Norwegian University
of Life Sciences

Master's Thesis 2024 30 ECTS
Faculty of Science and Technology

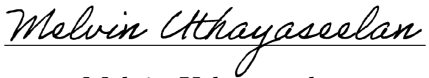
Data-driven optimization of an industrial cheese production process

Melvin Uthayaseelan
Industrial Economics

THIS PAGE IS INTENTIONALLY LEFT BLANK

Acknowledgment

This master's thesis represents the conclusion of graduate studies in Industrial Economy (Technology) with a specialization in Computer Science at the Norwegian University of Life Sciences (NMBU). The thesis was written for the Faculty of Science and Technology (REALTEK) and completed in the spring of 2024, marking the end of a five-year master's program as a civil engineer in industrial economics. I extend my deepest gratitude to my supervisors, Kristian Hovde Liland, a Senior Researcher and professor from NMBU's Faculty of Science and Technology, and Ingrid Måge, a Senior Researcher from Nofima, for their unwavering guidance and support throughout my master's thesis. Their profound interest in my work and exceptional mentorship have been invaluable throughout the entire period. The discussions and feedback they provided were crucial in shaping the direction of my research. Moreover, I would like to express my sincere appreciation to Tine Jæren, who provided me with the opportunity to collaborate with them on this project and shared critical information that significantly contributed to its formulation. I am deeply grateful for the time and effort they devoted to this project. As my journey at NMBU comes to a close, I reflect on the wealth of knowledge and experiences I have gained. Throughout this journey, I have encountered both challenging and rewarding days. However, the lessons learned, both inside and outside the classroom, hold immense value for me. Lastly, I would like to take a moment to express my heartfelt gratitude to the people who have been a constant source of support throughout my academic journey. My family and friends have been there every step of the way, providing me with the encouragement and motivation I needed to persevere. They have cheered me on during my successes and picked me up during my failures. Without their unwavering support, I wouldn't have been able to achieve my goals. I also want to thank all my fellow students and teachers at NMBU. The time I spent at this institution has been exceptional, mainly due to the incredible people I had the privilege of sharing this experience with. My teachers have played a pivotal role in shaping my academic career, always challenging me to push myself beyond my limits and inspiring me to think critically and creatively. My fellow students have been an endless source of inspiration and friendship, making each day at NMBU memorable and enriching. As I embark on the next phase of my journey, I am deeply grateful for the invaluable lessons learned and the enduring relationships built during my time at NMBU.



Melvin Uthayaseelan
Ås, 15. May 2024

THIS PAGE IS INTENTIONALLY LEFT BLANK

Abstract

As the world rapidly advances with groundbreaking technology, the food sector is no exception. Artificial intelligence is increasingly utilized in the food industry to enhance production and quality. However, ensuring quality in the food industry can be challenging due to environmental factors such as fluctuations in temperature and humidity levels, as well as changes in raw material sources. To guarantee high quality and safety standards, it is crucial to develop systems that monitor and regulate these variables, ensuring that consumers receive safe and enjoyable food products. This master thesis delves into applying machine learning and optimization techniques to predict the dry matter content and optimize the quality and production of Norvegia cheese at Tine Jæren. Dry matter plays a critical role in determining the quality of Norvegia cheese, affecting its flavor and texture significantly. The optimal dry matter content for Norvegia cheese should be between 57.6% and 57.7%, as it ensures the highest quality and customer satisfaction. The study developed machine learning models that can be used to gain insights from data and predict dry matter content. Different feature selection techniques were employed to identify the variables affecting dry matter content and refine the predictive models. The optimal models will be integrated as a predictive component into an optimization model utilizing an evolutionary algorithm to optimize and enhance the production and dry matter content of fresh Norvegia cheese. This approach optimizes essential dry matter parameters affecting dry matter to improve quality and production efficiency while ensuring customers receive safe and enjoyable food products. The best optimization model developed in this study has been integrated into a user-friendly website tailored for industry application. This website serves as both a prototype and a demonstration of how the optimization model can be implemented in daily operations. This approach can help manage resources and minimize waste, a crucial factor in sustainable food production. In summary, the application of machine learning and optimization techniques to predict dry matter content and optimize the quality and production of Norvegia cheese has the potential to improve the food industry and pave the way for more sustainable and efficient food production practices.

Table of Contents

List of Figures

List of Tables

Abbreviations

Abbreviation	Meaning
AI	Artificial Intelligence
ML	Machine Learning
CV	Cross-validation
LOGO	Leave-One-Group-Out
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
R2	R-Squared
Lasso	The Least Absolute Shrinkage and Selection Operator
LR	Linear Regression
MLR	Multiple Linear Regression
OLS	Ordinary Least Squares
IG	Information Gain
RF	Random Forest
HGBR	Histogram-based Gradient Boosting Regression
PCA	Principal Component Analysis
PC	Principal Component
PCR	Principal Component Regression
PLSR	Partial Least Squares Regression
SFS	Sequential Feature Selection
VIP	Variable Importance in Projection
RENT	Repeated Elastic Net Technique
BIC	Bayesian Information Criterion
VS1	Validation Study 1
VS2	Validation Study 2
LP	Linear Programming
NLP	Non-Linear Programming
DE	Differential Evolution
RC	Recombination Parameter
CRISP-DM	Cross-Industry Standard Process for Data Mining

THIS PAGE IS INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

1.1 Motivation

Food industries worldwide play a crucial role in sourcing raw materials from farmers and local suppliers, refining, processing, and packaging these into safe, consumable consumer products. Enhancing and maintaining food quality is essential for ensuring food safety and meeting customer expectations. However, many food industries face significant challenges in product quality, often due to various environmental factors. Technology is now an essential part of the food industry, with Artificial intelligence (AI) offering numerous opportunities to automate and optimize processes, reducing human error. Machine learning originated in the 1950s and has played a significant role in automating and enhancing tasks. From its early stage with simple algorithms like the perceptron, it has evolved into more advanced techniques such as artificial neural networks [1]. As technology advances and digitization increases, public institutions increasingly adopt machine learning and AI in the food industry. Leveraging machine learning outcomes can help reduce costs and optimize resources, serving as digital decision support to pave the way for innovation and success in an industry [2]. To continue being a vital part of the global supply chain, the food industry must adopt new technologies to maintain and improve food quality. This adaptation is crucial to ensure the industry can sustain its vital role in providing safe, high-quality food products.

Established in 2001, TINE is a cooperative enterprise owned by over 8,000 farmers, enabling them to process and add value to their produce. The company is renowned for its diverse range of dairy products, including milk, cheese, yogurt, butter, and juice [3]. Focusing on dairy products, TINE aims to be a leading and sustainable supplier of branded food and beverages. TINE is committed to ongoing innovation in the dairy sector, exploring new technologies and processes to enhance the production and quality of dairy products, including leveraging cutting-edge technology to improve efficiency and sustainability in their operations. The company is working towards this goal by closely monitoring consumer preferences and delivering goods and innovative products in consumers' desired formats. Like many other food industries, TINE faces challenges in maintaining and enhancing the quality of cheese production.

Cheese production is a complex process that faces unique challenges that are not present in other industries. Some of the most significant challenges are managing batches with additives and seasonal variations. Various additives are used in cheese production to enhance flavor, texture, and shelf life. The variation in texture, flavor, and overall composition of cheese can be impacted by the amount of additives used in different batches. The addition must be carefully managed to ensure that each batch remains consistent with expected outcomes. The composition of milk, the primary ingredient in cheese, can significantly vary with the seasons due to factors such as the health and diet of dairy animals and their environment. This can affect the fat and protein content of the milk. All cheese in a batch undergoes the same processing steps, resulting in a consistent outcome. Variations in processing steps can lead to inconsistencies in the final

product, affecting flavor, texture, and overall quality. This poses a challenge because batches may have different characteristics. Therefore, the process must be adjusted to accommodate natural fluctuations without affecting cheese quality.

1.2 Objective of the Thesis

The case study delves deep into the operations at Tine Jæren, a prominent producer of Norvegia cheese. Tine Jæren faces the challenge of maintaining consistent quality in cheese production due to fluctuations in the dry matter content of the cheese. The dry matter between 10.10.2022 and 15.01.2024 has a mean value and a standard deviation of 57.68 and 0.55, respectively. This gives a 99% confidence interval of (57.65, 57.71), which means we are 99% confident that the true population mean of the dry matter falls within this range. The desired level of dry matter for Norvegia is 57.6-57.7%. The dry matter content is a crucial indicator of cheese quality and significantly influences the taste. By effectively controlling the dry matter content, we can enhance the product's quality and flavor. In addition to enhancing quality, we can improve efficiency by reducing waste from processes or materials that do not contribute to maintaining optimal quality. This thesis aims to apply machine learning and optimization techniques to forecast and optimize the dry matter content in cheese production. This approach will enable a comprehensive understanding of the factors influencing variations in dry matter, enhance production efficiency, minimizing waste, improve quality, and ensure that the taste and texture of Norvegia cheese meet consumer expectations.

1.3 Research Focus and Questions

We have formulated a specific research question in light of the complexities outlined in the objectives. This question explores the potential application of machine learning and optimization techniques to enhance the quality of Norvegia cheese, focusing specifically on:

How can machine learning and optimization techniques be applied to improve the quality and production of Norvegia cheese?

To solve the problem formulated in the research question, exploration and data modeling have been conducted based on the following research questions:

- **Research question 1:** How can machine learning models be used to predict the dry matter content?
- **Research question 2:** How can we use the insights from the predictive models to create an optimization model to improve the quality of Norvegia Cheese?
- **Research question 3:** How can we use the results as digital decision support to improve the industry's production process?

Research question 1 is crucial for understanding the underlying patterns that affect the quality of Norvegia cheese. By applying machine learning models to predict dry matter content, this question aims to uncover the specific factors that directly impact cheese quality. Gaining insights from these models could enhance our understanding and significantly improve the consistency and efficiency of the cheese-making process. Research question 2 is crucial for identifying the critical factors from the model and using optimization techniques to enhance the quality of Norvegia Cheese. Research question 3 focuses on using the findings from Research questions 1 and 2 to enhance the overall production process in the industry. Limitations have

been defined in collaboration with the project supervisors, taking into account the project's constraints and aiming to achieve satisfactory results. These limitations will be mentioned and discussed throughout this thesis.

1.4 Thesis Structure

This chapter delves into the motivation behind the master's thesis and the challenges encountered by Tine Jæren in cheese production. Chapter 2 provides the necessary theoretical framework to address these challenges. The theory chapter is divided into three parts, each covering different topics. The first part will discuss datasets, data preprocessing, and an introduction to machine learning. The second part covers machine learning algorithms and feature selection techniques. Lastly, the third part focuses on optimization techniques and the CRISP-DM framework. Chapter 3 introduces the methodology, detailing how the data is processed and how machine learning algorithms are implemented. Chapter 4 discusses the results obtained from the machine learning models and uses insights from these predictive models to develop an optimization model that will be integrated into a website using Streamlit. Chapter 5 entails an analysis and discussion of the different models, along with reflections on and challenges faced during this thesis. Chapter 6 provides a conclusion and outlines future work. Figure 1 visually illustrates the structure of the master's thesis.

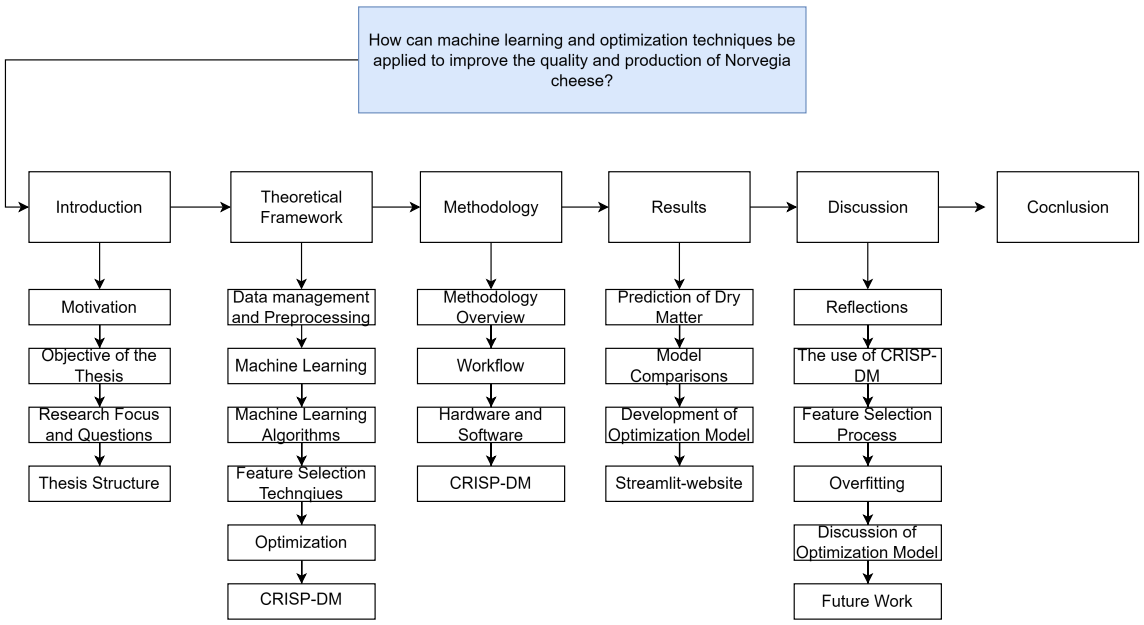


Figure 1: Thesis structure

Chapter 2

Theoretical Framework

2.1 Data Management and Preprocessing

A dataset is a structured collection of data, typically presented in a tabular format for analysis and processing. The data within a dataset generally are related and often used for a single project. Each column in a dataset describes a specific variable, and each row corresponds to an instance. The number of columns can vary from two (in a bivariate dataset) to multiple columns (in a multivariate dataset). A dataset can include many data types, such as categorical, numerical, time series and text data [4]. Working with raw data, identifying patterns and trends among the variables can hold valuable insights and unveil underlying structures. This will make a dataset a powerful tool for decision-making.

Real-world data is messy and is not often suited for data scientist tasks. Data quality is essential for decision-making, and good data quality can be obtained through data preprocessing. Data preprocessing transforms the data into a format that is more easily and effective to handle in data science projects. The better the data represents the problem, the better the results. Inaccuracies, missing values, and inconsistencies can give wrong results, leading to misleading conclusions [5]. Data preprocessing is crucial to ensure the dataset can be analyzed accurately and reliably. This extensive process involves correcting data errors, filling in missing values, resolving inconsistencies, enhancing the overall quality of the dataset, and making it suitable for practical analysis and insightful decision-making. Scaling is an important step in data science projects as it standardizes data, making it easier for decision-making tools to operate more effectively. This process helps reduce potential biases caused by natural variations in the data. Encoding categorical variables that are essential for the analysis is also important because computers do not understand string values. Through One-hot encoding, we convert these variables into a form that the computer can handle more effectively. Beyond preprocessing, understanding the type and scale of the data is essential for selecting appropriate analytical techniques and models.

Missing Values

Missing values are common in real-world datasets and can affect the performance of machine-learning models. The simplest way to deal with it is to remove rows and variables with many missing values. Another way is to use imputation techniques to fill in the missing values. One of the most common techniques is to use the mean or median values of each column to fill in the respective missing values. Iterative imputer is a more advanced technique for multivariate data. It uses an estimator, such as a machine learning model, to estimate the missing values. The Iterative imputer estimates the missing values being imputed using the data available in other features [6]. Using imputation techniques to fill in missing values can greatly improve the effectiveness and robustness of machine learning models. However, imputing missing values is not always reliable, especially when many are missing. In such cases, removing rows and variables would be more appropriate.

2.2 Machine Learning

Machine learning (ML) is a subfield of AI that teaches a computer to perform tasks without explicit programming. Instead, data are fed into and trained in an algorithm to improve outcomes with experience gradually. Over time, we have developed many algorithms to help us with tasks such as classification or prediction. The result of an ML model is a prediction generated from input data that matches the format used during its training, effectively applying learned patterns to new, unseen examples [7]. The term ML was coined by Arthur Samuel in 1959 at IBM, who was developing AI that could play checkers and even outplay its programmer [8]. Half a century later, many predictive models are embedded in numerous products we use daily. Predictive models can classify data, such as determining whether a human is on the road or a patient has cancer. ML can also help us to identify patterns and relationships between input features and target values. We can distinguish between supervised and unsupervised learning.

Supervised learning is a method where the data consists of n observations, each with p-dimensional features. The data can be represented as a matrix $\mathbf{X} = \left(\mathbf{x}_1^\mathsf{T}, \ldots, \mathbf{x}_n^\mathsf{T}\right) \in \mathbb{R}^{n \times p}$, and an associated vector of target values $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$, where each y_i represents a single target outcome, either a class label or a real number. In supervised learning, the input and output data are known in advance. The goal is to train a model using an ML algorithm that can detect the patterns in the input variables to predict the target variable [9]. Loss functions in ML algorithms help measure the cost of inaccuracies in predicting output variables. These functions compare expected and actual outcomes and guide the learning process by highlighting their differences. The goal is to minimize the loss functions by minimizing the differences between expected and actual outcomes. Supervised learning encompasses both classification and regression, with y_i representing either categorical or continuous values, depending on the specific task. Many supervised ML algorithms are available for identifying patterns and making predictions, with none inherently superior to the others [10]. Every dataset is unique and requires a specific machine-learning algorithm to generate reliable predictions.
Conversely, unsupervised learning seeks to uncover data patterns, relationships, or structures without relying on the labeled output $\mathbf{y}$. Various methods are available to accomplish these objectives. One such method is clustering, which involves grouping similar things and discovering hidden patterns and insights based on certain features or characteristics. [11]. Principal Component Analysis is another unsupervised learning technique used for data compression and feature extraction while preserving essential information. This method is explored further in later sections. Unsupervised learning is used in many places, such as stores, to determine what products customers prefer to buy. By tracking and grouping spending patterns, the store can create better ads and sales offers. Figure 2 illustrates the structure of a dataset commonly used for ML tasks. Supervised learning utilizes the matrix $\mathbf{X}$, which contains the input or independent variables, and the vector $\mathbf{y}$, corresponding to the response or target variables that the model aims to predict. This structure is used to investigate the relationship between $\mathbf{X}$ and $\mathbf{y}$ across various observations. In contrast, unsupervised learning only utilizes the matrix $\mathbf{X}$ to detect patterns and similarities without any associated target outputs

X

Observation	Variable 1	Variable 2	Variable 3
1	x_{11}	x_{12}	x_{13}
2	x_{21}	x_{22}	x_{23}
3	x_{31}	x_{32}	x_{33}

y

Observation	Response
1	y_1
2	y_2
3	y_3

Figure 2: Example of a dataset used in machine learning tasks, featuring matrix $\mathbf{X}$ employed for unsupervised learning tasks, while $\mathbf{X}$ and corresponding output vector $\mathbf{y}$ are specifically utilized in supervised learning tasks.

2.2.1 High-dimensional Data

A high-dimensional data space is often described as a dataset with a large number of features compared to the number of observations. In a high-dimensional data space, the data points are more sparse, making it harder to capture the underlying patterns. The model's learning capacity increases as the number of features also increases. This can make the model learn from noisy data points and random fluctuations. Challenges like these can often lead to overfitting. Overfitting is an undesirable model behavior in which the model learns to fit the training data so precisely that it captures noise or specific characteristics of the training data. This may not generalize well to new, unseen data, causing the model to perform poorly on test data [12]. High-dimensional data can also lead to multicollinearity. This statistical phenomenon occurs when more than two independent variables are highly correlated due to their high dimensionality. Highly correlated variables contain similar information, making it challenging to determine the feature's individual impact on the model output [13]. These challenges encountered in high-dimensional data often negatively affect the performance and accuracy of the predictions. Throughout this thesis, we will explore different ML algorithms and techniques to overcome these challenges.

2.2.2 Training, Validation and Testing

During preprocessing, the dataset is typically divided into training, validation, and testing subsets. The training set is utilized to train or fit the prediction model, enabling it to identify patterns and make predictions. When training a model on a specific dataset, it's crucial for the model to generalize effectively to unseen data. Achieving this balance is essential to avoid overfitting, where the model performs exceptionally well on the training data but poorly on test data. The validation set is used to evaluate the model's performance during hyperparameter optimization, which is detailed in Chapter 2.2.5. This provides insight into how the model behaves during processing. By examining the validation score, adjustments can be made to enhance the model's performance. Finally, the test data serves as the final evaluation of the model, assessing the algorithm's predictive power on unseen data. The results from the test data indicate the effectiveness of the learning process. Figure 3 illustrates the data being divided into training, validation and test set.

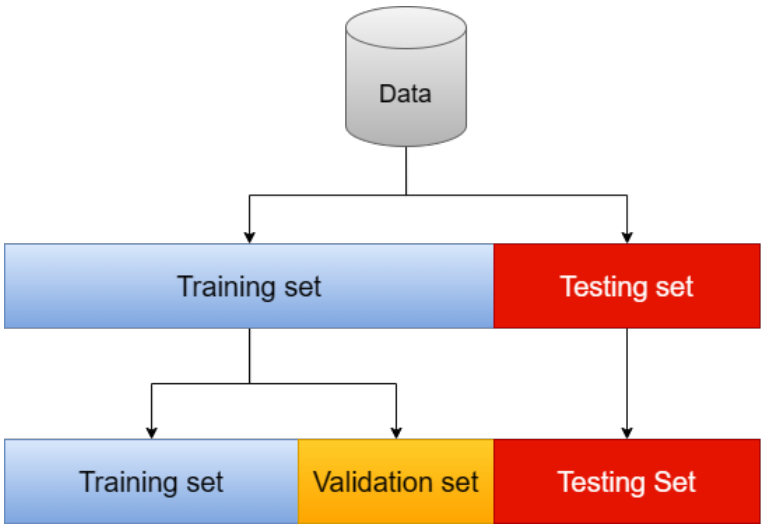


Figure 3: The figure illustrates the data divided into three sets: training, validation, and testing.

2.2.3 Cross-validation

Cross-validation (CV) is a technique used to estimate the performance of an algorithm, often serving as an alternative to a validation set. This technique offers a more robust assessment compared to the training data evaluation since it predicts data that was not used during the model's training process. CV divides the dataset into multiple folds, each used as the validation set while the remaining folds serve as the training set. The performance scores from each fold are then aggregated to obtain a single score. The primary purpose of CV is to prevent overfitting by evaluating the model on multiple validation sets, providing more realistic estimates of model performance. Various techniques exist for CV, and for our research, we will focus on the Leave-One-Group-Out (LOGO) method [14]. LOGO is a form of CV used in ML when datasets contain groups of data that may be related to each other in a way that affects the model's generalizability. With this technique, one group is held back as a test set while the rest of the data are used for training. This process is repeated for each group during the training process [15]. This technique ensures that the model can generalize well to new, unseen groups.

2.2.4 Performance Metrics

Performance metrics are used to evaluate the performance of an ML model. High performance is achieved when the algorithm predicts the correct value for each observation. Different types of performance metrics are used depending on the problem type. For regression problems, performance metrics like the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and coefficient of determination (R^2) are frequently used for evaluation.

Mean Squared Error

MSE is often used in regression problems to find the average squared difference between actual and predicted values. Squaring the differences eliminates negative values. The metric is also sensitive to significant errors due to the squared term, which makes MSE prone to the influence of outliers [16]. This could help the model to predict extreme values more accurately by penalizing large deviations. A model with no mistakes would have an MSE of zero. As the model error increases, the MSE value also rises. The formula for MSE is:

$$MSE = \frac{1}{n}\sum(y_i - \hat{y}_i)^2 \tag{2.1}$$

Where y_i is the i^{th} observed value, $\hat{y}_i$ is the corresponding predicted value, and n is the number of observations. MSE is also a commonly used loss function to minimize errors in regression problems, making it computationally efficient and straightforward for predictive modeling.

Root Mean Squared Error

RMSE is the standard deviation of the residuals, indicating how far from the regression line data points are. Like MSE, the lower the RMSE, the better the model's ability to predict accurately. Conversely, a higher RMSE signifies a lower performance of the model. RMSE values can range from zero to positive infinity and are expressed in the same units as the target variable.
The RMSE formula is as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum(y_i - \hat{y}_i)^2} \tag{2.2}$$

As shown above, the RMSE is like the MSE with a square root applied. This adjustment means that RMSE, while providing a scale of error that is more interpretable and similar to the original data, remains sensitive to outliers. This sensitivity can potentially lead to overfitting if not managed correctly during the training process [16].

R-squared

R^2 is known as the coefficient of determination and is a statistical measure used in ML to determine the proportion of variance in a dependent variable that can be predicted or explained by an independent variable [16]. In other words, R^2 calculates how well a regression model predicts the outcome of observed data. The formula for R^2 is given by:

$$R^2 = 1 - \frac{\text{Unexplained Variation}}{\text{Total Variation}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{2.3}$$

In the formula above, y_i is the i^{th} observed value, $\hat{y}_i$ is the corresponding predicted value, and $\bar{y}$ is the mean of the observed values. For training data, R^2 values typically range from 0 to 1. A value equal to zero indicates that the model does not explain any variance in the dependent variable based on the independent variables [16]. A higher R^2 value signifies a more significant proportion of the variance in the dependent variable is explained by the model. Specifically, a value of 1, or 100%, indicates that the model perfectly predicts the relationship between the independent variable and the dependent variable [16]. For validation or test data, R^2 values theoretically do not have a lower bound. A negative R^2 value means that the variance in the residuals is larger than in the data. This is because R^2 measures how well the model fits the data compared to a simple mean baseline. Therefore, if the predictions are worse than simply using a baseline like the mean of the dependent variable, R^2 can be negative [17].

2.2.5 Hyperparameter Optimization

Most ML algorithms are configured by hyperparameters. A hyperparameter is a configurable value in the model that can be adjusted to control how an ML algorithm behaves [18]. Finding the optimal hyperparameters is a crucial step in constructing a ML model, as the values of these hyperparameters often substantially influence the complexity, behavior, speed, and other aspects. Model tuning is a technique used to determine the most suitable hyperparameters. As discussed in Chapter 2.2.2, the validation set is typically used for this tuning process. The duration of the tuning process can vary significantly, ranging from seconds to many hours, depending on factors such as the dataset size, the algorithm used, and the hyperparameter optimization technique employed. Many hyperparameter optimization techniques exist, such as Grid search, Random search, and Bayesian optimization. Grid search can be used to find the optimal hyperparameters by defining a range for each hyperparameter and exhaustively evaluating every combination of values [18]. This technique is simple to use and implement but can be computationally expensive. On the other hand, random search randomly samples hyperparameter values from a predefined distribution for a specified number of iterations. Random search tends to be more efficient than grid search for high-dimensional hyperparameter space, as it allows for better exploration of the hyperparameter space. However, it may not cover the entire hyperparameter space evenly [18]. Bayesian optimization is a model-based technique used to search for optimal hyperparameters efficiently. The Bayesian optimizer employs probabilistic models to estimate the optimal hyperparameters and automates the tuning process by utilizing previous information to find more suitable hyperparameters. Optuna is a hyperparameter optimization software framework, that automates the search for optimal hyperparameters using Bayesian optimization. By leveraging a history record of trials, Optuna intelligently determines which hyperparameters to try in the next iteration. In a short amount of time, this framework can efficiently find good hyperparameters, enhancing a model's performance [19].

2.3 Machine Learning Algorithms

2.3.1 Linear Regression

Simple linear regression (LR) is a statistical model that estimates the linear relationship between a dependent variable Y and one independent variable x. Normally, the objective is to predict the value of the output variable Y based on the value of the input variable x. Multiple linear regression (MLR) is a generalization of simple LR, which estimates the linear relationship between a dependent variable Y and multiple independent variables collectively represented as matrix $\mathbf{X}$. This model allows for the prediction of Y considering a broader array of variables, providing a more comprehensive analysis of the factors influencing the dependent variable. The objective is to predict the value of the target variable considering a broader array of variables. More specifically, the MLR fits a line through a multidimensional cluster of data points. Furthermore, it can be utilized to understand the relationship among several variables. Regression modeling results can determine which variables influence the target variable or help explain the response [20]. Therefore, Regression can be used for both explanatory and predictive modeling purposes. Given a dataset of n observations denoted as $\{x_{i1}, x_{i2}, \ldots, x_{ip}\}_{i=1}^{n}$, the MLR model can be mathematically expressed as follows:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \varepsilon_i = X_i^{\mathsf{T}} \beta + \varepsilon_i, \quad i = 1, 2, \ldots, n \tag{2.4}$$

In this context, y_i denotes the continuous numeric response corresponding to the ith observation, while β_j represents the coefficients associated with the jth variable. x_{ij} signifies the jth variable for the ith observation and ε_i is referred to as the random noise not accounted for by the linear model. Multiple methods are available for obtaining parameter estimates in a model, such as Ordinary Least Squares (OLS) estimation. This technique aims to obtain the best estimates of regression coefficients β so that the MSE, which represents the sum of squared differences between observed values and predicted values divided by the number of observations, can be minimized. MSE is the most commonly used loss function in linear regression. By minimizing the MSE, the model aims to achieve the best fit to the data [20]. The OLS estimator of β is described below in formula 2.5:

$$\hat{\beta} = (X'X)^{-1} X'\mathbf{y} \tag{2.5}$$

The estimated regression coefficients are represented by $\hat{\beta}$ in vector form. The matrix $\mathbf{X}$ holds the values of independent variables, where each column corresponds to a variable, and each row corresponds to observations. The transpose of X is denoted by X'. The inverse of the matrix obtained by multiplying X and X' is written as $(X'X)^{-1}$, which is theoretically important in the estimation process. However, this expression is only valid when the matrix $(X'X)$ is non-singular and has full rank and a non-zero determinant. The vector $\mathbf{y}$ conveys the values of the response variable, which the model aims to predict [21]. Linear regression models rely on certain basic assumptions to ensure the validity of the model's results. The model needs a linear relationship between the independent and dependent variables. If the relationship is non-linear, the model can not capture the underlying patterns of the data. Additionally, the model assumes that the differences between observed and predicted values have uniform variance across all levels of the independent variables, known as homoscedasticity. The linear regression model is a powerful statistical tool if the assumptions are met. The model's estimations and predictive capabilities could be compromised if these conditions are unmet. Therefore, a thorough preliminary data assessment is crucial to ensure these conditions are met [22]. In this study, the term 'linear regression (LR)' will specifically denote multiple linear regression, and will be used in all subsequent discussions and analyses.

2.3.2 Elastic Net Regression

Traditional LR often provides satisfactory results but may struggle with high-dimensional data, as explained in Chapter 2.2.1. As discussed in Chapter 2.3.1, MLR aims to predict the dependent variable Y based on the independent variables X. This method is effective when there are few predictors and the matrix $\mathbf{X}$ maintains full rank, ensuring linear independence among columns. However, challenges arise when the number of predictors exceeds the number of observations, leading to a potentially singular matrix $\mathbf{X}$. This situation makes it impossible to find a unique solution, rendering the model unreliable and sensitive to minor data or model changes. Elastic net regression addresses these issues by integrating LR with Lasso and Ridge regression, two prominent regularization techniques. Throughout this chapter, λ will consistently be used as the regularization parameter for various regularization techniques.

The Least Absolute Shrinkage and Selection Operator (Lasso), or L1 regularization, introduces a penalty term to the loss function to address challenges in high-dimensional data analysis [23]. The L1 penalty, which is the sum of the absolute values of the model's coefficients, helps reduce some coefficients to zero, thereby removing certain features from the model. This simplification not only improves model interpretability by balancing complexity with performance but also serves as an effective feature selection tool, particularly beneficial in contexts with high-dimensional datasets where some features may be irrelevant [23]. The mathematical representation of the L1 penalty term is given by:

$$\text{Loss}_{L1} = \text{MSE} + \lambda \sum |\mathbf{w}_i| \tag{2.6}$$

In this context, MSE represents the original loss function before adding the penalty term. The hyperparameter λ controls the intensity of the regularization, and w_i are the model's coefficients. As λ increases, stronger regularization is applied, pushing more coefficients toward zero. Using the absolute value in the penalty encourages minimizing the coefficients [23].

Ridge regression, or L2 regularization, applies a distinct penalty term compared to L1 regularization: the sum of the squares of the model coefficients [23]. Unlike L1, L2 regularization aims to shrink coefficients but does not reduce them to zero. This method helps simplify the model and prevent overfitting, which is crucial in high-dimensional datasets with potentially highly correlated features. By distributing influence more evenly across all features, L2 regularization mitigates the adverse effects of multicollinearity by penalizing large coefficients. The same regularization parameter used for Lasso will be applied here as well. The mathematical expression for the L2 penalty term is as follows:

$$\text{Loss}_{L2} = \text{MSE} + \lambda \sum w_i^2 \tag{2.7}$$

Elastic net regression combines the strengths of these two regularization techniques, making it efficient for handling high-dimensional data. The elastic net regression is mathematically similar to LR, with the main distinction in the objective function used for optimization. While LR seeks to minimize the basic loss function, elastic net regression expands this objective by incorporating two additional penalty terms, enhancing the model's ability to handle complex data scenarios. The elastic net regression can be mathematically described as:

$$\text{Loss} = \text{MSE} + \lambda \left[(1-\alpha)|\theta|^2 + \alpha|\theta|_1 \right]. \tag{2.8}$$

Here, λ continues to serve as the regularization parameter, θ is the coefficient vector affected by the L1 and L2 penalties, and α is a hyperparameter that controls the balance between the

contributions of L1 and L2 regularization. By tuning λ and α, the loss function can be optimized [24]. Thus, strategically balancing the penalties from both L1 and L2 regularization reduces model complexity and improves performance in high-dimensional datasets.

2.3.3 Decision Tree

Decision trees are powerful supervised learning algorithms for classification and regression tasks, known for their greedy construction approach. They can effectively capture non-linear relationships between input and target variables. A decision tree model breaks down the data into subsets by making decisions through a series of questions. The initial question, at the tree root, forms the basis for classifying the data. If a split results in only one label remaining, it forms a leaf node. However, if multiple labels persist, the model continues by creating new parent nodes and asking more questions. The resulting sub-nodes are referred to as child nodes. This process iterates until pure nodes are obtained, forming a tree-like structure [25]. During the training process, the algorithm learns which questions to ask to make accurate predictions. Decision tree regression follows the same principles as classification but differs in dividing the data into continuous intervals based on feature thresholds. Figure 4 illustrates an example of a decision tree.

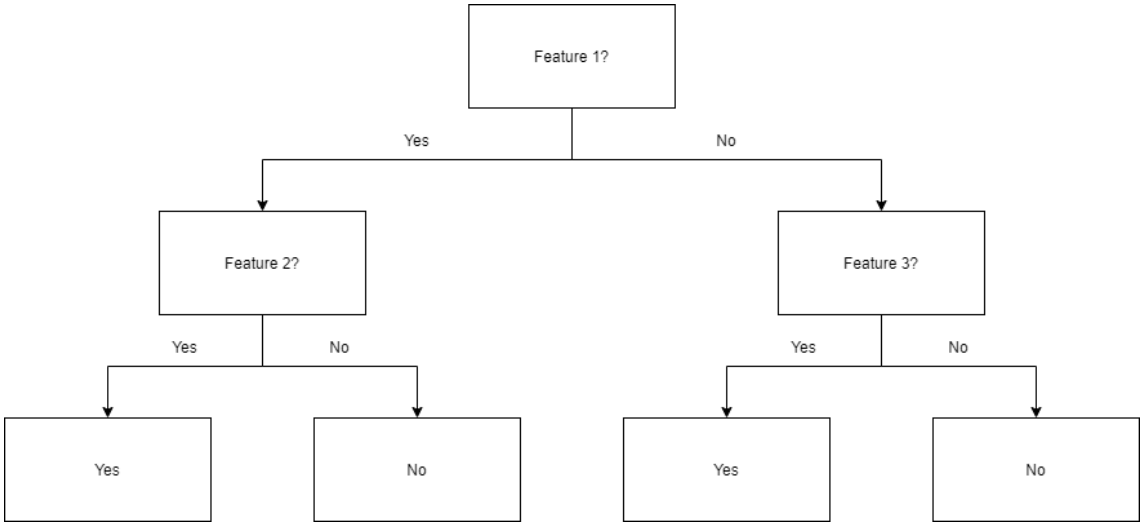


Figure 4: Decision tree example, adapted from an image by CollaborativeGeneticist available under a CC BY-SA 4.0 license via Wikimedia Commons. [26]

This greedy algorithm aims to split nodes at the most informative features. This ensures that the model consistently seeks conditions that maximize the information gain (IG) for effective data portioning. The IG is the objective function the decision tree algorithm seeks to optimize to decide which features to prioritize when constructing the tree. A higher IG is considered more helpful in making decisions and is typically chosen as the condition or splitting criteria. The IG is calculated based on the impurity measures. In the case of regression, specifically, the weighted MSE is used as the impurity measure for IG computation [27]. The features that have the impurity are the ones selected for splitting a node. After calculating the impurity measure for features at a given node, this value will be used to calculate the IG and determine which feature is more appropriate for making the split. The IG for a dataset split, based on a feature f, is defined by the following formula:

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \left(\frac{N_j}{N_p} \right) I(D_j) \tag{2.9}$$

IG in decision tree algorithms is calculated using Formula 2.9. The formula considers the Impurity measure (I), which evaluates how homogeneous the dataset is at a node, and the feature (f), which performs the split. The parent node's dataset is denoted as D_p, while D_j represents the dataset of the jth child node. The total number of samples at the parent node and the number of samples at each child node are represented by N_p and N_j, respectively [27].

2.3.4 Random Forest

Random Forest (RF) is an ensemble learning algorithm used for both classification and regression. It generates predictions based on a majority vote from numerous decision trees. It is called ensemble learning because it builds multiple models in parallel, in this case, many decision trees, to ensure diversity and robustness in its predictions. RF employs a feature bagging technique, randomly selecting features to formulate questions for splitting nodes, which ensures low correlation among decision trees. This technique makes sure that all the decision trees are not identical, and the questions asked for splitting nodes will vary. Additionally, it employs bootstrapping in randomly sampling subsets from a dataset for each decision tree. These newly constructed subsets contain observations from the original dataset and may contain observations multiple times or not at all. The newly constructed datasets are the same size as the original one and are used to train the decision trees [28]. When combined within the RF algorithm, these techniques result in the creation of multiple decision trees, each constructed from different subsets of the data through random sampling.

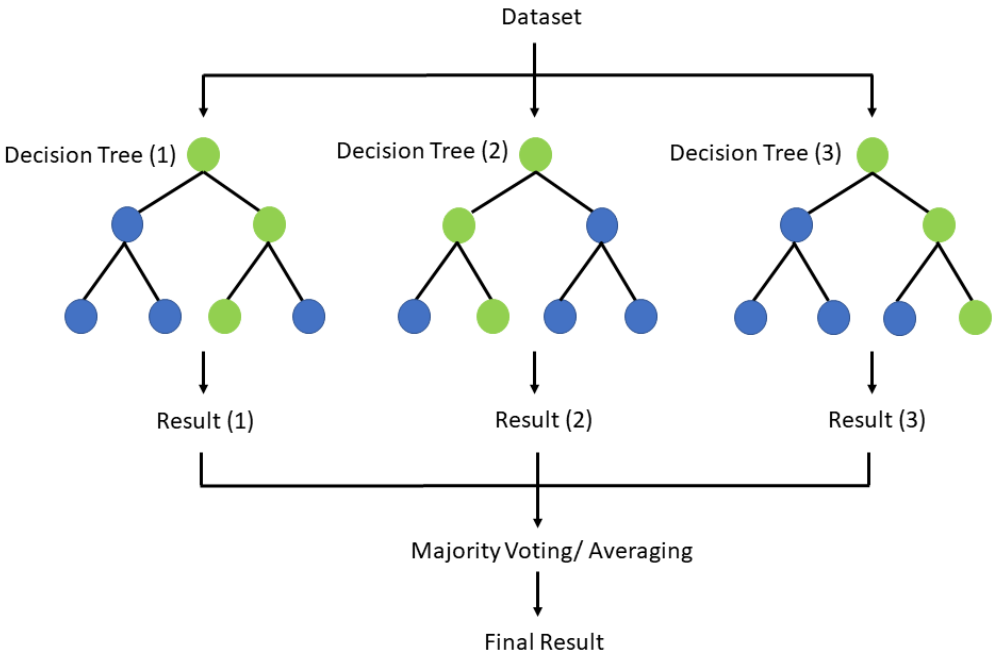


Figure 5: Explanation of the Random Forest algorithm, created by TseKiChun. Licensed under CC BY-SA 4.0 [29].

Figure 5 provides a graphical illustration of how RF operates. The process begins by drawing a random bootstrap sample of size n, where n observations are randomly selected with replacement

from the training set. Subsequently, a decision tree is grown using this bootstrap sample, and at each node, d features are randomly selected without replacement. The node is then split using the feature that optimizes the chosen objective function, such as maximizing the IG. This process is iterated multiple times to draw numerous decision trees from the original dataset. Finally, the predictions from each tree are aggregated, and the class label or continuous value is determined by a majority vote [30]. Due to the reduction in the variance of individual decision trees, RF can achieve high accuracy and effectively handle non-linear relationships. However, it can be computationally expensive and require substantial memory consumption [31].

2.3.5 Histogram-Based Gradient Boosting Regression

Histogram-based Gradient Boosting Regression (HGBR) is a ML algorithm specifically utilized for solving regression problems. This ML technique is part of the gradient boosting family of algorithms, an ensemble technique that builds multiple weak learners using decision trees for better prediction. It creates models sequentially, and each new model corrects errors made by the previous one [32]. The objective of gradient boosting is to minimize a loss function by adding multiple decision trees. For each iteration, the residuals or gradients of the loss function with respect to the current model's predictions are calculated. This gradient represents the direction to reduce the loss and find the global minima. The algorithm uses histograms for faster training and lower memory usage than traditional gradient-boosting algorithms. These types of models divide the continuous feature values into bins. The binning process helps to approximate the underlying data distribution and effectively reduces the computational complexity of finding optimal split points manually [33]. Another advantage of this algorithm is its native ability to handle missing values without the need for data preprocessing. During training, it determines whether samples with missing values should go left or right at each split point based on potential gains. For predictions, it assigns samples with missing values accordingly. If a feature encountered no missing values during training, samples with missing values are automatically directed to the child with the majority of samples [34]. HGBR is generally faster and often more accurate than RF on complex datasets, as it builds each tree sequentially, continually adjusting for errors identified by the previous trees.

2.3.6 Principal Component Analysis

Principal component analysis (PCA) is an unsupervised ML technique used to find patterns in a dataset by reducing the dimensionality of large datasets [35]. The primary objective of PCA is to transform the original dataset into new variables, known as the principal components (PCs), which capture the maximum variance in the original data. These PCs are orthogonal, meaning they are statistically uncorrelated or independent. This orthogonality ensures that each PC contributes uniquely to explaining the variance in the dataset without redundancy. The PC1 is calculated to capture the maximum variance in the dataset. This is achieved by identifying a line (or, in higher dimensions, a hyperplane) that passes through the origin. The data points are then projected onto this line, and the direction of this line is determined so that the sum of squared distances from the projected points to the origin is maximized. This process is effectively repeated to find the best fitting line through the multidimensional data, which becomes the PC1. The subsequent PCs are identified by finding lines (or hyperplanes in higher dimensions) perpendicular to the previous component [36]. An example of visualizing the first two PCs of a synthetic dataset is illustrated in Figure 6 below.

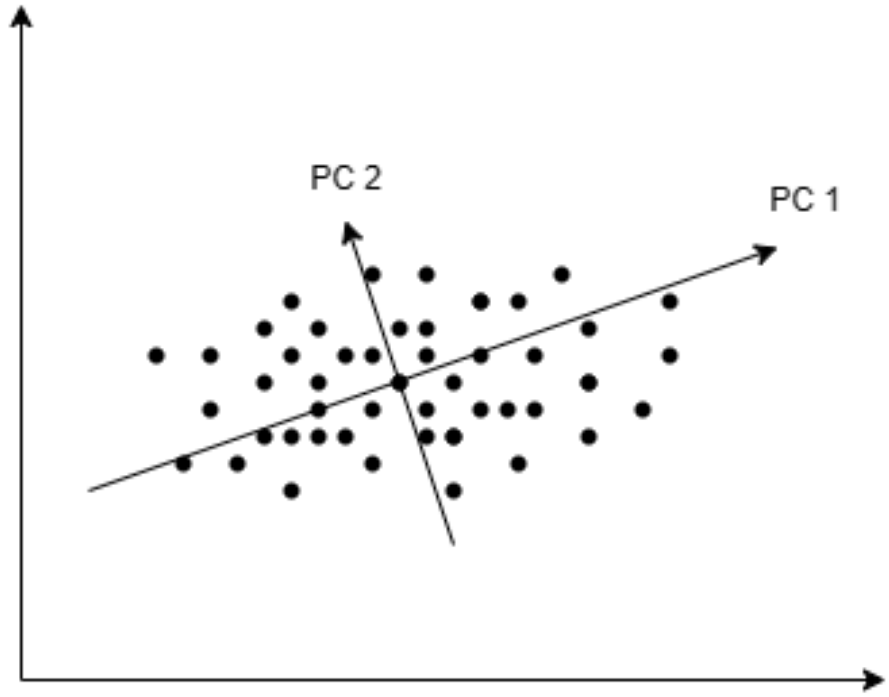


Figure 6: An example of visualizing the projection of dataset points along the first PC1 and PC2.

The proportions of each feature in a PC are known as *loadings*. The loadings inform us how the original features combine to form the PC. Essentially, a PC is a linear combination of the original features in a dataset, and the loadings are the coefficients in this linear combination. Given a set of predictors as $\{X_1, X_2, \ldots, X_p\}$, the PC1 can be formulated as a linear combination of these predictors:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \phi_{31}X_3 + \ldots + \phi_{p1}X_p, \tag{2.10}$$

where Z_1 represents the PC1, and ϕ_{j1} denotes the loading (coefficient) of the j^{th} feature in the PC1. These loadings $(\phi_{11}, \phi_{21}, \phi_{31}, \ldots, \phi_{p1})$ illustrate the contribution of each original feature to the PC, thereby indicating the structure and dimensionality of the dataset in the context of PCA [37]. The process of applying PCA results in the projection of data onto the PCs, offering a graphical representation that lays out the data along the axes defined by these components. This simplification allows for a clear view of the structure of the data. Such plots can be used to uncover hidden patterns and insights [37].

Score plot

A score plot displays these PCs on a graph, with each axis representing a PC. Each point on the score plot represents an observation from the original dataset projected onto these PCs. The color gradient represents the magnitude of the target variable. This allows us to visualize the observations in a reduced dimensional space defined by the PCs. Points that are closer together correspond to observations that have similar scores on the components [38]. By examining a score plot, we can identify patterns and explore the similarities and differences among the data points and how they are grouped or clustered along the PC axis. Figure 7 presents a score plot from PCA generated from a synthetic dataset. This plot visually illustrates the distribution of the dataset along the PCs identified by PCA.

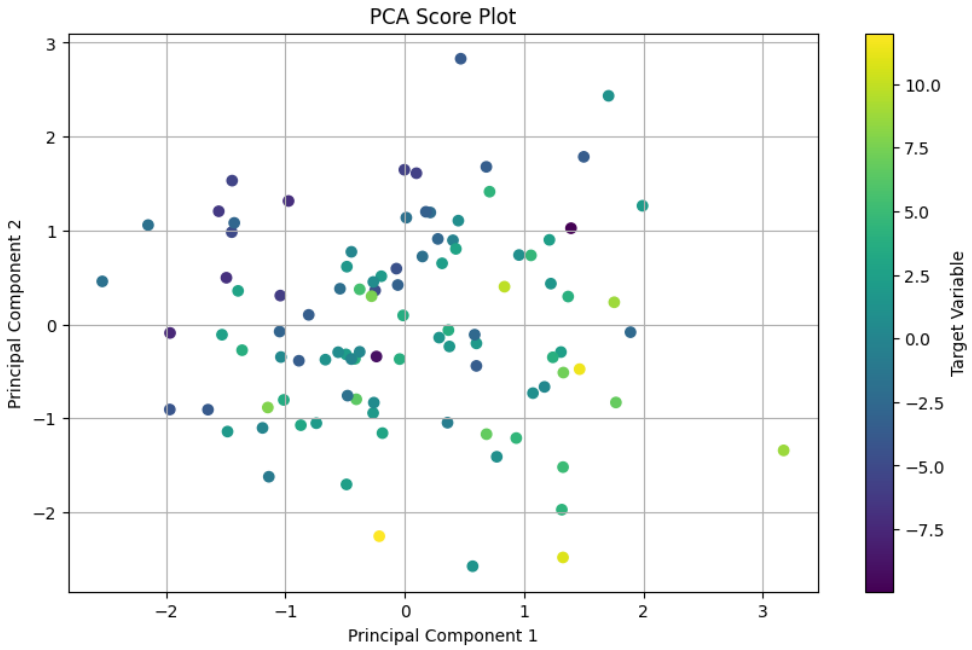


Figure 7: This figure illustrates a score plot from PCA generated from displaying the distribution of data points in terms of the first two PCs.

Loading plot

A loading plot is a visual representation of the loadings in PCA, which can help identify the most influential variables for each PC and determine how they are correlated or inversely correlated [39]. A loading plot displays each PC's loadings in terms of both direction and magnitude, illustrating how strongly each original variable influences a PC. Figure 8 shows a loading plot where each arrow represents an original variable, plotted based on its loadings in the PCs. The direction and length of the arrows show how each feature influences the PCs. The position of a variable on this plot can indicate its correlation with the PCs. Variables that are close together tend to be highly correlated, those that are far apart are likely negatively correlated or not correlated at all, and variables close to the origin have a lower contribution to the PC, whereas those far from the origin have a higher contribution. From the loading plot, we can observe that Feature 2, Feature 3, and Feature 4 strongly influence PC1, indicating their significant contribution to the variance along this component. Conversely, Feature 1 and Feature 5 demonstrate a stronger influence on PC2, signifying its substantial influence on the variance captured by the second PC. The smaller angle between the vectors of Feature 1 and Feature 2 suggests that they are positively correlated. In contrast, Feature 4 appears to be less or not correlated with both Feature 1 and Feature 2. This arrangement of features within the loading plot provides valuable insights into the underlying structure of the data.

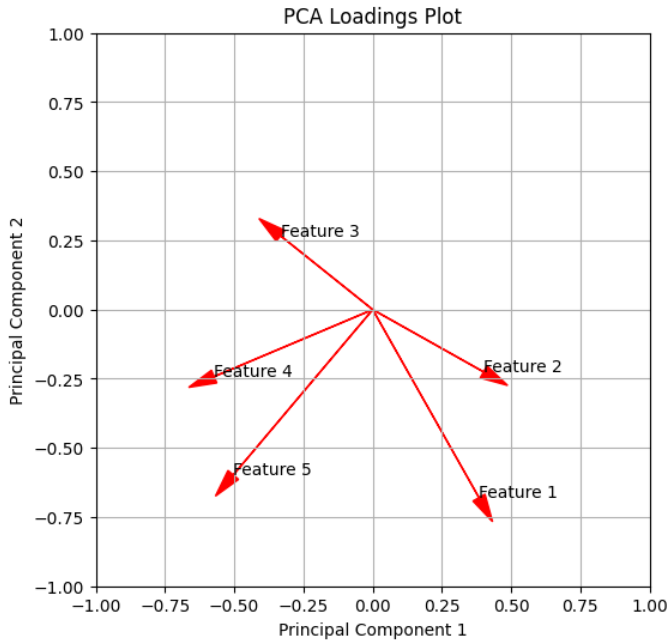


Figure 8: This PCA loading plot generated from synthetic data, illustrating how the features contribute to the first two principal components.

Cumulative Variance Explained Plot

A cumulative variance explained plot graphically represents the proportion of the dataset's variance explained by each PC. One of the initial steps in interpreting PCA results is to assess how much total variance is captured by each component [40]. This analysis aids in determining the number of PCs to retain for further examination. To identify the optimal number of PCs, one should look for a point where adding additional components does not contribute significantly to the explained variance. In Figure 9, three PCs are shown, collectively explaining 64.09% of the dataset's variance.

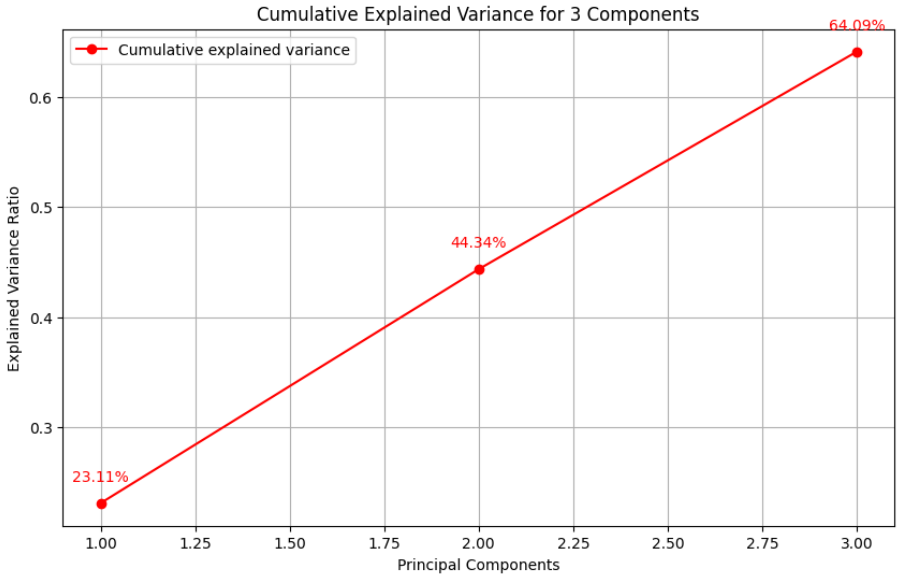


Figure 9: PCA Cumulative variance plot created from synthetic data, depicting the individual and cumulative explained variance by the first three PCs.

2.3.7 Principal Component Regression

While PCA offers valuable tools for dimensionality reduction, it specifically does not utilize the response variable in its analysis. Principal Component Regression (PCR) is a regression technique that combines PCA features and MLR. The process starts, as discussed in the section above, by using PCA to reduce the dimensionality of the dataset and transform the original predictors into a set of PCs. These PCs will be used as predictors, and MLR will be applied to predict the response variable [41]. However, this approach may not capture the full predictive potential of the dataset, especially in cases where understanding the direct relationship between predictors and the response variable is critical for accurate predictions. In the subsequent chapter, we delve into a technique that incorporates the response variable directly into its dimension reduction process, thereby offering a more targeted approach to modeling complex data scenarios.

2.3.8 Partial Least Squares Regression

Partial Least Squares Regression (PLSR) offers a robust solution to the challenges of high-dimensional data and multicollinearity. PLSR transforms the original predictors into a smaller set of uncorrelated components, which are derived through linear combinations of the original predictors. Unlike PCA and PCR, which only focus on the variance within the predictors, PLSR aims to maximize the covariance between the predictors and the response variables [42]. This technique can, therefore, be said to derive components in a more effective way for prediction than PCR by accounting for the covariance between $\mathbf{X}$ and $\mathbf{y}$. PLSR can effectively reduce the dimensionality while preserving the predictive power by focusing on the covariance.

PLSR visualizes the relationship between predictors and the response variable, producing plots that, while similar to those generated by PCA, serve a distinct purpose. Unlike PCA plots, which are designed to solely account for the variance within the predictor variables, PLSR plots aim to explain not only the variance in the predictors but also to maximize the covariance between $\mathbf{X}$ and $\mathbf{y}$ [43]. The interpretation of these plots is mostly the same as that of PCA plots, but with the response now taken into account. This makes PLSR particularly valuable in a predictive analysis context, as it offers deep insights into the dynamics between predictors and the response.

2.4 Feature Selection Techniques

2.4.1 Permutation Importance

Feature importance is a significant part of any model building and evaluation. Not all variables influence the model, so some techniques can be used to select the most important variables in the dataset. There are several ways to estimate feature importance from models [44]. Permutation importance is a technique used to determine the importance of features in ML models. This method involves individually shuffling the values of each feature in the dataset and observing the impact on the model's performance. If shuffling the feature values greatly reduces the model's accuracy, the feature is essential for the model's predictions[45]. It is important to note that permutation importance is calculated before model training, where features are permuted before training to assess their importance.

2.4.2 Sequential Feature Selection

Sequential Feature Selection (SFS) is a feature selection technique designed to enhance model performance by iteratively adding or removing features from a dataset. Supported by Scikit-learn, forward and backward selection methods are available [46]. Forward selection involves incrementally adding features one at a time until predefined criteria are fulfilled. Conversely, backward selection starts with the complete set of variables and systematically removes them individually until those criteria are met. These criteria often involve achieving a specified number of features or reaching a target performance score. SFS seeks to identify optimal feature subsets that maximize the model's performance score. This technique can be applied to many ML algorithms, serving as an estimator. However, it's important to know that SFS chooses the best feature at each step rather than considering all possible feature combinations. This approach can lead to suboptimal results since it focuses on short term gains and may overlook better-performing feature combinations.

2.4.3 Variable Importance in Projection

Variable Importance in Projection (VIP) scores are used in PLSR to identify the variables that most significantly explain the variance in the response. A VIP score is calculated for each predictor and represents its contribution to the model. This score can help determine which variables contribute most to explaining the response variable. The VIP score for each variable, denoted as j, is calculated based on the importance reflected by the a-th components and can be mathematically described as:

$$VIP_j = \sqrt{p \cdot \sum_{a=1}^{A} \left(SS_a \cdot \left(\frac{w_{aj}}{\|w_a\|} \right)^2 \right) \Big/ \sum_{a=1}^{A} SS_a} \tag{2.11}$$

The symbol p denotes the total number of predictor variables in the model. The optimal number of components determined through CV is denoted by A. SS_a represents the sum of squares explained by a specific component, indicating its variance contribution. w_{aj} signifies the standardized influence of the j-th predictor variable on the a-th component, obtained by dividing it by the Euclidean norm of w_a. This reveals how each predictor variable contributes to the standardized components, highlighting their relative influence. Squaring this normalized weight emphasizes each variable's importance [47]. A VIP score exceeding one is often used to identify crucial variables, with the threshold adjustable based on the application. Removing variables that contribute less to the model can mitigate overfitting and enhance model performance.

2.4.4 Repeated Elastic Net Technique

RENT is a feature selection method for binary classification and regression problems [48]. Since many studies aimed to investigate predictive power, RENT was made to investigate stability in the context of the feature selection process. RENT was benchmarked against six established feature selectors on eight multivariate datasets for binary classification and regression and shows a well-balanced trade-off between predictive performance and stability [48].

The methodology employs a training dataset X_{train}, composed of n_{train} observations, each within a p-dimensional feature space. We then train an ensemble of linear models, each enhanced with elastic net regularization. Specifically, each model is trained on a distinct subset of rows from X_{train}. These subsets are selected using repeated stratified K-fold CV, a method chosen to ensure that the subsets are both independent and identically distributed (i.i.d.), enhancing the representativeness and reducing bias and variance in model evaluation [48]. Each model in the ensemble, denoted as M_k where $k = 1, \ldots, K$, is assessed using a separate validation set. The weight distribution across all models is represented by $\beta_{k,j}$, where $j = 1, \ldots, p$. The weights from each model are aggregated into a matrix $\mathbf{B}$ with dimensions $K \times p$. This matrix helps us evaluate how important each feature is by looking at the weight vectors associated with all models. By examining these weight vectors, we can easily compare the significance of each feature [48]. The feature selection technique and the importance of each feature in the models are visualized in Figure 10, effectively illustrating how the methodology leverages the ensemble approach to refine predictions and feature relevance.

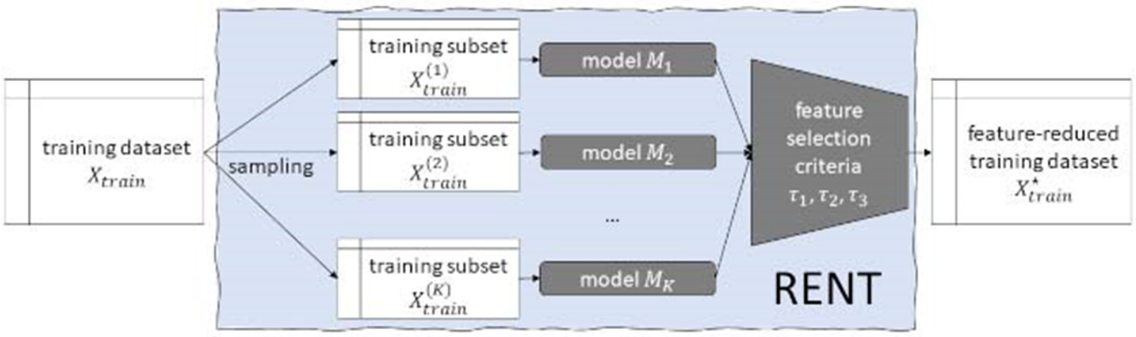


Figure 10: This figure illustrates the comprehensive framework of the RENT, showcasing its iterative process and key components [48].

We can then apply three criteria to select features, aiming to identify those with high stability in selection from the high-dimensional data. This ensures that the chosen features contribute to maintaining the predictive performance of the models, thereby enabling us to isolate the most impactful features for our analysis. The three criteria for selecting features are as follows: (1) Firstly, we assess the frequency of a feature selection across models. (2) Secondly, we examine how much a feature's weight fluctuates between positive and negative values. (3) Lastly, we evaluate whether the feature weights significantly differ from zero. To choose the features using the criteria, we introduce corresponding cutoff values $t_1, t_2, t_3 \in [0, 1]$. A feature is added to the selected feature if it satisfies all three criteria: $\tau_i \geq t_i, \ \forall I \ \in \{1, 2, 3\}$, ensuring a comprehensive

and informed approach to identifying the most relevant features for our analysis [48]. The criteria are mathematically formulated as follows:

$$1)\ \ \tau_1(\beta_n) = \frac{1}{K}\sum_{k=1}^{K} 1[\beta_{k,n} \neq 0] \tag{2.12}$$

$$2)\ \ \tau_2(\beta_n) = \frac{1}{K}\left|\sum_{k=1}^{K} \text{sign}(\beta_{k,n})\right| \tag{2.13}$$

$$3)\ \ \tau_3(\beta_n) = t_{K-1}\left(\frac{|\mu(\beta_n)|}{\sqrt{\frac{\sigma^2(\beta_n)}{K}}}\right) \tag{2.14}$$

The first criterion, $\tau_1(\beta_n)$ represents it measures the proportion of models in which the coefficient for feature n is non-zero, indicating the frequency of selection across different models [48]. The second criterion, $\tau_2(\beta_n)$, aggregates the signs of the coefficients by summing them up, taking the absolute value of the sum, and then dividing by the number of models K. This method captures how consistently the sign of a coefficient is either positive or negative across different models. Values near zero indicate that the sign of β_n is nearly random across different models, while values closer to one suggest that the sign is consistently the same, which implies a stable feature across the models. This measure provides an important indicator of the reliability of a feature's effect in predictive modeling [48]. The third criterion, $\tau_3(\beta_n)$ evaluates the statistical significance of β_n using a t-distribution with $k-1$ degrees of freedom. This process involves calculating a t-statistic for the standardized mean of β_n, which is adjusted by its variance $\sigma^2(\beta_n)$ and the total number of models k. The resulting t-statistic is then compared against the t-distribution, from which a p-value or a similar statistic is derived to indicate the significance of the feature's effect across different models [48]. All three criteria are bounded by the interval [0,1]. These quality metrics are hyperparameters for the RENT method, allowing the user to influence the selected features by tuning the thresholds t_1, t_2, t_3. Given that $\tau_3(\beta_n)$ is evaluated using a t-distribution, the threshold values t_3 for significance levels of 5% and 1% are set at $t_3 = 0.95$ and $t_3 = 0.99$, respectively [48].

RENT Hyperparameter Selection

RENT employs a two-step hyperparameter estimation process that begins with a grid search to identify the optimal regularization parameters (α and λ) using the Bayesian Information Criterion (BIC). BIC seeks a balance between predictive performance and model complexity, favoring models that achieve high accuracy with fewer features through stronger penalization. The formula for BIC is written as:

$$\text{BIC} = -2\log(\hat{L}) + X_{\text{train}}\log(\rho), \tag{2.15}$$

where $\hat{L}$ denotes the estimated likelihood of the predictive model, and ρ represents the number of estimated model parameters [48]. The estimated likelihood $\hat{L}$ refers to the probability of observing the given data under the assumed model parameters, quantifying how well the model fits the data [48]. By minimizing BIC, models with high information and low complexity are favored. The second step involves training the RENT ensemble with the optimal regularization parameter. Another grid search will then be used to search for the best cutoff values t_1, t_2, and t_3, which are used to decide the final selected features. This is not available for regression problems in RENT, so we will choose the cutoff values manually.

Validation study

Two validation study setups are applied to ensure the validity of the features selected with RENT. This validation study outlines a general method applicable to various feature selection methods, not just to RENT. However, it is included in the RENT package, facilitating easy demonstration and practical application within this specific framework. In Validation Study 1 (VS1), we draw randomly selected features, representing inefficient feature selections. We then train LR models on these features and predict them on an unseen test dataset denoted as X_{test}, comparing R^2 scores to predictions based on features selected by RENT. In Validation Study 2 (VS2), we compare the predictive performance of a model based on features selected with RENT on the real X_{test} labels to the predictive performance of models with l randomly permuted labels of X_{test}. The comparisons are performed using Student's T-tests, where the null hypothesis claims that the R^2 of RENT is lower or equal to the R^2 obtained from VS1 and VS2, respectively [48]. The tests are conducted at a significance level of 0.05. The P-values are used to determine if the null hypothesis can be rejected, thereby assessing the statistical significance of the results. The heuristic P-value serves as an additional metric, indicating how frequently random scores surpass the actual score achieved by the model [48]. The results from the validation study are a reliable indicator of whether the model based on features selected by RENT performs better than models based on randomness. Figure 11 depicts an example of a validation study result, where the X-axis represents the R^2 score, and the Y-axis represents the density. We can observe that the RENT model generally performs better than the models with randomly selected features and better than the models with permuted labels. The density plot in the image illustrates the distribution of the RENT prediction scores, with two peaks representing the densities of VS1 and VS2 scores.

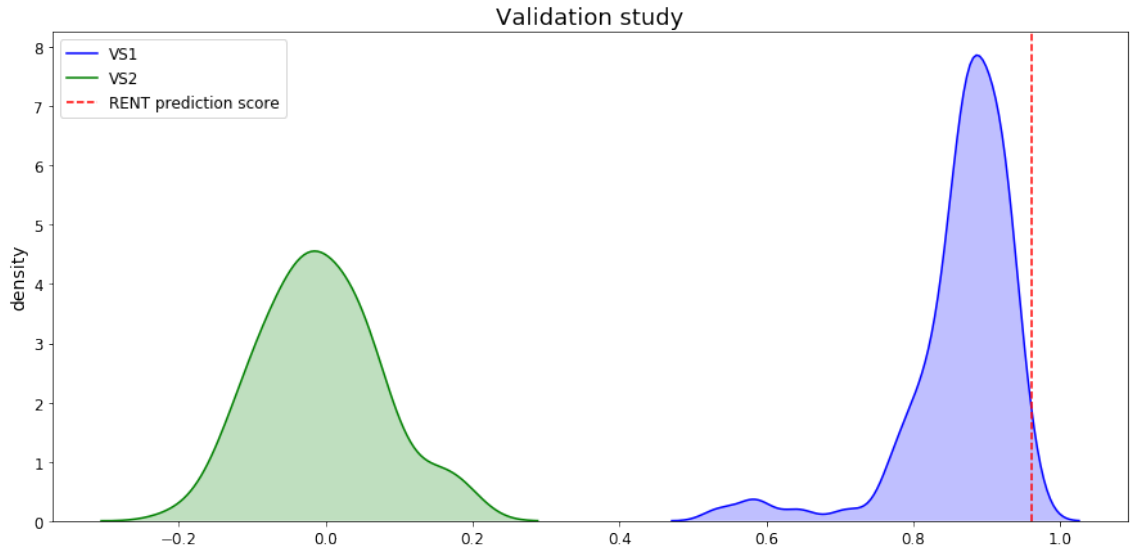


Figure 11: An example of a validation study result showing the RENT prediction score (R^2) compared to random feature selection (VS1) and permuted labels (VS2) [48].

2.5 Optimization

Optimization can be easily explained as selecting the correct inputs for a model to achieve the best possible outcome. By optimizing, we can improve the quality of the solution. Optimization is used in numerous real-world applications today for decision-making and strategic planning, ranging from allocating optimal resources to finding the best investments that will maximize profit. Optimization problems may include constraints or limits that act as rules that the optimization must follow. This will create boundaries restricting possible solutions [49]. The optimization process is mainly about maximizing or minimizing an objective function, representing the quantifiable value you are trying to optimize. The objective function should reflect the problems to be solved and the goals to be achieved. The minimization or maximization of an objective function f, the most beneficial solution within a defined set of feasible options, can be identified. The decision variables are the values the optimizer can change to improve the quality of the solution [49]. The more decision variables there are, the more complex the optimization problem will become. Consider the problem of identifying the maximum value of a function $f(x)$. The goal is to find the vector of optimal values represented as x^* that will result in the maximization of $f(x)$. This can be formulated as:

$$\begin{aligned} \text{maximize} \quad & f(x_1, x_2, x_3, x_4, \ldots, x_n) \\ \text{subject to:} \quad & g_1(x_1, x_2) \leq b_1, \\ & g_2(x_1, x_2) = b_2, \\ & x_1, x_2, \ldots, x_n \geq 0. \end{aligned}$$

Here, $f(x)$ denotes the objective function, g_1 and g_2 represent the constraints, and b_1 and b_2 are the boundaries of these constraints. x represents the decision variable used to find the optimal solution for the problem. This optimization problem requires finding the optimal points x^* such that $f(x^*) \geq f(x)$ for every x in the defined domain. Optimization algorithms are implemented to resolve most optimization problems, and like the machine learning algorithms described in Section 2.3, many optimization algorithms are used for different problems.

2.5.1 Linear and Non-Linear Programming

Optimization problems are usually divided into two categories: Linear and Non-linear Programming. Linear programming (LP) is used in optimization when the relationship between the decision variables, the constraints, and the objective function is linear. There are many Linear programming algorithms, such as the simplex method, the interior-point method, and the dual simplex algorithm [50]. Non-linear programming (NLP) is used when the relationship between the decision variables, the constraints, and the objective function is non-linear. NLP problems can be more challenging than LP and may require different algorithms, such as the gradient method, Newton method, or differential evolution [51].

2.5.2 Differential Evolution

Differential evolution (DE) is a powerful optimization algorithm for solving complex non-linear real-world optimization problems [52]. Unlike some optimization algorithms mentioned earlier, this method does not require gradient information, meaning the optimization problem does not need to be differentiable [53]. This algorithm was developed by Rainer Storn and Kenneth Price in the 1990s and is recognized today for its simplicity, efficiency, and ability to handle noisy

objective functions [53]. Although many packages offer implementation of the DE algorithm, we will discuss the version available in SciPy in this theoretical discussion.

DE is categorized as a genetic algorithm, a search technique used in optimization based on an evolutionary process. This algorithm iteratively tries to improve the candidate solution (individuals) by creating new solutions, combining and comparing multiple potential solutions to find the optimal one [53]. DE begins by randomly generating a population of potential solutions called the parent population, where each solution is represented as a vector of decision variables. It then uses mutation, crossover, and selection as part of the optimization process to create an offspring population or a solution. The main advantage of DE is that it has only three hyperparameters for the user to adjust. These include the population size, the mutation factor, and the recombination parameter. The population size significantly affects the algorithm's exploration ability [53]. In cases where the optimization problems have a large number of dimensions, the population size also needs to be significant to enable the algorithm to search in the multi-dimensional space and converge to an optimal solution. The mutation factor F is a positive hyperparameter in the range [0,2] used to scale and control the different vectors' amplification [54]. Small values of F can lead to smaller mutation step sizes, causing the algorithm to take longer to converge. Conversely, large values of F can lead to the algorithm overshooting good optima. The recombination parameter (RC) controls the diversity of the algorithm and should be in the range of [0,1]. Larger values of RC will increase the variation of the new population and the exploration [54]. Figure 12 illustrates a three-dimensional plot of a mathematical objective function. We also refer to this as a fitness landscape in the optimization context. In the DE algorithm, a fitness landscape like this would represent the search space where the algorithm tries to find the global optimum by creating optimal solutions from the candidate solutions.

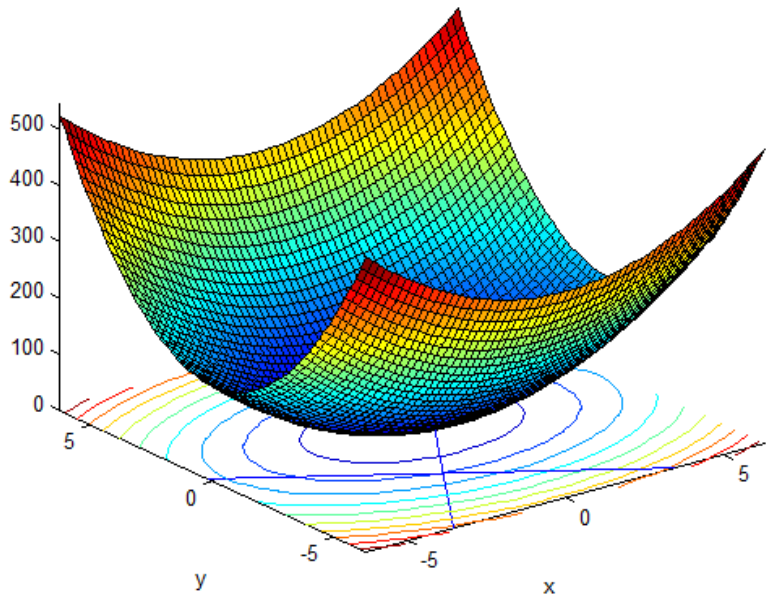


Figure 12: A 3D visualization of a non-linear function surface, illustrating the landscape explored during differential evolution optimization. Adapted from Andrebis (2010) [55].

2.6 Cross-Industry Standard Process for Data Mining

Cross-Industry Standard Process for Data mining (CRISP-DM) is an iterative process model that was created to make a structured and systematic procedure for planning, organizing, and executing data analysis projects [56]. Crisp-DM consists of six phases, and their relationship is illustrated in Figure 13. This model will potentially increase the efficiency of the data analysis project by providing a clear roadmap for the data mining process. This ensures the optimal allocation of resources and that each phase of the project, from data preparation to model evaluation, is executed precisely and aligned with the project's objectives.

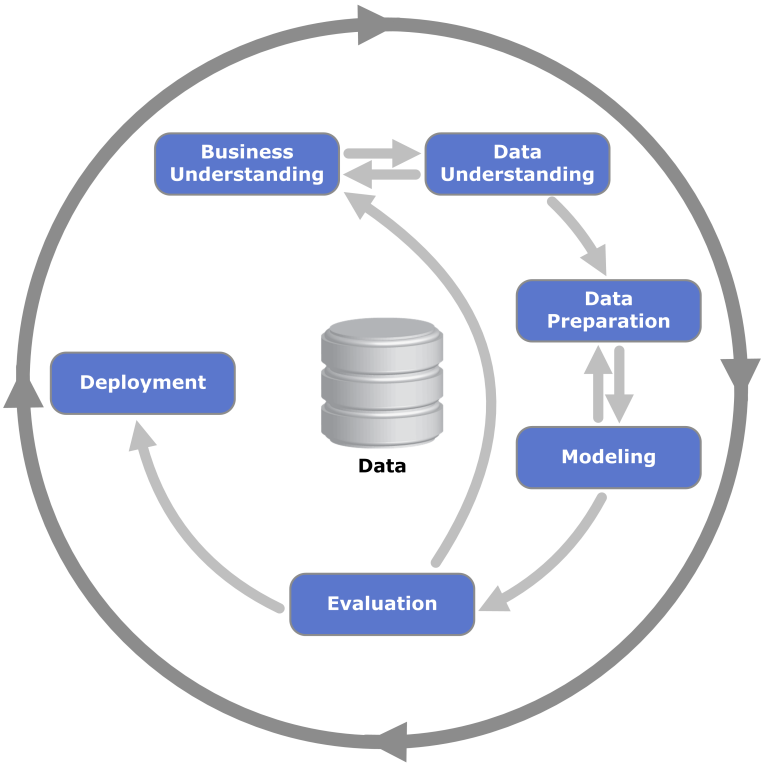


Figure 13: The figure illsutrates the CRISP-DM process model. Adapted from Kenneth Jensen, available under a CC BY-SA 3.0 license ([57]).

The Business understanding phase is the initial phase and focuses on understanding the objectives, requirements, and resources available from a business perspective. This means understanding the problem that the project aims to address. Success criteria are used to determine whether the project's goals have been met or if the process requires iteration. The phase following the evaluation determines whether these success criteria have been met. The second phase, data understanding, focuses on collecting the raw data from various sources and understanding the dataset's basic structures. This includes analyzing the data format, the number of records, and the identification of fields. The objective is to identify quality issues and interesting properties within the dataset. Techniques in data visualization can be effectively employed during this phase. Figure 13 illustrates the connection between the first and the second phases. The initial phase is closely linked to this phase, where insights from the data are required to define a research question and develop a project plan [56]. Data preparation is the third phase, where we prepare the final dataset for modeling. During this stage, the dataset undergoes extensive processing such as cleaning, handling missing values, transformation, removing irrelevant variables, and managing outliers. Additionally, the Construction of new variables may be a critical component in certain data science projects [56]. Modeling is the fourth phase and is

all about using the processed data to construct or refine one or more models. This involves selecting the appropriate model architecture, algorithms, and hyperparameters. The focus here is searching for the optimal architecture that aligns with the cleaned dataset and the predefined problem. This phase is connected to the data preparation because new insights about the data may emerge during the construction of the models. These insights can lead to further data processing, feature engineering, or even re-evaluating the choice of the model itself. Evaluation is the fifth phase, focusing on assessing and interpreting the performance of the models. The model selection should be thoroughly justified, and the results should be employed to determine whether the predefined success criteria have been met. Should these criteria remain unmet, a thorough review is necessary to identify potential oversights. Ultimately, a decision regarding the utilization of the results must be made. If the results are not optimal, it is essential to undertake steps to refine the process. This may include revisiting and refining earlier phases to ensure improved outcomes. The last phase is about developing a plan for deploying the model. The results and implementation should be intuitive for the users. The plan should outline simple, actionable steps for integrating the model with the current processes or systems. Additionally, it should include a strategy for ongoing support and maintenance of the model to ensure the model quality.

Chapter 3

Methodology

3.1 Methodology Overview

The methodology section consists of two parts. The first part focuses on understanding the relationship between the independent and the target variable, dry matter content. By modeling this relationship using ML algorithms, we aim to find the most influential variables affecting dry matter content and to develop predictive models. The second part involves taking the insights gained from the first part to design an optimization algorithm. This algorithm will be integrated into a user-friendly website accessible to the industry. The objective is to use this digital platform to reduce variability and achieve a more optimal dry matter content, thereby enhancing the quality of Norvegia Cheese. The aim is not only to improve product quality but also to improve decision-making and resource management.

The selected ML algorithms for the first part include HGBR, PLSR, and LR. The main feature selection techniques we will use to find an optimal feature set are SFS, VIP, and RENT, respectively. These ML algorithms and feature selection techniques will help us understand the relationship between the explanatory variables and the target variable, dry matter content. This strategic selection of algorithms is designed to leverage their unique strength and capabilities in uncovering meaningful patterns and for effective prediction and optimization. In the second part of our research, we will categorize the most influential variables identified in the first part into controllable and uncontrollable variables. Controllable variables can be adjusted during production, while uncontrollable variables are fixed values that cannot be changed. The optimal ML models from the first phase, which serves both predictive and optimization roles through differential evolution (DE), will be deployed on a user-friendly website. The integrated optimization model, designed for deployment, will adjust controllable variables based on real-time measurements taken on the production day. This is done by process operators entering different values for the uncontrollable variables. The website will then output optimal values for the controllable variables that can be used as a guide to achieve optimal dry matter content. However, there is a challenge with this website. Some uncontrollable variables are measured early in the process, while two critical variables necessary for predictive power are measured later, often after determining the controllable variables. This can make it challenging to utilize the website realistically, as there may be a need to update the controllable variable based on the uncontrollable variables. Therefore, our research assumes prior knowledge of the uncontrollable variables to address this issue, but we will also discuss potential methods to mitigate this problem.

3.2 Workflow

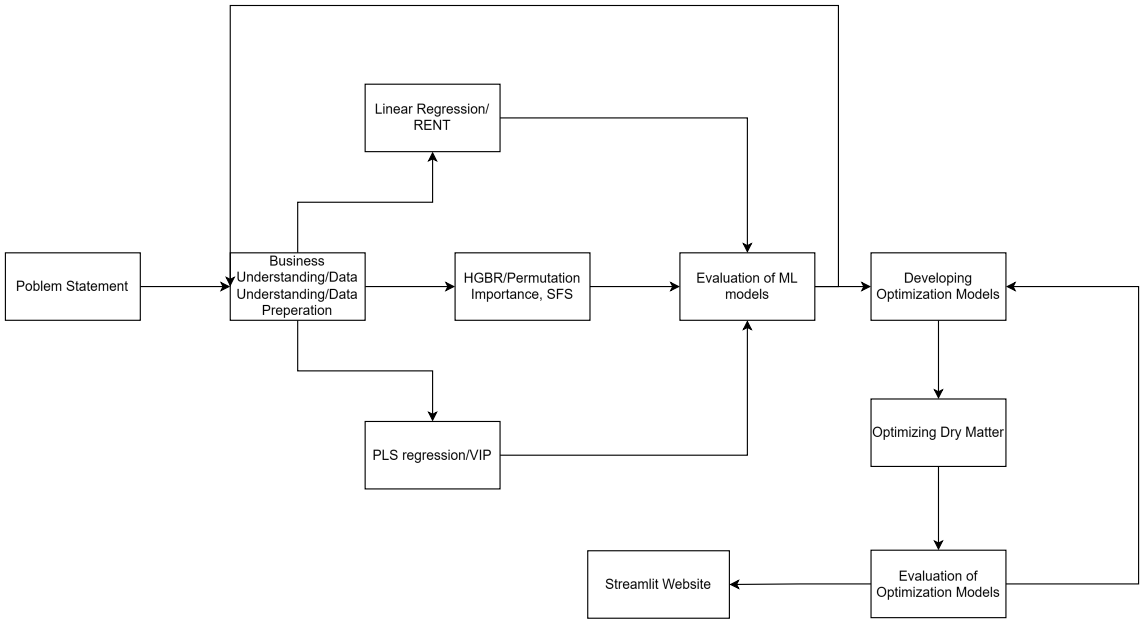


Figure 14: This figure illustrates the projects workflow described in a flowchart.

The workflow for the project is presented in Figure 14. The first step is to initiate the project by thoroughly understanding the problem that needs to be addressed. This involves analyzing the dataset provided by Tine Jæren, which is examined to determine several ML algorithms that can be applied to the problem. Specific feature selection techniques are chosen for each algorithm to ensure the best results. Once the algorithms and feature selection methods are finalized, the focus shifts to modeling. The selected models and corresponding feature selection techniques are applied to the dataset. The models are then evaluated to determine if they meet some success criteria defined in the business understanding. If the criteria are unmet, the process returns to the data understanding phase to refine the models. The steps in this workflow align with the CRISP-DM process, depicted in Figure 13. The insights from the best model selected during the modeling phase inform the design of an optimization model that employs DE combined with an optimal ML model. The optimization models are evaluated to determine if they meet the defined success criteria. If they do not, the models are refined until they meet the criteria. Finally, the best optimization model is integrated into the Streamlit website, making it accessible to stakeholders and decision-makers.

3.3 Hardware and Software

This project used Google Colab for model development, which provided 12 GB of RAM and GPU support. Python was the primary programming language, and it is widely used in data science and ML. Python's library made it easy to create ML models, with all necessary packages installed via Anaconda and PIP. Table 1 lists the Python packages used and their versions. The project also involved the utilization of Grammarly Premium and ChatGPT. ChatGPT facilitated code debugging and enabled the creation of code elements beyond the university curriculum. As a result, significant time savings were realized. Also, Grammarly Premium enhanced the writing in this master thesis. The outputs from both tools were cross-checked against available codes and online resources to ensure their proper functionalities. The project codes and accompanying figures are available on GitLab. The data itself and some codes contain sensitive information and are censored or not available on GitLab. Table 2 displays the latest version numbers of the Jupyter Notebook files available on GitLab at the time of writing this document. Access to the repository is provided through the following link:

Data_driven_cheese_optimization_spring2024

Package Name	Version	Purpose
Pandas	2.0.3	Data manipulation and analysis
NumPy	1.25.2	Fundamental numerical operations
Hoggorm	0.13.3	Graph plotting functionality
Hoggormplot	0.13.3	Visualization tool
SciPy	1.11.4	Advanced scientific computing and data manipulation. The optimization techniques are from this package
Optuna	3.6.1	A Hyperparameter optimization Framework
Scikit-learn	1.2.2	Machine learning toolkit
Matplotlib	3.7.1	Data visualization and graphical representation
Missingno	0.5.2	Data visualization aiding in missing data exploration
Streamlit	1.34.0	Development of interactive web applications
Seaborn	0.13.1	Statistical data visualization
mlxtend	0.22.0	Feature selection package

Table 1: Python packages utilized in this project.

File Name	Git Hash
Preprocessing	3731a728
Histogram_Based_Gradient_Boosting_Regression	2dc3f873
Partial_Least_Squares_Regression	dd18fd64
RENT	7e51f90
Model_Comparisons	113937d8
Optimization_Models	c1d788a8
Streamlit_website_py	3efe7a36

Table 2: Latest versions of Jupyter Notebook files available on GitLab at the time of writing.

3.4 CRISP-DM

3.4.1 Business Understanding

As highlighted in the introduction, the goal of this project is to use ML to predict dry matter content and optimize dry matter content within the production process using operational data (e.g., temperature, pH, time). The desired dry matter range is aimed to be between 57.6-57.7%. The development of predictive models will enable a deep understanding of the relationship between various predictors and dry matter content. This is fundamental for maintaining high-quality products such as Norvegia cheese. This section outlines the significance of accurately predicting and controlling the dry matter content, highlighting the impact on product quality and operational efficiency.

Dry matter content is a crucial determinant of product quality in the dairy industry, especially for cheese products like Norvegia. It impacts the product's physical and sensory properties and shelf life, directly influencing consumer satisfaction and compliance with regulatory standards. Controlling dry matter content is essential in maintaining product quality and consistency across numerous manufacturing processes. The absence of effective control over this aspect can lead to quality deviations, affecting the overall product value and possibly leading to high variations in dry matter content, increased production time, and customer dissatisfaction. By identifying key production parameters that significantly influence dry matter content, we can get better control over production parameters that will lead to optimal dry matter content, affecting product quality.

The economic benefits of effectively controlling dry matter content in production extend far beyond mere cost savings. It represents an important strategy to enhance operational efficiency and promote sustainability in the manufacturing industry. For a cooperative like Tine, owned by thousands of farmers, optimizing dry matter content represents a dedication to mutual prosperity and sustainable farming. By managing the dry matter content, companies like Tine can significantly reduce the overuse of resources, such as raw materials and energy, thereby minimizing waste generation at various stages of production. Enhancing production efficiency, achieved through the precise control of dry matter content, will result in a more streamlined manufacturing process, enabling faster production times while maintaining the quality of the final product Tine is known for. Furthermore, maintaining uniform product quality through optimized dry matter content helps businesses build more vital customer satisfaction and loyalty. This approach to optimizing dry matter content reduces operational costs and drives economic viability and sustainability, ultimately improving the company's overall financial health and competitive positioning. Table 3 displays the data goal and its corresponding success criteria.

Table 3: Data Goals and Success Criteria.

Data Goal	**Success Criteria**
1. Achieve good model performance:	- R^2 score > 0.50 for all models on test data.
2. Feature Selection Performance:	- Reduce feature set by $> 50\%$ for all models while maintaining performance.
3. Optimize Algorithm Effectiveness:	- Improve $> 80\%$ of dry matter content for all test data.

3.4.2 Data Understanding

To construct a predictive model and explore the factors contributing to significant variability in dry matter, we used a dataset provided by Tine Jæren. Such data is invaluable, offering a detailed view of the operational factors that determine the final characteristics of the cheese. To understand the data, we can investigate cheese production, from when the raw milk is received to the point where the fresh cheese is produced. By analyzing these variables, we could obtain insights that could lead to the optimization of the cheese production process, enhancing both efficiency and product quality. All the variables and their descriptions are listed in Appendix A.

The process of making cheese begins with the raw milk being stored in silos. At this stage, various tests are conducted to measure the fat content, protein levels, acidity (pH), and temperature of the milk. These measurements are prefixed with the label 'Si'. After that, the milk is pasteurized and standardized to eliminate bacteria and adjust the fat content. These measurements are prefixed with 'Pa'. Before the fermentation process begins, the milk's pH is adjusted by adding starter acid, which is recorded under the prefix 'Bs'. Then, a starter culture is added to the milk, which starts the fermentation process. During this process, rennet is added to the cheese vat, which coagulates the milk into curds and whey. The measurements taken during this phase are prefixed with 'Yk'. The curds are then transferred to a buffer tank, where they are held until they reach the desired pH level. The curds may then be further processed using a casomatic machine, and these measurements are prefixed with 'Bu'. The final product is fresh cheese, characterized by its soft texture and moisture content. The measurements at this stage include dry matter, fat, and pH, which are recorded under the prefix 'Ost'. Figure 15 illustrates the different steps involved in the process of making cheese.

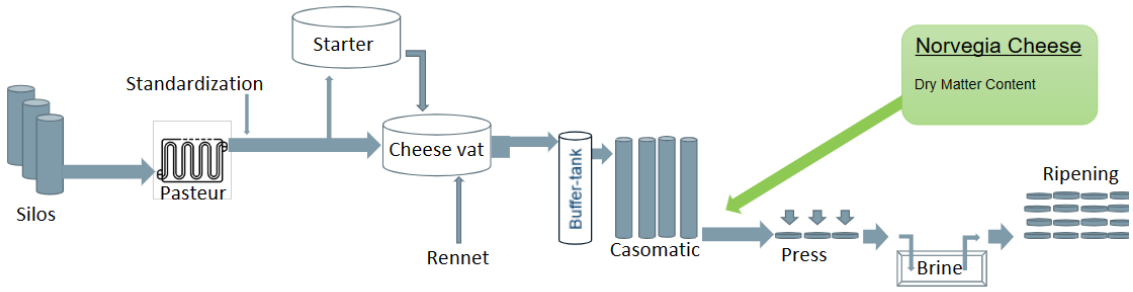
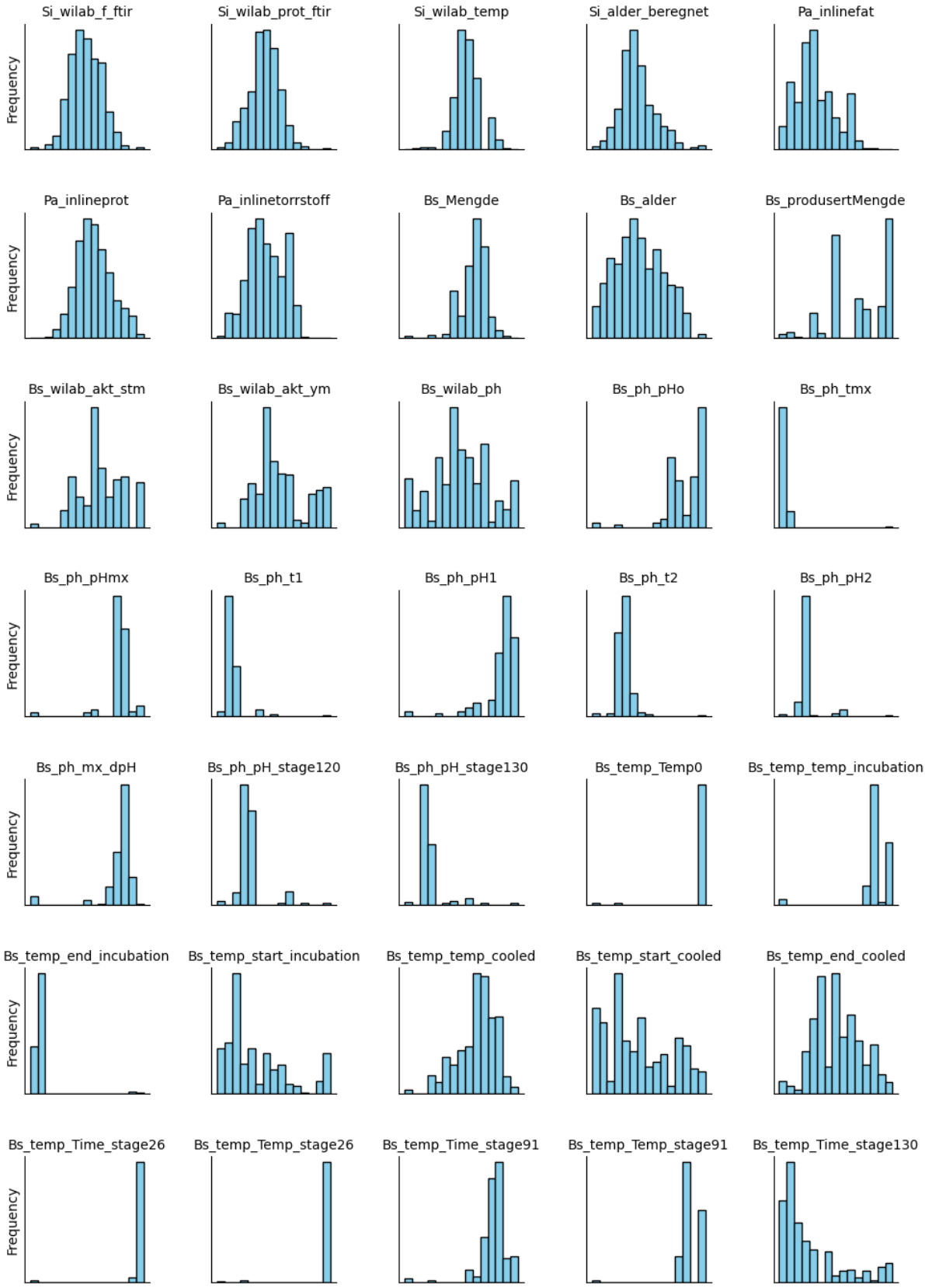
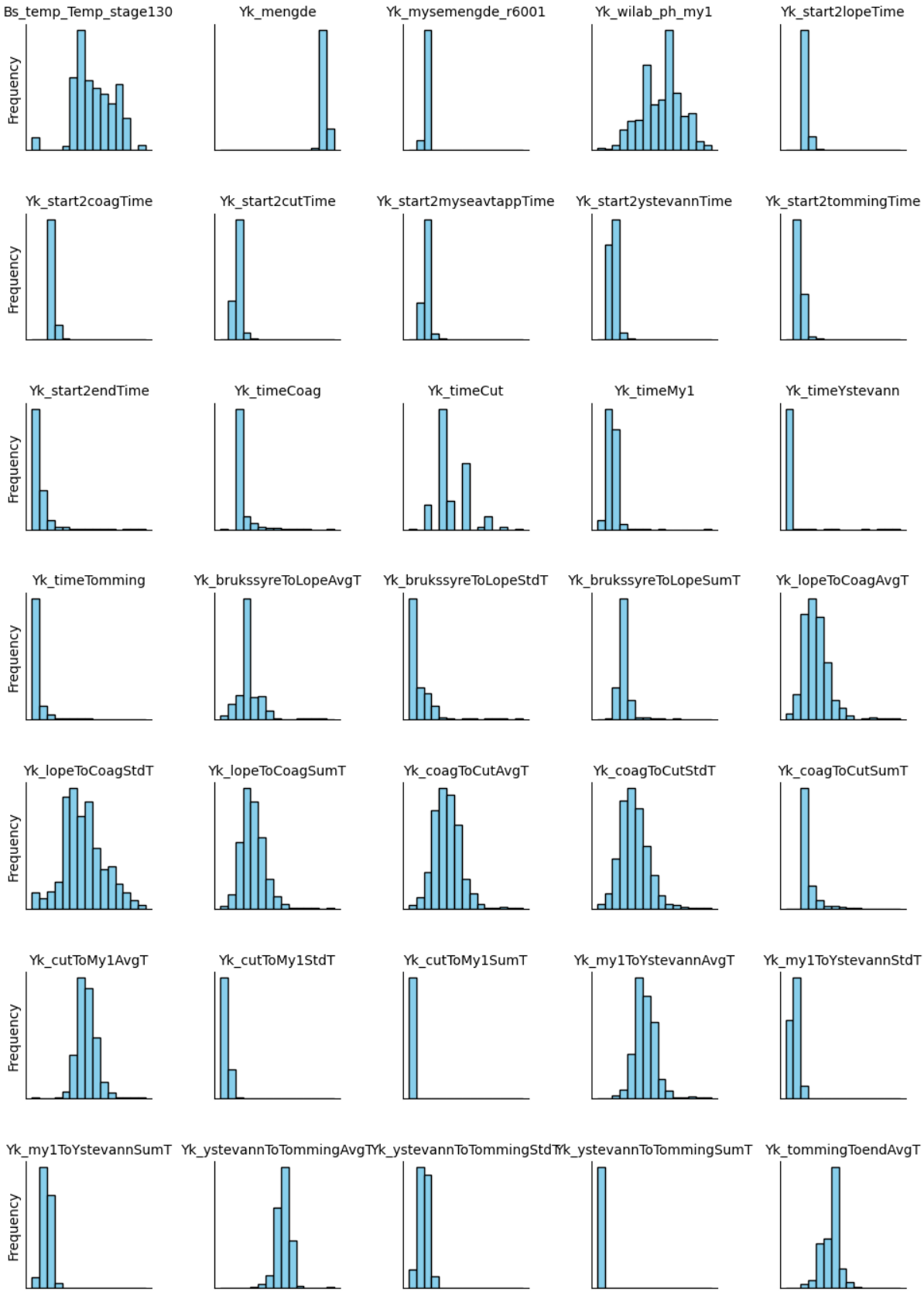


Figure 15: The image illustrates the key steps in making cheese, starting with raw milk and ending with the finished cheese product (Image credit: Lars Erik Solberg, Nofima).

Explorative Data Analysis

This section will cover the technical aspects of the data. Although the raw data has been transformed into a usable dataset, some additional preprocessing is still required. The dataset consists of 1,949 records and 103 features, including 13 string variables, one categorical variable, and the rest numerical. The string variables serve mainly to identify ID numbers throughout the process. We will keep all variables for the exploratory analysis, even though not all of them are important. As mentioned, the dataset captures information from multiple stages in the production chain. The dataset defines the target variable, dry matter content, as `Ost_inlineTS`. The distribution of the variables varies, with some having more spread-out data than others. Figure 16 shows histograms of all numerical variables, providing insight into the range of each variable. The value of the variables will be censored due to sensitive information.


Si_wilab_f_ftir
Si_wilab_prot_ftir
Si_wilab_temp
Si_alder_beregnet
Pa_inlinefat
Pa_inlineprot
Pa_inlinetorrstoff
Bs_Mengde
Bs_alder
Bs_produsertMengde
Bs_wilab_akt_stm
Bs_wilab_akt_ym
Bs_wilab_ph
Bs_ph_pHo
Bs_ph_tmx
Bs_ph_pHmx
Bs_ph_t1
Bs_ph_pH1
Bs_ph_t2
Bs_ph_pH2
Bs_ph_mx_dpH
Bs_ph_pH_stage120
Bs_ph_pH_stage130
Bs_temp_Temp0
Bs_temp_temp_incubation
Bs_temp_end_incubation
Bs_temp_start_incubation
Bs_temp_temp_cooled
Bs_temp_start_cooled
Bs_temp_end_cooled
Bs_temp_Time_stage26
Bs_temp_Temp_stage26
Bs_temp_Time_stage91
Bs_temp_Temp_stage91
Bs_temp_Time_stage130
Frequency


Bs_temp_Temp_stage130
Yk_mengde
Yk_mysemengde_r6001
Yk_wilab_ph_my1
Yk_start2lopeTime
Yk_start2coagTime
Yk_start2cutTime
Yk_start2myseavtappTime
Yk_start2ystevannTime
Yk_start2tommingTime
Yk_start2endTime
Yk_timeCoag
Yk_timeCut
Yk_timeMy1
Yk_timeYstevann
Yk_timeTomming
Yk_brukssyreToLopeAvgT
Yk_brukssyreToLopeStdT
Yk_brukssyreToLopeSumT
Yk_lopeToCoagAvgT
Yk_lopeToCoagStdT
Yk_lopeToCoagSumT
Yk_coagToCutAvgT
Yk_coagToCutStdT
Yk_coagToCutSumT
Yk_cutToMy1AvgT
Yk_cutToMy1StdT
Yk_cutToMy1SumT
Yk_my1ToYstevannAvgT
Yk_my1ToYstevannStdT
Yk_my1ToYstevannSumT
Yk_ystevannToTommingAvgT
Yk_ystevannToTommingStdT
Yk_ystevannToTommingSumT
Yk_tommingToendAvgT
Frequency

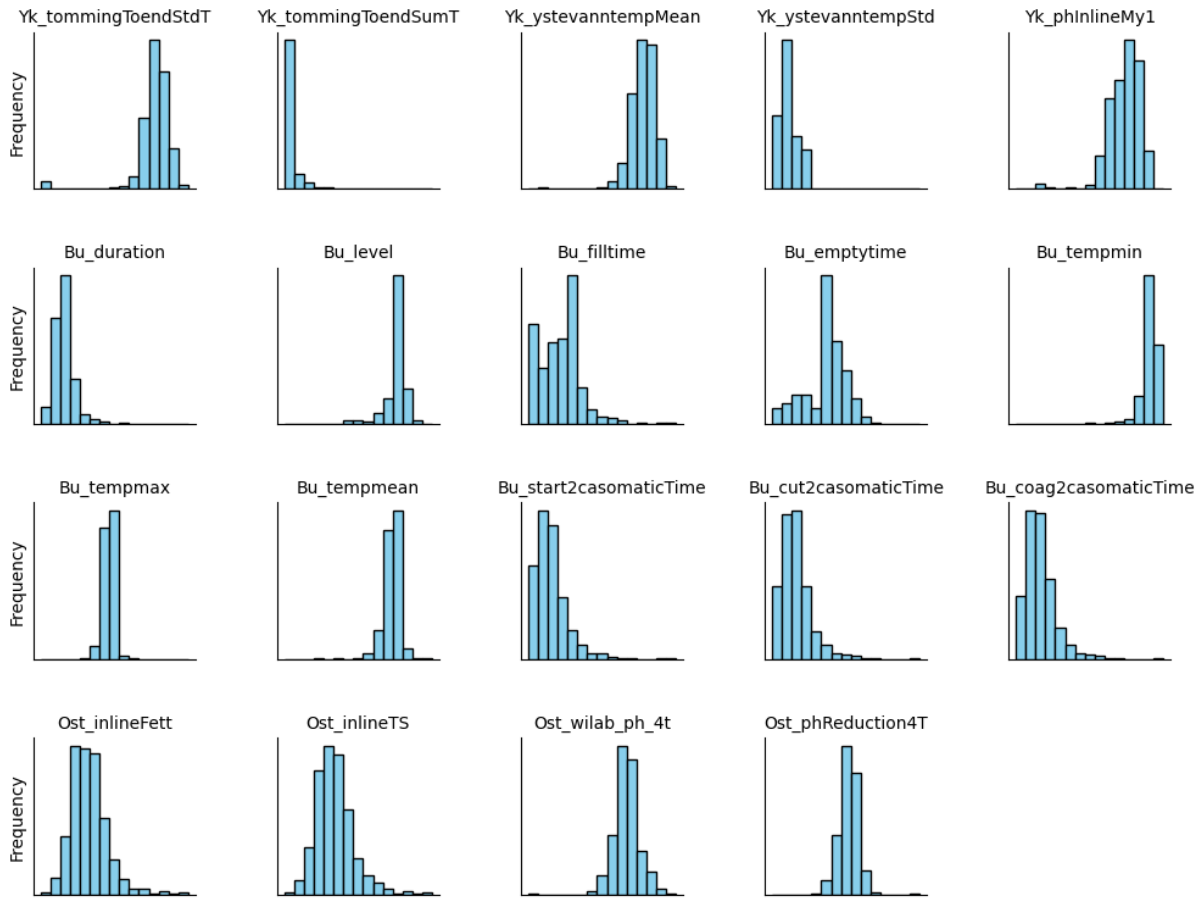


Figure 16: Histograms of all the numerical variables in the dataset.

The dataset being analyzed consists of several variables that exhibit different distribution patterns. Some variables follow an unimodal distribution, while others show a multimodal distribution with multiple peaks. Additionally, some variables have a uniform distribution. The dataset also contains potential outliers, which their long tails can identify in the histograms. One significant characteristic of the dataset is the presence of missing values. These missing values can range from one to a maximum of 1279 in some features and may occur due to sensor errors or production anomalies. If a variable has a high number of missing data, it can be challenging to accurately fill in the gaps and utilize that data in the future. Therefore, it is crucial to carefully preprocess variables with numerous missing values. Different techniques can handle missing values, such as iterative imputation as discussed in Chapter 2.1, discarding features with excessive missing data, or employing alternative methods. Iterative imputation is a commonly used technique that estimates the missing values using the observed values in the dataset. Discarding features with excessive missing data is another technique that can be used when the missing values are too many to handle. It is essential to select the appropriate method for handling missing values that align with the nature and specific requirements of the dataset. Given the importance of data quality, these strategies should be chosen carefully to maintain the dataset's integrity.

The data is organized chronologically in batches. Within each batch, there are multiple cheese vat batches, which are equivalent to samples. The batch number for each cheese vat batch is defined in the variable `Bs_Batch`, which contains a 17-digit number. Notably, there is a batch number equal to zero. This anomaly may indicate a discrepancy in the data collection process or potentially a data entry error. The starter culture is also produced batch-wise. Each starter culture is consistently used in multiple cheese vat batches within the same batch and likely shares similar properties. When analyzing data organized in batches, such as in cheese production, it is essential to consider the batch effects. These effects are caused by the uneven distribution of samples across batches, which can lead to variations that might affect the accuracy of comparisons or models. Therefore, it is crucial to consider batch effects to ensure that the results are reliable and meaningful. In our case, the starter culture is a critical factor to consider. Cheese vat batches (samples) within the same batch share the same starter culture, leading to dependencies or correlations between batches within the batch. Therefore, simply splitting the data into training and testing sets using random or standard CV methods may not adequately address these dependencies. Using LOGO CV, we can exclude entire batches from the training set while using them for testing. This approach ensures that the model evaluates unseen batches, providing a more realistic performance assessment and reducing the risk of bias due to overrepresented batches. Figure 17 illustrates the distribution of samples across batches, highlighting variations in sample counts among batches. Notably, some batches contain fewer than ten samples, which is significant as these smaller batches may impact the model's learning capability if not properly accounted for.

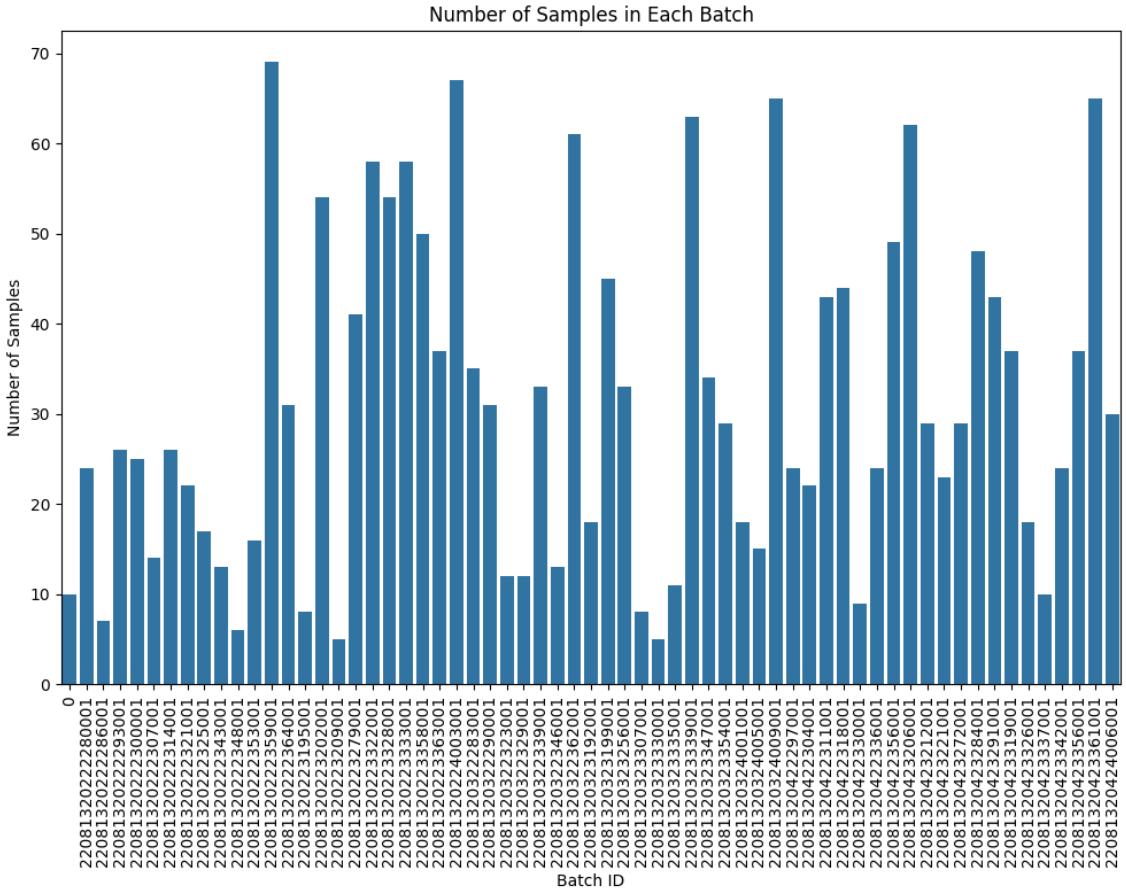


Figure 17: The image illustrates the number of cheese batches for each starter culture.

In Chapter 2.2, we discussed how some ML algorithms struggle with multicollinearity, which occurs when variables in the dataset are highly correlated. To identify highly correlated variables, we compute the correlations between all variables and then select the top 20 positive and negative correlations, as shown in Table 4 and Table 5, respectively. We analyze the top ten variables with the highest positive and negative correlations to streamline our analysis. However, it's crucial to acknowledge that there may be more variables with significant correlations beyond this selection. For the top positive correlations, there are strong relationships between certain features. For example, there is a very high positive correlation between `Yk_start2lopeTime` and `Yk_start2coagTime`, indicating that as one feature increases, the other also tends to increase. Similar strong positive correlations are observed between other pairs of features, suggesting that these features may be influenced by similar underlying factors or may have a direct relationship with each other. On the other hand, for the top negative correlations, there are strong negative relationships between certain pairs of features. For instance, a negative correlation of -1 between `Bs_artikkel_R6024` and `Bs_artikkel_R6024Cryostart` indicates a perfect negative linear relationship between these two features. In the production, these two variables represent two different acid cultures utilized, each with its own distinct composition and purpose. Other pairs of features also exhibit strong negative correlations, suggesting an inverse relationship between them. This aspect is essential when constructing predictive models and conducting subsequent analyses.

Table 4: Top 10 Positive Correlations.

Feature 1	Feature 2	Correlation
Yk_start2lopeTime	Yk_start2coagTime	0.999986
Yk_timeTomming	Yk_tommingToendSumT	0.999551
Bu_cut2casomaticTime	Bu_coag2casomaticTime	0.996603
Yk_timeYstevann	Yk_ystevannToTommingSumT	0.990985
Bu_start2casomaticTime	Bu_coag2casomaticTime	0.990714
Yk_timeMy1	Yk_my1ToYstevannSumT	0.988864
Bs_temp_Temp0	Bs_temp_Temp_stage26	0.987480
Bu_start2casomaticTime	Bu_cut2casomaticTime	0.986562
Yk_timeCoag	Yk_coagToCutSumT	0.982548
Bs_temp_end_incubation	Bs_temp_start_cooled	0.980447

Table 5: Top 10 Negative Correlations.

Feature 1	Feature 2	Correlation
Bs_artikkel_R6024	Bs_artikkel_R6024Cryostart	-1.000000
Ost_wilab_ph_4t	Ost_phReduction4T	-0.894052
Bs_temp_end_incubation	Bs_temp_Time_stage26	-0.791753
Bs_temp_end_incubation	Bs_temp_Time_stage91	-0.690754
Bs_temp_temp_incubation	Bs_temp_end_incubation	-0.689411
Bs_temp_end_cooled	Bs_artikkel_R6024	-0.677807
Bs_wilab_akt_ym	Bs_temp_start_cooled	-0.633775
Bs_temp_Temp0	Bs_temp_end_incubation	-0.630866
Bs_temp_end_incubation	Bs_temp_Temp_stage26	-0.624831
Bs_ph_t1	Bs_ph_pH1	-0.597400

Principal Component Analysis

In this section, we will utilize PCA for exploratory analysis of the dataset. As mentioned in Chapter 2.3.6, PCA plots focus on visual representations used to explore and understand the structure of high-dimensional data. We will present PCA plots including the cumulative explained variance plot, score plot, and loading plot.

Figure 18 presents the cumulative explained variance from the PCA. The X-axis represents the number of PCs, ranging from 0 to 84. Collectively, these components account for 100% of the explained variance. There is a sharp increase in explained variance for the first few PCs, which suggests that these initial components capture most of the variance in dataset $\mathbf{X}$. As additional PCs are included, the rate of increase in explained variance becomes more gradual for both lines, indicating that subsequent PCs contribute progressively less to the explanation of the variance as the number of components increases. For this PCA, a threshold of 95% cumulative explained variance was chosen to select the components. Therefore, 30 PCs were chosen and will be used for the subsequent PCA plots.

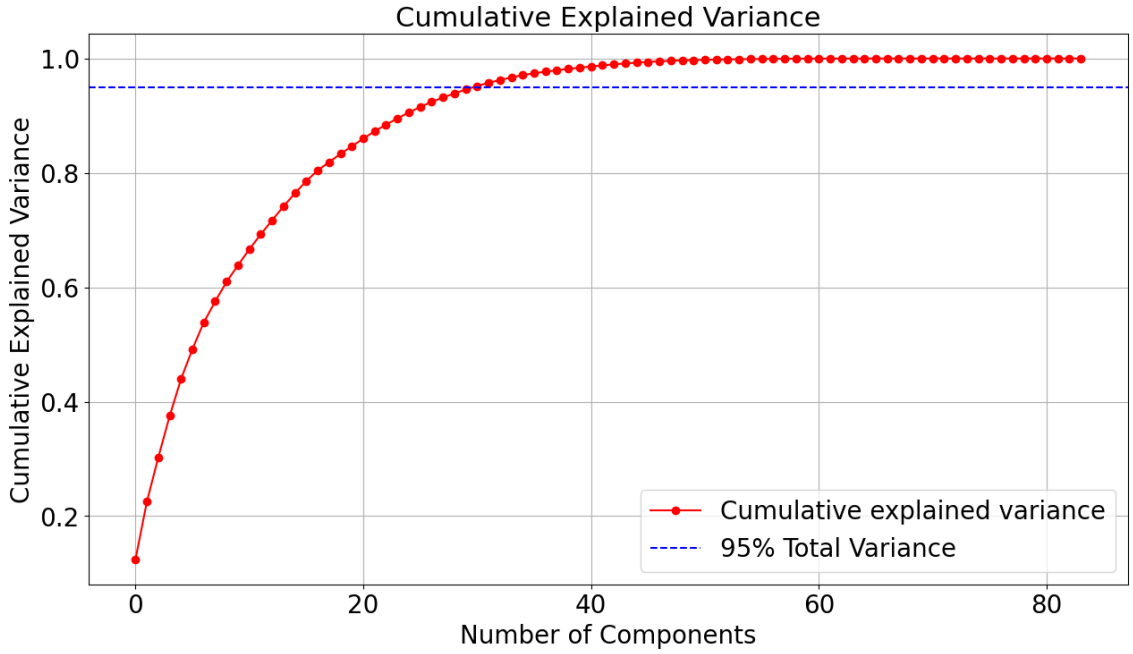


Figure 18: Cumulative explained variance in the dataset.

Figure 19 represents the score plot visualized with the first two PCs. There's noticeable variability in both the first and second PCs. There don't appear to be clear, distinct clusters, but there is some density variation. The bulk of the data points are clustered around the PC1, suggesting that most of the data variation can be explained by this component. The variation in color suggests that there's a relationship between the position of a data point on the PCA plot and its dry matter content. Even though there are some observations with very low dry matter content indicated by the dark blue color on the left side, the majority with higher dry matter content is predominantly in quadrants 1 and 4, which could suggest that higher values of PC1 are generally associated with increased dry matter content. However, there is no clear trend or distinct clustering solely based on the dry matter content across the two PCs, implying that these components alone may not perfectly segregate samples based on their dry matter content. Some points are far removed from the main cluster, especially along PC1. These points could represent samples from identical batches, aggregating due to batch effects.

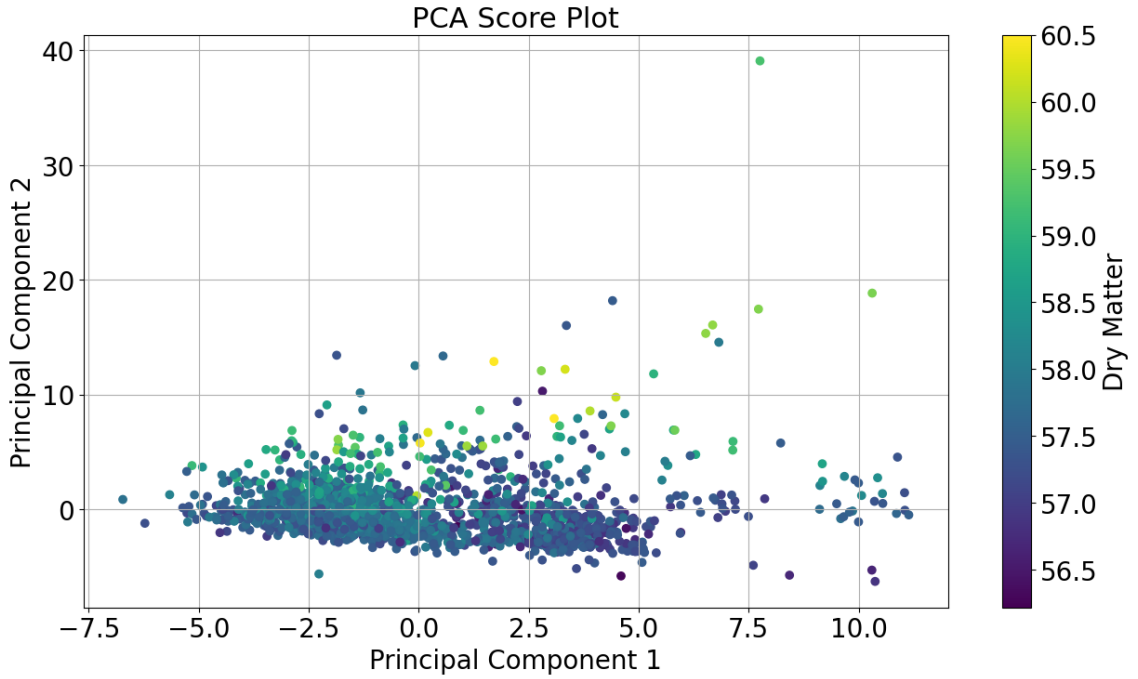


Figure 19: Score plot for explorative data analysis.

Figure 20 displays the loading plot and offers insights into how various variables influence the two PCs. Nine variables with the highest loadings were chosen for better interpretation and visualization. This selection criterion ensures that the most influential variables explaining the variance within the dataset are highlighted. Variables such as `Bs_artikel_R6024Crystostrt`, `Bs_temp_end_cooled`, `Bs_alder`, and `Yk_coagToCutAvgT` project towards the positive side of PC1. This indicates they have a strong positive correlation with PC1, suggesting that higher values of these variables align with higher values of PC1. From the small angle between these loadings, we know that they are positively correlated with each other. `Si_wilab_f_ftir` and `Si_wilab_prot_fir` show negative loadings on PC1, implying an inverse relationship with PC1. Observations with higher values of these variables tend to have lower values on PC1. From the plot, we can also observe that these variables are highly correlated. All the variables in the first and third quadrants are likely to be inversely correlated. This aligns with Table 5, which shows that `Bs_artikel_R6024Crystostrt` and `Bs_artikel_R6024` have a negative correlation of -1. Observations on the right side of the score plot, where dry matter content is mostly very low, are associated with variables that have positive loadings on PC1. This could indicate that variables like `Bs_artikel_R6024Crystostrt` and `Bs_temp_end_cooled` might indicate conditions or processes that result in a lower dry matter content. On the other hand, observations on the left side, associated with a bit higher dry matter content compared to the right side, correlate with the variables `Si_wilab_fir` and `Si_wilab_rot_fir`, which negatively load on PC1. It's important to note that not all variables have been selected for this visualization. Additional variables can be important for understanding other underlying patterns and variances in the dataset.

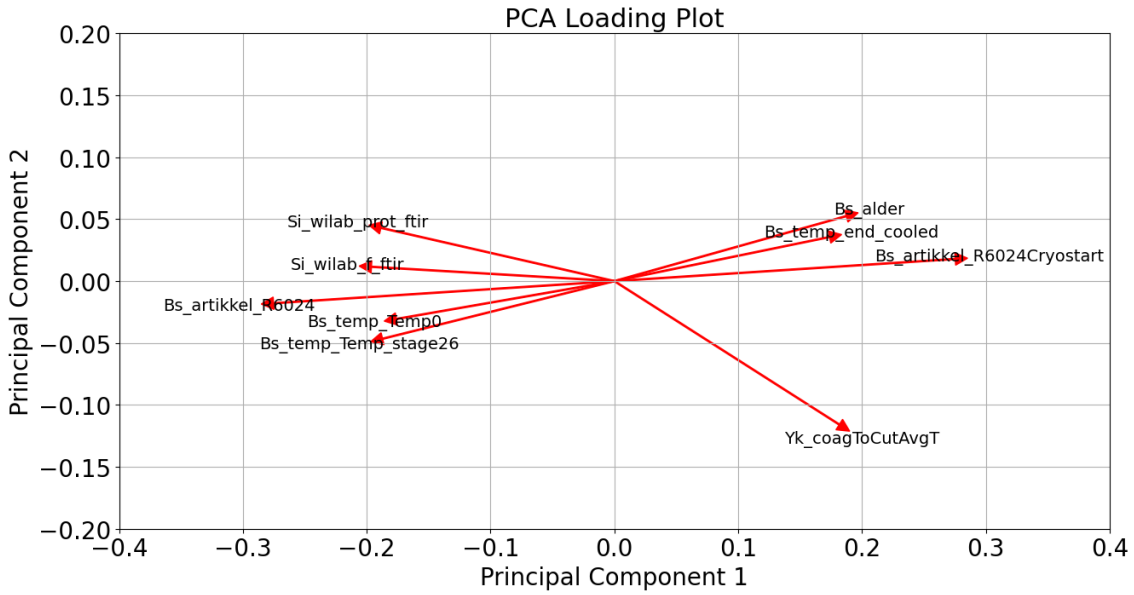


Figure 20: Loading plot from the PCA.

3.4.3 Data Preparation

Various data preparation steps have been designed for different algorithms to enhance their efficiency. The same preprocessing steps have been consistently applied across all the models to ensure a fair comparison and accurate evaluation of the algorithms' performance. This approach is crucial in maintaining the integrity and comparability of the results. The preprocessing technique was the same for all three ML models, except for handling missing values for HGBR. We treated missing values differently based on the algorithms' capabilities and requirements for complete data. The preprocessing for both functions started by removing all samples with `Bs_Batch` equal to zero, which amounted to ten instances. This step was essential to eliminate data corruption due to errors in the production process. Additionally, any missing values within the response variable were identified and removed. As the response variable is significant for accurate predictions, we decided that imputing missing values would be inappropriate, as it could introduce erroneous information into the model. Moreover, string features lacking predictive relevance were removed from the dataset to ensure compatibility with the modeling algorithms. Some examples of these variables are `Si_utstyr` and `sampleID`. Furthermore, `Ost_inlineFett` was removed from the analysis due to its dual nature as a response variable and its high positive correlation with dry matter. This could potentially introduce multicollinearity issues and distort the model's predictive accuracy. Lastly, the categorical feature `Bs_Artikkel` was One-hot encoded. For RENT and PLSR models, a different approach was taken to handle missing values. The iterative imputer discussed in Chapter 2.1 was used, utilizing the HGBR as the estimator model for imputation. This step was critical as these models require a complete dataset for the modeling phase. The HGBR model is designed to handle missing values during training. Given the batch effects within the dataset, it became imperative to employ techniques that could effectively tackle this issue. The dataset was organized chronologically by batches, making the modeling process smoother. The dataset was split into X_{train}, Y_{train}, X_{test}, and Y_{test} chronologically by time, with the training set containing 1,459 samples and the test set containing 295 samples. There were no overlapping batches between the training and test sets. Dividing the data this way ensures that temporal trends or shifts do not confound the model's learning process, reducing the impact of batch effects. This arrangement enhances the model's ability to generalize across unseen batches.

3.4.4 Modelling

The selected ML algorithms, HGBR, PLSR, and LR, have undergone thorough training and evaluation. To ensure reliability and accuracy, we use the LOGO CV approach. This technique helps us understand how well each model works and how accurate its predictions are in different scenarios. It also allows us to address batch effects and improve model performance on new data while preventing data leakage. Ultimately, the LOGO CV approach forms the foundation of our modeling strategy, ensuring that each algorithm undergoes rigorous testing and validation, resulting in the most reliable and effective predictive models.

Model 1 – Histogram-Based Gradient Boosting Regression

The process of modeling HGBR started with specific preprocessing steps designed for this model. Scaling of data was not required since tree-based algorithms can handle unscaled data. However, before using the model for prediction, it was crucial to select important variables that affect the target variable. To initiate the feature selection process, we trained the HGBR on the complete training dataset. After training the model, permutation importance was calculated 50 times to evaluate the importance of features in predicting the target variable. The features were ranked based on their mean importance scores, and the top 20 were identified for further analysis. These selected features were then analyzed using an SFS with backward elimination to identify a subset of 3 to 15 features that would maximize the model's R^2 score across a 5-fold CV. Once we identified the important features, we trained a new model with the same algorithm and selected features. We optimized the model's performance by tuning hyperparameters such as *max depth*, *max iterations*, *learning rate*, *minimum samples per leaf*, and *max bins* using Optuna.

Model 2 - Partial Least Squares Regression

The PLSR method requires specific preprocessing steps for modeling. The process involves applying PLSR to the entire dataset to identify the optimal number of components based on cumulative explained variance to prevent overfitting. Initially, two components were found to be optimal, determined by the highest R^2 score achieved. After refining the model, VIP (Variable Importance in Projection) scores were used for feature selection. A threshold of 0.8 was set for retaining the features, and the selected features were used to train a new model with three components, again identified by the highest R^2 score achieved. To examine the relationship between input features and dry matter content, the PLSR plots using *hoggormplot* provided nuanced visual interpretations of the data.

Model 3 - Linear Regression

The Linear Regression (LR) model was subjected to the same preprocessing steps as the PLSR model. The features for LR were chosen with the help of RENT. The optimal combinations of hyperparameters for the RENT algorithm were selected by evaluating the Bayesian Information Criterion (BIC), as explained in Section 2.4.4. To find the best parameters, we explored a range for the C parameter, which is the regularization parameter, from 1 to 50. Lower values indicate stronger regularization. Additionally, the $l1$ ratio, representing the balance between L1 and L2 penalties, was adjusted from 0.1 to 1 in increments of 0.1. A fixed test size of 0.25 was used for data division, ensuring a consistent framework for model assessment. CV techniques and data standardization were used to normalize input features and enhance the reliability and generalizability of the findings. The exploration involved generating 100 models, allowing for a comprehensive assessment of the hyperparameter space and ensuring the selection of the most effective model configuration.

3.4.5 Evaluation

A benchmarking procedure has been developed to assess the ability of ML models to predict unseen data. This procedure involves using performance metrics such as MSE, RMSE, and R^2 to determine the models' predictive power and goodness of fit. Codes have been developed for each ML algorithm to facilitate this process. Techniques like scatter plots, box plots, and PLSR plots are used to visualize the model's performance. These plots help to understand the relationship between the data and the underlying patterns. Figures and plots displaying the performance of each model are provided, and finally, all ML algorithms are compared, and conclusions are drawn based on their performance. The optimization model is evaluated using a custom scoring function. The Improvement Score is calculated based on the instances where the actual dry matter content has been increased to fall within the optimal value range of 57.6% to 57.7%. This score is then converted into a percentage, ranging from zero to one hundred percent (0-100%). The evaluation process and whether the success criteria were met will be discussed further in the discussion chapter, Chapter 5.2.

3.4.6 Deployment

During the final stage of the CRISP-DM process, we aim to implement the optimization model. We plan to integrate the model into a user-friendly website that uses real-time measurements via Streamlit. While we have developed a prototype website using Streamlit, we are not currently focusing on building a permanent website with real-time measurements for industry use. However, we will discuss ways to integrate such measurements effectively in the discussion section, which will create opportunities for future projects to create such a website.

Chapter 4

Results

In this chapter, we present the results of the evaluation procedure applied to each ML model described in Chapter 3. We begin by introducing the features selected and then present the performance metrics of each ML model. Additionally, we include several plots to understand the relationship between the independent and dependent variables.

4.1 Prediction of dry matter

4.1.1 Model 1 – Histogram-based Gradient Boosting Regression

In this section, we present the results of our evaluation process for our ML Model 1 using HGBR. The process involved identifying the top 20 features by using permutation importance. We then applied backward selection to refine the feature set and evaluate the model's predictive performance across various datasets. Table 6 displays the initial ranking of the top 20 variables based on permutation importance. Furthermore, Table 7 displays the final set of features after applying backward selection.

Top 20 Variables	Permutation Importance
Bu_cut2casomaticTime	0.106
Bs_temp_Temp_stage26	0.066
Bs_ph_pH2	0.035
Bu_duration	0.032
Bs_temp_Time_stage91	0.010
Bs_alder	0.009
Bu_start2casomaticTime	0.008
Bs_temp_Time_stage26	0.006
Ost_wilab_ph_4t	0.005
Yk_tommingToendAvgT	0.005
Bu_tempmean	0.004
Bu_tempmax	0.004
Ost_phReduction4T	0.003
Si_wilab_f_ftir	0.003
Yk_ystevannToTommingStdT	0.002
Yk_timeTomming	0.002
Yk_mengde	0.002
Yk_lopeToCoagAvgT	0.002
Bu_coag2casomaticTime	0.001
Yk_brukssyreToLopeStdT	0.001

Table 6: The top 20 variables selected using permutation importance.

Final selected features
Si_wilab_f_ftir
Bs_alder
Bs_ph_pH2
Bs_temp_Temp_stage26
Yk_mengde
Yk_tommingToendStdT
Yk_ystevannToTommingStdT
Bu_duration
Bu_coag2casomaticTime
Bu_tempmax
Bu_start2casomaticTime
Ost_wilab_ph_4t
Ost_phReduction4T

Table 7: Final selected features for Model 1, identified using Sequential Feature Selection (SFS) from the top 20 variables determined by permutation importance

The performance of HGBR model is presented systematically in Table 8 and Figure 21. Table 8 shows the accuracy metrics of the model, which provide a quantitative evaluation of its predictive capabilities across various statistical measures. On the other hand, Figure 21 presents a visual comparison between the actual and predicted values obtained by applying the HGBR model to both the training and test datasets. This visual representation highlights the model's ability to generalize from the training data to unseen data, illustrating discrepancies and alignments between predicted outcomes and actual results.

	Training set	Cross-validation	Test set
MSE	0.0256	0.0731	0.0937
RMSE	0.1600	0.2704	0.3060
R^2	0.9081	0.7375	0.7644

Table 8: Performance metrics for Model 1.

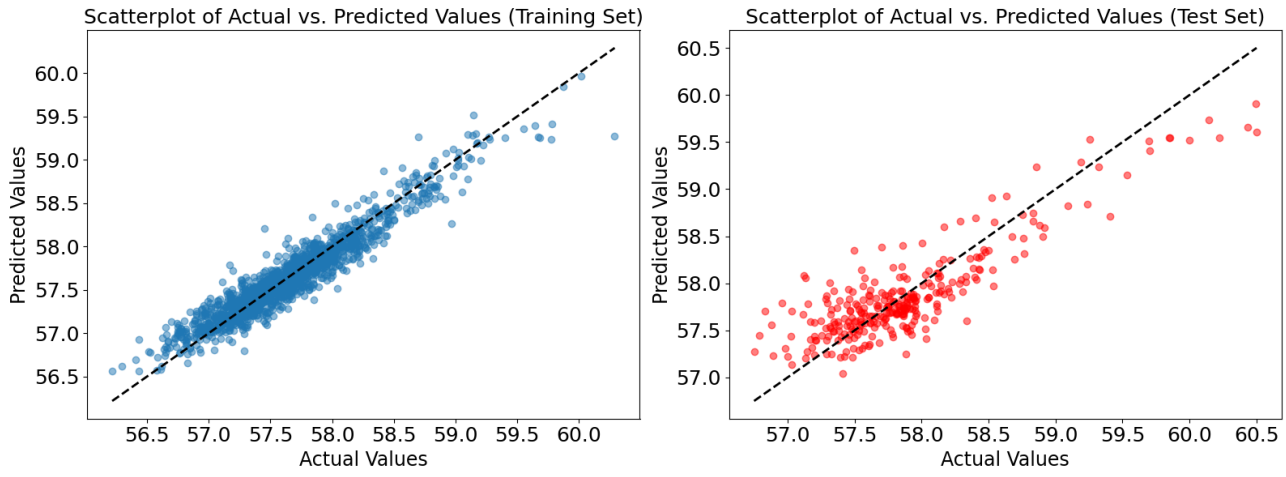


Figure 21: The image shows a scatterplot of actual versus predicted values from the Histogram-based Gradient Boosting Regression model for the training and test set.

The performance of the training data is illustrated by the dense clustering of points around the dashed line, reflecting a high degree of accuracy. This indicates that the model fits the training data well and has effectively captured the training set's underlying patterns. The CV results reveal a decrease in predictive performance, as evidenced by the increased MSE and RMSE, along with the decreased R^2 scores. Despite the decrease in performance, the score remains high, suggesting that the model possesses a reasonable ability to predict new, unseen data. The test set results, which serve as the final unbiased evaluation of the model, display a wider spread of points. This wider spread of data points suggests that the predictions are less accurate than the training set, potentially indicating the model's decreased ability to generalize to unseen data. However, our approach to evaluation has been designed to address the batch effect problem. By implementing LOGO CV, we made our predictions more realistic and robust, enhancing the reliability of our performance metrics on new unseen data. Moreover, by ensuring no overlap between batches in the training and test sets, we ensured that the scores we obtained were highly reliable and precise on new, unseen data. As mentioned in Chapter 2.3.5, HGBR is a non-linear, tree-based ensemble model. The model's high score could be due to the algorithm's ability to discover complex patterns through non-linearity. This could help the algorithm to effectively utilize the underlying structure of the data, possibly revealing influential variables and interactions that are not immediately obvious. Furthermore, as a tree-based ensemble method, it utilizes multiple weak learners working together to create a strong predictor, significantly enhancing the model's predictive power. However, this model type is prone to overfitting, which can be seen by the model's performance. Even after selecting the minimal tree depth, the training score remains high. Overfitting is a common issue observed in tree-based models and represents a well-known drawback. This is reflected in the result, where training accuracy is significantly higher than the test across all datasets.

4.1.2 Model 2 – Partial Least Squares Regression

This section aims to evaluate the results of the Model 2 using PLSR. The focus will be on assessing the model's predictive ability and the significance of the predictor variables identified through our analysis. We began the feature selection by computing the Variable Importance in Projection (VIP) scores. Table 9 presents a list of selected features with VIP scores greater than 0.8. These features have been used for modeling purposes.

Important Variables	VIP-score
Bu_start2casomaticTime	2.589995
Bu_cut2casomaticTime	2.498496
Bu_coag2casomaticTime	2.481256
Bu_duration	2.041004
Yk_start2endTime	1.962124
Bu_emptytime	1.762194
Yk_tommingToendSumT	1.758585
Yk_timeTomming	1.721259
Yk_tommingToendAvgT	1.497236
Bs_temp_Time_stage130	1.382967
Bs_alder	1.322590
Yk_mengde	1.317194
Ost_wilab_ph_4t	1.304591
Bs_temp_end_cooled	1.303960
Bu_tempmean	1.302259
Yk_mysemengde_r6001	1.273180
Bu_tempmax	1.267663
Bs_artikkel_R6024	1.265423
Bs_artikkel_R6024Cryostart	1.265423
Yk_ystevannToTommingSumT	1.240171
Ost_phReduction4T	1.128447
Yk_start2tommingTime	1.122613
Yk_timeYstevann	1.105214
Yk_tommingToendStdT	1.067067
Bs_wilab_ph	1.064797
Si_wilab_prot_ftir	1.003233
Bs_temp_temp_cooled	0.952943
Yk_lopeToCoagAvgT	0.921632
Si_wilab_f_ftir	0.908433
Bs_Mengde	0.883577
Bs_temp_Temp_stage91	0.879235
Yk_lopeToCoagSumT	0.874524
Bu_filltime	0.869147
Bs_ph_mx_dpH	0.818364
Yk_coagToCutAvgT	0.801372

Table 9: VIP-scores of Important Variables.

Table 9 presents the variables in descending order of their VIP scores, which measures their contribution towards the model's predictive accuracy. The variable with the highest VIP score of 2.589995 is `Bu_start2casomaticTime`. This is followed closely by `Bu_cut2casomaticTime` with a VIP score of 2.498496, and `Bu_coag2casomaticTime` with a VIP score of 2.481256. To select the most significant features based on their VIP scores, we set a cut-off threshold of 0.8 and included the variables in the table accordingly. Following this, we trained a new PLSR

model using these important features transformed into orthogonal components as predictors. Table 10 presents the detailed accuracy metrics. Figure 22 compares the actual and predicted values for the training and testing datasets, thus providing a comprehensive view of the model's predictive power.

	Training set	Cross-validation	Test set
MSE	0.0524	0.0673	0.1548
RMSE	0.2290	0.2594	0.3934
R^2	0.8117	0.7584	0.6105

Table 10: Performance metrics for Model 2.

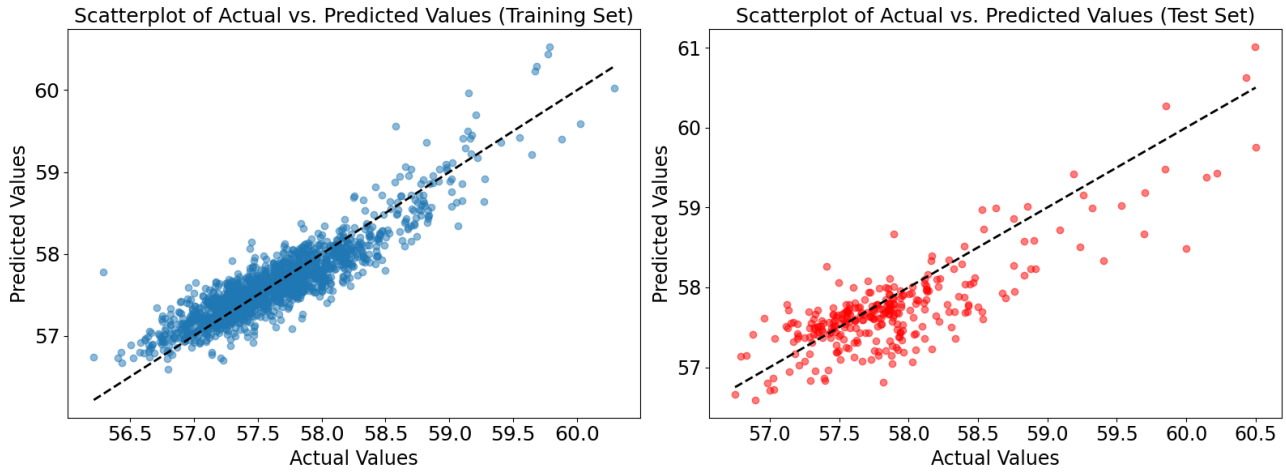


Figure 22: The figure displays a scatterplot comparing actual and predicted values generated by HGBR for both the training and test sets.

The dense clustering of points around the dashed line illustrates the high accuracy of the training data. Model 2 demonstrated strong performance in the training set, characterized by low MSE and RMSE values, and a high R^2 score. The CV shows small MSE and RMSE values, closely resembling those of the training set. This indicates the model's ability to generalize effectively across various data subsets. Such consistency underscores the model's robustness and reliability. The test set performs notably worse than both the training and CV sets, as anticipated. However, the performance scores are still moderate. The significant difference in performance between the training and test sets suggests the potential occurrence of overfitting. Figure 22 reveals a high density of data points along the reference line in the test set, especially in the values ranging from 57 to 58. This suggests that within this range the models' predictions are close to the actual value. As the actual values increase, the density of the points begins to spread out slightly more.

PLS Regression Plots

The visualizations of the PLSR plots are provided below and are important in illustrating the behavior of the model, offering insights into how well the model captures the underlying patterns of the data.

Figure 23 presents the score plot, which illustrates the distribution of the dataset across the first two principal components. The first component accounts for 66.5% of the variance in the response variables, which indicates a strong relationship with the dependent variables. It also explains 17.2% of the variance in the predictor variables, highlighting its importance in the model. The second component captures less variance, with 12.7% for the response variables and 9.2% for the predictor variables, indicating a secondary yet significant influence. The central cluster of data points suggests that most observations have common characteristics. Notably, many observations cluster together within the same batch, hinting at additional factors contributing to unexplained variance. This also confirms that many samples within the same batch share the same characteristics. Some observations, such as numbers 100, 101, and 102, are from the same batch, which logically explains why they are grouped together far in the bottom right corner. This grouping suggests they share similar characteristics, likely reflected by high values in the loading plot in that direction. However, data points significantly deviating from this central cluster suggest outliers or variations within the dataset. Some deviations are grouped, potentially signaling a consistent variation in the cheese-making process. Such a pattern might be expected in the food industry or imply that variations in a particular variable influence the overall variation. These insights could guide further investigation into specific areas of the production process for quality control or process optimization.

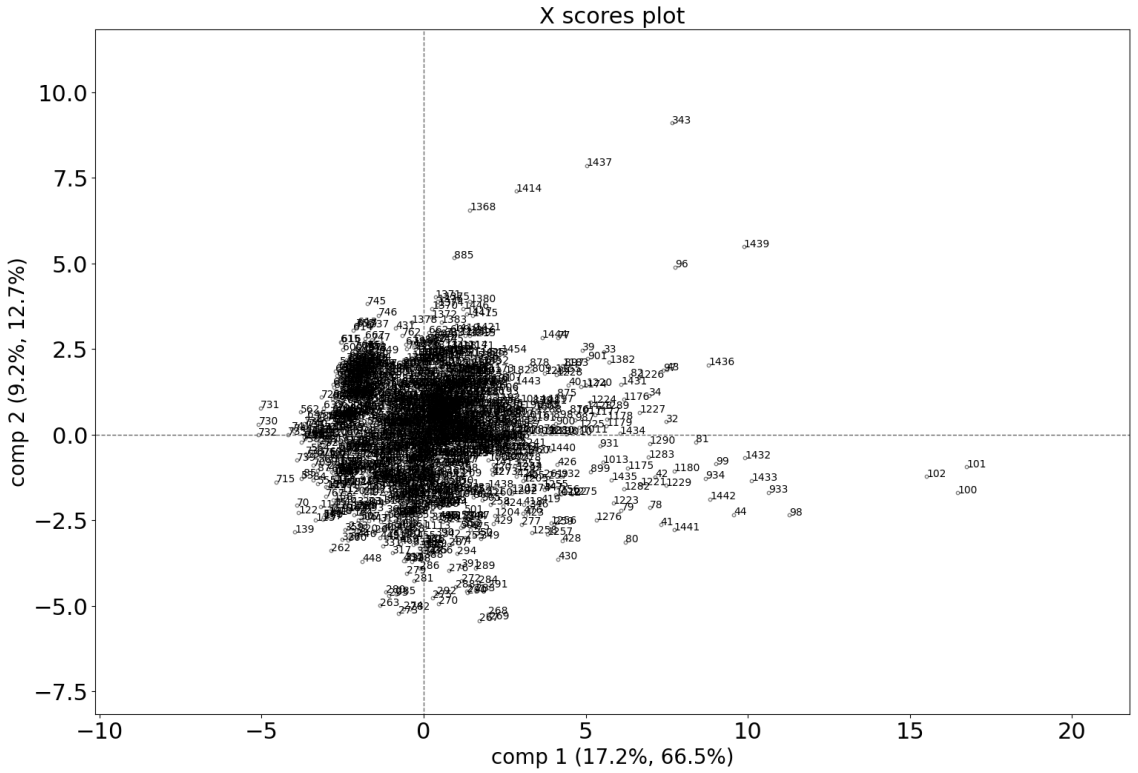


Figure 23: Score plot from PLS Regression.

Figure 24 shows the loading plot, which demonstrates the contribution of each predictor variable and the relationships among the original variables with respect to the first two components identified in the analysis. The variable `Bu_duration` exhibits a positive loading on component 1 and a negative loading on component 2. Conversely, `Yk_mengde` are positioned in the third quadrant, suggesting their significant negative impact on component 2. Some of the variables in the dataset that explain the most variance are located farthest away from the origin, and they are clustered in the bottom right corner. These variables include `Bu_start2casomaticTime`, `Bu_cut2casomaticTime`, `Bu_coag2casomaticTime`, `Yk_start2endTime`, and `Bu_duration`. They reflect the VIP scores in Table 9 for the most influential variables in making predictions. These variables represent time measurements, making them highly correlated with each other and the first component. Therefore, most of the time-related measurements are important for dry matter predictions. It is also worth noting that some variables, such as `Ost_phReduction4T` and `Ost_wilab_ph_4t`, are inversely correlated, meaning that if one variable increases, the other decreases, and vice versa. This relationship was observed in Chapter 3.4.2. Most variables are highly correlated with other variables from the same subprocess. It is logical to expect that measurements within the same subprocess will influence each other when adjustments are made. Within any subprocess, especially in industrial or manufacturing settings, measurements are often related because they reflect aspects of the same process. For example, if one step in a subprocess changes, such as the temperature or duration, it can affect other measurements within that subprocess, leading to correlations. However, some variables, like `Yk_mengde`, do not correlate highly with other variables within the same subprocess. Instead, this variable shows a stronger correlation and is more similar to variables such as `Bs_alder`, `Bs_temp_end_cooled`, and `Bs_temp_Time_stage130`.

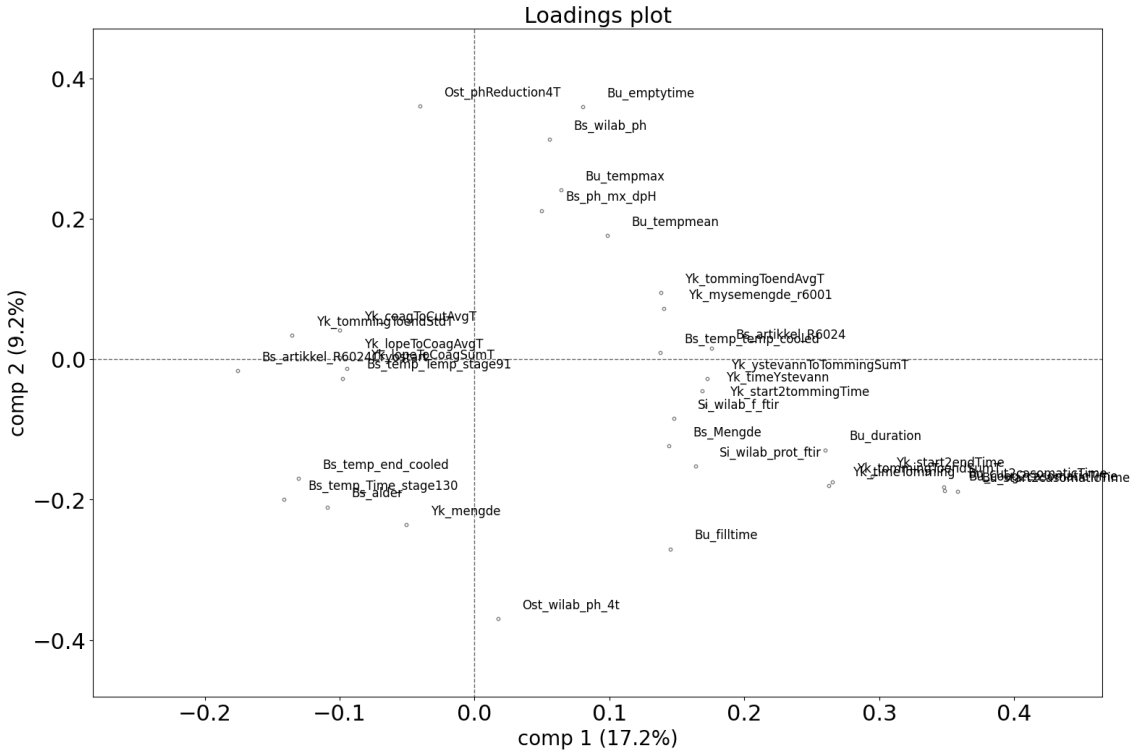


Figure 24: Loading plot from PLS Regression.

Figure 25 displays the explained variance plot shown in the dependent variable by the components derived from the PLSR. The graph depicts two lines tracking the explained variance across multiple components. The blue line corresponds to the explained variance for the response $\mathbf{y}$, representing how much of the variance in the response is captured by the model. Conversely, the red line corresponds to the validated explained variance, indicating the proportion of variance in the response variable that is explained by the model when subjected to a validation process. The X-axis shows the number of components ranging from 0 to 3. The y-axis shows the explained variance in the dependent variable. We can see that the first components add the most explanatory power, which suggests that the most critical factor captured by the model resides within the first component. This aligns with observations from other plots, where the most influential predictors are highly correlated with the first component. The small gap between the calibrated and validated lines suggests a good model generalization. The model's performance on the validation set is good, considering the validated explained variance does not drastically drop as more components are added, suggesting that the model's predictive power does not rely on overfitting to the training data.

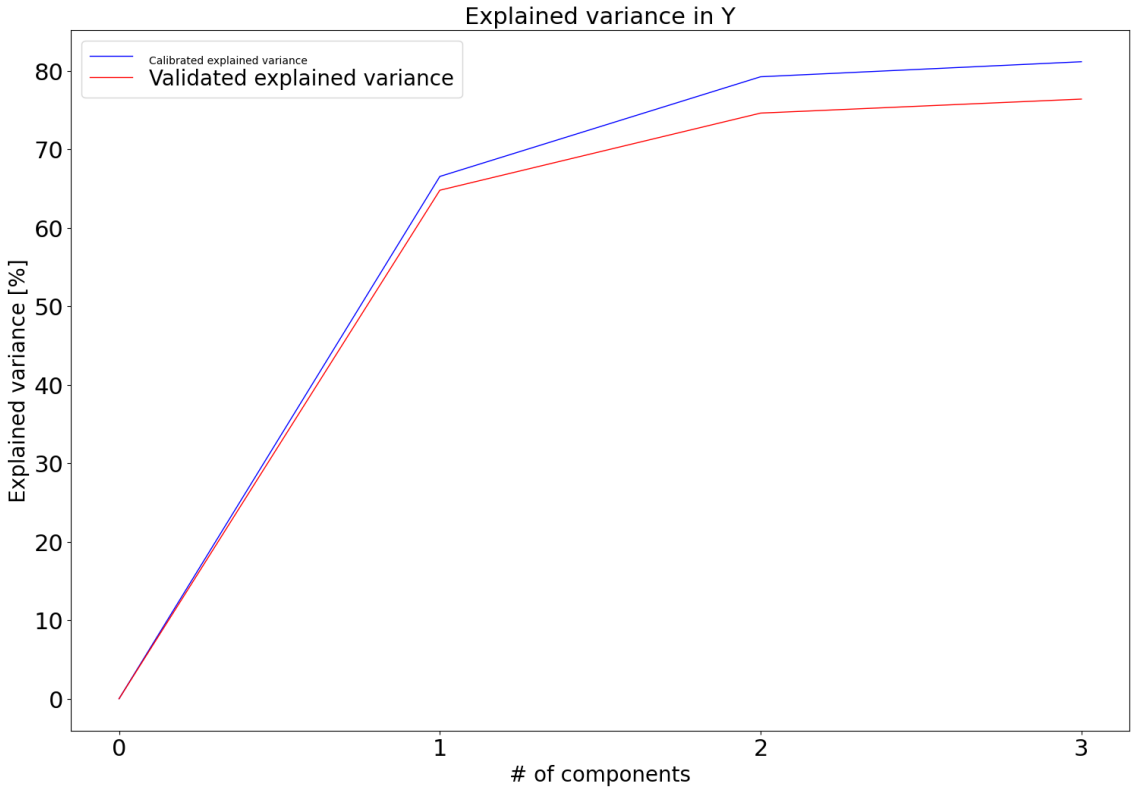


Figure 25: Explained variance in Y from PLSR.

4.1.3 Model 3 – Linear Regression

The following section presents the outcomes of Model 3 using LR, including the feature selection process using RENT and model performance evaluation. Additionally, validation study analysis are also presented. Table 11 enlists the features that the RENT method selected, using a cutoff of 0.95 across all three criteria. The selected features are considered significant predictors due to their substantial predictive power and importance in determining the dry matter content.

Selected features by RENT
Si_wilab_f_ftir
Bs_alder
Bs_ph_pH2
Bs_temp_end_cooled
Bs_temp_Time_stage91
Bs_temp_Temp_stage91
Bs_artikkel_R6024
Yk_mengde
Yk_lopeToCoagAvgT
Yk_coagToCutSumT
Yk_ystevannToTommingStdT
Yk_ystevannToTommingSumT
Yk_tommingToendAvgT
Yk_tommingToendStdT
Bu_filltime
Bu_emptytime
Bu_tempmax
Bu_tempmean
Bu_start2casomaticTime
Ost_wilab_ph_4t

Table 11: Selected Features by RENT.

For feature selection, RENT was utilized, and the performance of the model was tested on various datasets. Accuracy metrics for the model can be viewed in Table 12. Additionally, Figure 26 shows a comparison between predicted and actual results of the model on both training and test data.

	Training set	Cross-validation	Test set
MSE	0.0441	0.0597	0.1621
RMSE	0.2102	0.2444	0.4027
R^2	0.8414	0.7854	0.5921

Table 12: Model Performance Metrics for Linear Regression using RENT as the feature selection method.

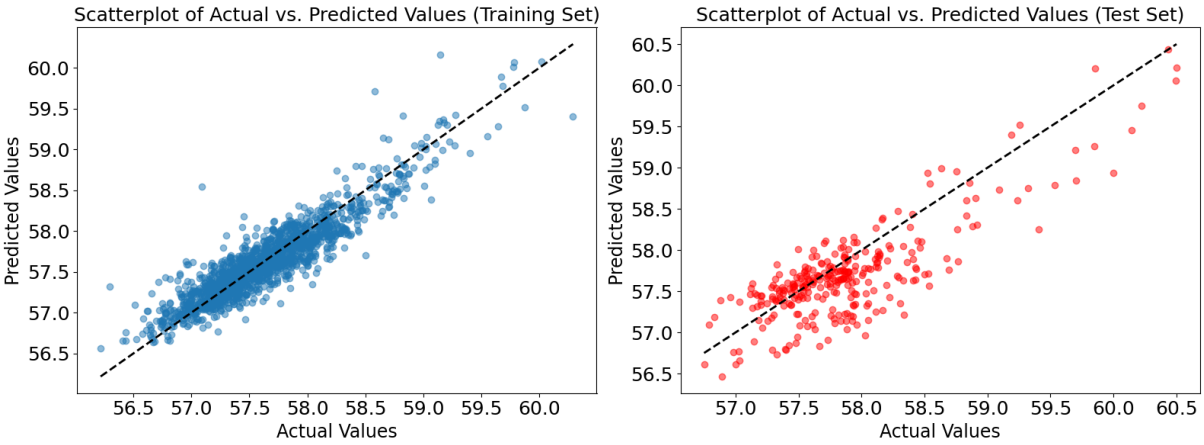


Figure 26: The image shows scatterplots of actual versus predicted values from the Linear Regression model for the training and test set.

Examining the model's performance using the RENT approach, the analysis begins with the training set. The model demonstrates a strong ability to capture the underlying data patterns, as indicated by a relatively low MSE and RMSE alongside a high score of $R2$. This suggests that the model fits well with training data. Moving to the CV results, there is a slight increase in the MSE and RMSE, with a decrease in R^2. The modest difference between the training and CV metrics further supports the model's effective generalization. However, the model's performance on the test set shows a decline, with significant increases in both MSE and RMSE, and a drop in the R^2 score. This indicates a decrease in predictive accuracy when applied to new, unseen data. This shift is visually confirmed by the graphical analysis of predicted versus actual values. While the training set exhibits a tight clustering of points around the line of identity, illustrating good calibration, the test set displays a more dispersed scatter of points. This increased spread reflects higher error rates and a reduced R^2 value, signaling challenges in the model's effectiveness on external data.

Validation Study

According to the theory presented in Chapter 2.4.4, we present the RENT model's validation study. The study aims to determine if the RENT model performs better than the benchmarks for different validation studies. The summarized results of this study can be found in Table 13 and Figure 27.

Table 13: Summary of RENT Model Validation Results.

	Mean	p-value
VS1	0.27	8.11e-160
Heuristic p-value (VS1)	—	0.108
VS2	-0.91	1.74e-115
Heuristic p-value (VS2)	—	0.0

Note: H_0 is rejected at the significance level of 0.05.

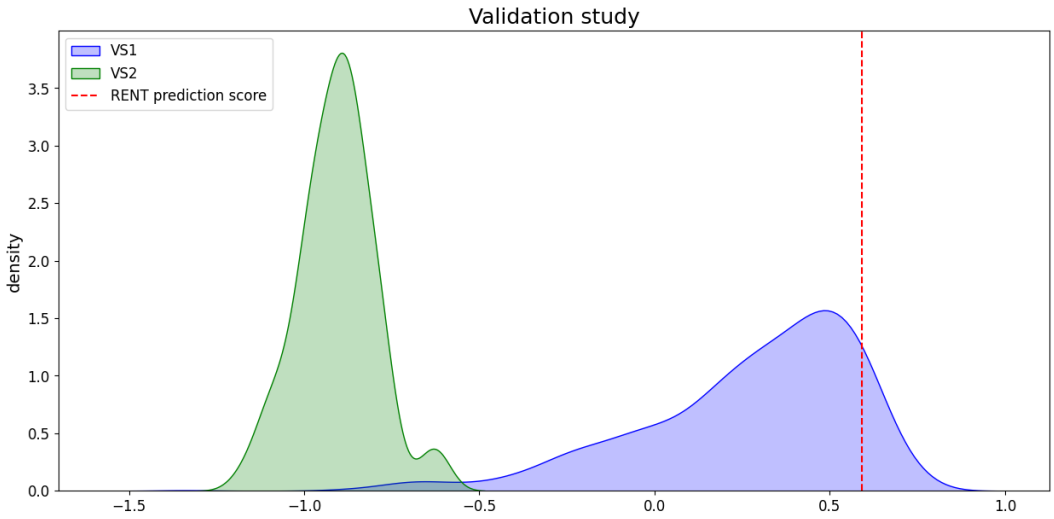


Figure 27: The image shows of two validation studies for the RENT mdodel.

We observed significant predictive performance in the validation study using the RENT method. The mean validation score from VS1 was 0.27 with a highly significant p-value, indicating a robust model compared to random feature drawing. The heuristic p-value for VS1 also suggests that the RENT prediction score is reliably higher than random chance, confirming the effectiveness of the selected features. However, the model is not perfect, as indicated by a heuristic p-value greater than the significance level of 0.05, which points to room for improvement in some areas. With a heuristic p-value of 0.11, approximately 11% of the random scores are higher than those obtained using the RENT method. This suggests that the RENT method's score is better than the scores obtained by chance about 89.2% of the time. Since the first p-value is much lower than 0.05, the conclusion drawn is that H_0 is rejected for VS1. The heuristic p-value is more of a supplementary measure, indicating that while the model is statistically significant, some variability in its predictive power might be improved. For Validation Score 2, the mean was -0.906, again with a p-value so small it is essentially zero, strengthening our confidence in the model's predictive abilities over random permutation of test labels. The heuristic p-value is exactly zero, underscoring that the RENT prediction score consistently outperforms random

predictions, allowing us to reject the null hypothesis with a high degree of certainty. The RENT score, marked by the dashed red line, clearly stands apart, validating the model's predictive power.

Figure 28 displays three distinct metrics: stability, performance, and runtime as functions of the number of models (K) for RENT. The red line indicates the consistency of feature selection across the different models, whereas the blue line signifies the R^2 score across various models. The green line illustrates the computation time associated with the number of K models. This line exhibits linear growth, which is expected as the number of models increases. The stability of the features selected is exceptionally high, suggesting that the feature selection is consistent irrespective of the number of models run. The blue line, representing the performance, is expected to be around an R^2 score of 0.60. Like the other models, this could be attributable to batch effects and a lack of sufficiently significant variables with predictive power. The performance levels off slightly after 50 models and remains constant, indicating that an increase in the number of models beyond a certain point does not result in significant improvements in predictive performance. This plateau could be due to data complexity, batch effects, or other forms of noise. As anticipated, the runtime increases linearly with the number of models. From the plot, we can deduce that a lower number of models, K, is adequate for achieving optimal stability and performance while minimizing runtime.

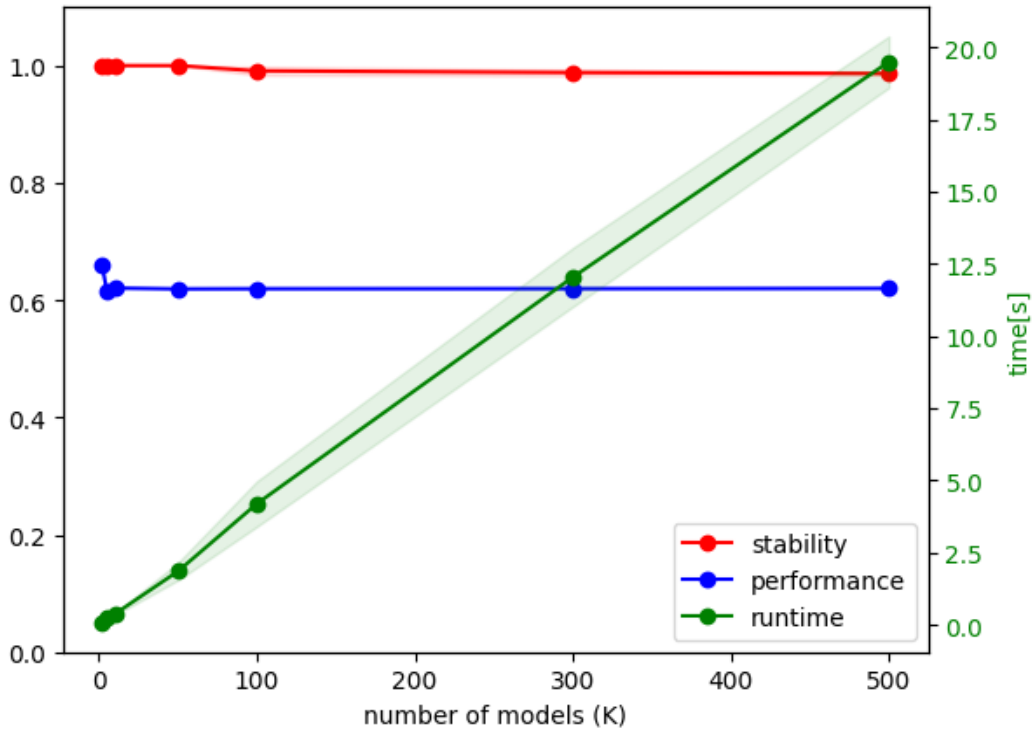


Figure 28: Stability performance in RENT.

4.2 Model Comparisons

In the previous section, the results of each model were presented individually. Moving forward, we will compare the models against one another. We will then determine which model(s) will be implemented in the optimization phase. First, we will examine the figures below that present a clear comparison of the performance scores for each model, using the optimal feature sets obtained from each. Additionally, we will examine the figures, which provides a summary of the performance scores obtained when the optimal feature sets from each model are applied to each other. This will allow us to observe the resulting scores and determine cross-model efficacy. Figures 29, 30, and 31 display a side-by-side summary of the results from the dry matter prediction across the different performance metrics using their unique feature subset.

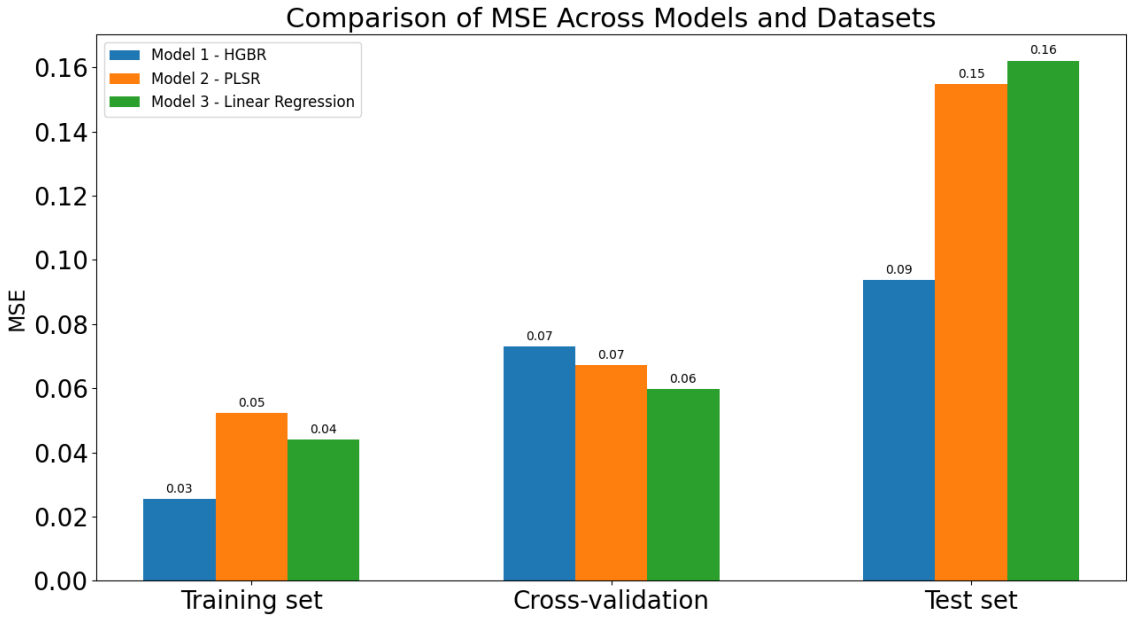


Figure 29: Model comparison for MSE.

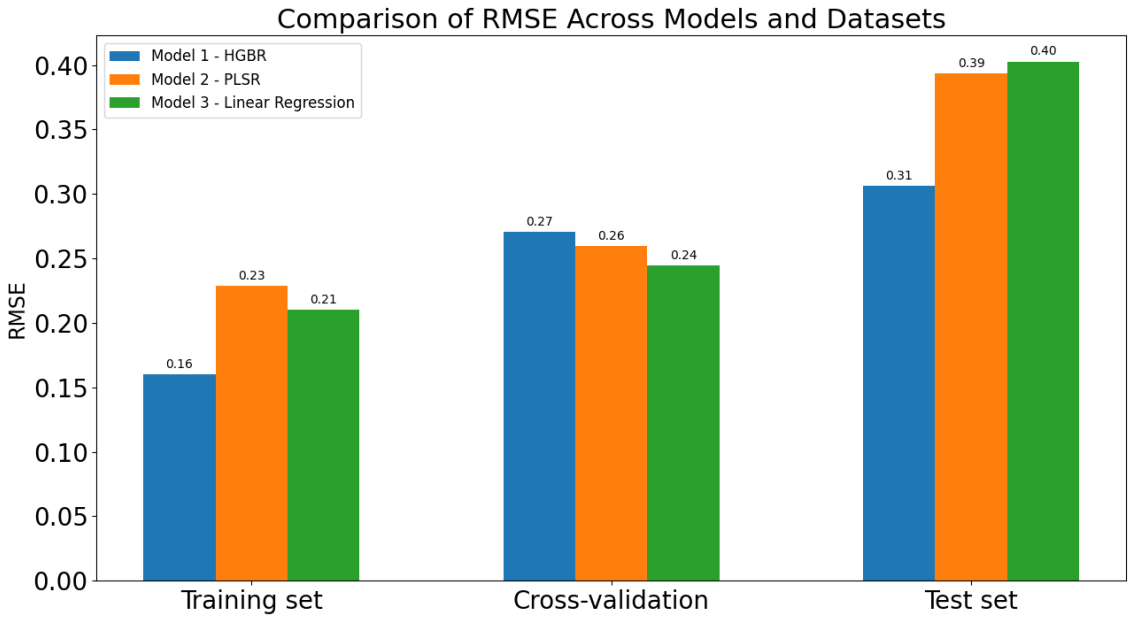


Figure 30: Model comparison for RMSE.

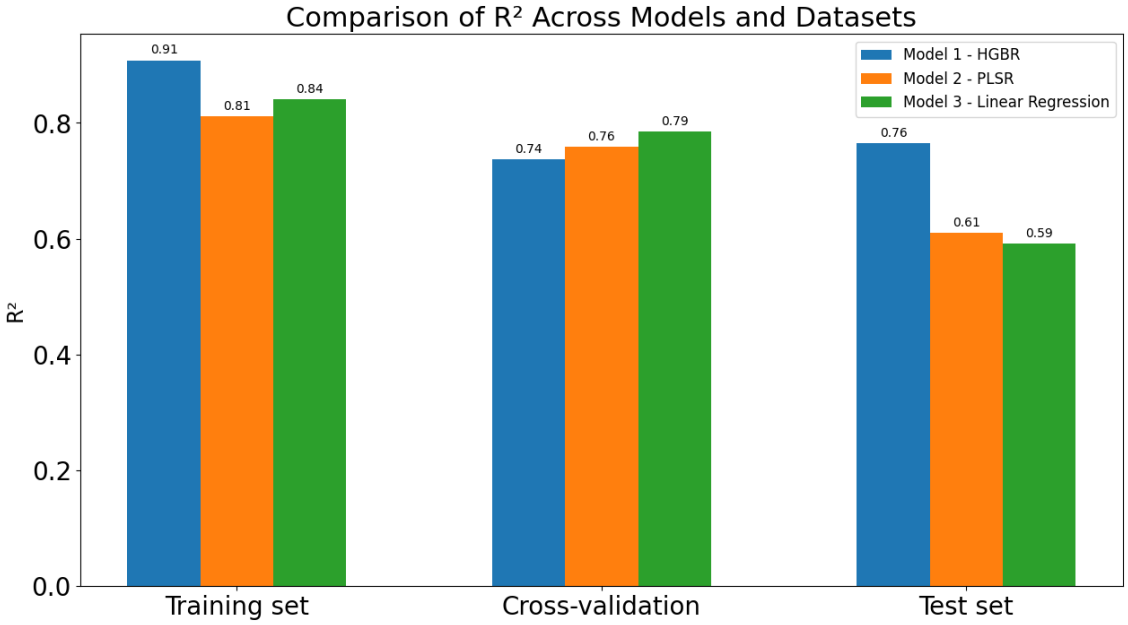


Figure 31: Model comparison for R^2.

The evaluation across the models demonstrates varied abilities to handle data complexity and generalize well. HGBR (Model 1) showcases the strongest performance on the test sets, with the lowest error rates and highest R^2 scores, suggesting its effectiveness in managing non-linear patterns, as evidenced by its consistent results despite a slight underperformance in CV compared to the other models. In contrast, PLSR (Model 2), while performing well in CV due to its simplicity and potential noise robustness, struggles on the test set. The LR model (Model 3) outperformed the others during CV but performed poorly in the training and testing phases. This could indicate that while tree-based methods like Model 1 effectively capture complex data variations, simpler models might require enhancements or more sophisticated regularization to improve their adaptability and performance on unseen data. To summarize, the HGBR model shows promise, especially in handling complex, non-linear patterns within the dataset. However, there is a noticeable trade-off between model complexity and performance on unseen data. The PLSR model, though not excelling on the test set, has merits in CV that need further exploration. Lastly, while LR model maintains relatively consistent performance from training to CV, it experiences the most substantial decline in performance when moving from CV to test set, compared to other models. This notable decrease suggests potential issues with overfitting or a lack of robustness against unseen data.

Figure 32 demonstrates the shared and unique features used by three models. Each circle represents the features utilized by a particular model. The features present in the overlapping regions of the circles are used by more than one model, indicating their significance and relevance across different regression techniques.

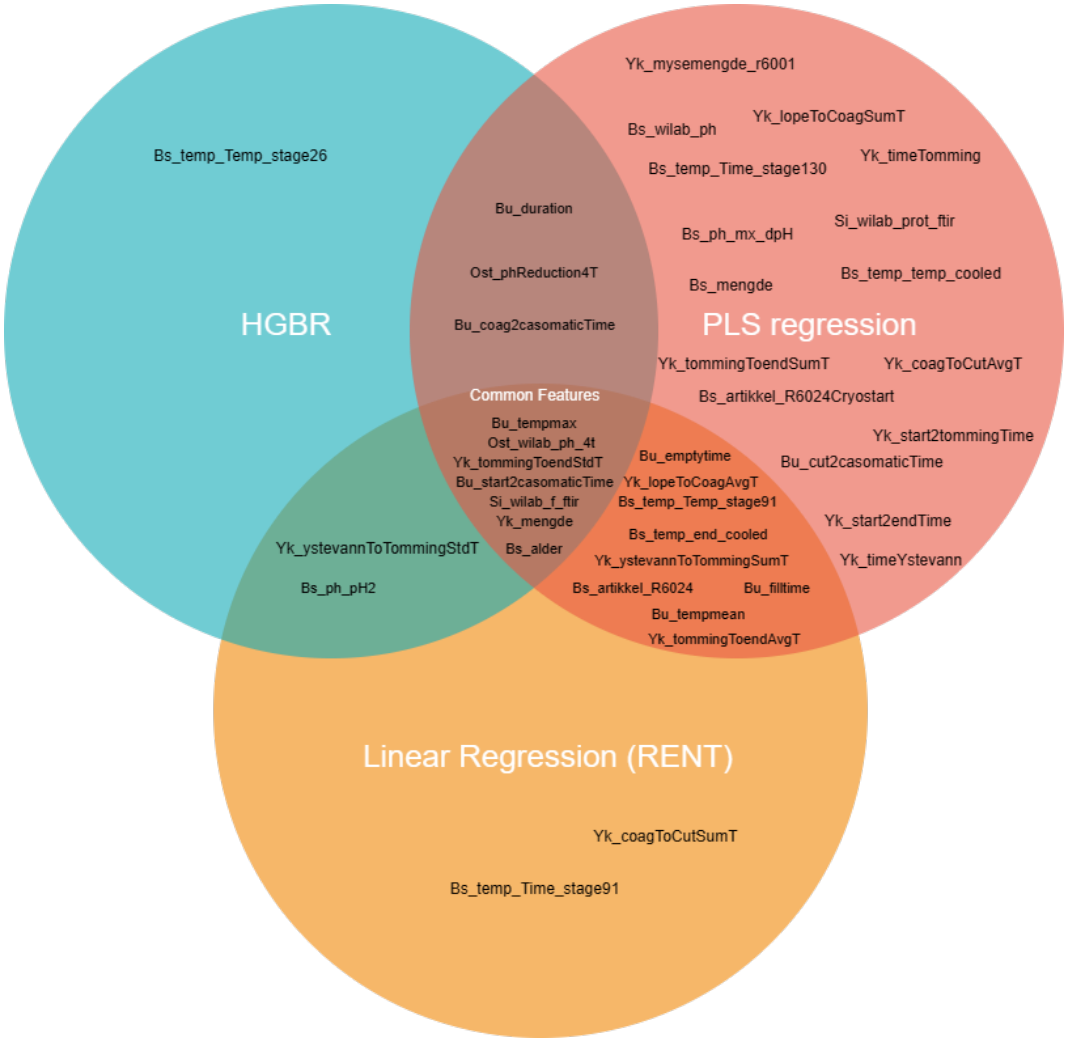


Figure 32: Venn diagram of the optimal features selected for the different models.

The Venn diagram showcases how different features are distributed across three models: HGBR, PLSR, and LR. The HGBR model's circle represents both unique and shared features. The RENT circle has two distinct features but shares many with PLSR, suggesting a consensus on the importance of these features across different models. The PLSR circle contains features that are specific to this model and those that are shared with one or both of the other models. The unique features in this circle could be particularly significant in predicting outcomes using PLSR, possibly due to the model's ability to handle multicollinearity or its method of reducing predictors to a smaller set of uncorrelated components. PLSR may require a broader set of features to achieve a predictive model. The features that intersect all circles are called the "Common features" and are considered essential across all models.

In this analysis, we compare the performance of the three models using optimal feature sets identified by each model. The comparisons are illustrated in Figure 33. By looking at the R^2 scores, we can determine how good each model is at capitalizing on feature sets from alternative methodologies, providing insights into their versatility and effectiveness in different contexts. The graph displays a group of "Common Features" that all the models use. Although the LR model performs slightly better on this subset, it is not universally effective across all models. The HGBR model still shows the highest R^2 scores on its feature subset, indicating that it outperforms other models in predicting test data. This is likely due to its ability to handle complex data patterns better than PLSR and RENT. While PLSR and RENT deliver decent performances, they do not achieve the same level of effectiveness as the HGBR model. This variation in performance may be due to the inherent differences in how these algorithms manage feature sets. The HGBR model, with its tree-based structure and gradient-boosting mechanism, tends to perform robustly across extensive feature sets, which may help mitigate issues such as noise and overfitting that can adversely affect other models. However, it is important to note that having many features is not universally problematic. For instance, PLSR is especially useful in handling high-dimensional data, a strength that should not be overlooked. In summary, this analysis not only tests the robustness of each model but also demonstrates the potential for improving model performance through informed feature selection. Understanding the strengths and weaknesses of each model can help us choose the correct algorithm for different datasets and contexts and ultimately improve the accuracy and effectiveness of our predictions.

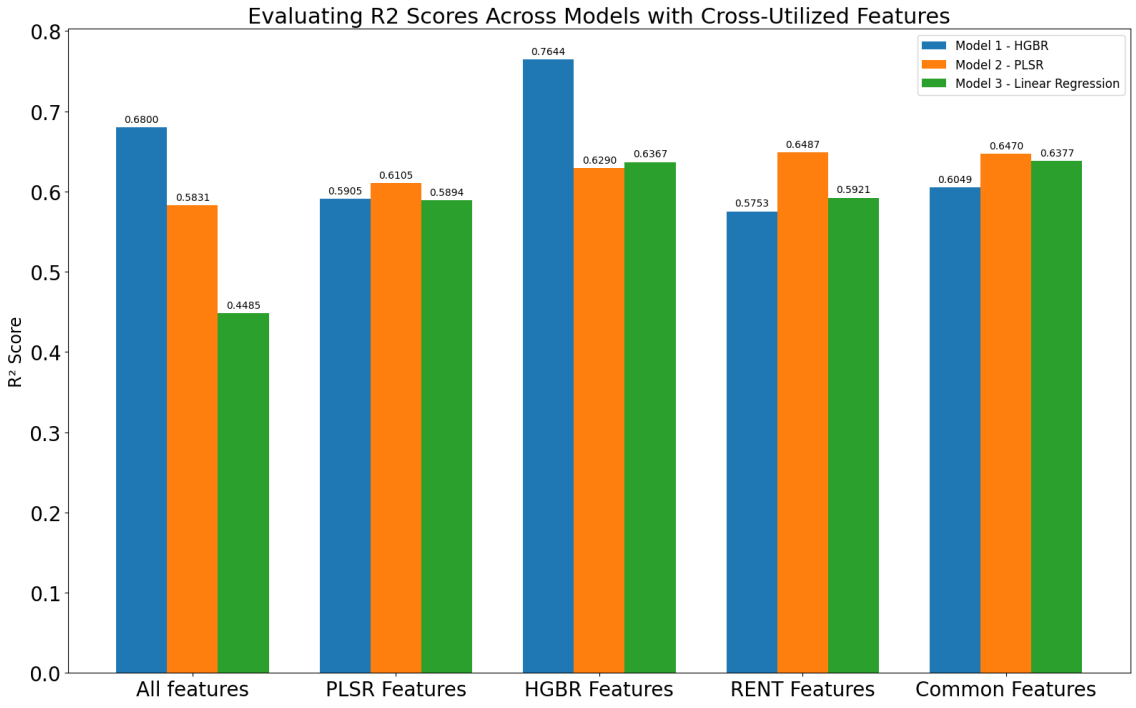


Figure 33: R^2 comparison across models using their own and each other's optimized feature subsets, highlighting the impact of feature selection on model performance.

4.3 Development of Optimization model

In this section, we will develop an optimization model by utilizing the information gathered from previous chapters. We will use the strengths of the HGBR and PLSR models to create optimization models using the Differential Evolution algorithm, as they have shown the best and moderately good performance scores, respectively. Once we have the optimization models, we will select the one with the best score to integrate into the website. The first and crucial step in making the optimization model is identifying the controllable and uncontrollable variables. We examined the chosen variables from each ML model and consulted an expert to select the significant variables and predictive power. Identifying the variables that can be utilized in practice is also essential. While building the optimization model, we have considered some interactions between variables that are not visible on the website's user interface. Instead, they are calculated automatically to enhance the performance of the ML algorithm used in the optimization model. The selected controllable and uncontrollable variables are displayed in Table 14.

Table 14: Chosen parameters.

Uncontrollable Parameters	Controllable Parameters	Interaction Parameters
Bu_start2casomaticTime	Bs_Mengde	Bu_duration * Bu_tempmean
Yk_start2endTime	Yk_coagToCutAvgT	Yk_timecoag*Yk_coagToCutAvgT
Yk_mengde	Yk_ystevannToTommingAvgT	Yk_timeYstevann*Yk_ystevannToTommingAvgT
Bu_duration	Bu_tempmean	
Yk_timeCoag		
Yk_timeYstevann		

Several factors can affect the cheese production process, some of which are beyond the manufacturer's control. Two such factors are the time measurements, namely `Bu_start2casomaticTime` and `Yk_start2endTime`. `Bu_start2casomaticTime` represents the duration from the start of the cheese vat to the end of the buffer batch, while `Yk_start2endTime` represents the total time taken from start to finish. These variables are uncontrollable, and their values are vital for the optimization model. Although predicting these values in advance can be challenging, they must be known for the optimization model to work correctly. Another important variable is `Bs_mengde`, representing the culture added to the cheese vat in liters. This variable is controllable, and manufacturers can adjust this parameter during production. To optimize the cheese production process, the optimization algorithm DE needs a set of boundaries that specify the minimum and maximum values for each controllable parameter. A domain expert has determined these boundaries. The DE utilizes a nonlinear objective function to guide the controllable parameters toward an optimal solution for each sample in the test set. The samples are then predicted by the ML model and evaluated against a set target range (57.6-57.7), with penalties applied if the predictions fall outside this range. This process is iterative, where the controllable variables are updated based on feedback from the objective function. These optimal values for the controllable variables help bring the production process closer to the optimal dry matter range. However, due to the sensitive nature of this information, the boundaries and optimized values for the parameters are censored.

4.3.1 Optimization Model 1 - HGBR

The development of the optimization model began with the selection of variables outlined in the previous chapter. These variables were used to create a model using HGBR. Techniques such as the LOGO CV, similar to those described in Chapter 3.4.4, were applied to handle batch effects. The same performance metrics were used to evaluate this model. The results from the HGBR model are presented in Table 15. This model performs worse than the HGBR

model previously discussed, primarily because many features have been excluded. Some of the variables measured in the process are not practical to include in the model. This is because they are measured too late in the process.

	Training set	Cross-validation	Test set
MSE	0.0713	0.1377	0.1857
RMSE	0.2671	0.3711	0.4310
R^2	0.7438	0.5054	0.5327

Table 15: Model Performance for Histogram-based Gradient Boosting Regression with chosen parameters.

After creating the ML model, we will use it for the Differential Evolution (DE) algorithm. The DE algorithm will iterate through each sample in the test set, optimizing the controllable parameters to achieve the optimal dry matter content. The `maxiter`, `popsize`, `mutation`, and `recombination` parameters are set to 10, 50, (0.2, 1), and 0.9, respectively. This combination will expand the solution space and assist the optimization algorithm in finding optimal solutions.

Figure 34 displays the dry matter content before and after optimization for the test set. The blue data points represent the actual dry matter content, while the red represents the dry matter content after optimization. The green shaded region represents the optimal dry matter content target. The lines between the actual and optimized values are used for better visualization and connect the same samples together. Figure 35 displays the distribution of actual and optimized dry matter content for the optimization model using HGBR. Table 16 displays the custom score, which counts the number of cases in which the actual dry matter content has been enhanced by the model, relative to the total number of cases.

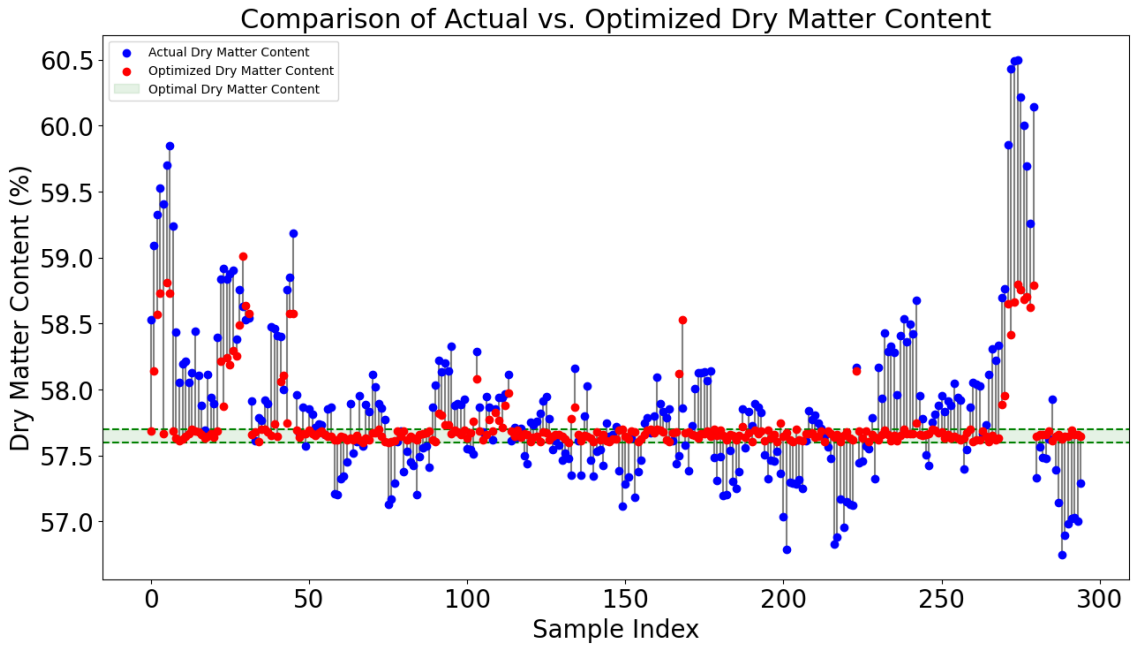


Figure 34: Optimized dry matter values using Differential Evoloution with Histogram-based Gradient Boosting Regression.

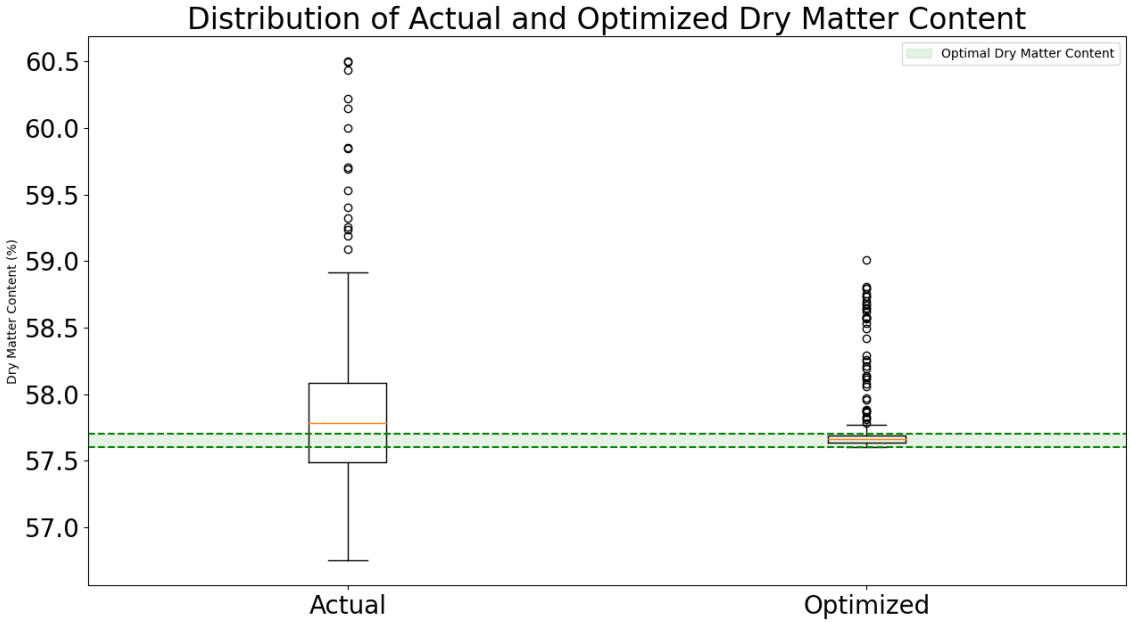


Figure 35: Distribution of Actual and Optimized Dry Matter Content for the optimization model using HGBR regression.

Metric	Value
Improvement Score	221 / 295
Accuracy Percentage	74.92%

Table 16: Evaluation of optimization model using HGBR

Figure 34 demonstrates that the optimization algorithm has successfully adjusted the dry matter content in many cases to achieve values within the optimal range of 57.6-57.7%. In some instances, it has brought the dry matter content closer to this optimal level. In Figure 35, the box plots illustrate that the variability has been reduced and the dry matter content is now closer to the optimal range. We observe that test samples with very high initial dry matter content struggle to reach within the optimal range, yet they still exhibit improvement. The Improvement score shows that in 221 out of 295 cases, the dry matter content reached the optimal range of 57.6-57.7%. This yields an accuracy score of 74.92%. Before optimization, there were 22 instances within the target range. Additionally, 82.37% of these observations continued to meet the optimal values after optimization. However, there are rare instances where some samples within the optimal range exhibited worse dry matter content after optimization. Although these cases are few, it is possible that the HGBR model predicts poorly in some instances. Figures 36, 37, 38, and 39 display the values for the controllable variables before and after optimization. Blue represents the original value, while red indicates the optimized value. The figures presented demonstrate adjusting the controllable parameters `Bu_tempmean`, `Yk_ystevannToTommingAvgT`, `Yk_coagToCutAvgT`, and `Bs_Mengde` to enhance the dry matter content.

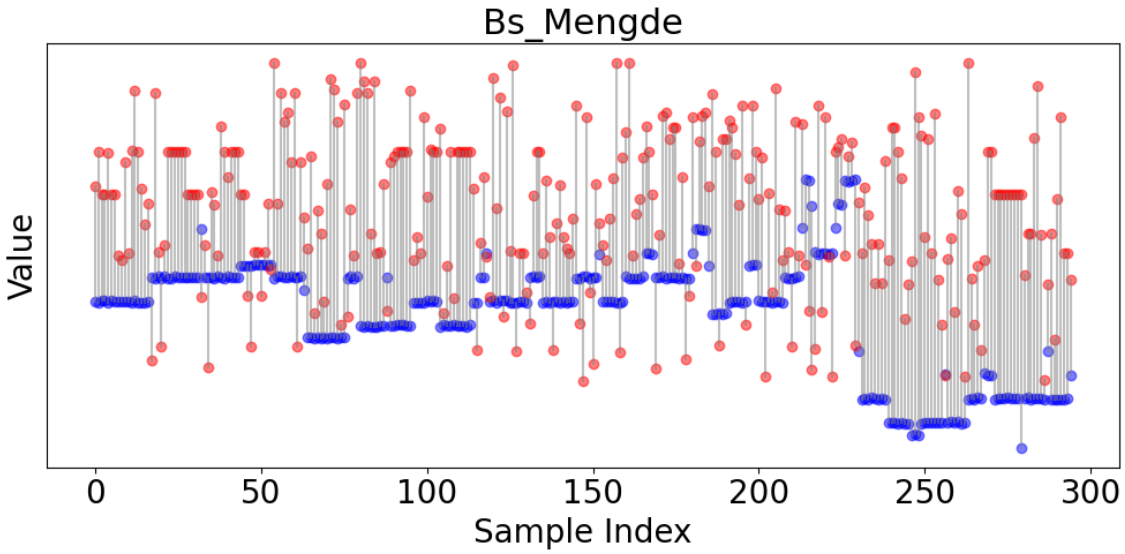


Figure 36: Optimized Bs_Mengde values.

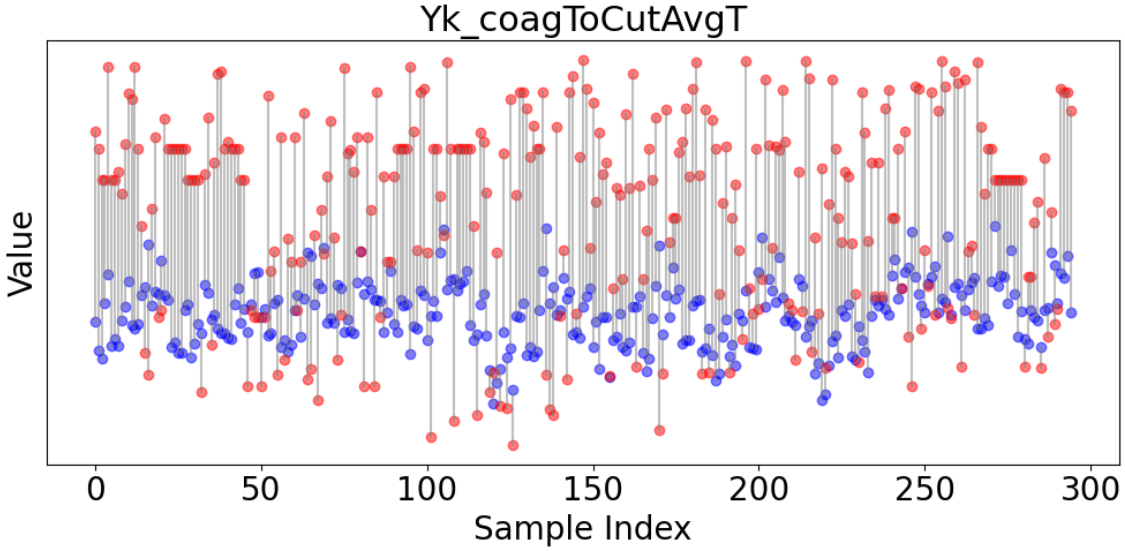


Figure 37: Optimized Yk_coagToCutAvgT values.

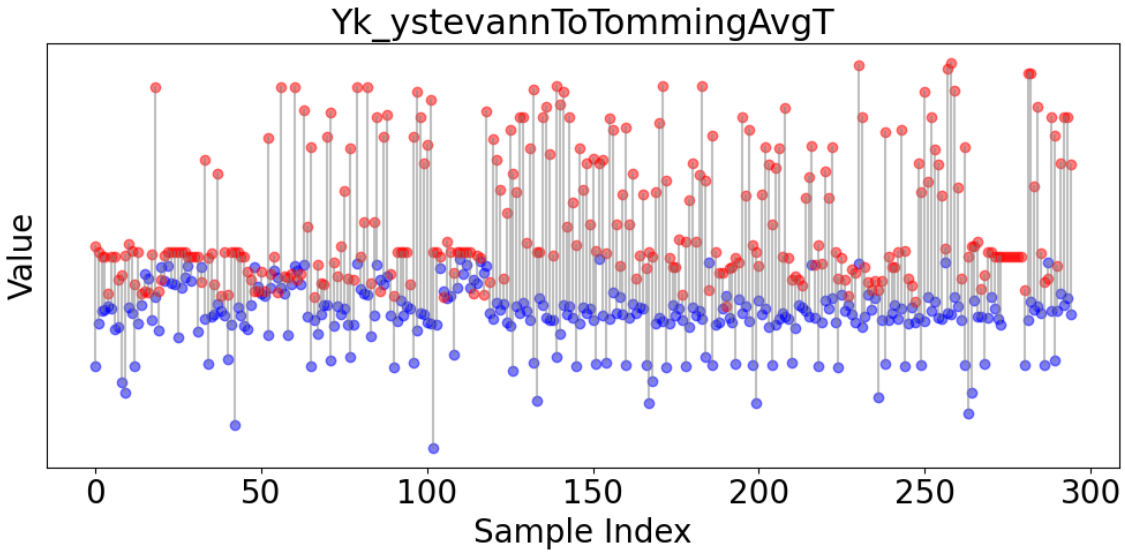


Figure 38: Optimized Yk_ystevannToTommingAvgT values.

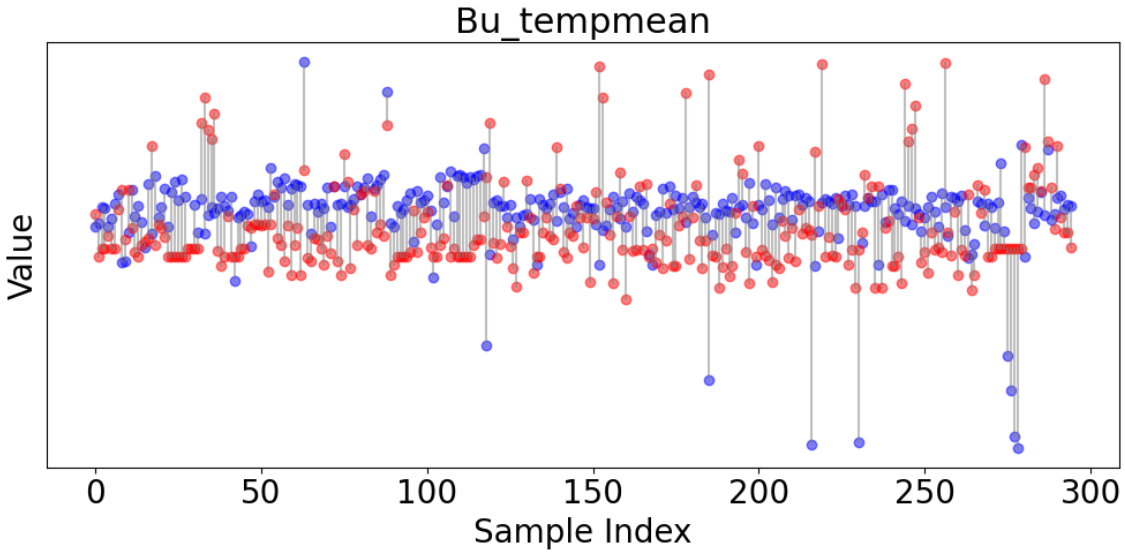


Figure 39: Optimized Bu_tempmean values.

4.3.2 Optimization Model 2 - PLSR

To develop the second optimization model, we started by selecting variables and creating a model using PLSR with three components. We used the same methods and techniques as the previous model. Table 17 below shows the performance of the PLSR model based on the chosen variables.

	Cross-validation	Training set	Test set
MSE	0.1414	0.1264	0.1887
RMSE	0.3760	0.3555	0.4344
R^2	0.4922	0.5461	0.5252

Table 17: Model Performance for PLSR with 3 Components.

This model uses the differential evolution algorithm and the same modeling techniques and parameters as the previous optimization model. The optimized dry matter content can be seen in Figure 40, both before and after the optimization process. Figure 41 provides a visual representation of the distribution of actual and optimized values through box plots. Table 18 contains the results of the evaluation of the optimization model.

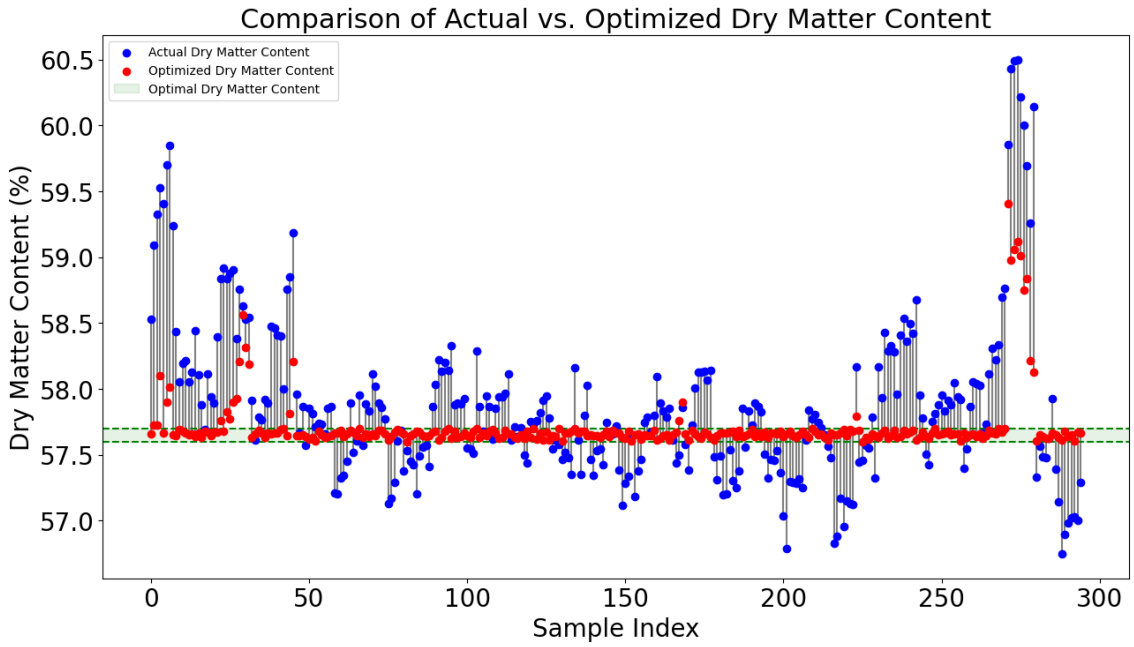


Figure 40: Optimized dry matter values using Differential Evoloution with PLSR.

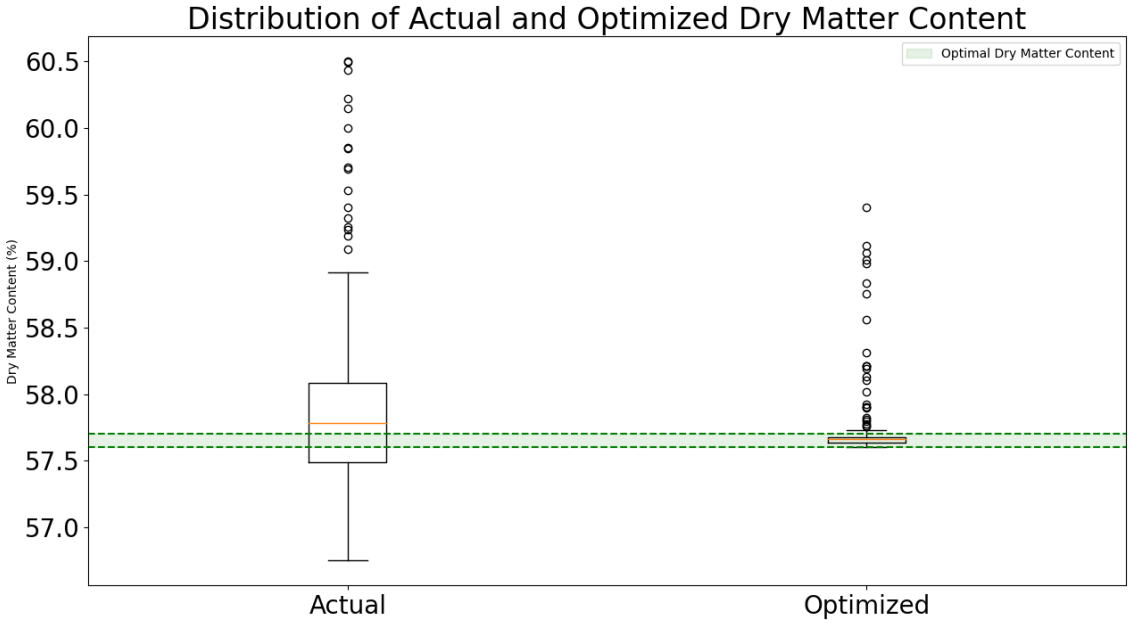


Figure 41: Distribution of Actual and Optimized Dry Matter Content for the optimization model using Histogram-based Gradient Boosting Regression.

Metric	Value
Improvement Score	245 / 295
Accuracy Percentage	83.05%

Table 18: Evaluation of optimization model using PLSR

Figures 40 and 41, along with Table 18, demonstrate that the optimization algorithm, which employs Differential Evolution (DE) with PLSR, has successfully adjusted the dry matter content in many instances to achieve a value within the optimal range. The distribution of the actual and optimized values, plotted through box plots, shows that the variability in the dry matter has been significantly reduced. This model surpasses the first in effectively reaching the target range of 57.6-57.7%. The improvement score indicates that the dry matter content has been enhanced and falls within the target range of 57.6-57.7% in 245 out of 295 cases, translating to a 90.51% accuracy of achieving the desired dry matter content range. Like the first optimization model, this model starts with 22 observations within the target range. After optimization, this number increases to 267 instances. Furthermore, 90.51% of the observations that were within the target range before optimization have remained within this range afterward. This marks an improvement over the first optimization model. Despite their small number, these cases warrant further investigation. Figures 42, 43, 44, and 45 displays the original values of the controllable parameters and their optimized values.

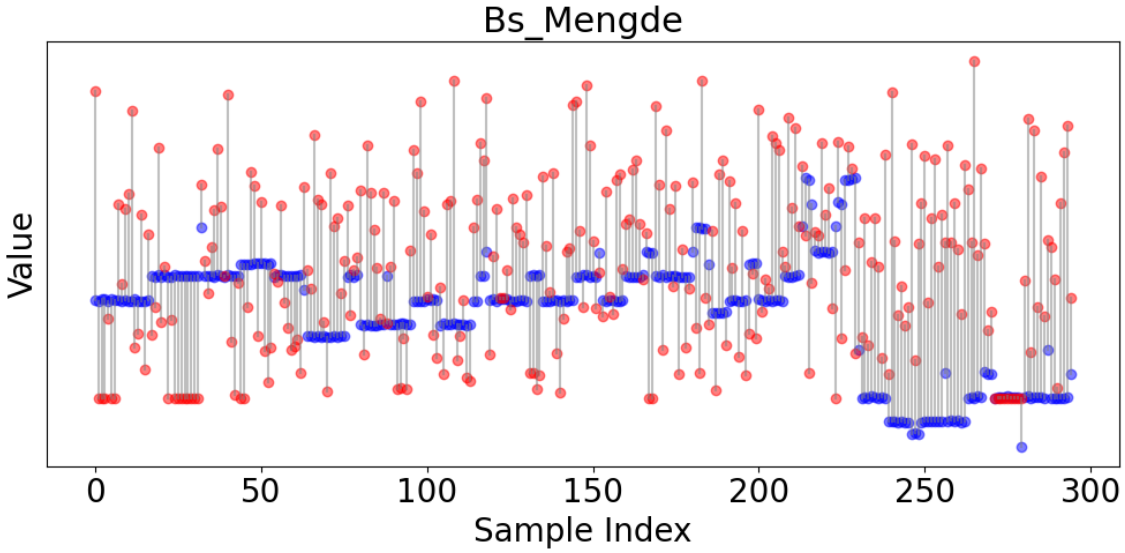


Figure 42: Optimized Bs_Mengde values.

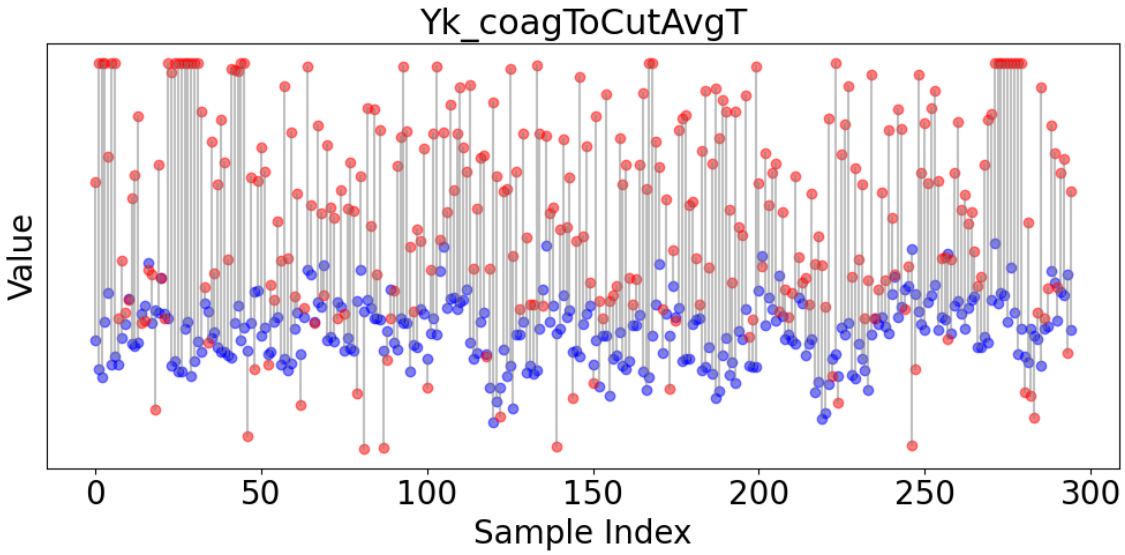


Figure 43: Optimized Yk_coagToCutAvgT values.

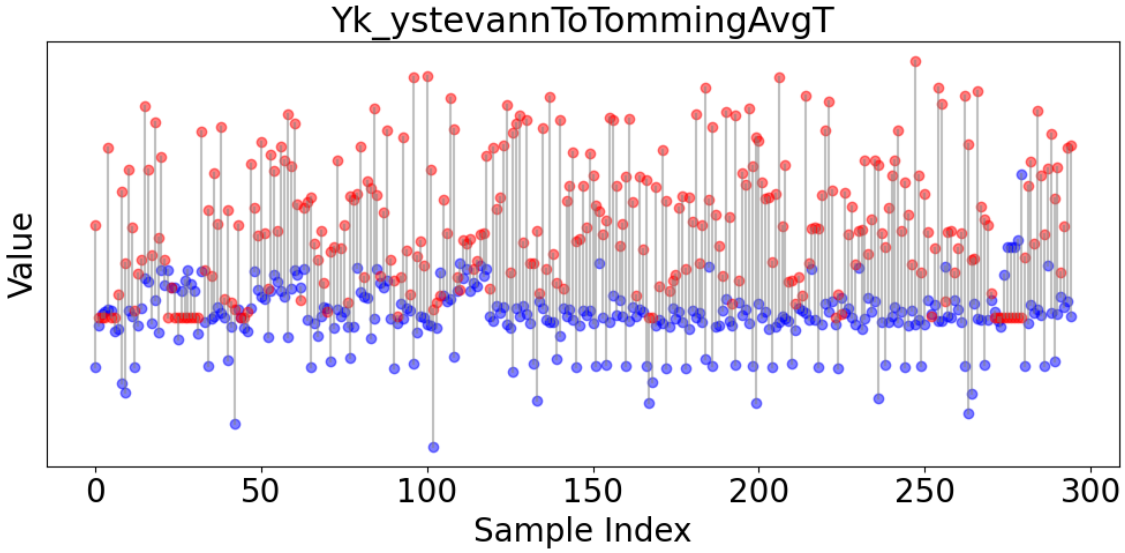


Figure 44: Optimized Yk_ystevannToTommingAvgT values.

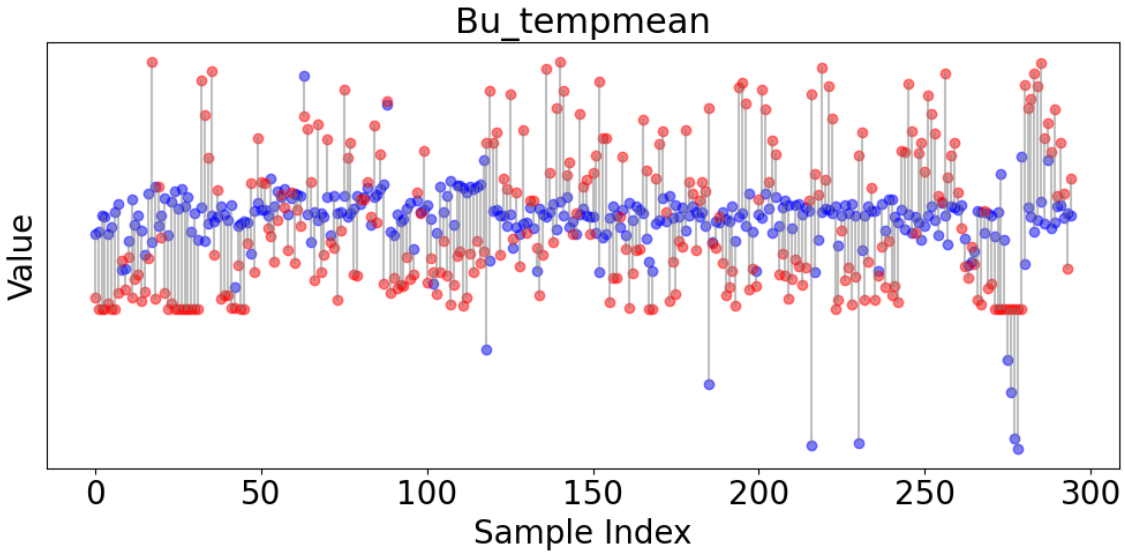


Figure 45: Optimized Bu_tempmean values.

4.4 Streamlit-website

In this section, we will utilize an optimization model integrated into a user-friendly website. We will employ PLSR as a predictive component in the optimization model because of its good performance. Within the scope of this research, we will use Streamlit to create a simple website to demonstrate the intention and functionality of the model. Further research will focus on developing a permanent website using real-time measurements with a well-designed interface integrated with the TINE value chain. This is beyond the scope of our current research. The website is intended for Norvegia cheese production and can be accessed daily. Production workers can use a computer to find parameter values to enhance the dry matter content. The process of using this website starts with finding all the values for the uncontrollable variables. As shown in Table 14, six uncontrollable parameters must be known in advance. After entering the uncontrollable values, you can click the "Optimize" button. The website will then print out the optimized controllable variables to enhance the dry matter content. Figure 46 will show us a screenshot of the website.

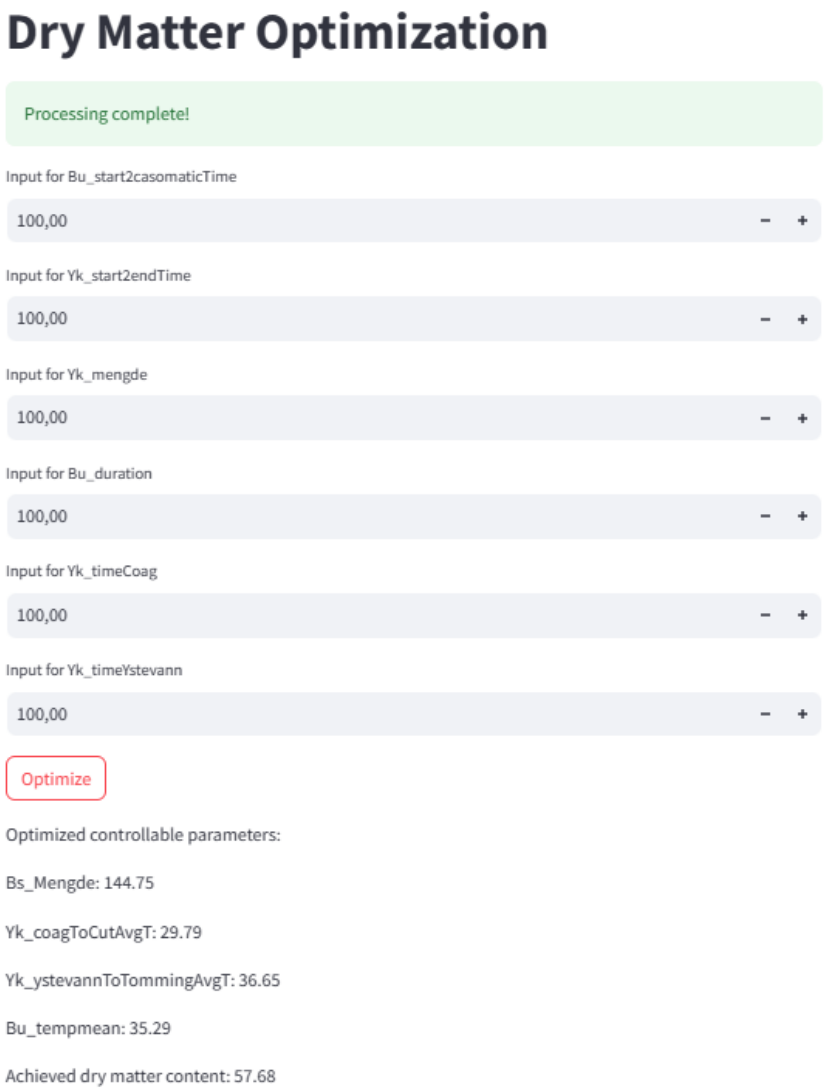


Figure 46: Screenshot of the website created with Streamlit.

Chapter 5

Discussion

5.1 Reflections

When I began working on this project, I had limited knowledge about the subject area, initially presenting some challenges. However, I viewed this as an opportunity to learn and grow. Although I had a basic understanding of cheese production, I was determined to deepen my knowledge and quickly adapt to the research demands. Throughout the project, I sought expert advice to help me select the right variables and understand their critical relationships. This guidance was crucial in improving my understanding and ensuring the reliability of our analytical processes. With each step, I refined our data analysis and developed my expertise in the field.

Our research employed various effective methods and models, including the HGBR algorithm, which produced robust results due to its ability to handle non-linear patterns in the dataset. While the HGBR algorithm was optimal for this particular dataset, we recognize the potential benefits of exploring additional non-linear machine learning algorithms. Such exploration could offer slight improvements in our results and interpretations, providing a broader comparative analysis that could refine the accuracy and depth of our conclusions. In addition to HGBR, the PLSR proved considerably effective, particularly in optimization tasks where its more straightforward, linear approach provided more stable interactions with our optimization algorithms. However, it is important to note that while exploring alternative approaches and advanced techniques could yield more insightful findings, the improvements over our current results might be minimal. We have already covered a relevant range of predictive methods, tested combinations of variable selection, and applied domain knowledge effectively in choosing variables. This comprehensive approach ensured a balanced and grounded analysis, reflecting a well-considered application of available methodologies. While feature selection was a central focus of our study, we did not fully explore feature engineering techniques. Considering interactions in all models could have led to a more comprehensive understanding of the underlying relationships and patterns within the data, emphasizing the need for deeper exploration. However, incorporating interactions could also result in overly complex models, so they were not considered for this research. Moreover, identifying potentially valuable variables not currently measured could uncover new insights and enhance the model's accuracy. This exploration would involve reviewing the process and domain-specific knowledge. Integrating these methods could have expedited the discovery of additional features that might have influenced the dataset, such as variables related to milk composition. These could include factors like the dairy animals' diet and environmental conditions, such as seasonal variations. It's also worth noting that outlier detection methods were not used. Incorporating advanced outlier detection techniques could potentially improve the performance of our models and lead to more precise identification and handling of outliers, resulting in more reliable and robust analyses

5.2 The use of CRISP-DM

Throughout this project, we followed the CRISP-DM process to gain a thorough understanding of our data from multiple perspectives. The advantage of this iterative process is that it provides a structured way to organize our work. During this project, the experience has underscored the importance of the CRISP-DM framework in enhancing the effectiveness of our efforts. The business aspect helped us understand the economic factors that made it necessary to control dry matter content. Understanding the problem and the data available to use the CRISP-DM model effectively is essential. Without this knowledge, the model won't provide a solution. The model has been around for a while and can be modified to work better for modern projects. Additionally, it's essential to remember that the model needs to be tailored to the specific project at hand and that it's continuously evolving. We also spent time examining the technical aspects of the dataset during the data understanding and preparation stages. We developed and assessed several models to optimize dry matter content effectively in the modeling and evaluation stages. During the CRISP-DM cycle, we ensured that the evaluation and optimization models aligned with our success criteria. One cycle could consist of using different missing values techniques, split techniques, and other changes that affected the evaluation. Based on our data analysis, we adjusted our criteria regularly to ensure they were realistic and achievable. The initial criteria were overly optimistic and could not be achieved due to factors like batch effect. Ultimately, we established and attained suitable, realistic success criteria. All the machine learning and optimization models have met the success criteria. As explained in Chapter 2.6, the method contains six phases. Almost all the phases were used in this project, though the deployment phase was not fully implemented. In our case, a simple website using Streamlit was developed. In practical applications, advanced technology is required to integrate it into the industry, allowing all workers and administrators to access it. Deriving data and analysis from this website is also crucial for its improvement, especially in the early stages. This is beyond the scope of our research and, therefore, will not be fully implemented in this thesis.

5.3 Feature Selection Process

We used permutation importance and backward selection in our study on feature selection for the HGBR model. It's important to note that the backward feature selection assumes that the initial model includes all relevant features. However, this may not always be true based on the permutation importance. Permutation importance can be biased by unrealistic data instances [58]. By having correlated features, it can decrease the importance of the associated feature. As mentioned in data understanding Chapter 3.4.2 we have many correlated features. This implies that certain correlated variables could strongly influence the model yet appear mediocre in their permutation importance results. By using only permutation importance, the interpretation of feature importance becomes considerably more difficult [58]. Moreover, the sequence in which features are eliminated in the backward feature selection can significantly influence the composition of the final feature set, raising questions about the optimality of the selected features. Although the feature set yields a good performance score, there remains room for improvement. The uncertainty about the optimality of this set underscores the need for further investigation. The combination of the techniques was initially chosen to tackle the multicollinearity problem by using permutation importance and a backward feature selection to improve the model's ability to predict accurately. Nonetheless, alternative feature selection methods may better handle multicollinearity. Exploring other feature selection approaches for the HGBR model could provide deeper insights into feature interdependencies, refining our feature selection process for enhanced precision.

RENT is a method that incorporates built-in functions, which presented some challenges throughout the modeling process. Modifying these inherent functionalities was particularly challenging.

For the feature selection process, the cutoff values were chosen to select optimal features. The cutoff values t_1, t_2, and t_3 in RENT serve as critical hyperparameters that influence the number of features selected. Lowering t_1 may allow for the selection of more features, thus potentially capturing more complex patterns in the data, but also increases the risk of including noisy, less stable features. This could enhance the model's performance on the training set but may not generalize well to unseen data, leading to overfitting. Conversely, a higher t_1 threshold prioritizes stability by selecting only the most consistently deemed important features. t_2 and t_3 add further refinement to the selection process, controlling the stability of feature weights [48]. The binary classification function in RENT includes a grid search to identify optimal cutoff values. However, this function is not available for regression problems. Therefore, this was manually experimented with various combinations and identified some values for the criteria that produced satisfactory results. If the grid search functionality had been available for regression problems, other variables might have been selected, leading to better performance. Therefore, these cutoff values require careful consideration. Further exploring this balance could offer deeper insights into the model's behavior under different stability constraints. Another aspect to discuss is the selection of features for the validation study. In the validation study, we found that the p-value was crucial in determining whether the RENT method was more effective than random selection in feature selection. If the p-value was significant, it indicated that RENT was better than random selection. However, it's important to note that the RENT method used in our research has selected many variables. Therefore, the p-value's significance should depend on using a limited number of features. When many features are included, random selection can have a higher chance of selecting the most effective features by chance. As a result, the advantage of using RENT may decrease in scenarios with many features to choose from.

We used VIP scores to assess the significance of each variable in the PLSR. These scores are computed based on the contribution of each variable to the model's predictive ability. However, the PLSR model we used in our analysis had limited predictive accuracy, as indicated by a moderate R^2 score. This suggests that the model might not have accurately captured the intricate relationships within the data. As a result, the VIP scores obtained from this model may not accurately reflect the true importance of the variables. Therefore, the VIP scores must be interpreted carefully, particularly when the underlying model's performance is suboptimal. In our research, the threshold for determining important variables, typically considered a VIP score greater than 1, can be considered a tuning parameter. In our case, a lower threshold of 0.8 was adopted in our approach. This adjustment resulted in a better performance on the test set, suggesting that a broader range of variables might be necessary for better predictions. This threshold was tested manually, but in the future, we should determine the optimal threshold more systematically, which could potentially yield better predictions.

5.4 Overfitting

The HGBR model utilized Optuna for hyperparameter selection. Although Optuna significantly saved time, ensemble models that use decision trees for their predictions tend to overfit the training data. Overfitting may occur due to overly complex models that capture noise and irrelevant patterns. To prevent excessive complexity, the parameter values were restricted to very small numbers using Optuna. The `max_depth`, `n_estimators`, and `min_samples_leaf` are the main tuning parameters that can lead to complexity if their values are set too high. Therefore, regularization terms were not included because of the computation costs and the complexity of the hyperparameter search. It is worth noting that including more tuning parameters might have given us different results.

5.5 Discussion of Optimization Model

The optimization model aims to increase the dry matter content by using uncontrollable variables known in advance to obtain optimal controllable variable values. However, this assumption may make the model more challenging for practical application since most uncontrollable values are time measurements, which are difficult to predict. In Chapter 3.1, we learned that we could use real-time measurements to measure the uncontrollable parameters. Four uncontrollable parameters can be measured before the controllable parameters are adjusted, and two uncontrollable parameters come at the end of the process, one of which is `Bu_start2casomaticTime`. This variable is highly significant, as observed in all models. However, by analyzing historical data, we could provide estimates for the uncontrollable values. Identifying the variables influencing these variables could allow for partial control, better estimating the variables used in the optimization model, and leading to more robust results. Moreover, the optimization model can account for uncertainty by incorporating uncertainty analysis. Integrating probabilistic methods, such as stochastic programming techniques, can help the model to consider uncertainty in input parameters or constraints and optimize decisions under uncertainty to achieve robust solutions. We learned in Chapter 2.5 that linear and non-linear programming can handle constraints effectively. For instance, we can optimize variables such as dry matter content while considering additional data such as manufacturing costs or energy consumption. By incorporating these factors into the optimization framework, we can minimize their impact while achieving optimal dry matter content. However, the lack of relevant data limits further exploration of such possibilities. This indicates the need for more research in this area.

There is another aspect of the optimization model that merits discussion, particularly concerning the ML models used: PLSR and HGBR. These models are crucial for predicting the optimized dry matter content in the optimization mdel. However, the performance of these models, based on the selected parameters, was not entirely satisfactory. Their average performance suggests that some predictions may be inaccurate. In rare cases, it was observed that the dry matter content actually deteriorated compared to the initial measurements. Therefore, it is important to acknowledge that the success of the optimization model heavily depends on the robustness of these predictive models. We must focus on identifying more influential variables, engaging in feature engineering, and exploring similar optimization techniques to enhance their accuracy.

5.6 Future Work

Even though this work has provided valuable insights into improving Norvegia cheese production, some challenges still need to be addressed and further explored. Future work will involve identifying more influential variables for prediction to enhance the ML and optimization models. This can improve the performance and the feature selection process. Additional data can also be collected and used for improved modeling. Further investigation into batch effects and how they can be properly handled should also be pursued. As we mentioned earlier, the data we collected was obtained sequentially. This means that time series analysis could be used to better understand the data. However, this approach requires a significant amount of data to be collected, which we did not have in our research. Nonetheless, this approach could be utilized in future research. A permanent website, integrated with the optimization model, must be developed to provide better visualization for the industry. We have assumed that we know the uncontrollable variables, which is crucial for the optimization to function effectively. In the previous chapter, we discussed some techniques that could be used to address this issue. These must be further investigated to determine their usability or to find other methods for identifying the uncontrollable variables.

Chapter 6

Conclusion

This project has utilized the dataset provided by Tine Jæren and employed various machine learning algorithms and feature selection techniques to develop a predictive model to understand the relationships between the dependent and independent variables. These insights have created an optimization model to enhance the dry matter content. This model was integrated into a user-friendly, easy-to-use website to demonstrate how the optimization model could be deployed in the industry. The motivation behind this project is strongly driven by the use of new technology to enhance and improve the quality of products in the food industry. The problem statement and the research questions were introduced in Chapter 1.3.

The first research question we address is: How can machine learning models be used to predict the dry matter content? We have discovered that predictive models can be developed by understanding the relationships and mapping between $\mathbf{X}$ (independent variables) and $\mathbf{y}$ (dependent variable). The process begins with exploring and preprocessing the data, followed by exploratory data analysis. During the modeling phase, it is necessary to identify suitable machine learning algorithms and feature selection techniques, which vary depending on the chosen algorithm. Our study utilized Permutation Feature, Sequential Feature Selector, and RENT with HGBR, PLSR, and LR, respectively. By training these algorithms, they learn the relationship between $\mathbf{X}$ and $\mathbf{y}$, enabling them to predict the response variable.

The second research question we address is: How can we use insights from the predictive models to create an optimization model that improves the quality of Norvegia Cheese? By analyzing the relationships between $\mathbf{X}$ (independent variables) and $\mathbf{y}$ (dependent variable) using tools such as ML, we can identify the variables that contribute to variance within the dataset and those that are influential for predictive power. After selecting two models that perform best on test data and CV, these models are employed in the optimization model. The optimization process begins by identifying influential variables that can be practically adjusted. By categorizing these variables into controllable and uncontrollable groups, we use non-linear programming to adjust the controllable variables based on the uncontrollable variables, aiming to enhance and achieve an optimal dry matter content between 57.6%-57.7%.

The third and final research question is: How can we utilize the results as digital decision support to improve the industry's production process? Integrating the optimization model into a user-friendly website is not just a technological feat but a practical solution that enhances the quality control of Norvegia Cheese's dry matter. By examining the optimized values of the controllable parameters, we gain insights into how adjustments to these variables affect the dry matter content. This analysis enables us to perform various scenario analyses, providing actionable intelligence that can lead to more informed decision-making and potentially transformative adjustments in the production process.

The research questions have guided our exploration of how and why controlling the dry matter content is crucial and its significant impact on production. These questions have also helped us answer the main problem statement, which is: **How can machine learning and optimization techniques be applied to improve the quality and production of Norvegia cheese?**. In conclusion, this thesis not only enhances our understanding of the critical factors influencing Norvegia cheese production but also paves the way for future advancements in the field by integrating machine learning and optimization techniques. By addressing the key questions that guided this research, we have laid the groundwork for more efficient and effective production processes that promise to elevate the quality and sustainability of cheese manufacturing.

Appendix A

Appendix A: Overview of Variables

Name	Description
batchid	Batch ID
batch	Batch number
artikkel	Article number
artikkelBeskrivelse	Description of the article
start	Start of ystekar batch
stopp	End of ystekar batch
Si_batchid	Batch ID for silo
Si_start	Start time for silo batch
Si_utstyr	Equipment used for silo (Silotank)
Si_wilab_f_ftir	Fat content (%) in silo milk, measured in the lab
Si_wilab_prot_ftir	Protein content (%) in silo milk, measured in the lab
Si_wilab_temp	Temperature in silo (manually registered)
Si_alder_beregnet	Calculated age in hours from start of silo batch to start of cheese batch
Pa_start	Start time for pasteurization
Pa_utstyr	Pasteurization line used
Pa_inlinefat	Fat content (%) measured with inline NIR
Pa_inlineprot	Protein content (%) measured with inline NIR
Pa_inlinetorrstoff	Dry matter content (%) measured with inline NIR
Bs_Batch	Batch ID for culture used in cheesemaking
Bs_artikkel	Culture article
Bs_Mengde	Amount of culture added to the cheese vat in liters
Bs_alder	Hours from culture production completion to its use in the cheese vat
Bs_produsertMengde	Amount of culture produced in this batch
Bs_start	Start time for culture production
Bs_stopp	End time for culture production
Bs_wilab_akt_stm	Culture activity measured in standard milk
Bs_wilab_akt_ym	Culture activity measured in today's cheese milk
Bs_wilab_ph	pH of the culture
Bs_ph_pHo	pH at start from the culture production pH curve
Bs_ph_tmx	Time from start to maximum pH (days)
Bs_ph_pHmx	Maximum pH recorded
Bs_ph_t1	Time from start to when pH curve begins to fall (days)
Bs_ph_pH1	pH when curve begins to fall

Continued from previous page

Name	Description
Bs_ph_t2	Time from start to pH curve end of fall (days)
Bs_ph_pH2	pH when curve ends falling
Bs_ph_mx_dpH	Rate of pH fall
Bs_ph_pH_stage120	pH at stage 120
Bs_ph_pH_stage130	pH after cooling at stage 130
Bs_temp_Temp0	Temperature at start from the culture production temperature curve
Bs_temp_temp_incubation	Temperature during incubation/stage 91
Bs_temp_end_incubation	Temperature at end of middle level (incubation)
Bs_temp_start_incubation	Temperature at start of middle level (incubation)
Bs_temp_temp_cooled	Temperature after cooling
Bs_temp_start_cooled	Temperature at start of cooled period
Bs_temp_end_cooled	Temperature at end of cooled period
Bs_temp_Time_stage26	Time from start to stage 26 (heating to max completed)
Bs_temp_Temp_stage26	Temperature at stage 26 (heating to max completed)
Bs_temp_Time_stage91	Time from start to stage 91
Bs_temp_Temp_stage91	Temperature at stage 91
Bs_temp_Time_stage130	Time from start to stage 130
Bs_temp_Temp_stage130	Temperature at stage 130
Yk_utstyr	Cheesemaking vat
Yk_mengde	Amount of milk in cheesemaking vat in liters
Yk_mysemengde_r6001	Whey amount, R6001
Yk_wilab_ph_my1	pH of whey, measured at MY1
Yk_start2lopeTime	Minutes from start to rennet addition (step 11)
Yk_start2coagTime	Minutes from start to coagulation (step 13)
Yk_start2cutTime	Minutes from start to cutting (step 15)
Yk_start2myseavtappTime	Minutes from start to whey removal (step 38)
Yk_start2ystevannTime	Minutes from start to cheese water addition (step 41)
Yk_start2tommingTime	Minutes from start to beginning of emptying (step 57)
Yk_start2endTime	Minutes from start to end (step 58)
Yk_timeCoag	Minutes during coagulation step
Yk_timeCut	Minutes during cutting
Yk_timeMy1	Minutes during first whey removal
Yk_timeYstevann	Minutes during post-heating
Yk_timeTomming	Minutes during post-heating
Yk_brukssyreToLopeAvgT	Average temperature
Yk_brukssyreToLopeStdT	Standard deviation of temperature
Yk_brukssyreToLopeSumT	Area under time-temperature curve per second
Yk_lopeToCoagAvgT	Average temperature
Yk_lopeToCoagStdT	Standard deviation of temperature
Yk_lopeToCoagSumT	Area under time-temperature curve per second
Yk_coagToCutAvgT	Average temperature
Yk_coagToCutStdT	Standard deviation of temperature
Yk_coagToCutSumT	Area under time-temperature curve per second
Yk_cutToMy1AvgT	Average temperature
Yk_cutToMy1StdT	Standard deviation of temperature
Yk_cutToMy1SumT	Area under time-temperature curve per second
Yk_my1ToYstevannAvgT	Average temperature
Yk_my1ToYstevannStdT	Standard deviation of temperature

Continued from previous page

Name	Description
Yk_my1ToYstevannSumT	Area under time-temperature curve per second
Yk_ystevannToTommingAvgT	Average temperature
Yk_ystevannToTommingStdT	Standard deviation of temperature
Yk_ystevannToTommingSumT	Area under time-temperature curve per second
Yk_tommingToendAvgT	Average temperature
Yk_tommingToendStdT	Standard deviation of temperature
Yk_tommingToendSumT	Area under time-temperature curve per second
Yk_ystevanntempMean	Average temperature of cheese water when added to the cheesemaking vat
Yk_ystevanntempStd	Standard deviation of temperature of cheese water when added to the cheesemaking vat
Yk_phInlineMy1	pH of first whey removal measured with inline sensor
Bu_utstyr	Buffer tank
Bu_duration	Minutes from "start" to "stop" of buffer batch
Bu_level	Fill level in the buffer tank
Bu_filltime	Minutes from start of filling to full tank
Bu_emptytime	Minutes from start of emptying to empty tank
Bu_tempmin	Minimum temperature in buffer tank
Bu_tempmax	Maximum temperature in buffer tank
Bu_tempmean	Average temperature in buffer tank
Bu_start2casomaticTime	Minutes from start of cheesemaking to "stop" of buffer batch
Bu_cut2casomaticTime	Minutes from cutting step in cheesemaking to "stop" of buffer batch
Bu_coag2casomaticTime	Minutes from coagulation step in cheesemaking to "stop" of buffer batch
Ost_inlineFett	Fat content (%) in fresh cheese, measured with inline NIR
Ost_inlineTS	Dry matter content (%) in fresh cheese, measured with inline NIR
Ost_wilab_ph_4t	pH in fresh cheese (manual measurement)
Ost_phReduction4T	Difference between pH in cheese milk and pH in fresh cheese

Bibliography

[1] Alexander L Fradkov. "Early history of machine learning". In: *IFAC-PapersOnLine* 53.2 (2020), pp. 1385–1390.

[2] Indrajeet Kumar et al. "Opportunities of artificial intelligence and machine learning in the food industry". In: *Journal of Food Quality* 2021 (2021), pp. 1–10.

[3] TINE. *Om TINE*. 2024. URL: https://www.tine.no/om-tine (visited on 04/16/2024).

[4] Medium. *How Important is Data in Machine Learning?* Accessed: 07.04.2024. 2021. URL: https://medium.com/nerd-for-tech/how-important-is-data-in-machine-learning-259d51e86435.

[5] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*. Vol. 72. Springer, 2015.

[6] Medium. *Iterative Imputation with Scikit-learn*. Accessed: 07.04.2024. 2021. URL: https://towardsdatascience.com/iterative-imputation-with-scikit-learn-8f3eb22b1a38.

[7] IBM. *What is machine learning (ML)?* Accessed: 04.04.2024. 2024. URL: https://www.ibm.com/topics/machine-learning.

[8] Arthur L Samuel. "Some studies in machine learning using the game of checkers". In: *IBM Journal of research and development* 3.3 (1959), pp. 210–229.

[9] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. "Supervised learning". In: *Machine learning techniques for multimedia: case studies on organization and retrieval*. Springer, 2008, pp. 21–49.

[10] Bing Liu and Bing Liu. "Supervised learning". In: *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data* (2011), pp. 63–132.

[11] Peter Dayan, Maneesh Sahani, and Grégoire Deback. "Unsupervised learning". In: *The MIT encyclopedia of the cognitive sciences* (1999), pp. 857–859.

[12] Medium. *The Relationship Between High Dimensionality and Overfitting*. Accessed: 26.04.2024. 2023. URL: https://vtiya.medium.com/the-relationship-between-high-dimensionality-and-overfitting-5bca0967b60f.

[13] IBM. *https://www.ibm.com/topics/multicollinearity*. Accessed: 26.04.2024. 2023. URL: https://www.ibm.com/topics/multicollinearity.

[14] Isaac Kofi Nti, Owusu Nyarko-Boateng, Justice Aning, et al. "Performance of machine learning algorithms with different K values in K-fold cross-validation". In: *International Journal of Information Technology and Computer Science* 13.6 (2021), pp. 61–71.

[15] Medium. *Cross-Validation Techniques*. Accessed: 06.04.2024. 2021. URL: https://medium.com/geekculture/cross-validation-techniques-33d389897878.

[16] Davide Chicco, Matthijs J Warrens, and Giuseppe Jurman. "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation". In: *Peerj computer science* 7 (2021), e623.

[17] Davide Chicco, Matthijs J Warrens, and Giuseppe Jurman. "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation". In: *Peerj computer science* 7 (2021), e623.

[18] Bernd Bischl et al. "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 13.2 (2023), e1484.

[19] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.

[20] Sanford Weisberg. *Applied linear regression*. Vol. 528. John Wiley & Sons, 2005.

[21] G.A.F. Seber and A.J. Lee. *Linear Regression Analysis*. Wiley Series in Probability and Statistics. Wiley, 2012. ISBN: 9781118274422. URL: https://books.google.no/books?id=X2Y6OkXl8ysC.

[22] Michael A. Poole and Patrick N. O'Farrell. "The Assumptions of the Linear Regression Model". In: *Transactions of the Institute of British Geographers* 52 (1971), pp. 145–158. ISSN: 00202754, 14755661. URL: http://www.jstor.org/stable/621706 (visited on 04/17/2024).

[23] Medium. *https://www.ibm.com/topics/multicollinearity*. Accessed: 26.04.2024. 2024. URL: https://medium.com/@fernando.dijkinga/explaining-l1-and-l2-regularization-in-machine-learning-2356ee91c8e3.

[24] Medium. *Elastic Net Regression detailed guide !* Accessed: 26.04.2024. 2023. URL: https://medium.com/@shruti.dhumne/elastic-net-regression-detailed-guide-99dce30b8e6e.

[25] Min Xu et al. "Decision tree regression for soft classification of remote sensing data". In: *Remote Sensing of Environment* 97.3 (2005), pp. 322–336.

[26] CollaborativeGeneticist. *Decision Tree Depth Example*. https://commons.wikimedia.org/wiki/File:Rando 2021.

[27] Lorraine Li. *Classification and Regression Analysis with Decision Trees*. Accessed: 04.04.2024. 2019. URL: https://towardsdatascience.com/https-medium-com-lorrli-classification-and-regression-analysis-with-decision-trees-c43cdbc58054.

[28] Andy Liaw, Matthew Wiener, et al. "Classification and regression by randomForest". In: *R news* 2.3 (2002), pp. 18–22.

[29] TseKiChun. *Random Forest Explain*. https://commons.wikimedia.org/wiki/File:Random$_f orest_e xplain.p$ 2021.

[30] Oliver Tomic. *Random Forest*. Accessed: 06.04.2024, Presented in school curriculum by [Oliver Tomic]. 2020.

[31] Medium. *About Random Forest Algorithms*. Accessed: 17.04.2024. 2023. URL: https://medium.com/@dishantkharkar9/about-random-forest-algorithms-62163357db25.

[32] Stefanos Fafalios, Pavlos Charonyktakis, and Ioannis Tsamardinos. "Gradient boosting trees". In: *Gnosis Data Analysis PC* (2020), pp. 1–3.

[33] Hossein Yavari. "Solution gas-oil ratio estimation using histogram gradient boosting regression, machine learning, and mathematical models: a comparative analysis". In: *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects* 46.1 (2024), pp. 379–396.

[34] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[35] Medium. *Principal Component Analysis (PCA) in Machine Learning*. Accessed: 04.04.2024. 2023. URL: https://medium.com/aimonks/principal-component-analysis-pca-in-machine-learning-407224cb4527.

[36] Hervé Abdi and Lynne J Williams. “Principal component analysis”. In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.

[37] Steven M Holland. “Principal components analysis (PCA)”. In: *Department of Geology, University of Georgia, Athens, GA* 30602 (2008), p. 2501.

[38] Dong Hyun Jeong et al. “Understanding principal component analysis using a visual analytics tool”. In: *Charlotte visualization center, UNC Charlotte* 19 (2009).

[39] Hervé Abdi and Lynne J Williams. “Principal component analysis”. In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.

[40] Medium. *How to read PCA biplots and scree plots*. Accessed: 16.04.2024. 2018. URL: https://bioturing.medium.com/how-to-read-pca-biplots-and-scree-plots-186246aae063.

[41] Saleh M Al-Alawi, Sabah A Abdul-Wahab, and Charles S Bakheit. “Combining principal component regression and artificial neural networks for more accurate predictions of ground-level ozone”. In: *Environmental Modelling & Software* 23.4 (2008), pp. 396–403.

[42] Hervé Abdi. “Partial least square regression (PLS regression)”. In: *Encyclopedia for research methods for the social sciences* 6.4 (2003), pp. 792–795.

[43] Kevin Dunn. “Process improvement using data”. In: *Retrieved on* (2019), pp. 11–2016.

[44] Taylor Jensen. *Feature Importance for Any Model using Permutation*. Accessed: 07.04.2024. 2022. URL: https://medium.com/@T_Jen/feature-importance-for-any-model-using-permutation-7997b7287aa.

[45] Md Al Mehedi Hasan et al. “Feature selection for intrusion detection using random forest”. In: *Journal of information security* 7.3 (2016), pp. 129–140.

[46] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python, SequentialFeatureSelector”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[47] Tahir Mehmood et al. “A review of variable selection methods in Partial Least Squares Regression”. In: *Chemometrics and Intelligent Laboratory Systems* 118 (2012), pp. 62–69. ISSN: 0169-7439. DOI: https://doi.org/10.1016/j.chemolab.2012.07.010. URL: https://www.sciencedirect.com/science/article/pii/S0169743912001542.

[48] Anna Jenul et al. “RENT: A Python Package for Repeated Elastic Net Feature Selection”. In: *Journal of Open Source Software* 6.63 (2021), p. 3323. DOI: 10.21105/joss.03323. URL: https://doi.org/10.21105/joss.03323.

[49] David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*. Vol. 2. Springer, 1984.

[50] David G Luenberger. “Linear and nonlinear”. In: (1984).

[51] David G. Luenberger. *Bruno Scalia C. F. Leite*. Accessed: 06.04.2024. 2022. URL: https://towardsdatascience.com/nonlinear-programming-theory-and-applications-cfe127b6060c.

[52] Everton Gomede. *Differential Evolution: An Evolutionary Optimization Algorithm*. Accessed: 06.04.2024. 2023. URL: https://medium.com/aimonks/differential-evolution-an-evolutionary-optimization-algorithm-de3bc166d481.

[53] Derviş Karaboğa and Selçuk Ökdem. "A simple and global optimization algorithm for engineering problems: differential evolution algorithm". In: *Turkish Journal of Electrical Engineering and Computer Sciences* 12.1 (2004), pp. 53–60.

[54] Ali Wagdy Mohamed and Hegazy Zaher Sabry. "Constrained optimization based on modified differential evolution algorithm". In: *Information Sciences* 194 (2012), pp. 171–208.

[55] Andrebis. *Graph of a paraboloid, its level sets and two line constraints*. Public domain. Useful for visualizing the Lagrange Multiplier method. 2010. URL: `https://commons.wikimedia.org/wiki/File:Name_of_the_file.png`.

[56] Christoph Schröer, Felix Kruse, and Jorge Marx Gómez. "A systematic literature review on applying CRISP-DM process model". In: *Procedia Computer Science* 181 (2021), pp. 526–534.

[57] Kenneth Jensen. *File:CRISP-DM Process Diagram.png*. https://commons.wikimedia.org/wiki/File:CRISP-DM_Process_Diagram.png. Licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license (`http://creativecommons.org/licenses/by-sa/3.0/`). 2012.

[58] Christoph Molnar. "A guide for making black box models explainable". In: *URL: https://christophm.github. io/interpretable-ml-book* 2.3 (2018), p. 10.

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

Postboks 5003
NO-1432 Ås
Norway