illustrated.

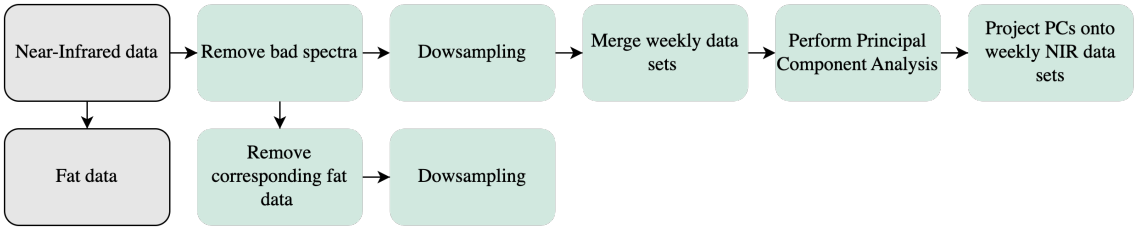


Figure 4.1: Figure that shows the workflow of the near-infrared data and fat data.

4.1.2 Process data

Data Extraction

From domain knowledge at Nofima and Bioco, several feature engineering tasks were needed for the process data. Firstly, the extraction of useful data for further modeling where performed. Here, the process data went from a relatively wide data set with 49 features to a less wide one with only five of the original features, including the timestamps. The extracted features are seen in Table 4.1.

Table 4.1: Table of the extracted feature relevant for further analysis.

Feature	Content
Time	Time stamps
F01	Flow rate rest raw materials
F02	Flow rate water
F03	Flow rate after rest raw materials and water are mixed in the grinder
PT03	Pressure in the hydrolysis pipe

Data Creation

The sensors are placed at different positions in the process thus an important task has been to correct the timeline and *lag* features. This has been done in order to retrieve a more accurate picture of the rest raw materials throughout the process and to better meet the somewhat same rest raw materials at each sensor. As most of the features in Table 4.1 are placed at the beginning of the process, these features were used as a reference, and features appearing later in the process had to be lagged. Consequently, a function was made in order to lag these features. The function required two inputs, which column to be lagged and by how many minutes. Further, the selected column was shifted with the given value and the earlier points were filled with zero as if the sensor had not yet measured the rest raw materials. From domain knowledge about the process, the first pressure sensor in the hydrolysis pipe lagged for 30 minutes. In other words, the data were shifted by 30 time points the time points were replaced with 0. The intention behind filling the time points ahead of the lagged data relies on the idea that the rest raw materials of interest had not yet reached the pressure sensor.

The concept of retrieving a more accurate picture of the rest raw materials throughout the process resulted in one original feature being replaced by a new feature. Feature *F03* measured the rest raw materials after it had been mixed with water in a mixer tank. However, *F03* would not reflect the correct rest raw materials due to the mixer. Consequently, a new feature, SUM_F01_F02 was created that aggregated the flow rate of water and the flow rate of the rest raw materials, which is symbolized in Equation 4.2.

$$col_{Sum} = col_{F01} + col_{F02} \tag{4.2}$$

Similarly, a new feature, F01_rel, depicting the relative flow rate of the rest raw materials of the aggregated flow rate of water and the rest raw materials were created and are symbolized in Equation 4.3. The rest raw materials are of greater interest for further analysis and it is therefore gainful to create a feature with a relative flow of the rest raw materials.

$$col_{F01\_rel} = \frac{col_{F01}}{col_{F01} + col_{F02}} \tag{4.3}$$

Finally, the principal components and fat from the near-infrared data were merged together with the process data to create the final data set for further analysis. Figure 4.2 is a heat map showing the correlations between features after feature engineering steps. Some very high correlations are observable and presented as darker fields. High correlation implies redundant information, and based on these correlations, features F01 and F02 were removed for further analysis.

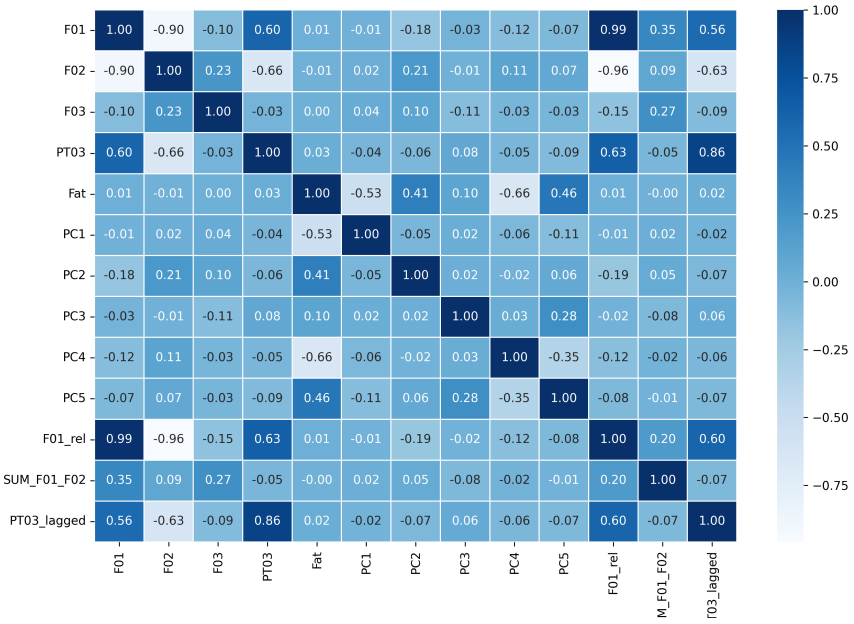


Figure 4.2: Figure is illustrating the heat map of the correlation matrix of all features after cleaning.

Data Filtering

The retrieved process data contained both data from when the process was running and other aspects of the process, such as cleaning the pipes. The higher and lower values seen in the histogram plots in Chapter 3 are likely to originate from the cleaning process. In other words, the data set contained a considerable amount of irrelevant data thus data filtering was needed as it was of no interest for further analysis.

Firstly, a method to remove irrelevant data was using a threshold value for a specific feature. The team at Bioco informed that a given set of values for the water flow rate, F02, were flushing of pipes and not from the actual process. Therefore two threshold values, y_1 and y_2, were used to remove data gathered during the cleaning of pipes. In Figure 4.3, the time series of feature F02 are plotted with both threshold lines.

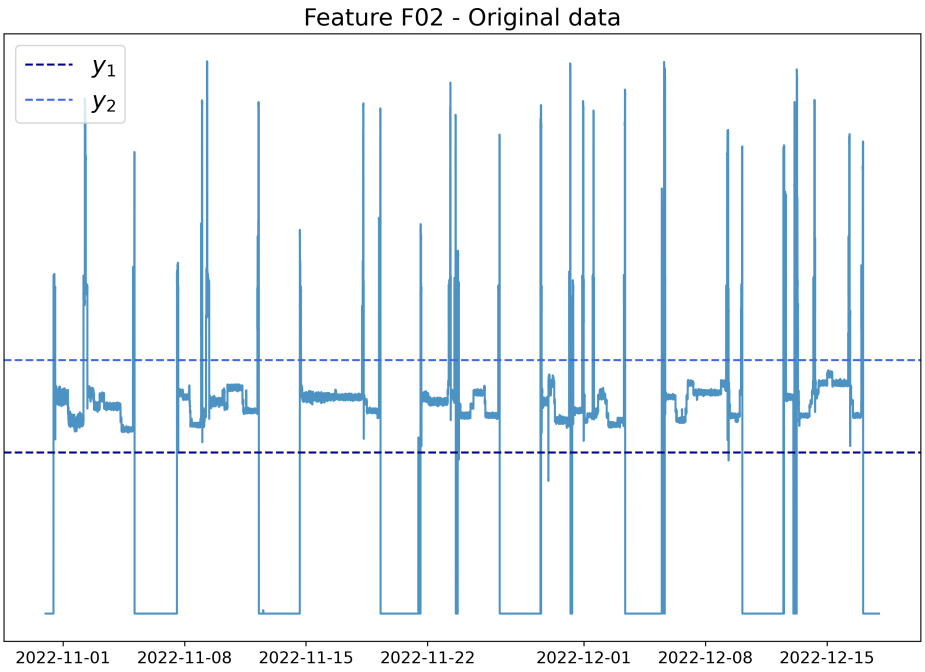
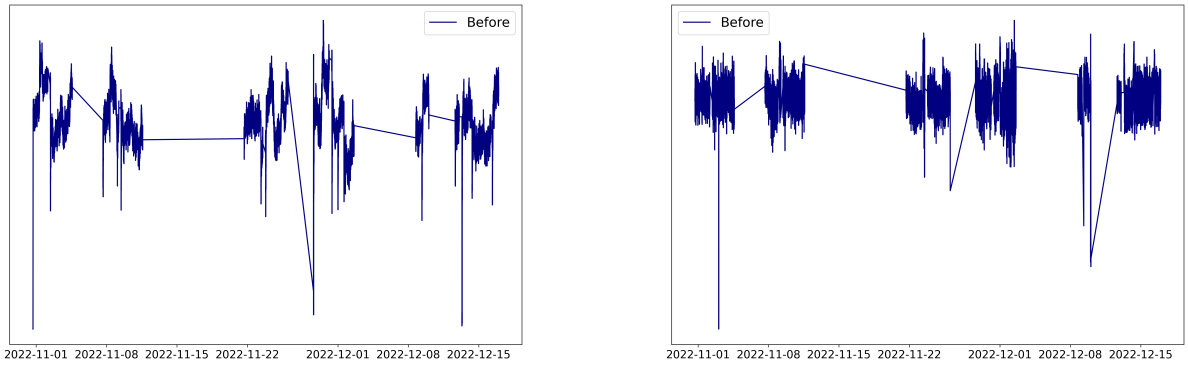


Figure 4.3: Illustration of the line plot of feature F02, the water flow rate, and threshold lines.

This removed the majority of unwanted process data, but there were still higher peaks and lower drops from when the process was either starting or shutting down. Therefore, data were removed by comparing values next to each other. If the difference between two consecutive values was higher than a given threshold, the value was stored in a new column. Otherwise, the value was set to zero in the new column. This gave rise to a new column of values that indicated unwanted data. Lastly, values of the difference outside the 25/75% percentiles were removed. This addressed the remaining data that were not from the process running.

However, the data still had left unwanted data in other features as seen in Figure 4.4 where Figure 4.4a is the lagged version of the pressure sensor and Figure 4.4b displays the sum of the flow rate of water and rest raw materials. Both figures depict multiple outliers that heavily influenced the machine learning models and the metrics. Thus it was decided that these outliers were of no interest and therefore removed using a simple Z-score to eliminate these spikes.



(a) Feature PT03_lagged

(b) Feature SUM_F01_F02

Figure 4.4: The figures present outliers in created features.

Data Split

In machine learning splitting of data is important, and the train set is used for model building, whereas the test set is used for model evaluation.

In Figure 4.5, the final data set is represented with one sample per minute to retain and investigate a regular timeline. Here, it can be observed multiple missing values. The figure identifies mainly two things. Firstly, there is no data in the third week, week 46, and the vast majority of the sixth week, week 49, is without data. Secondly, the time steps are irregular within the weeks as well.

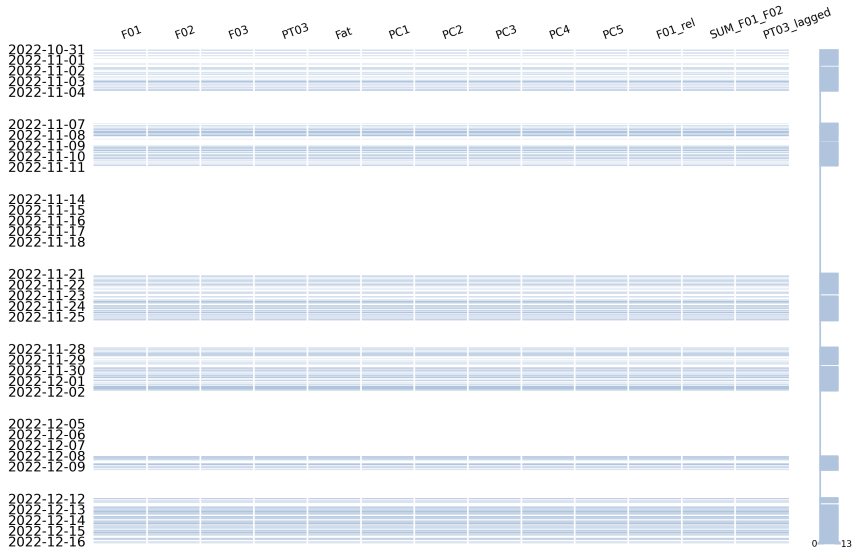


Figure 4.5: Figure illustrates regular time steps, and white areas depict where there are missing values.

The sensor data of interest to predict was the pressure data. In other words, PT03_lagged was defined as the output variable (y) whereas the other features were defined as the input variable (X).

Further, the approach of splitting the time series data into a training set and test set differ from the typical train/test split within the machine learning field. As the data sets were divided into their representative weeks, it was reasonable to use the weeks as the base. Figure 4.6 represents how the preprocessed data set was split. All weeks, except the last one, were defined as training data. Subsequently, the last week was set to be test data. Not only was the information in the sequence kept in contrast to random splitting, but the ratio of training and test data was approximately 70 : 30 which is a desired ratio.

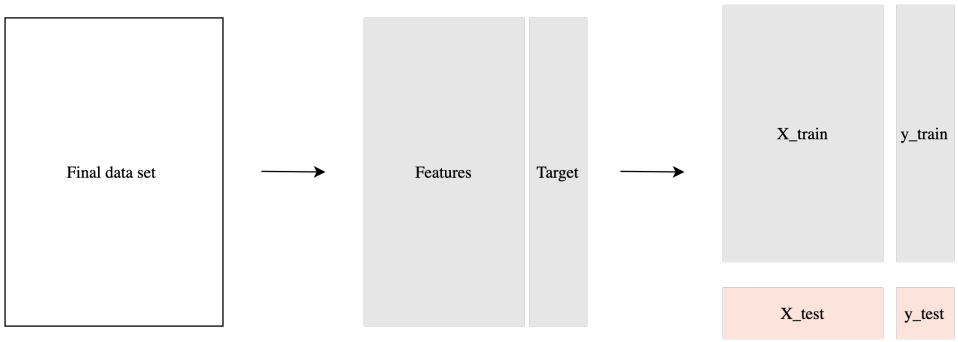


Figure 4.6: Illustration of how the preprocessed data set was split.

A more detailed representation of the time steps is seen in Figure 4.7 which is the X_test resampled to one sample per minute in order to detect where there are no data. From the representation, it can be clearly observed that the time stamps are irregular, and there are three greater gaps where the process has not been running during week 50.

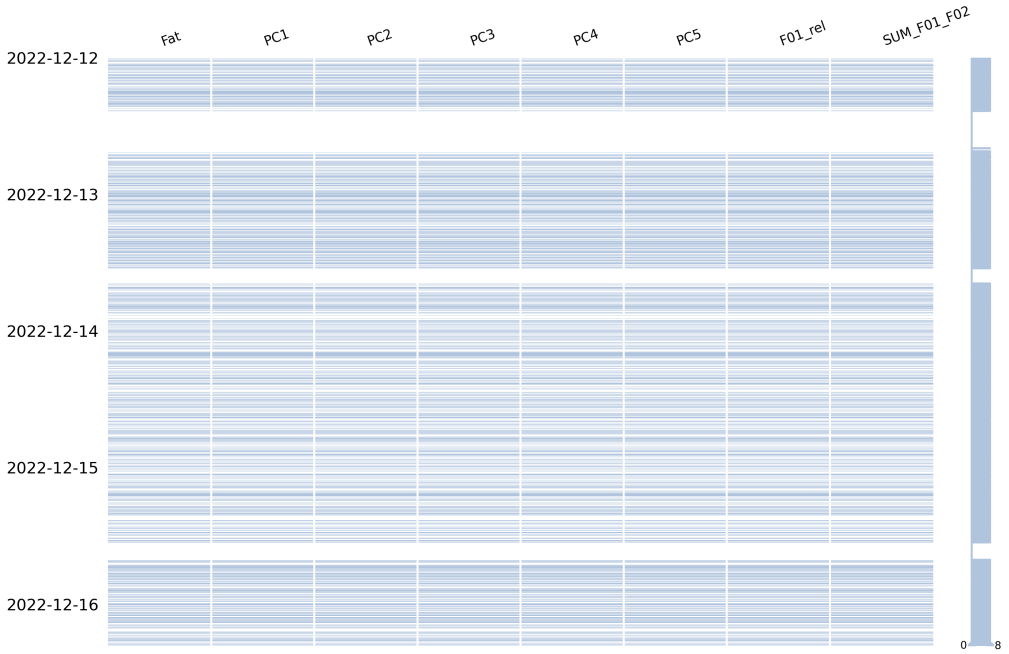


Figure 4.7: Figure represents regular time steps of features in the test data. The white area depicts where there are missing values.

4.2 Smoothing

Moving from feature engineering, a more classical time series prepossessing method was performed. Sensor data are prone to noise and the rest raw materials are also of great variance. Smoothing is a well-known approach when dealing with noisy data and the aim was to reveal underlying trends and patterns.

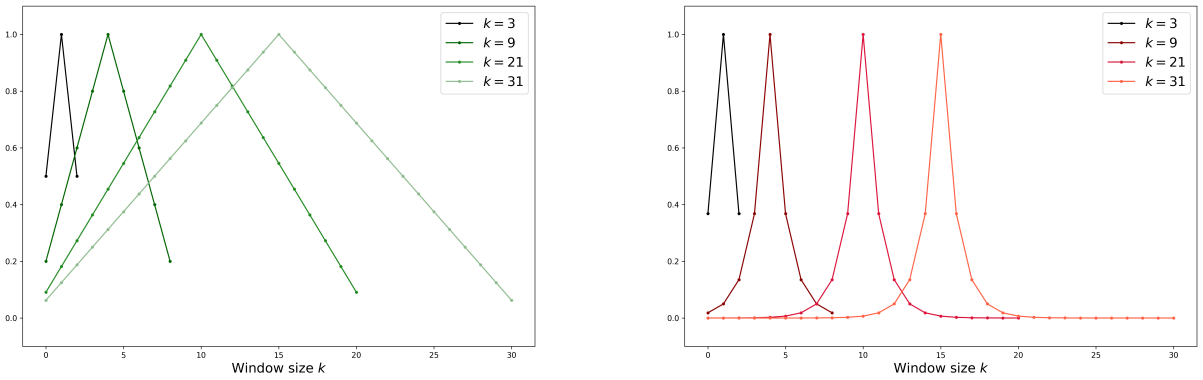
In light of smoothing being a preprocessing step before machine learning modeling, a pipeline was developed to incorporate a smoothing function before model building. The smoothing function was within a transformer class that took two parameters as input, smoothing method, and window size. The transformer applied the smoothing parameters to the input data.

Four types of smoothing methods were performed on the time series data, *moving average*, *moving median*, *weighted moving average*, and *exponential smoothing*. The methods used the built-in *rolling* function of the pandas' library and computed the mean or median in that specific rolling window. The two latter methods were implemented through a parameter of the rolling function.

Different inputs of the window size were included to investigate the effect. Smoothing with window size $k = 1$ implied no smoothing due to the fact that the rolling window smoothed over only a single time point thus returning the original value. Following that, the window size increased by two until reaching $k = 31$ which was set at the upper limit. $k = 31$ implies smoothing over a time range at around 30 minutes, given the window is within a single time segment. In view of domain knowledge, a greater window size would be irrelevant regarding the process itself. In brief, each smoothing method was performed with 16 different window sizes, ranging from $k = 1$ to $k = 31$.

As explained in Chapter 2, the smoothing methods exclude values at the beginning and the end of the time series with a size equal to the smoothing window resulting in missing values of the size equal to the window size. This was addressed by linear interpolating the first, or last, value that was not NaN resulting in the shape of the time series being constant.

Figure 4.8 show how the weights are allocated at different window sizes k. Recall that sum of the weights should sum to one as in $\sum_{j=-k}^{k} a_j = 1$.



(a) Linear smoothing

(b) Exponential smoothing

Figure 4.8: Illustration of how the weights are decaying at different window sizes.

4.3 Random Forest Regression

The smoothed data was modeled using the RandomForestRegressor from the scikit-learn library. There are numerous reasons for the choice of using RandomForestRegressor. Firstly, the machine learning algorithm is non-parametric, meaning it does not make assumptions about the underlying distribution. Therefore, the data did not need to be scaled beforehand. Secondly, it is possible to extract feature importance from the algorithm. The importance of a feature in RandomForestREgressor is estimated using Gini importance. In brief, Gini importance is estimated using the Gini impurity, a criterion to minimize the error of predicted value, and is further elaborated in Chapter 2. In other words, RandomForestRegression gives the ability to investigate the importance of each feature used to learn from the data. The feature importance was used to understand which features the model learned from and then to go back and further investigate the most important ones. Lastly, the fact that the RandomForestRegressor is an ensemble method makes it desired when handling time series with noise regarding overfitting and can also handle non-linearity in the data.

The model was trained and tested on several combinations of features from the process data, near-infrared data, and fat data. The final models used the preprocessed data set presented in Table 4.2.

Table 4.2: Table showing the preprocessed variables used in modeling.

Feature	Content
Time	Time stamps
F01_rel	Relative flow rate of rest raw materials
SUM_F01_F02	Aggregated flow rate of water and rest raw materials
PT03_lagged	Lagged version of the pressure
PC1	Principal component one
PC2	Principal component two
PC3	Principal component three
PC4	Principal component four
PC5	Principal component five
Fat	Fat content predicted from near-infrared spectra

As the perspective of this thesis is toward preprocessing, no hyperparameter tuning was performed on this model.
The models were evaluated using R^2, and root mean squared error (RMSE).
Furthermore, a baseline model was developed to compare with. The model selected was the *average method* presented in Chapter 2. This method takes the average of the historical data and sets all future predictions equal to this average value.

4.4 XGBoost Regression

As a comparison to the RandomForestRegressor, the ensemble method XGBRegressor was modeled. The theory behind XGBoost Regression can be found in Chapter 2, Section 2.5.6. Equal to the RandomForestRegressor, the XGBoostRegressor also uses a collection of decision trees in order to predict new values. Yet, the XGBoostRegressor differs in the sense that

it is based on gradient boosting to improve the model. Furthermore, the benefits of using XGBoostRegressor are very much similar to the ones presented about RandomForestRegressor. Equal to the RandomForestRegressor, the default parameters were used, and no tuning of the parameters where performed.

4.5 Software

All analyses were performed in Google Colab with Python, version 3.9.16.

Additionally, the artificial intelligence chatbot from OpenAI, *chatGPT*, was used to solve errors in code quickly and generate example data due to its fast and specific response to defined problems. It is vital to stress that such new and advanced technology outputs have been met with critical reflections and healthy skepticism.

Chapter 5

Results

Firstly, the data filtering and feature engineering results will be presented. Secondly, a selection of the results of the four smoothing methods on the target data and the results of the machine learning modeling are shown. Lastly, a comparison of the two machine learning models will be put forward.

The nature of the time series in this thesis makes the result too complex to visualize in a global view. Therefore, the result will mainly be presented in a detailed view.

5.1 Data Filtering and Feature Engineering

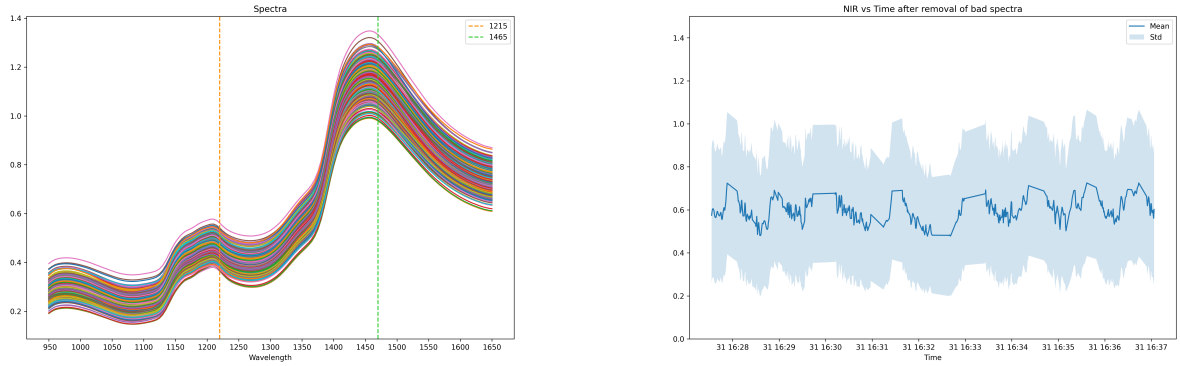
5.1.1 Near-infrared data

The near-infrared data were collected at a high sample rate and contained unwanted data as the sensor had measured not only the rest raw materials but also other materials. As elaborated in Chapter 4, the unwanted data were detected and removed based on percentiles. The data were then downsampled to one sample per minute and subsequently compressed through Principal Component Analysis. In Table 5.1, the different shapes are listed, and the filtering of bad spectra narrowed the data set considerably. However, the downsampling of the data minimized the data set a lot, thus making it less demanding in terms of resources to work with further.

Table 5.1: The table presents the sample sizes before and after filtering bad spectra in the near-infrared data and the data downsampled to one sample per minute.

Data set	Original shape	Shape after filtering	Shape after downsampling
Week 44	592 432	296 216	5 296
Week 45	646 105	323 053	5 230
Week 47	752 621	376 311	5 791
Week 48	665 286	332 642	5 260
Week 49	234 002	117 000	1 684
Week 50	688 573	344 287	5 540
Total	**3 579 019**	**1 789 510**	**28 801**

In Figure 5.1a, the spectra are plotted after filtering unrelated spectra. Here, the spectra followed the same expected pattern that depicted the physical properties of the rest raw materials. Additionally, when the near-infrared data from a time perspective were investigated, it was observed that the time series pattern had changed. The seasonal peaks were flattened due to the filtering of unwanted spectra that did answer for those peaks. This can be seen in Figure 5.1b.



(a) Spectra after filtering

(b) Absorption versus time.

Figure 5.1: Figures presenting the spectra after data filtering. The filtering of bad spectra yielded cleaner spectra, and the peaks were reduced in the time domain.

The Principal Component Analysis was computed on the cleaned near-infrared data, and then the principal components were projected onto the downsampled data. Figure 5.2 shows how much of the explained cumulative variance was captured by the five first principal components. It was observable that the first principal component answered for almost 100% of the explained variance, subsequently leading to a substantial drop in the cumulative variance from the first principal component to the second. The high value of explained variance indicated that the principal components could retain most of the information in the data even after compressing the data set.

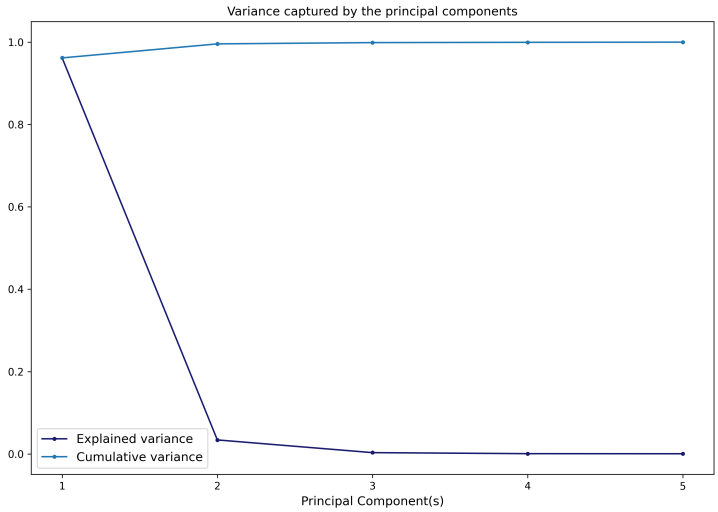
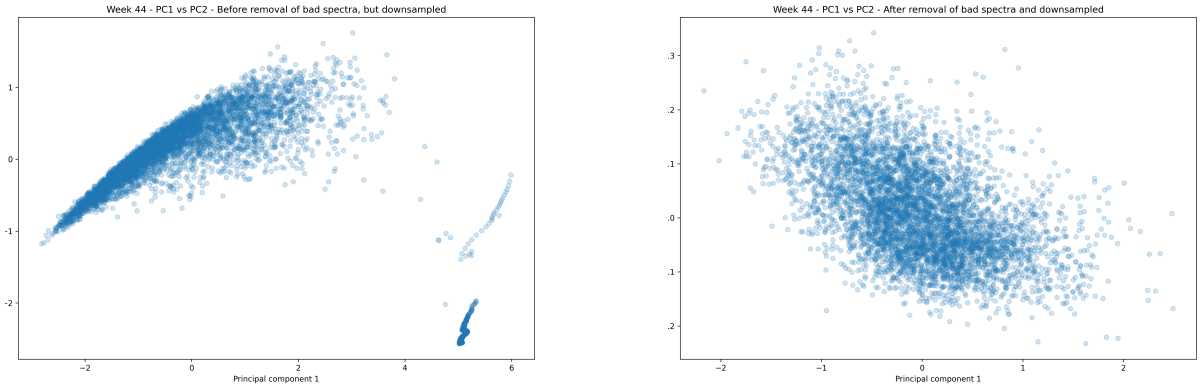


Figure 5.2: The figure presents the explained variance and cumulative variance captured by the PCA.

Moreover, the importance of removing the unwanted spectra can be seen in Figure 5.3 where

Figure 5.3a is a scatter plot of the first principal component and against the second principal component. The scatter plot shows how the components answer for other materials than the rest raw materials as several clusters can be identified. In Figure 5.3b, however, the data have been downsampled and the shape differs from the shape in Figure 5.3a, and only one cluster can be detected. This indicates that the principal components identify only one material in the cleaned and downsampled data, which is desired.

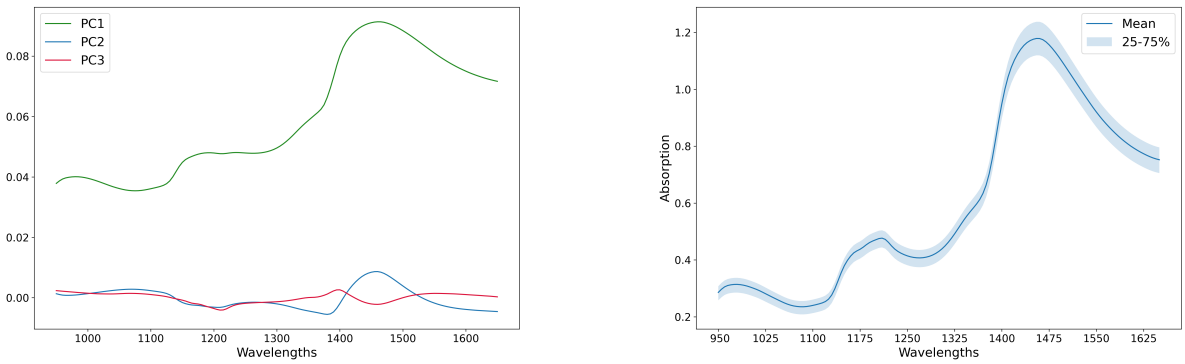


(a) PCA scores before filtering

(b) PCA scores after filtering

Figure 5.3: The figures present PCA scores before and after filtering of bad spectra in week 44. Here the timestamps have been downsampled.

Furthermore, Figure 5.4a presents the loading plots of the three first principal components. As presented in Chapter 2, the loadings can be used to explore the influence each feature has on the principal components. To enable comparison, Figure 5.4b displays the mean of the spectra along with a ribbon showing the 25th and 75th percentiles.



(a) Loading plot

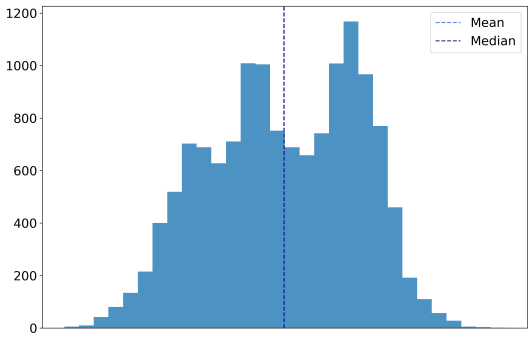
(b) The mean of the cleaned near-infrared data

Figure 5.4: Figures present the loading plot of the three first principal components and the mean of the spectra. Here, PC1 can be seen to be very similar to the mean of the spectra.

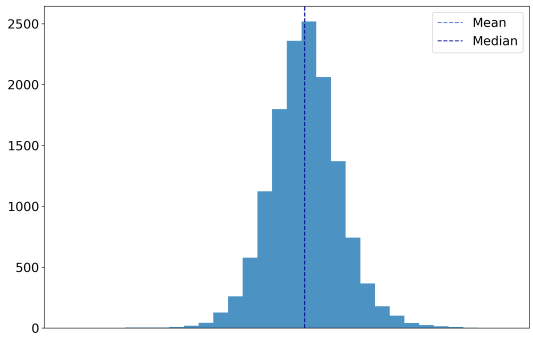
5.1.2 Process data

The process data originally contained a considerable amount of unwanted data. Thus, data filtering was performed to make it suitable for further analysis and to ensure that only relevant and accurate data were included in the final data set. Several feature engineering steps were also carried out, as some features were highly correlated. Figure 5.5 presents the distribution

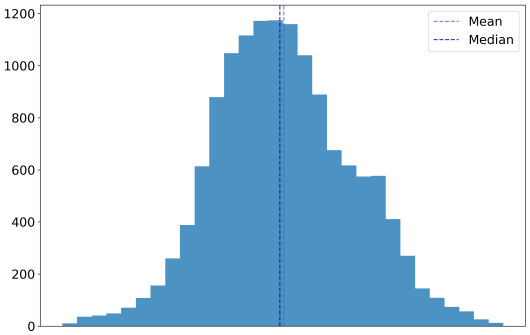
plot of three created features with their corresponding mean and median. Overall, the mean and median are more aligned than the histogram presented in Chapter 3. Figure 5.5a and Figure 5.5b display features used as input features, whereas Figure 5.5c present the distribution of target data.



(a) Feature F01_rel



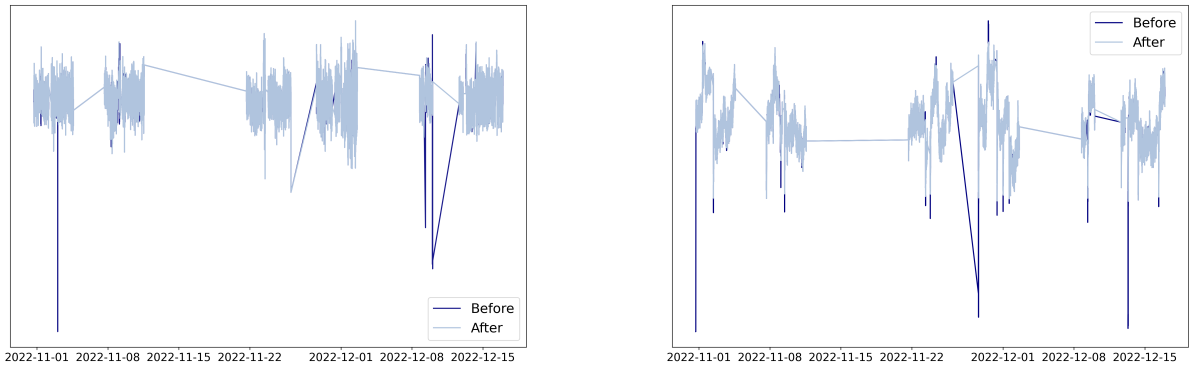
(b) Feature SUM_F01_F02



(c) Feature PT03_lagged

Figure 5.5: Distribution of features from the process data after data cleaning and feature engineering.

In Chapter 2, different types of outliers were presented, and how outliers could be events of interest or unwanted data. In Figure 5.6, the darker time series illustrates the data before data filtering, where several spikes deviate from the rest of the data. The brighter time series, however, shows the data after removing outliers, and the spikes are eliminated. Figure 5.6a portrays the created feature SUM_F01_F02 and Figure 5.6b illustrates the shifted pressure feature, PT03_lagged.



(a) Feature SUM_F01_F01

(b) Feature PT03_lagged

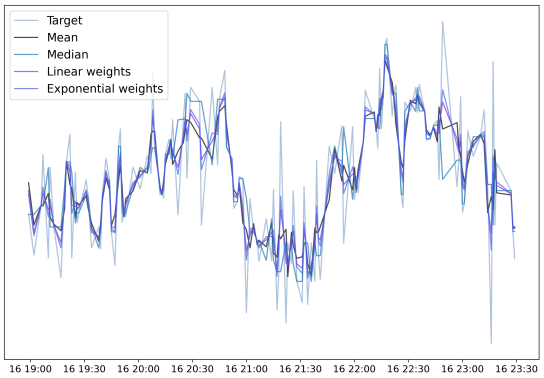
Figure 5.6: Distribution of features from the process data after feature engineering steps.

5.2 Smoothing

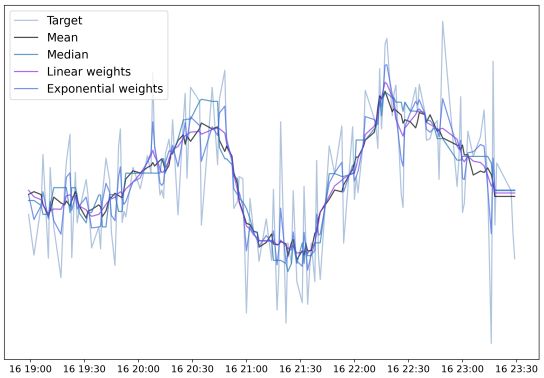
This section presents selected results of the different window sizes. The selection of window sizes covers the range of all sizes explored.

In order to prepare the final data set for modeling, four different smoothing methods were applied. The results of the smoothing methods at different window sizes can be found in Figure 5.7, which depicts a detailed view of the time series. The timeline covered in the plots spans the final four hours of week 50.

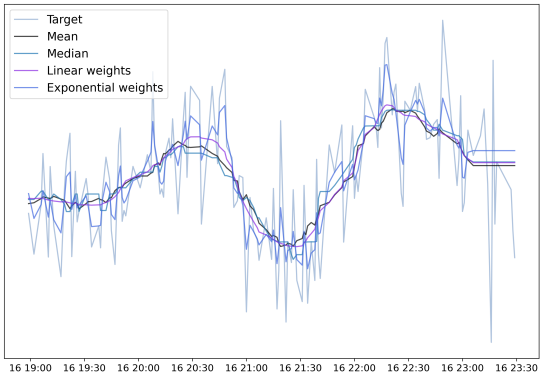
Firstly, the smoothing methods with a smaller window size of $k = 3$ are plotted in Figure 5.7a. Already at such a narrow window, it can be observed that some of the peaks are not as high as the original time series, presented as *Target*. In other words, the time series are visibly smoothed out. The four methods yield more or less the same smoothing result at this window size.
Figure 5.7b shows the results of the smoothing methods using a larger window size of $k = 9$. Here it is observable that the exponential smoothing method differs from the other three methods because it seems to hold more noise in the data.
In Figure 5.7c, it is possible to observe how the different smoothing methods eliminate noise in the data when the k is increased to $k = 21$. Again, exponential smoothing differs from the other methods, and the time series contains peaks similar to the result of exponential smoothing at $k = 9$. The linearly weighted method seems to yield the most smoothed time series, and the mean and median methods yield similar results. At this broad window size, it is possible to observe how smoothing reveals the underlying pattern, and it is possible to detect curves in Figure 5.7c. Also, here the linear interpolation of the missing value at the end of the time series is clearer and can be seen as straight lines at the end.
Finally, the largest window size of $k = 31$ in Figure 5.7d emphasizes the results of the previous smoothing plots. Exponential smoothing at this greater window size will not yield a smoother time series as opposed to linear smoothing. Again, linear smoothing seems to yield the most smooth time series. The curves are clearly visible at this window size, and the linear interpolation of the last observations also reveals how a greater window may result in a loss of information.



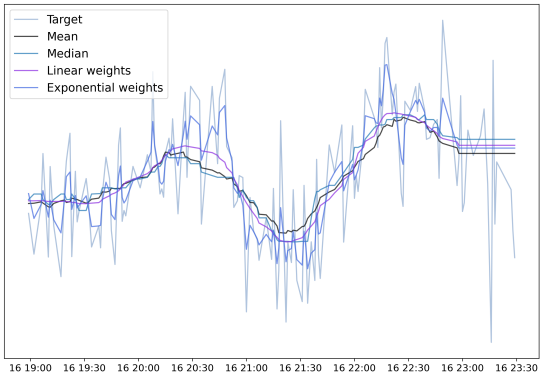
(a) $k = 3$



(b) $k = 9$



(c) $k = 21$



(d) $k = 31$

Figure 5.7: Detailed view of the target feature PT03, and how different smoothing methods are affected by the window sizes.

5.3 Random Forest Regression

After smoothing the data with a given smoothing method and window size, the data were modeled using RandomForestRegressor.

The first result is from $k = 1$, meaning the data were not smoothed before modeling. In other words, no preprocessing except data cleaning and feature engineering was performed on the input data. The result can be seen in Figure 5.8. Here, it can be observed two time series. The original time series is lighter, and the predicted values are visualized with a more purple color. First, it can be observed how the Random Forest Regressor does not perform optimally. To illustrate, the last time segment shows a positive trend that the machine learning algorithm cannot predict. However, the two time series are plotted with transparency, and one can observe some overlapping of the target and predicted values. The overlapping can be seen as a darker purple area and reinforces that the model predicted on no smoothed data does not predict well. Secondly, the predicted time series does not seem to pattern and variation in the target data and seems to be shifted either upwards or downwards.

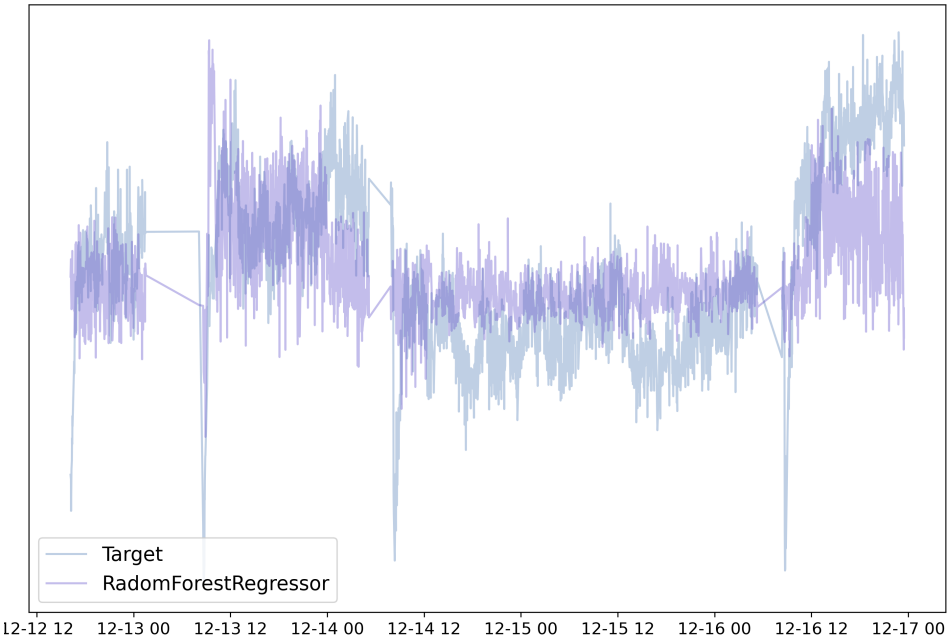
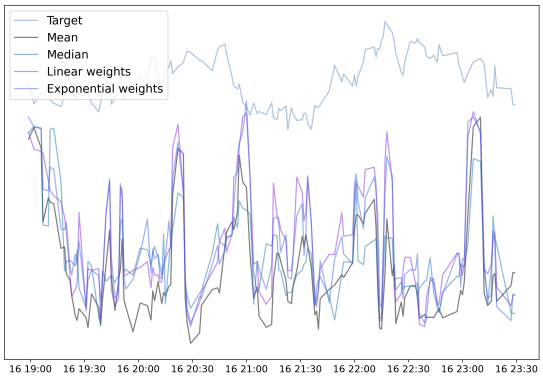
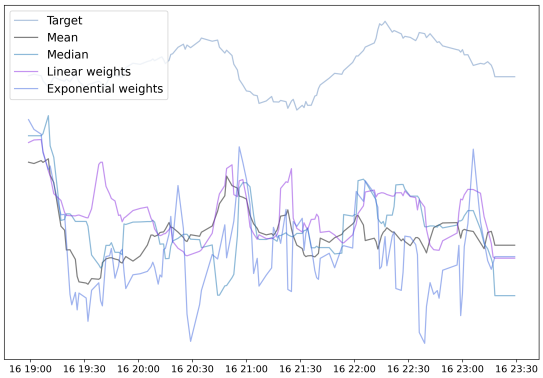


Figure 5.8: Global view of predicted values with no smoothing.

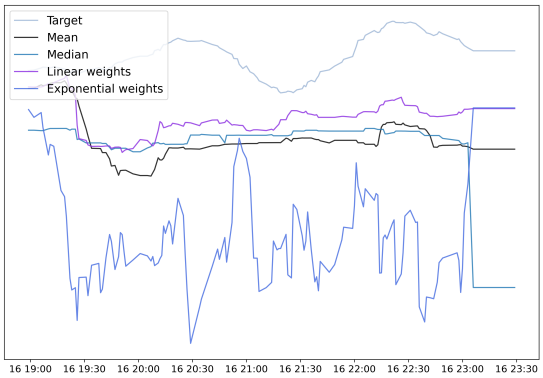
Figure 5.9 shows results from the different smoothing methods and four window sizes in a local view. Firstly, the results of window size $k = 3$ are plotted in Figure 5.9a, where the main observation is how all the different methods are shifted from the target data. This underlines that the machine learning algorithm could not catch the positive trend at the last time segment. Overall, the predicted time series seem to fluctuate more than the target.
Further, window size $k = 9$ is illustrated in Figure 5.9b and presents multiple distinguish time series as an effect of the different smoothing methods on the RandomForestRegressor. Except for the exponential weighting, the methods seem to be less fluctuated. The mean and linear smoothing also appear to predict somewhat the same pattern, but are not able to perform well.
Figure 5.9c displays the window size $k = 21$ and illustrates a greater difference than the two smaller window sizes. Here, it can be observed a more visible effect yielded by the different smoothing methods. Again the exponential smoothing separates itself from the other methods with a result much similar to $k = 9$. Mean and linear smoothing methods still follow the same pattern, and median smoothing yields a more flatten time series.
Moving to the last and most considerable window size $k = 31$ presented in Figure 5.9d, the result presented at more narrow window sizes is emphasized. The predictions after median smoothing are roughly a flat time series, whereas the exponential smoothing present no considerable differences. However, the mean and linear smoothing are shifted upwards toward the target values.



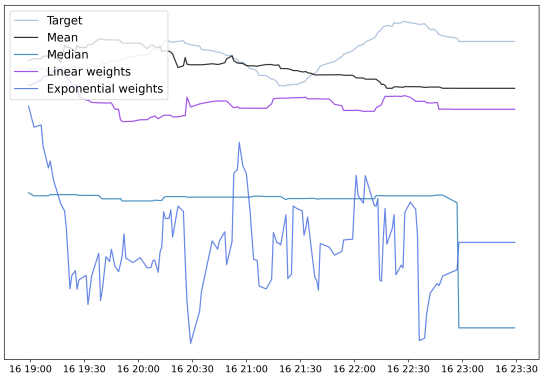
(a) $k = 3$



(b) $k = 9$



(c) $k = 21$



(d) $k = 31$

Figure 5.9: Detailed view of predicted values using RandomForestRegressor.

Table 5.2: Feature importance for RandomForestRegressor.

(a) Exponential smoothing $k = 31$

Rank	Feature	Importance
1	F01_rel	0.615
2	PC3	0.089
3	PC2	0.082
4	SUM_F01_F02	0.062
5	PC4	0.043
6	PC5	0.041
7	PC1	0.037
8	Fat	0.032

(b) Mean smoothing $k = 21$

Rank	Feature	Importance
1	F01_rel	0.611
2	SUM_F01_F02	0.078
3	PC3	0.075
4	PC2	0.071
5	Fat	0.049
6	PC5	0.043
7	PC1	0.038
8	PC4	0.036

With the RandomForestRegressor, extracting feature importance and exploring which features the model learns mostly from is possible. The different models yielded relatively the same rank of features, only with smaller variations. Table 5.2 presents the feature importance of two

machine learning models to illustrate the minor differences. All the models learn mainly from feature F01_rel which is the relative flow rate of the rest raw materials. The other features are of limited importance to the RandomForestRegressor.

5.4 XGBoost Regression

The section is structured similarly to the section above, presenting the different results mainly in a detailed view.

Figure 5.10 presents the result of no smoothing and predicted values using the XGBRegressor. The results show similarities to the result of RandomForestRegressor; the machine learning model does not generalize well to unseen data and only some overlapping parts can be detected as darker areas in the figure.

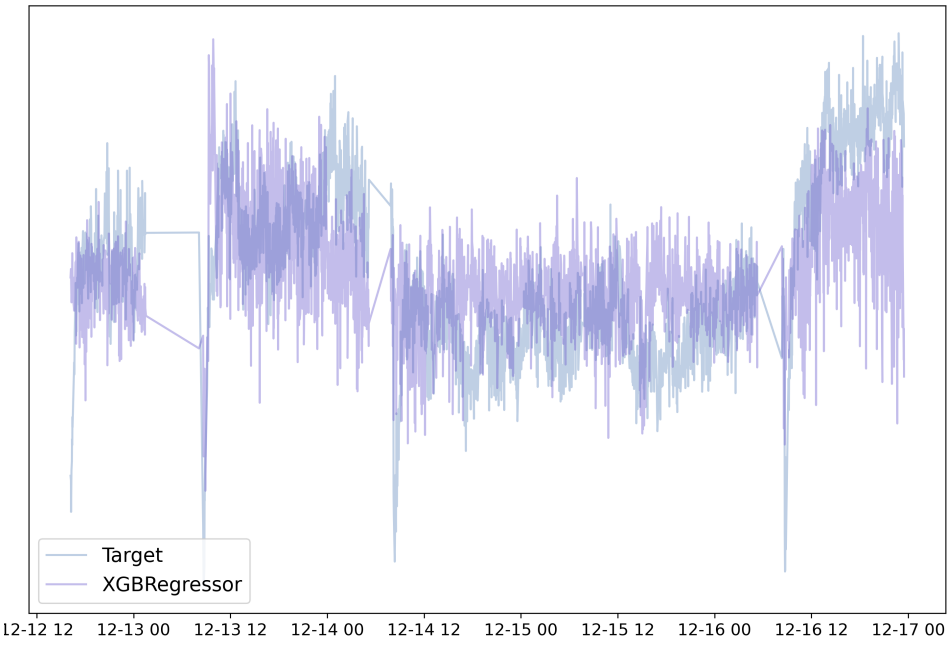
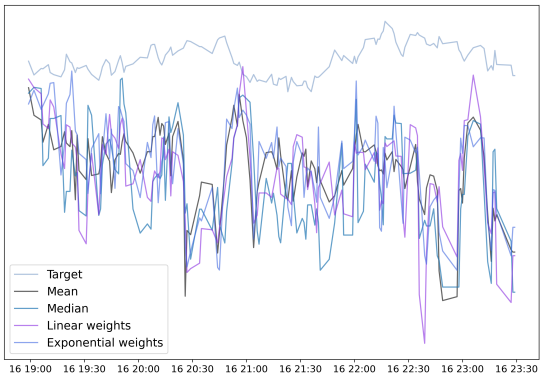
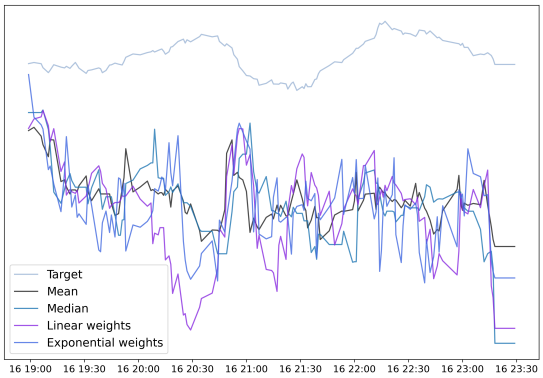


Figure 5.10: Global view of predicted values with no smoothing using XGBRegressor. The darker areas show where the target and predicted time series overlap.

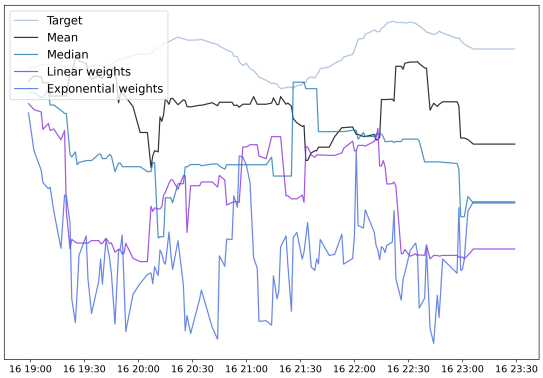
There are various results when further investigating different window sizes in Figure 5.11. Figure 5.11a illustrates how a smoothing method with a narrow window size of $k = 3$ still holds noise in the input data, and the XGBRegressor predicts fluctuated time series. The predicted time series follows a different pattern than the smoothed target data and are more shifted downwards on the y-axis. Further, the linear smoothing appears to have higher and lower peaks than the other smoothing methods.
Moving over to Figure 5.11b and $k = 9$, the noise still influences the predictions, and the results present unstable time series still fluctuating as opposite to the target data. A more significant window size as $k = 21$ in Figure 5.11c yields different predicted time series. Noticeable is again the exponential smoothing that reveals more noise than the other methods.
Figure 5.11d demonstrate the widest window size at $k = 31$. The XGBRegressor here predicts rather more smoothed data except for exponential smoothing. Median smoothing also influences the predictions, and the predicted time series varies to a certain extent.



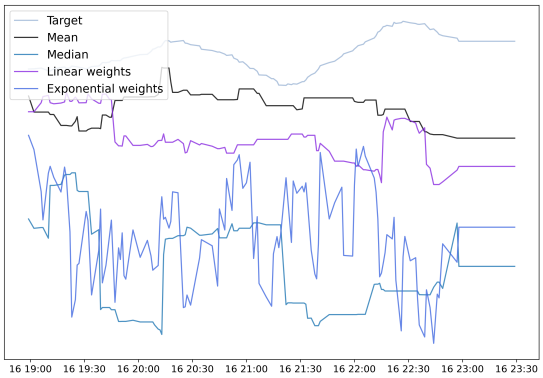
(a) $k = 3$



(b) $k = 9$



(c) $k = 21$



(d) $k = 31$

Figure 5.11: Detailed view of predicted values using XGBRegressor.

The feature importance presented in Table 5.3 shows how the XGBRegressor learns mainly from the relative flow of the rest raw materials. The other features are of low importance.

Table 5.3: Feature importance for XGBRegressor.

(a) Exponential smoothing $k = 31$

Rank	Feature	Importance
1	F01_rel	0.548
2	SUM_F01_F02	0.106
3	PC3	0.090
4	PC2	0.066
5	PC1	0.058
6	PC4	0.055
7	PC5	0.039
8	Fat	0.037

(b) Linear smoothing $k = 31$

Rank	Feature	Importance
1	F01_rel	0.609
2	SUM_F01_F02	0.092
3	PC3	0.082
4	PC4	0.056
5	PC2	0.056
6	PC1	0.040
7	PC5	0.039
8	Fat	0.026

5.5 Method comparison

When the input data had not been preprocessed with any smoothing methods, RandomForestRegressor and XGBRegressor predicted similar time series and are presented in Figure 5.12. As revealed in sections earlier, both the predictions from RandomForestRegressor and XGBRegressor are to a certain degree unreliable in that sense they do predict poorly. Neither of the models gave the impression of generalizing well to new data nor catching trends and patterns seen in the target data.

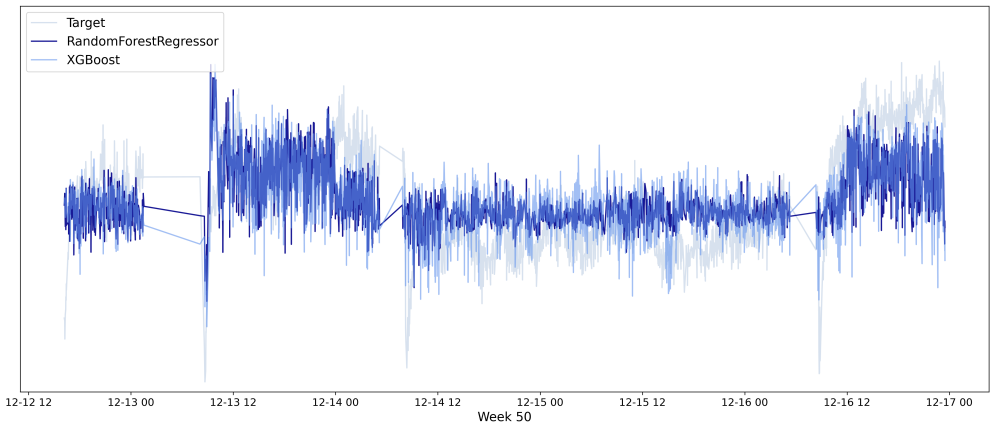


Figure 5.12: Global view of predicted time series after no smoothing.

Figure 5.13 explores the predicted time series in a more detailed view of the last time segment. Here it is observable how the two machine learning algorithms follow a different pattern yet somewhat similar pattern to each other. The XGBRegressor appears to vary more as the time series has several dips of greater extent.

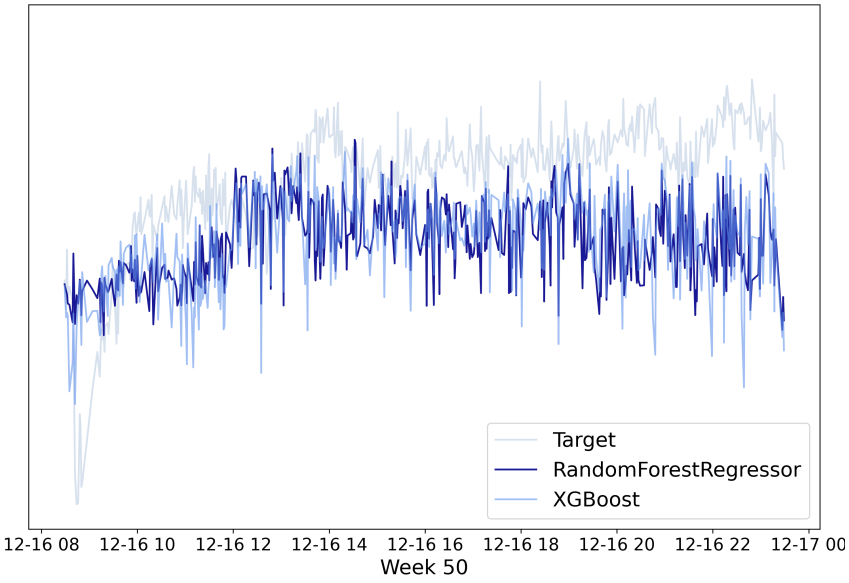


Figure 5.13: Comparison of the two soft sensors in a detailed view. Here, predictions were made after no smoothing.

In Figure 5.14, a global view of the predicted time series that had been smoothed with linearly weighted mean and a window size of $k = 23$ is presented. In the plot, the input target, which was smoothed with the same method and window size, is also included. Equal to earlier results, the models do not seem to be able to predict the smoothed target.

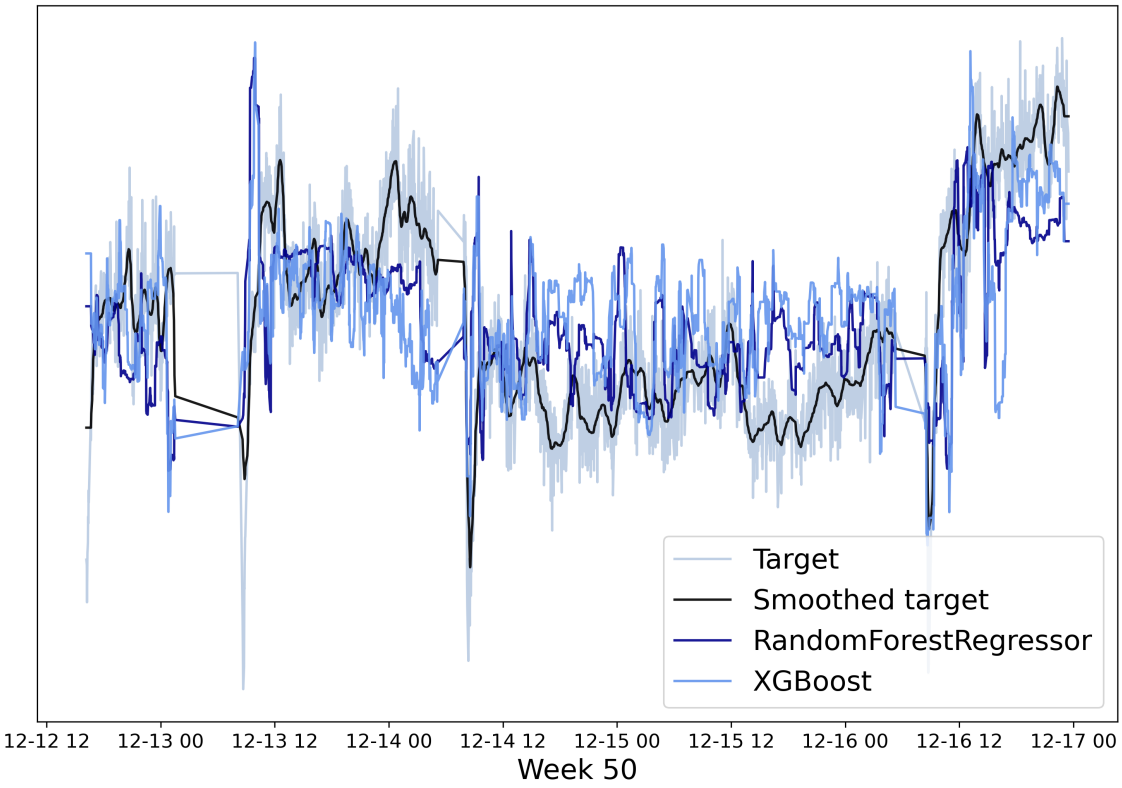
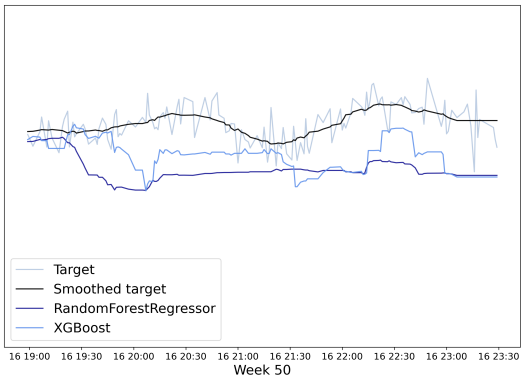
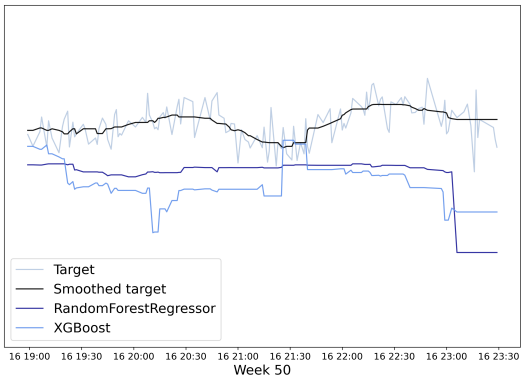


Figure 5.14: Global view of predicted time series after weighted smoothing with $k = 23$.

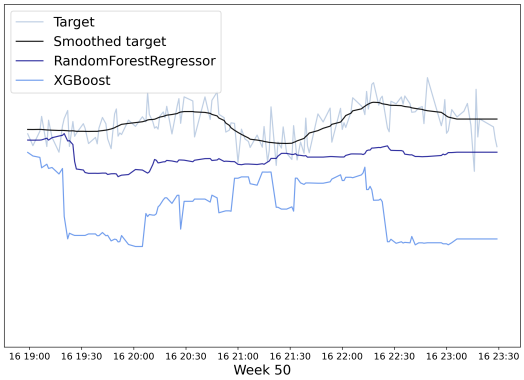
Figure 5.15 portrays the different smoothing methods with window size $k = 21$ and their predicted time series as well as the original target and the smoothed target. First and foremost, the exponential smoothing stands out as equal to the previous results presented. Figure 5.15d reveals how the exponential smoothing at $k = 21$ does not eliminate noise at a significant level hence the predicted time series yielded by RandomForestRegressor and XGBRegressor both contain fluctuations along with poorly estimates. The algorithms suggest similar time series where RandomForestRegressor is shifted slightly above the XGBRegressor. Moreover, RandomForestRegressor predicts roughly smooth time series with little noise and variation. XGBRegressor tends to predict more variation and a greater range of values.



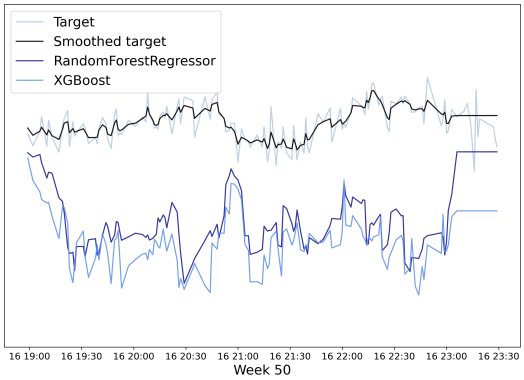
(a) Mean



(b) Median



(c) Linear smoothing

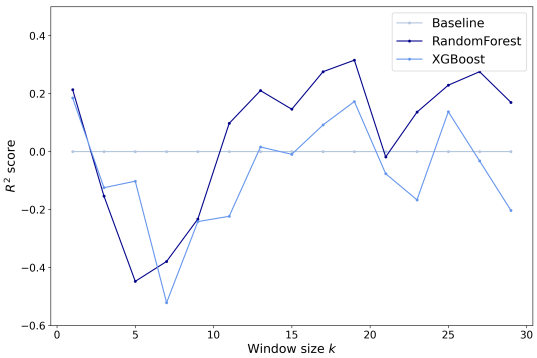


(d) Exponential smoothing

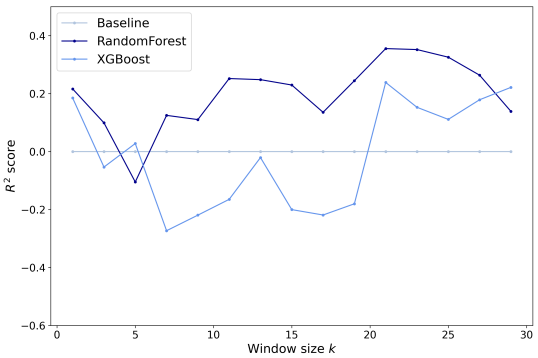
Figure 5.15: Detailed view of predicted target smoothed with $k = 21$ and different methods.

5.5.1 Metrics

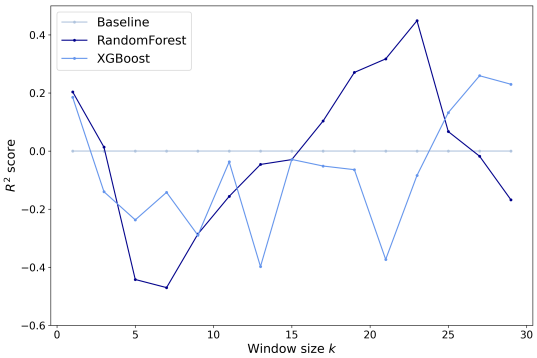
When comparing machine learning algorithms, exploring the metrics to evaluate how well the models perform is common practice. The different metrics are presented and elaborated further in Chapter 2. Figure 5.16 presents the R^2 values alongside with baseline model which are the average of the input target data and equal to $R^2 = 0$. The R^2, or R-squared, represents a fraction of the variance in the target feature and the model's strength. Firstly, it is observable that multiple predictions do not beat the baseline model, especially data preprocessed with exponential smoothing. Secondly, the first value at the x-axes is the window size $k = 1$ and represents no preprocessing in terms of smoothing. In Figure 5.16d, the best predictions can be found at $k = 1$ meaning that the models predicted better on no preprocessing rather than smoothed data using the exponential smoothing. Additionally, predictions from the RandomForestRegressor seem to perform slightly better than XGBRegressor. It can also be observed that the RandomFoestRegressor preprocessed with median smoothing is the one smoothing method that is to some extent the only method above the baseline method. However, the highest R^2 value is obtained through linear smoothing with window size $k = 23$.
Moreover, a fascinating observation is how the XGBRegressor preprocessed with linearly weighted smoothing somewhat oscillates and oscillates opposite to the RandomForestRgressor.



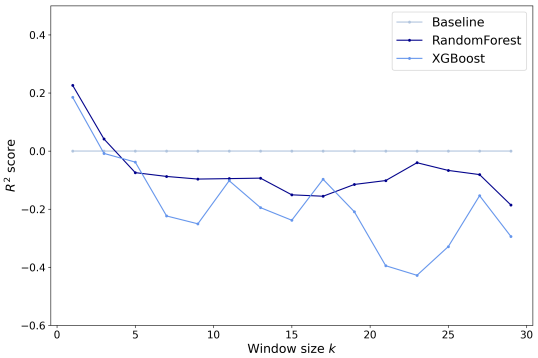
(a) Mean



(b) Median



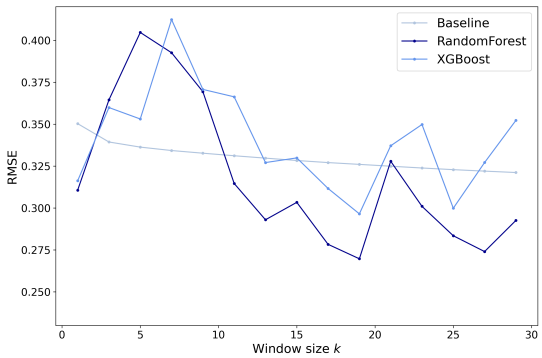
(c) Linear weights



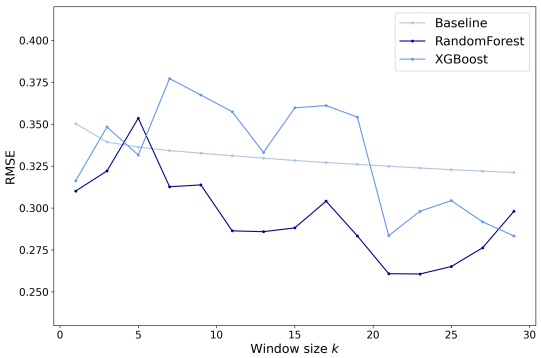
(d) Exponential weights

Figure 5.16: Figures present plotted R^2 values across the window sizes.

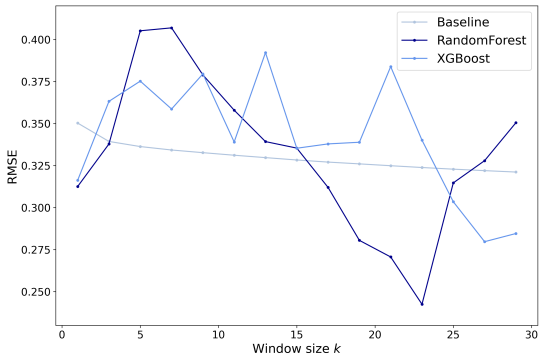
Furthermore, Figure 5.17 presents the RMSE values which measure the average distance between predicted values and true values subsequently indicating a magnitude of errors. A smaller RMSE value is desired. Equal to the R^2 figures, the RMSE values of the baseline model are included as values below the baseline indicates that the machine learning model performs better than the baseline method. The same results retrieved from the R^2 plots can be seen in the RMSE plots. Overall, the RandomForestRegressor performs better than the XGBRegressor.



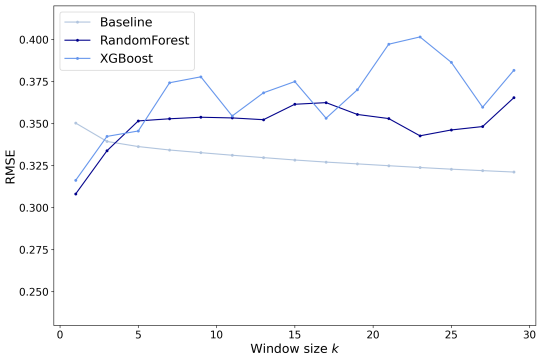
(a) Mean



(b) Median



(c) Linear weights



(d) Exponential weights

Figure 5.17: Figures present plotted RMSE values across the window sizes.

Chapter 6

Discussion

This section discusses the methods performed and the results presented, and interprets them.

This thesis aimed to explore the first step in a data science project, which is the preprocessing part. The preprocessing of industrial time series from Bioco is vital as the data are of great variance due to the composition of the raw materials and sensor data. To fully utilize the time series data to develop soft sensors, the preprocessing must enhance the data quality and prepare the data for further analysis. The objectives of the thesis were formulated as two research questions that are restated here.

- What kind of preprocessing steps are needed for industrial time series?
- How do different smoothing methods and window sizes affect the predictions of the pressures in the hydrolysis pipe?

The development of the preprocessing steps performed was explored through trial and error. It was early on discovered the need for data filtering as the time series presented a vast amount of irrelevant data. Thus removal of bad spectra was performed, and only process data from when the process was running were kept. Furthermore, multiple feature engineering tasks were identified. The near-infrared data were sampled at such a high time resolution; hence feature extraction was performed through Principal Component Analysis and yielded five principal components. Feature selection was made on the process data to obtain the most relevant features for further analysis. Additionally, three features were created to correct time lags and address the correlations between features. Lastly, considering the present noise in industrial time series data, the data were smoothed with four various smoothing methods and 16 different window sizes. The methods were mean smoothing, median smoothing, linearly weighted smoothing, and exponentially weighted smoothing.

Four various smoothing methods were presented and performed on the final data set. The results showed distinct differences from the methods, and the window sizes yielded varied results. Exponential smoothing is notably different from the other methods as the results still consist of noise and fluctuations, especially at broader window sizes. This is a consequence of the τ parameter in the algorithm applied, which depicts the degree of decay. In this thesis, the default value was used, which is $\tau = 1$, a relatively small decay parameter leading to broader windows being irrelevant. This is an effect of the outer values being close to zero meaning the values far from the center are barely included, resulting in a less smooth time series. The metrics plots presented in Chapter 5 underline how smoothing with an exponential window yielded no greater results as both RandomForestRegressor and XGBRegressor did not surpass the baseline model, and the best result was obtained using window size $k = 1$, meaning no smoothing.

Furthermore, the linearly weighted smoothing yielded the most smooth time series with fewer fluctuations, especially at larger window sizes. Additionally, when investigating the metrics, the best predictions were made with linear smoothing at $k = 23$ and RandomForestRegressor. The XGBRegressor model obtained the best metrics with linearly weighted smoothing and window size $k = 27$. Yet, the linearly weighted smoothing did not surpass the baseline method at multiple window sizes, especially when the window sizes were narrow.
The method of mean smoothing is, to a certain degree, similar to linearly weighted smoothing, given that the arithmetic mean is calculated over the window sizes. Mean smoothing has no weights, meaning the values in the window size are equally important. Accordingly, the time series preprocessed with the mean smoothing is less smother than the time series preprocessed with linearly weighted smoothing. None of the machine learning models performed significantly well, implying that the preprocessed time series are too complex to model. Moreover, when comparing mean smoothing and linearly weighted smoothing, they yielded, to a certain degree, similar effects on the machine learning models. While larger window sizes resulted in better predictions, the smaller ones did not beat the baseline model.
When the input data had been preprocessed with median smoothing, the RandomForestRegressor predicted time series that generally surpassed the average method. This is in contrast to the three other methods with more varied results. Median smoothing can thus be said to yield more robust and stable predictions.

On one hand, when choosing a smaller window size, the smoothing method may reveal short-term fluctuations and change points in the process. On the other hand, a broader window size may yield a better representation of the long-term patterns in the time series. Taking into account the autocorrelation and partial autocorrelation plots presented in Chapter 3, it was observed that there were strong correlations between the values in the first hour. This implies complex time series and that the values hold relevant information about past values over a longer time range. In light of the result presented above, it is observed that wider window sizes generate smoother time series and improve predictions from the machine learning models. This does not apply to the exponentially weighted smoothing. Given that a smoothing method is supposed to enhance better machine learning models and the time series are of complex autocorrelation structure, a larger k is proposed when smoothing the data.

6.1 Remaining Challenges

There are multiple aspects to discuss regarding feature engineering. First and foremost, the near-infrared data were only preprocessed in the time domain, not the spectral domain. Consequently, the first principal component of the Principal Component Analysis captured almost 100% of the explained variation. The chemical information is likely stored in the second and third principal components, and the first principal component presents multiplicative scattering effects. This is also underlined when investigating the loading plot of the first principal component. The loading plot presents a pattern highly similar to the mean of the data, which is typical for data with multiplicative scattering effects [36]. If the spectra had been preprocessed with methods like Standard Normal Variate, this scatter information would have been corrected, and the PCA would have yielded more informative principal components. Furthermore, the first principal component is the second least important feature the RandomForestRegressor learns from and is also low-ranked regarding the XGBRegressor, implying that the relevant information is stored in the other principal components.

Multiple features were created, and they ranked high on feature importance. For better predictions, another feature should be created. The target was the pressure sensor, and adding a lagged feature of the pressure data may enhance better predictions. By creating a lagged

feature, the machine learning models may learn a temporal relationship between a value and its lagged version or between a lagged version and another feature, consequently better understanding underlying patterns. Thus adding lagged features should be considered to improve prediction.

The time series data were split into single training and test sets, and the machine learning models may be overfitting as the results present poor model performance. If the machine learning models were optimized with a validation set, the models might have predicted better results as the models had then learned to generalize well to unseen data.
The time series used in this thesis are not of regular time steps as described earlier. By implementing a higher smoothing window, the window size may result in uneven smoothing and not detect the underlying pattern. Since the time series are of multiple time segments, a greater window size may include data from other time segments and negatively affect the denoising of the time series owing to the broad smoothing window ranging over multiple time segments.

The smoothing method excluded the values at the beginnings and the ends equal to $l = (k-1)/2$ resulting in a loss of information. In this thesis, these missing values were linearly interpolated using the last numerical value. Different methods of addressing these missing values should be explored further. The exponentially weighted smoothing method should be a subject for tuning the decay parameter to explore opportunities for more significant preprocessing results.

6.2 Future Work

It is recommended to consider training these models using an alternative data-splitting methodology to enhance the generalization of the machine learning models. This would involve utilizing a different training and test data set with a different partitioning strategy than the one used in this thesis. Cross-validation should also consider where it is essential to retain the temporal order of the data and not randomly select data sets. A suggestion should be to extract time segments and use them as a reference when performing cross-validation.
Furthermore, the cleaning of irrelevant data should be the subject of more robust methods such as change-point detection and density-based outlier detection. The approach in this thesis was ad-hoc and should, in the future, be automated and more general-purposed.
In a long-term view, preprocessed time series can further be modeled into digital objects such as digital twins and upon up for advanced process control.

Chapter 7

Conclusion

In this thesis, multiple preprocessing tasks were investigated to explore the effects preprocessing methods have on soft sensor development. Bioco has enabled utilizing the rest raw materials from poultry production at Nortura and made the rest raw materials into high-quality end products for human and animal consumption. The process line and the composition of the rest raw materials are prone to variation, thus, the time series gathered from the process line must be preprocessed before further soft sensor development. Data filtering and feature engineering steps were carried out to enhance the data quality. Furthermore, four smoothing methods were presented and performed with different window sizes before the smoothed data were used to predict new values. Here, two decision trees model, Random forest regression and XGBoost regression, were selected.

The time series data presented a vast amount of irrelevant data; hence data filtering was performed. This addressed unwanted spectra and insignificant process data from when the process was not running. Furthermore, two features were created to correct timelines, and a third feature was produced to provide more information about the rest raw materials. These created values showed to be of great importance to the soft sensors. Feature extraction of the spectra was provided through Principal Component Analysis and resulted in five principal components. Lastly, several features were highly correlated and provided redundant information. Thus, a subset of the features was selected for further analysis.

Feature engineering and data filtering yielded less varied input data, yet the time series data were complex. The smoothing methods yielded different results where a broader window size clarified the differences. The predictions of no smoothing, meaning $k = 1$, resulted in similar results for both the machine learning models, where they beat the baseline method, which was the average method. Overall, smaller window sizes did not outperform the baseline, and the best predictions were found at $k > 20$. The median smoothing method was the only method that resulted in most predictions above the baseline in contrast to the exponentially weighted smoothing, which yielded results that did not surpass the average method. The early partial autocorrelation plots presented high correlations of values within the first hour, indicating that past values hold relevant information regarding current values. This underlines the need to use a greater window size.

Process data from the food industry are complex and prone to noise. Even after heavy data cleaning, feature engineering, and smoothing, the data presented considerable fluctuation and noise. Therefore, the machine learning model struggled to predict the time series, and bearing in mind the lack of a validation set, the models may be overfitting. However, the results from different smoothing methods demonstrated better accuracy at larger window sizes.

There are multiple improvements needed to enhance better predictions. Firstly, the near-infrared data should be preprocessed in the spectral domain to correct the multiplicative scattering effects, thus obtaining more information regarding the chemical composition of the rest raw materials. Further, a lagged version of the pressure sensor should be included as input data to gain more insight into the relationship between values and features. In addition, the data cleaning was performed manually and with an ad-hoc approach, and an automated method should be developed and adapted to gain more sustained preprocessing.

Preprocessing of industrial time series is a vital task and aims to enhance data quality for further analysis. Data filtering, feature engineering, and smoothing are preprocessing methods explored in this thesis, and the results show the importance of carefully selecting appropriate methods. Feature importance from the machine learning models demonstrates the effects of feature engineering as the created features are ranked high. The accuracy of the predicted time series indicates that larger smoothing windows yield better predictions, whereas a narrow window size worsens the accuracy. Median smoothing generally surpassed the baseline model at each window size, and indicates a more robust and sustain smoothing method.

Bibliography

[1] Statistisk sentralbyrå. *03551: Slakt godkjende til folkemat (tonn), etter statistikkvariabel, type godkjende slakt og år.* URL: https://www.ssb.no/statbank/table/03551/tableViewLayout1/ (visited on 01/18/2023).

[2] Charlotte Krog. *Innovativ bruk av restråstoff fra kylling og kalkun.* 2021. URL: https://phpstack-775229-2636887.cloudwaysapps.com/nyhetsartikler/innovativ-bruk-av-restr%C3%A5stoff-fra-kylling-og-kalkun (visited on 01/17/2023).

[3] *Vi foredler naturens fineste produkter.* URL: https://www.norilia.no/hovedartikler/refining-nature-039-s-finest-products (visited on 01/17/2023).

[4] Diana Lindberg et al. "Kartlegging av restråstoff fra jordbruket". In: *Nofima AS.* Nofima rapportserie 67/2016 (2016). ISSN: 978-82-8296-476-0. URL: https://nofima.brage.unit.no/nofima-xmlui/bitstream/handle/11250/2428846/Rapport%5C%2B67-2016.pdf?sequence=1&isAllowed=y (visited on 01/18/2023).

[5] OECD and FAO. *OECD - FAO Agricultural Outlook 2022-2031.* 2022, pp. 163–177. URL: https://www.oecd-ilibrary.org/content/publication/f1b0b29c-en.

[6] FAO, ed. *The State of Food and Agriculture 2019. Moving forward on food loss and waste reduction.* 2019. Rome: Food and Agriculture Organization of the United Nations, 2019. ISBN: 978-92-5-131789-1.

[7] Thomas A. Aloysius et al. "Chicken Protein Hydrolysates Have Anti-Inflammatory Effects on High-Fat Diet Induced Obesity in Mice". In: *Medicines* 6.1 (2018), p. 5. ISSN: 2305-6320. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6473722/ (visited on 01/12/2023).

[8] United Nations. *The Sustainable Development Goals Report 2022.* United Nations, 2022. ISBN: 978-92-1-101448-8. URL: https://unstats.un.org/sdgs/report/2022/The-Sustainable-Development-Goals-Report-2022.pdf (visited on 01/11/2023).

[9] Nofima. *Smart sensor and optimisation systems for future food biorefineries.* 2021. URL: https://nofima.com/projects/smartbio/ (visited on 04/19/2023).

[10] Nick Hotz. *What is CRISP DM?* 2018. URL: https://www.datascience-pm.com/crisp-dm-2/ (visited on 02/03/2023).

[11] *Forskrift om animalske biprodukter som ikke er beregnet på konsum (animaliebiprodukt-forskriften).* 2016. URL: https://lovdata.no/dokument/SF/forskrift/2016-09-14-1064/*#* (visited on 02/01/2023).

[12] Tone Aspevik et al. "Valorization of Proteins from Co- and By-Products from the Fish and Meat Industry". In: *Top Curr Chem (Z)* 375 (2017), p. 53. URL: https://doi.org/10.1007/s41061-017-0143-6 (visited on 01/18/2023).

[13] National Aeronautics and Space Administration. *Herschel Discovers Infrared Light*. URL: https://spaceplace.nasa.gov/review/posters/herschel/Herschel-ir-activity.pdf (visited on 03/18/2023).

[14] UiB. *Infrarød spektroskopi*. 2020. URL: https://www.uib.no/kj/57286/infrar%C3%B8d-spektroskopi (visited on 03/18/2023).

[15] Bruker Corporation. *Why FT-NIR spectroscopy?* URL: https://www.bruker.com/en/products-and-solutions/infrared-and-raman/ft-nir-spectrometers/what-is-ft-nir-spectroscopy.html (visited on 04/06/2023).

[16] Jens Wold, Martin Kermit, and Astrid Woll. "Rapid Nondestructive Determination of Edible Meat Content in Crabs (Cancer Pagurus) by Near-Infrared Imaging Spectroscopy". In: *Applied spectroscopy* 64.7 (2010). DOI: 10.1366/000370210791666273.

[17] Sebastian Raschka and Vahid Mirjalili. *Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing, 2019. ISBN: 978-1-78995-575-0.

[18] Terence C Mills. *Applied Time Series Analysis: A Practical Guide to Modeling and Forecasting*. 1st ed. United Kingdom: Academic Press Inc, 2019. ISBN: 978-0-12-813117-6.

[19] Stefan Schrunner. *DAT320: Basics - Introduction to time series and sequential data*. 2022.

[20] Elizabeth Ann Maharaj, Pierpaolo D'Urso, and Jorge Caiado. *Time Series Clustering and Classification*. 1st ed. Chapman and Hall/CRC, 2019. ISBN: 978-1-4987-7321-8.

[21] Rob j Hyndman and Geogre Athanansopoulos. *5.2 Some simple forecasting methods*. 3rd ed. 2021. URL: https://otexts.com/fpp3/simple-methods.html (visited on 04/28/2023).

[22] Fazel Famili et al. "Data preprocessing and intelligent data analysis". In: *Intelligent Data Analysis* 1.1 (1997). URL: https://www.sciencedirect.com/science/article/pii/S1088467X98000079 (visited on 02/17/2023).

[23] Vijay Kotu and Bala Deshpande. *Data Science: Concepts and Practice*. Morgan Kaufmann, 2018. ISBN: 978-0-12-814762-7.

[24] Ane Blázquez-García et al. "A review on outlier/anomaly detection in time series data". In: *ACM Computing Surveys* 54 (2021). URL: https://arxiv.org/pdf/2002.04236.pdf.

[25] David Diaz, Mine Çetinkaya-Rundel, and Christopher Barr. *OpenIntro Statistics*. 4th ed. OpenIntro, 2019.

[26] Rob J. Hyndman. "Moving Averages". In: *International Encyclopedia of Statistical Science*. Berlin, Heidelberg: Springer, 2011, pp. 866–869. ISBN: 978-3-642-04898-2. URL: https://doi.org/10.1007/978-3-642-04898-2_380 (visited on 02/22/2023).

[27] *What is Artificial Intelligence (AI)? — IBM*. URL: https://www.ibm.com/topics/artificial-intelligence (visited on 03/01/2023).

[28] *3.1. Cross-validation: evaluating estimator performance*. URL: https://scikit-learn.org/stable/modules/cross_validation.html#time-series-split (visited on 05/10/2023).

[29] *1.10. Decision Trees*. URL: https://scikit-learn/stable/modules/tree.html (visited on 04/07/2023).

[30] David C- Lay, Stephen R. Lay, and Judi J. McDonald. *Linear Algebra and its Applications*. 5th ed. Pearson Education, 2016. ISBN: 978-1-292-09223-2.

[31] Oxana Rodionova, Sergey Kucheryavskiy, and Alexey Pomerantsev. "Efficient tools for principal component analysis of complex data- a tutorial". In: *Chemometrics and Intelligent Laboratory Systems* 213 (2021). DOI: 10.1016/j.chemolab.2021.104304. URL: https://www.sciencedirect.com/science/article/pii/S0169743921000721 (visited on 10/22/2022).

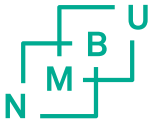
[32] William Mendenhall and Terry Sincich. *A Second Course in Statistics Regression Analysis*. 7th ed. Essex: Pearsons, 2014. ISBN: 978-1-292-04290-9.

[33] Petr Kadlec and Bogdan Gabrys. "Local Learning-Based Adaptive Soft Sensor for Catalyst Activation Prediction". In: *AIChE Journal* 57.5 (2010). DOI: 10.1002/aic.12346.

[34] Bioco. *Hva er enzymatisk hydrolyse?* URL: https://www.bioco.no/hovedartikler/hva-er-enzymatisk-hydrolyse (visited on 04/12/2023).

[35] *sklearn.decomposition.PCA*. URL: https://scikit-learn/stable/modules/generated/sklearn.decomposition.PCA.html (visited on 04/16/2023).

[36] Jean-Michel Roger, Alexandre Mallet, and Federico Marini. "Preprocessing NIR Spectra for Aquaphotomics". In: *Molecules* 27.20 (2022). ISSN: 1420-3049. URL: https://www.mdpi.com/1420-3049/27/20/6795 (visited on 11/28/2022).



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway